Applications of Deep Learning to Audio Generation

Yuanjun Zhao, *Student Member, IEEE,* Xianjun Xia, *Student Member, IEEE,* and Roberto Togneri *Senior Member, IEEE*

Abstract

In the recent past years, deep learning based machine learning systems have demonstrated remarkable success for a wide range of learning tasks in multiple domains such as computer vision, speech recognition and other pattern recognition based applications. The purpose of this article is to contribute a timely review and introduction of state-of-the-art deep learning techniques and their effectiveness in speech/acoustic signal processing. Thorough investigations of various deep learning architectures are provided under the categories of discriminative and generative algorithms, including the up-to-date Generative Adversarial Networks (GANs) as an integrated model. A comprehensive overview of applications in audio generation is highlighted. Based on understandings from these approaches, we discuss how deep learning methods can benefit the field of speech/acoustic signal synthesis and the potential issues that need to be addressed for prospective real-world scenarios. We hope this survey provides a valuable reference for practitioners seeking to innovate in the usage of deep learning approaches for speech/acoustic signal generation.

Digital Object Identifier 10.1109/MCAS.2019.2945210
Date of current version: 19 November 2019

I. Introduction

Since the phonograph was invented in 1877 by Edison, different types of acoustic sounds like voice, music and birdsong can be preserved as audio recordings [1]. After decades of exploration and analysis, essential properties of various acoustic signals have been revealed for a deep understanding of the mechanism for production, propagation and perception. The diverse and complex information contained in acoustic signals is useful for spatial location, determining the species of a sound source, identity of a speaker and the message being conveyed [2]. Ever since the advantages of digital signal processing and machine learning technology were consolidated into acoustic signal processing systems, the effectiveness of algorithms that can describe, categorize and interpret all manner of sounds has been improved significantly. Recently, the focus of research in machine learning algorithms has moved onto more practical and emerging fields like speech/speaker recognition, speech synthesis and acoustic event/scene recognition. In this paper, a thorough survey of state-of-the-art deep learning algorithms applied to Speech Synthesis (SS) and Voice Conversion (VC) will be provided. Other types of acoustic signals like music and singing will also be included. However the analysis can be used for reference to any form of audio generating system or application.

The task of speech synthesis is to build natural-sounding synthetic voices with either knowledge-based or data-based approaches [3]. A typical text-to-speech (TTS) system is shown in Fig. 1. In the front-end, the raw texts are firstly converted into the equivalent of written-out words for the task of text normalization. Then the phonetic transcriptions are assigned to each word and the texts are marked into prosodic units. This step is usually called text-to-phoneme conversion. The output of the front-end is the symbolic linguistic representation. The back-end of the TTS system is referred to as the synthesizer. The task of the synthesizer is to convert the symbolic linguistic representation into sound.

Conventional speech synthesis technology are diphone synthesis, concatenative speech synthesis and statistical parametric speech synthesis (SPSS). Unit selection based concatenative techniques have been the main approaches to speech synthesis, of which the quality of the generated speech depends on the available corpora. Suitable sub-word units are automatically chosen from selected corpora of natural speech [4]. Unit selection based toolkits, like the open-source multilingual TTS synthesis platform MaryTTS [5], have been embedded in commercial applications and can bring synthetic speech with a high level of quality. In contrast to retain unmodified speech components in unit selection based methods, statistical parametric speech synthesis systems use parametric models to generate universal descriptions of speech subsets with similar sounding segments. Specifically, the speech are described by models using parameters instead of stored exemplars. These parameters are represented by statistics such as means and variances of probability density functions found in the training data. One of the most popular statistical parametric synthesis techniques is the Hidden Markov Model (HMM) synthesis, which allows more variations on the speech data by statistically modeling and generating the speech parameters of a speech unit with the maximum likelihood criterion based HMMs [6]. An annual challenge named the Blizzard Challenge[1] is held to compare research techniques in corpus-based speech synthesizers, which advances the better understanding and exploration of effective speech synthesis methods.

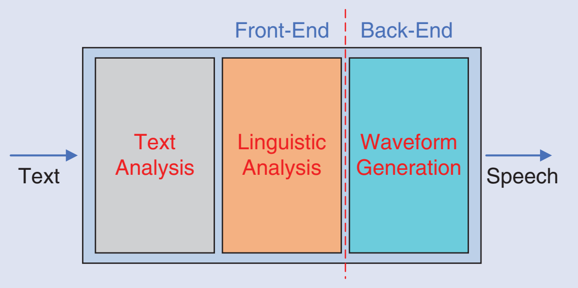


Figure 1. Overview of a typical TTS system.

Compared to the speech synthesis, the voice conversion is a different technique to create high-quality synthetic speech. The task of voice conversion is to convert a sentence spoken by an original speaker to a resultant utterance with the same sentence as before. The resultant utterance sounds as being spoken from a different speaker. The information of the timbre and the prosody of the original and target speakers are considered in a voice conversion system. These two features are generally associated with the dynamic spectral envelope of the voice signal, pitch/energy contours and rhythmic distribution of phonemes [7]. A diagram of the typical voice conversion system is given in Fig. 2. In the training phase, the speech signal from the source speakers and the target speakers are fed into the VC system. After the speech analysis and the mapping feature computation, the raw speech signals are converted into a suitable representation for the further processing and modification. The speech segments of the training signals are aligned with respect to time. The conversion function is then trained on these aligned mapping features. In the conversion phase, the mapping features of a new input utterance are also extracted and then converted by the

[1]http://www.festvox.org/blizzard/blizzard2017.html.

Y. Zhao, X. Xia and R. Togneri are with the School of Electrical, Electronics and Computer Engineering, The University of Western Australia, Perth, WA 6009, Australia (e-mail: yuanjun.zhao@research.uwa.edu.au; xianjun.xia@research.uwa.edu.au; roberto.togneri@uwa.edu.au).

trained conversion function. After the conversion step, the speech features are computed from the converted features and used for synthesizing the converted utterance waveform.

Since the original formulation of the voice conversion problem was proposed in 1985, many VC techniques have been introduced [8]. The most popular VC approach is the STRAIGHT (Speech Transformation and Representation using Adaptive Interpolation of weiGHTed spectrum), which utilizes a combination of spectral representations and voice conversion techniques [9]. With this VC system, the spectral, acoustical and rhythmic parameters can be manipulated easily. Via suitable modification of the speech waveform, voice conversion can convert non-/para-linguistic information as expected [10]. Table I lists several existing frameworks that are frequently applied for producing speech and acoustic signals. Details of speech synthesis and voice conversion techniques can be found in the overview papers and books of [3], [6], [7], [11].

In recent years, the research on anti-spoofing countermeasures to detect deliberate and unknown attacks by artificial (e.g. synthesized) speech is becoming more active [12]. Compared to other state-of-the-art

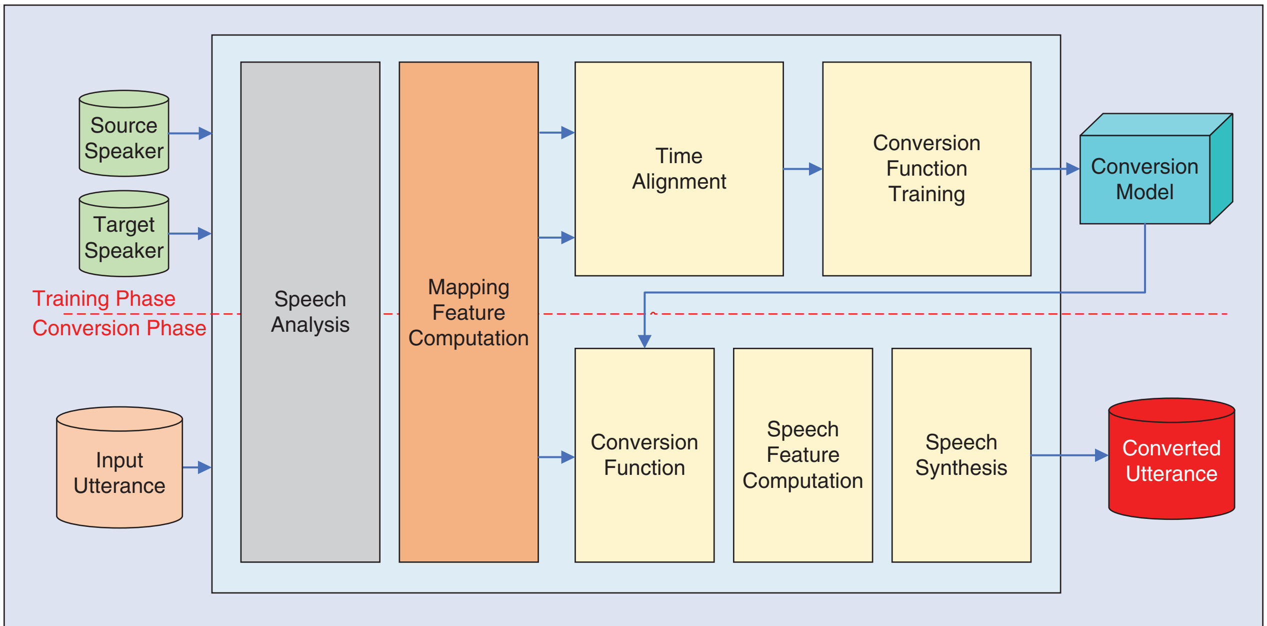


Figure 2. A typical voice conversion system.

Table I.
Frameworks providing speech generation tools and methods.

Framework	Language	URL
MaryTTS	Java	http://mary.dfki.de/
eSpeak	C	http://espeak.sourceforge.net/
HTS	C	http://hts.sp.nitech.ac.jp/
Festvox/Festival	C++	http://festvox.org/
Idlak	C++	https://github.com/idlak/idlak
Merlin	Python	https://github.com/CSTR-Edinburgh/merlin
Ossian	Python	https://github.com/CSTR-Edinburgh/Ossian
GAN TTS	Python	https://github.com/r9y9/gantts
Sprocket	Python	https://github.com/k2kobayashi/sprocket
STRAIGHT	Matlab	https://github.com/HidekiKawahara/legacy_STRAIGHT
WORLD	C++/Matlab/Python	https://github.com/mmorise/World

anti-spoofing methods like [13], [14], deep learning based approaches provide improved capability for protecting automatic speaker verification (ASV) systems from speaker spoofing via synthetic speech [15].

Besides the acoustic signals produced by human vocal systems (e.g. speech and song), another ultimate acoustic 'language' is that of music. To compose a natural-sounding music, several types of musical content need to be generated, such as the melody, polyphony, counterpoint, chords and lead sheet. By a combination of all these musical components, music, which is well-known for its emotional effect, can act as a form of expression and preciseness [16]. More details about music generation can be found in refs. [17], [18].

Machine learning has been the dominating technology in a variety of signal processing applications. In processing raw data from the natural information source, conventional machine learning approaches are limited by the complicated requirements of feature extraction [19]. Deep learning (DL), as a subfield of machine learning, involves a hierarchical computing framework in which multiple layers of learning algorithms are stacked in a specific order. With an approximation of nonlinear functions, deep learning algorithms are applied to achieve abstract representations and learn feature vectors from the original data. We have seen impressive progress of deep learning over the last decade, to many engineering and science application areas ranging from web searches, online content filtering and recommendations systems to computer vision, speech recognition and other machine intelligence tasks [20].

In Fig. 3 we depict the historical development of the deep learning algorithms discussed in this paper. The artificial neural network (ANN) [21] originated in the 1940s and led to the first wave of artificial intelligence (AI) algorithms with the creation of the single-layer perceptron (SLP) and the multi-layer perceptron (MLP) [22], [23]. To build a standard neural network (NN), varying amounts of neurons are used to yield real-valued activations and, by adjusting the weights and biases, the NNs can behave as expected in specific tasks like computer vision and speech recognition [24]. The development of ANNs stagnated because of the incapability of processing the exclusive-or circuit and computing devices with low processing capacity. The next wave of research boomed in the 1980s when the backpropagation algorithm (BP) [25] was proposed. As an efficient gradient descent algorithm, BP effectively accelerated the training of ANNs. However, in the late 1990s, ANNs and the BP algorithm were largely abandoned by the community. It was generally believed that backpropagation would get trapped in poor local minima and the average error could not be reduced any more. In addition to that, insufficient labeled training data can cause the problem of over-fitting [19].

