Audio Enhancing with Wasserstein GANs

Prof. A. Alahi, Brian Sifringer, Thomas Stegmüller
Visual Inteligence for Transportation Lab, EPFL, Switzerland
{alexandre.alahi, brian.sifringer, thomas.stegmuller}@epfl.ch

Abstract—**A direct consequence of the democratization of smartphones is the ubiquity of microphones. Nonetheless, achieving acceptable audio quality from such microphones remains a challenging task. In parallel, the introduction of the Generative Adversarial Network (GAN) framework in 2014 introduced a new paradigm in the field of machine learning and to unprecedented results in the context of image generation. In contrast, the field of audio has not been the subject of such transfiguration in the last few years. In this report we present a first step in the direction of making studio quality recording available to any beholder of a smartphone by extending the GAN framework to the domain of audio content.**

I. Introduction

In the past few years the number of available entry class microphones, as well as their quality has significantly increased, nevertheless the resulting audio quality is often disappointing and presents undesired artifacts. Such a low quality is not surprising considering the amount of transformations that can happen to the audio signal between the moment it is emitted and the one it is finally stored in memory.

The first transformation that is applied to the signal is due to the fact that the room is rarely anechoic and its walls will therefore reflect and absorb sounds dependently on the frequency content and amplitude of the emitted signal. The second transformation is due to the directivity of the microphone, which is a desired property when having a phone call, but not when recording a potentially multi-sources signals. Once the signal finally reaches the microphone it is further deteriorated by the mechanical and electrical non-linearity of the membrane and coil respectively. Before the sampling and quantization can occur, one should further expect the unavoidable resistor's thermal noise to be added.

From the above paragraph it is quite apparent that amateur audio recordings are prone to artifacts and would strongly benefit from a post-processing stage. A software solution does indeed seem like the most viable solution considering that control over the recording environment is not always achievable, while hardware improvement is impeded by space and budget constraints.

As in any machine learning task the choice of the dataset is decisive. After multiple tries at reproducing the effect of a low quality recording on audio samples, we first observed, that the reproduction where a poor representation of the desired effect, and secondly, that the alterations on the original signal where far from being uniformly present across a given track. This last remark is important as it would make training on such a dataset significantly less accessible. Considering the scarcity of the literature on the topic of audio enhancing, we opted for a proxy task from which we hope to gain insight on the methods that can be used to solve the original task. The proxy task consists of learning how to change the instrument being played in the track. The details regarding the design of the dataset can be found in Section III.

II. Related work

Our task lies at the intersection of style transfer and super-resolution. As opposed to images, the application of these topics to musical audio content was not the subject of great interest in the past few years.

Indeed, most of the recent work related to audio content is devoted to the field of speech-synthesis (Sotelo et al. 2017, Oord, Li, et al. 2017). Although musical content has received some attention, it is mostly focused towards its generation using auto-regressive methods (Oord, Dieleman, et al. 2016, Mehri et al. 2016). One of the first successful application of the GAN framework to audio was produced by Donahue et al. 2018 who showed promising synthesis results.

In the range of applications closer to our task, Kuleshov et al. 2017 proposed a method based on Convolutional Neural Networks (CNN) to efficiently increase the temporal resolution of speech and musical signals. Their approach worked with up-scaling ratios as high as 6, and showed results that significantly exceeded previous scores on standard metrics. Their method was further improved by Kim et al. 2018, who extended their work by adapting the GAN framework to audio super-resolution tasks and including some notions borrowed from images style transfer, such as using the features maps of a pre-trained model to provide perceptually relevant feedback on the quality of the generated samples.

More recently, Md Shahrin et al. 2020 proposed an insightful review on the limitations of interpreting spectrogram as images. In particular, they shed light on the intrinsic asymmetries between the frequency and time dimension of a spectrogram, and its incoherence with the equivariance in translation of CNN architectures. Some possible remedy to this issue are provided, among which using the Mel-Spectrogram representation, or aligning the frequencies with the channel dimension.

III. Dataset

As stated in Section I the task that we aim to tackle is the mapping in time domain from one instrument to another. Considering that our implementation is principally based on the work of Kim et al. 2018, and that the presented methods require pairs of input/target signals, it is needed to dispose of a dataset consisting of pairs of synchronized tracks played by two distinct instruments. To our knowledge such a dataset does not exist, fortunately, the MIDI file format can be used to create one.

