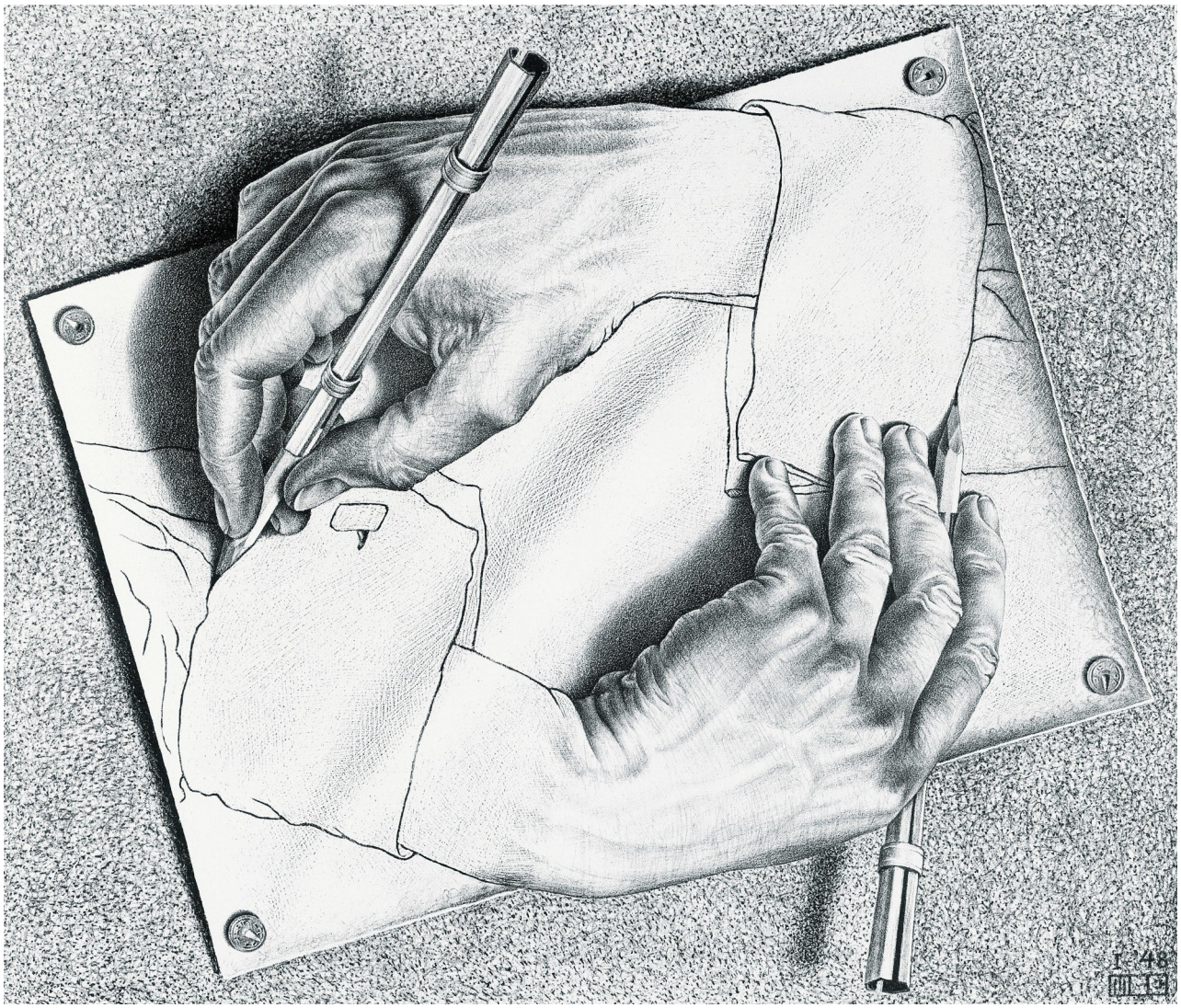
Generative Adversarial Networks for Speech Processing

Daniel Michelsanti

Master's Thesis
M.Sc. Vision, Graphics and Interactive Systems - Spring 2017

Supervisor: Zheng-Hua Tan



Cover image: *Drawing Hands*, by M.C. Escher

AALBORG UNIVERSITY
STUDENT REPORT

Vision, Graphics and Interactive Systems 10
Department of Electronic Systems
School of Information and Communication Technology
http://sict.aau.dk

Title:

Generative Adversarial Networks for Speech Processing

Theme:

Master's Thesis

Project period and hand-in:

Wednesday, 1^{st} February 2017 - Thursday, 8^{th} June 2017

Project group:

17gr1046

Participant:

Daniel Michelsanti

Supervisor:

Zheng-Hua Tan

Number of pages: 67

Synopsis:

Deep learning approaches have gained popularity in a variety of fields, such as computer vision, speech processing, and natural language processing, due to their impressive performance and their flexibility. Among them, a new framework for deep generative model estimation has been recently proposed: generative adversarial network. This framework has already shown good performance in different image processing and computer vision tasks, but its adoption for speech-related tasks is still limited. In this project we explore some of the possibilities that an adversarial training can offer for speech processing. In particular, two applications have been considered: speech enhancement and automatic speech generation. Regarding speech enhancement, experimental results show that the adopted approach overall outperforms the classical short-time spectral amplitude minimum mean square error method, and is comparable to a deep neural network-based technique. On the other hand, the results on automatic speech generation indicate that our models are able to generate plausible spectrograms, even though some artefacts can be heard in the reconstructed signals. We provide generated samples for a subjective evaluation of the quality.

PREFACE

This report covers the master's thesis project made by Daniel Michelsanti in the Centre for Acoustic Signal Processing at Aalborg University during the spring semester of 2017. The 30-ECTS project has been conducted to fulfil the requirements of the degree in Vision, Graphics and Interactive Systems.

The author would like to thank Prof. Zheng-Hua Tan for his advices and general supervision throughout the semester. A thank also goes to Hong Yu for providing data and speaker verification results for the baseline systems and Morten Kolbæk for his assistance and software used for the speaker verification and DNN speech enhancement baseline systems.

SUMMARY

This report investigates the use of generative adversarial networks for speech processing. Specifically, the two applications that are taken into account are speech enhancement and automatic speech generation.

Generative adversarial networks are a recently introduced framework for deep generative model estimation. The idea is to perform an alternate training of two models: a generator that tries to capture the real data distribution, and a discriminator that tries to determine if a given sample is real or from the generative model. The generator improves because it is trained to fool the discriminator. Even though this approach has shown good performance in a variety of computer vision tasks, like image super-resolution, text-to-image synthesis, and image-to-image translation, its adoption for speech-related tasks is still limited. This motivates the investigation of the possibilities that an adversarial training can offer for speech processing.

We limit our attention to speech enhancement and automatic speech generation. In the first case, we make use of a framework previously adopted for image-to-image translation to perform spectral enhancement. This is the first attempt of using conditional generative adversarial networks to enhance speech to our knowledge. The obtained results show that the approach overall outperforms the classical short-time spectral amplitude minimum mean square error method, and is comparable to an ideal ratio mask estimation technique that utilises deep neural networks.

Regarding the automatic speech generation, we use three models to generate speech spectrograms: a traditional generative adversarial network, and two extensions, one that allows to learn disentangled representations in an unsupervised way, and the other that uses a condition to generate data. The results indicate that our approach can generate plausible spectrograms, but some artefacts can be heard in the reconstructed signals, making sometimes hard to recognise the spoken word. We provide some samples to let the reader subjectively evaluate the quality of them.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The large amount of information available today requires the design of algorithms that can deal with it. One way to address this is by using generative models [33]. The idea behind generative models is to have a compact representation able to capture the main characteristics of the data (generally high-dimensional) and generate new samples similar to the original ones. This can be performed adopting deep networks, consisting of multiple layers that can learn features at different levels of abstraction.

However, modelling high-dimensional distributions is not the only motivation behind studying generative models. Other reasons are provided by Goodfellow [16]. They are also particularly suitable for those applications where realistic synthetic samples are required: data augmentation, image editing, style transferring etc.

One of the most promising deep generative models is generative adversarial network (GAN). Here we have two networks that play a game against each other, where one of them, the generator, tries to generate data as similar as possible to the training set, while the other, the discriminator, needs to distinguish between the samples coming from the real distribution and the generated ones.

Although GANs have been extensively used in computer vision to generate images [63, 65], learn representations [63, 5], manipulate images [92], and obtain super-resolved images [42], their application to speech processing is still limited. This motivates an investigation of the performance that an adversarial training can achieve in speech-related tasks. In this project the focus is on speech enhancement and automatic speech generation, and the research questions we address are the following ones:

1. How can we apply GANs to spectral speech enhancement and what performance can be achieved compared to other methods?

2. How can we generate speech signals with a GAN-based approach and how good are they?

This report consists of the following parts:

- Chapter 1: this chapter introduces the motivation behind this project and defines the research questions that this work addresses.
- Chapter 2: this chapter presents the basic concepts of speech processing.
- Chapter 3: this chapter presents the basic theory behind deep learning, with a particular focus on GANs.
- Chapter 4: this chapter shows how we apply GANs to tackle the speech enhancement problem.
- Chapter 5: this chapter shows how we use GANs to generate speech signals.
- Chapter 6: in this chapter the conclusions are drawn.
- Appendices: in this part the reader can find some additional materials.

CHAPTER 2

BASICS OF SPEECH PROCESSING

In order to deal with speech-related applications, some important concepts about speech processing need to be introduced. This is what we will do in this Chapter, so that the reader can understand the other sections of the report more easily.

2.1 Speech Production

Speech is a signal produced by humans that allows them to communicate with each other. The three main parts involved in speech production are the lungs, the larynx, and the vocal tract [46] (Figure 2.1).

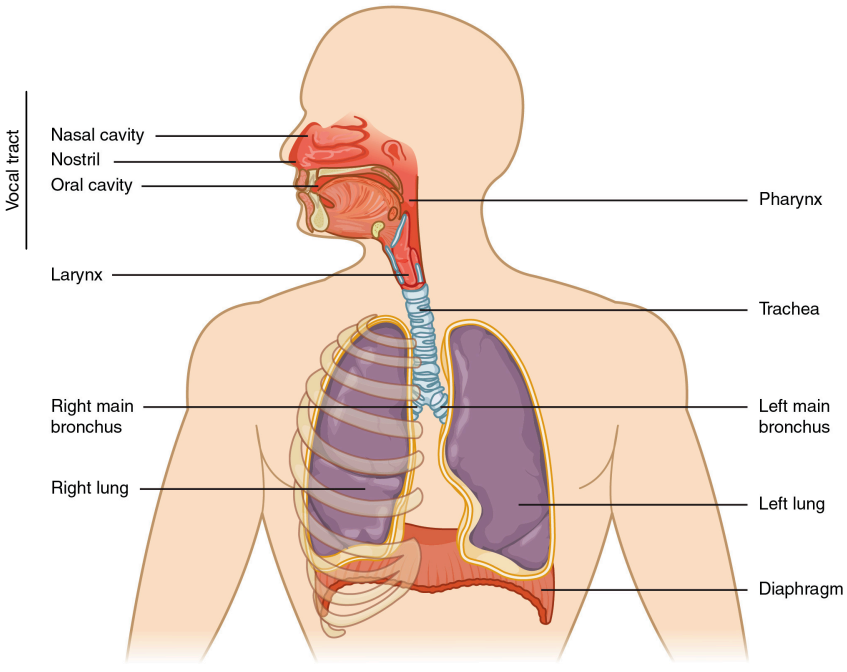


Figure 2.1. Speech production system.

Lungs provide the adequate airflow to the other organs by changing the respiratory pattern, with a longer expiratory phase [24]. The larynx consists of several structures and controls the vocal folds [46]. When the opening between the two folds, called glottis, is wide, we say that the vocal folds are in the breathing state, because they do not oppose any relevant resistance to the airflow coming from the lungs [46]. In order to generate voiced sounds, such as vowel, the vocal folds vibrate and alternate between a closed and an open phase. The time duration of this cycle is called pitch period and its reciprocal is the fundamental frequency (F_0) [46]. F_0 is *"the lowest harmonic component in voiced sounds"* [24], and it is 80 Hz or lower for males and 300 Hz or above for female and children [8]. The majority of the consonants are generated in the unvoiced state, where the vocal folds do not vibrate, but they are tense and close to each other [46]. Finally, the vocal tract includes the oral cavity and the nasal cavity [46] and allows to produce additional sounds through articulatory organs (tongue, lips, and velum) [24].

As described in [46], it is possible to use a simple linear model for the speech production system (Figure 2.2). Based on the state of the vocal folds, voiced or unvoiced, we have a pulse generator (periodic source) or a random noise generator (aperiodic source) as an excitation signal of the vocal tract represented by a linear filter. Then, a high-pass filter is applied to the output in order to model the sound radiation at the lips.

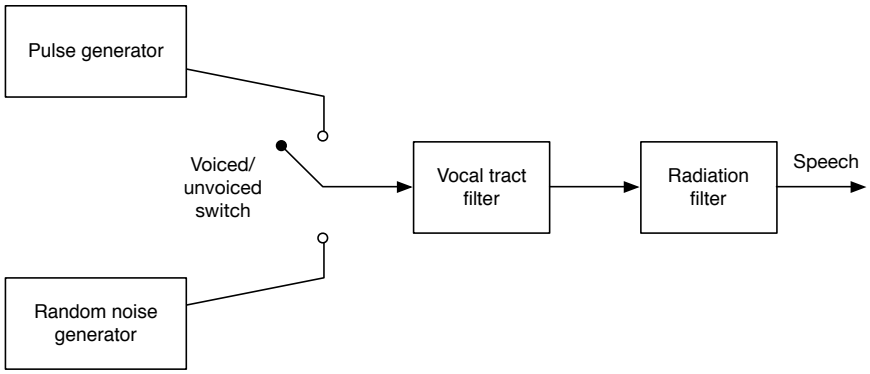


Figure 2.2. Speech production model. Inspired by [46].

2.2 Time-Frequency Representation of Speech Signals

One of the most popular method to perform the Fourier analysis of a signal is the discrete Fourier transform (DFT). The DFT is applied to a finite-length signal sequence $x(n)$ with $0 \leq n \leq N-1$ and is defined as [8]:

$$X^d(\omega) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}\omega n}, \quad 0 \leq \omega \leq N-1. \tag{2.1}$$

From X^d it is possible to recover the original signal with the inverse DFT [8]:

$$x(n) = \frac{1}{N} \sum_{\omega=0}^{N-1} X^d(\omega) e^{j\frac{2\pi}{N}\omega n}, \quad 0 \leq n \leq N-1. \tag{2.2}$$

If $x(n)$ is a real-valued sequence, then the DFT has the following symmetry property:

$$X^d(N-\omega) = (X^d(\omega))^* \tag{2.3}$$

where $()^*$ is the complex conjugate. This follows directly from the DFT definition:

$$\begin{aligned} X^d(N-\omega) &= \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}(N-\omega)n} && (2.4)\\ &= \sum_{n=0}^{N-1} x(n) e^{-j2\pi n} e^{j\frac{2\pi}{N}\omega n} && (2.5)\\ &= \sum_{n=0}^{N-1} x(n) e^{j\frac{2\pi}{N}\omega n} && (2.6)\\ &= \sum_{n=0}^{N-1} \left(x(n) e^{-j\frac{2\pi}{N}\omega n} \right)^* && (2.7)\\ &= (X^d(\omega))^*, && (2.8) \end{aligned}$$

where in Equation 2.6 we used the Euler formula to find that $e^{-j2\pi k} = \cos(2\pi k) - j\sin(2\pi k) = 1 - j0 = 1 \; \forall k$ and in Equation 2.7 we used the hypothesis of $x(n)$ as a real-valued sequence. From the complex-conjugate symmetry property, it follows that:

$$|X^d(N-\omega)| = |X^d(\omega)|. \tag{2.9}$$

Since speech signals are generated by a non-stationary process, which means that its statistical properties change over time, the previously described DFT is not a suitable technique for the Fourier analysis: the DFT is defined for the entire signal, and it would be impossible to capture its variations [8]. However, it is still possible to apply the DFT to a block of neighbouring samples (a frame generally between 10 and 35 ms), assuming that in this frame the signal is stationary. We can repeat this procedure for successive overlapping windowed frames. This is the main idea behind the discrete short-time Fourier transform (STFT) that can be defined as follows:

$$X(n,\omega) = \sum_{m=-\infty}^{\infty} x(m) w(n-m) e^{-j\frac{2\pi}{N}\omega m}, \quad 0 \leq n, \omega \leq N-1, \tag{2.10}$$

where $w(n-m)$ is a window function centred at $n = m$, used to reduce the introduction of frequency components in the spectrum of the signal, a phenomenon known as spectral leakage.

The magnitude of the STFT, $|X(n,\omega)|$, is called spectrogram [8] (Figure 2.3) and *"describes the speech signal's relative energy concentration in frequency as a function of the time"* [46]. A spectrogram is a widely used time-frequency (T-F) representation of a signal because it clearly shows the main characteristics of it, like the changes of the frequency peaks, or formants, over time [8].

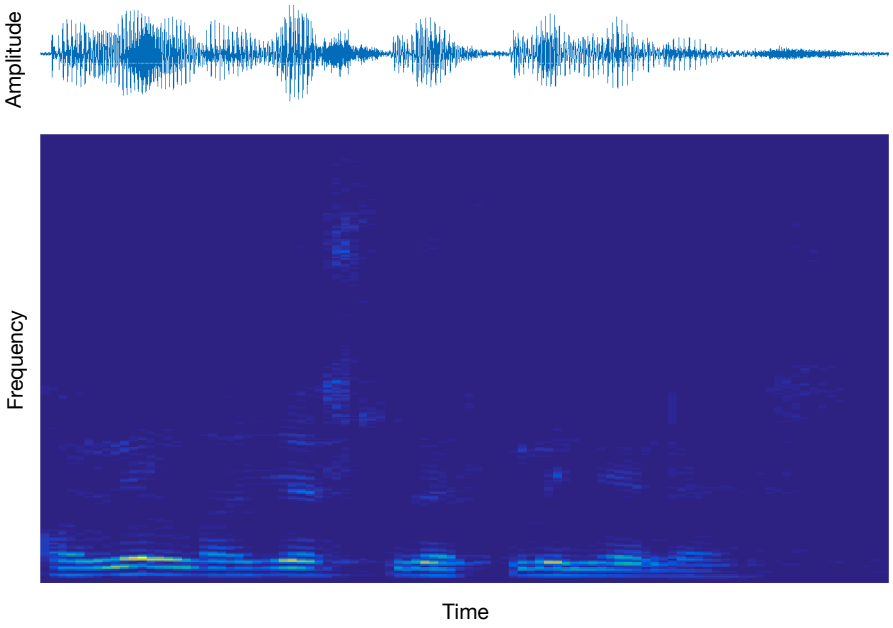


Figure 2.3. Spectrogram (bottom) of the sentence "Only lawyers love millionaires" pronounced by a male speaker, along with the waveform (top) of the signal. The MATLAB implementation used to generate the spectrogram is the one in [91].

Generally, we can distinguish between two kinds of spectrogram, based on the window length (Figure 2.4). If $w(n)$ has a short duration, then we have a wideband spectrogram, with high time resolution, but poor frequency localisation. When a long duration window is used, the spectrogram is called narrowband. In this case, we have more spectral details, but a poor time resolution [8].

2.3 Speech Reconstruction

Once that we have the STFT representation of a signal, $X(n,\omega)$, we can reconstruct (or synthesise) it using the overlap-and-add method [46] according to:

$$y(n) = \sum_{m=-\infty}^{\infty} \left[\frac{1}{N} \sum_{\omega=0}^{N-1} X(m,\omega)\, e^{j\frac{2\pi}{N}\omega n}, \right] \tag{2.11}$$

where the term in the brackets is an inverse DFT and can be seen as [46]:

$$x_m(n) = x(n)w(m-n). \tag{2.12}$$

Now, we can rewrite the Equation 2.11 as [46]:

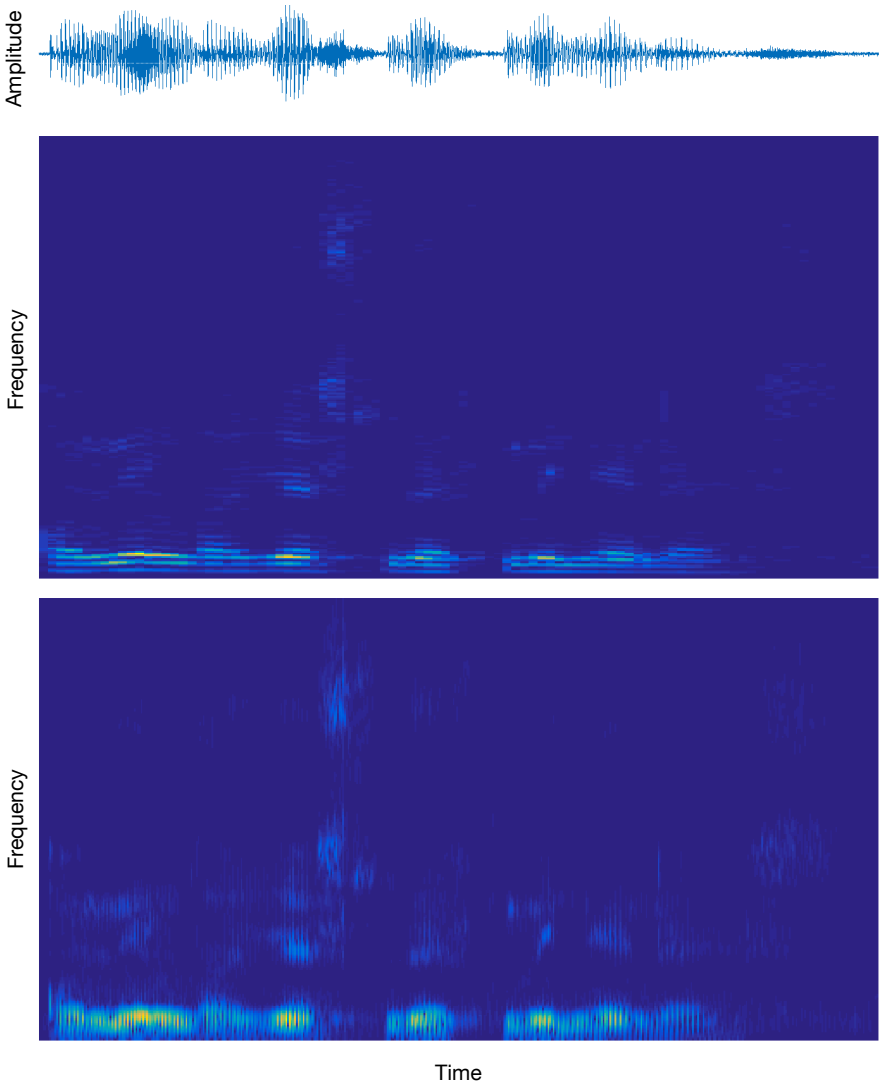


Figure 2.4. Difference between narrowband (centre) and wideband (bottom) spectrograms. On top, the waveform of the signal is reported.

$$y(n) = \sum_{m=-\infty}^{\infty} x_m(n) = x(n) \sum_{m=-\infty}^{\infty} w(m-n) \tag{2.13}$$

$$= c\, x(n), \tag{2.14}$$

where in 2.14 we assumed that the summation is equal to a constant c.

Sometimes, we want to apply some modification to $X(n, \omega)$. Since the values that a STFT can assume are complex, we can see it in polar form:

$$X(n, \omega) = |X(n, \omega)| e^{j\phi(n,\omega)}, \tag{2.15}$$

where $|X(n,\omega)|$ is the spectrogram of the signal (as seen in Section 2.2), and $\phi(n,\omega)$ represents its phase. In speech enhancement (see Section 4 for further details), we modify the spectrogram of the signal, and then we synthesise the new waveform by using the modified spectrogram and the phase of the noisy signal, which has been shown to be a good estimate of the clean signal's phase [4].