In 2006, which is considered the first year of deep learning, several breakthroughs regarding new network architectures and training methods revived the interest in neural networks. In [26], a new layer-wise-greedy-learning based method was proposed for training very deep neural networks (DNN). Hidden layers in a network are pre-trained one layer at a time using the unsupervised learning approach and this considerably helps to accelerate subsequent supervised learning through the backpropagation algorithm [19], [27]. Also in 2006, a Convolutional Neural Network (CNN) trained by BP set a new record of 0.39% on the handwriting digits database MNIST [28], which was a significant progress in the performance since the classical prototype LeNet-5 [29].

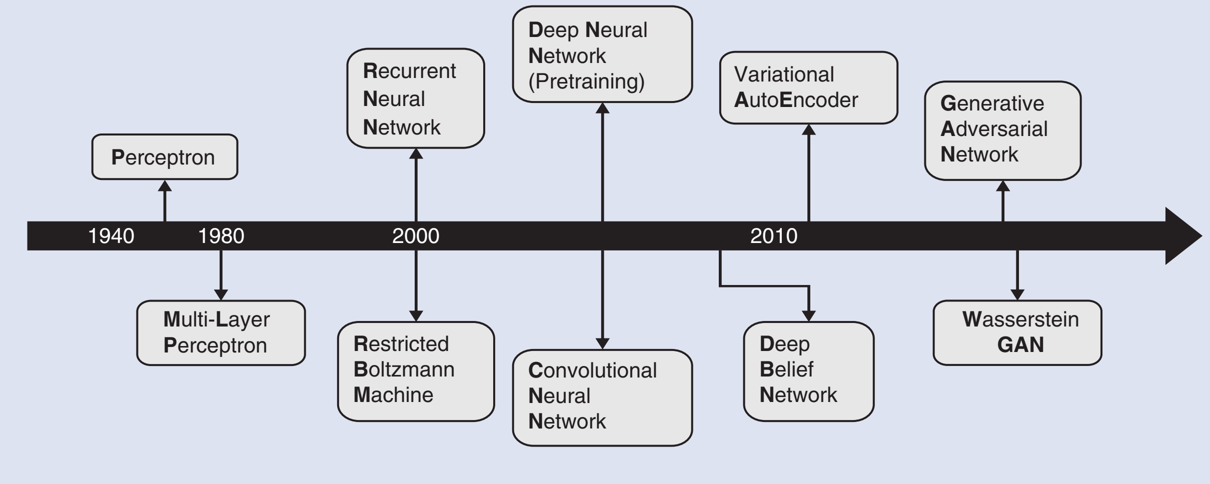


Figure 3. A timeline of deep learning algorithms introduced.

After the ImageNet challenge[2] in 2012, CNNs began to rule the field of image classification [30]. New innovations have been made every year and the research in CNNs has proliferated. The AlexNet, as the winner of 2012 challenge, was considered to be the break-through CNN variant with a top-5 test error rate (a score used to indicate if the target label is one of the top 5 predictions with the highest probabilities) of only 17.0% on ILSVRC-2010 test set [31]. In 2013, the ZF Net gave a better accuracy based on a slight modification of the AlexNet model and proposed a novel feature maps visualization [32]. Another AlexNet type architecture based CNN is the VGG Net and it was frequently applied in the community because of its appealing uniform network architecture [33]. Among all the submissions, the GoogLeNet proposed by Google won the 2014 ImageNet challenge [34]. In GoogLeNet, the core innovation is a repeated inception module, with which the number of the parameters can be significantly reduced. The next popular CNN is the Microsoft ResNet (residual network) which was the winner of the 2015 ImageNet challenge [35]. The residual block was proposed in this network to guarantee the gradients computed can be directly used into a more effective weights updating process.

The Autoencoders (AEs) are another popular type of unsupervised pre-training deep feedforward neural network (FNN) [36]. Different variations of AEs have been introduced to enhance the ability of extracting informative representations, including the denoising autoencoder [37], [38], the sparse autoencoder [39], the variational autoencoder (VAE) [40] and the contractive autoencoder (CAE) [41].

In speech/acoustic recognition and other pattern recognition problems, generative and discriminative models are the two main approaches. The generative models attempt to provide the distribution of data or the joint distribution of data and the corresponding targets, while the discriminative models directly predict the distribution of targets conditioned on the data [42]. The Generative Adversarial Networks (GANs) proposed by Goodfellow can be considered as a hybrid model consisting of both the generative and the discriminative models [43]. In the GAN's framework, a generator G and a discriminator D are trained simultaneously to capture the data distribution and to estimate the probability that an output is from the training data.

With the rapid development of efficient computation techniques and the growth of computing power, implementing large-scale deep learning approaches with computers is no longer a fantasy. The advent of fast Graphic Processing Units (GPUs) and the availability of huge amounts of data significantly expedite the training and fine-tuning of deep learning models. For speech synthesis, deep learning aims to remove the restrictions of the conventional method in statistical parametric synthesis based on Gaussian-Hidden Markov Models and other classical models [44]. A properly designed acoustic model plays an important role in HMM-based statistical parametric speech synthesis systems. Generated speech from shallow-structured HMM-based synthesizers have been generally known for poor fidelity compared with natural speech. To address this problem, deep learning approaches are adopted to offset this deficiency. Generative models like Restricted Boltzmann Machines (RBMs) and Deep Belief Networks (DBNs) have displaced traditional Gaussian models and concrete improvements have been obtained with regard to the quality of speech [45], [46]. In contrast to the generative deep models, discriminative models of the DNN can also be applied for performing speech synthesis. In [47], a DNN-based approach was proposed to predict spectral and excitation parameters. A better quality of synthetic speech was achieved with a similar number of parameters to the HMM-based synthesizer. For voice conversion, deep learning is usually used for learning the associations between the spectral mapping features, which allows modification of speech properties in a VC system. In [48], a two-layer feedback neural network based on bidirectional associative memory was reformulated to model the spectral envelope space of the speech signal. Experimental results showed that the proposed method has a better modeling ability than the traditional use of Gaussians with diagonal covariance. In [49], a stacked joint-autoencoder was applied to construct a regression function, which was used in a VC task. This method demonstrated features that do not suffer from the averaging effect inherent with the back-propagation algorithm.

As a fast growing domain, deep learning has recently been applied for musical audio synthesis and music composition. A fundamental motivation is to learn a representation of diverse musical styles or musical content based on several widely available databases. Generally, research works focus on four main dimensions of music generation: the objective (e.g., a melody or a sequence of chords), the representation (e.g., raw musical signal or spectral features), the architecture (e.g., DNNs, CNNs or RNNs) and the learning strategy [17]. In [50], an autoencoder based synthesizer was proposed for compressing and reconstructing magnitude short time Fourier transform frames. This synthesizer was re-trained for music synthesis applications and several

[2]http://www.image-net.org/.

samples were generated.[3] In [51], an interactive musical audio synthesis system[4] based on ANNs was introduced. In this system, frames of low-level features were learned by an ANN and a high level representation of the musical audio was learned through an autoencoder.

The purpose of this paper is to offer a timely overview of the applications of deep learning approaches to audio generation problems, especially speech generation. Deep learning algorithms involved are categorized as either discriminative or generative methods, taking GANs as a distinctive hybrid method. The background on the different deep learning architectures and up-to-date applications are provided to readers in this area. The rest of the paper is organized as follows. In Section II we start by reviewing recent deep discriminative algorithms and the remarkable progress in speech synthesis and acoustic signals generation. While the same understanding and analysis are shown for deep generative algorithms in Section III. Specific discussion of GANs and their variations are reported in Section IV. The reasons why deep learning can be beneficial for pattern recognition problems and issues to be studied further are given in Section V. We conclude the paper in Section VI.

[3]http://soundcloud.com/ann_synth
[4]https://github.com/woodshop/deepAutoController

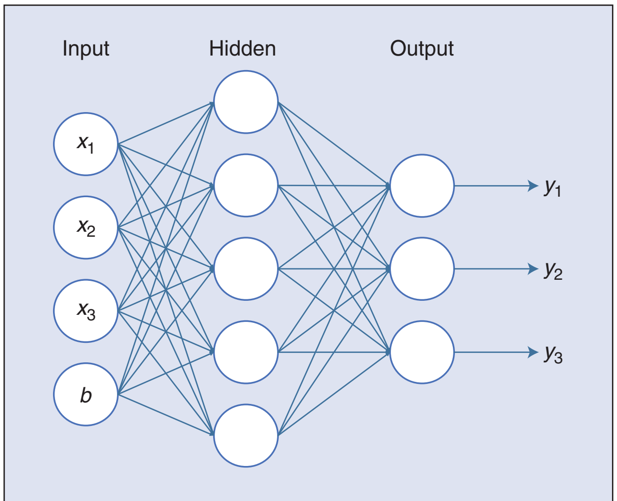


Figure 4. A single (one hidden) layer MLP.

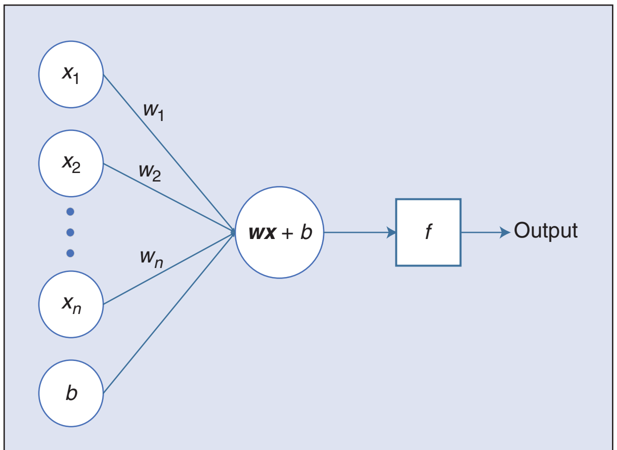


Figure 5. Computation procedure at each node.

II. Discriminative Algorithms

The distinction between discriminative and generative models is the probability distribution modeled. Generally, an output variable $\mathbf{y}$ needs to be estimated by a standard pattern recognition model given an input variable $\mathbf{x}$. A deep discriminative algorithm, like a DNN, utilizes multi-layer hierarchical architectures to directly compute the probability of $\mathbf{y}$ given an $\mathbf{x}$, i.e. to estimate $p(\mathbf{y}|\mathbf{x})$. The most common discriminative models used in machine learning include logistic/linear regression, support vector machines (SVMs), random forests and neural networks. With different structural elements, deep discriminative models contain various implementations in terms of tasks and functions. In this paper we only focus on the discriminative models with deep hierarchical architectures such as the MLP, CNNs and RNNs.

A. Deep Discriminative Architectures

1) Multi-Layer Perceptron

A multi-layer perceptron, as a type of feedforward artificial neural network, consists of at least three layers of computing units (also known as "nodes" or "neurons"). A diagram of a single layer MLP is shown in Fig. 4. Except for an input and an output layer, one or more hidden layers can be inserted as required. Each node in a hidden layer or output layer contains a nonlinear activation function depending on how the network has been configured. The computation process of the lth layer is shown in Fig. 5 and is described by:

$$y^l = f^l(\mathbf{w}^l\mathbf{x}^l + b^l) \tag{1}$$

where $\mathbf{x}$ are inputs to the network while weights and biases are denoted as $\mathbf{w}$ and b, respectively. The activation function f is nonlinear to enhance the capacity of the modeling representation, especially beneficial for nonlinear classification problems. Like in [26], the backpropagation algorithm is employed to update weights and biases of all the layers with the loss calculated. Specific objective functions are set to estimate the distance between the prediction and the actual value. With trained network parameters, we can use the MLP to predict the output in practical pattern recognition problems. More details of the backpropagation algorithm can be found in [52].