A. The MIDI file format

The MIDI file format is a standardized protocol to store and communicate audio content based on a symbolic representation. Meaning that no audio is actually stored, only the information to synthesize it. As in traditional audio format, a track is composed of multiple channels. Each channel is a stream of messages encoding an event. An event can either be a note, in which case the message stores its pitch and intensity, or it can be a control signal, in that case the message encodes the type of the control (volume, reverb, chorus, etc.), as well as its intensity. On top of that, the file also stores meta-data, such as the tempo or the instrument to play.

B. Design of the dataset

From the above paragraph, it is clear that changing the instrument in a MIDI file sums up to a simple change of meta-parameter, and is thereby a convenient way to build the required dataset. By no mean do we claim that the quality of a sound under that format comes close to a non-synthetic one, but it is of minor interest to us as we are mostly concerned by the relative difference between the two synthetic tracks.

To create the required pairs of signals we used the MAESTRO dataset (Hawthorne et al. 2019) which is composed of more than 200 hours of piano solos, as well as the corresponding MIDI file for each audio track. As some tracks are over 30 minutes, it is obviously impossible to feed them directly to the models. Based on 20 Hz-20 kHz as a rough estimate of the range of frequencies perceptible by human ears, and a sampling frequency of 44.1 kHz we compute a minimum of over 2200 samples needed to capture a single full period of the lowest frequency content. As will be further discussed in Section IV, the models are well suited to have entry signals with power of 2 length and hence we choose 8192 as the window length.

When cropping signals, one has to consider how to optimize the reconstruction that will take place later on. A well documented way to do so is the overlap-and-add (OLA) method with Hanning windows.The latter is often used to compute the Short-time Fourier transform (STFT). Indeed the Hanning window has the nice property that the fade-in and fade-out parts of the window sum up to 1 when the overlap is set to 50 %. We chose to crop the signals using a rectangular window and only apply the Hanning trick at the reconstruction time. It is true that cropping the signal in such a crude way introduces artifacts as the Fourier transform of a rectangle contains infinite frequencies, but we believe that the latter is preferable than feeding the models with windowed signals which tend to have attenuated high frequencies near the boundaries.

IV. Method

In Section II some of the existing methods that have been used on tasks similar to ours were presented. The work of Kuleshov et al. 2017 and Kim et al. 2018 on the topic of audio super-resolution (SR) seemed like the most viable path to follow for multiple reasons. The first being the resemblance between the SR task and ours. Secondly, their approach treats the audio signal in time domain as opposed to other methods based on spectrogram representation of the signal. The spectrogram based approaches benefit from the abundant literature on the field of image transformation and style transfer, but incurs the strong disadvantage that a specific iterative algorithm (Griffin et al. 1984) is needed to recover the raw audio from the modified spectrograms. That last point is peculiarly problematic as it could restrain us from discerning artifacts due to our method from the ones due to the limitation of the Griffin-Lim algorithm. A last point in favor of the two aforementioned methods is that they operate on pairs of input/target signals which is a significantly easier setup than the usual unsupervised style transfer task proposed by Gatys et al. 2016.

A. The generative adversarial framework

In the original GAN framework introduced by Goodfellow et al. 2014, two models, namely a generator (G) and a discriminator (D) with parameters θ_G and θ_D, are trained jointly. In our specific application the generator learns the mapping from an input signal, x_i, to a target signal, x_t. On the other hand the discriminator learns to distinguish samples, x_r, from the real distribution, p_r, from the generated ones, $x_g = G(x_i)$, i.e. sampled from the generator distribution, p_g. This can be formally stated as the following two player minimax game:

$$\min_G \max_D \mathbb{E}_{x_t\sim p_r}\left[\log\left(D(x_t)\right)\right] + \mathbb{E}_{x_g\sim p_g}\left[\log\left(1-D(x_g)\right)\right] \quad (1)$$

From this formulation, the natural losses for the discriminator and generator are:

$$\mathcal{L}_D = -\left[\log\left(D(x_t)\right) + \log\left(1-D(G(x_i))\right)\right] \quad (2)$$
$$\mathcal{L}_G = \log\left(1-D(G(x_i))\right) \quad (3)$$

