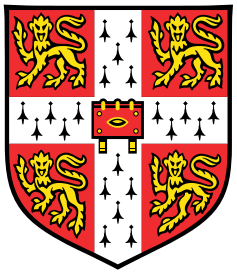
Generative Adversarial Networks for Speech Recognition Data Augmentation



Tianyu Wu

Department of Engineering
University of Cambridge

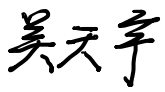
This dissertation is submitted for the degree of
Master of Philosophy

St John's College August 2018

I would like to dedicate this thesis to my loving parents

Declaration

I, Tianyu Wu of St John's college, being a candidate for the MPhil in Machine Learning, Speech and Language Technology, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose. This thesis contains 12450 words.

Signed 

Date 17/8/2018

Tianyu Wu
August 2018

Acknowledgements

I would like to thank Professor Phil Woodland for his patient guidance and constant support throughout the project. I would also like to thank Dr. Chao Zhang, who gives me a lot of valuable advice both in academic study and daily life. Chao also modified the HTK toolkit for this project, which makes the accomplishment of this work possible.

Abstract

Performance of a speech recogniser improves with increased training data size but collecting a large matched training dataset is difficult. This thesis aims to investigate the use of generative adversarial networks for acoustic data generation given an initial small training data set. We build the basic generation approach for phone units and context windows. Two data generation schemes are developed and compared, which are based on conditional GANs and unconditional GANs separately. We show that both schemes can generate high-quality data, which have the similar distribution as the real acoustic data. However, after re-training the acoustic model by the augmented data, no performance improvements are achieved. Based on the evaluation results, we believe that the generated data contain some unnatural variations, which hinders the acoustic model from learning a correct distribution. To address this problem, several plans are proposed for the future work.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Motivation

Deep neural networks (DNNs) based acoustic modelling has achieved a great success for automatic speech recognition (ASR) tasks in recent years, where a large amount of training data is required[1–3]. It has been shown that the performance of a speech recognition system is highly related to its training dataset size [4]. A larger training dataset tends to contain more pattern variations, which contributes to a robust parameter estimation and avoids overfitting problem. However, collecting a large matched acoustic data set could be very difficult. On the one hand, the scenarios in which an ASR system will be adopted may have some acoustic properties, such as background noise, microphone type and speaker's individual characteristics, which are very different from those of the training data. Such differences may lead to a significantly degraded performance for DNNs based ASR system [5]. This means that the acoustic data are best to be collected under different environments, which is especially difficult for some low resource languages. On the other hand, training deep neural networks as a classifier requires well-labelled data, while manually transcribing acoustic data is very expensive and sometimes involves privacy issues.

One approach to address this problem is data augmentation, which is a common strategy to increase data quantity in neural networks based pattern recognition tasks [6–9]. In speech recognition, data perturbation based augmentation scheme has been investigated a lot, where new data are produced by perturbing original data in a certain way. For example, the work in [10, 11] try to increase data quantity by adding simulated noise to the original clean speech data, which can minimise the gap between the training dataset and noisy speech. Other methods, such as vocal tract length perturbation (VTLP), elastic spectral distortion and speed perturbation[12, 9, 13, 14], are widely used for low resource speech recognition tasks. These

methods try to synthesise data by distorting the original speech spectra, which effectively increase the pattern variations for sparse data.

The advantage of the above data perturbation methods is that they can artificially produce labelled data without the need of collecting additional real examples, and they all achieved decent improvements in related datasets. However, these data perturbation methods also have several limitations, which are listed as follows. Firstly, these methods perturb data according to certain designed rules. Although these rules could potentially mimic certain speech distortions, the variations in the real world, such as speaker's age, gender and accent, can be more complicated [9]. Secondly, the quantity of data that they can generate is determined by the number of noise or distortion type that can be applied. In order to overcome these problems, a Generative Adversarial Networks (GANs) based augmentation method is proposed in this work, which has the potential to learn real pattern variations from data itself.

GAN is a powerful framework of training generative models through an adversarial process [15]. It can be used to learn a target distribution in a fully unsupervised fashion, which has achieved impressive results in the computer vision area [16, 17]. In speech area, this technique is mainly used in tasks, such as speech synthesis [18, 19], voice conversion [20, 21] as well as speech enhancement [22], while limited work has been done for speech recognition tasks. The work in this thesis aims to utilise GANs to generate high-quality acoustic data and then boost speech recogniser's performance with the augmented training data set.

1.2 Thesis contribution

In this thesis, the use of GANs for speech recognition data augmentation given an initial small training set is investigated. Firstly, we propose the methods for GAN based acoustic data generation, in which separate GANs are trained for each phonetic unit and the basic generation unit is the frame level acoustic feature.

Next, we have a thorough review of GANs and its variations. Based on these work, we design several robust GAN architectures, which are capable of generating acoustic data in a rather stable way.

Furthermore, two generation schemes for acoustic training set are developed. One is based on conditional GANs, which generates label preserved data directly. Another is based on unconditioned GANs, which is combined with a semi-supervised learning scheme to generate transcribed training sets. We show that both schemes can produce feature maps with the similar distribution as the real data.

The basic acoustic model used in this work is context dependent DNN-HMMs. The feature maps used for training the acoustic models are augmented by the simulated data generated by GANs. Several evaluation metrics are designed for measuring the system performance from different angles. Although the desired performance gains have not been observed, we carry out a detailed analysis on the experimental results and point out the potential problems in this project. The improvements which can be made in future work are also presented.

1.3 Thesis organization

The rest of this thesis is organised as follows. Chapter 2 briefly introduces the background knowledge for GANs, speech features and acoustic models. The related work on speech data augmentation is also reviewed in this chapter. Chapter 3 presents the methodology for GAN based speech data augmentation, and several preliminary experiments are included. The quality of the generated data, as well as the performance of the re-trained acoustic models, are evaluated in chapter 4, and chapter 5 gives the conclusion and future work.

Chapter 2

Background

2.1 Generative Adversarial Network (GAN)

2.1.1 Vanilla GAN

Generative Adversarial Network was first proposed by Goodfellow et al.in 2014 [15] as a promising framework of training generative models. This framework can be viewed as a game between two players:

- The first player is a generator, G, which takes a simple distribution (normally a standard multivariate Gaussian noise) as input and then transforms it to a distribution on the space of true samples, such as images.
- The second player is a discriminator, D, whose input is a sample either taken from the true distribution or synthesised by a generator and output is the probability that this sample is real.

In this game, both players aim to minimise their own loss until reaching a point at which neither player can improve itself unilaterally. Ultimately, a powerful generator can be obtained, which is capable of producing samples that the discriminator cannot classify them between real and fake.

More specifically, the training process for GANs is to solve a minimax problem with the following cross entropy:

$$\min_G \max_D V(G,D) = \mathbf{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbf{E}_{z\sim p_z(z)}[log(1-D(G(z)))] \tag{2.1}$$

where G denotes the generator, D denotes the discriminator, p_{data} is the true data distribution, $p_z(z)$ is the prior distribution on input noise z which is then mapped to the space of model

distribution p_g represented by $G(z)$ and $D(x)$ is the probability that x sampled from the true distribution. Moreover, both G and D are represented by multilayer perceptrons with parameter θ_g and θ_d respectively.

In practice, as suggested by [15], equation 2.1 is not an appropriate loss function for training G because it cannot provide sufficient gradient in the early stage, where $log(1 - D(G(z)))$ tend to saturate. Instead, the generator can be trained by minimising a non-saturating loss $-\mathbf{E}_{z\sim p_z(z)}[log(D(G(z)))]$, which still leads to the same solution but provide much stronger gradient at the beginning. In addition, the training should alternate between k steps of optimising discriminator D and one step of optimising generator G. Ideally, the global optimum of the system can be reached when $p_{data} = p_g$, which is proved as follows.

Theoretical results

According to equation 2.1, when G is fixed, the cross entropy loss for D given an arbitrary sample x can be expressed as:

$$Loss(D) = -p_{data}(x)\log D(x) - p_g(x)\log[1 - D(x)]. \tag{2.2}$$

By taking the derivative of equation 2.2 with respect to $D(x)$ and making it equal to zero, we can obtain the optimal discriminator $D^*(x)$:

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \tag{2.3}$$

Then we can substitute above result into equation 2.1 and the loss function for generator now becomes[1]:

$$\begin{aligned} Loss(G) &= \mathbf{E}_{x\sim p_{data}(x)}[\log D^*(x)] + \mathbf{E}_{z\sim p_z(z)}[log(1 - D^*(G(z)))] \\ &= \mathbf{E}_{x\sim p_{data}(x)}[\log D^*(x)] + \mathbf{E}_{x\sim p_g}[log(1 - D^*(x))] \\ &= \mathbf{E}_{x\sim p_{data}(x)}[\log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}] + \mathbf{E}_{x\sim p_g}[log(\frac{p_g(x)}{p_{data}(x) + p_g(x)})] \\ &= \mathbf{E}_{x\sim p_{data}(x)}[\log \frac{p_{data}(x)}{\frac{1}{2}(p_{data}(x) + p_g(x))}] + \mathbf{E}_{x\sim p_g}[log(\frac{p_g(x)}{\frac{1}{2}(p_{data}(x) + p_g(x))})] - \log 4 \end{aligned} \tag{2.4}$$

[1]This loss function is in the original minmax game form rather than the non-saturation form, which is only used for analysis purpose

Looking at the formula for Kullback–Leibler divergence (KL divergence) and Jensen-Shannon divergence (JS divergence)

$$KL(p_1||p_2) = \mathbf{E}_{x\sim p_1} \log \frac{p_1}{p_2} \tag{2.5}$$

$$JS(p_1||p_2) = \frac{1}{2}KL(p_1||\frac{p_1+p_2}{2}) + \frac{1}{2}KL(p_2||\frac{p_1+p_2}{2}) \tag{2.6}$$

we can find that equation 2.4 actually measure the JS divergence between the true distribution p_{data} and the model's distribution p_g:

$$Loss(G) = 2JSD(p_{data}||p_g) - \log 4 \tag{2.7}$$

Since Jensen-Shannon divergence between two distributions is always non-negative and equals zero only when two distributions are the same, we now prove that the global optimum can be achieved only if $p_{data} = p_g$. Note that when the support of the data distribution and the support of the model's distribution have no intersection, which is a common case in the early learning stage, the JS divergence between them is always equal to $\log 2$ [23]. If we then use the optimal discriminator in 2.3, the loss in equation 2.7 will always be zero. This reminds us that the discriminator should not be optimised too far ahead of the generator otherwise it cannot provide a usable gradient for training G.

To address the above problem, the work Wasserstein GAN (WGAN) [23] proposed to use the training criteria that minimising the Earth-Mover or Wasserstein distance between p_{data} and p_g. The corresponding loss functions are changed into $Loss(D) = -\mathbf{E}_{x\sim p_{data}(x)}[D^*(x)] + \mathbf{E}_{z\sim p_z(z)}[D^*(G(z))]$ and $Loss(G) = -\mathbf{E}_{z\sim p_z(z)}[D^*(G(z))]$ respectively. This method successfully avoids the above gradient vanishing problem but also has the drawback that it should enforce a Lipschitz constraint on discriminator. In further research [24], it shows that the non-saturating loss function proposed in the original paper [15] could be more stable in practice.

2.1.2 Spectral normalization

A major problem of GANs framework is that its training process is hard to be stabilised. As discussed in section 2.1.1, during training, the performance of the discriminator needs to be carefully controlled so that the generator model can be optimised properly. Normalizing the discriminator could be very useful to alleviate this problem and one successful approach is called spectral normalization (SN) [25]. When optimising discriminators, this work

suggests dividing each weight matrix by their spectral norm (max singular value), which can effectively control the Lipschitz constant of the discriminator. By restricting the space that the discriminator can be selected from, the training process of GANs will be more stable. The authors of SN also point out that this approach can lead to discriminators of higher rank and is much computational cheaper compared to other regularization methods, such as gradient penalty [26]. Furthermore, in Google's work [24], they prove that spectral normalization can improve GAN's performance consistently over different datasets.

2.1.3 Deep convolutional GAN

Apart from restricting the discriminator, the problem that GAN is unstable to train can also be alleviated by modifying the generator and discriminator architecture. The work in [17] combined the GAN idea with the deep convolutional networks, which is called deep convolutional GAN (DCGAN). Both the generator and discriminator networks now contain five convolutional layers. In addition, instead of using deterministic spatial pooling functions, such as max pooling, they proposed to use strided convolutions to downsampling or upsampling features. They proved that this structure configuration can effectively stabilise the training process for GAN across a range of datasets. In this work, we will utilise the idea of DCGAN combined with spectral normalization technique for data generation.