2) Convolutional Neural Networks

Convolutional Neural Networks (CNNs) have been usually exploited in the fields of image classification, object detection, face recognition and speech processing. Compared with general DNNs, matrix multiplication is replaced by the convolution to dramatically reduce the amount of weights in modeling. This makes CNNs a powerful tool to relieve the memory curse in image processing related tasks since the large number of pixels in a single picture always leads to a similarly large number network parameters. Consequently, the complexity of the network can be decreased. Another advantage of CNNs is that the images, as raw inputs, can be directly imported to the network without the feature extraction procedure. In addition, with a workstation equipped with GPUs, training a CNN is more efficient.

Generally, a basic CNN includes three types of layers: convolutional, pooling and fully-connected layers. For example, Fig. 6 demonstrates the famous LeNet-5. In this network, there are three convolutional layers, two pooling layers and two fully-connected layers. For each convolutional layer, there are several convolution kernels to compute feature maps from the previous layer. Distinct feature representations of original inputs can be learned by the convolution procedure and transferred to subsequent layers. A region of neighboring neurons are mapped to single neuron on the subsequent layer and this region is referred to as the receptive field of this single neuron. To achieve a feature map, the input is convolved by a trainable kernel and the result is applied with an element-wise nonlinear activation function. Due to the property of parameter sharing in CNNs, the kernels are shared by all pixel areas of the input. Finally, all the feature maps are obtained by utilizing a variety of convolutional kernels. The pooling layers are introduced to reduce the overall size of the signal to avoid over-fitting problems caused by high dimensional inputs. As in classical neural networks like the MLP, neurons in all types of layers compute the dot product between the input vectors and the weights, to which biases are added. Then the weighted sum is passed through a nonlinear activation function to obtain the output vectors. Denoting the pooling function as pool$(\cdot)$, for each feature map we have:

$$y^l_{i,j,k} = \mathrm{pool}(f^l_{m,n,k}(\mathbf{w}^{l^T}_k \mathbf{x}^l_{m,n} + b^l_k)), \forall (m,n) \in \Re_{ij} \qquad (2)$$

where $\Re_{ij}$ is a local neighborhood around location (i,j) in the kth feature map of the lth layer [53]. Note that the weights and the bias are shared by the neurons belonging to the same layer. From (1) and (2) it is obvious that the changes between the classical NN architectures and the CNNs are only in the layer category and configuration.

High-level feature representations can be extracted gradually by stacking several convolutional and pooling layers. The one or more fully-connected layers added before the output layer aim to learn nonlinear combinations of the high-level features passed from the previous layers and convey the higher-level representation to output layers. Fully-connected layers can be optionally replaced by convolution layers, of which the size of the convolution kernels is 1×1 [54]. The last layer in a CNN is known as the output layer, which is configured based on the type of task that the network needs to

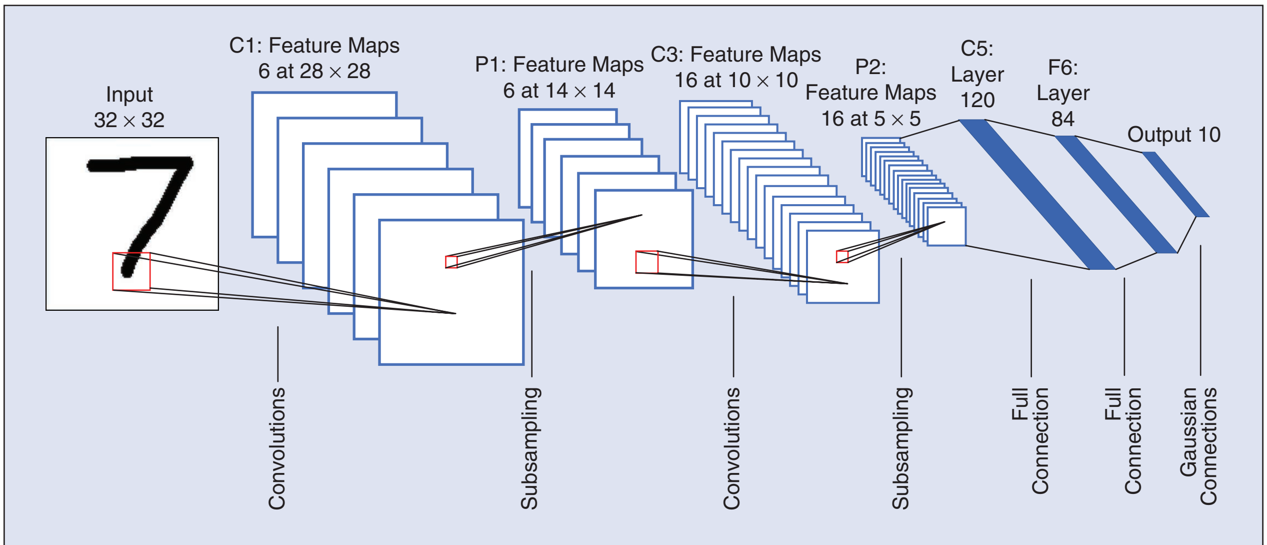


Figure 6. Architecture of LeNet-5 [29].

perform. The softmax operator is a commonly used output layer for classification tasks, of which the resultant vectors present a classification probability distribution summing to 1. For training a CNN, the goal is to minimize the objective function defined for the task. With an appropriate optimizer, the best fitting set of parameters (the weights and bias terms) can be solved.

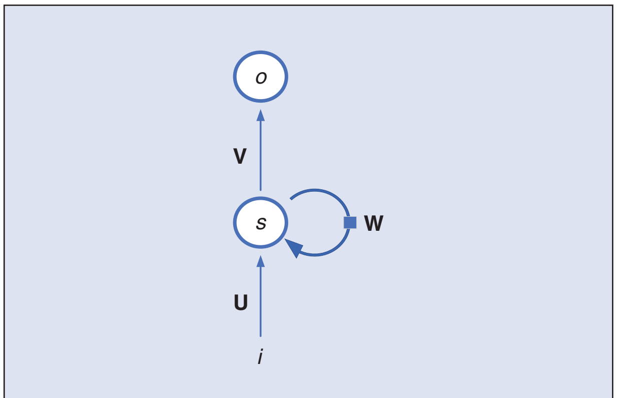


Figure 7. A simple computation unit of RNNs.

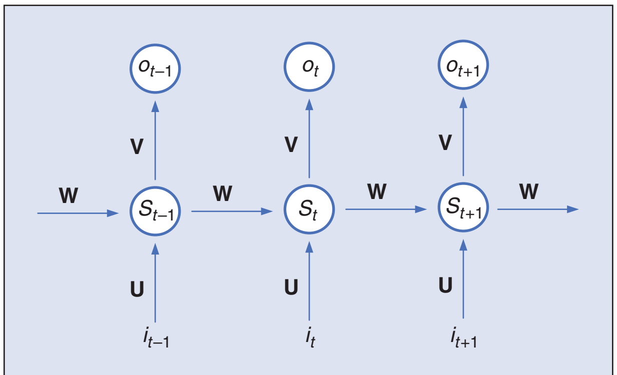


Figure 8. Unfolded structures of RNNs.

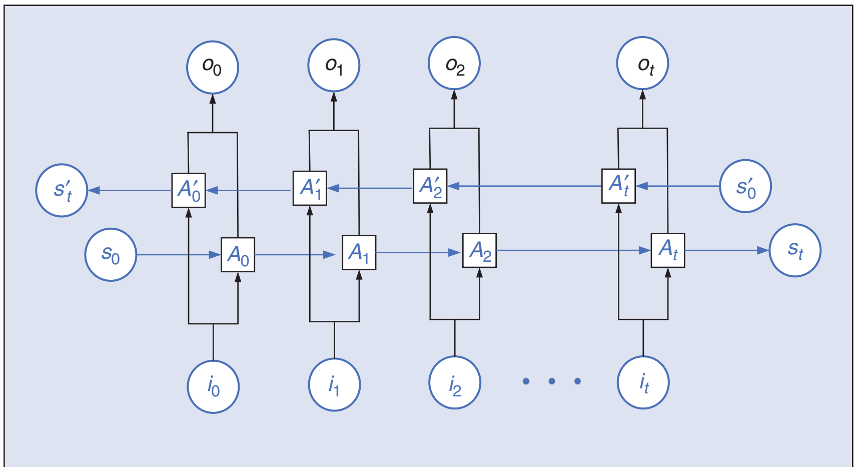


Figure 9. A simple bi-directional RNN.

3) Recurrent Neural Networks

Recurrent neural networks (RNNs) are developed to process pattern recognition tasks of sequential data like text, genomes, spoken words or numerical times series data from real-world commerce [55]. Compared to the MLP and CNNs, RNNs explore the temporal information of inputs by taking time and sequence into account. To be specific, in the case of feedforward networks as shown in Fig. 4, samples fed to the network are transformed straight into an output via supervised learning. While in RNNs, the output at time step $t-1$ affects the future decision one moment later at time step t by the feedback loops. This unique component makes RNNs have "memory," which is indispensable for language processing and sequence signal processing.

Fig. 7 indicates the basic schematic diagram in RNNs which is an integration of three kinds of layers: the input layer, the hidden layer and the output layer. The i denotes the input vector and o denotes the output vector. The hidden neurons are denoted as the vector s while $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{W}$ are the weights matrix. Note that $\mathbf{W}$ contains the information of the previous time steps for this neuron. After unfolding, the diagram of this basic RNN unit is represented as in Fig. 8. From Fig. 8, it is explicitly shown that the value of hidden layers depend on both the inputs and the hidden neurons of the previous time steps. The output and hidden layers can be calculated as below:

$$\begin{aligned} o_t &= g(Vs_t) \\ s_t &= f(Ui_t + Ws_{t-1}) \end{aligned} \tag{3}$$

For language processing, it is necessary to model not only previous words or phrases in a sentence, but also subsequent linguistic units at time step $t+1$. Bi-directional RNNs are proposed to predict or label each element of a sequence with its past and future contexts. Outputs of two RNNs can be concatenated to process the sequence from opposite directions. Fig. 9 is a graph demonstrating an example of such bi-directional RNNs.

In practical applications, it is found that the performance of the RNNs degrades when data sequences are long. Exploding gradients and vanishing gradient problems result in an uncontrolled propagation of the gradients in the training procedure. Presetting a threshold is an effective way to avoid the exploding gradients issue, but the vanishing gradient is tricky to restrain. Three methods have been adopted to optimize the RNN training process. One approach is to initialize the weights of RNNs in a proper manner to eliminate vanishing gradient issue. Or we can replace common activation functions (e.g. sigmoid or tanh) with rectified linear units (ReLU) [56], [57]. Deploying improved RNN structures, such as the Long Short Term Memory (LSTM) network [55], [58], [59] or the Gated Recurrent Unit (GRU) [60]–[62], has become the preferred solution.

In the following, selected applications of deep discriminative algorithms in speech and audio signal generation are reviewed.

B. Applications

1) Speech Generation

Deep discriminative models based methods are actively being investigated for acoustic modeling and adaption, feature learning and waveform generation in speech generation. For acoustic modeling, DNNs are used to replace the Gaussian mixture models (GMMs) for the evaluation between frames of acoustic observations and HMM states [63]. In [64], a mixture density network (MDN) was used as an acoustic model in statistical parametric speech synthesis. MDNs give full probability density functions over real-valued output features conditioned on the corresponding input features. This approach addressed the restrictions of the lack of ability to predict variances and the unimodal nature of the objective functions in statistical parametric speech synthesis. DNNs can also be used to replace the decision trees in HMM-based statistical parametric speech synthesis applications [47]. This alternative scheme assisted to break the limitations in the conventional decision tree-clustered context-dependent HMM-based approach, such as the inefficient expression of complex context dependencies and fragmented training data. In [65], DNN-based expressive speech synthesis was comprehensively investigated, especially for multiple emotions representations. In [66], a proof-of-concept system for speech texture synthesis and voice conversion was introduced. A cost function with respect to the input waveform samples was optimized in this system. Realistic speech babble and utterance in a different voice can be reconstructed.