In practice, it is common to use the following alternative, also know as the non-saturating loss, for the generator:

$$\mathcal{L}_G = -\log\left(D(G(x_i))\right) \quad (4)$$

The modification brought by Eq. 4 leverages the problem that can happen when the generator is poor in comparison to the discriminator who provides weak gradients, hence takes longer to converge. In the original paper they proceed to show that optimizing Eq. 1 for a fix generator and a discriminator with potentially infinite capacity yields the optimal discriminator:

$$D^* = \frac{p_r}{p_r + p_g} \quad (5)$$

Replacing D by D^* in Eq. 1 yields:

$$\begin{aligned}&\mathcal{L}\left(D^*, G\right)\\ &= \mathbb{D}_{KL}\left(p_r, \frac{p_r+p_g}{2}\right) + \mathbb{D}_{KL}\left(p_r, \frac{p_r+p_g}{2}\right) - \log(4) \quad (6)\\ &= 2\mathbb{D}_{JS}\left(p_r, p_g\right) - \log(4) \quad (7)\end{aligned}$$

Where $\mathbb{D}_{KL}$ is the Kullback-Leibler divergence, while $\mathbb{D}_{JS}$ is the Jensen-Shannon divergence which is minimized when $p_r = p_g$. Although intellectually pleasing, the above result does not provide any guarantee in practice as Eq. 5 only holds under full optimization of the discriminator and sufficient capacity for the discriminator.

B. Wasserstein GAN

In the previous sub-section we refreshed some of the theoretical properties of the Vanilla GANs, we'll now discuss some of their weakness, as revealed by Arjovsky et al. 2017. It was observed that optimizing a generator under the GAN framework amounted to minimizing the Jensen-Shannon divergence between the real data distribution, p_r, and the generator distribution, p_g. Although this metric does not suffer from the same problem as the Kullback-Leibler divergence, which is infinite when p_r and p_g do not overlap, it still does a poor job of capturing the closeness of two distributions. Let's consider an example proposed by Fleuret 2020 to illustrate the limitation of the $\mathbb{D}_{JS}$ divergence. Let's define the two distributions as depicted in Fig. 1:

$$p_r(t) = \frac{1}{2}\mathbb{1}_{t\in[0,1]} + \frac{1}{2\delta}\mathbb{1}_{t\in[x_0,x_0+\delta]} \tag{8}$$

$$p_g(t) = \frac{1}{2}\mathbb{1}_{t\in[0,1]} + \frac{1}{2\delta}\mathbb{1}_{t\in[x_0+x,x_0+x+\delta]} \tag{9}$$

Where $\delta \in \mathbb{R}^+$ and $x_0 \in [0, 1-\delta]$ while x is a learnable parameter. The Jensen-Shannon divergence yields the following measure for the similarity between the two distributions:

$$\mathbb{D}_{JS} = \min(\delta, |x|)\left(\frac{1}{\delta}\log\left(\frac{1+2\delta}{2\delta}\right) - \left(\frac{1+\delta}{\delta}\right)\log\left(\frac{1+\delta}{\delta}\right)\right) \tag{10}$$

One can observe from Eq. 10 that whenever $|x| > \delta$ the

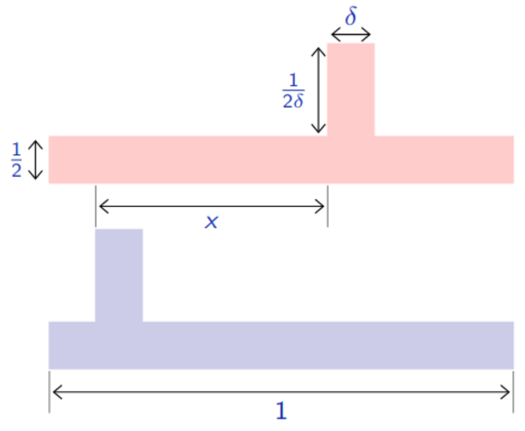

Fig. 1. Distributions illustrating the limitation of $\mathbb{D}_{JS}$, Fleuret 2020

similarity measure is independent of x and hence convergence of p_g towards p_r cannot be hoped for as the gradient is null. As claimed in Arjovsky et al. 2017, the Earth-Mover (EM) distance, also known as Wasserstein-1 is a better candidate cost function to optimize the generator on:

$$\mathbb{W}(p_r, p_g) = \inf_{\gamma\sim\Pi(p_r,p_g)} \mathbb{E}\left[||x-y||\right] \tag{11}$$