In other applications, we only know the spectrogram of the signal and we need a phase in order to reconstruct it. Assuming zero or random phase generally tends to yield poor results, so we can think to estimate the signal from the modified STFT magnitude. Griffin and Lim [18] proposed an iterative approach to accomplish this. Suppose to have a signal $x(n)$, its STFT, $X(n,\omega)$, and the modified STFT (MSTFT), $Y(n,\omega)$. We can consider the distance measure between the signal and the MSTFT as the squared error between $X(n,\omega)$ and $Y(n,\omega)$ [18]:

$$D(x(n), Y(n,\omega)) = \sum_{n=-\infty}^{\infty} \frac{1}{N} \sum_{\omega=0}^{N-1} |X(n,\omega) - Y(n,\omega)|^2, \tag{2.16}$$

$$= \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} [x_m(n) - y_m(n)]^2, \tag{2.17}$$

where $x_m(n) = x(n)w(m-n)$ and $y_m(n) = \frac{1}{N}\sum_{\omega=0}^{N-1} Y(m,\omega)e^{j\frac{2\pi}{N}\omega n}$. Equation 2.17 follows from the Parseval's theorem. In order to minimise $D(x(n), Y(m,\omega))$, we can set the gradient to zero and find that [18]:

$$x(n) = \frac{\sum_{m=-\infty}^{\infty} w(m-n)y_m(n)}{\sum_{m=-\infty}^{\infty} w^2(m-n)}. \tag{2.18}$$

In our case, we want to estimate $x(n)$ given $|Y(n,\omega)|$. If we denote with $x^i(n)$ the estimated version of $x(n)$ at the iteration i, the algorithm consists of the following steps:

1. Compute the STFT of $x^i(n)$, $X^i(n,\omega)$.
2. Replace $|X^i(n,\omega)|$ with $|Y(n,\omega)|$.
3. Calculate $x^{i+1}(n)$ from Equation 2.18 using the modified STFT.
4. Go back to step 1.

Griffin and Lim [18] also showed that this algorithm reduces the squared error between $|X(n,\omega)|$ and $|Y(n,\omega)|$.

2.4 Mel Frequency Cepstral Coefficients

It is possible to represent a speech signal by using acoustic features. The idea is to keep the important aspects of the signal in a compact representation. One of the most used representation, especially in speech and speaker recognition [68], is a set of mel-frequency cepstral coefficients (MFCCs) which are extracted using the

following steps (Figure 2.5). First, the signal is multiplied by a 20 ms-long Hamming window every 10 ms [68]. This allows to have short-time segments where we can assume the signal as stationary. At this point, we compute the magnitude squared spectrum from the DFT and we apply a mel-spaced bank of triangular filters [68]. The mel-frequency scale is linear below 1000 Hz, and logarithmic above 1000 Hz, and can be approximated using the formula [83]:

$$f_{mel} = 2595 \log_{10} \left(1 + \frac{f_{Hz}}{700} \right). \tag{2.19}$$

The reason for using the mel scale is given by the human perception of sound, which does not follow a linear scale [83]. Then, the logarithm of the filter bank output is computed and the discrete cosine transform (DCT) is applied in order to transform the frequency domain in a time-like domain known as quefrency [3]:

$$c_n = \sum_{m=1}^{M} \log(S_m) \cos \left[\frac{\pi n}{M} \left(m - \frac{1}{2} \right) \right], \tag{2.20}$$

where S_m is the output of the previous step. DCT is preferred to the inverse DFT because it is a good approximator of the Karhunen-Loève transform (KLT), which has an optimal decorrelation property [64].

Figure 2.5. Steps needed for MFCCs extraction.

Usually, the N MFCCs, $c_0(n),\ c_1(n),\ \ldots,\ c_{N-1}(n)$, obtained as explained before are concatenated with the logarithm of the signal energy, that for the first K samples is

$$E_{log} = \log \sum_{n=1}^{K} x^2(n), \tag{2.21}$$

and with the first and second order time derivatives of the coefficients, also called delta, $d_m(n)$, and delta-delta, $a_m(n)$, respectively, in order to take into account the speech dynamics. The delta coefficients are computed as follows [48]:

$$d_m(n) = \frac{\sum_{p=-p_0}^{p_0} p\, c_m(n+p)}{\sum_{p=-p_0}^{p_0} p^2} \tag{2.22}$$

where $2\,p_0 + 1$ frames are involved in the computation. The delta-delta features are obtained in the same way, using $d_m(n+p)$ instead of $c_m(n+p)$ in Equation 2.22. To summarise, when we talk about MFCCs, we generally have a total of $3 \cdot (N+1)$ features: N MFCCs, 1 energy feature, N delta MFCC features, N delta-delta MFCC features, 1 delta energy feature, and 1 delta-delta energy feature.

CHAPTER 3

DEEP LEARNING

This Chapter will present the fundamental concepts about deep learning. First we will describe the perceptron, as it is the building block of more complex models. Then, various methods are explained, until we present deep generative models, with a specific focus on GANs.

3.1 Perceptron

The concept of perceptron has been introduced by Rosenblatt in 1957 [72]. This model is inspired by the biological neuron, which can either fires or not based on the stimulations from other neurons. Specifically, two neurons are connected with the synapses, characterised by a weight that controls the connection strength. The electrochemical signals propagate through the dendrites until they reach the cell body (or soma), where the axon hillock generates spikes transmitted by the axon to the other neurons. Figure 3.1 shows the representation of a biological neuron.

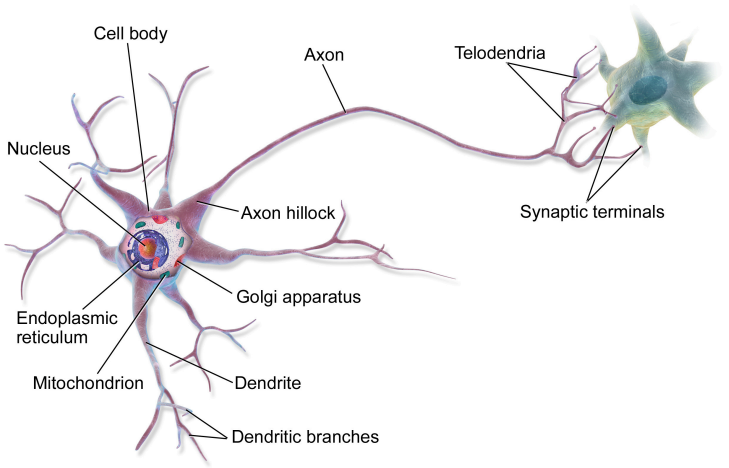


Figure 3.1. Representation of a biological neuron.

We can see the perceptron as a mathematical model of a neuron (Figure 3.2) that outputs a biased linear combination of the input after the application of a non-linearity:

$$y = f(b + \sum_{i=1}^{N} w_i x_i) = f(b + \bar{w}^T \bar{x}), \tag{3.1}$$

where $\bar{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_N \end{bmatrix}^T$ are the input signals, $\bar{w} = \begin{bmatrix} w_1 & w_2 & \dots & w_N \end{bmatrix}^T$ are the learnable weights, b is the bias applied to the linear combination of the input, and f is a non-linear activation function, generally the sign function:

$$\text{sgn}(x) = \begin{cases} -1 & \text{if } x < 0, \\ 1 & \text{otherwise.} \end{cases} \tag{3.2}$$

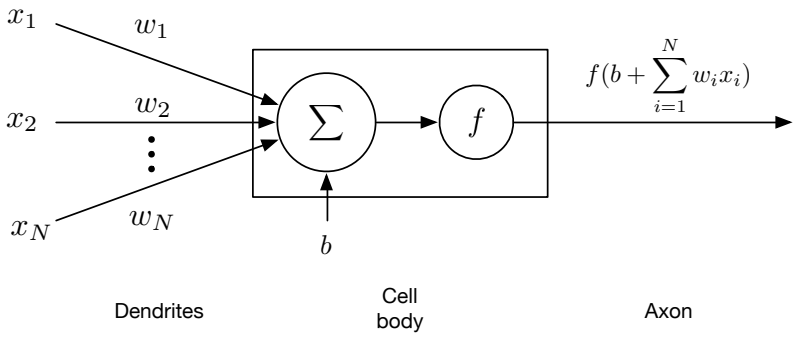


Figure 3.2. Mathematical model of a neuron. Inspired by [1].

To make the notation simpler, we redefine $\bar{w} = \begin{bmatrix} w_0 & w_1 & \dots & w_N \end{bmatrix}^T$ and $\bar{x} = \begin{bmatrix} x_0 & x_1 & \dots & x_N \end{bmatrix}^T$, with $w_0 = b$ and $x_0 = 1$, so that $y = f(\bar{w}^T \bar{x})$. Now, we can try to use the perceptron to solve a binary classification problem. We can consider a dataset of M samples, $D = \{(\bar{x}^{(1)}, d^{(1)}), \dots, (\bar{x}^{(M)}, d^{(M)})\}$, where $\bar{x}^{(m)}$ is the generic input vector that can belong to the set A or to the set B, and

$$d^{(m)} = \begin{cases} -1 & \text{if } \bar{x}^{(m)} \in A, \\ 1 & \text{if } \bar{x}^{(m)} \in B \end{cases}$$

is the desired output of the perceptron for that input. If we indicate with $w_i(t)$ the value that the i^{th} weight assumes at the step t, the learning algorithm used to determine $\bar{w}$ consists of the following steps:

1. Initialise w_i randomly.
2. Select a training sample $\bar{x}^{(m)}$ randomly.
3. If the output of the perceptron differs from $d^{(m)}$, update the weights:

$$w_i(t+1) = w_i(t) + d^{(m)} x_i^{(m)}(t).$$

4. Repeat 2 and 3 until all the samples in the training set are correctly classified.

The main limitation of this procedure is that its convergence is guaranteed only for linearly separable[1] data, which means that a perceptron cannot even learn the boolean XOR function [50] (Figure 3.3).

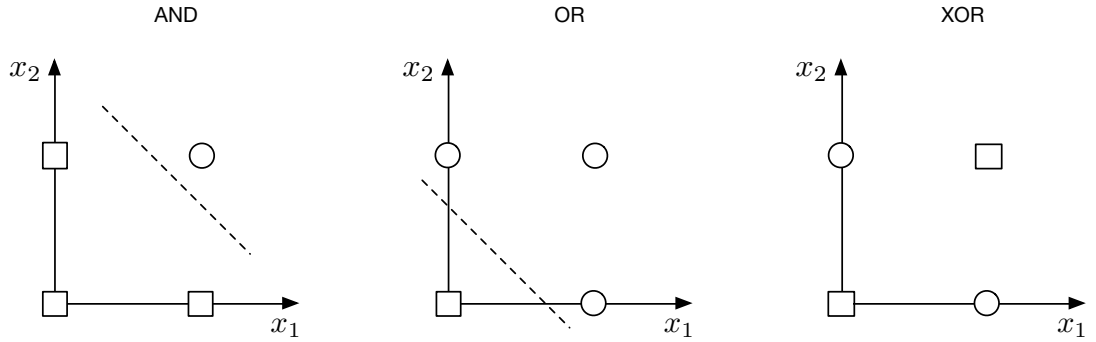


Figure 3.3. Limitations of the perceptron. It is possible to see that the perceptron can learn the AND and the OR functions (since they are linearly separable), but not the XOR function.

3.2 Multilayer Perceptron

A way to classify not linearly separable data is by using a model known as multilayer perceptron (MLP) where the responses of the perceptron units (or nodes) are combined in multiple layers. In particular, in a MLP we can find at least three layers of nodes (input, output and one or more hidden layers) which are able to learn features at different levels of abstraction. The layers are generally called fully connected because each node of a layer is connected with all the nodes of the previous layer (Figure 3.4).

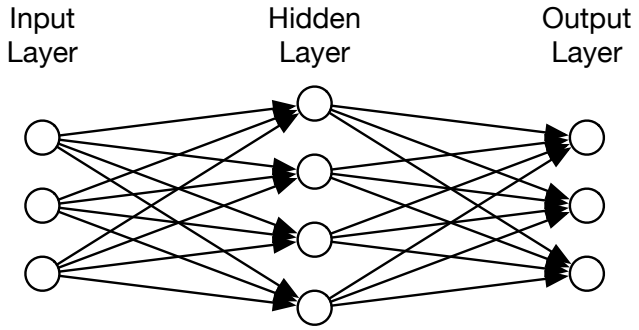


Figure 3.4. Multilayer perceptron.

More formally, a MLP consists of $N+1$ units $\bar{x} = \begin{bmatrix} x_0 & \dots & x_N \end{bmatrix}^T$ (with $x_0 = 1$) as input, an hidden layer with $M+1$ units $\bar{z} = \begin{bmatrix} z_0 & \dots & z_M \end{bmatrix}^T$ (with $z_0 = 1$), and an output layer with K units $\bar{y} = \begin{bmatrix} y_1 & \dots & y_K \end{bmatrix}^T$. We can denote with $W =$

[1] Two sets of points are linearly separable if there exists at least one hyperplane in the Euclidean space where the points are defined that separates the two sets.

$\begin{bmatrix}\bar{w}_0 & \dots & \bar{w}_M\end{bmatrix}$ the $(N+1)\times(M+1)$ matrix of the weights between the input and the hidden layer, and with $V = \begin{bmatrix}\bar{v}_1 & \dots & \bar{v}_K\end{bmatrix}$ the $(M+1)\times K$ matrix of the weights between the hidden and the output layers. Then, for the generic hidden unit z_j we have:

$$z_j = g(\bar{w_j}^T \bar{x}), \tag{3.3}$$

and for the generic output unit y_h:

$$y_h = f(\bar{v_h}^T \bar{z}), \tag{3.4}$$

where g and f are two activation functions.

The configuration of a MLP does not allow to use the learning algorithm seen for the perceptron because it does not provide a way to update the weights of the layers that are not the output one. The solution is backpropagation [45, 85, 73], but before we need to introduce the concept of loss function.

In order to solve a classification or a regression problem with a MLP, we consider a loss function that associates a cost to the specific output of the network. For example, in case of regression with a single output, $f(x) = x$ and the loss function generally used is the squared error:

$$E(W, V) = \frac{1}{2}\sum_m (r^{(m)} - y^{(m)})^2, \tag{3.5}$$

where $r^{(m)}$ is the desired output for the $\bar{x}^{(m)}$ input and $y^{(m)}$ is the actual output. For binary classification, $f(x) = \frac{1}{1+e^{-x}}$, known as sigmoid function, and the loss function is the cross-entropy error function:

$$E(W, V) = -\sum_m [r^{(m)} \log y^{(m)} + (1 - r^{(m)}) \log (1 - y^{(m)})]. \tag{3.6}$$

This can be extended to multiclass classification by using a softmax function for f, $f(\bar{x})_l = \frac{e^{x_l}}{\sum_{i=1}^{L} e^{x_i}}$ for $l = 1 \dots L$, and again the cross-entropy error function as loss function.

Given the loss function, the problem now can be seen as an optimisation problem, where we want to find the parameters W and V that minimise the error. We can use backpropagation together with an optimisation method like gradient descent. The idea is to forward propagate the information through all the layers of the network until the output, calculate the error and propagate it backwards. In particular, backpropagation allows to compute the gradient of the loss function with respect to the weights (the elements of V and W) using the chain rule[2], and then use it to update the weights via gradient descent. In its simplest form, gradient descent updates the weights as it follows:

$$w(t+1) = w(t) - \Delta w, \tag{3.7}$$

with $\Delta w = \eta \frac{\partial E}{\partial w}$, where η is a hyperparameter called learning rate that controls how much the parameters should change.

[2]The derivation of the backpropagation algorithm is not reported, but it can be found in [2].

In order to accelerate the training, some extensions to the classical gradient descent method have been proposed. For example, it is possible to include a momentum term that uses the gradient to update the velocity instead of the position of the point in weight space [73]:

$$\begin{aligned} v(t+1) &= \mu v(t) - \eta \frac{\partial E}{\partial w} \\ w(t+1) &= w(t) + v(t+1), \end{aligned} \tag{3.8}$$

where μ is the momentum parameter, generally set to 0.9.

Recently, a new method for gradient optimisation, considered the default algorithm to use [1], has been proposed by Kingma and Ba [34]: Adam. As reported in [1], the update can be seen as:

$$\begin{aligned} m(t+1) &= \beta_1 m(t) + (1-\beta_1) g(t+1) \\ v(t+1) &= \beta_2 v(t) + (1-\beta_2) g^2(t+1) \\ w(t+1) &= w(t) - \eta \frac{m(t+1)}{(\sqrt{v(t+1)} + \epsilon)}, \end{aligned} \tag{3.9}$$

where $g(t)$ indicate the gradient of the loss function at step t and the recommended values of the parameters are $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$.

When the training set is large, computing the loss function for all the samples is impractical, then the gradient is computed over batches, and the approach is called mini-batch gradient descent. When the mini-batch consists of only one element the procedure is known as stochastic gradient descent (SDG). In the literature SDG is used as synonym of mini-batch gradient descent when the batch size is reported.

The issue that arises now with gradient descent is that it is not possible to use the sign function as activation function, because it is not differentiable at 0. In practice, other non linear functions are used:

- The sigmoid function, σ, was generally used in the past, but it presents two drawbacks [1] . First, it has two saturation regions (one for large positive input and one for large negative input), where the gradient is close to zero making it difficult to update the weights. Then, the outputs of the sigmoid function is not zero-centred which is undesirable because it may bias the weights' update.

- The second issue of the sigmoid function can be solved by adopting the hyperbolic tangent (or tanh) defined as:

$$\tanh(x) = 2\sigma(2x) - 1. \tag{3.10}$$

- The first issue of the sigmoid function can be partially solved by using the rectified linear unit (or ReLU) defined as:

$$f(x) = \max(0, x), \tag{3.11}$$

where the saturation region still occurs for large negative input. For this reason a Leaky ReLU (or l-ReLU) activation function can be adopted, where a slope is used for the negative input, like:

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{otherwise} \end{cases} \tag{3.12}$$

with α set to a small value.

Figure 3.5 shows some activation functions used in neural networks.

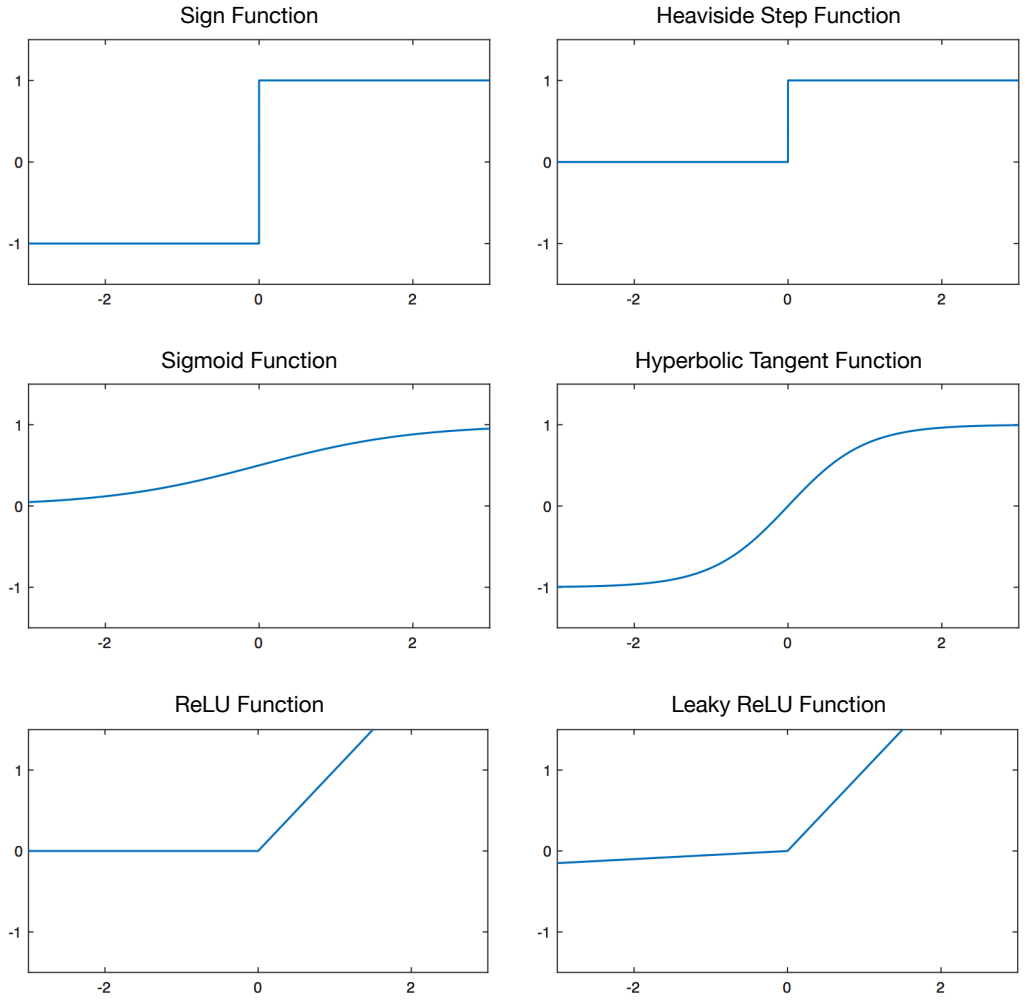


Figure 3.5. Some activation functions used in neural networks.

3.3 Convolutional Neural Networks

We have seen that MLP allows to learn a mapping between an input vector and an output vector, but we have not said if it is able to learn any mapping function. In their work, Hornik et al. [25] showed that a MLP even with one hidden layer having a finite number of nodes is a universal approximator. In other words, *"standard*

multilayer feedforward networks are capable of approximating any measurable function to any desired degree of accuracy, in a very specific and satisfying sense" [25]. However, the function to learn may require a high number of parameters, making the training of the network hard. We can think to reduce the number of trainable parameters, by adopting a different architecture known as convolutional neural network (CNN). CNNs are used today in many different tasks, mainly in the computer vision community, in particular after that Krizhevsky et al. [39] adopted them in the ILSVRC-2012 competition [74], outperforming the other methods with a high margin. However, CNNs were shown to be a technique able to reach a high accuracy in classification tasks as far back as in 1989, when LeCun et al. [41] used a CNN to classify handwritten digits.

The basic idea behind CNNs is to combine local features in order to obtain features with a higher level of abstraction. In practice, this is realised by extracting the same features in the different locations of the input using the weight sharing concept: a hidden node is not connected to all the input nodes, but only to a neighbourhood of input nodes and the connection weights do not change when a different neighbourhood region of the input is considered. This is equivalent to the application of a convolutional filter with learnable weights to the input (Figure 3.6). For this reason, such a layer is called convolutional layer.

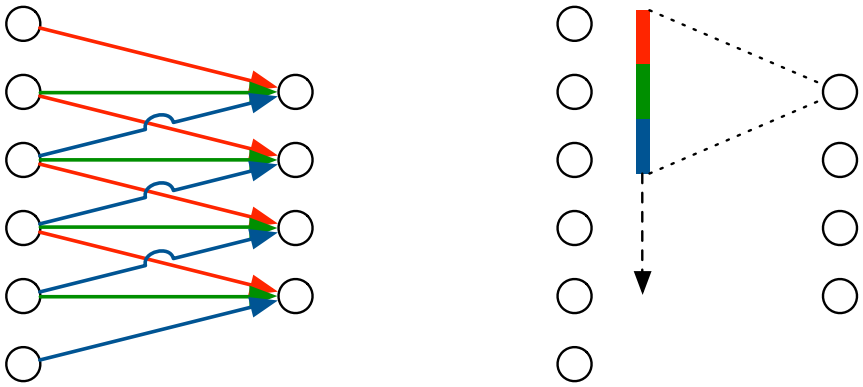


Figure 3.6. Convolutional layer. On the left, the connections with the same colour share the weights. On the right, the same concept is represented as a convolution between the input and a 3×1 filter with a stride of 1. It is possible to apply more filters to the input in order to detect different kinds of feature.