LSTM based networks are also demonstrated for their surprising performance in statistical parametric speech synthesis systems. In [67], a simplified LSTM architecture was proposed and can achieve similar performance in speech synthesis with fewer parameters than vanilla LSTM. Another LSTM RNN was introduced in [68], which utilized data from multiple languages and speakers. Experimental results showed that this multilingual statistical parametric speech synthesis system can generate speech in multiple languages from a signal model and the naturalness is satisfactory. SampleRNN, as a state-of-the-art RNN based model, was proposed in [69] for unconditional audio generation. This model is able to capture underlying sources of variations in the temporal sequences over very long time spans. The samples generated by the SampleRNN are preferred by human raters. In [70], RNNs with bi-directional LSTM (BLSTM) cells were adopted to capture the correlations between any two instants in a speech. This hybrid system of DNN and BLSTM-RNN can outperform both the traditional HMM-based and the more recent DNN-based statistical parametric speech synthesis systems with a smoother speech trajectory. The use of the deep BLSTM-RNN was also investigated in [71] for voice conversion. A sequence-based conversion method was proposed to improve the naturalness and the continuity of the converted speech. Deep BLSTM-RNNs were used to model both the frame-wise relationship between source/target voice and the long-range context-dependencies in the acoustic trajectory. The resultant speech showed a better MOS performance than DNN based methods.

2) Other Types Of Audio Signal

Recently, deep neural networks have been applied in music generation to meet the demands of music composition on various application platforms. In [72], an end-to-end melody and arrangement generation framework was proposed. This framework can generate melody tracks with several accompanying tracks played by diverse types of instruments via applying a multi-instrument co-arrangement model. In [73], a set of parallel, tied-weight recurrent networks was trained to model the polyphonic music. Two modified architectures were proposed and combined for the music generation and prediction task. Experimental results[5] demonstrated that the proposed models can reproduce complex rhythms, melodies and counterpoints in some cases. LSTMs were adopted in [74] for the generation of polyphonic music. In this work, a chord LSTM was used to predict a chord progression based on a chord embedding. Then another LSTM was used to generate polyphonic music from the predicted chord progression. The produced music had a clear long-term structure that sounds as harmonic as the ones played by a musician.

In [75], a gated PixelCNN [76] was applied in a singing synthesizer. The harmonic spectral envelope was modeled by the network. In [77], the relationship between

[5]https://www.cs.hmc.edu/ ddjohnson/tied-parallel/

the musical score and its acoustic features was modeled in frames by a DNN. Subjective experimental results demonstrated that the DNN-based singing voice synthesizer outperformed the conventional HMM-based system in terms of naturalness. In listening tests, the deep learning based singing synthesizer can generate a sound quality on-par or exceeding the previous state-of-the-art concatenative methods. The timbre and fundamental frequency carried in the natural songs can be jointly modeled by deep learning approaches, which allows for much faster experimentation because of the superior tools like GPUs [78]. Another interesting application is the conditional rhythm composition [79], in this work a deep network was proposed as a combination of LSTMs and feed forward layers. This architecture was applied to learn long drum sequences from the metrical information and bass lines.

III. Generative Algorithms

For generative algorithms, they provide the joint probability distribution of the input and the output of the deep learning model. In other words, generative models aim to estimate $p(\mathbf{x}, \mathbf{y})$. However, $p(\mathbf{y}|\mathbf{x})$ can also be obtained with Bayes' theorem to indirectly perform pattern classification tasks [42]. Several widely used generative algorithms are Gaussian Mixture Model (GMM), Hidden Markov Model (HMM), Restricted Boltzmann Machine (RBM), Deep belief Network (DBN) and Variational autoencoders (VAEs).

A. Deep Generative Architectures

1) Variational Autoencoder

The variational autoencoder (VAE) [40] is derived from the original autoencoder but with stronger assumptions concerning the distribution of latent variables. In order to understand the VAEs we need to first introduce the autoencoder.

An autoencoder is a type of NN that learns data codings in an unsupervised manner [80]. Dimensionality reduction is the main application of autoencoders due to the capability of representing a set of data [36]. Fig. 10 is a classical schematic structure of an autoencoder with three fully-connected hidden layers.[6] This is identical to the MLP which is built by many single layer perceptrons as a feedforward neural network. Note that for autoencoders, the number of nodes on the output layer is the same as the input layer. Generally, the size of the hidden layers are smaller than that of the input or output layers. This unique type of architecture is designed for compressing inputs to a compact code and uncompressing that code into the outputs. The outputs are expected to be closely matched to the inputs. The two main components of an autoencoder for completing the compress and the uncompress operations are called the encoder and the decoder, respectively. The output layer of the encoder (that is also the input layer of the decoder) is usually referred to as a code, latent variables or a latent representation. In pattern recognition tasks like image processing, the latent representation can be used as extracted features for a further supervised learning. While in unsupervised learning tasks, variations of the autoencoder (such as VAEs) can be used to generate new data from the training inputs. Therefore, autoencoders have emerged as one of the most popular approaches in the field of machine learning in recent years [81]–[85].

Although standard autoencoders have been successfully applied, there are several problems limiting the development of autoencoders. The most fundamental one

[6]The cat image used in Fig. 10 is from the YouTube video: https://www.youtube.com/watch?v=YCaGYUIfdy4

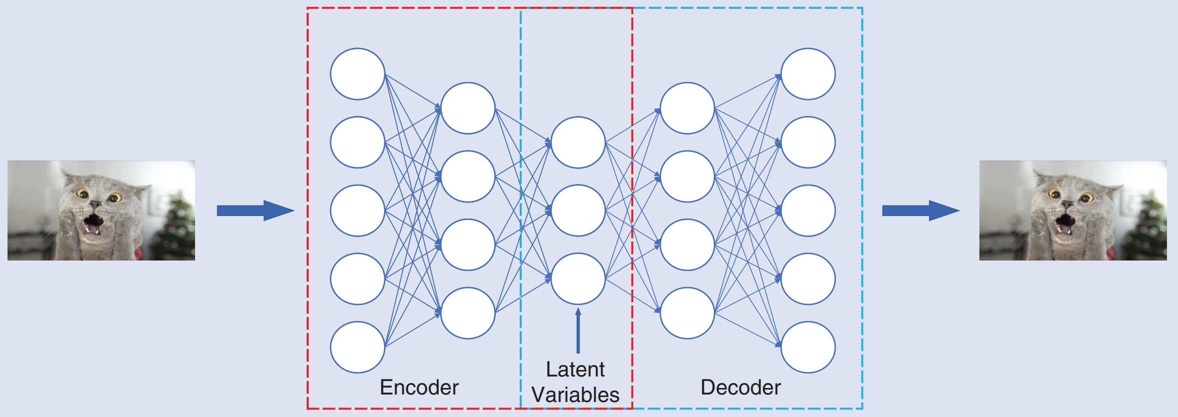


Figure 10. A standard autoencoder with three fully-connected hidden layers.

is that the latent space for generation may not be continuous and easily interpolated. This makes it difficult to build a generative model and produce variations in the input image, caused by a discontinuous region from the latent space.

Compared to the standard autoencoders, VAEs are able to generate new outputs which are similar to the training data in a desired direction. The latent spaces of VAEs are continuous to allow random sampling and interpolation by design. To achieve this unique property, the output of the encoder (i.e. the latent variables) is transformed to two vectors as shown in Fig. 11: one for the means μ and another for the standard deviations σ. These statistical parameters form a vector of random variables $\mathbf{X}$, of which the probability distribution of all elements X_i is normally distributed. The length of this random variable vector is the same as the encoding vector in standard autoencoders. The ith element of μ and σ are the mean and standard deviation of the ith random variable X_i. After sampling, a real-value vector is obtained and passed forward to the decoder for generation. Intuitively, the latent representation of an input is controlled by the mean and standard deviation vector. Consequently, new samples can be created by the decoder with a slightly varied latent encoding.

To achieve ideal latent variables, which are as close as possible to each other while still being distinct, the Kullback-Leibler (KL) divergence is introduced into the loss function. KL divergence is used to measure by how much two probability distributions diverge from each other. A lower KL divergence means the probability distribution of the generated samples are closer to the target distribution. For VAEs, the KL loss in a Gaussian case, as given in (4),

$$D_{KL}(Q \parallel P) = \frac{1}{2}\sum_{i=1}^{n}(\sigma_i^2 + \mu_i^2 - \log(\sigma_i^2) - 1) \qquad (4)$$

is the sum of all the KL divergences between the approximate posterior Q and the standard normal distribution P [40]. Obviously, the minima will be reached when $\mu_i = 0$ and $\sigma_i = 1$. Besides the KL loss, the reconstruction loss (such as the mean square error) is also adopted to maintain the similarity of nearby latent variables.

A VAE can also be extended for supervised learning, namely as the Conditional VAE (CVAE). The CVAE models latent variables and data, both conditioned on some random variables, which allows producing data with specific attributes. More details can be found in [86].

2) Deep Belief Network

The deep belief network (DBN) is a type of generative graphical model and is usually considered as a stack of simple restricted Boltzmann machines (RBMs). The edges between layers of a DBN are both directed and undirected. Hidden units are connected to those of other layers but not with units belonging to the same layers.

To understand DBNs, the definition of RBMs is introduced first. As their name implies, RBMs are derived from Boltzmann machines, which are stochastic recurrent neural networks. The difference is that in an RBM all the neurons are paired from two groups of units: visible and hidden units. In addition, there are no connections between neurons within a group. This is for a more efficient training algorithm, in particular the gradient-based algorithm. Fig. 12 shows an example of an RBM. It is noted that the connections between the visible layer and the hidden layer are bidirectional. The contrastive divergence (CD) algorithm is most often used to train an RBM by optimizing the weight matrix [87]. After training, the hidden layer can perform an exact feature representation of the visible layer and reconstruct the visible layer.

Returning back to the DBN, which is a combination of multiple RBMs, this composition leads to a fast, layer-by-layer unsupervised training procedure. The network architecture of a DBN is shown in Fig. 13. When the first RBM is trained by the CD algorithm, another RBM is stacked, taking the outputs from the hidden layer of the

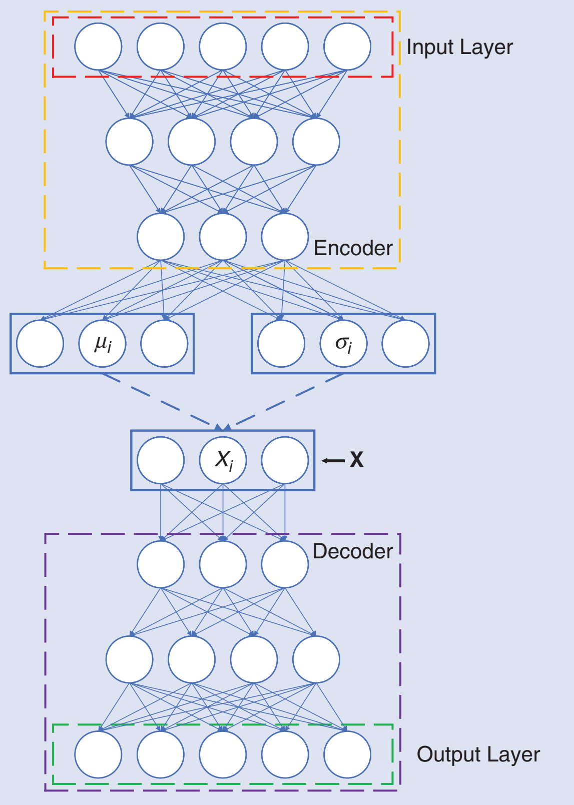


Figure 11. An example VAE configuration.

first RBM as the input. New weights and biases are assigned for the new visible layer to train the new RBM by the CD algorithm again. This process is iterated until reaching a desired stopping criterion.