Where $\Pi(p_r, p_g)$ is the set of all joint distributions $\gamma(x, y)$ with marginals p_r and p_g. Stated in a more accessible way; $W(p_r, p_g)$ measures the minimum mass displacement to transform p_r into p_g or vice-versa as the EM measure is actually a distance. Thank to a duality theorem from Kantorovich and Rubinstein (Villani 2008) Eq. 11 can be re-written under a more tractable formulation:

$$\mathbb{W}(p_r, p_g) = \sup_{||f||_L\leq 1} \mathbb{E}_{x_t\sim p_r}\left[f(x_t)\right] - \mathbb{E}_{x_g\sim p_g}\left[f(x_g)\right] \tag{12}$$

Where $||f||_L \leq 1$ is the set of 1-Lipschitz functions. If we let the discriminator play the role of f, we obtain the following formulation for the optimization problem:

$$G^* = \arg\min_G \mathbb{W}(p_r, p_g) \tag{13}$$

$$= \sup_{||D||_L\leq 1} \mathbb{E}_{x_t\sim p_r}\left[D(x_t)\right] - \mathbb{E}_{x_g\sim p_g}\left[D(x_g)\right] \tag{14}$$

The Lipschitz constraint can either be enforced by weight clipping (Arjovsky et al. 2017) or gradient penalty (Gulrajani et al. 2017)

C. Architecture overview

Our architecture is inspired by Kim et al. 2018 and relies on three models, namely a generator (G), a discriminator (D) and an autoencoder (A) that all work with the natural representation of the signals, i.e. in time domain. As explained in Section IV-A, the generator learns the mapping from an input sample x_i to a target sample x_t, and receives feedback on the likelihood of the generated samples from the discriminator whose objective is to tell apart generated and real samples. Finally, the autoencoder provides additional feedback to the generator with respect to the high-level features of the generated samples. All three models share common properties: (i) they implement a variation of the inception modules (Szegedy et al. 2015), (ii) they rely on subpixel respectively superpixel layers to increase and decrease the spatial resolution along the temporal dimension (Shi et al. 2016).

D. Inception modules

Inception modules have empirically been proven to be improve upon standard architectures, as they mitigate the problem of choosing kernel sizes. Due to the harmonic property of musical audio content, applying kernels of different sizes does seem like a reasonable inductive bias. It was observed in Section III-B that low frequency content spreads over thousands of samples and therefore small kernels will only encompass a full period in the high-level features layers. This issue can be mitigated by working with larger filters. It is noteworthy that to effectively reduce the index of the first layer at which a period is fully captured, one has to increase the kernel size exponentially. This last remark further motivates the need for inception modules.

As proposed in Kim et al. 2018, the kernels have dimensions $3\times1/9\times1/27\times1/81\times1$ with respective number of filters in the first inception module $24/24/8/8$, hence totaling 64 filters, not counting the ones used to reduce the channel dimension [1]. The fraction of the total number of filters attributed to each kernel size remains constant throughout the model. This last point would probably benefit from further exploration and would be the subject of minor changes in the implementation. We implemented the version of inception modules with dimensionality reduction as depicted in Fig. 2. Concretely, this translates to first convolving the input features map with C_{temp} filters of dimension 1×1 before applying the second convolution on the

[1] From here on the 1x1 filters will be systematically ignored when counting the total number of filters.

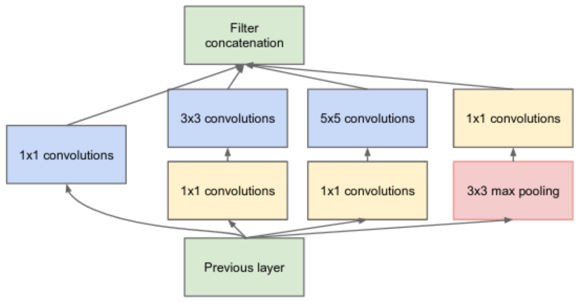

Fig. 2. Original inception module with dimensionality reduction, Szegedy et al. 2015

resulting activation. This operation is repeated independently and with adequate padding for each kernel size, such that output features maps can easily be stacked along the channel dimension. The parameter C_{temp} is chosen to be a fraction of the number of filters involved in the second convolution and is therefore not necessary the same number for each kernel size [2].