A convolution can also be seen as a matrix operation [9]. Suppose that we want to convolve a 4×4 input matrix, I, with a 3×3 filter, W, using a unit stride. If we flatten the matrix I into a 16×1 vector $\bar{i}$, the 4×1 flatten output of the convolution can be represented as:

$$\bar{o} = C \times \bar{i}, \tag{3.13}$$

with:

$$C = \begin{pmatrix} w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} \end{pmatrix}$$

where $w_{i,j}$ is the element of the filter in the position (i, j), as in [9]. With this representation, *"the backward pass is easily obtained by transposing C"* [9]. In some situations, we would like to upscale the input (e.g. in the generator of a GAN, as we will see in Section 3.4.4). In this case, we use C^T to compute the forward pass, and $(C^T)^T = C$ for the backward pass, performing the so called transposed convolution[3] [9].

Generally, in a CNN we also find a pooling layer, which performs a subsampling of the previous layer output. The usual way is to extract the maximum value of very small regions of the previous layer (max-pooling). This has two advantages: first, it reduces the number of learnable parameters; then, it makes the representation robust to small local transformations [59].

To summarise, the architecture of a CNN has an alternation between convolutional and pooling layers, usually followed by a fully connected layer to perform classification (Figure 3.7).

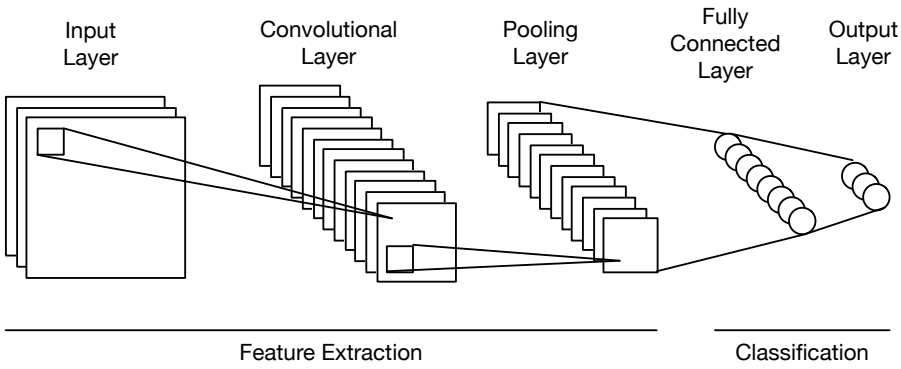


Figure 3.7. Representation of a CNN for a RGB image as input. In this case only one convolutional layer and one pooling layer are shown, but generally we have multiple layers to extract higher order features. Inspired by [22].

One of the issues that deep CNNs have is that the network tends to follow the noise, especially when the training set is small, a problem known as overfitting. This can be prevented in several ways, for example by stopping the training when the performance on the validation set falls or by using a regularisation term which penalises large weights. Another effective way to address the issue is dropout [79]. The basic idea is to randomly drop the nodes of the network with a probability p (usually $p = 0.5$) during training. Intuitively, this technique allows to generalise better because the nodes of the network need to learn a representation which is less dependent from the other nodes.

Another problem that arises with CNNs is the internal covariate shift. Basically, when the information flows through a deep network, the weights adjust the data so that its distribution in each layer can change during training. This issue is solved by normalising the layer inputs, a method known as batch normalisation [27].

[3] A transposed convolution is also called deconvolution or fractionally strided convolution.

3.4 Deep Generative Models

The neural network models we have seen so far are discriminative models, because their goal is to model the posterior probability, $p(y|x)$, of the label y given the input x, or to learn directly a mapping between the input space and the labels [56]. On the other hand, generative models try to learn $p(x|y)$ [55], which can be used together with the class priors $p(y)$ to calculate $p(y|x)$ via the Bayes rule [55]:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}. \tag{3.14}$$

Even though deep discriminative models are the ones that have obtained more successes in a variety of tasks, deep generative models have the advantage of capture the underlying structure of the data. In particular, they can provide (explicitly or not) an estimator, p_{model}, of the data distribution p_{data} and, as a consequence, generate samples from p_{model} [16].

Since the focus of this project is on generative adversarial networks (GANs), we are going to present them in details. Before, we quickly introduce the concept of maximum likelihood (ML) estimation, and the fully visible belief networks (FVBNs) and the variational autoencoders (VAEs) because they are the other two most popular deep generative models, and it is relevant to show why GANs are a valid alternative to them.

3.4.1 Maximum Likelihood Estimation

As in [16], we consider the version of the models that use the maximum likelihood (ML) principle in order to simplify the discussion.

In our case, we can consider a training set as a sample of M independent and identical distributed observations, $\bar{x}^{(m)}$, from the distribution p_{data}. In ML, we assume that $p_{data}(\bar{x}) = p_{model}(\bar{x}|\bar{\theta}_0)$, where $\bar{\theta}_0$ is an unknown vector of parameters. If we define the likelihood as:

$$l(\bar{\theta}|\bar{x}^{(1)}, \dots, \bar{x}^{(M)}) = p(\bar{x}^{(1)}, \dots, \bar{x}^{(M)}|\bar{\theta}), \tag{3.15}$$

then, the ML method consists of estimating $\bar{\theta}_0$ by finding the $\bar{\theta}$ that maximises the likelihood:

$$\bar{\theta}^* = \arg\max_{\bar{\theta}} l(\bar{\theta}|\bar{x}^{(1)}, \dots, \bar{x}^{(M)}), \tag{3.16}$$

which can be rewritten as:

$$\bar{\theta}^* = \arg\max_{\bar{\theta}} \prod_{m=1}^{M} p_{model}(\bar{x}^{(m)}|\bar{\theta}), \tag{3.17}$$

for independent and identically distributed observations. For convenience, the log-likelihood is generally used:

$$\mathcal{L}(\bar{\theta}|\bar{x}^{(1)}, \dots, \bar{x}^{(M)}) = \log l(\bar{\theta}|\bar{x}^{(1)}, \dots, \bar{x}^{(M)}) = \sum_{m=1}^{M} \log p_{model}(\bar{x}^{(m)}|\bar{\theta}). \tag{3.18}$$

We can interpret the ML estimation as the minimisation of the Kullback-Leibler (KL) divergence between p_{data} and p_{model}, $D_{KL}(p_{data}\|p_{model})$. The KL divergence is defined as $D_{KL}(p(x)\|q(x)) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} \, \mathrm{d}x$. Since we do not have access directly to p_{data}, we define an empirical data distribution p^*_{data} that approximates p_{data} [16]. From the definition we can write:

$$D_{KL}(p^*_{data}(\bar{x}) \,||\, p_{model}(\bar{x}|\bar{\theta})) = \int_{-\infty}^{\infty} p^*_{data}(\bar{x}) \, \log \frac{p^*_{data}(\bar{x})}{p_{model}(\bar{x}|\bar{\theta})} \, \mathrm{d}x \tag{3.19}$$

$$= \int_{-\infty}^{\infty} p^*_{data}(\bar{x}) \log p^*_{data}(\bar{x}) \, dx$$

$$- \int_{-\infty}^{\infty} p^*_{data}(\bar{x}) \log p_{model}(\bar{x}|\bar{\theta}) \, dx \tag{3.20}$$

$$= - \mathrm{H}[p^*_{data}(\bar{x})] + \mathrm{H}[p^*_{data}(\bar{x}), p_{model}(\bar{x}|\bar{\theta})], \tag{3.21}$$

where $\mathrm{H}[p^*_{data}(\bar{x})]$ is the entropy of p^*_{data} and does not depend on $\bar{\theta}$, so it can be ignored. $\mathrm{H}[p^*_{data}(\bar{x}), p_{model}(\bar{x}|\bar{\theta})]$ represents the cross-entropy of p^*_{data} and p_{model}, and can be seen as $\mathrm{E}_{p^*_{data}}[-\log p_{model}]$. Now, from Equation 3.18, we can write:

$$\bar{\theta}^* = \arg\max_{\bar{\theta}} \frac{1}{M} \sum_{m=1}^{M} \log p_{model}(\bar{x}^{(m)}|\bar{\theta}) \tag{3.22}$$

$$= \arg\min_{\bar{\theta}} \frac{1}{M} \sum_{m=1}^{M} - \log p_{model}(\bar{x}^{(m)}|\bar{\theta}), \tag{3.23}$$

where $\frac{1}{M} \sum_{m=1}^{M} - \log p_{model}(\bar{x}^{(m)}|\bar{\theta})$ converges to $\mathrm{E}_{p^*_{data}}[-\log p_{model}]$, according to the strong law of large numbers. Thus, as reported in [16], *"minimising the KL divergence between p^*_{data} and p_{model} is exactly equivalent to maximising the log-likelihood of the training set"*.

3.4.2 Fully Visible Belief Networks

FVBNs [13] are models that define $p_{model}(\bar{x}, \bar{\theta})$ explicitly, making the maximisation of the likelihood easy to perform. The main issue is related to the design of a model that is computationally tractable, and in the specific case of FVBNs this is obtained by using the chain rule of probability as shown in [16]:

$$p_{model}(\bar{x}) = \prod_{i=1}^{n} p_{model}(x_i|x_1, \ldots, x_{i-1}). \tag{3.24}$$

FVBNs are the basis of some state-of-the-art generative models like WaveNet [84], a deep neural network that generates raw audio waveform. The drawback of this approach is that the generation is really slow, because each sample x_i needs to be generated separately using a deep neural network [16]. GANs, on the other hand, do not have this limitation, and the generation of samples can be parallelised.

3.4.3 Variational Autoencoders

VAEs [35, 69] belong to another class of generative models that provide a computationally intractable density function, but the likelihood is maximised using deterministic approximations [16]. In variational methods a lower bound to the likelihood is maximised [16]:

$$L(\bar{x}|\bar{\theta}) \leq \log p_{model}(\bar{x}|\bar{\theta}). \tag{3.25}$$

The main problem related with VAEs is that the gap between L and the true likelihood can be such that the model learns a distribution different from p_{data} [16]. Even though the likelihood obtained with VAEs is high, they tend to produce lower quality samples [16]. For example, if we consider the generation of image samples, they are blurry due to the minimisation of the mean square error as the reconstruction loss.

3.4.4 Generative Adversarial Networks

GANs [17] differ from the two previously mentioned generative models because they do not provide a probability distribution explicitly, but they directly sample from p_{model} [16].

The GAN framework consists of two players that play a game against each other, where a generator (G) tries to capture the data distribution and draw samples from it, and a discriminator (D) tries to distinguish the samples presented to it between real (coming from p_{data}) and fake (coming from p_{model}). Both players improve in this game because the training process for G consists of fooling D (Figure 3.8).

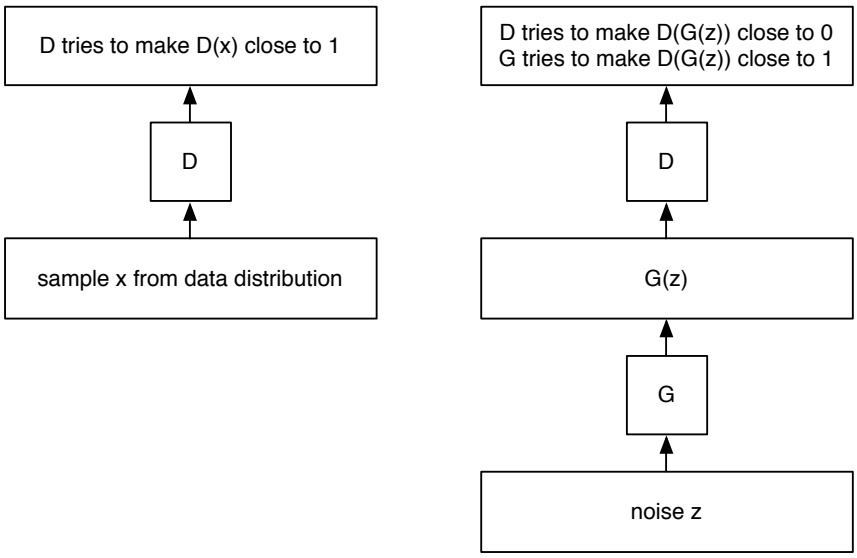


Figure 3.8. GAN framework. Inspired by [16].

G and D are two functions both differentiable with respect to their inputs and their parameters, and generally they are neural networks [16]. The architectures mostly used today [16] are based on deep convolutional GAN (DCGAN) [63], whose generator is shown in Figure 3.9. DCGAN allows to address some instability issues

that occur in the training phase of GANs applied to high resolution images. The key ideas behind the approach are three: the application of batch normalisation to most of the networks' layers, the absence of pooling layers as done in [78], and the adoption of the Adam optimiser for training.

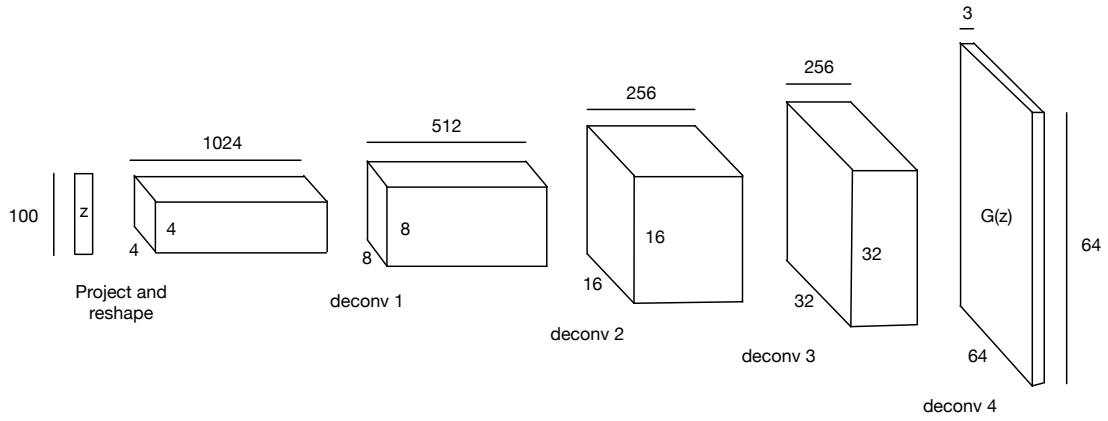


Figure 3.9. DCGAN generator. All the deconvolutional layers use 5×5 filters with a stride of 2. Inspired by [63].

Using a notation similar to the one in [16], the inputs of D and G are an observed variable $\bar{x}$ and a latent noise variable $\bar{z}$, respectively, and their parameters are denoted as $\bar{\theta}^{(D)}$ and $\bar{\theta}^{(G)}$, respectively. In the training procedure, a simultaneous gradient descent is adopted to minimise the loss function of the discriminator, $J^{(D)}(\bar{\theta}^{(D)}, \bar{\theta}^{(G)})$, by updating $\bar{\theta}^{(D)}$, and the loss function of the generator $J^{(G)}(\bar{\theta}^{(D)}, \bar{\theta}^{(G)})$, by updating $\bar{\theta}^{(G)}$. The loss function of D is [16]:

$$J^{(D)}(\bar{\theta}^{(D)}, \bar{\theta}^{(G)}) = -\frac{1}{2}\,\mathrm{E}_{\bar{x}\sim p_{data}} \log D(\bar{x}) - \frac{1}{2}\,\mathrm{E}_{\bar{z}} \log(1 - D(G(\bar{z}))), \tag{3.26}$$

while for G we can use [16]:

$$J^{(G)}(\bar{\theta}^{(D)}, \bar{\theta}^{(G)}) = -J^{(D)}(\bar{\theta}^{(D)}, \bar{\theta}^{(G)}), \tag{3.27}$$

in order to have a zero-sum game, also called min-max because its *"solution involves minimisation in an outer loop and maximisation in an inner loop"* [16]:

$$\bar{\theta}^{(G)*} = \arg\min_{\bar{\theta}^{(G)}} \max_{\bar{\theta}^{(D)}} V(\bar{\theta}^{(D)}, \bar{\theta}^{(G)}), \tag{3.28}$$

where $V(\bar{\theta}^{(D)}, \bar{\theta}^{(G)}) = -J^{(D)}(\bar{\theta}^{(D)}, \bar{\theta}^{(G)})$. In practice, $J^{(G)}(\bar{\theta}^{(D)}, \bar{\theta}^{(G)})$ as seen in Equation 3.27 is not used, because $\log(1 - D(G(\bar{z})))$ tends to saturate in the first stages of training. This occurs because G is poor and it is easy for D to distinguish between real and fake samples [17]. An alternative loss function for G, heuristically motivated, is [16]:

$$J^{(G)}(\bar{\theta}^{(D)}, \bar{\theta}^{(G)}) = -\frac{1}{2}\,\mathrm{E}_{\bar{z}} \log D(G(\bar{z})). \tag{3.29}$$

It is also possible to choose a $J^{(G)}$ that minimises the KL divergence between p^*_{data} and p_{model} (and equivalently performs a maximum likelihood learning as seen in Section 3.4.1), such as [16]:

$$J^{(G)}(\bar{\theta}^{(D)}, \bar{\theta}^{(G)}) = -\frac{1}{2}\,\mathrm{E}_{\bar{z}} \exp\left(\sigma^{-1}(D(G(\bar{z})))\right). \tag{3.30}$$

One of the advantage of this adversarial training procedure is that the generated samples have generally a better quality if compared to the models that adopt a mean square error-based loss function (Figure 3.10).

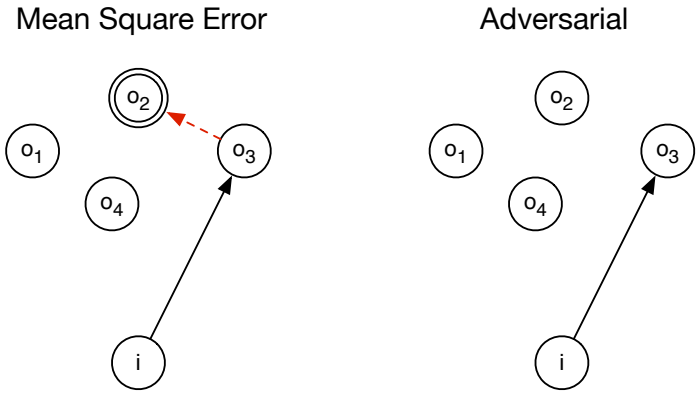


Figure 3.10. Advantages of the adversarial loss over the mean square error. In the training process of models based on mean square error, we have an input, i, which is associated with a desired output, o_2. If the o_j, $j = 1, \ldots, 4$, are all valid outputs for i and during training the model maps i to o_3, then we have a certain error, represented by the red arrow. This induces i to be mapped to the mean of the valid outputs. This is the reason why images produced with this method are blurry. On the other hand, in GANs, when G generates o_3 from i, the discriminator accepts it as a valid output, since input and output are not paired. Inspired by [15].

However, the main drawback is that solving the game between G and D means finding a Nash equilibrium, which is harder than optimising an objective function [16]. For this reason, several techniques are used to have stable models [6]. For example, sometimes D shows a really high confidence in classifying the samples between real and fake, making the generator gradient large [16]. A possible solution to this problem is to reduce the confidence of D to choose the correct class by one-sided label smoothing [75], which consists of replacing the target of real data from 1 to $1 - \alpha$ (generally $\alpha = 0.1$). Another way to improve GANs is by using labels when available. Many different approaches are possible. For example, Springenberg [77] proposes a multi-class output for D, that should produce uncertain predictions when generated samples are presented to it, and high confidence about the class when real data is the input. Odena [57] uses a discriminator that can predict $N + 1$ classes, the N classes of the training set and an extra one for the fake samples. Mirza and Osindero [51] propose a conditional model of GAN (cGAN) where both G and D are conditioned to generate samples of specific classes (Figure 3.11).

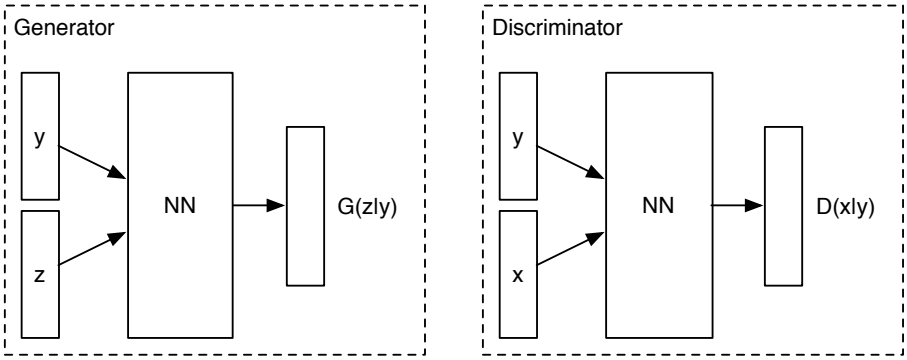


Figure 3.11. Conditional GAN. NN indicates that the architectures of both G and D are neural networks. Inspired by [51].

cGANs are useful because they allow to control what we would like to generate. Nevertheless, most of the data is unlabelled, making supervised techniques not suitable for the task. On the other hand, if we use the original GAN framework, no restriction are imposed on how G should use the noise $\bar{z}$. This means that each dimension of $\bar{z}$ is not responsible for a semantically meaningful aspect of the data. In order to learn a disentagled representation of the data in an unsupervised way, Chen et al. [5] propose a GAN extension, InfoGAN, that maximises the mutual information between a subset of the latent variables and the observations. In the original GAN, the minimax objective is:

$$\min_{\bar{\theta}^{(G)}} \max_{\bar{\theta}^{(D)}} V(\bar{\theta}^{(D)}, \bar{\theta}^{(G)}), \tag{3.31}$$

where:

$$V(\bar{\theta}^{(D)}, \bar{\theta}^{(G)}) = \mathrm{E}_{\bar{x} \sim p_{data}}[\log D(\bar{x})] + \mathrm{E}_{\bar{z}}[\log(1 - D(G(\bar{z})))]. \tag{3.32}$$

In InfoGAN, the input of G is divided into two parts: $\bar{z}$, which represents the noise, and $\bar{c}$, the latent code that has a correspondence with the semantic aspects of the data. If we train this model using the original GAN, G can ignore $\bar{c}$, so the mutual information between $\bar{c}$ and $G(\bar{z}, \bar{c})$ is used as a regularisation term. The mutual information between two random variables X and Y is [5]:

$$I(X;Y) = H(X) - H(Y|X) = H(Y) - H(Y|X), \tag{3.33}$$

so it is the reduction of uncertainty in X given Y. In our case, maximising $I(\bar{c}; G(\bar{z}, \bar{c}))$ allows not to let G ignore $\bar{c}$ when it generates a sample. Then, the minimax objective becomes [5]:

$$\min_{\bar{\theta}^{(G)}} \max_{\bar{\theta}^{(D)}} V_I(\bar{\theta}^{(D)}, \bar{\theta}^{(G)}), \tag{3.34}$$

with:

$$V_I(\bar{\theta}^{(D)}, \bar{\theta}^{(G)}) = V(\bar{\theta}^{(D)}, \bar{\theta}^{(G)}) - \lambda I(\bar{c}; G(\bar{z}, \bar{c})). \tag{3.35}$$

Maximising the mutual information is not easy, since it requires the posterior probability $p(\bar{c}|\bar{x})$, but it is possible to define a distribution $Q(\bar{c}|\bar{x})$ that approximates $p(\bar{c}|\bar{x})$ and allows to obtain a lower bound $L_I(G,Q)$ of $I(\bar{c};G(\bar{z},\bar{c}))$, as shown in [5]. Thus, the minimax game for InfoGAN has the following objective [5]:

$$\min_{\bar{\theta}^{(G)},\bar{\theta}^{(Q)}} \max_{\bar{\theta}^{(D)}} V_{InfoGAN}(\bar{\theta}^{(D)},\bar{\theta}^{(G)},\bar{\theta}^{(Q)}), \tag{3.36}$$

where:

$$V_{InfoGAN}(\bar{\theta}^{(D)},\bar{\theta}^{(G)},\bar{\theta}^{(Q)}) = V(\bar{\theta}^{(D)},\bar{\theta}^{(G)}) - \lambda L_I(G,Q). \tag{3.37}$$

The adoption of GANs for computer vision and image processing tasks is extensive [28, 63, 5, 42, 90], but their use in speech-related tasks is very limited. To our knowledge, the only relevant work is [52], where a deep visual analogy network [66] is used as a generator to perform voice conversion. The results are promising, even though no objective measures are used to evaluate the quality or the intelligibility of the generated samples. Another related work is the one by Mogren [53] that proposes a generative adversarial model to train a recurrent neural network for classical music generation.