Note that DBNs can be applied for the purpose of supervised learning by adding a final layer of variables that represent the desired outputs and backpropagating error derivatives. An alternative way is to employ the weights from a trained DBN for the pre-training of an NN for classification tasks.

B. Applications

1) Speech Generation

Recently, generative models have been adopted as powerful tools for speech waveform production. WaveNet [88], [89], proposed by the DeepMind group of Google, has caught the attention of the speech synthesis community. WaveNet is a deep generative model of raw audio waveforms and is able to create a human voice which sounds very natural. It is composed of fully convolutional neural networks, involving various dilation factors. The receptive field is enlarged exponentially with the depth of the network and spans a large number of timesteps. For training, raw recordings are fed as input sequences. Synthetic utterances can be generated by sampling the trained network. Note that for producing meaningful utterances, the network's predictions are conditioned on both the audio samples and the text to be spoken. By conditioning the network on the identity of the speaker, one can use WaveNet to generate the same sentence in different voices. Interestingly, WaveNet can also be applied to model other kinds of acoustic signals, such as music. Now that WaveNet has been embedded in the Google Assistant application, it has received a lot of positive feedback [89].

As an amazing production-quality speech synthesis system, the WaveNet and its variants are embraced by many companies and research labs. Deep Voice, a truly end-to-end neural speech synthesizer employing the WaveNet, was proposed by the Baidu Silicon Valley AI Lab in 2017 [90]–[92]. A variant of WaveNet is implemented for the audio synthesis model in this system with fewer parameters being required. In Deep Voice 2, the Tacotron [93], a text-to-speech synthesis system, was combined with a WaveNet-based spectrogram-to-audio vocoder. Evaluation results demonstrated that high-quality speech can be achieved by this integrated TTS synthesis system. A similar work describes a natural TTS synthesizer Tacotron 2 [94]. This Tacotron 2 was composed of a recurrent network, which was for mapping character embeddings to Mel-scale spectrograms, and a modified WaveNet model, which acted as a vocoder to yield time-domain waveforms from those spectrograms. This model can achieve a mean opinion score (MOS) of 4.53/5.00, which suggested very high-fidelity generated speech.

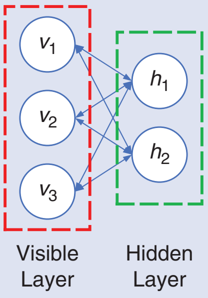


Figure 12. An example of an RBM.

In [45], the RBM and DBN were applied to represent the distribution of the low-level spectral envelopes used for HMM-based parametric speech synthesis. Experimental results showed that both modified methods were able to generate spectral envelope parameter sequences better than the conventional Gaussian-HMM based approach. In [95], a deep autoencoder structure was proposed to extract robust spectral features for statistical parametric speech synthesis systems. By using the autoencoder, low-dimensional features can be compressed from the original high dimensional spectral envelope without degradation. A similar feature extraction procedure is also explored in [83], [96], [97].

For voice conversion, generative algorithms have been applied in many frameworks and systems to improve the naturalness, clarity and speaker individuality. In [49], a regression function constructed by a stacked joint-autoencoder was applied to a voice conversion task. Subjective listening tests were carried out to prove that the proposed approach has a higher quality and similarity than another system integrated with DNNs. In [98], a statistical voice conversion technique with the WaveNet-based waveform generation was introduced. The waveform samples of the converted voice were created by a WaveNet vocoder conditioned on the converted acoustic features. The experimental results showed that a higher conversion accuracy on speaker individuality was achieved with the proposed VC method, compared to the conventional VC techniques.

2) Other Types of Audio Signal

In [50], an autoencoder based music synthesizer was presented. The autoencoder adopted was built by a four layer deep topology and both sigmoid and ReLU activations were used. In [99] and [100], a WaveNet architecture was used to train raw audio models to produce expressive music. An automatic music generation method was proposed based on the combination of the symbolic models and the raw audio models. The musical samples achieved[7] gave a high level of the naturalness and continuity. In another work, a powerful WaveNet-style autoencoder model was proposed to condition an autoregressive decoder for capturing longer term structure without external conditioning [101]. The model can learn a manifold of embeddings, which allows for morphing between instruments and interpolating in timbre. In [102], autoregressive discrete autoencoders (ADAs) were explored to train autoregressive models to learn long-range correlations in raw audio waveforms. This technique can be employed to unconditionally produce piano music directly in the raw audio domain. In [103], a novel singing synthesizer was proposed based on a modified version of the WaveNet network. Compared to other works, the features produced by a parametric vocoder were modeled in this work. Given a musical score with lyrics, this approach can generate a synthetic singing voice that can learn both timbre and expression.

IV. Hybrid Model: Generative Adversarial Networks

Generative adversarial networks (GANs) are a class of neural networks developed for unsupervised machine learning. Generally, a standard GAN consists of two main components: a generator G and a discriminator D, which are trained by contesting with each other under a zero-sum game strategy [43]. The discriminator is trained by traditional supervised learning algorithms to perform two-class classification. In contrast, the generator is trained to deceive the discriminator by creating samples of the same distribution as the training data [52], [104].

These types of generative models have been employed in many applications, including high-quality image generation [105], [106], speech/image synthesis [107]–[109], image translation [110], [111], semantic segmentation [112], [113] and object detection [114], [115].

A. How GANs Work

The final goal of a GAN is to learn how to generate new samples from the training samples. Assume that the probability distribution of the training data is $p(x)$. In traditional generative models, we sample from $p(x)$ to obtain new samples whereas GANs attempt to learn a mapping from a random input (or noise) to the training sample. Take image generation as an example, as shown by Fig. 14 the generator is denoted as $G(z;\theta_g)$ where θ is the network parameter and $p_z(z)$ is a prior on input noise variables. The input to the generator G is a 1-dimension random vector, z, and the output is an image produced by G. To force images that are produced to be in the same distribution as the training images, a discriminator is connected to G and denoted as $D(x;\theta_d)$. The input of D is selected randomly from either a real image or a generated

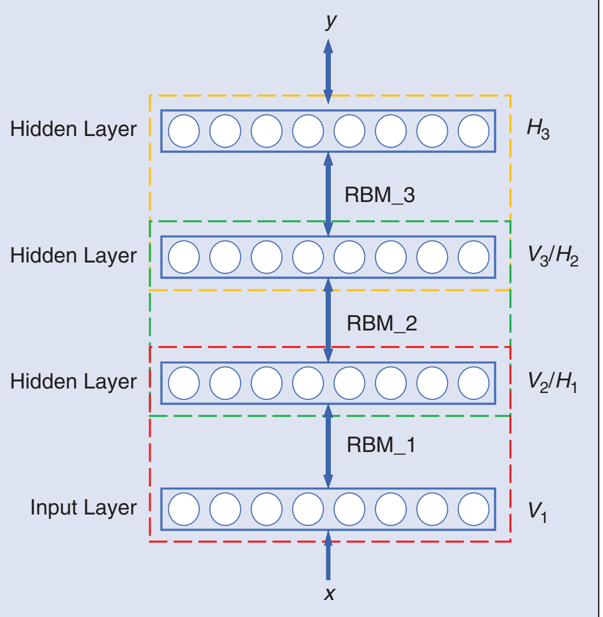


Figure 13. The network architecture of a standard DBN.

[7]http://people.bu.edu/bkulis/projects/music/index.html

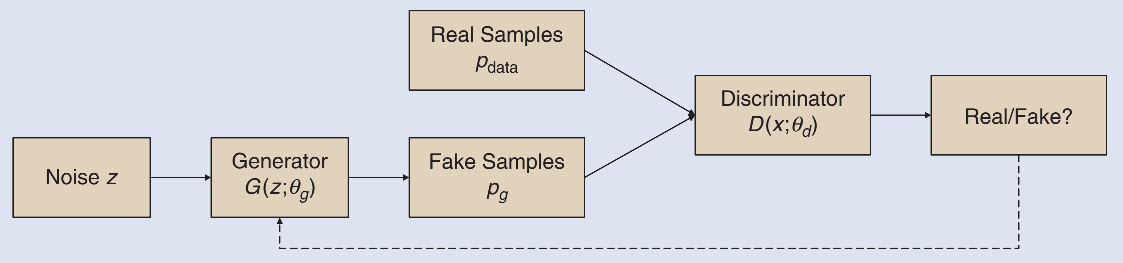


Figure 14. Block diagram of a GAN.

image from G. The output of D is a value in the range [0,1] presenting the probability of whether the input is from the real image p_{data} or generated image p_g. The feedback from D will affect the training of G to prompt the generation of images with higher fidelity, which will be fed to D for further examination. On the contrary, the discriminator is expected to distinguish 'fake' images from real images. This adversarial training procedure, as shown in Fig. 14, is repeated until a balance is reached between the generator G and the discriminator D.

Generally, CNNs are assigned as the primary network for the discriminator to do the image classification. The generator is, in a sense, a type of decoder: taking a vector of random noise and upsampling it to an image. The networks are each optimizing an individual but opposing objective function. As a minimax game between D and G, the value function $V(G,D)$ is given below:

$$\min_G \max_D V(D,G) = \mathrm{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathrm{E}_{z\sim p_z(z)}[\log(1-D(G(z)))] \quad (5)$$

In practice, rather than training G to minimize $\log(1–D(G(z)))$, we always maximize $\log D(G(z))$ to supply sufficient gradients early in the training [43].

B. Wasserstein GAN: A Vital Improvement

Although GANs have shown great success in various applications, the training of a practicable GAN is a tricky process. This process is well known to be slow and unstable [116]. Some of the common problems are listed below:

- **Unstable training**: In the GAN's training procedure, two adversarial models are trained simultaneously. However, the connection between the costs of these two models is independent. Updating the gradient of both models at the same time cannot guarantee a convergence. Moreover, if the performance of the discriminator is too perfect the loss function (5) will fall to zero, which results in no gradient to update. Hence it is a dilemma that the discriminator needs to behave neither too badly nor too well.
- **Mode collapse**: During the training, researchers have found that sometimes the generated images from G are in the same pattern [117]. In other words, the generator is short of sufficient diversity that the 'fake' images created look totally the same. This problem is referred to as mode collapse, a common failure scenario for practical applications of GANs. The generator trapped by mode collapse fails to learn the representation of the real-world data distribution and keeps creating images with low variety.
- **Lack of an evaluation metric**: The objection function of GANs cannot give a clear indication of the training progress. The relationship between the training loss curves and the training progress is confused and hard to be interpreted. There is also no explicit indicator as to the stopping criterion of the training process.

To address the above problems, many possible solutions have been discussed and analyzed and to date, the best solution is provided by the Wasserstein GAN (WGAN) [118]. In the WGAN, the Wasserstein distance is introduced to replace the conventional metrics, i.e. Kullback-Leibler (KL) or Jensen-Shannon (JS) divergence, for quantifying the similarity between two probability distributions. Compared to the KL and JS divergence, the Wasserstein distance can reflect the differential between two distributions even if there is no overlapping part. Only the Wasserstein metric keeps a smooth measure for a stable learning process using gradient descents.

The modified loss functions of the generator and the discriminator in a WGAN are $L(G) = -E_{x\sim p_g}[D(x)]$ and $L(D) = \mathrm{E}_{x\sim p_g}[D(x)] - \mathrm{E}_{x\sim p_{\text{data}}}[D(x)]$, respectively. According to the loss functions, the discriminator is trained to learn a K-Lipschitz continuous function to compute the Wasserstein distance, instead of directly telling the fake samples apart from the real ones. A smaller Wasserstein distance means the output of the generator is closer to the real data distribution. For maintaining the K-Lipschitz continuity of D, weights clipping is applied after updating every gradient to enforce a Lipschitz constraint.