E. Subpixel and superpixel layers

Subpixel layers were introduced by Shi et al. 2016 as an efficient solution to increase spatial resolution without introducing checkerboard artifacts. In the 1-dimensional case, it works by placing entries from the channel dimension in the temporal dimension, thereby changing the overall shape of the input tensor from $C \times W$ to $\frac{C}{r} \times rW$, where r is the up-scaling ratio. Superpixel layers are the inverse operator of subpixel layers and are thus used to decrease the resolution along the temporal resolution. Our 1-dimensional implementation of the aforementioned layers respects the property that chaining a subpixel layer with a superpixel layer is the identity operator (Fig. 3).

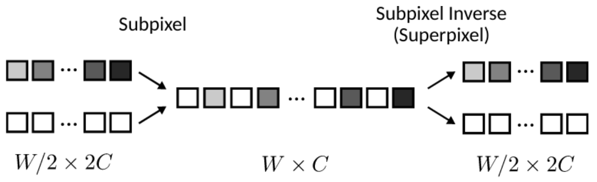

Fig. 3. 1D representation of subpixel and superpixel layers, Kim et al. 2018

F. Generator

The generator network is almost identical to the one proposed in Kim et al. 2018 (Fig. 4). It implements a U-Net type of architecture (Ronneberger et al. 2015). The compressing part of the generator is composed of $B_g$down-sampling blocks, similarly the expanding part is composed of $B_g$up-sampling blocks, in our case $B_g = 8$. The model advantageously uses skip-connections as (i) it helps the reconstruction by providing additional information to the expanding part in the forward-pass and (ii) it helps the training of the compressing part by preventing vanishing gradients in the backward-pass. It was

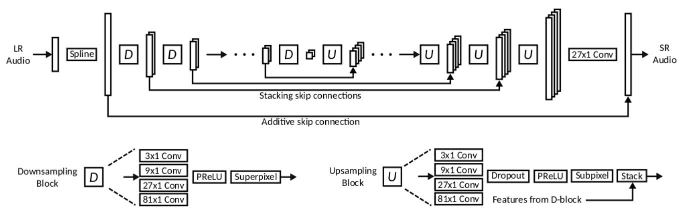

Fig. 4. Base architecture of the generator, Kim et al. 2018

experimentally observed that the additive skip-connection from the input to the output of the generator was potentially harmful, or at least not beneficial in the context of our application. Nevertheless, we do not claim that a similar conclusion can be drawn for a super-resolution task, where it does indeed seem like a reasonable prior to constrain the model to only learn to generate the high frequency content. Similarly to the approach proposed by Kuleshov et al. 2017, we only double the number of filters per down-sampling block for the first d_{max} of them. Therefore, the d^{th} down-sampling block for $d \in \{0, ..., B_g - 1\}$ contains $2^{6+min(d, d_{max})}$ filters. Correspondingly, the d^{th} up-sampling block contains $2^{6+min(B_g-1-d, d_{max})}$ filters. Finally, the number of input channels of the d^{th}, $d \in \{1, ..., B_g - 1\}$, up-sampling block can be computed by adding half the number of filters of the $(d-1)^{th}$ up-sampling block and twice the number of filters of the $(B_g - d - 2)^{th}$ down-sampling block. The factors $\frac{1}{2}$ and 2 in the previous computation are due to the subpixel respectively superpixel layers. The parameter d_{max} is typically used as a capacity regulator.

G. Discriminator

The discriminator network is almost identical to the one proposed in Kim et al. 2018 (Fig. 3). It is based on multiple inception modules to implement the features extraction and relies on two fully-connected layers to implement the decision making part of the model. When the discriminator is used

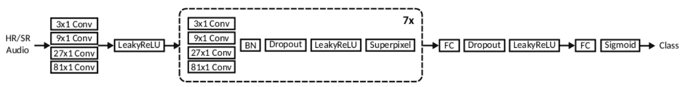

Fig. 5. Base architecture of the discriminator, Kim et al. 2018

in the Wasserstein GAN framework, a minor modification is needed to ensure that the Lipschitz constraint is respected. As the constraint is realized by penalizing the discriminator's gradient on individual and independent samples, batch normalization (Ioffe et al. 2015) is not adapted as it correlates the samples inside a given batch (Gulrajani et al. 2017) and it is therefore recommended to instead use layer normalization (Ba et al. 2016). A collateral benefit from this type of normalization is that its stability is independent of the batch size

[2]Implementation is available at: https://github.com/stegmuel/audio-super-resolution

H. Autoencoder

The autoencoder network is in fact a generator without any skip-connection, and a smaller number of blocks $B_a = 4$. Since the autoencoder's role is to provide a way to compare generated and real samples with respect to high-level features, we are mostly interested in the embedding space values as opposed to the quality of its output. Hence, it is desired that all the information flows through the bottleneck, which motivates the absence of skip-connections.