2.1.4 Conditional GAN (CGAN)

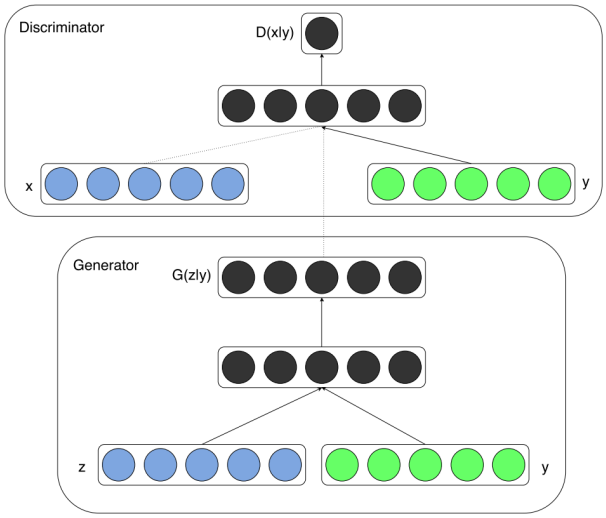


Fig. 2.1 A simple structure of conditional GAN

For the unconditional version GANs, it is impossible to control the mode of the data being generated because they are mapped from low dimensional random noise. The work conditional generative adversarial nets [27] introduced a method which can direct the data generation process. They proposed to condition both the discriminator and generator on additional information, where the conditions could be the class labels or some part of the data. This structure extends the application of GANs, which makes GAN useful for the tasks such as data augmentation, image transfer, etc. A simple structure of conditional GAN is illustrated in figure 2.1, where y is an embedding of the conditions, x is the input samples for the discriminator and z is the input noise for the generator. Similar to equation 2.1, the corresponding objective function for this networks is defined as:

$$\min_G \max_D V(G,D) = \mathbf{E}_{x\sim p_{data}(x)}[\log D(x|y)] + \mathbf{E}_{z\sim p_z(z)}[log(1-D(G(z|y)))] \tag{2.8}$$

This conditional version GAN is also heavily used in this work.

2.2 Speech recognition

2.2.1 Mel-scaled filter banks

Normally in ASR tasks, a speech signal is often represented by multiple fixed duration feature vectors, which describes the short-term speech spectrum. Each vector corresponds to a fixed length window of the speech (frame). The Mel-scaled log filter bank (FBANK) features, which are computed by doing filter-bank analysis, are often utilised for acoustic models based on deep neural networks [28]. Mel filter banks are filter banks where the consecutive triangular filters are used. They are equally spaced along the mel-scale in frequency domain, where mel-scale for a frequency f is defined as:

$$Mel(f) = 2595\log_{10}(1+\frac{f}{700}) \tag{2.9}$$

This scaling simulates the way that human ear works, which has better resolution at low frequencies and less at high frequencies. To implement it, the Fourier transformation needs to be applied to each frame of the speech and the corresponding power spectrum should be calculated. The filter banks can be calculated subsequently to capture the energy at each critical band.

2.2.2 Deep neural networks based acoustic modelling

In speech recognition, most systems are based on hidden Markov acoustic models (HMM). The conventional HMMs use Gaussian mixture models (GMM) to represent the state output probability $p(x|s)$, which is the likelihood of feature x from HMM emitting state s. The acoustic models using deep neural networks (DNN) to replace GMM has been proposed in recent years and shown much higher speech recognition accuracy [1, 2], which is called DNN-HMMs model. In this model, the posterior probability $p(s|x)$ produced by DNN is converted to a pseudo likelihood $p(x|s)$ based on the Bayes rule, which is defined as:

$$p(x|s) \propto \frac{p(s|x)}{p(s)} \tag{2.10}$$

where $p(s)$ is the training data prior and can be calculated from training samples. The standard HMM transition structure is retained and the Viterbi algorithm is used for decoding.

2.3 Related Work on Acoustic Data Augmentation

2.3.1 Vocal tract length perturbation (VTLP)

VTLP is a mature speech data augmentation scheme which generates new samples through perturbing the speech spectra of initial training audio. It was first proposed in work [12], in which a warping factor α is randomly selected from [0.9, 1.1] to warp the frequency axis for each utterance in training set. The vocal length tract of the speaker is thus changed, which leads to a new version of the original utterance with distorted speech spectra.

In the work [9], a modified version of VTLP was suggested in which the warping factor α of the speaker is first calculated and then a deterministic perturbation is applied by using the following factors to warp the frequency axis of original data:

$$\begin{aligned} \alpha \rightarrow [\alpha \pm \Delta, ..., \alpha \pm k\Delta, ..., \alpha \pm K\Delta] \\ k = 1, ..., K \end{aligned} \tag{2.11}$$

where, Δ is a fixed length shift along the α axis and 2K is the number of replicas of the original data. Both of the above VTLP schemes achieved performance improvements in many ASR tasks.

2.3.2 Speed perturbation

Speed perturbation technique, which was proposed in [14], is a speech data augmentation method which produces additional training set by changing the speed of the original audio signal. In [14], a time warped signal is produced by a factor α. The original audio signal $x(t)$ is then converted to $x(\alpha t)$. After applying Fourier transform, this is equivalent to produce shifts in the frequency domain. However, compared to VTLP, speed perturbation also changes the duration of the original signal, which affects the number of frames in the utterance. The work [14] shows that speed perturbation can yield better performance than traditional VTLP methods. In this thesis, this method is used as the baseline, where the speed factors 0.9 and 1.1 are used to produce two replicas of the original data.

Chapter 3

Methodology

3.1 GANs for speech data generation

As described in section 2.1, Generative Adversarial Networks are a powerful framework to estimate generative models, which are capable of reproducing a target data distribution. However, this framework cannot be adopted in speech directly. The issue is that speech recognition tasks involve variable length utterances and requires accurate variable length word labels while GANs usually generate fixed length output (e.g. images). To address this problem, we perform frame level speech data generation rather than utterance level. In addition, in order to match the input vectors of a DNN acoustic model, each output vector from a GAN generator is an extended frame obtained by stacking a window of neighbouring acoustic frames. Thus, the basic data unit generated by GANs is a fixed size feature map on speech spectrum, and the length of each frame is 10ms. In our experiments, a 40×16 dimension feature map is used as the data input for the GAN discriminator and the output for the GAN generator, where each frame is represented by a 40 dimension FBANK feature and expanded with 6 and 9 neighbouring frames in each side separately. The reason why setting the context width to 16 is that the proposed GAN has a deep convolutional structure, in which the strided convolutions (discriminator) and fractional-strided convolutions (generator)[1] are used to down sample and up sample the feature map by a factor of 2 [17]. Since three such layers will be used in both generator and discriminator, it is convenient to set each dimension of the feature map to an integer multiple of 8 (2^3). Note that the final feature map size used for acoustic model training is 40×13, in which the number of context frames in both sides are balanced to the same value (-6, +6). The last three frames of the generated data will be discarded in this stage.

[1] 'strided' means that the stride of convolution is 2

Furthermore, we propose to use separate GANs for each phone so that the phone labels of the generated data are known. The number of unique phones used in our experiments is 48, which means 48 GANs will be trained. This set of GANs is referred as 'GANs array' in following sections. However, in order to train an acoustic model by augmented data, the state labels of the generated samples should also be known. It is impossible to train separate GANs for each state especially under triphone system. One method to solve this problem is using conditional GANs. When training acoustic models for triphone system, we could construct decision trees to cluster the triphone states [29]. The tied triphone states tend to have similar characteristics, which means that they can be treated as one 'class' from the generation perspective. Thus, we can condition each GAN on the tied triphone states so that the state labels of the generated data can be controlled from the input. In this work, 808 tied triphone states are used, including three silence states. Considering each phone have a separate GAN, a mapping system is developed to map the true state label to the GAN label. Take phone 'aa' as an example, the corresponding mapping system is shown in figure 3.1. Note that all the tied states are represented in the form 'phone_HMM-state_label', and this is also used in chapter 4. This scheme has the advantages that each class can have sufficient training samples and the generated data can be used directly for later on acoustic model training. Another scheme is using the unconditional GANs to generate samples from random Gaussian noise directly. A semi-supervised learning strategy is implemented to deal with the unlabelled fake samples.

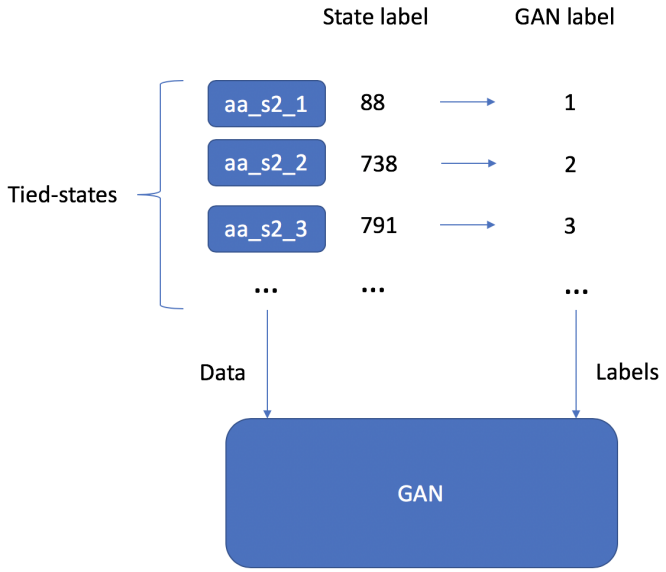


Fig. 3.1 Processing tied-triphone state labels belonging to phone 'aa'

3.2 GANs architectures

In this section, three GANs architectures are proposed, including one unconditional GAN and two conditional GANs. They are described in figure 3.2, 3.6 and 3.10 respectively. In these figures, **FC** stands for fully connected layers, **ConvTrans** stands for transposed convolutional layers and **Conv** stands for convolutional layers. Note that the parameters of each layer are indicated in the bracket behind them. For example, **FC(100, 5120)** means that the input and output size of the fully connected layer is 100 and 5120. **Conv(3,1,1)** means that the layer uses 3×3 convolution kernel with stride 1 and 1 zero-padding. The batch normalization layers and ReLU functions are applied in Generator networks, while the spectral normalization layers and leaky ReLU functions are applied in discriminator networks. The slopes of all leaky ReLU functions are set to 0.1. More details are described in the following sections.

3.2.1 Unconditional GANs

The structure configuration of GAN is vitally important, which determines whether this model can be optimised properly. Figure 3.2 shows the architecture of the unconditional GANs used in this work, which has a deep convolutional structure combined with spectral normalization layers. This architecture was first proposed in paper [25], which is called SNDCGAN. The benefits of this architecture have been described in chapter 2.

For the generator part, the input noise with dimensionality 100 is sampled from a Gaussian distribution. A fully connected layer is used to transfer the noise to a vector with dimensionality 5120, which is then mapped to a tensor with size $5 \times 2 \times 512$. Afterwards, four transposed convolutional layers are used to generate the feature maps, in which three of them have stride 2. The discriminator part has a similar structure. It takes feature maps ($40 \times 16 \times 1$) as inputs and then uses seven convolution layers plus one fully connected layer to classify them, where three of the convolutional layers have stride 2. The non-saturation loss function is used, which can be written as $Loss(D) = -\mathbf{E}_{x \sim p_{data}}[log(D(x))] - \mathbf{E}_{\hat{x} \sim p_g}[log(1 - D(\hat{x}))]$ and $Loss(G) = -\mathbf{E}_{\hat{x} \sim p_g}[log(D(\hat{x}))]$ respectively. Based on the analysis in section 2.1, we can easily obtain that the global optimum of the losses for discriminator and generator are $\log(4)$ and $\log(2)$ separately [2]. Therefore, in an ideal situation, the costs of generator and discriminator should converge to these two values during training (approximately 1.386 and 0.693 respectively).

[2] By substituting equation 2.3 into these two loss functions and set $p_{data} = p_g$.

