ATIAM 2023 - ML Project
Exploring the latent space of a temporal VAE for audio loops generation

Sarah Nabi[1] Philippe Esling[1]
[1] Institut de Recherche et Coordination Acoustique Musique (IRCAM)
UMPC - CNRS UMR 9912 - 1, Place Igor Stravinsky, F-75004 Paris
{nabi, esling}@ircam.fr

December 2023

Abstract

In recent years, deep generative models have achieved impressive results in generating high-quality audio samples offering promising tools for musical creation. These approaches mostly rely on *latent-based generative models* such as *Variational Auto-Encoders* (VAE) [5] which can provide a trainable analysis-synthesis framework reconstructing audio samples through a *latent* representation that captures high-level signal features. We can then sample from the latent distribution to generate new sounds with properties similar to those of the training set. However, controlling these models is challenging as the latent dimensions are usually not directly interpretable. These latent variables are learned without supervision and highly depend on the initial dataset. In this project, we propose to explore and intuitively assess with a multimodal approach the latent space of a temporal VAE, such as RAVE [1], pre-trained on electronic music loops, using a hand motion dataset.

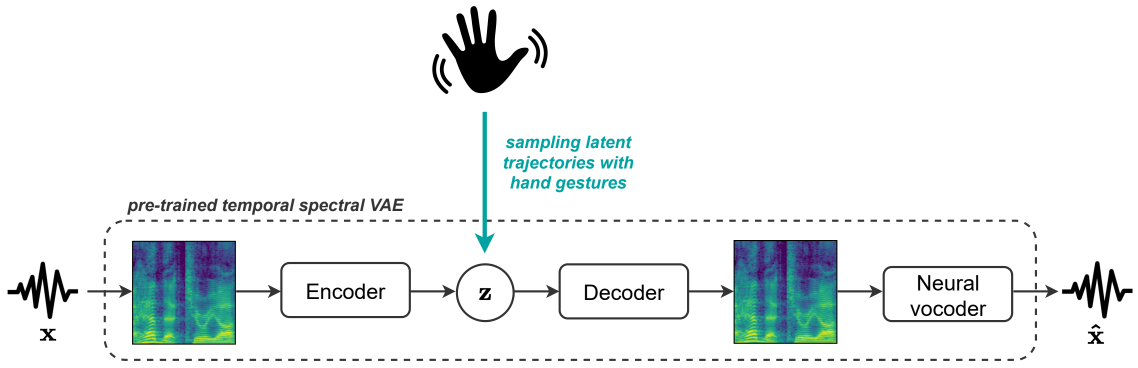
sampling latent trajectories with hand gestures
pre-trained temporal spectral VAE
x
Encoder
z
Decoder
Neural vocoder
x̂

Figure 1: Overview of the proposed approach to explore and intuitively assess the latent space of a pre-trained temporal spectral VAE by sampling audio latent trajectories with hand gestures to generate new electronic music loops.

1 Introduction

In recent years, neural audio synthesis has become an actively research topic offering new creative prospects in terms of composition and sound design. Generative models such as DDSP [3], GANSynth [2] and RAVE [1], have achieved impressive results in generating high-quality audio samples that reproduce the properties of a given training set. Modeling the synthesis process is challenging as audio data is very high-dimensional. Hence, deep generative approaches such as VAEs [5] introduce latent variables to constrain the model and capture high-level features in a lower dimensional space called the *latent space*. However, the latent representations are very abstract and still too high-dimensional to be directly humanly understandable, which precludes straightforward and intuitive control. Moreover, we lack evaluation metrics to assess the quality of these latent spaces. Hence, the goal of this project is to address these two challenges by leveraging another dataset composed of hand movements to explore and sample the latent space of a pre-trained VAE-based audio synthesis model in order to assess its quality and coverage. You will first implement and train the synthesis model on the provided electronic music loops dataset and then evaluate it by sampling latent trajectories with hand gestures. A diagram of the proposed approach is depicted in Figure 1.

2 Getting started

In this project we will use PyTorch [1], a versatile machine-learning Python framework that you will use to code your models. You should first set up your environment for the project and get familiar with Pytorch.

1. Create the github repository that will host your wonderful code and share it with us.

2. Create a virtual environment with Python 3.10. We advise you to use the `conda` or `miniconda` package manager to set your dependecies properly [2]. This is not mandatory, but can save you a lot of time in case of problems.