CHAPTER 4

GENERATIVE ADVERSARIAL NETWORKS FOR SPEECH ENHANCEMENT

Speech enhancement is crucial in several applications, including automatic speaker verification (ASV), speech recognition, and hearing assistive devices. The general objective of speech enhancement is to reduce the noise from a degraded signal in order to improve its quality and intelligibility, when the receiver is a human user, or the robustness of the system to noise, when the receiver is an automatic speech system. Among the approaches that can be adopted to achieve this goal, we can mention statistical methods, such as short-time spectral amplitude minimum mean square error (STSA-MMSE) [46], and deep learning techniques, like deep neural networks (DNNs) [36, 87], deep autoencoders [47], and CNNs [61].

This Chapter will give a detailed description of the work that we have done in order to use cGAN for speech enhancement. This has lead to the submission of a paper [49] to the Interspeech 2017 conference that can be found in the Appendix B.

4.1 Problem Formulation

The speech enhancement task can be formulated formally by the following model [37]:

$$y(n) = x(n) + d(n), \tag{4.1}$$

where $x(n)$ and $d(n)$ are the speech and the uncorrelated noise signals, and $y(n)$ is the observed noisy signal. The objective of speech enhancement is to provide an estimate, $\hat{x}(n)$, for $x(n)$, given $y(n)$.

Equation 4.1 can also be written using the T-F representations of the signals:

$$Y(n, \omega) = X(n, \omega) + D(n, \omega). \tag{4.2}$$

In this case, we can estimate $X(n, \omega)$ and then reconstruct the signal as seen in Section 2.3 (Figure 4.1).

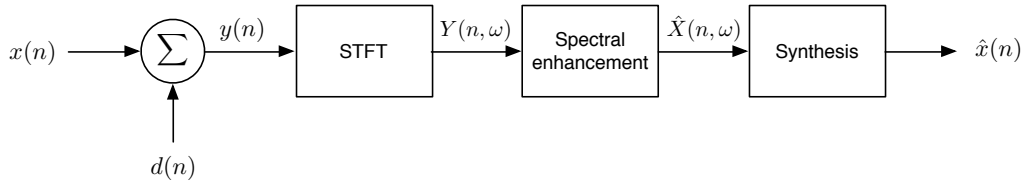


Figure 4.1. Steps needed to perform spectral enhancement. Inspired by [7].

4.2 System Overview

Our system is based on the work of Isola et al. [28], that proposed a cGAN framework (Pix2Pix) for image-to-image translation tasks. Our goal is to use Pix2Pix to learn a mapping between a noisy spectrogram and a clean one. An overview of the system is shown in Figure 4.2, where we distinguish between a training phase and a test phase.

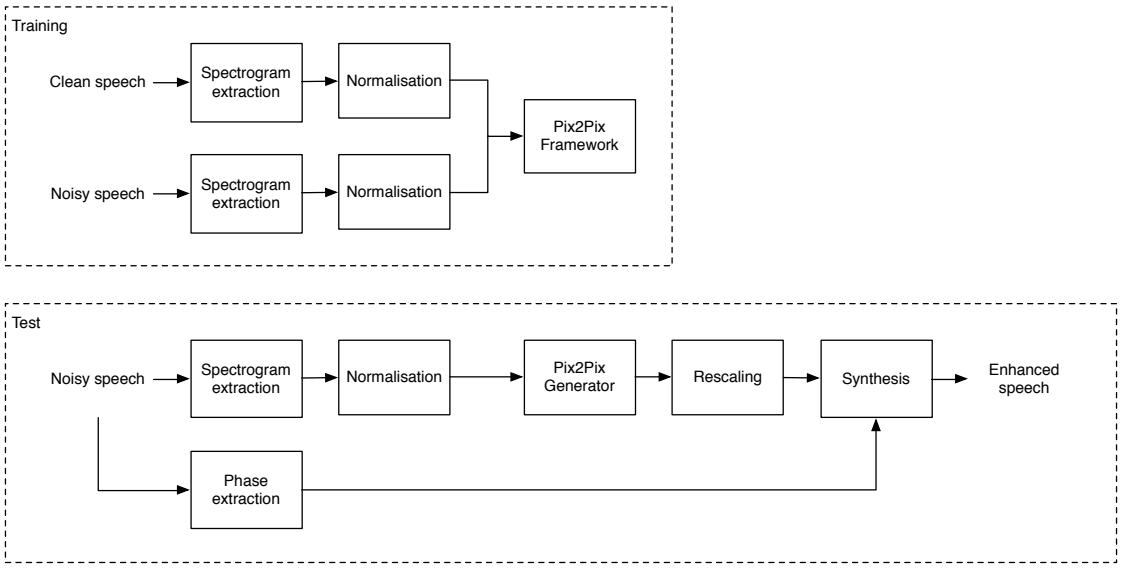


Figure 4.2. Block diagram of the Pix2Pix-based speech enhancement system.

During training, we have two signals available: $x(n)$ and $y(n)$. From both of them we compute the spectrogram ($|X(n, \omega)|$ and $|Y(n, \omega)|$) using a 512-point STFT with a hamming window size of 32 ms and a hop size of 16 ms. For speech signals having a sample rate of 16 kHz, the resolution of the spectrogram is $16\,\text{kHz}/512 = 31.25\,\text{Hz}$ per frequency bin. Due to the symmetry property of the DFT (Equation 2.3), we can consider only the first 257 elements of each frame of the spectrogram. Then, we

normalise the spectrogram in order to have values within the range $[-1, 1]$. Specifically, for the value v of each element of the original (clean or noisy) spectrogram, we apply the following linear mapping to find the normalised value u:

$$u = 2\,\frac{v - a}{b - a} - 1, \tag{4.3}$$

where a and b are the minimum and the maximum values of the noisy spectrogram respectively. Furthermore, since we decided to have G and D with an input size of $256 \times 256 \times k$ (where $k = 1$ for G, and $k = 2$ for D) to make the design of the architecture simple, we concatenate all the spectrograms and then perform a split every 256 frames. We also remove the highest frequency bin, which has a small impact on the T-F representation of the signal, being just the highest 31.25 Hz band of it. At this point, we can use the preprocessed signals in Pix2Pix.

We have seen before (Section 3.4.4) how a cGAN works. Pix2Pix differs from the original formulation because it does not use the noise variable $\bar{z}$. The reason is that G learns to ignore it. In order to produce some stochasticity in the output, the noise is introduced in the form of dropout. Unfortunately this approach failed to achieve the expected outcome [28]. However, our interest is in learning a mapping from the spectrogram of a noisy speech to an enhanced counterpart, so this is a minor issue.

In Pix2Pix the adversarial loss is combined with the L1 distance between the output of G and the ground truth. The use of different losses in GANs have been reported also in other works, where generally the L2 distance [62] or a perceptual loss [42, 90] is adopted. The choice of the L1 distance in Pix2Pix is motivated by the less blurred output if compared to the L2 distance [28].

Also, Isola et al. [28] used a U-Net [71] for G, where the skip connections between the encoder and the decoder avoid the bottleneck for the information flow represented by the innermost layer of the network. On the other hand, D is a PatchGAN [28] with the objective of classifying all the patches of the input image as real or fake. In this way D can capture the high-frequency of the data, leaving to the L1 loss the task of modelling the low frequency structure.

Our implementation of the framework is based on [44], where the main differences with [28] are two:

- The use of 5×5 filters for the convolutions and the translated convolutions.
- The adoption of a fully-connected layer that is fed into a single sigmoid output for the last layer of D.

GANs are trained according to the parameters reported in [28]. In particular, we use SDG with the Adam optimiser (see Section 3.2) for 10 epochs and a batch size of 1. We initialise the weights from a normal distribution with zero mean and a standard deviation of 0.02. The scaling factor used to add the L1 loss to the GAN loss is 100. Also, as in [44], we update G twice per each iteration in order to reduce the convergence speed of D. Figure 4.3 illustrates how we train G and D in our work, and Table 4.1 shows their configuration.

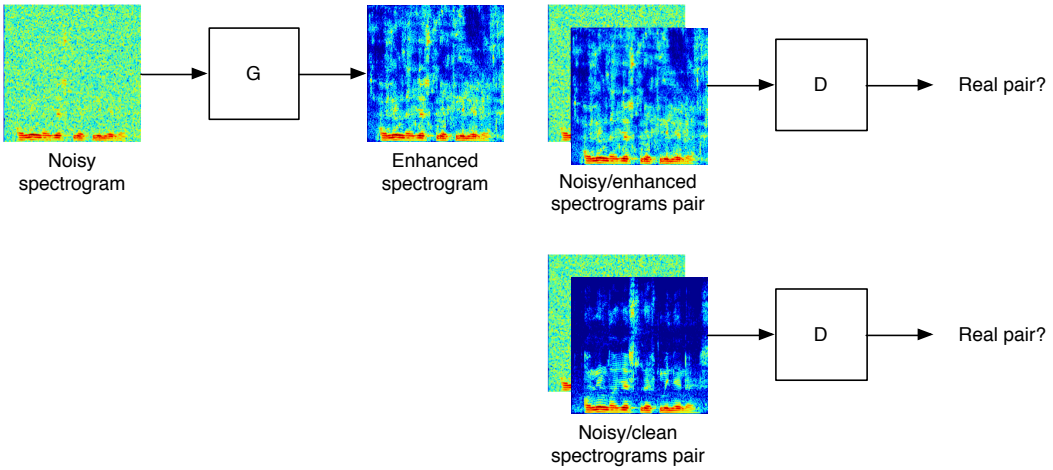


Figure 4.3. Training of the Pix2Pix framework for speech enhancement. G generates the enhanced spectrogram from a noisy one, while D tries to classify its input as a real pair (noisy and clean spectrograms) or a fake pair (noisy and enhanced spectrograms).

Generator			
	Layer		Dimension
Encoder	Input	-	(256, 256, 1)
	E1	conv	(128, 128, 64)
	E2	l-ReLU/conv/BN	(64, 64, 128)
	E3	l-ReLU/conv/BN	(32, 32, 256)
	E4	l-ReLU/conv/BN	(16, 16, 512)
	E5	l-ReLU/conv/BN	(8, 8, 512)
	E6	l-ReLU/conv/BN	(4, 4, 512)
	E7	l-ReLU/conv/BN	(2, 2, 512)
	E8	l-ReLU/conv/BN	(1, 1, 512)
Decoder	D1	ReLU/deconv/dropout/SC	(2, 2, 1024)
	D2	ReLU/deconv/dropout/SC	(4, 4, 1024)
	D3	ReLU/deconv/dropout/SC	(8, 8, 1024)
	D4	ReLU/deconv/BN/SC	(16, 16, 1024)
	D5	ReLU/deconv/BN/SC	(32, 32, 512)
	D6	ReLU/deconv/BN/SC	(64, 64, 256)
	D7	ReLU/deconv/BN/SC	(128, 128, 128)
	D8	ReLU/deconv/tanh	(256, 256, 1)

Discriminator		
Layer		Dimension
Input	-	(256, 256, 2)
H1	conv	(128, 128, 64)
H2	l-ReLU/conv/BN	(64, 64, 128)
H3	l-ReLU/conv/BN	(32, 32, 256)
H4	l-ReLU/conv/BN	(32, 32, 512)
H5	l-ReLU/FC/sigmoid	(1)

Table 4.1. Network configurations for the generator and the discriminator. BN denotes batch normalisation, SC a skip connection, and FC a fully connected layer.

At test time, we extract the spectrogram and the phase of the noisy signal, perform the normalisation (Equation 4.3), and use the trained G to enhance the spectrogram. As for training, we use 256×256 spectrograms, but this time we zero-pad the spectrogram of each test sample, and not concatenate different ones. Then we rescale the spectrogram with the following equation:

$$\hat{u}' = (\hat{u} + 1)\,\frac{b - a}{2} + a, \tag{4.4}$$

where $\hat{u}$ and $\hat{u}'$ are the values of the normalised and the rescaled spectrogram respectively. Using the estimated spectrogram (with a row of zeros for the previously removed high frequency bins) and the noisy phase, we can reconstruct $\hat{x}(n)$.

4.3 Experiments

In this section we are going to describe how we perform the experiments in order to evaluate the performance of the system.

4.3.1 Evaluation Metrics

In the assessment of speech signals, two different aspects are generally considered: quality and intelligibility. Speech quality indicates how good a speech signal is perceived by a human user, while speech intelligibility refers to how comprehensible the speech is for a listener [38]. One way to measure these aspects is by performing subjective tests based on a panel of listeners, such as the Mean Opinion Score (MOS) [29]. In this method the listeners have to rate the quality of the speech using a five-point scale: excellent (5), good (4), fair (3), poor (2), and bad (1). The average value of all the scores given by the listeners for each speech is the MOS value [38]. Unfortunately, this methods are costly, and since the quality rating is highly subjective, a good estimate of the quality requires many listeners. As a consequence, objective measures have been proposed, and among them perceptual evaluation of speech quality (PESQ) [70] (which also has a wide-band extension [30]) and short-time objective intelligibility (STOI) [80] are the most used estimators.

PESQ is a measure that takes into account the human auditory perception to evaluate the quality of speech. This is done by using a perceptual model to convert the clean and the enhanced (after time alignment) speech signals. Then, the difference between the representations of the signals is used by a cognitive model [38] to produce a score between -0.5 and 4.5. For the wide-band extension, the following output mapping function is applied [30]:

$$y = 0.999 + \frac{4.999 - 0.999}{1 + e^{-1.3669\,x + 3.8224}} \tag{4.5}$$

On the other hand, STOI is a measure highly correlated with speech intelligibility [80] and can assume values in the range $[-1, 1]$, where -1 is associated with the lowest intelligibility. To obtain this score the clean and the enhanced signals are required, and the process consists of the following steps. First, a T-F representation of each signal is computed, using 15 one-third octave bands. Then, the temporal envelops of short-time segments (384 ms) are normalised and clipped. After that, a linear correlation coefficient between clean/noisy pair of segments is calculated, and the final score is obtained by averaging over all frames and bands.

We use the implementation from [46] for PESQ and the one from [80] for STOI.

We also evaluate the performance in terms of equal error rate (EER) of an ASV system. In particular, we use the Gaussian mixture model - universal background model (GMM-UBM) framework, which consists of three main stages. First, a general model, UBM, is built by training a GMM with a large set of speech data using the expectation-maximisation algorithm. The mixture density is [67]:

$$p(\bar{x}|\bar{\theta}) = \sum_{i=1}^{M} w_i p_i(\bar{x}), \tag{4.6}$$

where $\bar{x}$ is a D-dimensional feature vector (generally MFCCs), $p_i(\bar{x})$ is the i-th Gaussian component, w_i is the i-th mixture weight that should satisfy the condition $\sum_{i=1}^{M} w_i = 1$, and $\bar{\theta}$ is the set of parameters of the model (the mean vectors, the covariance matrices, and the mixture weights for each distribution). In our case $D = 57$ and $M = 512$.

After the computation of the UBM, we need to train a model for each speaker and for each passphrase. Generally, we do not have access to a large set of data for a specific speaker, so we cannot use the same approach as the one used to build the UBM. Therefore, we derive the speaker model from the UBM using a maximum a posteriori (MAP) adaptation, which can be seen as a regularised form of the ML estimation (Section 3.4.1). In MAP we also include prior knowledge, $p(\bar{\theta})$, obtaining the following formulation:

$$\bar{\theta}^* = \arg\max_{\bar{\theta}} p(\bar{\theta}|\bar{x}^{(1)}, \ldots, \bar{x}^{(M)}) = \arg\max_{\bar{\theta}} \frac{p(\bar{x}^{(1)}, \ldots, \bar{x}^{(M)}|\bar{\theta})\, p(\bar{\theta})}{p(\bar{x}^{(1)}, \ldots, \bar{x}^{(M)})}. \tag{4.7}$$

Since $p(\bar{x}^{(1)}, \ldots, \bar{x}^{(M)})$ is independent of $\bar{\theta}$, we have:

$$\bar{\theta}^* = \arg\max_{\bar{\theta}} p(\bar{x}^{(1)}, \ldots, \bar{x}^{(M)}|\bar{\theta})\, p(\bar{\theta}). \tag{4.8}$$

Finally, the log-likelihood ratio for a test utterance $\bar{x}^*$ is computed as [67]:

$$\Lambda(\bar{x}^*) = \log p(\bar{x}^*|\bar{\theta}_H) - \log p(\bar{x}^*|\bar{\theta}_{UBM}), \tag{4.9}$$

where $\bar{\theta}_H$ and $\bar{\theta}_{UBM}$ are the parameters of the hypothesised speaker model and the UBM respectively.

4.3.2 Baseline Methods

In our experiments, we use two approaches as baselines: Short-Time Spectral Amplitude Minimum Mean Square Error (STSA-MMSE), and an ideal ratio mask (IRM) based deep neural network speech enhancement (DNN-SE) algorithm.

STSA-MMSE is a classical speech enhancement method that estimates the STSA in a MMSE sense. The MMSE estimator is a Bayesian estimator that can be seen as the conditional expectation of the parameters $\bar{\theta}$ given the observations $\bar{z}$:

$$\bar{\theta}^* = \mathrm{E}[\bar{\theta}|\bar{z}] = \int \bar{\theta}\, p(\bar{\theta}|\bar{z}) d\theta. \tag{4.10}$$

In our context, this leads to the application of a gain function, F, to $Y(n,\omega)$ [10]:

$$\hat{X}(n,\omega) = F(\,\xi(n,\omega), \gamma(n,\omega)\,)\; Y(n,\omega) \tag{4.11}$$

where [19]

$$\xi(n,\omega) = \frac{P_{xx}(n,\omega)}{P_{nn}(n,\omega)} \tag{4.12}$$

is the a priori signal to noise ratio (SNR) with P_{xx} and P_{nn} being the power spectrum of the clean and noise signals respectively, and [19]

$$\gamma(n,\omega) = \frac{|Y(n,\omega)|^2}{P_{nn}(n,\omega)} \tag{4.13}$$

is the a posteriori SNR. Assuming that the spectral components are modelled with a Gaussian distribution [10], the gain function is [10]:

$$\begin{aligned} F(\,\xi(n,\omega), \gamma(n,\omega)\,) = & \frac{\sqrt{\pi\, v(n,\omega)}}{2\,\gamma(n,\omega)} \exp\left(-\frac{v(n,\omega)}{2}\right) \\ & \left[(\,1 + v(n,\omega)\,) I_0\left(\frac{v(n,\omega)}{2}\right) + v(n,\omega) I_1\left(\frac{v(n,\omega)}{2}\right)\right], \end{aligned} \tag{4.14}$$

where I_0 and I_1 are the modified Bessel functions of zero and first order and

$$v(n,\omega) = \frac{\xi(n,\omega)}{1+\xi(n,\omega)}\,\gamma(n,\omega). \tag{4.15}$$

In order to estimate the a priori SNR, we use the decision directed approach [10]:

$$\hat{\xi}(n,\omega) = \alpha \frac{|\hat{X}(n-1,\omega)|^2}{P_{nn}(n-1,\omega)} + (1-\alpha)\,\mathrm{P}[\gamma(n,\omega) - 1], \tag{4.16}$$

where α is a smoothing factor (in our case $\alpha = 0.98$) and

$$\mathrm{P}[x] = \begin{cases} x & \text{if } x \geq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{4.17}$$

The approach described above assumes that the distributions of the DFT coefficients of both speech and noise signals are Gaussians, but in our experiments we assume that the magnitude of the clean speech DFT has a generalised-Gamma distribution [21] as in [11], with parameters $\gamma = 2$ and $\nu = 0.15$. The noise power spectral density (PSD) has been estimated using the work of Hendriks et al. [20], where a noise PSD estimate based on the first 1000 samples of each utterance, assumed as a speech-free region, is used for the noise tracker initialisation.

The IRM-based DNN-SE (from now on simply DNN-SE) we use is based on [36]. The IRM estimation is performed with a DNN with three 1024-unit hidden layers and a 64-unit output. The input vector is a combination of 31 MFCCs, 15 amplitude modulation spectrogram, 13 relative spectral transform - perceptual linear prediction

and 64 gammatone filter bank energies, with their delta and delta-delta for a context of 2 past and 2 future frames. In total, we have a $3 \cdot (31 + 15 + 13 + 64) \cdot 5 = 1845$-dimensional feature vector. The target IRM is computed as in [33]

$$\mathrm{IRM}(n, \omega) = \left(\frac{|X(n, \omega)|^2}{|X(n, \omega)|^2 + |D(n, \omega)|^2} \right)^{0.5} \tag{4.18}$$

where, in this case, the T-F representation of the signals is based on a gammatone filter bank with 64 filters linearly spaced on a Mel frequency scale. The DNN is trained for 30 epochs with SDG, using the mean square error as loss function and a batch size of 1024. We can enhance a degraded signal with the DNN-SE by applying the estimated IRM to the T-F representation of the noisy signal and then reconstructing the time domain signal.

4.3.3 Datasets

The two speech corpora we use are TIMIT [14] and RSR2015 [40]. Specifically, we take 4380 utterances of male speakers from TIMIT to train the UBM. Regarding RSR2015, we consider three sets. The first one, consisting of Text ID from 2 to 30 of sessions 1, 4, and 7 for the male speakers from m051 to m100, is used to train our system and DNN-SE. For the second one, we select Text ID 1 of sessions 1, 4, and 7 for the male speakers from m002 to m050 to train the speaker models. For the last one, with the purpose of evaluating the systems, we choose the same text ID and speakers used for training the models, but sessions 2, 3, 5, 6, 8, and 9. Table 4.2 summarises the allocation of RSR2015 data.

Set	Purpose	Text ID	Session ID	Speaker ID
1	Pix2Pix / DNN-SE training	2-30	1, 4, 7	m051-m100
2	Speaker models training	1	1, 4, 7	m002-m050
3	Evaluation	1	2, 3, 5, 6, 8, 9	m002-m050

Table 4.2. Allocation of RSR2015 speech data used for our experiments.

RSR2015 represents a compromise. It has been chosen because we are interested in evaluating the performance of an ASV system, and it is widely used for this purpose.

In order to simulate real-life noisy conditions, five noise types have been added to the clean signals at different signal to noise ratio (SNR):

- Airplane, collected by Fondazione Ugo Bordoni (FUB) and available on request from the OCTAVE project [12].
- Babble, obtained by adding 6 random speech samples from the Librispeech corpus [60].
- Cantine, recorded by Thomsen et al. [82].
- Market, collected by FUB as for the airplane noise.
- White Gaussian noise, generated in MATLAB.

4.3.4 Setup

Six Pix2Pix-based speech enhancement front-ends have been investigated: five noise specific front-ends (NS-Pix2Pix), each trained on one kind of noise, and one noise general front-end (NG-Pix2Pix), trained on all the five types of noise. Also for the DNN-SE front-ends, we train five noise-specific front-ends (NS-DNN) and one noise general front-end (NG-DNN). The degraded speech used for training has been generated by adding noise to clean speech at 10 and 20 dB SNR. Using a high SNR is motivated by the fact that G can capture the structure of the noisy input easier and generate a clean spectrogram. However, exploring the possible improvements that a use of lower SNR could allow to achieve can be done as a future work. To test the systems, the noise has been added at five SNR: 0, 5, 10, 15, and 20 dB. In the future, a test with lower SNR is useful especially for intelligibility evaluation. The performance of the ASV system on enhanced clean data is reported to show the front-ends' behaviour on noise-free conditions.