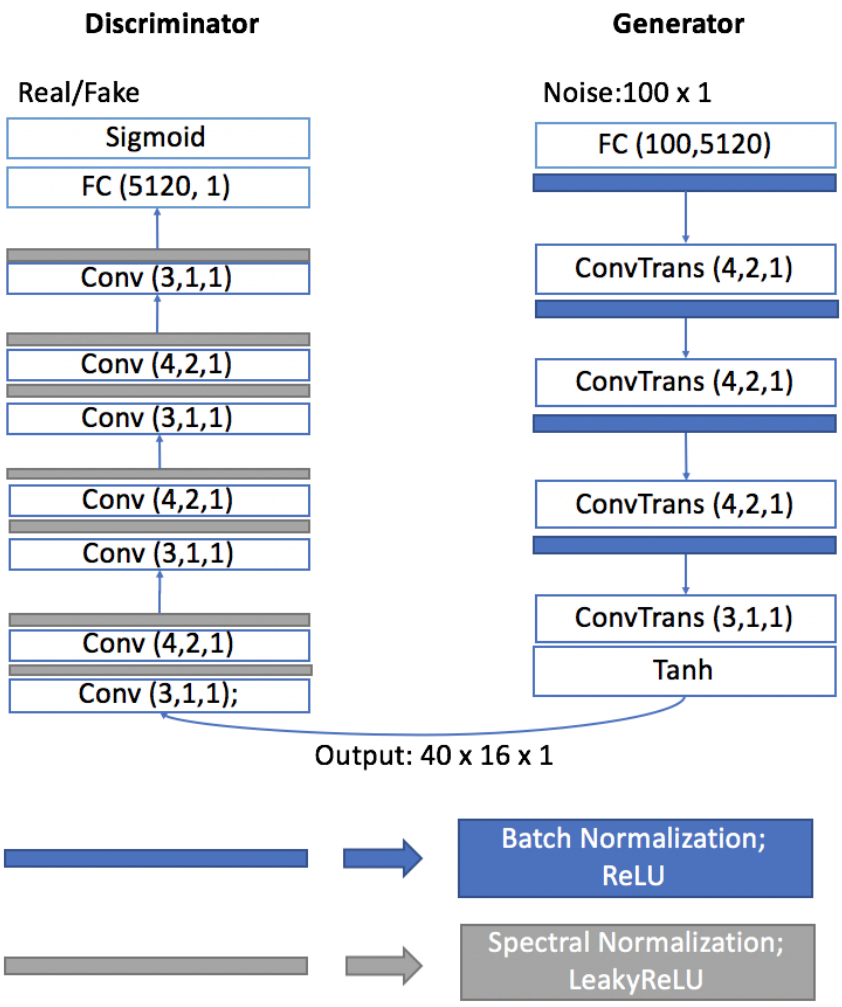


Fig. 3.2 Proposed GAN architecture: SNDCGAN

Preliminary experiments on MNIST

To check whether this structure can be trained in a steady way, we carried out a preliminary experiment on MNIST database, which contains a collection of handwriting digits [30]. The output shape of the generator and input shape of the discriminator were changed to $32 \times 32 \times 1$ to match the MNIST samples size. It has to be admitted that the models working on MNIST data set don't mean that they can work on the acoustic data set properly because the acoustic feature map is more complicated. Nevertheless, this experiment can provide a basic assessment of the proposed model configurations and helps to filter the bad model settings in the early stage.

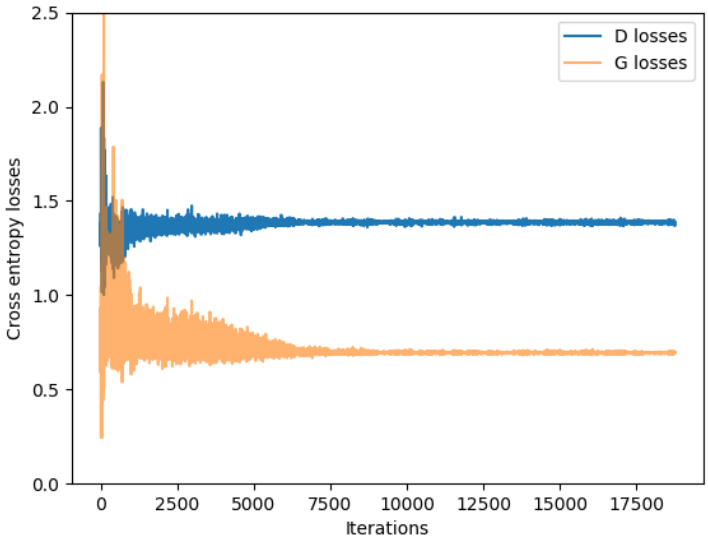
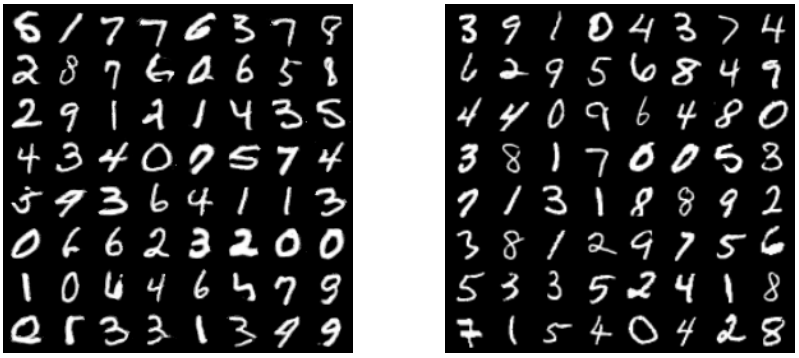


Fig. 3.3 SNDCGAN: training curve

Figure 3.3 plots the cross entropy losses of the discriminator and generator in each training iteration, where the discriminator is updated once followed by updating the generator once. The step ratio between discriminator and generator is important, and in this case, the best setting is $D : G = 1 : 1$. The number of epochs of training is 20 and each epoch has 930 training iterations. The reason why plotting the training curve in iteration level rather than epoch level is that we can inspect the convergence of the loss more precisely, which effectively reflects the stability of the training process. In figure 3.3, it can be observed that both D loss and G loss are converged to the desired global optimum, which proves that the training process is stable. In addition, the generated fake samples and the real samples are compared in figure 3.4. Most of the fake handwriting digits are clear and recognizable, and only a few of them are distorted in an unnatural way. Furthermore, almost all the digits are

generated with rich variations. This indicates that the diversity of the generated samples are ensured.



(a) Fake samples: 20 epoches (b) Real samples

Fig. 3.4 SNDCGAN: samples

3.2.2 Conditional GAN: Condition + SNDCGAN

In the original CGAN paper [27], a simple multilayer perceptron(MLP) structure was used for both generator and discriminator, and the authors point out that this is only used for proving their theory. As shown in figure 3.5, the quality of the MNIST samples generated by the original structure is rather low, which means that a more sophisticated CGAN structure is needed for producing usable samples.

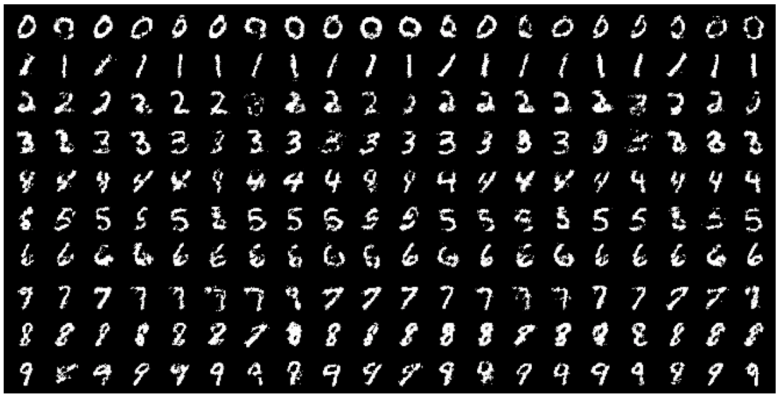


Fig. 3.5 Samples presented in CGAN paper

The first plan is conditioning the SNDCGAN described in section 3.2.1 on the state labels directly, which is referred as 'Condition + SNDCGAN'. The corresponding architecture is displayed in figure 3.6, where N is the number of classes. The loss functions for D

and G become $Loss(D) = -\mathbf{E}_{x\sim p_{data}}[log(D(x|y))] - \mathbf{E}_{\hat{x}\sim p_g}[log(1-D(\hat{x}|y))]$ and $Loss(G) = -\mathbf{E}_{\hat{x}\sim p_g}[log(D(\hat{x}|y))]$, where y is the conditioned state label.

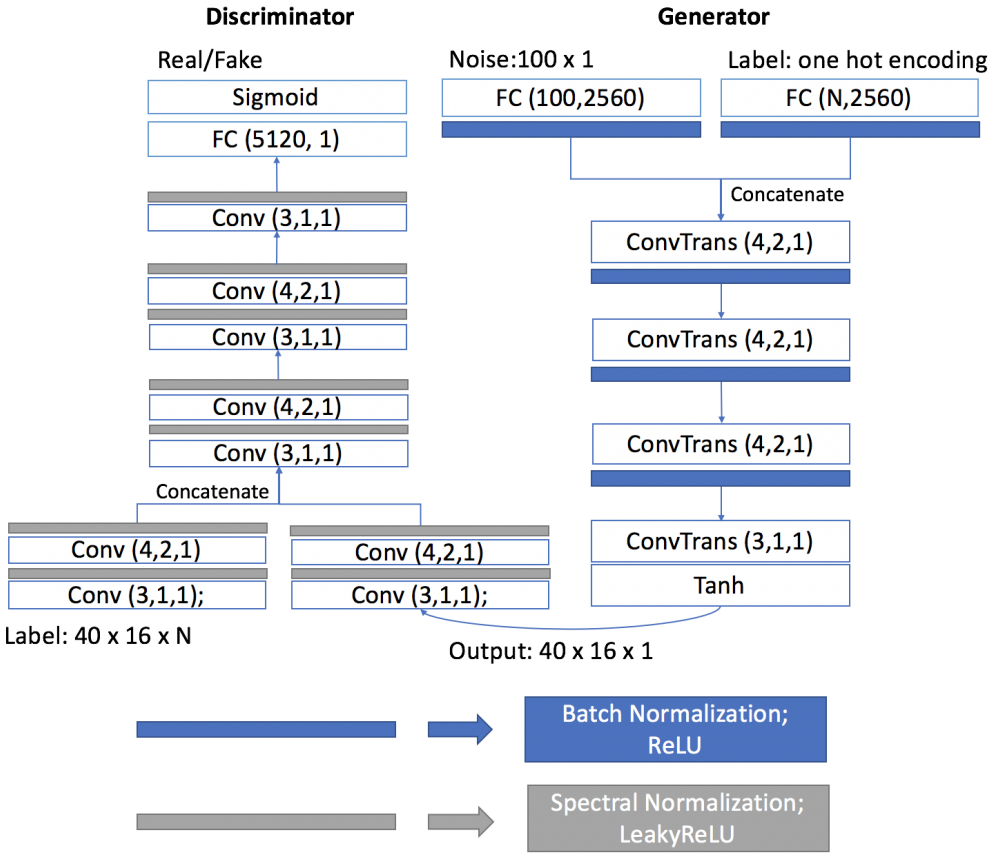
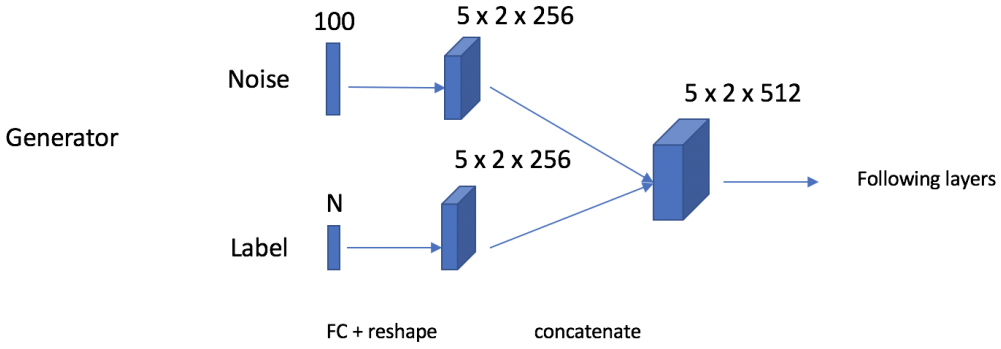


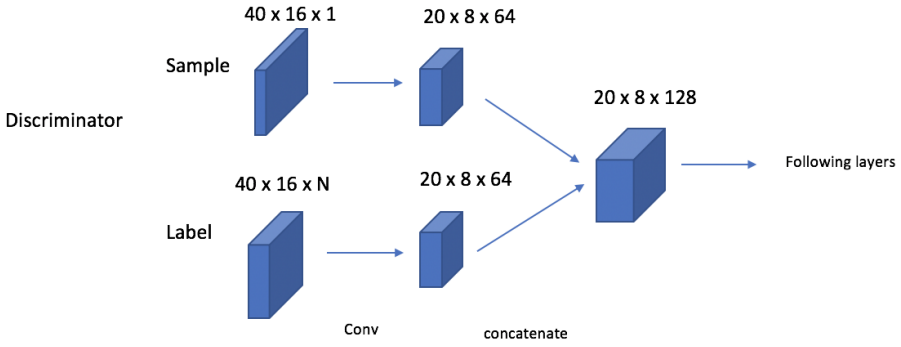
Fig. 3.6 Proposed CGAN structure 1: Condition + SNDCGAN.

Since the convolution layers are used, the data in networks are not flattened but presented by tensors with the shape ($hight \times weight \times channel$). Thus, the way to appending the additional information to the input of the discriminator and the generator is not as straightforward as the way for MLP. For the generator, the label information is encoded into a one-hot vector and then transferred by a fully connected layer to a vector with dimensionality 2560. This vector is then mapped to a tensor with shape $5 \times 2 \times 256$. In the meantime, the similar operations are done for the input noise, which generate another tensor with shape $5 \times 2 \times 256$. These two tensors will be concatenated to form a new tensor with shape $5 \times 2 \times 512$ before being sent into the first **ConvTrans** layer. For the discriminator part, the labels are encoded into a tensor with shape $40 \times 16 \times N$, where one of the channels is full of 1 and the rest channels are all 0. This form of encoding can be thought as an analogue to the one-hot embedding in a higher dimensional space. Next, two convolution layers are used to transfer

the encoded labels into a tensor with size $20 \times 8 \times 64$, and the same operations are done for the input feature maps. The two $20 \times 8 \times 64$ tensors will be concatenated before being sent to the next convolution layer. For a better understanding, the details of the above steps are illustrated in figure 3.7.



