ASSISTED SOUND SAMPLE GENERATION WITH MUSICAL CONDITIONING IN ADVERSARIAL AUTO-ENCODERS

Adrien Bitton, Philippe Esling, Antoine Caillon, Martin Fouilleul

ASSISTED SOUND SAMPLE GENERATION WITH MUSICAL CONDITIONING IN ADVERSARIAL AUTO-ENCODERS

Adrien Bitton, Philippe Esling, Antoine Caillon, Martin Fouilleul

Institut de Recherche et Coordination Acoustique-Musique (IRCAM)
CNRS - UMR 9912, UPMC - Sorbonne Universite
1 Place Igor Stravinsky, F-75004 Paris, France
adrien.bitton@ircam.fr

ABSTRACT

Deep generative neural networks have thrived in the field of computer vision, enabling unprecedented intelligent image processes. Yet the results in audio remain less advanced and many applications are still to be investigated. Our project targets real-time sound synthesis from a reduced set of high-level parameters, including semantic controls that can be adapted to different sound libraries and specific tags. These generative variables should allow expressive modulations of target musical qualities and continuously mix into new styles.

To this extent we train *auto-encoders* on an orchestral database of individual note samples, along with their intrinsic attributes: note class, *timbre domain* (an instrument subset) and *extended playing techniques*. We condition the decoder for explicit control over the rendered note attributes and use *latent adversarial training* for learning expressive style parameters that can ultimately be mixed. We evaluate both generative performances and correlations of the attributes with the latent representation. Our ablation study demonstrates the effectiveness of the *musical conditioning*.

The proposed model generates individual notes as magnitude spectrograms from any probabilistic latent code samples (each latent point maps to a single note), with expressive control of orchestral timbres and playing styles. Its training data subsets can directly be visualized in the 3-dimensional latent representation. Waveform rendering can be done offline with the *Griffin-Lim algorithm*. In order to allow real-time interactions, we fine-tune the decoder with a pretrained magnitude spectrogram inversion network and embed the full waveform generation pipeline in a *plugin*. Moreover the encoder could be used to process new input samples, after manipulating their latent attribute representation, the decoder can generate sample variations as an *audio effect* would. Our solution remains rather light-weight and fast to train, it can directly be applied to other sound domains, including an *user's libraries* with *custom sound tags* that could be mapped to specific generative controls. As a result, it fosters creativity and intuitive audio style experimentations. Sound examples and additional visualizations are available on Github[1], as well as codes after the review process.

[1] https://github.com/acids-ircam/Expressive_WAE_FADER

1. INTRODUCTION

Modern music production techniques rely on large and heterogeneous sound sample libraries along with diverse digital instruments and effects. It opens to a great variety of sound design possibilities and limitless contents to compose with, however principled interactions and scaled visualisations are still lacking in order to efficiently explore such potential and use it to generate *target sound qualities*.

Unsupervised generative models learn an underlying data distribution solely based on the observation of examples, in order to consistently generate novel content. They have been successfully applied to complex computer vision tasks such as processing facial expressions, landscapes, visual styles and paintings. Some solutions to audio emerged more recently, including pioneer musical systems such as *NSynth* (Neural Synthesizer [1]) for real-time high-quality sound synthesis. However, the heavy model architecture and prohibitive training time restrict its dissemination. The learned internal representation remains mostly uninformative and its many generative parameters are still too little correlated to explicit semantic qualities.

In this paper, we develop a high-level sound synthesis system with meaningful data visualisations and explicit musical controls. It is a lighter *non-autoregressive model* that can be trained fast on small datasets, including an user's personal libraries. Our goal is to learn expressive style variables from any sound tags, so that the model fosters creativity and *assists digital interactions* in music production. Considering note samples of orchestral instruments, we could for instance synthesise novel timbres or *playing style hybrids*.

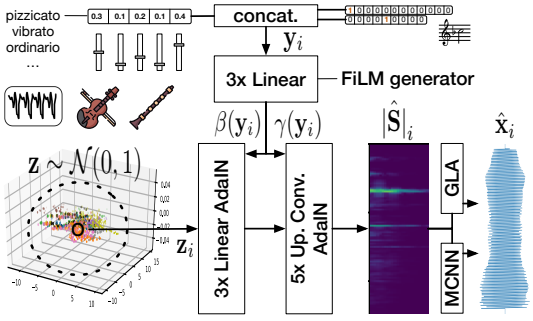

Figure 1: High-level note sample generation from the latent representation and musical conditioning in the decoder with FiLM. Intermediate features are modulated by the note targets and expressive style controls in order to synthesize new timbres and effects.

We train *Wasserstein Auto-Encoders* (WAEs [2][3]) on Mel- spectrogram magnitudes to organise a generative latent representation of individual note samples spanning the *tessitura* of 12 orchestral instruments. The considered database has intrinsic attributes: note classes, playing styles and timbres (each instrument subset), that we wish to control when generating new notes from the latent space. Thus we extend the WAE model with musical conditioning in the decoder and *Adaptive Instance Normalization* (AdaIN [4]). Using *Feature Wise Linear Modulation* (FiLM [5]) and adversarial training with a *Fader latent discriminator* [6], our WAE-Fader model effectively learns these generative controls along with expressive style variables that can be mixed continuously.

We evaluate these features in terms of generative performances and representation. We perform an *ablation study* and show that the model can sustain a good test reconstruction quality while achieving an accurate attribute-conditional generation. The success of the method relies on an attribute-free latent representation so that the decoder is pushed to learn the conditioning. These distributions can be visualized directly in the 3-dimensional latent space where clusters denote an undesired attribute encoding. We measure it with inter-class statistics and latent post-classification. The experiment validates correlations between low attribute encoding and effective conditioning.

We obtain an expressive note sample generator with *3-dimensional* representations of the training sound domains, decoding *probabilistic latent samples* with explicit control over the rendered note qualities. The learned style variables of the orchestra can ultimately be mixed continuously, as *faders* do, in order to intuitively explore new musical effects. Generated spectrogram magnitudes can approximately be inverted to waveform with the *Griffin-Lim* iterative algorithm (GLA [7]). Ultimately we fine-tune the decoder with a pretrained inversion network [8] for *real-time waveform synthesis*. We embed the resulting generative system in a *plugin* allowing for *MIDI* mapping, live exploration and *Digital Audio Workstation* (DAW) integration.