Three tests have been conducted using the following front-ends: no enhancement, STSA-MMSE, NS-DNN, NS-Pix2Pix, NG-DNN, and NG-Pix2Pix.

- Test 1. PESQ and STOI are computed to evaluate the speech quality and the intelligibility.
- Test 2. The EER of the ASV system when enhanced clean speech is used for training is computed.
- Test 3. Multi-condition training is performed and the EER scores calculated. For no enhancement, STSA-MMSE, NS-DNN, and NS-Pix2Pix, enhanced clean speech and one kind of enhanced noisy speech are used to build the speaker models, whereas all noise types are used to train the ASV system when we evaluate NG-DNN and NG-Pix2Pix.

4.4 Results and Discussion

Table 4.3 shows the results of the first test. We can see that the average PESQ scores of NS-Pix2Pix and NG-Pix2Pix are better than the other front-ends. Especially between 5 and 15 dB SNR, our system produces the best outcome, regardless of the noise type. NG-Pix2Pix outperforms NS-Pix2Pix at 0 dB with one exception (market noise), and its performance is close to the ones of DNN-SE. At 20 dB, the results for babble and cantine noises indicate that all the speech enhancement methods introduce some distortion, since the PESQ score is higher when no enhancement is performed. For airplane noise, the best technique is STSA-MMSE, but for market and white noises our front-ends obtain again better results.

Regarding STOI, our system results are closer to STSA-MMSE than to DNN-SE front-ends, which are superior overall. However, in some cases (market and cantine noises at low SNR) STOI scores obtained with Pix2Pix are the same or really close to the DNN-SE ones. At 20 dB, we have a behaviour similar to the one observed with PESQ, where the results are better when no enhancement is adopted.

		PESQ						STOI					
	SNR	0	5	10	15	20	mean	0	5	10	15	20	mean
Airplane	(a)	1.34	1.63	2.02	2.47	3.00	2.09	0.64	0.74	0.82	**0.88**	**0.93**	0.80
	(b)	1.54	1.79	2.17	2.72	**3.26**	2.30	0.66	0.74	0.81	0.87	0.91	0.80
	(c)	1.65	1.94	2.30	2.73	3.16	2.36	**0.69**	**0.76**	**0.83**	**0.88**	0.92	**0.82**
	(d)	1.57	2.02	**2.51**	**2.91**	3.18	2.44	0.66	0.75	0.81	0.85	0.89	0.79
	(e)	1.65	1.94	2.29	2.70	3.14	2.35	**0.69**	**0.76**	0.82	0.87	0.91	0.81
	(f)	**1.67**	**2.07**	**2.51**	2.88	3.13	**2.45**	0.67	0.74	0.79	0.83	0.86	0.78
Babble	(a)	1.20	1.42	1.79	2.40	**3.13**	1.99	0.44	0.56	0.67	0.77	0.85	0.66
	(b)	1.14	1.31	1.61	2.07	2.65	1.76	0.43	0.56	0.66	0.74	0.81	0.64
	(c)	**1.25**	1.51	1.87	2.31	2.78	1.95	**0.50**	**0.63**	**0.72**	**0.79**	**0.86**	**0.70**
	(d)	1.20	1.48	1.98	2.52	2.93	2.02	0.46	0.59	0.71	0.78	0.83	0.67
	(e)	1.24	**1.52**	1.88	2.31	2.78	1.95	0.49	0.62	**0.72**	**0.79**	0.85	**0.70**
	(f)	1.20	1.49	**2.00**	**2.53**	2.93	**2.03**	0.46	0.60	0.71	0.77	0.82	0.67
Cantine	(a)	1.35	1.65	2.07	2.57	**3.30**	2.19	0.54	0.66	0.75	**0.83**	**0.90**	0.74
	(b)	1.38	1.68	2.12	2.67	3.23	2.22	0.55	0.66	0.74	0.82	0.87	0.73
	(c)	1.46	1.75	2.15	2.63	3.12	2.22	0.59	**0.69**	0.76	**0.83**	0.89	0.75
	(d)	1.45	1.84	2.38	**2.82**	3.13	2.32	0.58	0.68	0.75	0.80	0.85	0.73
	(e)	1.47	1.77	2.18	2.64	3.11	2.24	**0.60**	**0.69**	**0.77**	**0.83**	0.89	**0.76**
	(f)	**1.49**	**1.91**	**2.43**	2.81	3.08	**2.34**	0.59	**0.69**	0.75	0.80	0.84	0.73
Market	(a)	1.26	1.51	1.89	2.38	3.04	2.02	0.51	0.62	0.73	0.81	**0.88**	0.71
	(b)	1.24	1.45	1.76	2.22	2.79	1.89	0.51	0.62	0.71	0.79	0.85	0.70
	(c)	1.35	1.63	2.00	2.46	2.94	2.08	**0.56**	**0.67**	**0.75**	**0.82**	**0.88**	**0.73**
	(d)	**1.36**	1.71	2.21	**2.72**	**3.09**	**2.22**	0.55	0.66	0.74	0.80	0.85	0.72
	(e)	**1.36**	1.63	2.00	2.45	2.93	2.07	**0.56**	**0.67**	**0.75**	**0.82**	**0.88**	**0.73**
	(f)	1.35	**1.72**	**2.24**	2.68	3.02	2.20	**0.56**	**0.67**	0.74	0.79	0.83	0.72
White	(a)	1.15	1.31	1.60	2.01	2.57	1.73	0.50	0.61	0.72	0.81	**0.89**	0.71
	(b)	1.35	1.58	1.88	2.25	2.71	1.95	0.53	0.63	0.73	0.81	0.87	0.72
	(c)	**1.38**	1.66	2.00	2.39	2.88	2.06	**0.58**	**0.67**	**0.75**	**0.82**	0.88	**0.74**
	(d)	1.23	1.54	2.11	**2.74**	**3.14**	2.15	0.53	0.64	0.73	0.80	0.86	0.71
	(e)	1.35	1.63	1.96	2.29	2.65	1.98	0.57	0.66	0.74	0.81	0.88	0.73
	(f)	1.32	**1.69**	**2.22**	2.68	3.01	**2.19**	0.55	0.65	0.73	0.78	0.83	0.71

Table 4.3. Performance in terms of PESQ and STOI. The 5 front-ends used are: No enhancement (a), STSA-MMSE (b), NS-DNN (c), NS-Pix2Pix (d), NG-DNN (e), NG-Pix2Pix (f).

Table 4.4 reports the results of the second and third tests. For the clean speaker models, we can see that sometimes the DNN-SE front-ends show better results than the other techniques (e.g. at low SNR for babble noise). However, on average the Pix2Pix front-ends outperform the baseline methods, with two exceptions: in presence of babble noise the EER for NG-DNN is 8.73%, slightly lower than NS-Pix2Pix; when signals are degraded with white noise, DNN-SE front-ends outperform the noise-specific Pix2Pix-based enhancement method.

If we focus on the performance on multi-condition training, we see a general improvement of the deep learning-based front ends in comparison with the clean speaker model case. The DNN-SE front-ends are overall better than the other approaches. Our front-ends show better results than STSA-MMSE, even though a higher EER is reported when NS-Pix2Pix is used to reduce white noise. However, especially at high SNR (15 and 20 dB), NS-Pix2Pix is the best approach in all noise conditions with the exception of white noise.

		Test 2							Test 3						
	SNR	0	5	10	15	20	clean	mean	0	5	10	15	20	clean	mean
Airplane	(a)	21.09	15.99	13.61	11.66	9.18	6.99	13.08	32.28	26.87	21.10	16.38	9.86	5.83	18.72
	(b)	17.69	12.58	8.17	6.53	6.27	5.80	9.51	25.51	15.48	8.16	6.12	5.44	5.44	11.03
	(c)	16.99	10.55	7.48	6.99	6.15	6.12	9.05	14.78	8.26	5.44	5.53	4.76	4.76	7.26
	(d)	17.19	8.84	**5.44**	5.05	**4.63**	**3.74**	7.48	16.67	7.14	5.10	**4.03**	**3.78**	4.42	6.86
	(e)	15.99	8.99	6.12	6.12	5.58	5.67	8.08	**11.38**	**6.12**	**4.78**	4.72	4.23	**4.00**	**5.87**
	(f)	**15.31**	**7.89**	5.58	**4.77**	4.76	5.44	**7.29**	13.27	6.43	5.78	5.44	5.27	4.78	6.83
Babble	(a)	19.05	14.63	11.69	11.04	9.18	6.99	12.10	21.77	15.37	11.93	9.52	8.16	6.12	12.15
	(b)	29.04	20.40	12.59	7.82	6.29	5.80	13.66	33.50	23.13	16.23	12.63	8.84	7.12	16.91
	(c)	17.01	10.54	7.82	6.46	6.12	5.78	8.96	16.26	9.52	6.99	6.08	5.78	5.17	8.30
	(d)	18.83	11.22	7.62	**5.70**	5.10	**4.08**	8.76	20.75	10.88	6.12	**4.76**	**4.08**	4.36	8.49
	(e)	**16.67**	**10.39**	**7.50**	6.34	5.78	5.67	**8.73**	**16.00**	**9.18**	**5.44**	**4.76**	**4.08**	**4.00**	**7.19**
	(f)	21.05	13.64	8.50	5.97	**4.76**	5.44	9.90	21.72	12.44	6.46	5.34	5.22	4.78	9.33
Cantine	(a)	20.72	19.20	14.74	11.81	8.50	6.99	13.66	24.11	17.22	12.93	10.88	9.18	7.48	13.63
	(b)	19.09	12.37	8.16	6.80	6.12	5.80	9.72	19.05	12.59	8.21	6.91	6.12	6.32	9.87
	(c)	18.71	**8.58**	6.12	5.49	5.31	5.10	8.22	12.93	5.91	**4.42**	4.25	4.27	**3.78**	5.93
	(d)	**17.33**	9.18	**5.44**	**5.10**	5.10	**4.16**	**7.72**	14.29	6.87	4.76	**4.00**	**4.08**	4.76	6.46
	(e)	19.94	9.18	6.12	5.78	5.44	5.67	8.69	**11.61**	**5.78**	5.10	4.57	**4.08**	4.00	**5.86**
	(f)	17.57	8.84	5.73	5.31	**4.76**	5.44	7.94	14.10	7.48	5.44	5.44	5.27	4.78	7.08
Market	(a)	29.40	20.07	15.00	11.96	8.93	6.99	15.39	36.05	26.06	18.37	13.32	9.18	5.44	18.07
	(b)	25.51	17.35	11.90	8.28	7.35	5.80	12.70	29.25	21.07	13.95	10.98	7.82	6.67	14.97
	(c)	21.43	**9.86**	**6.88**	6.46	5.78	5.92	9.39	19.33	**8.16**	6.24	5.41	4.53	4.29	7.99
	(d)	**17.91**	10.33	7.14	**5.92**	5.17	**3.61**	**8.35**	18.49	9.18	5.82	**4.42**	**3.74**	4.76	7.74
	(e)	21.77	10.59	7.48	6.22	5.76	5.67	9.58	**18.37**	**8.16**	**5.78**	4.44	4.42	**4.00**	**7.53**
	(f)	19.58	11.22	7.48	6.12	**5.07**	5.44	9.15	19.30	9.37	6.37	5.44	5.10	4.78	8.39
White	(a)	45.90	43.20	34.61	26.28	16.91	6.99	28.98	35.88	24.40	18.37	15.81	14.97	5.85	19.21
	(b)	30.95	21.17	13.95	10.20	8.50	5.80	15.10	30.95	20.07	7.48	6.46	6.46	4.76	12.70
	(c)	39.46	20.75	9.86	7.82	6.12	6.02	15.01	27.21	**9.52**	**6.12**	**5.02**	4.65	5.78	9.72
	(d)	40.48	28.23	12.45	7.86	6.46	6.46	16.99	39.37	23.81	10.20	6.46	5.95	6.44	15.37
	(e)	40.14	21.77	10.88	8.16	6.80	5.67	15.57	**26.19**	11.22	7.14	5.10	**4.08**	**4.00**	**9.62**
	(f)	**30.61**	**17.33**	**9.40**	**7.14**	**5.78**	**5.44**	**12.62**	30.41	14.29	8.84	6.60	5.78	4.78	11.78

Table 4.4. ASV performance in terms of EER on clean speaker model (Test 1) and on multi-condition speaker model (Test 2). The 5 front-ends used are: No enhancement (a), STSA-MMSE (b), NS-DNN (c), NS-Pix2Pix (d), NG-DNN (e), NG-Pix2Pix (f).

In Figure 4.4 we report the spectrogram of a noisy utterance (white noise at 0 dB SNR) and visually compare it with the clean and enhanced versions. We can see that the cGAN approach allows to preserve the structure of the original signal better than NG-DNN and STSA-MMSE, even though the noise is not very well reduced especially at high frequencies. However, the method guarantees better performance than STSA-MMSE, which produces a very snowy spectrogram. More examples are shown in Appendix A.

Overall, we can consider our approach superior to STSA-MMSE, and comparable to DNN-SE. However it has some drawbacks. First of all, it accepts only 256 frames as input and produces 256 frames as output, which makes the technique not really suitable for real-time speech enhancement. Some experiments have been conducted trying to reduce the context, but we experienced a performance drop. This is something that should be investigated in the future. Another disadvantage is the complexity of the method. Our methods and DNN-SE have both a complexity way larger than STSA-MMSE. The total number of learnable parameters of our system are around 90M, against the 4M parameters of DNN-SE. However, only the genera-

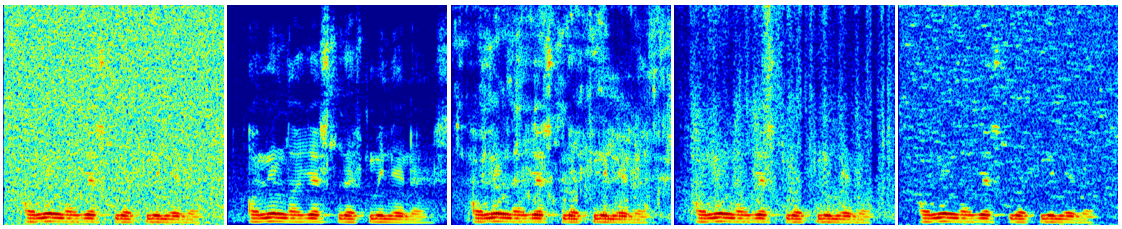


Figure 4.4. Spectrograms of noisy, clean, and enhanced signals. From left to right: noisy spectrogram (White noise at 0 dB SNR); clean spectrogram; NG-Pix2Pix enhanced spectrogram; NG-DNN enhanced spectrogram; STSA-MMSE enhanced spectrogram. The MATLAB implementation used to generate the spectrograms is the one in [86].

tor is involved in the enhancement of a speech signal, so we actually use a network with 85M parameters, which needs to run once every 256 frames, while for DNN-SE the network is required to run for each frame.

CHAPTER 5

GENERATIVE ADVERSARIAL NETWORKS FOR AUTOMATIC SPEECH GENERATION

Automatic speech generation (or speech synthesis) aims at producing signals that sound like human speech. Two main approaches are [84]: non-parametric concatenative speech synthesis [54, 26] and statistical parametric speech synthesis [88, 89, 84]. With the first technique, utterances are produced by concatenating samples of recorded speech stored in a database. On the other hand, the statistical parametric approach makes use of a generative model. Both methods have their strengths and weaknesses: concatenative synthesis generally produces speech that sounds more natural, but in contrast to the parametric approach it does not allow to control the speech characteristics.

In this Chapter we will describe our attempts to generate speech using GANs. Considering that to our knowledge no GAN-based systems have been proposed for speech synthesis yet, this is an open area of research.

5.1 Problem Formulation

The problem of speech synthesis is generally tackled by designing systems that map a text to a speech signal. For this reason this approach is known as text-to-speech (TTS), and its pipeline consists of a text analysis part, that converts a word sequence into a phoneme sequence with its context, and a speech synthesis part, that generates the speech waveform [84].

In our case, we make some simplifications and generate spectrograms of spoken digits using different GAN-based models. From this point of view, our approach is related to image synthesis [17, 63, 5, 51, 58], but it can be extended to a TTS system using a method similar to [65].

5.2 System Overview

As seen for speech enhancement (Section 4.2), in our system we can distinguish between a training and a test phase (Figure 5.1).

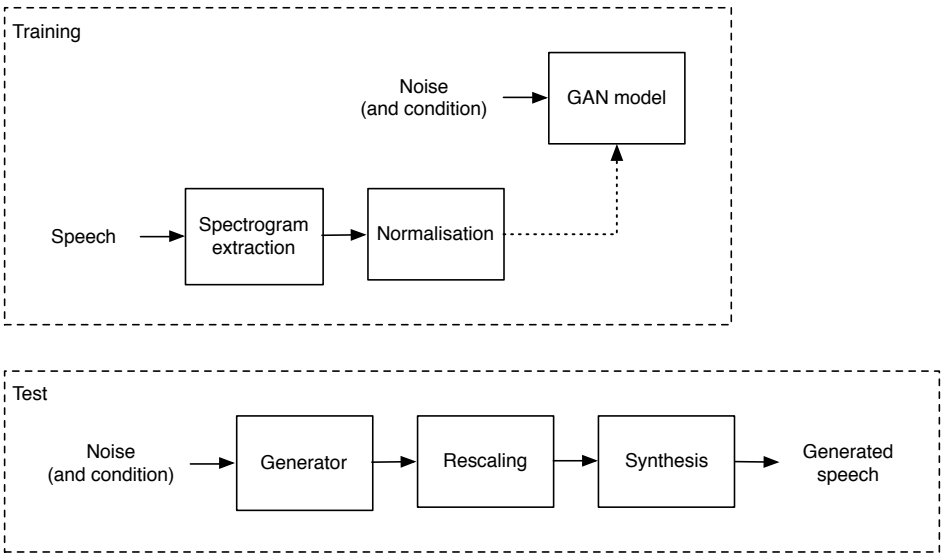


Figure 5.1. Block diagram of our GAN-based speech generation system.

During training, we use a noisy signal and a condition (based on the kind of model adopted) as input of the generator in order to get samples similar to the spectrograms of the signals from the database. We also tried to generate spectrogram-phase pairs, but we did not succeed in making the system converge. The spectrograms of the signals have been computed using a 255-point STFT with a hamming window size of 32 ms and a hop size of 8 ms. Also this time we consider the first 128 elements of each frame of the spectrogram due to symmetry. We zero-pad the spectrograms having fewer that 128 frames to make the architecture of the networks easy to implement. Then, the spectrograms are normalised in order to have values between -1 and 1.

The GAN-based models that we use are three. The first one is a classical GAN, whose G and D architectures are inspired by [58] and shown in Table 5.1. The implementation is based on [31]. The second model is an InfoGAN (Section 3.4.4), using G and D similar to the previous case, with the only difference on the size of the noise vector, which is 76, since we also provide a latent code to the generator. The latent variables can be categorical (one-hot vectors) or continuous (that can assume a value between -1 and 1). The distribution Q is parametrised with a neural network that shares all the convolutional layers with D and adds a fully connected layer on top with as many units as the dimension of the latent code. Our implementation is based on [32]. The third model is a conditional GAN that differs from the one proposed in [51], because we do not use the conditional information as input of D. Suppose that we want to generate one of N possible classes, then D is trained to distinguish between the real and fake samples for each specific class, which means that its output is a $2N$-dimensional vector (Figure 5.2). This approach has been suggested by Odena [57], but to our knowledge no one has applied it yet,

even though it is similar to the AC-GAN model [58]. Again, the architectures of G and D are the ones in Table 5.1, with G that takes a 120-dimensional noise vector, and D that outputs $2N$ units, followed by a softmax layer.

Generator		
	Layer	Dimension
Input	-	(110)
G1	FC/BN/ReLU	(8, 8, 768)
G2	deconv/BN/ReLU	(16, 16, 384)
G3	deconv/BN/ReLU	(32, 32, 256)
G4	deconv/BN/ReLU	(64, 64, 192)
G5	deconv/tanh	(128, 128, 1)

Discriminator		
	Layer	Dimension
Input	-	(128, 128, 1)
D1	conv/l-ReLU/dropout	(64, 64, 16)
D2	conv/BN/l-ReLU/dropout	(64, 64, 32)
D3	conv/BN/l-ReLU/dropout	(32, 32, 64)
D4	conv/BN/l-ReLU/dropout	(32, 32, 128)
D5	conv/BN/l-ReLU/dropout	(16, 16, 256)
D6	conv/BN/l-ReLU/dropout	(16, 16, 512)
D7	FC/sigmoid	(1)

Table 5.1. Network configurations for the generator and the discriminator. BN denotes batch normalisation, and FC a fully connected layer. The kernel size for the generator is 5×5, while for the discriminator is 3×3.

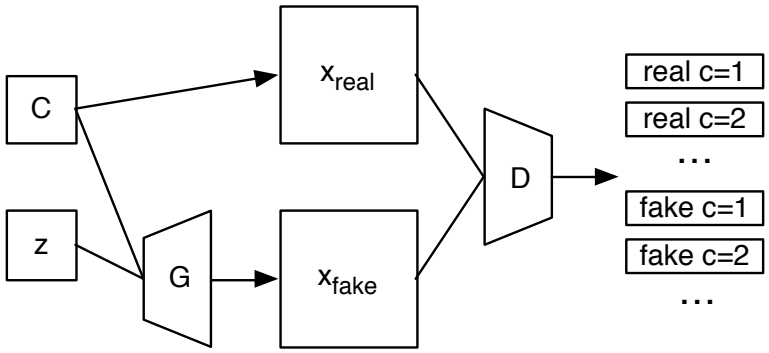


Figure 5.2. Architecture of the proposed GAN. C represents the condition and z the noise.

GANs are trained using SDG with the Adam optimiser and a batch size of 100. Training is stopped based on a visual inspection of the samples generated by G: for the classical GAN and our conditional GAN, we trained the model for 100 epochs, while for InfoGAN we stopped at the 30th epoch. For all the models, we adopted one-sided label smoothing (Section 3.4.4).

At test time, we generate the samples with G and rescale the spectrograms. Then we reconstruct the speech waveform using the work by Slaney [76] based on [18] (Section 2.3).

5.3 Experiments

In this section we will present how we perform the experiments.

5.3.1 Dataset

For our experiments we use the Aurora-2 dataset [23], based on the TIDigits database [43], which contains speech segments of a variety of speakers (males and females) pronouncing digit sequences. The signals are downsampled at 8 kHz and added to some noise signals. We only use 8440 clean speech segments from the training set. From them, we extract the spoken digits and get 25277 samples to train our system in the way we explained in Section 5.2. Figure 5.3 shows 100 spectrograms of spoken digits pronounced by a woman.