(a) Generator



(b) Discriminator

Fig. 3.7 Appending label information to the input layer of SNDCGAN

Preliminary experiments on MNIST

The 'Condition + SNDCGAN' structure was tested on MNIST database following the similar procedure in section 3.2.1. The number of classes, N, was set to 10 in this experiment. The corresponding training curve is plotted in figure 3.8. Both discriminator's loss and generator's loss have the tendency to converge. However, the generator's loss keeps fluctuating in a relatively large range, which indicates that the training of the generator is not very stable. The potential cause could be that the space we encoding the labels is very sparse especially for the discriminator part, which makes the optimisation difficult. In addition, this issue

could be amplified when using this structure on acoustic data because more classes will be involved.

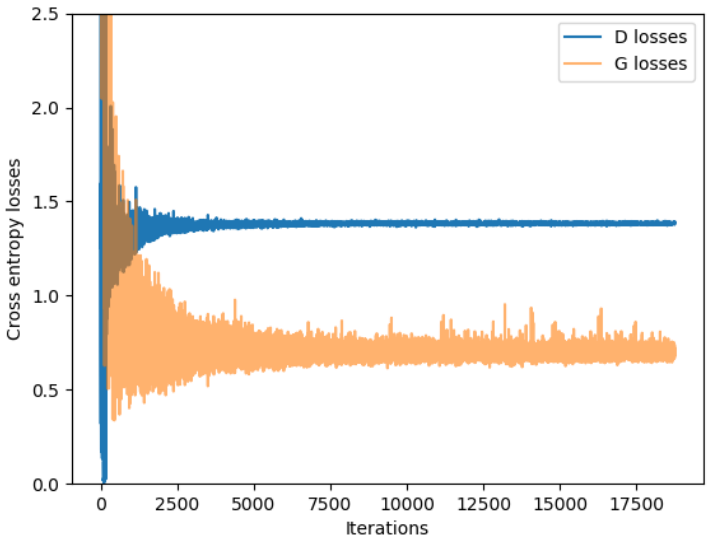
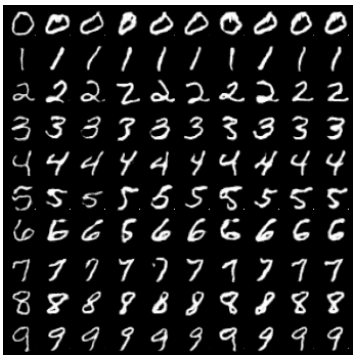


Fig. 3.8 Condition + SNDCGAN: training curve

Figure 3.9(a) shows the generated fake samples, where each row is conditioned on one label. It is clear that the generated fake samples still have rather high quality compared to the samples in figure 3.5. This proves that the SNDCGAN architecture is reliable.



(a) Fake samples: 20 epoches

(b) Real samples

Fig. 3.9 Condition + SNDCGAN: samples

3.2.3 Conditional GAN: SNDCGAN + Classifier

In this section, another conditional GAN architecture is proposed, which is referred as 'SNDCGAN + Classifier'. Instead of conditioning the discriminator on the label information,

we introduce an additional classifier to restrict the mode of the samples generated from the

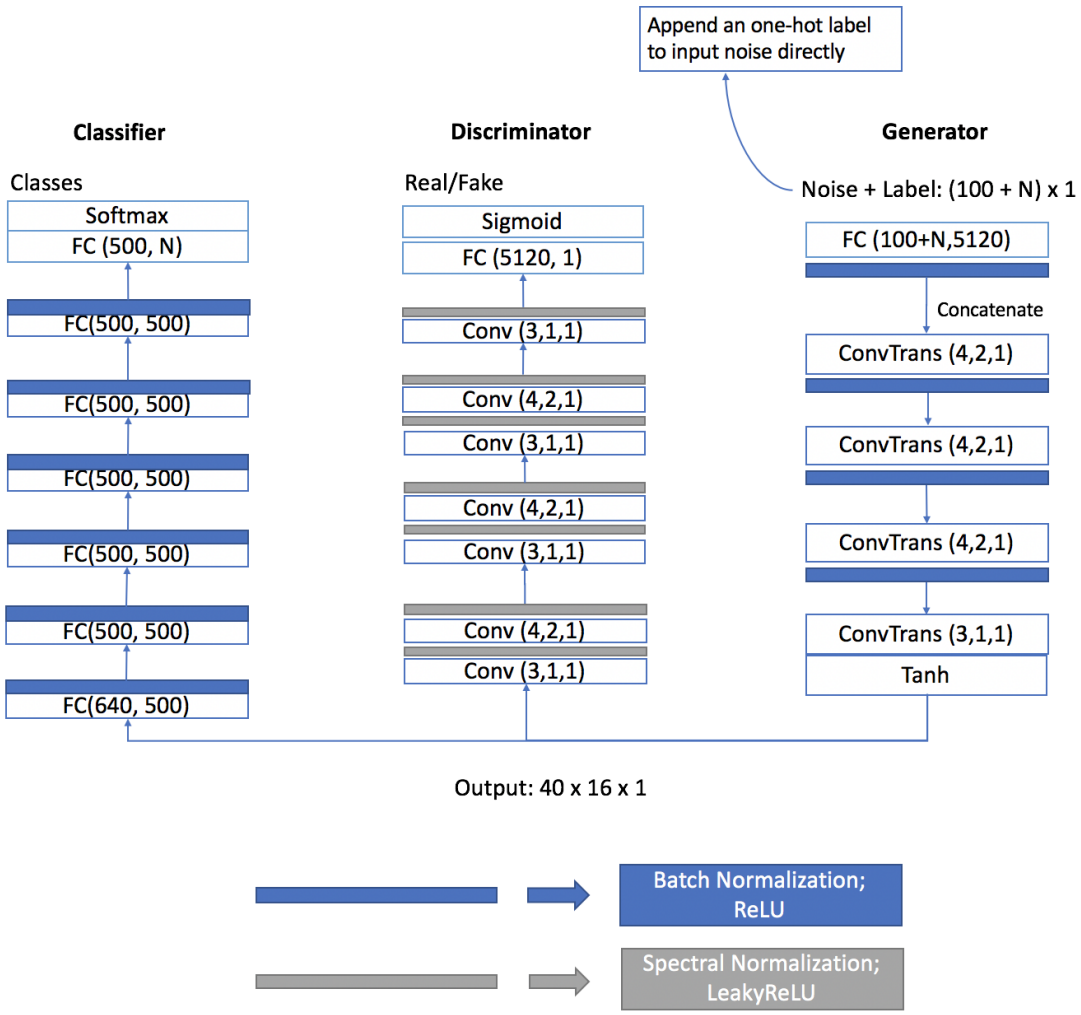


Fig. 3.10 Proposed CGAN structure 2: SNDCGAN + Classifier

conditioned generator. For the generator part, the one-hot label is now appended to the input noise directly, which form a new vector with dimensionality 100+N, where N is the number of classes. For the discriminator part, the original unconditioned structure is used. For the classifier part, a simple MLP structure is used, which contains five hidden layers with 500 hidden units in each layer. The classifier is fully trained by real data. The standard cross entropy training criteria are applied for discriminator and classifier. The loss function for the generator is slightly special, which is defined as

$$Loss(G) = -\mathbf{E}_{\hat{x}\sim p_g}[log(D(\hat{x}|y))] + \lambda \times Loss(C(\hat{x}|y)) \tag{3.1}$$

To minimize this loss, the generator should not only try to fool the discriminator but also try to ensure the class of the generated samples are corresponding to its conditions. The parameter λ controls the weight of the classifier's loss (or called misclassification penalty). In this task, λ was set to 0.01 so that the samples with relative high Loss(C) can still be generated. In other words, we do not want the generated samples are too naive. In addition, the training step ratio between discriminator, generator and classifier is tuned to $D : G : C = 1 : 1 : 1$. Compared to the structure proposed in section 3.2.2, the arrangements in this section significantly simplify the model structure for each part, which could contribute to the model optimisation.

Preliminary experiments on MNIST

The preliminary experiment for 'SNDCGAN + Classifier' architecture was also carried out on MNIST database. The corresponding training curve as well as the generated samples are displayed in figure 3.11 and 3.12 respectively. It can be seen that this structure leads to a better property of convergence compared to the former architecture 'Condition + SNDCGAN'. The diversity and quality of the generated samples are also relatively high. Since this structure configuration outperforms the 'Condition + SNDCGAN' structure, we will stick to using 'SNDCGAN + Classifier' architecture for following data augmentation tasks.

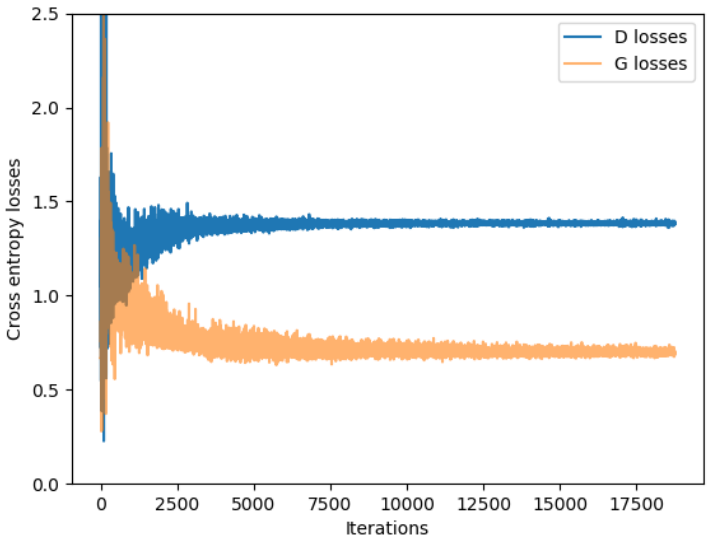


Fig. 3.11 SNDCGAN + Classifier: training curve

(a) Fake samples: 20 epoches (b) Real samples

Fig. 3.12 SNDCGAN + Classifier: samples

3.3 Data augmentation by conditional GANs array

The data augmentation scheme with conditional GANs array is presented in this section. The nice property of conditional GAN is that the labels of the generated data can be controlled. Thus, no extra transcribing work is needed in this scheme. The quality of the generated data can be tested by the original model trained by the initial data. The generated data and initial data can be merged directly because both of them have the frame level hard labels (or the reference alignment for initial speech data). The standard cross entropy training criteria and the stochastic gradient descent (SGD) algorithm will be used to re-train the acoustic model. The details of the pipeline are presented in algorithm 3.1, and the architecture is illustrated in figure 3.13.

Algorithm 1 Training Acoustic Model with conditional GANs array

1: Pre-train an acoustic model by the original training data set.
2: Train separate GANs for each phone unit (GANs array) by the original training data set. Each GAN is also conditioned on the tied triphone states so that the labels of the generated data can be controlled.
3: Test the quality of the generated data based on the pre-trained acoustic model.
4: Use the well-trained GANs array to generate fake data.
   Option 1: Generate same amount of data for each phone.
   Option 2: The amount of data for each phone and state are generated according to their prior distribution on real data.
5: Merge the generated data and the initial training data set, and re-train the acoustic model.

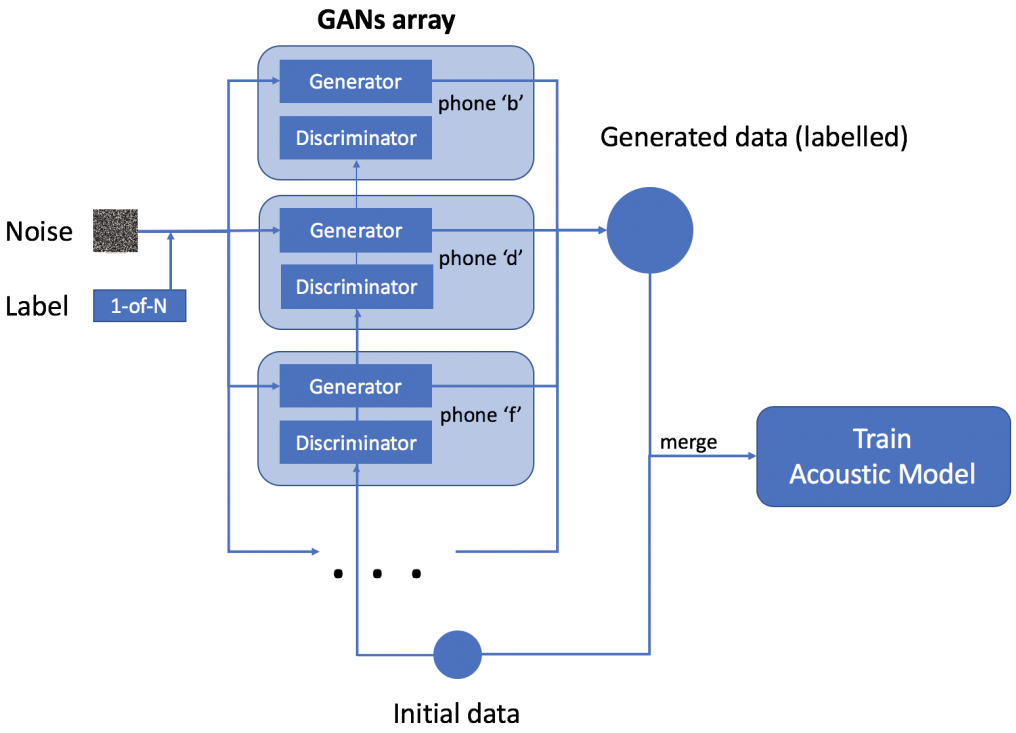


Fig. 3.13 Data augmentation pipeline with conditional GAN

3.4 Data augmentation by unconditional GANs array

This section introduces the data augmentation scheme with unconditional GANs array. The advantage of this scheme is that the data generated by unconditional GANs tend to have more variations, which is proved in section 4.3. However, the issue is that the true labels of the generated data are unknown because they are generated completely from random noise. To solve this problem, a semi-supervised learning strategy will be implemented, which assigns labels to the generated data by the original acoustic model (pre-trained on initial data). In detail, the pre-trained acoustic model can be utilised to classify the generated data. Since the phone level label is known (one GAN per phone), the classification range can be restricted to the states belonging to this specific phone, in which the original acoustic model should have relatively high accuracy (71.4% on TIMIT test set). Take the 'aa' GAN as an example, during generation, the amount of data need to be generated for each 'aa' state will be calculated first. Then the 'aa' data will be generated batch by batch by the well-trained GAN, where each batch of data should be classified by the pre-trained acoustic model. Once the amount of a certain state's data has reached the requirement, the following data which are classified into this state will be thrown out. The generation process for phone 'aa' will stop until all the states belonging to 'aa' has sufficient data. The details of the pipeline are presented in algorithm 3.2, and the architecture is illustrated in figure 3.14.