2. STATE-OF-ART

2.1. Generative models and regularized auto-encoders

Generative models aim to find the underlying probability distribution of the data $p(\mathbf{x})$ based on a set of examples in $\mathbf{x} \in \mathbb{R}^{d_x}$. To do so, we consider *latent variables* defined in a lower-dimensional space $\mathbf{z} \in \mathbb{R}^{d_z}$ ($d_z \ll d_x$), a higher-level representation that could have led to generate any given example. The latent variable generative model is defined by the joint probability distribution $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$, where the *prior* $p(\mathbf{z})$ is usually modelled with simpler distributions such as Gaussian or uniform while a complex conditional distribution $p(\mathbf{x}|\mathbf{z})$ maps latent codes to the data space. The model could be evaluated with the maximum marginal likelihood over the considered dataset. However for complex distributions that could model real-world data, integration cannot be computed in closed form.

Regularized auto-encoders have been used to reformulate the problem as an optimization by jointly learning the generative mapping $p_\theta(\mathbf{x}|\mathbf{z}) \in \mathcal{G}$ and an encoding distribution $q_\phi(\mathbf{z}|\mathbf{x}) \in \mathcal{Q}$ from families $\mathcal{G}$, $\mathcal{Q}$ of *approximate densities* both parameterized with neural networks. This was initially proposed through *Variational Inference* in the *Variational Auto-Encoder* (VAE [9]) that maximizes a lower bound of the data log-likelihood:

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x}|\mathbf{z})\big] - D_{KL}\big[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z})\big] \leq \log p_\theta(\mathbf{x}) \quad (1)$$

This amounts to optimizing the *Evidence Lower Bound Objective* (ELBO) that can be interpreted as follow, the first term is the Negative Log-Likelihood (NLL) data reconstruction cost and the second is the Kullback-Leibler Divergence (KLD) that quantifies the error made by using the approximate $q_\phi(\mathbf{z}|\mathbf{x})$ rather than the true $p_\theta(\mathbf{z})$. This latent regularization pushes the encoder to remain close to the prior latent density and can be weighted with a β parameter that balances these two objectives.

$$\mathcal{L}^{\text{ELBO}}_{\theta,\phi} = -\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x}|\mathbf{z})\big] + \beta \cdot D_{KL}\big[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z})\big] \quad (2)$$

The VAE is implemented with a *stochastic encoder* that parameterises an isotropic Gaussian latent distribution $q_\phi(\mathbf{z}|\mathbf{x})$ $\sim \mathcal{N}(\mu_\phi(\mathbf{x}), \sigma_\phi(\mathbf{x}))$ regularized against an unit variance prior. These assumptions allow analytical KLD computation and differentiable latent sampling for direct optimization of the ELBO.

The KLD forces each *individual* latent code to resemble the prior, which implicitly matches the whole encoded distribution. However a fitted ELBO value does not always result in an *effective inference*. Since the latent codes of different inputs are individually regularized, the KLD may prevent the encoder from learning any useful features (*posterior collapse* [10]) while the decoder only produces $p_\theta(\mathbf{x})$ regardless of the encoded information. Other conflicting solutions of the ELBO lead to undesired solutions and known limitations of VAEs such as *blurriness* of generated samples or *uninformative latent dimensions* ([11]).

With justifications stemming both from *Likelihood-free Optimization* (InfoVAE [3]) and the theory of *Optimal Transport* (WAE [2]), a more general framework for training regularized auto-encoders was recently proposed and that we call Wasserstein Auto-Encoders (WAEs). Considering a *deterministic decoder* $G_\theta : \mathbf{z} \to \mathbf{x}$ and any family of conditional encoder distribution $Q_\phi(\mathbf{z}|\mathbf{x}) \in \mathcal{Q}$, it is sufficient that the marginal $Q_Z(\mathbf{z}) := \mathbb{E}_X\big[Q(\mathbf{z}|\mathbf{x})\big]$ matches any prior P_Z. In comparisons with VAEs, WAEs can optimize any non-negative cost function C and *any divergence measure* D_Z between latent distributions, without requiring a stochastic encoder nor restricting the latent model to Gaussian prior:

$$\mathcal{L}_{\text{WAE}} := \inf_{Q(\mathbf{z}|\mathbf{x})\in\mathcal{Q}} \mathbb{E}_X \mathbb{E}_{Q(\mathbf{z}|\mathbf{x})}\big[C(\mathbf{x}, G(\mathbf{z}))\big] + \beta \cdot D_Z(Q_Z(\mathbf{z}), P_Z) \quad (3)$$

Thus we set our experiment in the more flexible WAE framework. These regularized auto-encoders are powerful unsupervised representation learning models, rather light-weight and fast to train, performing both inference (encoder) and generation (decoder). They are effective on small datasets (hundreds of training examples), learning a structured latent representation with disentangling capacities encouraged when $\beta > 1$. Once trained, probabilistic samples of the latent prior are consistently decoded into new samples and latent interpolations map to smooth data variations.

2.2. Maximum Mean Discrepancy Regularization

As shown for VAEs, the choice of *latent divergence* heavily impacts the resulting model performances. Since the point-wise KLD has strong intrinsic limitations, a more flexible regularization is required for WAEs. Such *differentiable* divergence on latent distributions was developed in the *Reproducing Kernel Hilbert Space* (RKHS) as a distance between probabilistic moments $\mu_{p,q}$ computed with a non-parametric kernel k:

$$\begin{aligned} ||\mu_p - \mu_q||^2_H &= \langle \mu_p - \mu_q, \mu_p - \mu_q \rangle \\ &= \mathbb{E}_{p,p}k(x, x') + \mathbb{E}_{q,q}k(y, y') - 2\mathbb{E}_{p,q}k(x, y) \end{aligned} \quad (4)$$

It defines the *Maximum Mean Discrepancy* (MMD [12]) between two distributions $x \sim p(x)$ and $y \sim q(y)$, where $\mathbb{E}_{p,q}$ is the expectation that can be evaluated with the *Radial Basic Function* (RBF) kernel of free parameter Σ:

$$k_{\text{(RBF)}}(x, y) = \exp\left(\frac{||x - y||^2}{-2\Sigma^2}\right) \tag{5}$$

To the extent of latent regularization, MMD can be computed between every deterministic mini-batch encoding $\mathbf{z}_{\text{encoder}} = Q(\mathbf{x})$ and random samples from any latent prior $\mathbf{z}_{\text{prior}} \sim P_Z$. Throughout the model optimization, MMD is thus matching the aggregated encoder posterior to the prior rather than regularizing each latent point individually. In comparisons with KLD, WAE-MMD allows for less constrained inference and richer latent representations. For instance, increasing β to two orders of magnitude above the reconstruction cost does not impede the decoder training. Since the WAE objective does not optimize the bounded NLL, the overall generative performances can be improved.
Other kernel functions can be used, which may be more discriminating at the expense of heavier computations. Alternatively to MMD, the WAE-GAN uses an adversarial latent discriminator to assess the divergence, thus optimizing a parametric function that could match even closer the encoder to the prior. However, since we consider a low-dimensional latent space of only 3 dimensions and remain with a simple isotropic Gaussian prior, MMD-RBF is sufficient yet light and stable to train on.