In [119], a gradient penalty is adopted to solve the problems of exploding gradients and vanishing gradients caused by weights clipping in the original WGAN. The new model is named the WGAN-gradient penalty (WGAN-GP). The WGAN-GP has been shown to perform better than the standard WGAN and enables stable training of a wide variety of GAN models. Almost no hyper-parameter tuning is required however the time for training is significantly increased.

Some other remarkable variants of the vanilla GAN include:

- Deep Convolutional GAN (DCGAN) [105]: This is a popular CNN-based GAN, which combines the CNNs in supervised learning and the GANs in unsupervised learning. A set of constraints on the DCGAN is proposed to make this network stable to train. This architecture is a basic component in many types of GANs.
- Auxiliary Classifier GAN (AC-GAN) [108]: Although the DCGAN can produce convincing image samples, it is an intractable problem that GANs cannot

Since GANs have been widely applied for image generation and synthesis, it makes sense to consider them for speech synthesis.

generate globally coherent, high resolution samples from datasets with high variability [108]. To address this problem, label conditioning is employed in the AC-GAN that results in high resolution image samples exhibiting global coherence [104], [120], [121]. Applying a new quantitative metric proposed for image discriminability, it is shown that the samples generated from the AC-GAN are more discriminable than previous models which create lower resolution images and perform a naive resize operation. The generated samples also give a comparable diversity to the training data.

- CycleGAN [122]: Analogous to language translation, image-to-image translation is defined as a problem of translating an image from one representation of a given scene to another, e.g., gray-scale to color, image to semantic labels and edge-map to photograph [122]. For image-to-image translation applications, these can be realized with the CycleGAN by learning the mapping between an input image and an output image, no matter whether the correlation information of the training samples is given or not. Comparisons against previous methods demonstrate that the CycleGAN can outperform the other approaches in quantitative experiments. The CycleGAN can be applied for image processing problems such as season transfer, collection style transfer and photo generation from paintings. A related work is [110].
- WaveGAN [123]: This is an early implementation of GANs for audio synthesis. The WaveGAN is proposed for raw audio synthesis in an unsupervised setting. Testing results suggest that WaveGAN is able to capture semantically-meaningful modes for small-vocabulary speech (such as the SC09 database analyzed in WaveGAN's work).

C. Applications in Audio Generation

Since GANs have been widely applied for image generation and synthesis, it makes sense to consider them for speech synthesis. In [123], the authors applied GANs to synthesize raw audio by introducing the WaveGAN, a time-domain approach, and the SpecGAN, a frequency-domain approach. The WaveGAN was built based on the DCGAN but modifying the transposed convolution operation to widen the receptive field for time-domain signals. Other hyper-parameters remained the same as the DCGAN. Experiments showed that the WaveGAN can produce intelligible words and even audio from other domains like bird vocalizations or piano. The subjective evaluation demonstrated a preference for the generated samples from the WaveGAN.

In [109], an emerging topic of utilizing GANs to synthesize speech for attacking automatic speaker recognition systems was investigated. Various state-of-the-art GANs were examined by fooling a CNN-based text-independent speaker recognizers with generated Mel-spectrograms. For targeted attacks, a modified objective function was proposed to access to universal properties of speech. By applying the WGAN-GP with the modified mixed loss function, it was able to differentiate between real samples from a target speaker and real speech samples from other speakers. Resultant adversarial examples performed well for targeted and untargeted attacks to the speaker recognition system.

A CycleGAN was used in [124] with six layers of fully connected neural networks as the generator and the discriminator. The feature used for training was a mixture of a Mel-spectrogram and the first and second derivatives. A WaveNet vocoder was trained to form the speech waveform. Perceptual evaluations suggested that an effective enhancement and an improvement of the perceptual cleanliness were achieved with the help of GAN-based models. The authors also investigated the quality of the generated speech with publicly available data.

Recently, the GAN based architectures have also been explored and investigated for the music generation problem. In [125], the generator was composed using CNNs for yielding melody in the symbolic domain, while the discriminator was trained to learn the distributions of melodies. This proposed GAN, named with MidiNet, can generate melodies from scratch or by conditioning on the melody of previous bars. In another work of [126], three models were proposed for symbolic multi-track music generation under the framework called MuseGAN.[8] These models were trained on a rock music dataset and used for piano-rolls generation (such as the bass, drums and strings). Interestingly, the models can be extended to generate additional tracks to accompany a given specific track composed by human.

Another emerging application of GANs is data augmentation for speech/acoustic signal processing. In [127],

[8]https://salu133445.github.io/musegan/

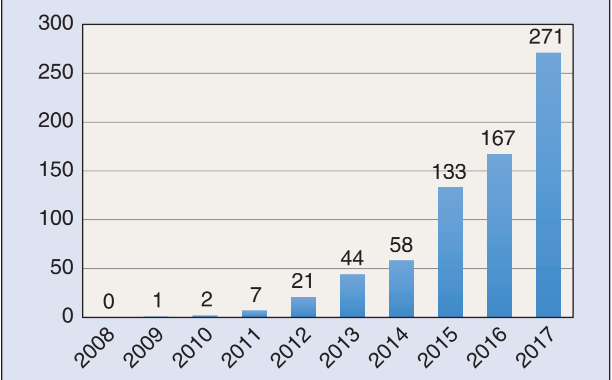


Figure 15. Citation report for total publications.

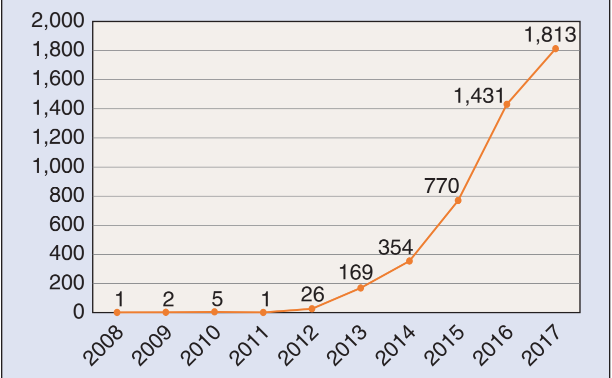


Figure 16. Citation report for sum of times cited per year.

Table II.
A comparison of deep learning based approaches on speech generation.

Paper	Evaluation Results
Zen et al. [47]	39% (Subjective Preference)
Wu et al. [67]	74% (Subjective Preference)
Mehri et al. [69]	80% (Subjective Preference)
Fan et al. [70]	59% (Subjective Preference)
Ling et al. [45]	39% (Subjective Preference)
Zen et al. [64]	3.81 ± 0.11 (5-scale MOS)
Oord et al. [88]	4.21 ± 0.08 (5-scale MOS)
Oord et al. [89]	4.41 ± 0.08 (5-scale MOS)
Ö. Arık et al. [90]	3.94 ± 0.26 (5-scale MOS)
Ö. Arık et al. [91]	3.53 ± 0.12 (5-scale MOS)
Ping et al. [92]	3.78 ± 0.30 (5-scale MOS)
Wang et al. [93]	3.82 ± 0.09 (5-scale MOS)
Shen et al. [94]	4.53 ± 0.07 (5-scale MOS)
Wan et al. [96]	4.53 ± 0.07 (5-scale MOS)

a DCGAN-based model was used for generating more samples to augment the DCASE 2017 dataset. Random values were added to the generated samples to alleviate the issue of sample bias and over-fitting. A support vector machine (SVM) hyper-plane was trained to sift the samples of suitable quality from the generated feature pool. It was confirmed that the usage of the augmented data can improve the acoustic scene classification performance on the DCASE development set.

V. Discussion

The latest developments and applications of deep learning algorithms are reviewed in this paper. Undoubtedly, the introduction of deep generative and discriminative methods in the field of speech/acoustic signal processing boosts the accuracy and the efficiency in recognition and synthesis problems. Fig. 15 and Fig. 16 are the citation analysis from the database of the Web of Science. The search keywords are 'speech synthesis' and 'deep learning.' The bars in Fig. 15 denote the numbers of the publications recorded in the database of the Web of Science from 2008 to 2017. The curve in Fig. 16 is the total number of citations per year. It is obvious that speech synthesis applications have gripped the deep learning research community.

In Table II, evaluation results of several deep learning based speech generation methods are given for a comparison. The data listed in the table are from the best performance of the subjective experiments in each referenced paper. Given the different speech corpora used for synthesis and different implementation protocols, Table II basically demonstrates an increasing quality of the generated speech samples by employing various deep learning techniques.

In this section, the key reasons behind the success of deep learning algorithms with audio generation are discussed and several potential issues for further consideration are presented.

A. Benefits of Deep Learning

The introduction of deep learning is an important breakthrough for the speech processing community. More practical problems in audio generation can be simplified and solved in a much more refined manner. The accessibility and expandability of deep learning techniques are convenient for users to implement audio synthesis systems, not only data experts but also non-expert developers. Given enough data, computational resources and suitable algorithms, deep learning is able to address problems from a variety of aspects in acoustic signal generation.

Listed are several key advantages of deep learning based audio generation systems:

- **Effective learning of representation**: Compared to other machine learning algorithms, deep learning based techniques can learn and create more comprehensive and informative representations from raw audio signals. Deep learning based approaches applied in audio generation systems can model high-dimensional, highly correlated features efficiently. This reduces the demand for hand-crafted feature engineering, which is the most time consuming aspect for speech synthesis systems. More complex sets of features are allowed for training and evaluation in practical tasks.
- **Powerful modeling of relationships**: With the various types of activation functions, NNs have the ability to model nonlinear and complicated relationships between inputs and outputs. The unique multi-layered architecture with nonlinear operations integrates feature extraction with acoustic modeling. NNs can also model multiple simultaneous acoustic events within one frame to create waveform files with high fidelity. This is a vital property for dealing with natural signals sampled from real-world scenarios and for producing fluent conversations or background sounds.
- **Flexible setting of networks**: Unlike many other machine learning methods, specific tasks in audio generation applying deep learning can be processed with a flexible architecture with diverse combinations of DNN modules. This offers a better representation ability with increased flexibility in the parameter configuration. Additionally, there are fewer restrictions placed on the format of the inputs and allows wider commercial deployments.

B. Future Issues

Even though deep learning technology has assisted the exploitation for more audio generation applications, there are still some issues to be resolved for a more advanced realization of the "intelligible chatbot":

- **Design of audio corpora**: With an increasing amount of sound receivers embedded in portable devices, more realistic and coarse data collected are used in the acoustic feature extraction and modeling processes. Moreover, when only a limited or an imbalanced amount of labeled audio signals is available, the performance of a deep neural network will inevitably degrade. Further efforts are required for designing elaborate audio corpora with diverse acoustic environments. More audio datasets composed of specific types of acoustic signals need to be created for recreational and professional generation tasks like music composing and video post-processing.
- **Audio generation systems for practical scenarios**: With the rapid development of computational resources and deep learning algorithms, time cost on training has been effectively reduced by machines equipped with GPUs. However, it is still unworkable for most low-power portable devices like mobile phones. To relieve the computational burden, advances in hardware are necessary for the next generation intelligent equipments for both industry and academia. From the viewpoint of algorithms, several possible future directions warrant further investigation, such as more efficient models adaption for cross-lingual TTS systems and speaking style transferring. Emotional speech synthesis applying deep learning approaches is another potential application in real-world scenes.

VI. Conclusion

In this paper, the most commonly used deep generative and discriminative algorithms were reviewed. A detailed statement for the application of deep learning to speech/acoustic signal processing, especially audio generation, was provided for readers in the machine learning community. Additionally, the benefits introduced by deep learning methods and several issues of further research were discussed in detail.

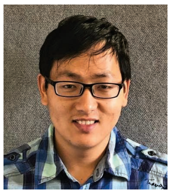
Deep learning methods have shown remarkable improvement for pattern recognition and generation problems. Other machine learning approaches have been outperformed by deep learning methods due to their powerful capability of feature representation and complex model understanding. By adopting up-to-date computational equipments like GPUs, deep learning techniques demonstrate impressive performance for yielding human-like speech. It is indeed the case that there is still a long way to go before the realization of a general intelligence or strong AI. However, there is wide confidence that deep learning will receive more enthusiasm and motivation for broader applications in the future.