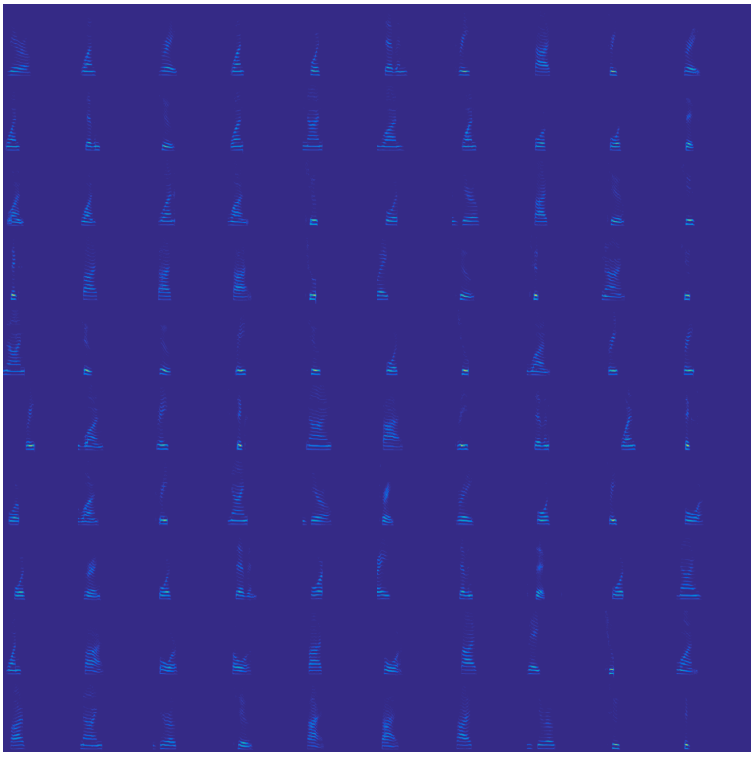


Figure 5.3. Spectrograms of spoken digits pronounced by a woman from the Aurora-2 dataset.

5.3.2 Setup

We conduct three different tests, one for each GAN model adopted. In the first one, we simply generate spectrograms with a classical GAN approach, in order to see how good the framework performs. As we said in Section 3.4.4, this technique does not provide a disentangled representation of the data, so it is not really useful in our case because we can generate data, but we have no control on them. Then, we try to use InfoGAN with one categorical latent code to capture the different spoken digits and two continuos codes for other changes, like the pitch and the timbre of the voice. Finally, a test using a conditional GAN is performed to see if we can have some improvements in the quality by constraining the sample generation with class labels.

As reported in [81], the evaluation of a generative model is still an open problem: the assessment of the performance depends on the specific application. Odena et al. [58] proposed metrics for the evaluation of image synthesis models, but for speech synthesis subjective tests are preferred, like in [84] where paired comparison tests and MOS tests have been conducted. Mainly due to time constraints we could not perform a listening study with human listeners, but this is interesting as a future work. However some audio samples of the generated speech are provided, in order to give to the reader the possibility to judge the results.

5.4 Results and Discussion

Figure 5.4 shows the spectrograms drawn from the generator of our GAN. If we compare them with the spectrograms of the dataset (Figure 5.3), we can see that the background is not really clean. This leads to some sort of noise that can be listened after the signal reconstruction. However, most of the times it is still possible to distinguish the digit pronounced and the speaker voice, indicating that G captures the important information of the data.

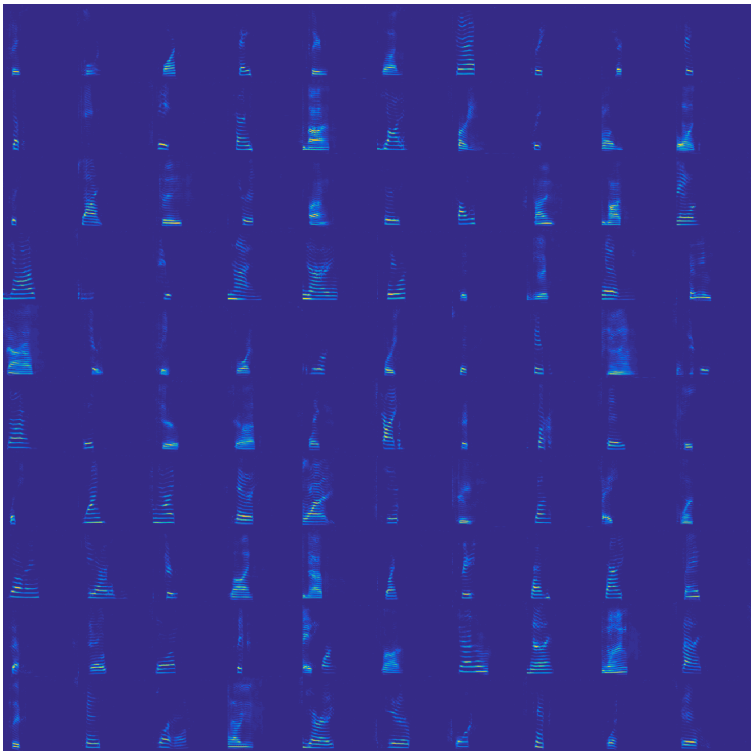


Figure 5.4. Spectrograms of spoken digits generated by our GAN.

In the second test we make use of InfoGAN. As reported before (Section 5.3.2), our idea was to capture the digit variations with a categorical latent code, and the voice variations with two continuos latent variables. After having listened to the generated samples, we did not find this correspondence. The categorical code is responsible to voice variations, while the role of the two continuos variable (c_0 and c_1) is not very clear: their variation leads to the generation of a different digit or to

the way that the digit is pronounced. Looking at the spectrograms, we can notice that increasing c_0 (Figure 5.5) adds more high frequency spectral components, while c_1 (Figure 5.6) seems to influence especially the duration of the signal. Regarding the quality of the generated speech, we experience lower performance, which can be related to the smaller number of training iterations used. When we tried to increase it, we saw that the generator was not producing plausible spectrograms anymore. The reason for this issue is not clear, and it is something we can think to investigate in the future.

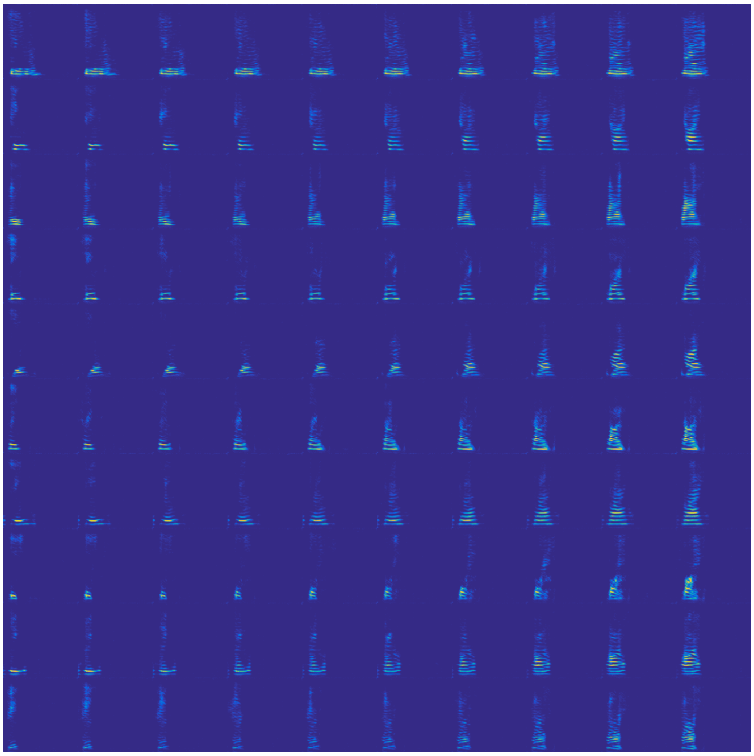


Figure 5.5. Effect of the continuos variable c_0 variations on the generation of spoken digits with InfoGAN. Each row has the categorical code fixed, while c_0 varies between -1 and 1.

The last test is conducted using our conditional GAN. We can see in Figure 5.7 that this method is effective, and we can actually generate the different digits. However, we do not experience a great improvement in the quality of the speech if compared to the classical GAN: we can still hear some background noise and sometimes it is hard to distinguish the kind of digit generated due to the presence of artefacts.

Overall the speech samples reconstructed from the spectrograms generated with our models tend to be noisy, and although it is possible in general to distinguish voice and digit type, sometimes artefacts produce an unnatural sound that makes it difficult to discriminate the digits.

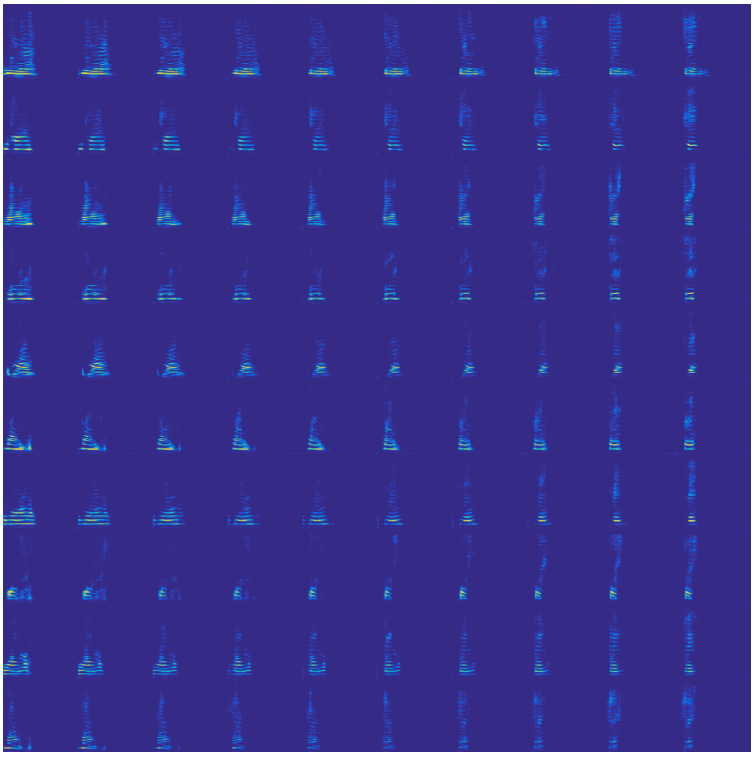


Figure 5.6. Effect of the continuos variable c_1 variations on the generation of spoken digits with InfoGAN. Each row has the categorical code fixed, while c_1 varies between -1 and 1.

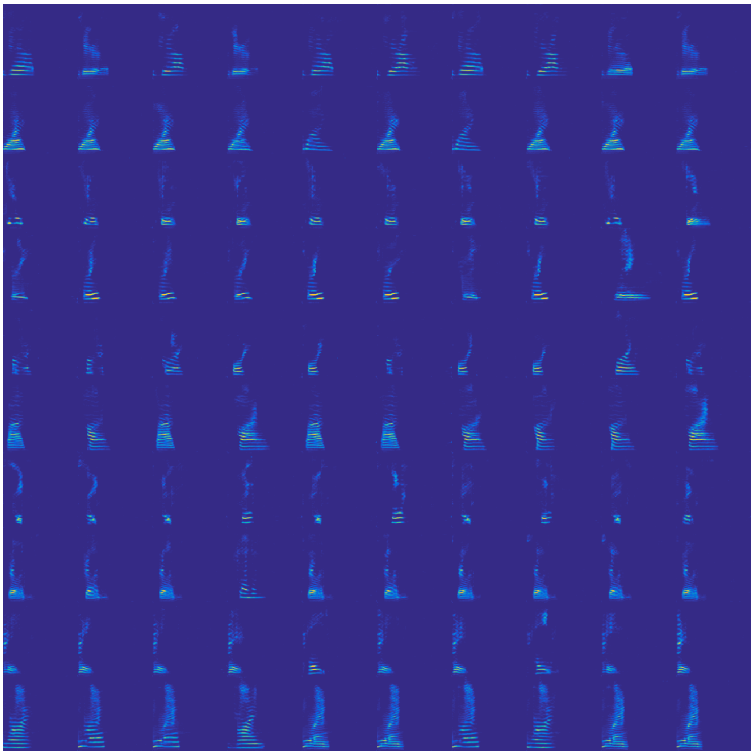


Figure 5.7. Spectrograms of spoken digits generated by our conditional GAN. Each row has the conditional information fixed.

CHAPTER 6

CONCLUSION

In this project we investigated the use of generative adversarial networks for speech processing. Specifically, we addressed two research questions:

1. How can we apply GANs to spectral speech enhancement and what performance can be achieved compared to other methods?

2. How can we generate speech signals with a GAN-based approach and how good are they?

In the first case, we showed that a possible way to enhance speech with GANs is by adopting a framework originally designed for image-to-image translation tasks. This allows to perform a mapping from noisy spectrograms to clean ones. The results indicate that this approach is superior to the classical STSA-MMSE SE technique and comparable to a DNN-based SE method. However, the need of a large context to enhance a speech signal makes it not suitable for real-time applications. A direction to work in the future can be the reduction of the required context, also to have a more compact network to use.

Regarding the second question, we tried to generate speech by using three different GAN-based models trained to have a generator that output spectrograms of spoken digits. We see that our models can capture some important information of the data (voice characteristics and kind of digits), but strong artefacts in the reconstructed signals can be heard, and they make it hard sometimes to understand the content of the speech. Even though we decided to generate speech spectrograms, we could also think to directly generate the waveform with a GAN-based model, as done for example by Mogren [53] with the purpose of generating classical music, and this can be a possible approach to explore in the future.

In conclusion, the focus of this thesis was on designing GAN-based systems for speech enhancement and automatic speech generation. Since GANs have been proposed only recently, they are still an open field of research. Although they have been mainly used in computer vision, the results that we obtained make us confident about their larger adoption to other fields, like speech processing, in the future.

BIBLIOGRAPHY

[1] Notes from the stanford CS class CS231n: Convolutional neural networks for visual recognition. `http://cs231n.github.io`, Accessed: 01-04-2017.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387310738.

[3] Bruce P Bogert, Michael JR Healy, and John W Tukey. The quefrency alanysis of time series for echoes: Cepstrum, pseudo-autocovariance, cross-cepstrum and saphe cracking. In *Proceedings of the symposium on time series analysis*, volume 15, pages 209–243. chapter, 1963.

[4] J. Chen, J. Benesty, J. Huang, and Diethorn E. J. Fundamentals of noise reduction. In *Springer handbook of speech processing*, pages 843–871. Springer Science & Business Media, 2007.

[5] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[6] Soumith Chintala, Emily Denton, Martin Arjovsky, and Michael Mathieu. How to train a gan? tips and tricks to make gans work. `https://www.youtube.com/watch?v=RvgYvHyT15E`, 2016, Accessed: 01-04-2017.

[7] I. Cohen and S. Gannot. Spectral enhancement methods. In *Springer handbook of speech processing*, pages 873–901. Springer Science & Business Media, 2007.

[8] Li Deng and Douglas O'Shaughnessy. *Speech processing: a dynamic and optimization-oriented approach*. CRC Press, 2003.

[9] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[10] Yariv Ephraim and David Malah. Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(6):1109–1121, 1984.

[11] Jan S Erkelens, Richard C Hendriks, Richard Heusdens, and Jesper Jensen. Minimum mean-square error estimation of discrete fourier coefficients with generalized gamma priors. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(6):1741–1752, 2007.

[12] Mauro Falcone, Benoit Fauve, Michele Cornacchia, et al. Corpora collection. *OCTAVE (Objective Control of TAlker VErification), Deliverable 17*, 2016.

[13] Brendan J Frey. *Graphical models for machine learning and digital communication*. MIT press, 1998.

[14] John S Garofolo, Lori F Lamel, William M Fisher, Jonathon G Fiscus, and David S Pallett. Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1. *NASA STI/Recon technical report n*, 93, 1993.

[15] Ian Goodfellow. Introduction to generative adversarial networks. NIPS 2016 Workshop on Adversarial Training, `https://www.youtube.com/watch?v=RvgYvHyT15E`, 2014, Accessed: 01-04-2017.

[16] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

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

[18] Daniel Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32 (2):236–243, 1984.

[19] Richard C Hendriks, Richard Heusdens, and Jesper Jensen. Improved decision directed approach for speech enhancement using an adaptive time segmentation. In *INTERSPEECH*, pages 2101–2104, 2005.

[20] Richard C Hendriks, Richard Heusdens, and Jesper Jensen. Mmse based noise psd tracking with low complexity. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 4266–4269. IEEE, 2010.

[21] Richard C Hendriks, Timo Gerkmann, and Jesper Jensen. Dft-domain based single-microphone noise reduction for speech enhancement: A survey of the state of the art. *Synthesis Lectures on Speech and Audio Processing*, 9(1):1–80, 2013.

[22] Lars Hertel, Erhardt Barth, Thomas Käster, and Thomas Martinetz. Deep convolutional neural networks as generic feature extractors. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–4. IEEE, 2015.

[23] Hans-Günter Hirsch and David Pearce. The aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions. In *ASR2000-Automatic Speech Recognition: Challenges for the new Millenium ISCA Tutorial and Research Workshop (ITRW)*, 2000.

[24] K. Honda. Physiological processes of speech production. In *Springer handbook of speech processing*, pages 7–26. Springer Science & Business Media, 2007.

[25] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[26] Andrew J Hunt and Alan W Black. Unit selection in a concatenative speech synthesis system using a large speech database. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 1, pages 373–376. IEEE, 1996.

[27] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[28] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.

[29] ITU. Methods for subjective determination of transmission quality. Available: https://www.itu.int/rec/T-REC-P.800-199608-I/eno, 1996. Accessed: 01-04-2017.

[30] ITU. Wideband extension to recommendation p.862 for the assessment of wideband telephone networks and speech codecs. Available: https://www.itu.int/rec/T-REC-P.862.2-200511-S/en, 2005. accessed: March 2017.

[31] Arthur Juliani. Deep convolutional generative adversarial network (dcgan) tutorial. Github repository: https://github.com/awjuliani/TF-Tutorials/blob/master/DCGAN.ipynb, 2016. Accessed: 01-04-2017.

[32] Arthur Juliani. Infogan tutorial. Github repository: https://github.com/awjuliani/TF-Tutorials/blob/master/InfoGAN-Tutorial.ipynb, 2016. Accessed: 01-04-2017.

[33] Andrej Karpathy, Pieter Abbeel, Greg Brockman, Peter Chen, Vicki Cheung, Rocky Duan, Ian Goodfellow, Durk Kingma, Jonathan Ho, Rein Houthooft, Tim Salimans, John Schulman, Ilya Sutskever, and Wojciech Zaremba. Generative models. `https://blog.openai.com/generative-models/`, Accessed: 01-04-2017.

[34] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[35] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[36] Morten Kolbæk, Zheng-Hua Tan, and Jesper Jensen. Speech intelligibility potential of general and specialized deep neural network based speech enhancement systems. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(1):153–167, 2017.

[37] Morten Kolbœk, Zheng-Hua Tan, and Jesper Jensen. Speech enhancement using long short-term memory based recurrent neural networks for noise robust speaker verification. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*, pages 305–311. IEEE, 2016.

[38] K. Kondo. *Subjective Quality Measurement of Speech: Its Evaluation, Estimation and Applications*. Signals and Communication Technology. Springer Berlin Heidelberg, 2012.

[39] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[40] Anthony Larcher, Kong Aik Lee, Bin Ma, and Haizhou Li. Text-dependent speaker verification: Classifiers, databases and rsr2015. *Speech Communication*, 60:56–77, 2014.

[41] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[42] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016.

[43] R Leonard. A database for speaker-independent digit recognition. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'84.*, volume 9, pages 328–331. IEEE, 1984.

[44] Yen-Chen Lin. pix2pix-tensorflow. Github repository: https://github.com/yenchenlin/pix2pix-tensorflow, 2016. Accessed: 01-04-2017.

[45] Seppo Linnainmaa. The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors. *Master's Thesis (in Finnish), Univ. Helsinki*, pages 6–7, 1970.

[46] Philipos C Loizou. *Speech enhancement: theory and practice*. CRC press, 2013.

[47] Xugang Lu, Yu Tsao, Shigeki Matsuda, and Chiori Hori. Speech enhancement based on deep denoising autoencoder. In *Interspeech*, pages 436–440, 2013.

[48] Jean-François Mari and René Schott. *Probabilistic and Statistical Methods in Computer Science*. Springer Science & Business Media, 2013.

[49] Daniel Michelsanti and Zheng-Hua Tan. Conditional generative adversarial networks for speech enhancement and noise-robust speaker verification. *Interspeech*, 2017. (In Press).

[50] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA, 1969.

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

[52] Shariq Mobin and Joan Bruna. Voice conversion using convolutional neural networks. *arXiv preprint arXiv:1610.08927*, 2016.

[53] Olof Mogren. C-rnn-gan: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*, 2016.

[54] Eric Moulines and Francis Charpentier. Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech communication*, 9(5-6):453–467, 1990.

[55] Andrew Ng. Notes from the stanford CS class CS229: Machine learning, generative algorithms. `http://cs229.stanford.edu/notes/cs229-notes2.pdf`, Accessed: 01-04-2017.

[56] Andrew Y Ng and Michael I Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems*, 2:841–848, 2002.

[57] Augustus Odena. Semi-supervised learning with generative adversarial networks. *arXiv preprint arXiv:1606.01583*, 2016.

[58] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585*, 2016.

[59] Christopher Olah. Conv nets: A modular perspective. `http://colah.github.io/posts/2014-07-Conv-Nets-Modular`, 2014, Accessed: 01-04-2017.

[60] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5206–5210. IEEE, 2015.

[61] Se Rim Park and Jinwon Lee. A fully convolutional neural network for speech enhancement. *arXiv preprint arXiv:1609.07132*, 2016.

[62] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.

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

[64] Kamisetty Ramam Rao and Patrick C Yip. *The transform and data compression handbook*, volume 1. CRC press, 2000.

[65] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 3, 2016.

[66] Scott E Reed, Yi Zhang, Yuting Zhang, and Honglak Lee. Deep visual analogy-making. In *Advances in Neural Information Processing Systems*, pages 1252–1260, 2015.

[67] D. A. Reynolds and W. M. Campbell. Text-independent speaker recognition. In *Springer handbook of speech processing*, pages 763–781. Springer Science & Business Media, 2007.

[68] D. A. Reynolds, W. M. Campbell, W. Shen, and E. Singer. Automatic language recognition via spectral and token based approaches. In *Springer handbook of speech processing*, pages 811–824. Springer Science & Business Media, 2007.

[69] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and variational inference in deep latent gaussian models. In *International Conference on Machine Learning*, 2014.

[70] Antony W Rix, John G Beerends, Michael P Hollier, and Andries P Hekstra. Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 2, pages 749–752. IEEE, 2001.

[71] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[72] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[73] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[74] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[75] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2226–2234, 2016.

[76] Malcolm Slaney. Spectrogram inversion toolkit for matlab. *IEEE Signal Processing Society SLTC Newsletter*, 2014.

[77] Jost Tobias Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015.

[78] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[79] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[80] Cees H Taal, Richard C Hendriks, Richard Heusdens, and Jesper Jensen. An algorithm for intelligibility prediction of time–frequency weighted noisy speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(7):2125–2136, 2011.

[81] Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

[82] Nicolai Bæk Thomsen, Dennis Alexander Lehmann Thomsen, Zheng-Hua Tan, Børge Lindberg, and Søren Holdt Jensen. Speaker-dependent dictionary-based speech enhancement for text-dependent speaker verification. *Interspeech 2016*, 2016.

[83] Vibha Tiwari. Mfcc and its applications in speaker recognition. *International journal on emerging technologies*, 1(1):19–22, 2010.

[84] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR abs/1609.03499*, 2016.

[85] P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, 1974.

[86] Kamil Wojcicki. Speech spectrogram. `https://it.mathworks.com/matlabcentral/fileexchange/29596-speech-spectrogram`, 2010, Accessed: 01-04-2017.

[87] Yong Xu, Jun Du, Li-Rong Dai, and Chin-Hui Lee. A regression approach to speech enhancement based on deep neural networks. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(1):7–19, 2015.

[88] Takayoshi Yoshimura. *Simultaneous modeling of phonetic and prosodic parameters, and characteristic conversion for HMM-based text-to-speech systems*. PhD thesis, Nagoya Institute of Technology, 2002.

[89] Heiga Zen, Keiichi Tokuda, and Alan W Black. Statistical parametric speech synthesis. *Speech Communication*, 51(11):1039–1064, 2009.

[90] He Zhang, Vishwanath Sindagi, and Vishal M Patel. Image de-raining using a conditional generative adversarial network. *arXiv preprint arXiv:1701.05957*, 2017.