2.3. Conditioning and feature normalizations

Regularization in auto-encoders encourages disentanglement of independent generative factors onto separate latent dimensions that would in turn control the corresponding decoded data variations. However this is only partly achieved on toy datasets (β-VAE [13]) and in most cases the unsupervised latent dimensions are hardly related to explicit generative parameters. An additional supervision signal may be applied to the generative neural network in order to control and render specific attributes of the data. Thus we consider observations $\mathbf{x}$ paired with attribute annotations $\mathbf{y}$, and condition the decoder as $G : \{\mathbf{z}, \mathbf{y}\} \rightarrow \mathbf{x}$.
The simplest conditioning for categorical attributes is to encode them into one-hot vectors that are concatenated to the latent codes before being processed by the decoder. However more advanced conditioning techniques have been developed as for visual style transfer, using full images as conditions (conditional style transfer [14]). In the *Feature-wise Linear Modulation* (FiLM [5]) approach, a separate generator learns a mapping from any style inputs to adaptive biases $\beta_{\text{FiLM}}(\mathbf{y})$ and scales $\gamma_{\text{FiLM}}(\mathbf{y})$ applied to the conditional network computations. This modulation may be placed anywhere within the architecture and proved to be particularly suited to *Adaptive Instance Normalization* (AdaIN [4]). Considering the *l-th* hidden layer output activations $\mathbf{h}^l = g^l(\mathbf{h}^{l-1})$ of a generative neural network, the conditional modulation is thus be computed as:

$$\text{AdaIN}(\mathbf{h}^l, \mathbf{y}) = \gamma^l_{\text{FiLM}}(\mathbf{y})\left[\frac{\mathbf{h}^l - \mu(\mathbf{h}^l)}{\sigma(\mathbf{h}^l)}\right] + \beta^l_{\text{FiLM}}(\mathbf{y}) \tag{6}$$

in which mean and standard deviation $\{\mu,\sigma\}$ are computed across features, independently for each channel and each sample. In the context of style transfer, it can be interpreted as aligning the mean and variance of the content features with those of the style condition. It is a versatile conditioning technique, requiring little additional computations (particularly when applied channel-wise in convolution layers). It also suits well to handling multiple conditions that may more efficiently be mapped throughout the network rather than arbitrarily concatenated to the input. Thus we will use FiLM and AdaIN for conditioning the decoder on both note and style classes. However, such normalization is not suited to classification tasks since content features are individually normalized. In order to preserve its inference power, we will use *Batch Normalization* (BN) on the encoder's hidden activations.

2.4. Adversarial latent training

Adversarial regularization was proposed as an alternative to MMD in the WAE-GAN. For simple low-dimensional latent distributions, the expense of an additional parametric adversarial regularizer is not required. Nonetheless, adversarial latent training remains relevant for expressive conditioning. As detailed in the previous section, adaptive conditioning techniques paired with specific feature normalizations substantially improved feed-forward style transfer. However, in an auto-encoder setting, if the latent space implicitly encodes the attributes of interest, the decoder bypasses the conditioning and does not learn any effective generative controls. This problem was tackled in image generation with the introduction of an adversarial *Fader* latent discriminator $\mathcal{F}$ (Fader Networks [6]) that competes with the non-conditional encoder in order to prevent correlations between attributes and latent distributions. As for the conditional models, we consider annotated data samples $\{\mathbf{x}, \mathbf{y}\}$ and for simplicity, a categorical one-hot representation $\mathbf{y} \in \{0, 1\}^n$ with a single $y_i = 1$ and its opposite $\bar{\mathbf{y}} := \mathbf{1}^n - \mathbf{y}$. Such attribute-free latent representation is implemented in two separate optimization steps, first latent classification of the true attribute $\mathcal{F} : \mathbf{z} \rightarrow \hat{\mathbf{y}} \sim p_\psi(\mathbf{y}|Q(\mathbf{x}))$, then adversarial confusion of the latent classifier at predicting the opposite:

$$\begin{aligned} \mathcal{L}^{\text{class.}}(\psi|\phi) &= -\sum_{\mathbf{x},\mathbf{y}} \log(p_\psi(\mathbf{y}|Q(\mathbf{x}))) \\ \mathcal{L}^{\text{adv.}}(\phi|\psi) &= -\sum_{\mathbf{x},\mathbf{y}} \log(p_\psi(\bar{\mathbf{y}}|Q(\mathbf{x}))) \end{aligned} \tag{7}$$

As the encoder is pushed to remain invariant to attributes, the decoder is forced to learn the conditioning in order to reconstruct every input samples along with their source attributes. Thus it replaces adversarial training in the high-dimensional pixel space with latent attribute confusion in the low-dimensional latent space in order to efficiently learn style transfer variables. Applied to facial expressions, these *Fader* variables can continuously modulate complex visual features such as gender (female $\leftrightarrow$ male) or age (younger $\leftrightarrow$ older). Moreover, in mixing several attributes, one could generate new style qualities.

2.5. Audio synthesis

Neural networks can be trained on spectrogram magnitudes (and other spectral features) for audio analysis purpose. It eases the subsequent modelling task, often involving pattern detection, from a pre-processed structured sound representation. However, for generative purpose, an inversion from magnitude to waveform is required since the complex phase information was discarded. It is

commonly done offline with GLA [7]. Further advances in generative neural networks for audio have targeted raw waveform modelling through specific architecture design. *Wavenet* [15][1] is amongst them the most popular solution. It uses several stacks of dilated causal convolutions in order to aggregate multiple temporal granularities and structure long-term dependencies, which is challenging at the high audio sample rate. The output is a single auto-regressive sample prediction given all the previous sample context $p(x_t|x_1..x_{t-1})$. It results in high-quality real-time audio synthesis. However this sample level modelling requires long training times, heavy architectures that offer little knowledge over their learned features.