Acknowledgment

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research.

Yuanjun Zhao received the Bachelor degree and Master degree in Electronic and Communication Engineering from the Harbin Institute of Technology (HIT) in 2013 and 2015, respectively. Now he is a Ph.D. candidate in the School of Electrical,

Electronic and Computer Engineering from The University of Western Australia (UWA) since 2016. His research interests include speech signal processing, automatic speaker recognition and speaker spoofing detection.



Xianjun Xia received the Bachelor degree in Electronic Engineering from the Hefei University of Technology (HFUT) in 2011 and Master degree in Signal and Information Processing from the University of Science and Technology of China (USTC) in 2014. Now he is a Ph.D. candidate in the School of Electrical, Electronic and Computer Engineering from The University of Western Australia (UWA) since 2015. His research interests include acoustic event detection, machine learning, speech and audio signal processing.

Roberto Togneri (M89-SM04) received the Ph.D degree in 1989 from the University of Western Australia. He joined the School of Electrical, Electronic and Computer Engineering at The University of Western Australia in 1988, where he is now currently an Associate Professor. He leads the Signal Processing and Recognition Lab and his research activities in signal processing and pattern recognition include: feature extraction and enhancement of audio signals, statistical and neural network models for speech and speaker recognition, audio-visual recognition and biometrics, and related aspects of language modelling and understanding. He has published over 150 refereed journal and conference papers in the areas of signal processing and recognition, the chief investigator on three Australian Research Council Discovery Project research grants since 2009, was an Associate Editor for *IEEE Signal Processing Magazine Lecture Notes* and *IEEE Transactions on Speech, Audio and Language Processing* from 2012 to 2016 and is currently the Area Editor for the *IEEE Signal Processing Magazine*, Columns and Forums.

References

[1] T. A. Edison, "The phonograph and its future," *North Amer. Rev.*, vol. 126, no. 262, pp. 527–536, 1878.
[2] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events," *IEEE Trans. Multimedia*, vol. 17, no. 10, pp. 1733–1746, 2015.
[3] H. Zen, K. Tokuda, and A. W. Black, "Statistical parametric speech synthesis," *Speech Commun.*, vol. 51, no. 11, pp. 1039–1064, 2009.
[4] A. J. Hunt and A. W. Black, "Unit selection in a concatenative speech synthesis system using a large speech database," in *1996 IEEE Int. Conf. Acoustics, Speech, and Signal Processing, Conf. Proc. (ICASSP-96.)*, vol. 1. pp. 373–376.
[5] M. Schröder, M. Charfuelan, S. Pammi, and I. Steiner, "Open source voice creation toolkit for the MARY TTS platform," in *Proc. 12th Annu. Conf. Int. Speech Communication Association*, 2011.
[6] J. Yamagishi, "An introduction to HMM-based speech synthesis," National Institute of Informatics, Tech. Rep., 2006.
[7] A. F. Machado and M. Queiroz, "Voice conversion: A critical survey," in *Proc. Sound and Music Computing (SMC)*, 2010, pp. 1–8.
[8] D. Childers, B. Yegnanarayana, and K. Wu, "Voice conversion: Factors responsible for quality," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing, (ICASSP'85)*, vol. 10. 1985, pp. 748–751.
[9] T. Toda, H. Saruwatari, and K. Shikano, "Voice conversion algorithm based on Gaussian mixture model with dynamic frequency warping of straight spectrum," in *Proc. 2001 IEEE Int. Conf. Acoustics, Speech, and Signal Processing, (ICASSP'01)*, vol. 2. pp. 841–844.
[10] T. Toda et al., "The voice conversion challenge 2016," in *Proc. Interspeech*, 2016, pp. 1632–1636.
[11] Y. Tabet and M. Boughazi, "Speech synthesis techniques: A survey," in *Proc. 2011 7th Int. Workshop Systems, Signal Processing and their Applications (WOSSPA)*. IEEE, pp. 67–70.
[12] Z. Wu, N. Evans, T. Kinnunen, J. Yamagishi, F. Alegre, and H. Li, "Spoofing and countermeasures for speaker verification: A survey," *Speech Commun.*, vol. 66, pp. 130–153, 2015.
[13] M. Todisco, H. Delgado, and N. Evans, "Constant Q cepstral coefficients: A spoofing countermeasure for automatic speaker verification," *Comput. Speech Lang.*, vol. 45, pp. 516–535, 2017.
[14] Y. Zhao, R. Togneri, and V. Sreeram, "Compressed high dimensional features for speaker spoofing detection," in *Proc. Asia-Pacific Signal and Information Processing Association Annu. Summit and Conf. (APSIPA ASC)*. IEEE, 2017, pp. 569–572.
[15] C. Zhang, C. Yu, and J. H. Hansen, "An investigation of deep-learning frameworks for speaker verification antispoofing," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 4, pp. 684–694, 2017.
[16] N. Agarwala, Y. Inoue, and A. Sly, "Music composition using recurrent neural networks," Technical Report, Tech. Rep., 2017.
[17] J.-P. Briot, G. Hadjeres, and F. Pachet, "Deep learning techniques for music generation-a survey," arXiv Preprint, arXiv:1709.01620, 2017.
[18] A. Huang and R. Wu, "Deep learning for music," arXiv Preprint, arXiv:1606.04930, 2016.
[19] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.
[20] K. Sundararajan and D. L. Woodard, "Deep learning for biometrics: A survey," *ACM Comput. Surv. (CSUR)*, vol. 51, no. 3, p. 65, 2018.
[21] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, 1943.
[22] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Rev.*, vol. 65, no. 6, p. 386, 1958.
[23] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ., San Diego, La Jolla Inst. for Cognitive Science, Tech. Rep., 1985.
[24] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, 2015.
[25] P. Werbos, "Beyond regression: New fools for prediction and analysis in the behavioral sciences," 1974.
[26] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.
[27] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *J. Mach. Learn. Res.*, vol. 11, no. Feb, pp. 625–660, 2010.
[28] C. Poultney, S. Chopra, Y. L. Cun et al., "Efficient learning of sparse representations with an energy-based model," in *Proc. Advances in Neural Information Processing Systems*, 2007, pp. 1137–1144.
[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[30] O. Russakovsky et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.
[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.
[32] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. European Conf. Computer Vision*. Springer-Verlag, 2014, pp. 818–833.
[33] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv Preprint, arXiv:1409.1556, 2014.
[34] C. Szegedy et al., "Going deeper with convolutions," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[35] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
[36] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
[37] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” in *Proc. 25th Int. Conf. Machine Learning*. ACM, 2008, pp. 1096–1103.
[38] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion,” *J. Mach. Learn. Res.*, vol. 11, no. Dec, pp. 3371–3408, 2010.
[39] A. Makhzani and B. Frey, “K-sparse autoencoders,” arXiv Preprint, arXiv:1312.5663, 2013.
[40] D. P. Kingma and M. Welling, “Auto-encoding variational Bayes,” arXiv Preprint, arXiv:1312.6114, 2013.
[41] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, “Contractive auto-encoders: Explicit invariance during feature extraction,” in *Proc. 28th Int. Conf. Machine Learning*. Omnipress, 2011, pp. 833–840.
[42] L. Deng and N. Jaitly, “Deep discriminative and generative models for speech pattern recognition,” in *Handbook of Pattern Recognition and Computer Vision*. New York: World Scientific, 2016, pp. 27–52.
[43] I. Goodfellow et al., “Generative adversarial nets,” in *Proc. Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.
[44] L. Deng and D. Yu, “Deep learning: Methods and applications,” *Foundations Trends® Signal Process.*, vol. 7, no. 3–4, pp. 197–387, 2014.
[45] Z.-H. Ling, L. Deng, and D. Yu, “Modeling spectral envelopes using restricted Boltzmann machines and deep belief networks for statistical parametric speech synthesis,” *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 21, no. 10, pp. 2129–2139, 2013.
[46] S. Kang, X. Qian, and H. Meng, “Multi-distribution deep belief network for speech synthesis,” in *Proc. 2013 IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8012–8016.
[47] H. Ze, A. Senior, and M. Schuster, “Statistical parametric speech synthesis using deep neural networks,” in *Proc. 2013 IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7962–7966.
[48] L.-J. Liu, L.-H. Chen, Z.-H. Ling, and L.-R. Dai, “Using bidirectional associative memories for joint spectral envelope modeling in voice conversion,” in *Proc. 2014 IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7884–7888.
[49] S. H. Mohammadi and A. Kain, “A voice conversion mapping function based on a stacked joint-autoencoder,” in *Proc. INTERSPEECH*, 2016, pp. 1647–1651.
[50] J. Colonel, C. Curro, and S. Keene, “Improving neural net auto encoders for music synthesis,” in *Audio Engineering Society Convention 143*. Audio Engineering Society, 2017.
[51] A. M. Sarroff and M. A. Casey, “Musical audio synthesis using autoencoding neural nets,” in *Proc. ICMC*, 2014.
[52] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*, vol. 1. Cambridge, MA: MIT Press, 2016.
[53] J. Gu et al., “Recent advances in convolutional neural networks,” *Pattern Recog.*, vol. 77, pp. 354–377, 2018.
[54] M. Lin, Q. Chen, and S. Yan, “Network in network,” arXiv Preprint, arXiv:1312.4400, 2013.
[55] Z. C. Lipton, J. Berkowitz, and C. Elkan, “A critical review of recurrent neural networks for sequence learning,” arXiv Preprint, arXiv:1506.00019, 2015.
[56] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in *Proc. 14th Int. Conf. Artificial Intelligence and Statistics*, 2011, pp. 315–323.
[57] V. Nair and G. E. Hinton, “Rectified linear units improve restricted Boltzmann machines,” in *Proc. 27th Int. Conf. Machine Learning (ICML-10)*, 2010, pp. 807–814.
[58] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[59] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with LSTM,” 1999.
[60] K. Cho et al., “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” arXiv Preprint, arXiv:1406.1078, 2014.
[61] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” arXiv Preprint, arXiv:1412.3555, 2014.
[62] R. Dey and F. M. Salemt, “Gate-variants of gated recurrent unit (GRU) neural networks,” in *Proc. 2017 IEEE 60th Int. Midwest Symp. Circuits and Systems (MWSCAS)*, pp. 1597–1600.
[63] Z.-H. Ling et al., “Deep learning for acoustic modeling in parametric speech generation: A systematic review of existing techniques and future trends,” *IEEE Signal Process. Mag.*, vol. 32, no. 3, pp. 35–52, 2015.
[64] H. Zen and A. Senior, “Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis,” in *Proc. 2014 IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3844–3848.
[65] J. Lorenzo-Trueba, G. E. Henter, S. Takaki, J. Yamagishi, Y. Morino, and Y. Ochiai, “Investigating different representations for modeling and controlling multiple emotions in DNN-based speech synthesis,” *Speech Commun.*, vol. 99, pp. 135–143, 2018.
[66] J. Chorowski, R. J. Weiss, R. A. Saurous, and S. Bengio, “On using backpropagation for speech texture generation and voice conversion,” in *Proc. 2018 IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2018, pp. 2256–2260.
[67] Z. Wu and S. King, “Investigating gated recurrent networks for speech synthesis,” in *Proc. 2016 IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2016, pp. 5140–5144.
[68] B. Li and H. Zen, “Multi-language multi-speaker acoustic modeling for LSTM-RNN based statistical parametric speech synthesis,” in *Proc. INTERSPEECH*, 2016, pp. 2468–2472.
[69] S. Mehri et al., “SampleRNN: An unconditional end-to-end neural audio generation model,” arXiv Preprint, arXiv:1612.07837, 2016.
[70] Y. Fan, Y. Qian, F.-L. Xie, and F. K. Soong, “TTS synthesis with bidirectional LSTM based recurrent neural networks,” in *Proc. 15th Annu. Conf. Int. Speech Communication Association*, 2014.
[71] L. Sun, S. Kang, K. Li, and H. Meng, “Voice conversion using deep bidirectional long short-term memory based recurrent neural networks,” in *Proc. 2015 IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4869–4873.
[72] H. Zhu et al., “Xiaoice Band: A melody and arrangement generation framework for pop music,” in *Proc. 24th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*. ACM, 2018, pp. 2837–2846.
[73] D. D. Johnson, “Generating polyphonic music using tied parallel networks,” in *Proc. Int. Conf. Evolutionary and Biologically Inspired Music and Art*. Springer-Verlag, 2017, pp. 128–143.
[74] G. Brunner, Y. Wang, R. Wattenhofer, and J. Wiesendanger, “JamBot: Music theory aware chord based generation of polyphonic music with LSTMs,” in *Proc. 2017 IEEE 29th Int. Conf. Tools with Artificial Intelligence (ICTAI)*, pp. 519–526.
[75] M. Blaauw and J. Bonada, “A singing synthesizer based on pixelCNN,” in *María de Maeztu Seminar on Music Knowledge Extraction Using Machine Learning (Collocated with NIPS)*. 2016. Accessed on: Oct. 1, 2017. [Online]. Available: http://www.dtic.upf.edu/~mblaauw/MdM_NIPS_seminar/
[76] A. van den Oord et al., “Conditional image generation with pixelCNN decoders,” in *Proc. Advances in Neural Information Processing Systems*, 2016, pp. 4790–4798.
[77] M. Nishimura, K. Hashimoto, K. Oura, Y. Nankaku, and K. Tokuda, “Singing voice synthesis based on deep neural networks,” in *Proc. INTERSPEECH*, 2016, pp. 2478–2482.
[78] E. Gómez, M. Blaauw, J. Bonada, P. Chandna, and H. Cuesta, “Deep learning for singing processing: Achievements, challenges and impact on singers and listeners,” arXiv Preprint, arXiv:1807.03046, 2018.
[79] D. Makris, M. Kaliakatsos-Papakostas, I. Karydis, and K. L. Kermanidis, “Combining LSTM and feed forward neural networks for conditional rhythm composition,” in *Proc. Int. Conf. Engineering Applications of Neural Networks*. Springer-Verlag, 2017, pp. 570–582.
[80] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, 2013.
[81] T. N. Sainath, B. Kingsbury, and B. Ramabhadran, “Auto-encoder bottleneck features using deep belief networks,” in *Proc. 2012 IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4153–4156.
[82] K. Janod et al., “Denoised bottleneck features from deep autoencoders for telephone conversation analysis,” *IEEE/ACM Trans. Audio, Speech Lang. Process. (TASLP)*, vol. 25, no. 9, pp. 1809–1820, 2017.
[83] S. Takaki and J. Yamagishi, “A deep auto-encoder based low-dimensional feature extraction from FFT spectral envelopes for statistical parametric speech synthesis,” in *Proc. 2016 IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5535–5539.