[91] Hristo Zhivomirov. Inverse short-time fourier transformation (istft) with matlab implementation. `https://it.mathworks.com/matlabcentral/fileexchange/45577-inverse-short-time-fourier-transformation--istft--with-matlab-implementation`, 2015, Accessed: 01-04-2017.

[92] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016.

APPENDIX A

SPECTROGRAMS OF NOISY, ENHANCED, AND CLEAN SIGNALS

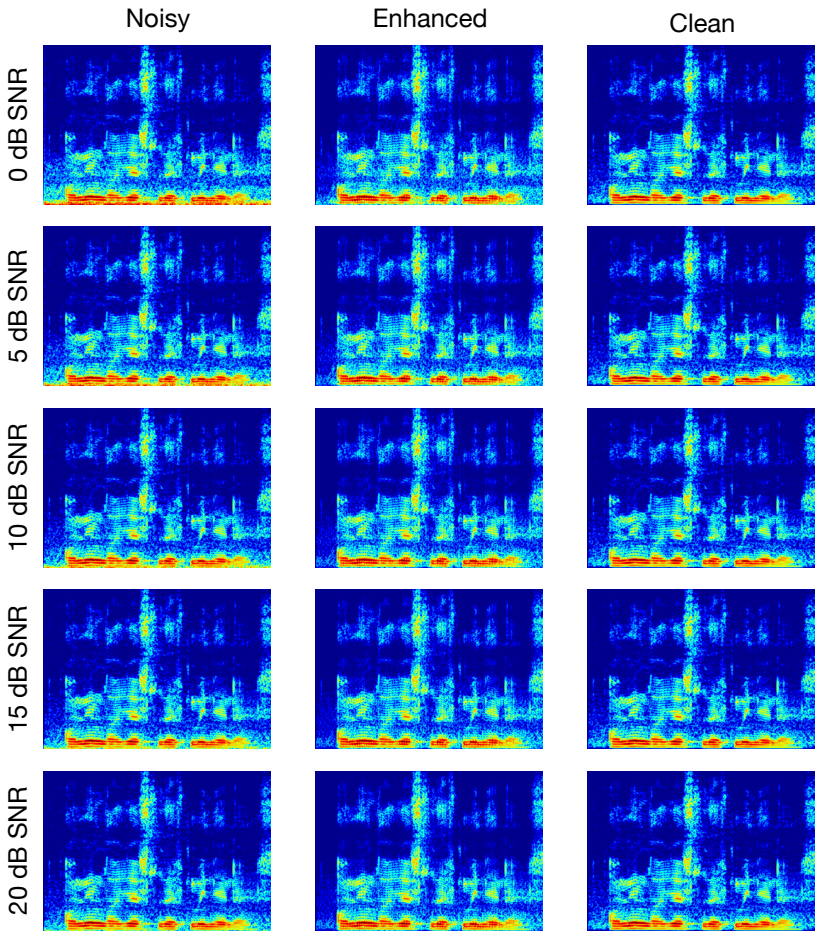


Figure A.1. Comparison between noisy (airplane), enhanced (NG-Pix2Pix), and clean signals in terms of spectrogram.

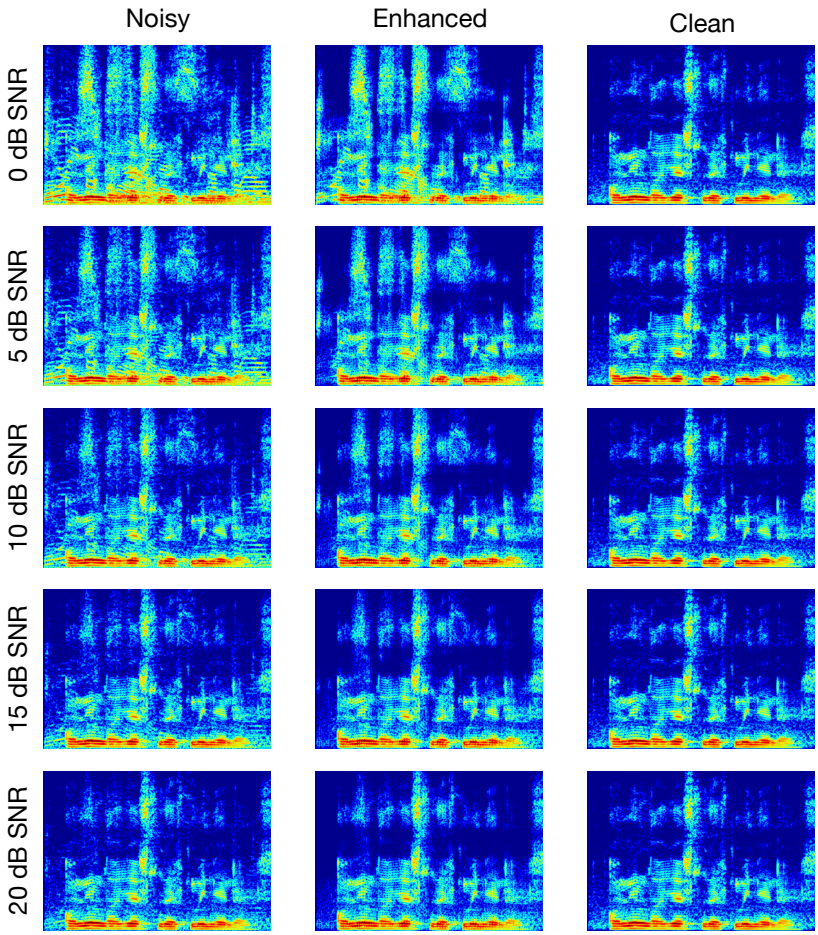


Figure A.2. Comparison between noisy (babble), enhanced (NG-Pix2Pix), and clean signals in terms of spectrogram.

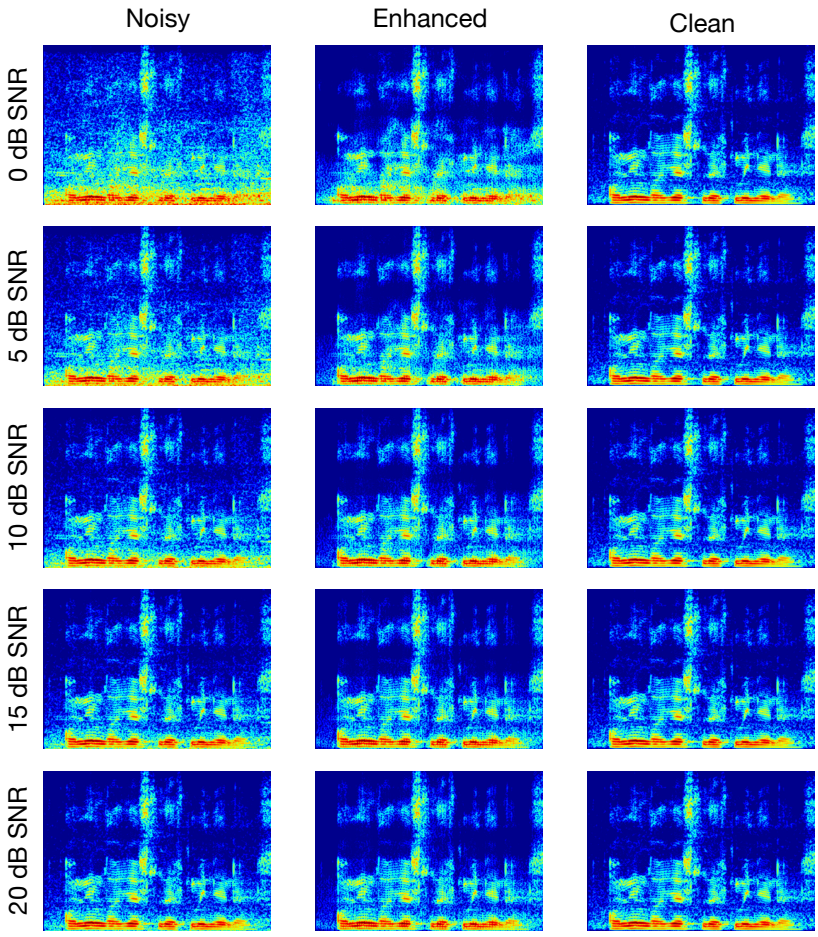


Figure A.3. Comparison between noisy (cantine), enhanced (NG-Pix2Pix), and clean signals in terms of spectrogram.

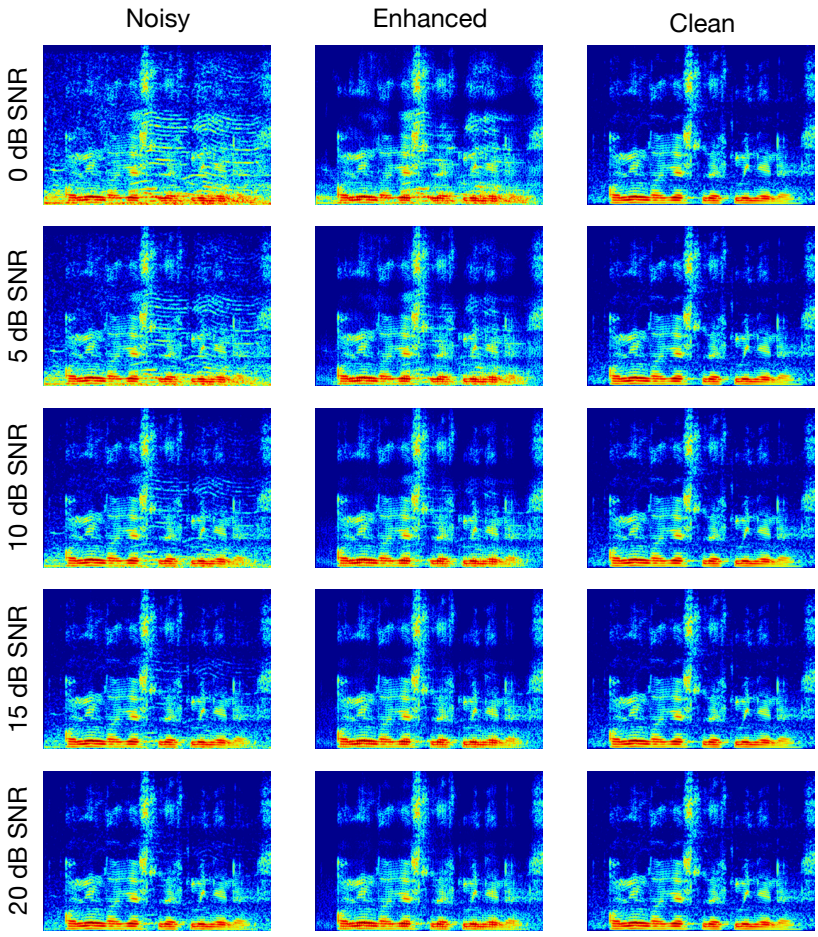


Figure A.4. Comparison between noisy (market), enhanced (NG-Pix2Pix), and clean signals in terms of spectrogram.

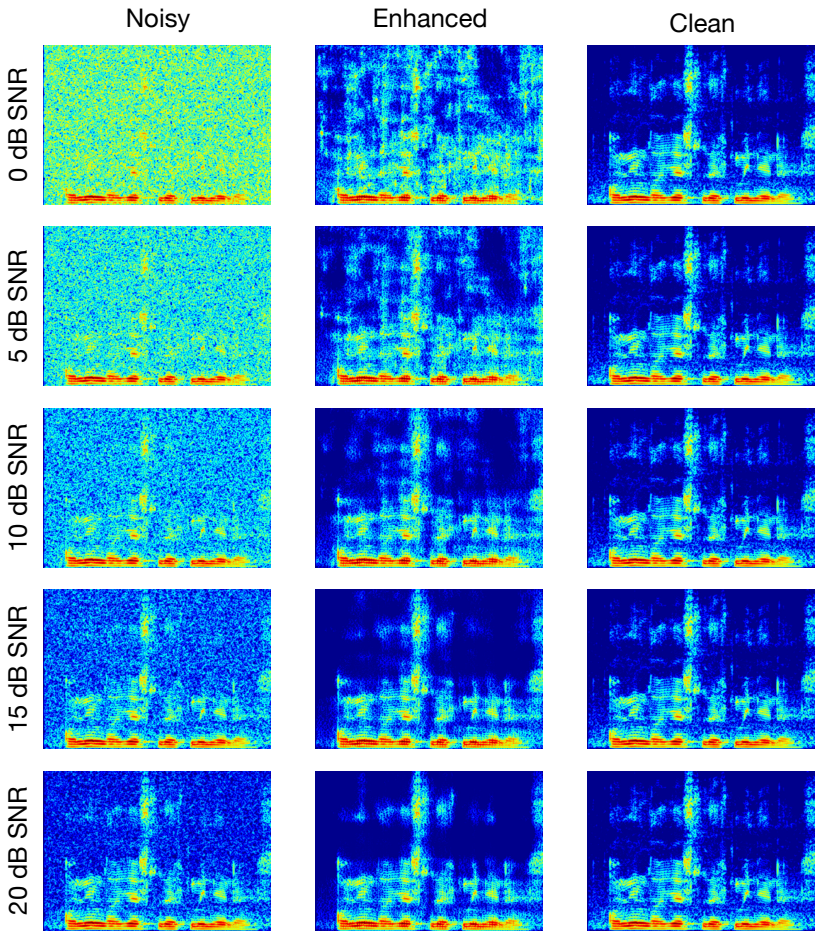


Figure A.5. Comparison between noisy (white), enhanced (NG-Pix2Pix), and clean signals in terms of spectrogram.

APPENDIX B

PAPER

This project has led to a paper that has been accepted to the Interspeech 2017 conference in Stockholm. The reader can find the submitted article in the following pages of this appendix.

Conditional Generative Adversarial Networks for Speech Enhancement and Noise-Robust Speaker Verification

Daniel Michelsanti and Zheng-Hua Tan

Department of Electronic Systems, Aalborg University, Denmark

dmiche15@student.aau.dk, zt@es.aau.dk

Abstract

Improving speech system performance in noisy environments remains a challenging task, and speech enhancement (SE) is one of the effective techniques to solve the problem. Motivated by the promising results of generative adversarial networks (GANs) in a variety of image processing tasks, we explore the potential of conditional GANs (cGANs) for SE, and in particular, we make use of the image processing framework proposed by Isola et al. [1] to learn a mapping from the spectrogram of noisy speech to an enhanced counterpart. The SE cGAN consists of two networks, trained in an adversarial manner: a generator that tries to enhance the input noisy spectrogram, and a discriminator that tries to distinguish between enhanced spectrograms provided by the generator and clean ones from the database using the noisy spectrogram as a condition. We evaluate the performance of the cGAN method in terms of perceptual evaluation of speech quality (PESQ), short-time objective intelligibility (STOI), and equal error rate (EER) of speaker verification (an example application). Experimental results show that the cGAN method overall outperforms the classical short-time spectral amplitude minimum mean square error (STSA-MMSE) SE algorithm, and is comparable to a deep neural network-based SE approach (DNN-SE).

Index Terms: generative adversarial networks, speech enhancement, speaker verification

1. Introduction

Dealing with degraded speech signals is a challenging yet important task in many applications, e.g. automatic speaker verification (ASV) [2], speech recognition [3], mobile communications and hearing assistive devices [4, 5, 6]. When the receiver is a human user, the objective of SE is to improve quality and intelligibility of noisy speech signals. When it is an automatic speech system, the goal is to improve the noise-robustness of the system, e.g. to reduce the EERs of an ASV system under adverse conditions. In the past, this problem has been tackled with statistical methods like Wiener filter and STSA-MMSE [7]. Lately, deep learning methods have been used, such as DNNs [6, 8], deep autoencoders (DAEs) [5], and convolutional neural networks (CNNs) [9]. However, to our knowledge, no one has tried to use GANs for SE yet.

GANs are a framework recently introduced by Goodfellow et al. [10], which consists of a generative model, or generator (G), and a discriminative model, or discriminator (D), that play a min-max game between each other. In particular, G tries to fool D which is trained to distinguish the output of G from the real data. The architectures used in most of the works today [11] are based on deep convolutional GAN (DCGAN) [12] that successfully tackles training instability issues when GANs are applied to high resolution images. Three key ideas are used to accomplish this goal. First, batch normalization [13] is applied to most of the layers. Then, the networks are designed to have no pooling layers as done in [14]. Finally, the training is performed adopting the Adam optimizer [15].

So far GANs have been successfully applied to a variety of computer vision and image processing tasks [1, 12, 16, 17]. However, their adoption for speech-related tasks is rare with one exception in [18], in which the authors of the report applied a deep visual analogy network [19] as a generator of a GAN for voice conversion, and the results are presented as example audio files without speech quality or intelligibility or other measures. In a related field, for music, the GAN concept was applied to train a recurrent neural network for classical music generation [20].

Very recently, a general-purpose cGAN framework called Pix2Pix was proposed for image-to-image translation [1]. Motivated by its successful deployment on several tasks, we adapt the framework in this work, aiming to explore the potential of cGANs for SE, as part of the overall goal of investigating the feasibility and performance of GANs for speech processing. Specifically, we use Pix2Pix to learn a mapping between noisy and clean speech spectrograms as well as to learn a loss function for training the mapping.

2. Pix2Pix framework for speech enhancement

In GANs, G represents a mapping function from a random noise vector $\mathbf{z}$ to an output sample $G(\mathbf{z})$, ideally indistinguishable from the real data $\mathbf{x}$ [10]. In cGANs, both G and D are conditioned on some extra information $\mathbf{y}$ [1], and they are trained following a min-max game with the objective:

$$L(D,G) = \mathbb{E}_{\mathbf{x},\mathbf{y}\sim\, p_{data}(\mathbf{x},\mathbf{y})}[\log(D(\mathbf{x},\mathbf{y}))] + \mathbb{E}_{\mathbf{z}\sim\, p_{\mathbf{z}}(\mathbf{z}),\mathbf{y}\sim\, p_{data}(\mathbf{y})}[\log(1 - D(G(\mathbf{z},\mathbf{y}),\mathbf{y}))]. \quad (1)$$

Pix2Pix differs from other cGAN works, like [21], because it does not use $\mathbf{z}$. Isola et al. [1] report that adding a Gaussian noise as an input to G, as done in [22], was not effective. Hence, they introduce noise in the form of dropout, but this technique failed to produce stochastic output. However, we are more interested in an accurate mapping between a noisy spectrogram and a clean one than a cGAN able to capture the full entropy of the distribution it models, so this represents a minor issue. Figure 1 shows how the data and the condition are used during training in the particular case of this paper.

In addition to the adversarial loss $L(D,G)$ that is learned from the data, Pix2Pix utilizes also L1 distance between the output of G and the ground truth. The choice of combining different losses, like L2 distance [23] or perceptual losses for a specific task [16, 17], has been shown to be beneficial. In Pix2Pix, L1 distance is preferred to L2 because it encourages less blurring [1] and it tends to generalize better if compared to perceptual losses.

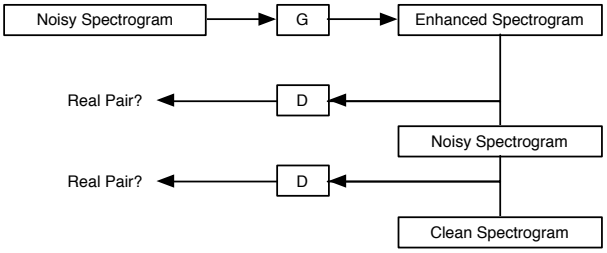


Figure 1: *Generator (G) and discriminator (D) in the Pix2Pix framework for speech enhancement. G generates an enhanced spectrogram from a noisy input by fooling D, which tries to classify a spectrogram as clean or enhanced, conditioned on the respective noisy spectrogram.*

Furthermore, G and D, adapted from [12], are a U-Net [24] and a PatchGAN, respectively. Since in image-to-image translation tasks, the input and the output of G share the same structure, G is an encoder-decoder where each feature map of the decoder layers is concatenated with its mirrored counterpart from the encoder to avoid that the innermost layer represents a bottleneck for the information flow. Besides, D is built to model the high frequencies of the data, as the low frequency structure is captured by the L1 loss. This is achieved by considering local image patches. In particular, D is applied convolutionally across the image to classify each patch as real or fake. Then, the obtained scores are averaged together to get a single output. This architecture has the advantage of being smaller and can be applied on images of different sizes [1]. When the patch size of D has the same size of the input image, D is equivalent to a classical GAN discriminator.

Our Pix2Pix implementation is based on [25], with G that gets a 256×256 1-channel image, while D a 256×256 2-channel image. The main differences with the original framework are the adoption of 5×5 filters in the convolutional layers, and the last layer of D which is flattened and fed into a single sigmoid output as in [12].

2.1. Preprocessing and training

For speech signals with a sample rate of 16 kHz, we computed a time-frequency (T-F) representation using a 512-point short time Fourier transform (STFT) with a hamming window size of 32 ms and a hop size of 16 ms. In this way, the frequency resolution is 16 kHz / 512 = 31.25 Hz per frequency bin. We considered only the 257-point STFT magnitude vectors which cover the positive frequencies due to symmetry. Our generator G accepts $256 \times 256 \times 1$ input, so for training we concatenated all the speech signals and then split the spectrogram every 256 frames, while for testing we zero-padded the spectrogram of each test sample in order to have the number of frames equal to a multiple of 256 and then performed the split accordingly. We also removed the last row of the spectrogram, which is a choice that has a negligible impact since it represents only the highest 31.25 Hz band of the signal, but this allows us to have a power-of-2 input size which makes the design of G and D easier. Before the data are fed to our system, they are also normalized to be in the range $[-1, 1]$.

We trained the GANs using stochastic gradient descent (SGD) and adopting the Adam optimizer, for 10 epochs with a batch size of 1 according to [1], updating G twice per each iteration to avoid a fast convergence of D [25]. The networks' weights have been initialized from a normal distribution with zero mean and a standard deviation of 0.02 [1]. The L1 loss has been added to the GAN loss using a scaling factor of 100 [1].

To enhance a speech signal with Pix2Pix, we first compute the T-F representation of it, and then we forward propagate the spectrogram magnitude through G. Finally, we reconstruct the signal with the inverse STFT using the phase of the noisy input.

3. Experiments

3.1. Evaluation metrics

The performance of our system is evaluated in terms of PESQ [26] (in particular the wide-band extension [27]), STOI [28], and EER of ASV. PESQ and STOI have been chosen as they are the most used estimators of speech quality and speech intelligibility, respectively. The implementations used in this paper are from [7] for PESQ and from [28] for STOI.

Regarding the ASV evaluation, we use the classical Gaussian Mixture Model - Universal Background Model (GMM-UBM) framework [29], which is suitable for short utterances as in this work. We first built a general model, UBM, which is a GMM trained with an expectation-maximization algorithm using a large amount of speech data not belonging to the target speakers. Then, a target speaker model for each specific passphrase and each speaker was derived by maximum a posteriori adaptation of the UBM. The approach of adapting UBM is used in order to have a well-trained model for a speaker even when there is no much data available. At this point, for a test utterance we calculate the log likelihood ratio between the claimant speaker model and the UBM. The features extracted from the speech data are 57-dimensional mel-frequency cepstral coefficients (MFCCs), and the GMM mixture number is 512.

3.2. Baseline methods

We compare the results of our approach with other two methods we consider as baselines: STSA-MMSE and an Ideal Ratio Mask (IRM) based DNN-SE algorithms.

STSA-MMSE is a statistical-based SE technique, where the a priori signal to noise ratio (SNR) is estimated with the Decision-Directed approach [30] and the noise Power Spectral Density (PSD) is estimated with the noise PSD tracker in [31]. The noise PSD estimate is initialized with the first 1000 samples of each utterance, assumed to be a speech-free region.