The *Multi-head Convolutional Neural Network* (MCNN [8]), a recent alternative for audio waveform modelling, was designed as a feed-forward real-time magnitude spectrogram inversion system that is not restricted to linear frequency scale. It proved to outperform GLA quality for speech. The use of differentiable GPU-based STFT computations enables a faster optimization onto spectral losses, rather than *auto-regressive* sample predictions:

$$\begin{aligned} \mathbf{x} \xrightarrow{|\text{STFT}|} |\mathbf{S}| \xrightarrow{\text{MCNN}} \hat{\mathbf{x}} \xrightarrow{\text{STFT}} \hat{\mathbf{S}} \Longrightarrow \mathcal{L}^{\text{MCNN}}(\mathbf{S}, \hat{\mathbf{S}}) \\ \mathcal{L}^{\text{MCNN}} = \lambda_0 \cdot \mathcal{L}^{\text{SC}} + \lambda_1 \cdot \mathcal{L}^{\text{logSC}} + \lambda_2 \cdot \mathcal{L}^{\text{IF}} + \lambda_3 \cdot \mathcal{L}^{\text{WP}} \end{aligned} \tag{8}$$

where $|\mathbf{S}|$ can be any spectrogram magnitude (including Mel-scaled frequencies). The model is well tailored to audio with multiple heads of 1-dimensional temporal up-sample convolutions. These heads focus on different spectral components and sum into waveform. It remains light-weight and could be adapted in an end-to-end waveform auto-encoder. Four objectives were originally proposed, using the complex STFT for the *Instantaneous Frequency* (IF) and *Weighted Phase* (WP) losses, that we could not optimize successfully. Hence we will only use the *Spectral Convergence* (SC) and log-scale magnitude (logSC) losses:

$$\mathcal{L}^{\text{SC}}(\mathbf{S}, \hat{\mathbf{S}}) = \||\mathbf{S}| - |\hat{\mathbf{S}}|\|_F / \||\mathbf{S}|\|_F \text{ with } \|.\|_F \text{ the Frobenius norm}$$

$$\mathcal{L}^{\text{logSC}}(\mathbf{S}, \hat{\mathbf{S}}) = \|\log(|\mathbf{S}| + \epsilon) - \log(|\hat{\mathbf{S}}| + \epsilon)\|_1 \tag{9}$$

3. METHOD

Our experiment begins with the WAE-MMD, isotropic unit variance Gaussian prior $\mathbf{z}_{\text{prior}} \sim \mathcal{N}(0, 1)$, RBF kernel and BN in both encoder and decoder in order to structure a 3-dimensional generative latent sound representation. Given a magnitude spectrogram $|\mathbf{S}|$ and a corresponding set of annotated attributes $\mathbf{y}$, we are learning $Q : |\mathbf{S}| \rightarrow \mathbf{z}$ and $G : \mathbf{z} \rightarrow |\hat{\mathbf{S}}|$ such as $|\mathbf{S}| \approx |\hat{\mathbf{S}}| = G(Q(|\mathbf{S}|))$ with *Binary Cross-Entropy* (BCE) reconstruction cost:

$$\begin{aligned} \mathcal{L}_{\text{WAE}} = \text{BCE}(|\mathbf{S}|, |\hat{\mathbf{S}}|) + \beta \cdot \text{MMD}_{\text{RBF}}(\mathbf{z}, \mathbf{z}_{\text{prior}}) \\ \text{BCE}(x, \hat{x}) = -\big[x \log \hat{x} + (1 - x) \log(1 - \hat{x})\big] \; ; |x| < 1 \end{aligned} \tag{10}$$

We can sample random codes from the latent prior and consistently decode new magnitude samples, however there is no control on the output features. For the orchestra, we consider $\mathbf{y} = \{\mathbf{y}_{\text{note}}, \mathbf{y}_{\text{style}}\}$ with $\mathbf{y}_{\text{note}} = \{\text{semitone, octave}\}$. We define the timbre attribute as the class of an instrument subset, which comprises the *Ordinario* mode as well as diverse extended playing techniques such as *Staccato*, *Flatterzunge* or *Pizzicato*. When considering a single subset, we thus aim at controlling the playing techniques of the considered instrument as $\mathbf{y}_{\text{style}}$. When considering multiple instruments,

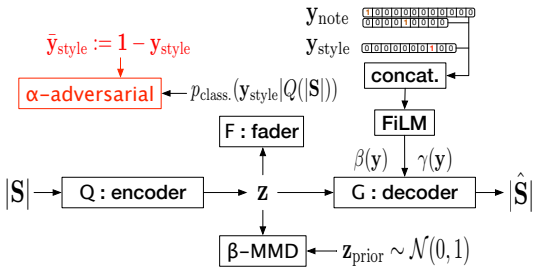

Figure 2: How information flows in the adversarial optimization of the WAE-Fader

instead we aim at controlling the different timbres, either in *Ordinario* or with mixed playing styles within each instrument subset. For explicit controls over the rendered attributes, we condition the decoder as $G : \{\mathbf{z}, \mathbf{y}\} \rightarrow |\hat{\mathbf{S}}|$ using AdaIN. An additional FiLM generator is fed with concatenated one-hot vectors of the three attribute classes (semitone, octave and style). It learns an adaptive mapping to biases $\beta_{\text{FiLM}}(\mathbf{y})$ and scales $\gamma_{\text{FiLM}}(\mathbf{y})$ that are used to modulate the normalized decoder activations. In order to effectively learn the style conditioning and expressively modulate timbres or playing techniques, we use adversarial training with a Fader latent discriminator $\mathcal{F} : \mathbf{z} \rightarrow \hat{\mathbf{y}}_{\text{style}} \sim p_{\text{class.}}(\mathbf{y}_{\text{style}}|Q(|\mathbf{S}|))$ that competes with the non-conditional encoder in classifying the considered styles from latent codes:

$$\begin{aligned} \mathcal{L}_{\text{class.}} &= -\log p_{\text{class.}}(\mathbf{y}_{\text{style}}|Q(|\mathbf{S}|)) \\ \mathcal{L}_{\text{WAE-Fader}} &= \mathcal{L}_{\text{WAE}} - \alpha \cdot \log p_{\text{class.}}(\bar{\mathbf{y}}_{\text{style}}|Q(|\mathbf{S}|)) \end{aligned} \tag{11}$$

with $\bar{\mathbf{y}}_{\text{style}} := \mathbf{1} - \mathbf{y}_{\text{style}}$ and α that weights the adversarial loss in the encoder. Classification is optimized on the NLL with *Softmax* probabilities. The resulting attribute confusion prevents the latent space from implicitly encoding the style distributions, thus the decoder is forced to use the conditioning to reconstruct the source features from the attribute-free code. Ultimately these learned style variables can continuously be mixed, as actual *faders* do. We refer to this final model as WAE-Fader, that still uses MMD regularization. It allows for controlling the strength of each rendered attribute and intuitively exploring hybrid sound effects from any custom tags, here either chosen from extended playing techniques or from diverse orchestral timbre domains.