I. Loss functions

Multiple losses are needed and used to encourage the generation of perceptually pleasing results. The main loss for the generator is the Mean Squared Error (MSE) in time domain. Its formula for a single sample of width W is the following: [3]:

$$\mathcal{L}_{time} = \frac{1}{W} \left\| x_t - x_g \right\|_2^2 \tag{15}$$

Training solely with this loss tends to produce results which lack, what could be described as sharpness. This observation can be attributed to (i) the low level of the features on which the samples are compared and (ii) the averaging property of the loss.

A first potential step in the direction of a more appropriate loss is to base the evaluation of the reconstruction on the power spectrogram of the signals. With the convention that X is the Short-time Fourier transform of x taken on K frequency bins and W time bins we get:

$$\mathcal{L}_{freq} = \frac{1}{KW} \left\| |X_t|^2 - |X_g|^2 \right\|_F^2 \tag{16}$$

A second step is to incorporate what is often designated as the content loss in the domain of style transfer (Johnson et al. 2016). The embedding space with shape $C_{ae} \times W_{ae}$ of a pre-trained autoencoder can be used for that purpose. In particular, if we let $\phi(x)$ be the representation of x at the bottleneck of the autoencoder, we obtain the following loss:

$$\mathcal{L}_{ae} = \frac{1}{C_{ae} W_{ae}} \left\| \phi(x_t) - \phi(x_g) \right\|_F^2 \tag{17}$$

Finally, the Wassersstein GAN framework is used with the objective of improving the perceived sharpness of the generated samples. It leads to the following additional loss for the generator:

$$\mathcal{L}_{adv} = -D(G(x_i)) \tag{18}$$

On the other hand, the discriminator is trained with respect to the following objective:

$$\mathcal{L}_{disc} = -\left[D(x_t) - D\left(x_g\right)\right] + \lambda_p \left(\left\| \nabla_{x_g} D(x_g) \right\|_2 - 1 \right)^2 \tag{19}$$

V. Experiments

This section is dedicated to the experimental procedures that were realised. Rather than aiming directly to obtain the most pleasing generated samples, we design a set of experiments in sort of determining which of the different losses and parameters are beneficial and which are not. All experiments are realised on a common dataset sampled from the large raw data as described in Section III. All optimizations in the subsequent sections are done with the ADAM optimizer, $1e-3$ as learning rate and the default values for the two exponential decay rates (Kingma et al. 2014).

[3]From here on the losses are always specified for a single sample.

A. Evaluation of the autoencoder

The first experiment consists in the evaluation of the autoencoder's ability to provide meaningful feedback on the generated samples. For that purpose, the autoencoder is trained with the objective of minimizing the L2-loss in time domain (Eq. 15). When training the autoencoder, we typically observed improved convergence rates when the input and target types of signals are passed separately and alternatively, i.e. back-propping the loss in-between. Since the role of the autoencoder is to compare high-level features of generated, x_g, and target samples, x_t, it is desired that their respective representation in the embedding space is discernable, $\phi(x_t) \neq \phi(x_g)$. Instead of comparing $\phi(x_g)$ to $\phi(x_t)$, we compare $\phi(x_i)$ to $\phi(x_t)$, which is a reasonable choice as the generated sample lies somewhere in-between the input and target signals. A possible method to achieve the desired result, is to use the t-SNE algorithm (Maaten et al. 2008) to visualize a low-dimensional representation of randomly selected channels at the bottleneck's output. The result obtained with a bottleneck of dimension 128×512, 10 batches, and 64 pairs of input/target signals per batch is depicted in Fig. 6. The result does not imply that a given