Algorithm 2 Training Acoustic Model with unconditional GANs array

1: Pre-train an acoustic model by the original training data set.
2: Train separate GANs for each phone unit (GANs array) by the original training data set.
3: Use the well-trained GANs array to generate fake data.
   Option 1: Generate same amount of data for each phone.
   Option 2: The amount of data for each phone and state are generated according to their prior distribution on real data.
4: Classify the generated data by the pre-trained acoustic model within the restricted states range. Then assign corresponding labels to the generated data.
5: Merge the generated data and the initial training data set, and re-train the acoustic model.

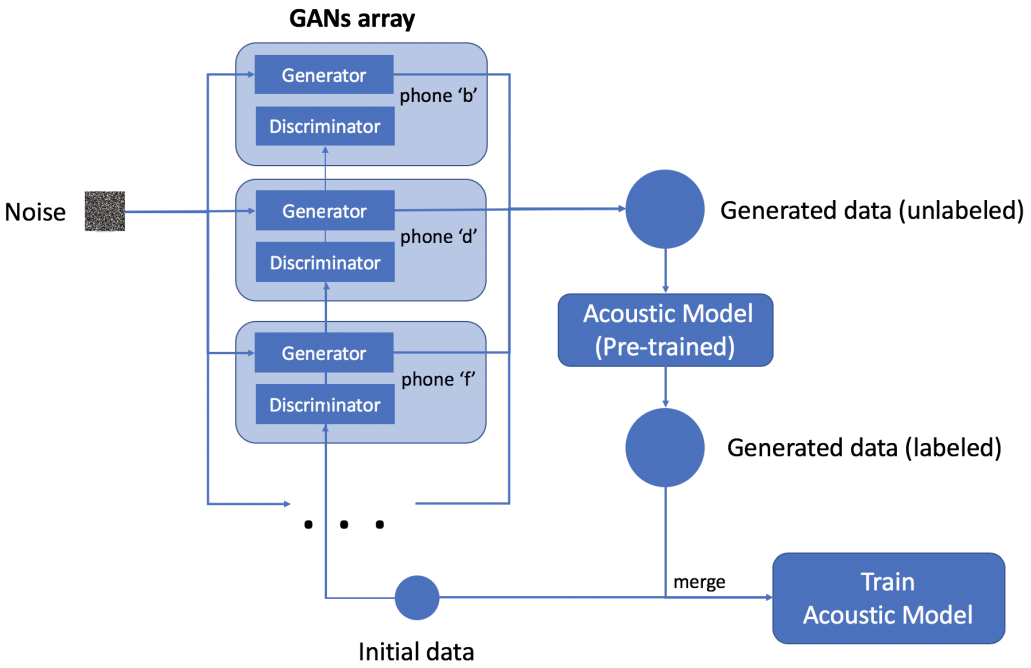


Fig. 3.14 Data augmentation pipeline with unconditional GAN

Chapter 4

Experiments and Results

4.1 Experimental setup

The experiments in this work are carried out based on TIMIT dataset, which is a small speech corpus designed for the development and evaluation of automatic speech recognition systems [31]. It contains 6300 sentences spoken by 630 speakers of eight major dialects of American English with time-aligned transcriptions in phone level. The sampling frequency of the speech waveform is 16KHz. In addition, the corpus initially contains a training set and a test set, which are balanced for the dialectal and phonic coverage. In our experiments, we further split the original training set into 3 hours new training set and 0.15 hours cross-validation (CV) set data. The CV set data are mainly used for testing the frame level accuracy of the trained acoustic model. The test set is the original one, which contains 1.14 hours data.

In our experiments, context-dependent deep neural networks based hidden Markov models (DNN-HMM) are built with an extended version of HTK 3.5 [32]. No language model is used, which reduces the accuracy but allows a clear focus on the acoustic model. Decision trees based states clustering is implemented and totally 808 tied triphone states are produced, including 3 silence states. The architecture of the DNN models is a multilayer perceptron which contains 5 hidden layers with 500 neurons in each layer. The models are trained with cross-entropy criteria, and the stochastic gradient descent (SGD) based back propagation algorithm is used for optimisation. The initial learning rate for SGD is set to 0.0005, while the wight decay and momentum are set to 0.001 and 0.5 respectively. The same hyper-parameters are used for all the experiments.

For generative adversarial networks, all the models are implemented with PyTorch [33]. The model structures presented in section 3.2.1 (Unconditional GAN) and 3.2.3 (Conditional GAN) are used, which has been verified on MNISIT dataset [30] with slight differences on sample shape. The non-saturation loss functions mentioned in section 2.1.1 are used.

The Adam solver is used for optimising both discriminator and generator (and classifier for CGAN), and the corresponding hyper-parameters are listed below:

- Learning rate α: 0.0002
- The first order momentum $\beta 1$: 0
- The second order momentum $\beta 2$: 0.9
- Batch size: 64
- Dimension of input latent noise: 100

In each training iteration, the training step ratio for unconditional GAN is set to $D : G = 1 : 1$, and for conditional GAN is $D : G : C = 1 : 1 : 1$. The training will stop when both average G loss and D loss have less than 0.001 changes in next epoch.

4.2 Evaluation metrics

The experiments in this chapter can be split into two parts. The first part is training GANs with the original data set, and the second part is re-training acoustic model with the augmented data. It is necessary to evaluate the system performance in each step.

4.2.1 Evaluation of GANs

A reliable GAN framework is the foundation for all the following experiments. Thus, in order to achieve a comprehensive evaluation on it, several evaluation metrics are used. Firstly, the stability of the training process for GAN models can be assessed by its training curve where the losses of the generator and discriminator in each iteration are plotted. The rate of convergence can be observed from this graph as well, which has been illustrated in the preliminary experiments in section 3.2.

The performance of GANs is also determined by the quality of the generated data, which is hard to be measured. In image area, the reliability of GANs can be roughly assessed by human subjective feelings. Other evaluation metrics, such as Inception score (IS) and Frechet Inception Distance (FID), are established based on the so-called inception models trained with ImageNet dataset [34, 35]. They can be used to measure the quality of the synthesised images in a quantitative way, although they will also introduce some mismatches between evaluation results and human feelings. However, none of the above methods can be adopted in this task directly because the synthesised data in this work are in speech spectrum

level. Instead, we utilise the pre-trained acoustic model to assess the performance of GANs. In general, this is realised by sending a batch of generated samples into the pre-trained model and then inspecting their average posterior probabilities (posteriors computed by the pre-trained model) as well as the classification results. We assume that the generated data should have the similar distribution as the real data. Thus, the test results should be similar to the evaluation output achieved from the test set data.

The evaluation strategies for unconditional GANs and conditional GANs are different. For unconditional GANs, the phone labels of the samples are known (separate GAN for each phone) but the state labels of the generated samples are unknown. In order to ensure the diversity of the generated samples, the posterior distribution over the triphone states of the target phone should have high entropy while the posterior distribution in phone level should have low entropy. In other words, for a well-trained unconditional GAN, most of the generated samples should be classified into the target phone but their variability over triphone states should be high. For conditional GANs, since the state label of each sample is known, we only need to consider their state-level classification results, which should be similar to the results obtained from the test set data.

4.2.2 Evaluation of re-trained acoustic model

The performances of re-trained acoustic models are evaluated by recognising the TIMIT test set, for which the phone level transcriptions are available. The standard decoding pipeline in HTK is used. The HTK toolkit is also used to produce the optimal alignment based on the reference transcriptions using dynamic programming. Then the phone error rate (PER) can be calculated by the following equation,

$$Phone\ Error\ Rate = \frac{D+S+I}{N} \times 100\% \tag{4.1}$$

where D denotes the number of deletion errors, S denotes the number of substitution errors, I refers to the number of insertion errors and N is the number of total labels. Apart from PER, the training set and cross-validation set frame level accuracies are also useful for measuring the classification performance of the DNN acoustic models, which have a certain correlation with the final PER.

4.2.3 Baseline method: Speed perturbation

The speed perturbation (SP) method is also implemented [14], which is used as the baseline speech data augmentation method. In this experiment, the speed of the original audio signal in TIMIT is changed with factors 0.9 and 1.1 separately. By combining them with the audio data with original speed, a 9-hours training set can be produced. The model performance after using this method is displayed in table 4.1. It is apparent that this method does improve the mode performance. Compared to the original model, both the training set and CV set frame level accuracies are increased and a relative 3.7% PER reduction is achieved.

Table 4.1 Baseline: speed perturbation method

Data	Hours	Frame level acc		PER
		Training set	CV set	
Original	3	69.67	50.27	25.17
SP + Original	9	70.60	51.28	24.23

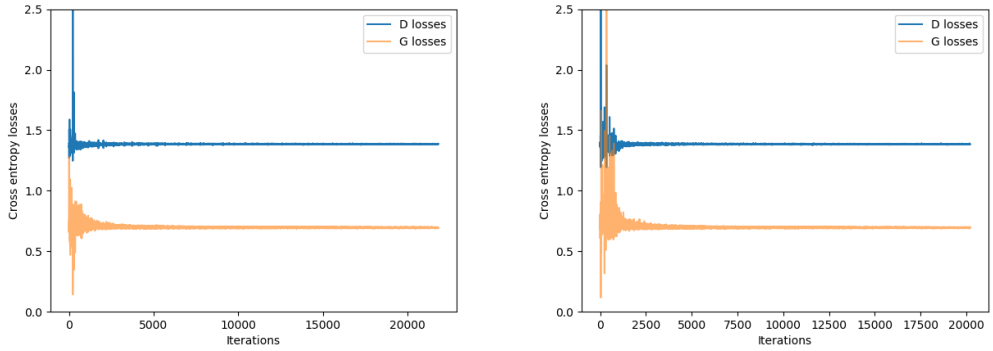
4.3 Experiments for training GANs

The first part of the work is training GANs by initial TIMIT training set, where 48 context independent phones are used. Since each phone has a separate GAN, the training set is divided into 48 subsets according to the phoneme of the data. When training conditional GANs, the state level labels should also be provided. In the following subsections, the performance of the trained conditional and unconditional GANs are assessed respectively. An extra evaluation on single GAN system is carried out, where one single conditional GAN is trained for all the phones.

4.3.1 Conditional GANs array

For conditional GANs, the architecture presented in section 3.2.3 is used. Since each phone has a different number of tied states in triphone system, the number of classes that each GAN conditioned on is different as well. In addition, the amount of training data for each phone is also different. Thus, the performance of each GAN should be inspected separately. However, it is not feasible to plot all of them in this section. Instead, we choose one vowel 'aa' and one consonant 'm' as typical cases to illustrate the evaluation results.

Firstly, the training curve for 'aa' CGANs and 'm' CGANs are plotted in figure 4.1. Both of them have a very stable learning process, where the D loss and G loss converges rapidly within 5000 iterations.