The resulting generative system maps any latent coordinate $\mathbf{z} \sim \mathcal{N}(0, 1) \in \mathbb{R}^3$ to target note spectrograms with expressive musical style controls. Inversion from spectrogram magnitudes to audio waveforms can be done offline with GLA. Alternatively, we pre-train a MCNN on a larger corpus of musical note samples to allow real-time rendering. In order to improve the final audio quality, we fine-tune the full generative model by freezing the encoder parameters and jointly optimizing the learned decoder with the pretrained MCNN as:

$$\mathbf{x} \xrightarrow{\text{STFT}} \mathbf{S} \xrightarrow{|\text{Mels}|} |\mathbf{S}| \xrightarrow{Q} \mathbf{z} \xrightarrow{G \circ \text{MCNN}} \hat{\mathbf{x}} \xrightarrow{\text{STFT}} \hat{\mathbf{S}} \Longrightarrow \mathcal{L}^{\text{MCNN}}(\mathbf{S}, \hat{\mathbf{S}}) \tag{12}$$

This waveform pipeline $\{G \circ \text{MCNN}\}$ is embed in a plugin for live interactions and DAW integration. Using a MIDI interface, we can for instance trigger target note classes $\mathbf{y}_{\text{note}}$ with keys and map the continuous generative parameters to faders. These are the latent dimensions $\mathbf{z}$, that can also be randomly sampled, and most interestingly the adversarially learned style variables $\mathbf{y}_{\text{style}}$ that can be mixed to explore new sound effects.

4. EXPERIMENT

4.1. Dataset

We use the Studio-On-Line (SOL [16]) library of around 15000 individual note samples, across the tessitura of 12 orchestral instruments grouped in 4 families and with many extended playing techniques, that may be specific or shared across instrument families. These are *Wind* (Alto-Saxophone, Bassoon, Clarinet, Flute, Oboe), *Brass* (English-Horn, French-Horn, Tenor-Trombone, Trumpet), *String* (Cello, Violin) and *Keyboard* (Piano). Notes are consistently tagged with the intrinsic attributes of the dataset: note classes (12 semitones across 9 octaves), several dynamics and playing styles of every instrument. We define two style experiments for the orchestra. If training on a single instrument, we aim for expressive synthesis of its playing styles. If training on multiple instruments, we aim for timbre control. Each instrument subset defines a timbre domain, either in *Ordinario* (its common mode) or with all styles mixed.
Audio files are down-sampled to 22050Hz and pre-processed into Mel-spectrograms with a FFT size of 2048, hop size of 256 and 500 bins ranging the full spectrum. As we consider a generator of *individual* notes, we set a common audio length of 34560 samples (∼1.6s) from the attack which amounts to 128 STFT frames. We choose this duration as a trade-off between input and latent dimensionality, limiting the amount of silence after shorter playing modes (eg. *Pizzicato*) while keeping some sustain for longer notes (from which some sustain and decay may have been cropped). Magnitudes are floored to 1e-3 and log-scaled in [0,1] according to the BCE range. Each playing style subset of each instrument is split into 80% training, 10% validation and 10% test notes. In average each instrument has 10 playing styles and 100 to 200 notes for each.

4.2. Implementation details

Architecture of the WAE-Fader: Our experiments have been implemented in the *PyTorch* environment and our codes will be shared with this dependency. All convolution layers use 2-d. square kernels, an input zero-padding of half the kernel size and are followed by 2-dimensional feature normalization. All fully-connected linear layers are followed by 1-dimensional feature normalization. The non-linear activation used after every normalization is CELU. The deterministic encoder has 5 convolution layers with [12, 24, 48, 96, 128] output channels, kernel size 5 and stride 2, that down-sample the input spectrograms into 128 output maps that are flattened into an intermediate feature vector of size 8192. It is followed with a bottleneck of 3 linear layers of output sizes [1024, 512, 3] mapping to the latent space. For input Mel-spectrograms of size (500,128), it amounts to a dimensionality reduction of more that 5 orders of magnitude. All normalizations are BN. The decoder mirrors this structure with 3 linear layers of output sizes [512, 1024, 8192]. This vector is then reshaped into 128 maps. To avoid the known *checkerboard artifacts* [17] of the transposed convolution, we use *nearest neighbor* up-sampling followed with convolution of stride 1. These maps are processed with 4 up-sampling of ratios 3, the last one directly mapping to the input dimensionality of (500,128), and 5 convolutions with [96, 48, 24, 12, 1] output channels and kernel sizes [5, 5, 7, 9, 7]. All normalizations are AdaIN and the decoder output activation is sigmoid, bounded in [0,1] according to the BCE range. The FiLM conditioning is applied feature-wise at the output of the first two linear layers and channel-wise after. It amounts to 3688 modulation weights computed by an additional FiLM generator of 3 linear layers of output sizes [512, 1024, 3688]. Its output is split into biases and scales of sizes [512, 1024, 128, 96, 48, 24, 12]. The Fader latent discriminator has 3 linear layers of output sizes [1024, 1024, n_{style}] with *LeakyReLU* activations and a dropout ratio of 0.3, mapping latent codes to probabilities of the n_{style} classes.

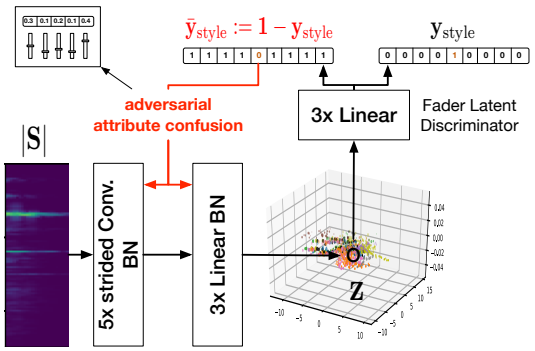

Figure 3: The Fader latent discriminator tries to infer the true style attribute while the encoder adversarially aims at fooling it. It encourages attribute invariance in the latent representation and learning of continuous generative controls in the decoder.

