A hierarchical approach to techno loop generation

Nils Demerlé, Antoine Caillon

Abstract

Recent advances in neural audio synthesis have made possible the creation of exciting new tools for composition and sound design. Early work on unconditional audio generation using autoregressive networks [1] has yielded impressive results, at the cost of expensive training and long synthesis time. In this project, you will explore how a hierarchical audio modeling approach helps with learning long temporal dependencies while allowing fast sampling.

1 Introduction

Deep learning applied to audio signals proposes exciting new ways to perform speech generation, musical composition and sound design. Recent works in deep audio modelling have allowed novel types of interaction such as unconditional generation [1, 2, 3] or timbre transfer between instruments [4]. However, these approaches remain computationally intensive, as modeling audio raw waveforms requires dealing with extremely large temporal dimensionality. Previous work [5, 6] have shown how modeling complex datasets can be achieved through the use of a multiscale decomposition of the problem. For representation learning purposes, stacking VAEs has proven to yield more compact representations while improving reconstruction scores [7]. Therefore, the goal of this project is to build a hierarchical model targeting the representation learning of a techno loop dataset, allowing the sampling of new samples through the exploration of a smooth and high-level latent space. A diagram of the proposed approach is depicted in Figure 1.

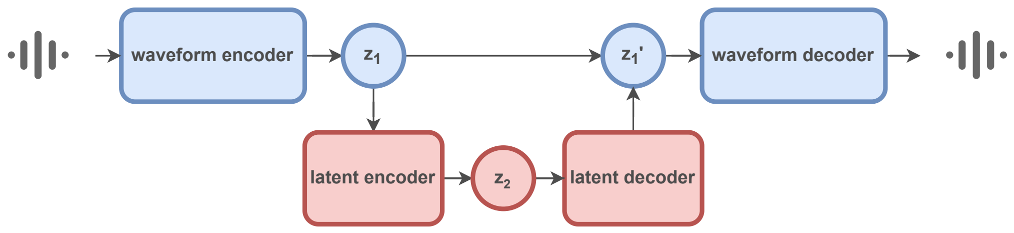

Figure 1: Proposed approach. The model is composed of two VAEs, progressively compressing an input audio signal into a high level and compact latent representation.

2 VAE MNIST

You should first get familiar with PyTorch and generative models such as the Variational Auto-Encoder [8]. You will implement and train your own VAEs on MNIST which is a rather simple collection of hand-written digits from 0 to 9. You can compare different architectures (e.g. linear layers, convolutions, batch-normalization etc.) as well as training on harder datasets (e.g. Fashion-MNIST). Once the model has achieved a reasonable reconstruction quality, you can generate new data by sampling and interpolation in its latent space.

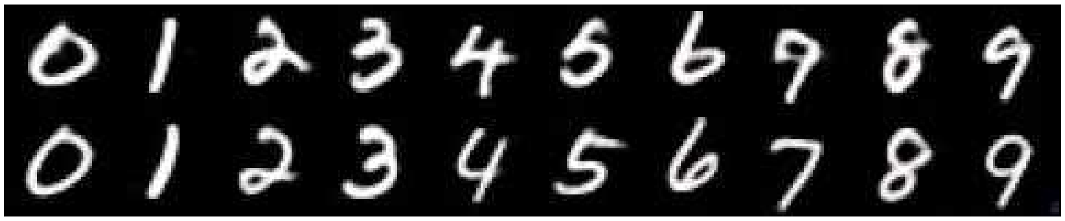
Figure 2: Examples from the MNIST dataset. Each digit is a 28×28 pixels grey scale image.

This part of the subject must be done *individually* using a python notebook.

3 Raw audio VAE

3.1 Dataset

We will provide you a large scale dataset of beat aligned techno loops, sampled at 16kHz, alongside the relevant dataloaders. If you are interested in building your own custom dataset, please do, but be aware that this is a time consuming task and that you will not be evaluated on this.

3.2 Reconstruction loss

Parallel audio waveform autoencoding is introduced in [9], where the authors leverage feed-forward convolutional networks to speed up the autoregressive generation process in [10]. They propose a perceptually-motivated distance between waveforms called *spectral distance* as the reconstruction objective

$$l(\mathbf{x}, \mathbf{y}) = \|\log(\text{STFT}(\mathbf{x}) + \epsilon) - \log(\text{STFT}(\mathbf{y}) + \epsilon)\|_1 , \tag{1}$$

where STFT is the amplitude of the Short-Term Fourier Transform. Using only the amplitude of the STFT makes the loss permissive to phase variations.