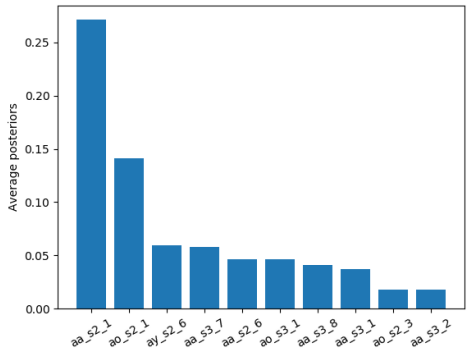


(a) CGAN for 'aa': training curve (b) CGAN for 'm': training curve

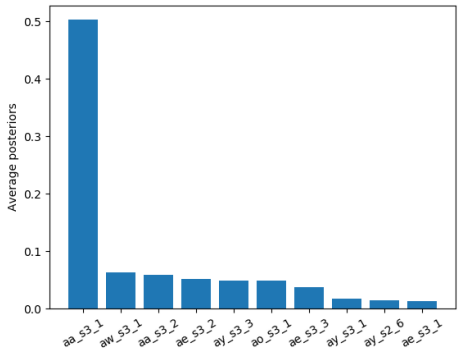
Fig. 4.1 CGANs' training curve: 'aa' and 'm'

Next, the quality of the generated data is evaluated in two phases. Ten thousand samples are generated for each tied state first. These samples are then tested by the original acoustic model. In the first phase, the average posteriors of these samples over 808 states are computed. The top 10 average posteriors and their corresponding states can be plotted. Figure 4.2 shows the part of the results for such plots, including the data for state 'aa_s2_1', 'aa_s3_1', 'aa_s4_1', 'm_s2_1', 'm_s3_1' and 'm_s4_1'. It can be observed that the target phone-state always achieves the highest average posterior and the other relative high posterior states tend to have the similar characteristics of the target phone state. This indicates that the generated feature maps are close to the real distribution.

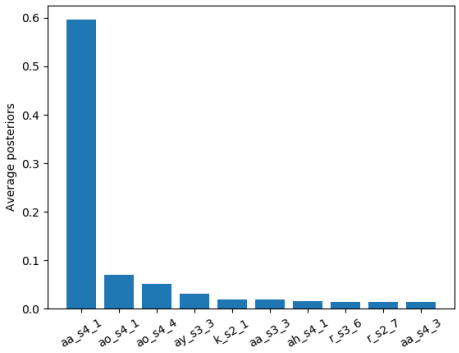
In the second phase, the corresponding top1, top3 and top5 classification accuracies for the generated samples are computed. The same classification procedure for test set is also applied. If the generated dataset has the similar distribution as the real dataset, the classification results for them should be similar as well. Table 4.2 and table 4.3 display the classification results for part of the states belonging to phone 'aa' and phone 'm'. It is clear that the original model's classification accuracies for the generated data are rather high, which means that most of the generated data are reliable. Nevertheless, the generated data of many states, such as 'aa_s3_1', 'm_s3_3', achieve more than twice of the top1 accuracies for the corresponding test set data, which indicates that the generated data may lack variations. Two potential factors may cause this problem. One is that both GANs and the original acoustic model may overfit the training set to a certain extent. In this situation, the generated data could be similar to the training data, and the original acoustic model has much higher classification accuracy for them. The other factor could be that the objective function for conditional GANs generator, which is equation 3.1, may encourage the model to generate



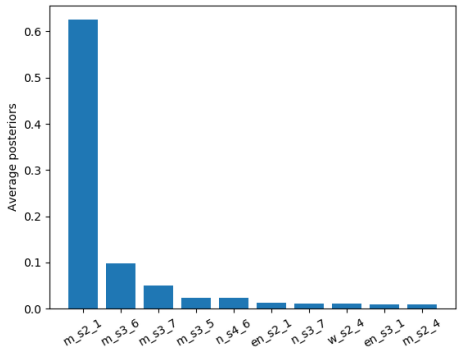
(a) Average posteriors top10 (state: aa_s2_1)



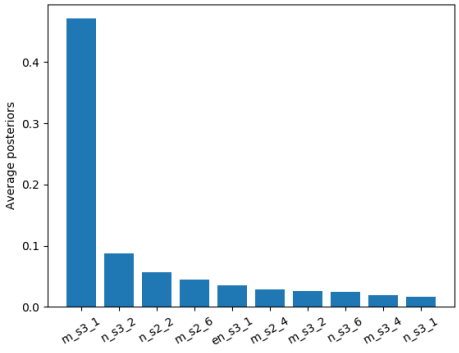
(b) Average posteriors top10 (state: aa_s3_1)



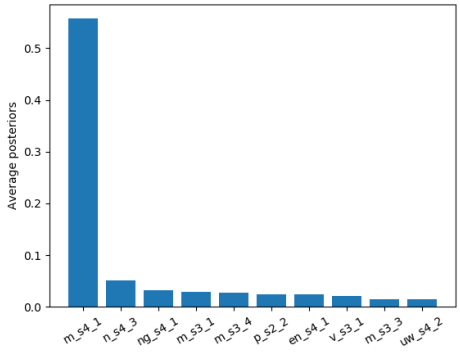
(c) Average posteriors top10 (state: aa_s4_1)



(d) Average posteriors top10 (state: m_s2_1)



(e) Average posteriors top10 (state: m_s3_1)



(f) Average posteriors top10 (state: m_s4_1)

Fig. 4.2 Fidelity test for fake feature maps generated by CGANs: ’aa’ and ’m’

samples which can be easily classified. This is because the naive samples can lead to a lower classifier's loss.

To further investigate the distribution of the generated data, a 3-hours simulated acoustic dataset is produced, where the amount of data generated for each state is according to the prior distribution of the original dataset. The cross entropy losses for the generated dataset as well as the TIMIT subsets, including training set, CV set and test set, are computed by the pre-trained acoustic model. The corresponding results are displayed in the form of box plot in figure 4.3. It is clear that the losses for CV set and test set have similar distribution and they achieve the highest average losses. The average loss for the training set is the lowest because the pre-trained acoustic model is trained based on this dataset. The average loss for the generated dataset is lower than that for the test set but higher than that for the training set. The variance of the generated data's losses is also situated between them. This indicates that the GANs generative model does learn some variations rather than simply replicate the original training set, although the real data may contain more variations which have not been captured by either the acoustic model or GANs.

Table 4.2 Classification accuracies for CGANs' samples and TIMIT test set (phone: 'aa')

Tied states	Generated samples			Test set		
	%top1	%top3	%top5	%top1	%top3	%top5
aa_s2_1	10.3	27.4	41.1	13.6	37.8	51.1
aa_s2_2	56.2	79.6	86.8	30.9	56.4	67.2
aa_s2_3	65.9	95.4	97.3	40.9	75.5	83.9
aa_s3_1	67.7	93.7	96.1	23.6	55.7	72.8
aa_s3_2	59.4	93.2	97.8	26.1	53.9	68.4
aa_s3_3	69.3	91.1	95.7	29.6	50.2	61.5
aa_s4_1	51.3	67.3	73.5	47.1	75.7	83.3
aa_s4_2	31.8	65.4	77.8	43.7	82.6	85.4
aa_s4_3	60.4	85.2	91.1	21.5	38.9	52.2

4.3.2 Unconditional GANs array

For unconditional GANs array, the architecture presented in section 3.2.1 is used. The similar evaluation procedure for conditional GANs array is applied, where the GANs for phone 'aa' and 'm' are selected as typical cases to illustrate the evaluation results.

Firstly, the training curves for unconditional GANs (for phone 'aa' and 'm') are plotted in figure 4.4. Compared to the conditional GANs, the convergence speed for unconditional GANs is slightly faster. Especially for phone 'm', the D loss and G loss only have small

Table 4.3 Classification accuracies for CGANs' samples and TIMIT test set (phone: 'm')

Tied states	Generated samples			Test set		
	%top1	%top3	%top5	%top1	%top3	%top5
m_s2_1	73.3	94.2	96.4	31.3	65.3	76.0
m_s2_2	70.1	94.9	97.3	62.1	85.9	92.2
m_s2_3	39.8	63.7	73.8	24.1	51.8	58.9
m_s3_1	57.1	78.7	87.5	42.8	68.5	78.1
m_s3_2	27.1	67.1	80.9	25.6	63.6	82.4
m_s3_3	73.2	88.8	93.2	33.8	58.9	71.5
m_s4_1	49.5	75.9	84.4	38.1	65.7	76.3
m_s4_2	37.2	69.7	83.4	39.4	67.6	79.4
m_s4_3	70.3	86.6	93.6	45.3	68.8	80.5

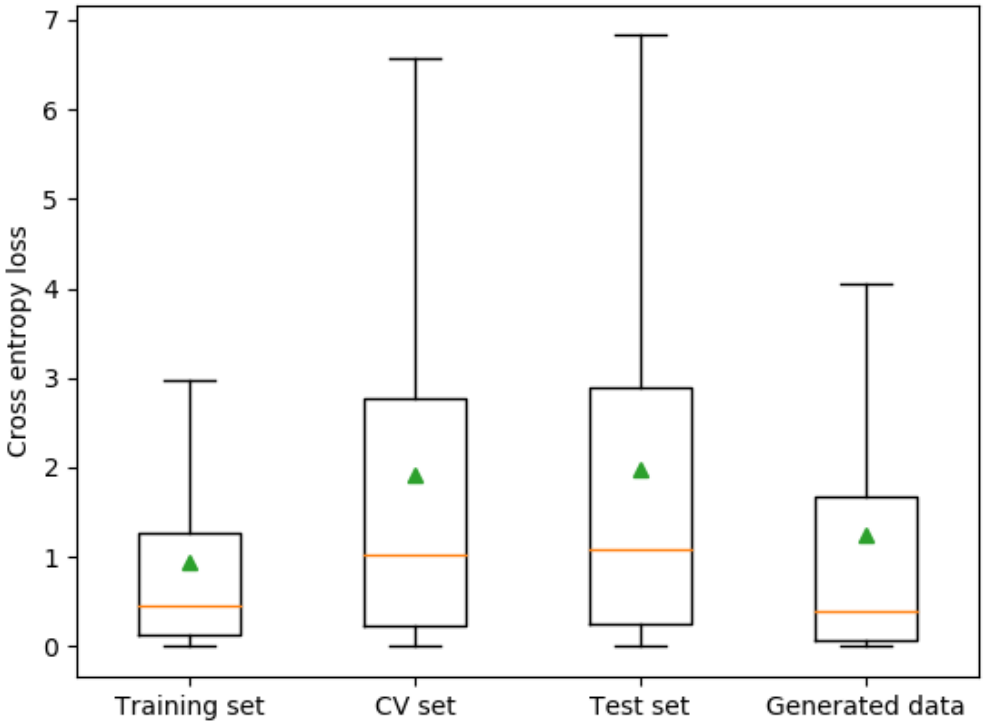
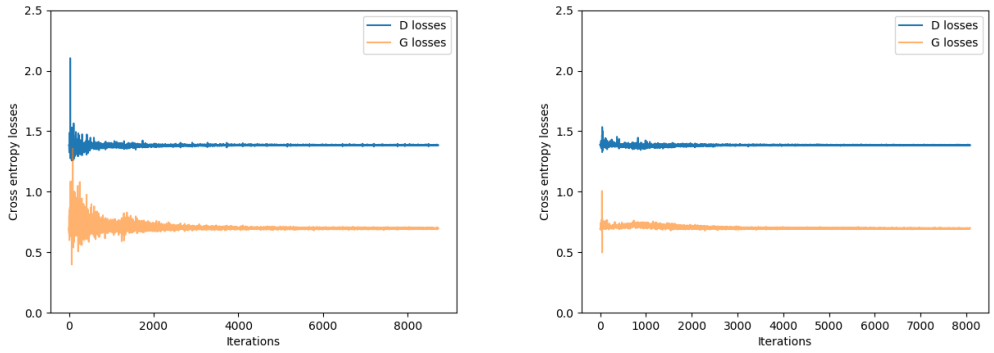


Fig. 4.3 Cross entropy losses for each TIMIT sub-dataset & fake dataset generated by conditional GANs

fluctuations in the first 2000 iterations. This reveals that training unconditional GANs is much easier.