Training parameters: We train our models with the Adam optimizer, an initial learning rate of 5e-4 and a batch size of 90. All model weights are initialized with Xavier uniform distribution. Depending on the considered data subset size, between 200 and 800 epochs are needed. A single instrument (1000-1500 notes) can be modelled in less than 2 hours on one NVIDIA TITAN Xp GPU. Training over all instruments and styles at once (around 11000 notes) takes less than 12 hours. In the first part of the training (30 to 100 epochs), we only optimize the reconstruction and classification objectives. Then we gradually introduce the MMD regularization (β-warmup) and the adversarial feedback in the encoder (α-warmup) until the first half of training epochs. The rest of the training jointly optimizes all training objectives at their target strengths $\beta = 40$ and $\alpha = 4$. These value were estimated in order to approximately balance the gradient magnitudes back-propagated by each loss. However, for the baseline WAE-MMD models we warmup β to 500 since it does not prevent from optimizing the reconstruction cost.

Signal reconstruction: The above described model trains on inputs with 128 frames of Mel-spectrogram, which amount to 34560 waveform samples according to our STFT settings. The generated Mel-magnitudes can be approximated back to the linear frequency scale and iteratively inverted with GLA for 100 to 300 iterations. To allow real-time rendering and a possibly improved audio quality, we reproduce the original MCNN architecture for Mel-spectrogram magnitudes inversion. We use 8 heads, $\lambda_0 = 1$ and $\lambda_1 = 6$. We could not successfully optimize the complex losses, however, we compute these magnitude losses on both the linear and Mel frequency scales. We pretrain this model on a larger dataset of around 50 hours audio comprising SOL and subsets of the *Vienna Symphonic Library* (VSL). Ultimately, we fine-tune the trained decoder with this pretrained MCNN. To do so, we freeze

the encoder weights and optimize $G \circ \text{MCNN}$ on the model train set. In equation (12), the auto-encoder pair G, Q maps to Mel-spectrogram magnitudes $|\mathbf{S}|$ which are inverted to signals by the MCNN. However, the loss computation $\mathcal{L}^{\text{MCNN}}(\mathbf{S}, \hat{\mathbf{S}})$ is not necessarily restricted to this frequency scale. Thus we evaluate and sum $\mathcal{L}^{\text{SC}}$ $\mathcal{L}^{\text{logSC}}$ from equation (9) on both linear and Mel frequency scaled magnitudes.

classified attribute (n_{style})	train set	validation	test
Semitone (12)	1.00	0.99	0.99
Octave (9)	1.00	0.99	1.00
Ordinario timbres (12)	1.00	1.00	1.00
Extended timbres (12)	1.00	1.00	1.00
Violin playing styles (10)	1.00	0.97	0.95
Clarinet playing styles (10)	1.00	0.96	0.94
Piano playing styles (10)	1.00	0.92	0.95
Trumpet playing styles (10)	1.00	0.92	0.94
Alto-Saxophone pl. styles (10)	1.00	0.98	1.00
Tenor-Trombone pl. styles (11)	1.00	0.90	0.90

Table 1: Reference **F1-scores** of the pretrained data classifiers used for the evaluation of conditional note generations

4.3. Evaluations

Generative performances: First, we evaluate the ability of our models to produce accurate spectrograms by computing the reconstruction scores on the test set with *Root-Mean Squared Error* (RMSE) and *Log-Spectral Distance* $\text{LSD} = \sqrt{\sum \left[10 \log_{10}(|\mathbf{S}|/|\hat{\mathbf{S}}|)\right]^2}$. Regarding the conditioning aspects, we first pretrain data classifiers to reliably discriminate the different attribute classes and report their performances in Table 1. These classifiers share the same architecture as the encoder but map to the n_{style} classes of interest. We use them as references to evaluate the effectiveness of the conditioning. Then, we sample an evaluation batch of 1000 random latent points from the prior, along with random semitone and octave targets. This evaluation batch is decoded to each attribute of the model (either playing styles or timbres) and classified with the corresponding reference classifier. A high accuracy means an effective conditioning for the task of musical note generation. We report the average accuracy for all the target conditions, with random octaves both in [0-8] (full orchestral range) or in [3-4] where models train on the overlap of every instrument tessitura.

Latent space structure: The effectiveness of the conditioning relies on learning an attribute-free latent representation of the data. If the attribute distributions are clustered, the decoder may learn their correlations with latent dimensions and bypass the conditioning signal. This phenomenon is alleviated with adversarial training of the non-conditional encoder against a Fader latent discriminator. As we map to 3-dimensional spaces, we can directly visualize this latent organization. We also propose two evaluations of the attribute representations. First, we compute the average inter-class latent statistics with MMD. In this case, low values mean that the attribute distributions blend in the final representation. Second, we also perform a post-classification task by training classifiers at predicting the attributes from the learned latent representation. These models use the same architecture as the Fader discriminator, and we report their final accuracy. In this case, low scores mean that the latent representation did not encode the attributes.

model	test rec.		note cond. acc.				style cond. acc.	
	MSE	LSD	$st._{34}$	$oct._{34}$	$st._{08}$	$oct._{08}$	$style_{34}$	$style_{08}$
Violin playing styles (n_{style}=10) 1475 training note samples								
WAE-MMD	0.76	68.2	NA	NA	NA	NA	NA	NA
WAE-note	0.69	55.4	0.73	0.72	0.47	0.43	NA	NA
WAE-style	0.74	59.6	0.47	0.39	0.30	0.22	0.20	0.17
WAE-Fader	0.80	91.1	**0.96**	**0.77**	0.97	0.48	**0.88**	**0.93**
Ordinario timbres (n_{style}=12) 1784 training note samples								
WAE-MMD	1.04	88.3	NA	NA	NA	NA	NA	NA
WAE-note	0.84	71.6	0.99	0.96	0.62	0.53	NA	NA
WAE-style	0.80	65.7	0.64	0.58	0.30	0.24	0.33	0.19
WAE-Fader	1.01	105	**1.00**	**1.00**	0.94	0.68	**0.95**	**0.70**
Extended timbres (n_{style}=12) >11000 training note samples								
WAE-MMD	0.93	175	NA	NA	NA	NA	NA	NA
WAE-note	0.69	173	0.99	0.98	0.72	0.64	NA	NA
WAE-style	0.65	172	0.84	0.83	0.44	0.39	0.61	0.34
WAE-Fader	1.32	182	**1.00**	**1.00**	0.90	0.71	**0.95**	**0.64**

Table 2: The ablation study confirms the effectiveness of the **WAE-Fader** conditioning, both on target notes and playing styles or timbres. The conditional latent sampling is either performed with random octaves in [3,4] (the overlap of every tessitura) and [0-8] (the full orchestra range), we report the accuracy of the conditioning with respect to the targets $note_{34,08}$ (st. is semitone classe and oct. is octave classe) and $style_{34,08}$.