For the DNN-SE algorithm, we use the same procedure and parameters of [6]. The IRM is estimated by using a DNN with three hidden layers of 1024 units each, and an output layer with 64 units. The input of the DNN is a 1845-dimensional feature vector, which is a robust representation of a frame that combines MFCCs, amplitude modulation spectrogram, relative spectral transform - perceptual linear prediction (RASTA-PLP), and gammatone filter bank energies, with their delta and double delta for a context of 2 past and 2 future frames. The training label is represented by the IRM, which is computed as in [32] from the T-F representation based on a gammatone filter bank with 64 filters linearly spaced on a Mel frequency scale and with a bandwidth equal to one equivalent rectangular bandwidth [33]. The system is trained for 30 epochs with SGD, using the mean square error as error function and a batch size of 1024. In order to enhance a test signal, the DNN provides an estimation of the IRM which is applied to the T-F representation of the noisy signal. Finally, the time domain signal is synthesized.

3.3. Datasets

We use two corpora, TIMIT [34] and RSR2015 [35], as follows:

- Set 1 (TIMIT) - 4380 utterances of male speakers are used for UBM training.
- Set 2 (RSR2015) - Text ID from 2 to 30 of sessions 1, 4, and 7 for 50 male speakers (from m051 to m100) are selected to train Pix2Pix and DNN-SE.
- Set 3 (RSR2015) - Text ID 1 of sessions 1, 4, and 7 for 49 male speakers (from m002 to m050) are used to train the speaker models.
- Set 4 (RSR2015) - Sessions 2, 3, 5, 6, 8, and 9 of the same text ID and speakers used for training the models, are selected for evaluation.

The choice of RSR2015 as the main database for training and testing can be seen as a compromise, because we were interested in the evaluation of an ASV system, which provides another objective measure of the performance, and RSR2015 is widely used for this task.

We used 5 different noise types to simulate real-life conditions: Babble, obtained by adding 6 random speech samples from the Librispeech corpus [36]; white Gaussian noise generated in MATLAB; Cantine, recorded by the authors; Market and Airplane, collected by Fondazione Ugo Bordoni (FUB) and available on request from the OCTAVE project [37]. Noise data, which were added to the utterances in Set 2, 3, and 4 at different SNR values, used for training and testing are different.

3.4. Setup

Inspired by [2], we investigate two different kinds of Pix2Pix-based SE front-ends: 5 noise specific front-ends (NS-Pix2Pix), each of them trained on only one type of noise, and 1 noise general front-end (NG-Pix2Pix), trained on all types of noise. The same has been done for the DNN-SE front-ends, obtaining 5 noise specific front-ends (NS-DNN) and 1 noise general front-end (NG-DNN). For training, we add noise to clean speech at two different SNRs, 10 and 20 dB. With higher SNR it should be easier to train a G able to capture the underlying structure of the noisy input and generate a clean spectrogram, but a test with lower SNRs for training is worth to explore in the future. For testing, results for 5 different SNR conditions are reported: 0, 5, 10, 15, and 20 dB, as is commonly done for ASV, but an interesting future work is to test on lower SNRs, particularly for intelligibility evaluation. In addition, to find the behavior of the front-ends on noise free conditions, ASV performance on enhanced clean speech data is also reported.

In all the tests, the performance of the following front-ends are presented: No enhancement (when no SE algorithm is used on noisy data), STSA-MMSE, NS-DNN, NS-Pix2Pix, NG-DNN, and NG-Pix2Pix. In total, three different tests have been conducted:

- Test 1 - In the first test, we compute PESQ and STOI for the different front-ends to estimate speech quality and intelligibility.
- Test 2 - In the second test, the ASV system is trained with enhanced clean speech (except for the No enhancement front-end where clean speech is used) and tested on the 5 types of noise.
- Test 3 - The last test is performed to evaluate the effects of a multi-condition training on ASV. For No enhancement, STSA-MMSE, NS-DNN, and NS-Pix2Pix the speaker models are built from enhanced clean speech and one kind of enhanced noisy speech, while for NG-DNN and NG-Pix2Pix all kinds of noise are used.

4. Results and Discussion

The results of Test 1 are shown in Table 1. It is observed that the average PESQ scores of NS-Pix2Pix and NG-Pix2Pix are always better than the other front-ends. The best performance improvement is achieved between 5 and 15 dB SNR regardless of the noise type. At 20 dB, our approach outperforms the others on Market and White noises, but for Airplane noise STSA-MMSE is the best one, while for Babble and Cantine noises the absence of enhancement is superior indicating that all the SE techniques introduce an amount of distortion surpassing the benefit of noise reduction. At 0 dB, NG-Pix2Pix generally outperforms the noise specific version with an exception (Market noise) and its scores are close to DNN-SE ones.

In terms of STOI, Pix2Pix front-ends perform similarly to STSA-MMSE. However, DNN-SE front-ends are superior in almost all the conditions, even though Pix2Pix front-ends achieve the same or very close results in some situations, e.g. low SNRs for Cantine and Market noises. At 20 dB, we observe the same behavior as the PESQ scores, where the evaluation of not enhanced signals gives a better outcome.

Table 1: *PESQ and STOI performance for the 5 front-ends: No enhancement (a), STSA-MMSE (b), NS-DNN (c), NS-Pix2Pix (d), NG-DNN (e), NG-Pix2Pix (f).*

		PESQ						STOI					
	SNR	0	5	10	15	20	mean	0	5	10	15	20	mean
Airplane	(a)	1.34	1.63	2.02	2.47	3.00	2.09	0.64	0.74	0.82	**0.88**	**0.93**	0.80
	(b)	1.54	1.79	2.17	2.72	**3.26**	2.30	0.66	0.74	0.81	0.87	0.91	0.80
	(c)	1.65	1.94	2.30	2.73	3.16	2.36	**0.69**	**0.76**	**0.83**	**0.88**	0.92	**0.82**
	(d)	1.57	2.02	**2.51**	**2.91**	3.18	2.44	0.66	0.75	0.81	0.85	0.89	0.79
	(e)	1.65	1.94	2.29	2.70	3.14	2.35	**0.69**	**0.76**	0.82	0.87	0.91	0.81
	(f)	**1.67**	**2.07**	**2.51**	2.88	3.13	**2.45**	0.67	0.74	0.79	0.83	0.86	0.78
Babble	(a)	1.20	1.42	1.79	2.40	**3.13**	1.99	0.44	0.56	0.67	0.77	0.85	0.66
	(b)	1.14	1.31	1.61	2.07	2.65	1.76	0.43	0.56	0.66	0.74	0.81	0.64
	(c)	**1.25**	1.51	1.87	2.31	2.78	1.95	**0.50**	**0.63**	**0.72**	**0.79**	**0.86**	**0.70**
	(d)	1.20	1.48	1.98	2.52	2.93	2.02	0.46	0.59	0.71	0.78	0.83	0.67
	(e)	1.24	**1.52**	1.88	2.31	2.78	1.95	0.49	0.62	**0.72**	**0.79**	0.85	**0.70**
	(f)	1.20	1.49	**2.00**	**2.53**	2.93	**2.03**	0.46	0.60	0.71	0.77	0.82	0.67
Cantine	(a)	1.35	1.65	2.07	2.57	**3.30**	2.19	0.54	0.66	0.75	**0.83**	**0.90**	0.74
	(b)	1.38	1.68	2.12	2.67	3.23	2.22	0.55	0.66	0.74	0.82	0.87	0.73
	(c)	1.46	1.75	2.15	2.63	3.12	2.22	0.59	**0.69**	0.76	**0.83**	0.89	0.75
	(d)	1.45	1.84	2.38	**2.82**	3.13	2.32	0.58	0.68	0.75	0.80	0.85	0.73
	(e)	1.47	1.77	2.18	2.64	3.11	2.24	**0.60**	**0.69**	**0.77**	**0.83**	0.89	**0.76**
	(f)	**1.49**	**1.91**	**2.43**	2.81	3.08	**2.34**	0.59	**0.69**	0.75	0.80	0.84	0.73
Market	(a)	1.26	1.51	1.89	2.38	3.04	2.02	0.51	0.62	0.73	0.81	**0.88**	0.71
	(b)	1.24	1.45	1.76	2.22	2.79	1.89	0.51	0.62	0.71	0.79	0.85	0.70
	(c)	1.35	1.63	2.00	2.46	2.94	2.08	**0.56**	**0.67**	**0.75**	**0.82**	**0.88**	**0.73**
	(d)	**1.36**	1.71	2.21	**2.72**	**3.09**	**2.22**	0.55	0.66	0.74	0.80	0.85	0.72
	(e)	**1.36**	1.63	2.00	2.45	2.93	2.07	**0.56**	**0.67**	**0.75**	**0.82**	**0.88**	**0.73**
	(f)	1.35	**1.72**	**2.24**	2.68	3.02	2.20	**0.56**	**0.67**	0.74	0.79	0.83	0.72
White	(a)	1.15	1.31	1.60	2.01	2.57	1.73	0.50	0.61	0.72	0.81	**0.89**	0.71
	(b)	1.35	1.58	1.88	2.25	2.71	1.95	0.53	0.63	0.73	0.81	0.87	0.72
	(c)	**1.38**	1.66	2.00	2.39	2.88	2.06	**0.58**	**0.67**	**0.75**	**0.82**	0.88	**0.74**
	(d)	1.23	1.54	2.11	**2.74**	**3.14**	2.15	0.53	0.64	0.73	0.80	0.86	0.71
	(e)	1.35	1.63	1.96	2.29	2.65	1.98	0.57	0.66	0.74	0.81	0.88	0.73
	(f)	1.32	**1.69**	**2.22**	2.68	3.01	**2.19**	0.55	0.65	0.73	0.78	0.83	0.71

The ASV performances (Tests 2 and 3) are reported in Tables 2 and 3, where the results of the baseline systems are from [38]. For the clean speaker models, Pix2Pix front-ends generally outperform the baseline methods. One exception is seen for Babble noise, where the NG-DNN front-end gives an EER of 8.73%, marginally better than NS-Pix2Pix (8.76%). At low SNR, DNN-SE front-ends sometimes show better results than Pix2Pix, but overall our approach can be considered superior.

On the other hand, the performances of DNN-SE front-ends on multi-condition training are generally better, which presents a substantial improvement if compared with the clean speaker model situation. Our approach is generally better than STSA-MMSE, although the NS-Pix2Pix front-end shows lower per-

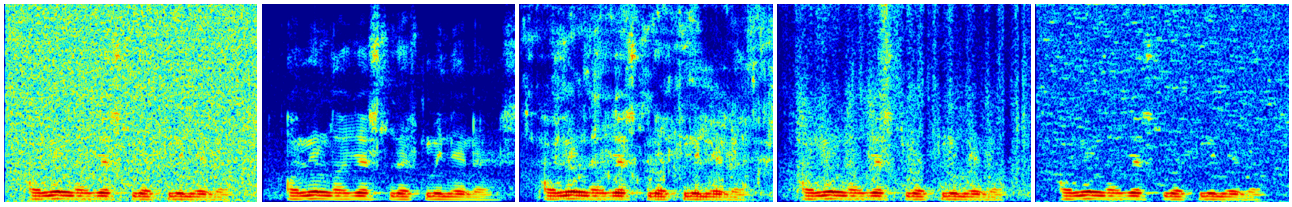


Figure 2: *From left to right: noisy spectrogram (White noise at 0 dB SNR); clean spectrogram; spectrogram of the signal enhanced with NG-Pix2Pix; spectrogram of the signal enhanced with NG-DNN; spectrogram of the signal enhanced with STSA-MMSE.*

Table 2: *ASV performance in terms of EER on clean speaker model*

	SNR	0	5	10	15	20	clean	mean
Airplane	No enhancement	21.09	15.99	13.61	11.66	9.18	6.99	13.08
	STSA-MMSE	17.69	12.58	8.17	6.53	6.27	5.80	9.51
	NS-DNN	16.99	10.55	7.48	6.99	6.15	6.12	9.05
	NS-Pix2Pix	17.19	8.84	**5.44**	5.05	**4.63**	**3.74**	7.48
	NG-DNN	15.99	8.99	6.12	6.12	5.58	5.67	8.08
	NG-Pix2Pix	**15.31**	**7.89**	5.58	**4.77**	4.76	5.44	**7.29**
Babble	No enhancement	19.05	14.63	11.69	11.04	9.18	6.99	12.10
	STSA-MMSE	29.04	20.40	12.59	7.82	6.29	5.80	13.66
	NS-DNN	17.01	10.54	7.82	6.46	6.12	5.78	8.96
	NS-Pix2Pix	18.83	11.22	7.62	**5.70**	5.10	**4.08**	8.76
	NG-DNN	**16.67**	**10.39**	**7.50**	6.34	5.78	5.67	**8.73**
	NG-Pix2Pix	21.05	13.64	8.50	5.97	**4.76**	5.44	9.90
Cantine	No enhancement	20.72	19.20	14.74	11.81	8.50	6.99	13.66
	STSA-MMSE	19.09	12.37	8.16	6.80	6.12	5.80	9.72
	NS-DNN	18.71	**8.58**	6.12	5.49	5.31	5.10	8.22
	NS-Pix2Pix	**17.33**	9.18	**5.44**	**5.10**	5.10	**4.16**	**7.72**
	NG-DNN	19.94	9.18	6.12	5.78	5.44	5.67	8.69
	NG-Pix2Pix	17.57	8.84	5.73	5.31	**4.76**	5.44	7.94
Market	No enhancement	29.40	20.07	15.00	11.96	8.93	6.99	15.39
	STSA-MMSE	25.51	17.35	11.90	8.28	7.35	5.80	12.70
	NS-DNN	21.43	**9.86**	**6.88**	6.46	5.78	5.92	9.39
	NS-Pix2Pix	**17.91**	10.33	7.14	**5.92**	5.17	**3.61**	**8.35**
	NG-DNN	21.77	10.59	7.48	6.22	5.76	5.67	9.58
	NG-Pix2Pix	19.58	11.22	7.48	6.12	**5.07**	5.44	9.15
White	No enhancement	45.90	43.20	34.61	26.28	16.91	6.99	28.98
	STSA-MMSE	30.95	21.17	13.95	10.20	8.50	5.80	15.10
	NS-DNN	39.46	20.75	9.86	7.82	6.12	6.02	15.01
	NS-Pix2Pix	40.48	28.23	12.45	7.86	6.46	6.46	16.99
	NG-DNN	40.14	21.77	10.88	8.16	6.80	5.67	15.57
	NG-Pix2Pix	**30.61**	**17.33**	**9.40**	**7.14**	**5.78**	**5.44**	**12.62**

Table 3: *ASV performance in terms of EER on multi-condition speaker model*

	SNR	0	5	10	15	20	clean	mean
Airplane	No enhancement	32.28	26.87	21.10	16.38	9.86	5.83	18.72
	STSA-MMSE	25.51	15.48	8.16	6.12	5.44	5.44	11.03
	NS-DNN	14.78	8.26	5.44	5.53	4.76	4.76	7.26
	NS-Pix2Pix	16.67	7.14	5.10	**4.03**	**3.78**	4.42	6.86
	NG-DNN	**11.38**	**6.12**	**4.78**	4.72	4.23	**4.00**	**5.87**
	NG-Pix2Pix	13.27	6.43	5.78	5.44	5.27	4.78	6.83
Babble	No enhancement	21.77	15.37	11.93	9.52	8.16	6.12	12.15
	STSA-MMSE	33.50	23.13	16.23	12.63	8.84	7.12	16.91
	NS-DNN	16.26	9.52	6.99	6.08	5.78	5.17	8.30
	NS-Pix2Pix	20.75	10.88	6.12	**4.76**	**4.08**	4.36	8.49
	NG-DNN	**16.00**	**9.18**	**5.44**	**4.76**	**4.08**	**4.00**	**7.19**
	NG-Pix2Pix	21.72	12.44	6.46	5.34	5.22	4.78	9.33
Cantine	No enhancement	24.11	17.22	12.93	10.88	9.18	7.48	13.63
	STSA-MMSE	19.05	12.59	8.21	6.91	6.12	6.32	9.87
	NS-DNN	12.93	5.91	**4.42**	4.25	4.27	**3.78**	5.93
	NS-Pix2Pix	14.29	6.87	4.76	**4.00**	**4.08**	4.76	6.46
	NG-DNN	**11.61**	**5.78**	5.10	4.57	**4.08**	4.00	**5.86**
	NG-Pix2Pix	14.10	7.48	5.44	5.44	5.27	4.78	7.08
Market	No enhancement	36.05	26.06	18.37	13.32	9.18	5.44	18.07
	STSA-MMSE	29.25	21.07	13.95	10.98	7.82	6.67	14.97
	NS-DNN	19.33	**8.16**	6.24	5.41	4.53	4.29	7.99
	NS-Pix2Pix	18.49	9.18	5.82	**4.42**	**3.74**	4.76	7.74
	NG-DNN	**18.37**	**8.16**	**5.78**	4.44	4.42	**4.00**	**7.53**
	NG-Pix2Pix	19.30	9.37	6.37	5.44	5.10	4.78	8.39
White	No enhancement	35.88	24.40	18.37	15.81	14.97	5.85	19.21
	STSA-MMSE	30.95	20.07	7.48	6.46	6.46	4.76	12.70
	NS-DNN	27.21	**9.52**	**6.12**	**5.02**	4.65	5.78	9.72
	NS-Pix2Pix	39.37	23.81	10.20	6.46	5.95	6.44	15.37
	NG-DNN	**26.19**	11.22	7.14	5.10	**4.08**	**4.00**	**9.62**
	NG-Pix2Pix	30.41	14.29	8.84	6.60	5.78	4.78	11.78

formance when it deals with white noise.

In general, Pix2Pix can be considered competitive with DNN-SE (better PESQ and EER on the clean speaker models, but worse STOI and EER for multi-condition training) and overall superior to STSA-MMSE.

Figure 2 shows the spectrograms of a noisy utterance (White noise at 0 dB SNR), together with its clean and enhanced versions with NG-Pix2Pix, NG-DNN, and STSA-MMSE. It is observed that the spectrogram enhanced by the cGAN approach preserves the structure of the original signal better than the other SE techniques, while at the same time more noises remain especially at high frequency regions, as compared with NG-DNN. The spectrogram enhanced by STSA-MMSE is snowy all over the places.

5. Conclusion

In this paper we investigated the use of conditional generative adversarial networks (cGANs) for speech enhancement. We adapted the Pix2Pix framework, intended to solve generic image-to-image translation problems, and evaluated the performance of this approach in terms of estimated speech perceptual quality and speech intelligibility, together with equal error rate of a Gaussian Mixture Model - Universal Background Model based speaker verification system. The results we obtained allow us to conclude that cGANs are a promising technique for speech denoising, being globally superior to the classical STSA-MMSE algorithm, and comparable to a DNN-SE algorithm.

Future work includes a more extensive evaluation of the framework in more critical SNR situations, and modifications aiming at making it specific for the task. For example, a model with G generating a small size output window from a fixed number of successive frames can be built as it is often done in deep neural networks for speech processing, and a specific perceptual loss to be added to the cGAN loss can be designed.

6. Acknowledgements

The authors would like to thank Hong Yu for providing data and speaker verification results for the baseline systems and Morten Kolbæk for his assistance and software used for the speaker verification and DNN speech enhancement baseline systems.

This work is partly supported by the Horizon 2020 OCTAVE Project (#647850), funded by the Research European Agency (REA) of the European Commission, and the iSocioBot project, funded by the Danish Council for Independent Research - Technology and Production Sciences (#1335-00162).

7. References

[1] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arXiv preprint arXiv:1611.07004*, 2016.

[2] M. Kolbœk, Z.-H. Tan, and J. Jensen, "Speech enhancement using long short-term memory based recurrent neural networks for noise robust speaker verification," in *Spoken Language Technology Workshop (SLT), 2016 IEEE*. IEEE, 2016, pp. 305–311.

[3] M. L. Seltzer, D. Yu, and Y. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 7398–7402.

[4] J. Chen, Y. Wang, S. E. Yoho, D. Wang, and E. W. Healy, "Large-scale training to increase speech intelligibility for hearing-impaired listeners in novel noises," *The Journal of the Acoustical Society of America*, vol. 139, no. 5, pp. 2604–2612, 2016.

[5] X. Lu, Y. Tsao, S. Matsuda, and C. Hori, "Speech enhancement based on deep denoising autoencoder," in *Interspeech*, 2013, pp. 436–440.

[6] M. Kolbæk, Z.-H. Tan, and J. Jensen, "Speech intelligibility potential of general and specialized deep neural network based speech enhancement systems," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 1, pp. 153–167, 2017.

[7] P. C. Loizou, *Speech enhancement: theory and practice*. CRC press, 2013.

[8] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, "A regression approach to speech enhancement based on deep neural networks," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 23, no. 1, pp. 7–19, 2015.

[9] S. R. Park and J. Lee, "A fully convolutional neural network for speech enhancement," *arXiv preprint arXiv:1609.07132*, 2016.

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

[11] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

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

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

[14] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

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

[16] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," *arXiv preprint arXiv:1609.04802*, 2016.

[17] H. Zhang, V. Sindagi, and V. M. Patel, "Image de-raining using a conditional generative adversarial network," *arXiv preprint arXiv:1701.05957*, 2017.

[18] S. Mobin and J. Bruna, "Voice conversion using convolutional neural networks," *arXiv preprint arXiv:1610.08927*, 2016.

[19] S. E. Reed, Y. Zhang, Y. Zhang, and H. Lee, "Deep visual analogy-making," in *Advances in Neural Information Processing Systems*, 2015, pp. 1252–1260.

[20] O. Mogren, "C-rnn-gan: Continuous recurrent neural networks with adversarial training," *arXiv preprint arXiv:1611.09904*, 2016.

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

[22] X. Wang and A. Gupta, "Generative image modeling using style and structure adversarial networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 318–335.

[23] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2536–2544.

[24] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 234–241.

[25] Y.-C. Lin, "pix2pix-tensorflow," Github repository: https://github.com/yenchenlin/pix2pix-tensorflow, 2016, accessed: March 2017.

[26] A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra, "Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, vol. 2. IEEE, 2001, pp. 749–752.

[27] ITU, "Wideband extension to recommendation p.862 for the assessment of wideband telephone networks and speech codecs," Available: https://www.itu.int/rec/T-REC-P.862.2-200511-S/en, 2005, accessed: March 2017.

[28] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "An algorithm for intelligibility prediction of time–frequency weighted noisy speech," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 2125–2136, 2011.

[29] A. K. Sarkar and Z.-H. Tan, "Text dependent speaker verification using un-supervised hmm-ubm and temporal gmm-ubm," *Proceedings of INTERSPEECH (to appear)*, 2016.

[30] Y. Ephraim and D. Malah, "Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 6, pp. 1109–1121, 1984.

[31] R. C. Hendriks, R. Heusdens, and J. Jensen, "Mmse based noise psd tracking with low complexity," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. IEEE, 2010, pp. 4266–4269.

[32] Y. Wang, A. Narayanan, and D. Wang, "On training targets for supervised speech separation," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 22, no. 12, pp. 1849–1858, 2014.

[33] D. Wang and G. J. Brown, *Computational auditory scene analysis: Principles, algorithms, and applications*. Wiley-IEEE Press, 2006.

[34] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1," *NASA STI/Recon technical report n*, vol. 93, 1993.

[35] A. Larcher, K. A. Lee, B. Ma, and H. Li, "Text-dependent speaker verification: Classifiers, databases and rsr2015," *Speech Communication*, vol. 60, pp. 56–77, 2014.

[36] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5206–5210.

[37] M. Falcone, B. Fauve, M. Cornacchia *et al.*, "Corpora collection," *OCTAVE (Objective Control of TAlker VErification), Deliverable 17*, 2016.

[38] H. Yu, Z.-H. Tan, Z. Ma, and J. Guo, "Adversarial network bottleneck features for noise robust speaker verification," *Proceedings of INTERSPEECH (to appear)*, 2017.