[84] Y. Pu et al., “Variational autoencoder for deep learning of images, labels and captions,” in *Proc. Advances in Neural Information Processing Systems*, 2016, pp. 2352–2360.
[85] J. Walker, C. Doersch, A. Gupta, and M. Hebert, “An uncertain future: Forecasting from static images using variational autoencoders,” in *Proc. European Conf. Computer Vision*. Springer-Verlag, 2016, pp. 835–851.
[86] K. Sohn, H. Lee, and X. Yan, “Learning structured output representation using deep conditional generative models,” in *Proc. Advances in Neural Information Processing Systems*, 2015, pp. 3483–3491.
[87] G. E. Hinton, “A practical guide to training restricted Boltzmann machines,” in *Neural Networks: Tricks of the Trade*. New York: Springer-Verlag, 2012, pp. 599–619.
[88] A. van den Oord et al., “WaveNet: A generative model for raw audio,” in *Proc. 9th ISCA Speech Synthesis Workshop*, Sunnyvale, CA, Sept. 13–15, 2016, p. 125.
[89] A. v d Oord et al., “Parallel WaveNet: Fast high-fidelity speech synthesis,” arXiv Preprint, arXiv:1711.10433, 2017.
[90] S. Ö. Arık et al., “Deep voice: Real-time neural text-to-speech,” in *Proc. 34th Int. Conf. Machine Learning, Machine Learning Research*, vol. 70. D. Precup and Y. W. Teh, Eds. Sydney, Australia: PMLR, International Convention Centre, Aug. 6–11, 2017, pp. 195–204.
[91] A. Gibiansky et al., “Deep voice 2: Multi-speaker neural text-to-speech,” in *Advances in Neural Information Processing Systems 30*, I. Guyon et al., Eds. Curran Associates, 2017, pp. 2962–2970.
[92] W. Ping et al., “Deep voice 3: 2000-speaker neural text-to-speech,” arXiv Preprint, arXiv:1710.07654, 2017.
[93] Y. Wang et al., “Tacotron: Towards end-to-end speech synthesis,” in *Proc. INTERSPEECH*, 2017, pp. 4006–4010.
[94] J. Shen et al., “Natural TTS synthesis by conditioning WaveNet on Mel spectrogram predictions,” in *Proc. 2018 IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2018, pp. 4779–4783.
[95] H. Choi, J. Kim, J. Park, J. Kim, and M. Hahn, “Low-dimensional representation of spectral envelope using deep auto-encoder for speech synthesis,” in *Proc. 2018 2nd Int. Conf. Mechatronics Systems and Control Engineering*. ACM, pp. 107–111.
[96] V. Wan, Y. Agiomyrgiannakis, H. Silen, and J. Vt, “Google’s next-generation real-time unit-selection synthesizer using sequence-to-sequence LSTM-based autoencoders,” in *Proc. Interspeech 2017*, pp. 1143–1147.
[97] Y.-J. Hu and Z.-H. Ling, “Extracting spectral features using deep autoencoders with binary distributed hidden units for statistical parametric speech synthesis,” *IEEE/ACM Trans. Audio, Speech Lang. Process. (TASLP)*, vol. 26, no. 4, pp. 713–724, 2018.
[98] K. Kobayashi, T. Hayashi, A. Tamamori, and T. Toda, “Statistical voice conversion with WaveNet-based waveform generation,” in *Proc. Interspeech*, vol. 2017. 2017, pp. 1138–1142.
[99] R. Manzelli, V. Thakkar, A. Siahkamari, and B. Kulis, “An end to end model for automatic music generation: Combining deep raw and symbolic audio networks,” in *Proc. Musical Metacreation Workshop at 9th Int. Conf. Computational Creativity*, Salamanca, Spain, 2018.
[100] R. Manzelli, V. Thakkar, A. Siahkamari, and B. Kulis, “Conditioning deep generative raw audio models for structured automatic music,” arXiv Preprint, arXiv:1806.09905, 2018.
[101] J. Engel et al., “Neural audio synthesis of musical notes with WaveNet autoencoders,” in *Proc. 34th Int. Conf. Machine Learning, Machine Learning Research*, vol. 70. D. Precup and Y. W. Teh, Eds. PMLR, Aug. 6–11 2017, pp. 1068–1077.
[102] S. Dieleman, A. v d Oord, and K. Simonyan, “The challenge of realistic music generation: Modelling raw audio at scale,” arXiv Preprint, arXiv:1806.10474, 2018.
[103] M. Blaauw and J. Bonada, “A neural parametric singing synthesizer modeling timbre and expression from natural songs,” *Appl. Sci.*, vol. 7, no. 12, p. 1313, 2017.
[104] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training GANs,” in *Proc. Advances in Neural Information Processing Systems*, 2016, pp. 2234–2242.
[105] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” arXiv Preprint, arXiv:1511.06434, 2015.
[106] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski, “Plug & play generative networks: Conditional iterative generation of images in latent space,” *CVPR*, vol. 2, no. 5, p. 7, 2017.
[107] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, “Generative adversarial text to image synthesis,” in *Proc. 33rd Int. Conf. Machine Learning, Machine Learning Research*, vol. 48. M. F. Balcan and K. Q. Weinberger, Eds. New York: PMLR, June 20–22, 2016, pp. 1060–1069.
[108] A. Odena, C. Olah, and J. Shlens, “Conditional image synthesis with auxiliary classifier GANs,” in *Proc. 34th Int. Conf. Machine Learning, Machine Learning Research*, vol. 70. D. Precup and Y. W. Teh, Eds. Sydney, Australia: PMLR, International Convention Centre, Aug. 6–11 2017, pp. 2642–2651.
[109] W. Cai, A. Doshi, and R. Valle, “Attacking speaker recognition with deep generative models,” arXiv Preprint, arXiv:1801.02384, 2018.
[110] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, July 2017.
[111] M.-Y. Liu and O. Tuzel, “Coupled generative adversarial networks,” in *Proc. Advances in Neural Information Processing Systems*, 2016, pp. 469–477.
[112] P. Luc, C. Couprie, S. Chintala, and J. Verbeek, “Semantic segmentation using adversarial networks,” in *Proc. NIPS Workshop Adversarial Training*, 2016.
[113] W. Zhu, X. Xiang, T. D. Tran, and X. Xie, “Adversarial deep structural networks for mammographic mass segmentation,” arXiv Preprint, arXiv:1612.05970, 2016.
[114] X. Wang, A. Shrivastava, and A. Gupta, “A-fast-RCNN: Hard positive generation via adversary for object detection,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2017.
[115] J. Li, X. Liang, Y. Wei, T. Xu, J. Feng, and S. Yan, “Perceptual generative adversarial networks for small object detection,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2017.
[116] M. Arjovsky and L. Bottou, “Towards principled methods for training generative adversarial networks,” arXiv Preprint, arXiv:1701.04862, 2017.
[117] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, “Unrolled generative adversarial networks,” arXiv Preprint, arXiv:1611.02163, 2016.
[118] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein generative adversarial networks,” in *Proc. 34th Int. Conf. Machine Learning, Machine Learning Research*, vol. 70. D. Precup and Y. W. Teh, Eds. Sydney, Australia: PMLR, International Convention Centre, Aug. 6–11, 2017, pp. 214–223.
[119] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, “Improved training of Wasserstein GANs,” in *Proc. Advances in Neural Information Processing Systems*, 2017, pp. 5767–5777.
[120] A. Odena, “Semi-supervised learning with generative adversarial networks,” arXiv Preprint, arXiv:1606.01583, 2016.
[121] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune, “Synthesizing the preferred inputs for neurons in neural networks via deep generator networks,” in *Proc. Advances in Neural Information Processing Systems*, 2016, pp. 3387–3395.
[122] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2017.
[123] C. Donahue, J. McAuley, and M. Puckette, “Synthesizing audio with generative adversarial networks,” arXiv Preprint, arXiv:1802.04208, 2018.
[124] J. Lorenzo-Trueba, F. Fang, X. Wang, I. Echizen, J. Yamagishi, and T. Kinnunen, “Can we steal your vocal identity from the internet? Initial investigation of cloning Obamas voice using GAN, WaveNet and low-quality found data,” in *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, pp. 240–247.
[125] L. Yang, S. Chou, and Y. Yang, “MidiNet: A convolutional generative adversarial network for symbolic-domain music generation,” in *Proc. 18th Int. Society for Music Information Retrieval Conf., ISMIR 2017*, Suzhou, China, Oct. 23-27, 2017, pp. 324–331.
[126] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, “MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment,” 2018. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17286
[127] S. Mun, S. Park, D. K. Han, and H. Ko, “Generative adversarial network based acoustic scene training set augmentation and selection using SVM hyper-plane,” in *Proc. DCASE*, 2017, pp. 93–97.
[128] S. Sabour, N. Frosst, and G. E. Hinton, “Dynamic routing between capsules,” in *Proc. Advances in Neural Information Processing Systems*, 2017, pp. 3856–3866.