5. RESULTS

5.1. Ablation study

We defined both generative and representation evaluations to assess the effectiveness of our proposed musical conditioning. To study the benefits and compromises of each model feature, we train the base WAE-MMD and compare it with ablations of the WAE-Fader. The incremental model comparisons are WAE-MMD (no conditioning), WAE-note (semitone and octave conditioning), WAE-style (note and style conditioning) and WAE-Fader. In order to simplify the notation, we do not specify the MMD but this regularization is used for all models. We performed this ablation study on the violin subset that has the following annotated playing styles: *Ordinario, Sustained, Short, Non-vibrato, Staccato, Pizzicato-secco, Medium-vibrato-short, Tremolo, Medium-vibrato-sustained* and *Pizzicato-l-vib*. We also compare the WAE-Fader on instrument timbres, either in ordinario or for all extended techniques mixed per instrument subset. Table 2 reports the successive generative performances of the models. Table 3 reports the latent evaluations, showing how the conditioning tasks are reflected in the learned representations. It confirms the effectiveness of the expressive conditioning when the attribute-invariance assumption is achieved.

As we can see, conditioning WAE-note on the semitone and octave classes shows that the WAE-MMD model can partly learn the note controls with FiLM conditioning. Accordingly, the latent space structure does not exhibit strong correlations with the note classes anymore but with the style attributes that become the main unsupervised data feature. We also notice that this additional supervision improves the reconstruction quality. However, when adding the style conditioning in WAE-style, it seems that most performances drop. Indeed, the overall conditioning becomes little effective, both for the target note and style conditions. The final

model	inter-class MMD			post-class. acc.		
	st.	oct.	style	st.	oct.	style
Violin playing styles (n_{style}=10) 1475 training note samples						
WAE-MMD	0.25	0.26	0.30	0.92	0.94	0.56
WAE-note	0.03	0.10	0.50	0.04	0.49	0.82
WAE-style	0.12	0.16	0.35	0.25	0.55	0.59
WAE-Fader	0.02	0.01	0.46	0.08	0.34	0.64
Ordinario timbres (n_{style}=12) 1784 training note samples						
WAE-MMD	0.12	0.38	0.28	0.75	0.87	0.59
WAE-note	0.02	0.28	0.51	0.33	0.57	0.83
WAE-style	0.04	0.40	0.35	0.13	0.60	0.63
WAE-Fader	0.33	0.08	0.03	0.17	0.25	0.23
Extended timbres (n_{style}=12) >11000 training note samples						
WAE-MMD	0.02	0.41	0.12	0.71	0.89	0.48
WAE-note	5e-3	0.30	0.22	0.07	0.49	0.71
WAE-style	4e-3	0.32	0.18	0.07	0.56	0.55
WAE-Fader	3e-3	0.20	0.11	0.11	0.46	0.43

Table 3: The ablation study allows to monitor the latent organization in the different models and throughout their training, as shown in Figure 4. We use both inter-class statistics and latent post-classification to estimate the final attribute invariance in the learned representation.

results show that the adversarial latent training enables the WAE-Fader model to effectively learn the complete conditioning, at the expense of a possible drop in its reconstruction accuracy.
It also seems that the task of modelling the playing styles when learning on a single instrument is more challenging than changing the timbres across multiple instruments. This can be seen in the lower performance of the WAE-style model applied to the violin. This may also be explained by the reduced size of the training data when the learning is restricted to single instrument subsets. These observations are supported by the resulting audio outputs of the conditional note generations. Indeed, it appears that meaningful and expressive variations when switching to any attribute conditions are only achieved with our proposed WAE-Fader model. This is successful for conditioning applied on both timbre attributes or playing styles.

5.2. Expressive note sample generations

In this section, we report additional experiments on the WAE-Fader models when conditioned on the playing styles of different instruments and families. As shown in Table 4, our model seems to train successfully on playing styles in every instrument families, as well as across the 12 instrument timbres of the orchestra as shown in the previous ablation study. This amounts to a great variety of sound qualities spanning extended modes of the orchestra, and let us hypothesize that the model could be applied to other sound domains as long as the tags are consistent with the data. Furthermore, the style variables learned with the Fader latent discriminator are continuous independent controls that can be mixed. Hence, this can allow our system to modulate the strength of rendered styles and create new effects by combining multiple attributes. Our model can also be used for sample modifications, akin to traditional audio effects, by encoding a given sample and manipulating the attribute conditions in order to decode different sample transformations.

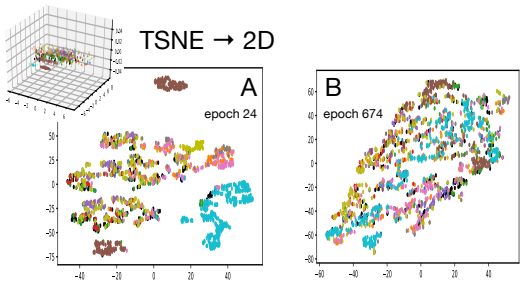

Figure 4: Latent organization as the WAE-Fader model trains on the ordinario timbres, each instrument domain being represented by a separate color. In ***A***, at epoch 24, the encoder does not optimize the adversarial loss yet. Its unsupervised representation exhibits the attribute classes. In ***B***, at epoch 674, the α-warmup is finished and the adversarial latent training had blended the attribute distributions. The 2-dimensional projections are performed with t-Distributed Stochastic Neighbor Embedding (TSNE).