(a) GAN for 'aa': training curve

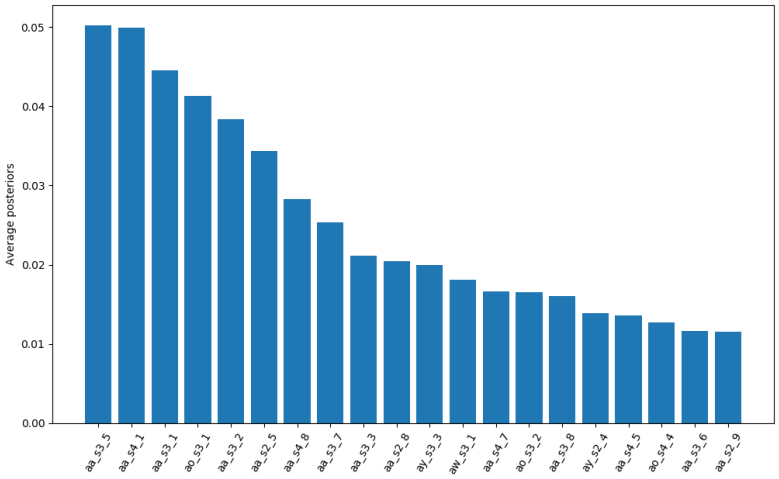
(b) GAN for 'm': training curve

Fig. 4.4 Unconditional GANs' training curve: 'aa' and 'm'

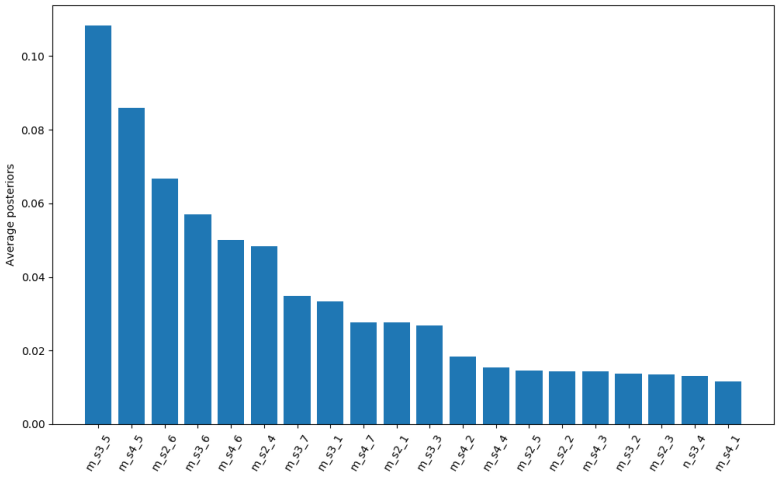
Afterwards, the quality of the generated data is also inspected. For each phone, we generate 10 thousand samples for which the state labels are unknown. Then the pre-trained acoustic model is utilised to compute the average posteriors for these samples. The top 20 average posteriors and their corresponding states for fake 'aa' data and fake 'm' data are plotted in figure 4.5(a) and 4.5(b) respectively. It can be observed that most of the states with high posteriors belong to the target phone, which indicates that the distribution of the target phone's data has been learned by the GANs generator. In addition, no state get significantly higher posteriors than other states, which means that the diversity of the generated data is also high and the 'mode collapse' [1] problem doesn't happen. During the generation process, we need to ensure all the triphone states can be generated.

To investigate the distribution of the generated dataset produced by unconditional GANs, the same cross entropy loss test used in section 4.3.1 is applied. As mentioned in section 3.4, to generate a usable dataset by unconditional GANs, the pre-trained acoustic model should be used to label the data. Then based on this manner, a 3-hours labelled dataset can be generated. The corresponding evaluation results are plotted in figure 4.6. It can be seen that more variations are created by the unconditional GANs scheme compared to the conditional GANs. The reason could be that the conditions applied to CGANs limit the variety of the generated data. The average loss for samples generated by unconditional GANs is still lower than that for TIMIT test set.

[1] mode collapse problem is the common failure case for GANs where the generated samples have extremely low variety



(a) Average posteriors top20 (phone: aa)



(b) Average posteriors top20 (phone: m)

Fig. 4.5 Fidelity test for fake feature maps generated by unconditional GANs: ’aa’ and ’m’

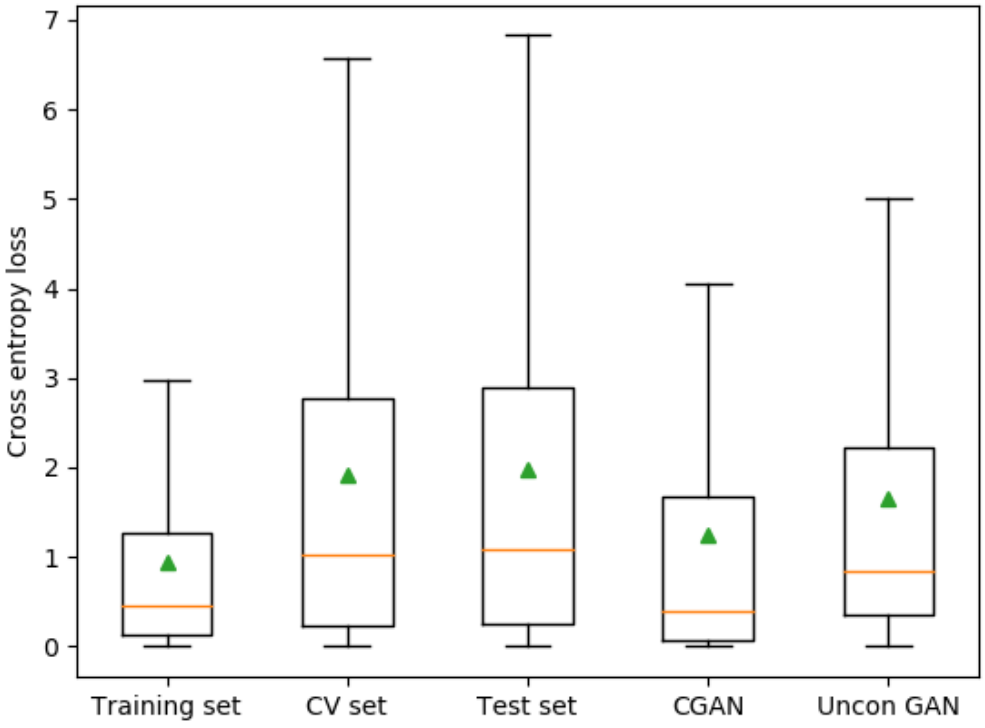


Fig. 4.6 Cross entropy losses for each TIMIT sub-dataset & fake dataset generated by CGANs and Uncon GANs

4.3.3 Single GAN for all the phones

We also considered training a single GAN to generate data for all the phones. This requires the generator to be conditioned on 808 tied triphone states, which makes the training process extremely difficult. As shown in table 4.4, the classification accuracies for samples generated by single GAN is very strange (9 states for phone 'aa', 'm', 'n' are picked). The accuracies are either very high, which means the generated data are rather simple, or very low, which means the generated data are completely wrong. Although different structures and hyperparameter settings can be applied to improve the performance of single GAN system, the search space for this is prohibitively expensive (training single GAN is very time-consuming) and the expected performance still could be much lower than GANs array system.

Table 4.4 Classification accuracies for samples generated by single CGAN

Tied states	Generated data (single GAN)		
	%top1	%top3	%top5
aa_s2_1	100	100	100
aa_s3_1	78.2	100	100
aa_s4_1	100	100	100
m_s2_1	80.3	100	100
m_s3_1	100	100	100
m_s4_1	0	0	0
n_s2_1	0	0	0
n_s3_1	98.6	99.8	99.9
n_s4_1	16.8	90.2	98.8

Training an unconditional single GAN for all the phones could potentially work in the generation part, while transcribing the generated data will become a big problem. The pre-trained acoustic model has a rather low accuracy for classifying between all the triphone states, which is lower than 50% on TIMIT test set. If we restrict the classification range to a certain phone's states, the corresponding accuracy could be increased to 71.4%. Due to the above reasons, the GANs array system will be used for final data augmentation task.

4.4 Experiments with conditional GANs array

In this section, the augmented data produced by conditional GANs array are used to re-train the acoustic model. For the experiments carried out under different settings, the same random seed is applied to ensure their results are comparable. Firstly, two generation modes are tested. One is 'uniform' mode, where all the phones get the same amount of data. Another is 'prior' mode, where the amount of data generated for each phone and state is according to

the prior distribution in the original training set. Under each mode, three hours simulated data are generated. Table 4.5 shows the performance of the acoustic models trained based on these simulated data. The performances of the acoustic models trained purely by simulated data are firstly investigated. The corresponding results are shown in the second and third line of table 4.5, where the first line shows the performance of the original model. The model under both modes achieves extremely high frame level accuracies for training set (over 99%) while the frame level accuracies for CV set are very low. This reflects that the generated data can be easily classified and the acoustic model trained based on these data cannot be generalised to the real situation.

Furthermore, we combine the generated data with the original TIMIT training set to train new acoustic models. The corresponding results are shown in the last two lines of table 4.5. Compared to the original acoustic model, the new models' frame level accuracies for the training set are much higher while the accuracies for CV set are slightly lower. In terms of the phone error rate (PER), the performances of the re-trained acoustic models are degraded significantly after using the augmented data. This result reveals that some unnatural variations are created in simulated data which direct the acoustic model to fit a wrong distribution. The corresponding PER under 'prior' mode is slightly lower than that under 'Uniform' mode. The reason could be that the target penalty for HMM states cannot be calculated correctly under 'uniform' mode. Target penalty is used for posteriors to log-likelihood conversions, which is defined as:

$$\log p(o(t)|C_k) = \log p(C_k|o(t)) + \log p(o(t)) - \log p(C_k) \tag{4.2}$$

where $o(t)$ is the input acoustic data, C_k is the ANN target and $\log p(o(t)) - \log p(C_k)$ is the target penalty. The calculated log-likelihood is then used for decoding. Due to this fact, the following experiments in this section will use 'prior' mode to generate data.

Table 4.5 CGANs scheme: acoustic model performance under two data generation modes

Data	Mode	Hours	%Frame level acc		%PER
			Training set	CV set	
Original		3	69.67	50.27	25.17
CGANs	'Uniform'	3	99.59	10.07	
CGANs	'Prior'	3	99.62	12.08	
CGANs + Original	'Uniform'	6	86.32	48.32	26.25
CGANs + Original	'Prior'	6	86.24	48.69	26.14

As discussed above, there are two potential problems for conditional GANs array. The first problem is that it could generate many naive samples which can be easily classified,

which has no contribution to improving acoustic model performance. Another problem is that the generated data may contain many strange variations, which bias the acoustic model to fit an incorrect distribution. To address these two problems, the generated samples are filtered based on their posteriors over target states computed by pre-trained acoustic model. The samples with too high or too low posteriors will be rejected because these samples are either too simple or too strange. The same amount of simulated data, which is 3 hours, are generated in each experiment and the corresponding results are shown in table 4.6 and 4.7.

In table 4.6, the acoustic models are trained by the generated data solely. The posteriors range in the table refers to the range of data that will be retained. For instance, the range '≥ 0.1' means that the samples for which the posterior higher than 0.1 will be retained. It can be observed that rejecting samples with too low posteriors can effectively increase the model's accuracy for CV set , which means the amount of the useful part of the data is increased. By contrast, rejecting samples with too high posteriors cannot improve CV set accuracy but do reduce the training set accuracy.

Table 4.6 CGANs scheme (prior mode): performance of acoustic models trained purely by filtered fake data

Data	Posteriors range	Hours	%Frame level acc	
			Training set	CV set
CGANs	no rejection	3	99.62	12.08
CGANs	≥ 0.1	3	99.94	15.59
CGANs	≥ 0.2	3	99.82	15.53
CGANs	≥ 0.3	3	99.76	15.54
CGANs	≤ 0.9	3	91.36	11.54
CGANs	≤ 0.8	3	98.93	10.23
CGANs	≤ 0.7	3	99.21	10.35

In table 4.7, the generated data are merged with the original data to re-train the acoustic model. Compared to the evaluation results of the 'no rejection' experiment in the second line of table 4.7, rejecting low posteriors samples leads to a notable reduction in PER, which is corresponding to the observations in table 4.6 (posters range: $\geq 0.1, \geq 0.2, \geq 0.3$). The experiment with posteriors range '≥ 0.2' leads to the lowest PER at 25.75. In addition, rejecting samples which have larger than 0.9 posteriors can also improve the model performance. By combining this two strategies, we carried out an additional experiment, where the data with posteriors in range 0.2 to 0.9 are retained. The corresponding evaluation results are shown in the last line of table 4.7. Although this setting achieves lowest PER among all the filtering experiments, its performance is still worse than the original model trained by the initial training set. Compared to the original model, under this setting, the

Table 4.7 CGANs scheme (prior mode): performance of acoustic models trained by filtered fake data plus original data

Data	Posteriors range	Hours	%Frame level acc		%PER
			Training set	CV set	
Original		3	69.67	50.27	25.17
CGANs + Original	no rejection	6	86.24	48.69	26.14
CGANs + Original	≥ 0.1	6	85.75	49.77	25.77
CGANs + Original	≥ 0.2	6	85.38	49.84	25.75
CGANs + Original	≥ 0.3	6	85.78	49.91	25.92
CGANs + Original	≤ 0.9	6	85.27	49.33	25.71
CGANs + Original	≤ 0.8	6	85.45	48.39	26.57
CGANs + Original	≤ 0.7	6	86.15	48.85	26.15
CGANs + Original	$0.2-0.9$	6	85.32	49.87	25.69

model's accuracy for training set increases dramatically to 85.32% while the frame level accuracy for CV set decreases to 49.87% and PER increases to 25.69%. This setting is further compared with the speed perturbation method in table 4.8 and it is apparent that the CGAN method doesn't work.

Overall, the experiments in this section show that CGANs array cannot create useful variations to improve the generalisation performance of the acoustic model. Instead, the data with wrong variations created by CGANs make the acoustic model being optimised in a wrong direction. In the meantime, we show that filtering generated data can improve the data's quality but they are still useless for data augmentation.

Table 4.8 CGANs scheme (prior mode): comparison with speed perturbation method

Data	Hours	%Frame level acc		%PER
		Training set	CV set	
Original	3	69.67	50.27	25.17
SP + Original	9	70.60	51.28	24.23
CGANs (0.2-0.9) + Original	6	85.32	49.87	25.69

4.5 Experiments with unconditional GANs array

In this section, the augmented dataset produced by unconditional GANs array are used to re-train the acoustic model. Similarly, two data generation modes, including 'prior' mode and 'uniform' mode, are used in the initial experiment and the corresponding results are shown in table 4.9. The first line of the table shows the performance of the original model.