3.3 Model

Your waveform VAE will be a fully convolutional neural network. You can draw inspiration from the architectures proposed in [9, 11], but keep it simple. The focus of this project is mainly about interaction and representation learning, so do not worry if the model is not reaching a perfect audio quality.

3.4 Learned representation

You will study two regularization objectives for your latent space:

- Regular Variational Auto Encoder (Kullback Leibler)
- Wassertein Auto Encoder [12]

The temporal compression ratio of the model should be around 128x. You can play with a weighting factor for the regularization term to avoid *posterior collapse*.

3.5 Evaluation

Study the properties of your latent space. Is it nice? Is it juicy? Is it scrumptiously crunchable? Compute some correlation scores between dimensions of the latent space and acoustic descriptors. Try and sample from it too, so that you can hear if it can already be used as a generative system.

4 Latent VAE

The second VAE aims at learning latent trajectories yielded by the first one (see Figure 1). Using this second model, you should be able to get a fully-fledged generative system able to unconditionally generate new techno loops.

4.1 Architecture

The architecture of the second VAE can roughly follow the one you have already implemented. However, we are interested in studying the effect of the compression ratio on the smoothness of the generated latent trajectories. Therefore you should :

1. Build a simple temporal VAE producing a small high level latent trajectory
2. Make your code flexible enough to change the representation length easily (down to a single point).

4.2 Training objective

Since we are no more modeling audio signals, we have to choose a reconstruction objective that is suited to the task. We propose you compare two different methods

1. Sample z (or use the mean value) from VAE 1, train VAE 2 on those fixed trajectories with log-likelihood
2. Extract and concatenate VAE 1 posterior parameters (μ_1, σ_1), train VAE 2 on distributions, using with the Kullback Leibler divergence between $\mathcal{N}(\mu_1, \sigma_1)$ and $\mathcal{N}(\hat{\mu}_1, \hat{\sigma}_1)$

4.3 Evaluation

Randomly sample a *smooth* trajectory inside your high-dimensional latent space, and listen to your generative model gradually evolve. This is the perfect time to try and create something nice! You can also evaluate the latent space smoothness by interpolating between existing trajectories from the dataset. Plot the corresponding evolution of several acoustical descriptors such as loudness and centroid to demonstrate the *smoothness* of your latent space.

5 Optional

If you have the time, choose one of the following bonuses !

5.1 CAIUS: Adversarial decoder fine-tuning

Following [11], add a discriminator to the output of your first VAE to improve its synthesis quality. You can benchmark several discriminiator architectures (multiscale [13], multiperiod [14]) or gan objectives (classical, non saturating, hinge...).

5.2 BONUS: Latent diffusion

Recent advances in generative modeling using probabilistic diffusion models [15] have yielded impressive results in image and video generation. An interesting research direction would be to replace the second VAE by a diffusion model to correctly model the temporal behavior of latent trajectories.

WARNING: this is hard. Do not start working on this section except you have already done everything else *perfectly*.

References

[1] Aaron 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. 9 2016.

[2] Sean Vasquez and Mike Lewis. MelNet: A Generative Model for Audio in the Frequency Domain. *arXiv*, 6 2019.

[3] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A Generative Model for Music. *arXiv*, 4 2020.

[4] Noam Mor, Lior Wolf, Adam Polyak, and Yaniv Taigman. A Universal Music Translation Network. *arXiv*, 5 2018.

[5] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 10 2017.

[6] Yusong Wu, Ethan Manilow, Yi Deng, Rigel Swavely, Kyle Kastner, Tim Cooijmans, Aaron Courville, Cheng-Zhi Anna Huang, and Jesse Engel. MIDI-DDSP: Detailed Control of Musical Performance via Hierarchical Modeling. 12 2021.

[7] Bin Dai and David Wipf. Diagnosing and Enhancing VAE Models. 3 2019.

[8] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, 12 2014.

[9] Alexandre Défossez, Neil Zeghidour, Nicolas Usunier, Léon Bottou, and Francis Bach. SING: Symbol-to-Instrument Neural Generator. *Advances in Neural Information Processing Systems*, 2018-December:9041–9051, 10 2018.

[10] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan. Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders, 7 2017.

[11] Antoine Caillon and Philippe Esling. RAVE: A variational autoencoder for fast and high-quality neural audio synthesis. 11 2021.

[12] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schölkopf. Wasserstein Auto-Encoders. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 11 2017.

[13] Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brebisson, Yoshua Bengio, and Aaron Courville. MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis. *arXiv*, 10 2019.

[14] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis. *Advances in Neural Information Processing Systems*, 2020-December, 10 2020.

[15] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. *arXiv*, 6 2020.