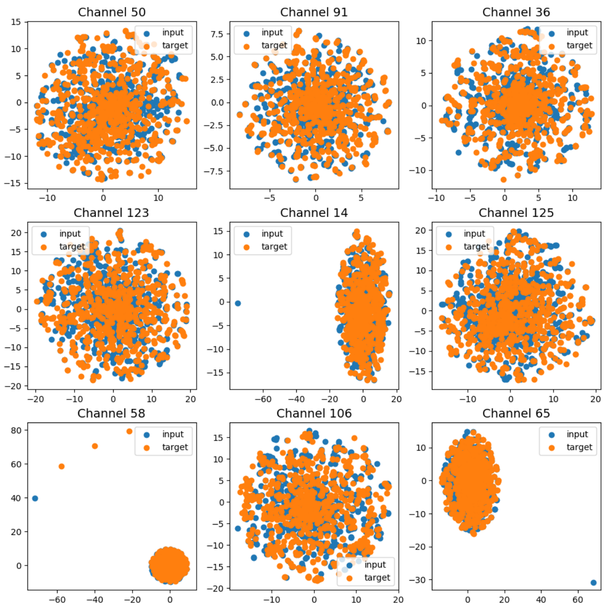

Fig. 6. Comparison of the low dimensional representation of randomly selected features from pairs of input and target type of signals.

pair of input/target signals is indistinguishable once mapped to the embedding space, but rather that as a whole inputs are as close to each other than they are to targets once encoded. Nonetheless, it could be beneficial to enforce clustering in the embedding space. The latter could be implemented by jointly training a classifier on the bottleneck's output and back-propping its loss to the autoencoder. Note that this could be implemented in a fully supervised way as the labels are available for free.

B. Evaluation of the different losses

The second experiment consists in the evaluation of the different losses. For that purpose, the generator is trained with the following composite losses:

$$\mathcal{L}_{g,1} = \mathcal{L}_{time} \tag{20}$$

$$\mathcal{L}_{g,2} = \mathcal{L}_{time} + \lambda_f \mathcal{L}_f \tag{21}$$

$$\mathcal{L}_{g,3} = \mathcal{L}_{time} + \lambda_{ae} \mathcal{L}_{ae} \tag{22}$$

$$\mathcal{L}_{g,4} = \mathcal{L}_{time} + \lambda_{adv} \mathcal{L}_{adv} \tag{23}$$

The parameters λ_f, λ_{ae}, and λ_{adv} are scalar values that can be used to fine tune the weight attributed to each loss. In this experiment, $\lambda_f = 10$ and $\lambda_{ae} = 1e-1$. This choice ensures that $\mathcal{L}_{time}$, $\mathcal{L}_{freq}$, and $\mathcal{L}_{ae}$have comparable level of importance in the weight update. This type of heuristic reasoning only makes sense if the loss function are identical and is thus not extended to the last setup. In the first three cases, the generator is trained until convergence; at which point the standard signal processing performance metrics are measured (Table I). The first metric is the signal-to-noise ratio (SNR):

$$\text{SNR}(x_g, x_t) = 10 \log_{10}\left(\frac{\|x_t\|_2^2}{\|x_t - x_g\|_2^2}\right) \tag{24}$$

The second metric is the log-spectral-distance (LSD). Using the same convention as in Section IV-I, it is defined as:

$$\text{LSD}(X_g, X_t) = \frac{1}{W}\sum_{w=1}^{W}\sqrt{\frac{1}{K}\sum_{k=1}^{K}\log_{10}\left(\frac{|X_g(w,k)|^2}{|X_t(w,k)|^2}\right)^2} \tag{25}$$

It is worth observing that minimizing the L2-loss in the time domain directly translates to maximizing the SNR metric. A similar observation can be made about the loss defined by Eq. 16 and the LSD metric. This last remark would actually be exact if the loss function was defined on the log-spectrogram instead of the power-spectrogram as we did. These two metrics are therefore not the most appropriate way to assess the perceptual quality of the generated samples as they are intrinsically biased towards specific losses. When analysing the results presented in Table I, we observe that (i) the results are coherent with the assertion made regarding the bias of these metrics and (ii) the SNR scores are quite low. Indeed, the scores are much lower than the ones typically achieved in super-resolution tasks (Kuleshov et al. 2017). Nonetheless we do believe that this is mostly due to the fact that our task is intrinsically more complicated. This is intuitively verified by noticing that our transformation is audibly much more noticeable than a down-scaling even at ratio as high as 6. Moreover, the SNR between input and target signals yields a value of -4.70 ± 3.23, whereas a simple spline taken on a piano sample down-scaled by a factor 6 already gives a SNR above 15 (Kuleshov et al. 2017). The SNR value obtained by comparing the input and target signals puts a perspective on the results of Table I, and provides a lower bound for the minimal improvement.