The second and third line of the table shows the performance of the model trained purely by generated samples, and we will focus on these two lines at first. Compared to the data generated by conditional GANs, the data generated by unconditional GANs are less naive. The training set frame level accuracies are situated at a reasonable level (for CGAN, the corresponding accuracies are over 99%). However, the CV set frame level accuracies are still very low, which indicates that the generated data contains certain patterns which are unmatched with the real data.

The last two lines of table 4.9 show the performances of the models trained by the merged dataset. According to the above analysis, it is not surprising that these models have higher phone error rate than that of the original model. In addition, it can be seen that the training data under 'prior' mode lead to lower phone error rate, which proves that the fake data should be generated according to the original data's states distribution. In this section, the following experiments will use 'prior' mode to generate data.

Table 4.9 Uncon GANs scheme: acoustic model performance under two data generation modes

Data	Mode	Hours	%Frame level acc		%PER
			Training set	CV set	
Original		3	69.67	50.27	25.17
Uncon GANs	'Uniform'	3	78.28	15.93	
Uncon GANs	'Prior'	3	74.34	15.13	
Uncon GANs + Original	'Uniform'	6	77.48	49.98	25.92
Uncon GANs + Original	'Prior'	6	73.06	49.99	25.46

Similar to section 4.4, the filtering experiments are also carried out, where the samples with too high or too low posteriors are rejected. Rejecting samples with too low posteriors is especially useful for the scheme with unconditional GANs because the state labels of the generated data are assigned by the pre-trained acoustic model. It is possible that the data with low posteriors have the incorrect labels due to the misclassification of the pre-trained acoustic model. Such data could hinder the acoustic model from learning a correct decision boundary.

In table 4.10, the acoustic models are trained by the generated data solely. It can be seen that the CV set accuracies increase gradually as more and more low posteriors samples are rejected (2nd to 4th line in table 4.10). This proves that blocking samples with too low posteriors can improve the effectiveness of the generated data. However, the corresponding frame level accuracies for the training set are also increased, which reflects that many marginal samples are also lost in this filtering process. As for rejecting samples with too high posteriors, no gains are observed in this part. In table 4.11, the performances of the models

Table 4.10 Uncon GANs scheme (prior mode): performance of acoustic models trained purely by filtered fake data

Data	Posteriors range	Hours	%Frame level acc	
			Training set	CV set
Uncon GANs	no rejection	3	74.34	15.13
Uncon GANs	≥ 0.1	3	79.81	19.98
Uncon GANs	≥ 0.2	3	85.25	20.43
Uncon GANs	≥ 0.3	3	85.62	21.11
Uncon GANs	≤ 0.9	3	73.96	14.34
Uncon GANs	≤ 0.8	3	76.01	13.01
Uncon GANs	≤ 0.7	3	73.28	12.11

trained by the merged datasets are investigated. All the rejection settings lead to a slightly worse performance with increased PER, and their influence on the accuracies for CV set are all very small. This indicates that the incorrect parts of the generated samples are difficult to be filtered out.

Table 4.11 Uncon GANs scheme (prior mode): performance of acoustic models trained by filtered fake data plus original data

Data	Posteriors range	Hours	%Frame level acc		%PER
			Training set	CV set	
Original		3	69.67	50.27	25.17
Uncon GANs + Original	no rejection	6	73.06	49.99	25.46
Uncon GANs + Original	≥ 0.1	6	76.26	49.20	25.78
Uncon GANs + Original	≥ 0.2	6	76.25	50.03	25.63
Uncon GANs + Original	≥ 0.3	6	78.22	50.2	25.49
Uncon GANs + Original	≤ 0.9	6	74.42	49.10	25.48
Uncon GANs + Original	≤ 0.8	6	70.13	49.72	25.56
Uncon GANs + Original	≤ 0.7	6	75.28	49.47	25.57

Table 4.12 Uncon GANs scheme (prior mode): performance of acoustic models trained by filtered fake data plus original data (entropy based filtering)

Data	Rejection mode	Hours	%Frame level acc		%PER
			Training set	CV set	
Uncon GANs + Original	no rejection	6	73.06	49.99	25.46
Uncon GANs + Original	ent ≤ 1	6	74.26	49.47	25.79
Uncon GANs + Original	cv max ent	6	76.73	48.95	25.84
Uncon GANs + Original	cv min ent	6	74.75	49.92	25.64

An additional entropy based rejection method is also tried, where the entropy of the posterior distribution for each sample is calculated. We reject the samples with too low entropies because they tend to have a sharp distribution, which means the original model is too confident to classify them. We also reject the samples with too high entropies, which means the data may not contain useful information for training. Three strategies are used and their results are shown in table 4.12. The rejection mode 'ent $\leq$1' means that the samples with entropy lower than 1 will be rejected, 'cv max ent' means that the samples with entropy larger than the maximum entropy for CV set samples will be rejected, and 'cv min ent' means that the samples with entropy lower than the minimum entropy for CV set samples will be rejected. However, no performance gains are obtained from above methods.

Table 4.13 Uncon GANs scheme (prior mode): performance of retrained acoustic model with increased augmented data quantity

Data	Hours	%Frame level acc		%PER
		Training set	CV set	
Uncon GANs + Original	6	73.06	49.99	25.46
Uncon GANs + Original	30	81.94	49.29	26.02

In the last experiment, the quantity of the augmented data is increased from 3 hours to 27 hours. The corresponding results are shown in table 4.13. It can be seen that increasing the quantity of generated data actually degrades the model performance in terms of PER. After using more generated data, the model's accuracy for training set increases dramatically while the accuracy for CV set decreases significantly. This mode is similar to the problem happened in CGANs experiment in section 4.4. The incorrect pattern variations are created in generated samples which makes them easy to be classified. However, the corresponding decision boundaries learned from these samples cannot be used for the real data. This demonstrates the unreliability of GANs for data augmentation.

Chapter 5

Conclusion and future work

5.1 Conclusion

In this thesis, a generative adversarial nets (GAN) based speech data augmentation method was proposed. Several GANs architectures were designed for frame level acoustic data generation. A GANs array system was also proposed in this work, where separate GANs were trained for each phonetic units. In addition, two data generation frameworks were developed based on conditional GANs array and unconditional GANs array separately. A thorough investigation has been made on proposed methods. The experimental results in chapter 4 show that the designed GANs structures have the ability to learn the distribution of the real acoustic data. However, some strange variations are also created in generated samples, which may not exist in real speech features. Thus, the data synthesised by GANs fail to reduce the phone error rate of the acoustic model in ASR task. Although the desired performance gains on TIMIT dataset have not been achieved, we still believe that GANs have the potential for successful speech data augmentation in future work.

5.2 Future work

Firstly, a larger speech corpus, such as Aurora4 [36], can be used in future work. The speech dataset used in this work is TIMIT, which contains only 3 hours training data. When training GANs, the training set has to be further split into 48 subsets because each phone has a separate GAN. As a result, the quantity of the training data for each GAN is extremely small, and estimating a reliable generator cannot be ensured.

Secondly, different acoustic model architectures can be tested in further work. In current work, we only use a simple DNN structure with fixed hyperparameter settings. The depth

(number of hidden layers) and width (number of hidden units) of the network can be further adjusted. Moreover, the other acoustic structures, such as RNN, LSTM and VDCNN, can also be used for testing this data augmentation method. Certain acoustic model configurations may be less sensitive to the unnatural variations created by GANs.

Last but not least, a more advanced method needs to be developed to measure the quality of the generated data. A major factor that causes the failure in this project is that we can not locate the problem of the generated data precisely, which makes it hard to figure out the solutions.

References

[1] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[2] Frank Seide, Gang Li, and Dong Yu. Conversational speech transcription using context-dependent deep neural networks. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, (August):437–440, 2011.

[3] Frank Seide, Gang Li, Xie Chen, and Dong Yu. Feature engineering in context-dependent deep neural networks for conversational speech transcription. *2011 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2011, Proceedings*, pages 24–29, 2011.

[4] Mark J F Gales, Kate M . Knill, Anton Ragni, and Shakti P . Rath. Speech recognition and keyword spotting for low resource languages: Babel project research at CUED. *Spoken Language Technologies for Under-Resourced Languages (SLTU)*, (May):14–16, 2014.

[5] Jinyu Li, Dong Yu, Jui Ting Huang, and Yifan Gong. Improving wideband speech recognition using mixed-bandwidth training data in CD-DNN-HMM. *2012 IEEE Workshop on Spoken Language Technology, SLT 2012 - Proceedings*, pages 131–136, 2012.

[6] Patrice Y Simard, Dave Steinkraus, and John C Platt. Best practices for convolutional neural networks applied to visual document analysis. In *null*, page 958. IEEE, 2003.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[8] Anton Ragni, Kate M. Knill, Shakti P. Rath, and Mark J.F. Gales. Data augmentation for low resource languages. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, (September):810–814, 2014.

[9] Xiaodong Cui, V Goel, and B Kingsbury. Data augmentation for deep neural network acoustic modeling. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 23(9):1469–1477, 2015.

[10] Tom Ko, Vijayaditya Peddinti, Daniel Povey, Michael L Seltzer, and Sanjeev Khudanpur. A study on data augmentation of reverberant speech for robust speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on. IEEE*, pages 5220–5224, 2017.

[11] Chanwoo Kim, Ananya Misra, Kean Chin, Thad Hughes, Arun Narayanan, Tara Sainath, and Michiel Bacchiani. Generation of large-scale simulated utterances in virtual rooms to train deep-neural networks for far-field speech recognition in Google Home. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2017-Augus:379–383, 2017.

[12] Navdeep Jaitly and Geoffrey E Hinton. Vocal tract length perturbation improves speech recognition. In *Proc. ICML Workshop on Deep Learning for Audio, Speech and Language*, volume 117, 2013.

[13] Naoyuki Kanda, Ryu Takeda, and Yasunari Obuchi. Elastic spectral distortion for low resource speech recognition with deep neural networks. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 309–314. IEEE, 2013.

[14] Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. Audio augmentation for speech recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[16] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5967–5976. IEEE, 2017.

[17] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[18] T. Kaneko, H. Kameoka, N. Hojo, Y. Ijima, K. Hiramatsu, and K. Kashino. Generative adversarial network-based postfilter for statistical parametric speech synthesis. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4910–4914, March 2017.

[19] Yuki Saito, Shinnosuke Takamichi, Hiroshi Saruwatari, Yuki Saito, Shinnosuke Takamichi, and Hiroshi Saruwatari. Statistical parametric speech synthesis incorporating generative adversarial networks. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 26(1):84–96, 2018.

[20] Chin-Cheng Hsu, Hsin-Te Hwang, Yi-Chiao Wu, Yu Tsao, and Hsin-Min Wang. Voice conversion from unaligned corpora using variational autoencoding wasserstein generative adversarial networks. *arXiv preprint arXiv:1704.00849*, 2017.

[21] Takuhiro Kaneko, Hirokazu Kameoka, Nobukatsu Hojo, Yusuke Ijima, Kaoru Hiramatsu, and Kunio Kashino. Generative adversarial network-based postfilter for statistical parametric speech synthesis. In *Proc. ICASSP*, volume 2017, pages 4910–4914, 2017.

[22] Santiago Pascual, Antonio Bonafonte, and Joan Serrà. SEGAN: Speech enhancement generative adversarial network. *Proc. Interspeech 2017*, pages 3642–3646, 2017.

[23] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*, 2017.

[24] Karol Kurach, Mario Lucic, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. The GAN landscape: losses, architectures, regularization, and normalization. *arXiv preprint arXiv:1807.04720*, 2018.

[25] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[26] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein GANs. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[27] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[28] Abdel-rahman Mohamed, Geoffrey Hinton, and Gerald Penn. Understanding how deep belief networks perform acoustic modelling. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 4273–4276. IEEE, 2012.

[29] S. J. Young, J. J. Odell, and P. C. Woodland. Tree-based state tying for high accuracy acoustic modelling. *Proceedings of the workshop on Human Language Technology - HLT '94*, page 307, 1994.

[30] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

[31] John S Garofolo, Lori F Lamel, William M Fisher, Jonathan G Fiscus, David S Pallett, Nancy L Dahlgren, and Victor Zue. TIMIT acoustic-phonetic continuous speech corpus. *Linguistic Data Consortium, Philadelphia*.

[32] Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, Anton Ragniand, Valtcho Valtchev, Phil Woodland, and Chao Zhang. The HTK book (for HTK version 3.5). *Cambridge university engineering department*, 3:175, 2015.

[33] Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration, 2017.

[34] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training GANs. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2234–2242. Curran Associates, Inc., 2016.

[35] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[36] David Pearce and J Picone. Aurora working group: DSR front end LVCSR evaluation AU/384/02. *Inst. for Signal & Inform. Process., Mississippi State Univ., Tech. Rep*, 2002.