5.3. Audio outputs and plugin development

As discussed previously, our proposed models can generate magnitude spectrograms, while controling their expressive qualities. These spectrograms can be either inverted to waveform offline with GLA or real-time if paired with MCNN. When fine-tuning the learned decoders with the pretrained MCNN on magnitude losses, we obtain a quality almost equivalent to the GLA approximation. We provide audio examples of test set reconstructions and conditional note generations inverted with both GLA and MCNN for individual listening evaluation on the companion webpage. While the audio quality of these results can still be improved, we can already confirm the ability of the model to provide semantic controls. As the learned style variables of WAE-Fader can be mixed continuously, we also provide some sound examples that were generated when modifying multiple orchestral attributes.
Our proposal provides intuitive sound synthesis of target sound qualities with learned style variables that can be modulated and combined. The unsupervised latent dimensions organize remaining data features, which can be directly visualized in a 3-d. space, in order to perform sampling or explicit control. These features allow to generate timbres, playing styles and hybrid effects across multiple attribute combinations through intuitive interactions. We provide a real-time implementation of our models by relying on the fine-tuned $\{G \circ \text{MCNN}\}$ generation. This implementation relies on the *LibTorch* C++ API, which converts trained *PyTorch* models, that we further embed in a *PureData* external. This plugin can be mapped to a MIDI controller or integrated in a DAW for composition and musical performance. This allows to play notes with a keyboard, while using continuous faders to control latent coordinates and mix style conditions.

6. CONCLUSION

We developed an expressive musical conditioning of the Wasserstein Auto-Encoders able to model a collection of orchestral note samples. The model learns effective target semitone and octave

model	test rec.		note cond. acc.				style cond. acc.		inter-class MMD			post-class. acc.		
	MSE	LSD	$st._{34}$	$oct._{34}$	$st._{08}$	$oct._{08}$	$style_{34}$	$style_{08}$	st.	oct.	style	st.	oct.	style
Clarinet	0.87	116	0.96	0.99	0.98	0.45	0.97	0.92	0.05	0.58	0.12	0.15	0.76	0.41
Piano	0.99	113	0.53	0.91	0.47	0.72	0.72	0.64	0.03	0.03	0.08	0.16	0.20	0.43
Trumpet	0.90	107	0.91	0.93	0.96	0.37	0.90	0.87	0.60	0.11	0.02	0.42	0.50	0.29
Alto-Sax.	1.22	131	0.96	0.99	0.98	0.40	0.76	0.71	0.14	0.09	0.50	0.08	0.48	0.48
T. Trombone	0.96	100	1.00	1.00	0.92	0.41	0.83	0.77	0.04	0.14	0.34	0.06	0.55	0.47

Table 4: Additional WAE-Fader results on the playing techniques of instruments in other orchestral families

controls as well as continuous style variables. We considered extended playing techniques and timbre subsets as attributes, and used adversarial latent training to encourage an attribute-invariant representation in the WAE-Fader. Our ablation study validates the effectiveness of style conditioning when this invariance condition is obtained.

We fine-tuned the decoders with a Mel magnitude spectrogram inversion network that allows real-time waveform rendering and are currently working on refining the audio quality. This results in a note sample generator with meaningful data visualizations and intuitive controls of audio styles. These parameters can be mixed, as faders, in order to explore hybrid sound effects. Our final generative model is embed in a plugin for MIDI mapping and live interactions. This system provides assisted music production and fosters creative sound experimentations. We provide sound examples from our orchestral models, either inverted offline with GLA or with the fine-tuned waveform generation pipeline. These sounds allow for subjective evaluation of both semantic and audio qualities of our solution.

Although we used clearly defined metadata attributes pertaining to instrumental playing styles, the model can potentially be applied to any sound domain. For instance, a user library with custom tags could be mapped to sound synthesis parameters. Furthermore, as the architecture is rather light and scales to small datasets, it could be trained on user libraries. Future experiments will target the quality of the waveform modelling systems for variable note lengths and real-time synthesis. Ultimately, our models could be implemented as a standalone instrument with physical controls that can be mapped to pretrained style variables. This would allow an intuitive and creative exploration across a vast amount of sound variations with a reduced set of adaptive parameters.

7. ACKNOWLEDGEMENTS

This work was supported by the MAKIMOno project 17-CE38-0015-01 funded by the French ANR and the Canadian NSERC (STPG 507004-17), the ACTOR Partnership funded by the Canadian SSHRC (895-2018-1023) and an NVIDIA GPU Grant. We acknowledge and thank the initial contributions of Jean-Baptiste Dakeyo and Geoffroy Thibault, as well as the further collaboration of Antoine Caillon and Martin Fouilleul.

8. REFERENCES

[1] J. Engel et al., "Neural audio synthesis of musical notes with wavenet autoencoders," *International Conference on Machine Learning (ICML)*, 2017.

[2] I. Tolstikhin et al., "Wasserstein auto-encoders," *International Conference on Learning Representations*, 2018.

[3] S. Zhao et al., "Infovae: Information maximizing variational autoencoders," *arXiv:1706.02262*, 2017.

[4] X. Huang et al., "Arbitrary style transfer in real-time with adaptive instance normalization," *International Conference on Computer Vision (ICCV)*, 2017.

[5] E. Perez et al., "Film: Visual reasoning with a general conditioning layer," *AAAI Conference*, 2018.

[6] G. Lample et al., "Fader networks: Manipulating images by sliding attributes," *Conference on Neural Information Processing Systems (NIPS)*, 2017.

[7] D. W. Griffin and J. S. Lim, "Signal estimation from modified short-time fourier transform," *IEEE Transactions on acoustics, speech, and signal processing*, 1984.

[8] G. Lample et al., "Fast spectrogram inversion using multi-head convolutional neural networks," *IEEE Signal Processing Letters, vol. 26, no. 1*, 2019.

[9] D. P. Kingma et al., "Auto-encoding variational bayes," *International Conference on Learning Representations*, 2014.

[10] S. Zhao et al., "Towards deeper understanding of variational autoencoding model," *arXiv:1702.08658*, 2017.

[11] S. Yeung et al., "Tackling over-pruning in variational autoencoders," *ICML workshop*, 2017.

[12] A. Gretton et al., "A kernel two-sample test," *Journal of Machine Learning Research*, vol. 13, 2012.

[13] I. Higgins et al., "beta-vae: Learning basic visual concepts with a constrained variational framework," *International Conference on Learning Representations (ICLR)*, 2017.

[14] G. Ghiasi et al., "Exploring the structure of a real-time, arbitrary neural artistic stylization network," *British Machine Vision Conference (BMVC)*, 2017.

[15] A. Van Den Oord et al., "Wavenet: A generative model for raw audio," *arXiv:1609.03499*, 2016.

[16] G. Ballet et al., "Studio online 3.0: An internet "killer application" for remote access to ircam sounds and processing tools," *Journee Informatique Musicale (JIM)*, 1999.

[17] A. Odena et al., "Deconvolution and checkerboard artifacts," *Distill*, 2016.