Composite loss name	SNR	LSD
$\mathcal{L}_{g,1}$	2.63 ± 5.21	1.65 ± 0.18
$\mathcal{L}_{g,2}$	1.59 ± 4.90	2.28 ± 0.18
$\mathcal{L}_{g,3}$	1.75 ± 5.21	2.21 ± 0.22
$\mathcal{L}_{g,4}$	-0.50 ± 5.42	2.46 ± 0.14

TABLE I
SNR and LSD metrics achieved by generator subject to different composite losses.

As one can observe from Table I the results achieved by the model trained with the adversarial criterion are the lowest. Audition of the generated samples confirms these values. These observations can be largely attributed to the fact under that criterion the training does not seem to converge and has to be arbitrarily stopped. Among the different things that were observed to improve stability, increasing the number of discriminator updates before doing a single generator update seems to be the most beneficial. Similarly, enforcing the Lipschitz constraint with a larger penalty also improves stability. A possible interpretation of the effectiveness of these two heuristic techniques is that they avoid large weight changes due the discriminator focusing on idiosyncrasies of the data.

VI. Conclusion

We present a first step in the direction of enhancing audio recordings with generative adversarial networks. As a collateral benefit we propose a new type of dataset that can easily accommodate different transformations and scales nicely. We put a significant emphasis on reviewing the different available technique and providing insights on the respective improvement they can bring to our application. Although our approach tackles the problem in a slightly indirect way, we strongly believe that our conclusions are also applicable to the original task.

Acknowledgements

The authors thank Yuejiang Liu for his careful advices on the training of generative adversarial networks.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein gan". In: *arXiv preprint arXiv:1701.07875* (2017).

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. "Layer normalization". In: *arXiv preprint arXiv:1607.06450* (2016).

[3] Chris Donahue, Julian McAuley, and Miller Puckette. "Adversarial audio synthesis". In: *arXiv preprint arXiv:1802.04208* (2018).

[4] François Fleuret. "11.2. Wasserstein GAN". EE559-Deep Learning. 2020.

[5] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. "Image style transfer using convolutional neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2414–2423.

[6] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[7] Daniel Griffin and Jae Lim. “Signal estimation from modified short-time Fourier transform”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 32.2 (1984), pp. 236–243.

[8] Ishaan Gulrajani et al. “Improved training of wasserstein gans”. In: *Advances in neural information processing systems*. 2017, pp. 5767–5777.

[9] Curtis Hawthorne et al. “Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset”. In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=r1lYRjC9F7.

[10] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *arXiv preprint arXiv:1502.03167* (2015).

[11] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. “Perceptual losses for real-time style transfer and super-resolution”. In: *European conference on computer vision*. Springer. 2016, pp. 694–711.

[12] Sung Kim and Visvesh Sathe. “Adversarial Audio Super-Resolution with Unsupervised Feature Losses”. In: (2018).

[13] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[14] Volodymyr Kuleshov, S Zayd Enam, and Stefano Ermon. “Audio super resolution using neural networks”. In: *arXiv preprint arXiv:1708.00853* (2017).

[15] Laurens van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE”. In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.

[16] Muhammad Huzaifah bin Md Shahrin and Lonce Wyse. “Applying visual domain style transfer and texture synthesis techniques to audio: insights and challenges”. In: *Neural Computing and Applications* 32.4 (2020), pp. 1051–1065.

[17] Soroush Mehri et al. “SampleRNN: An unconditional end-to-end neural audio generation model”. In: *arXiv preprint arXiv:1612.07837* (2016).

[18] Aaron van den Oord, Sander Dieleman, et al. “Wavenet: A generative model for raw audio”. In: *arXiv preprint arXiv:1609.03499* (2016).

[19] Aaron van den Oord, Yazhe Li, et al. “Parallel wavenet: Fast high-fidelity speech synthesis”. In: *arXiv preprint arXiv:1711.10433* (2017).

[20] 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*. Springer. 2015, pp. 234–241.

[21] Wenzhe Shi et al. “Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 1874–1883.

[22] Jose Sotelo et al. “Char2wav: End-to-end speech synthesis”. In: (2017).

[23] Christian Szegedy et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[24] Cédric Villani. *Optimal transport: old and new*. Vol. 338. Springer Science & Business Media, 2008.