3. It is also advised you structure your code as a *package* [3]. For example, you can start with the following structure:

```
<repo-name>/
 config/  # directory to store the config files
 data/    # directory to store the data in local /!\ DO NOT COMMIT /!\
 docs/    # for github pages content
 models/  # directory to store checkpoints in local /!\ DO NOT COMMIT /!\
 notebooks/  # jupyter notebooks for data exploration and models analysis
 README.md   # documentation
 requirements.txt   # python project dependencies with versions
 .gitignore   # indicate the files not to commit
 src/
    <package-name>/  # main package
        __init__.py
        datasets/       # data preprocessing and dataloader functions
           __init__.py
        helpers/        # global utility functions
           __init__.py
        models/         # models architecture defined as class objects
           __init__.py
 tests/   # tests package with unit tests
```

4. Install PyTorch in your virtual environment. Many tutorial are available on their website [4] and it is strongly advised to become familiar with its use before continuing the project. Look at the basic tutorials and learn how it works `https://pytorch.org/tutorials/beginner/basics/intro.html`. Learn how to implement basic models using `torch.nn` module.

5. Install and read about the support libraries for audio signal processing: `librosa` [5] and `torchaudio` [6].

6. Install `tensorboard` [7] in order to monitor your trainings.

[1] https://pytorch.org/
[2] https://docs.conda.io/projects/conda/en/latest/user-guide/getting-started.html
[3] https://docs.python-guide.org/writing/structure/
[4] https://pytorch.org/tutorials/
[5] https://librosa.org/doc/latest/index.html
[6] https://pytorch.org/audio/stable/index.html
[7] https://pytorch.org/docs/stable/tensorboard.html

3 Build a simple VAE on MNIST

This part of the subject must be done individually in a python notebook. First, you will implement and train a VAE [5] on a simpler problem: digits generation. Therefore, you will use the MNIST dataset which is composed of 70,000 examples of hand written digits from 0 to 9, and which is often used as a playground to test new methods. We suggest that you write your code in a *modular* way so that you can reuse it for the rest of the project.

You will first look at the main article's implementation [5], and then implement it on Pytorch. To help you, you can look at this nice tutorial : `https://keras.io/examples/generative/vae/` (programmed in Keras otherwise it's too simple).

1. Based on the tutorial or another source you will find, develop your very own VAE. You can compare different architectures (e.g. linear layers, convolutions, batch-normalization etc.)
2. Train your model on the MNIST dataset. As the output is binary, formulate what should be its loss function. We strongly recommend you to use tensorboard in order to monitor your trainings (losses, reconstruction, latent space plotting).
3. Compare your model's outcomes to the VAE results from [5].
4. Implement the warm-up procedure for the ELBO (the regularization coefficient β varying from 0 to 1 [4]) during N epochs. What is it made for?
5. Once the model has achieved a reasonable reconstruction quality, you can generate new data by sampling the latent space: decode random points, perform linear and/or spherical interpolations between two points.
6. Start writing the final report. Create an overleaf and share the link with us. In the state-of-the-art section, explain theoretically the VAE model and its loss. In another section which can be named ”experiments on MNIST”, describe the different experiments and implementations you did and discuss the qualitative results (e.g. the generated images).

4 Temporal Spectral VAE

Keep your MNIST VAE aside and now adapt your VAE in order to generate musical electronic loops.

4.1 Dataset - data preparation

We will provide you a the audio dataset we prepared using a subset of the Freesound Loop Dataset (FSLD) [8] composed of audio loops sampled at 44100 Hz and which consists of audio chunks of 65536 samples (e.g. $\approx$ 1,5 seconds).

A mandatory first step for adequate learning is to pre-process the data into known ranges and properties. Indeed, contrary to images where we directly give raw information to the model, we will rely on time-frequency representations. Here, we will focus on mel-spectrograms.

First, we extract the magnitude spectrograms from the raw waveforms. To do so, you can use a *Short-Time Fourier Transform* (STFT). Then, we convert these spectrograms in the mel scale (using a mel filter bank) to obtain mel-spectrograms. To ease the training, we additionally constrain the range of values in the mel-spectrograms by applying a *log1p* transform ($f(x) = log(1+x)$). We can then train our model on these representations.

To retrieve the audio signals from the generated spectrograms, we must invert these transformations. To invert the spectrogram, you can initially use the *Griffin-Lim* algorithm (or the *Phase Gradient Heap Integration* (PGHI) algorithm which is more efficient). We will then give you a pre-trained neural vocoder so that you can generate high-quality audio waveforms when exploring the latent space.

1. Based on the `torchaudio`, `librosa` and/or `tifresi` packages, implement the data processing transformations in a class `AudioTransform` with two methods:

[8] `https://zenodo.org/records/3967852`

    - `def forward(wav, **kwargs)`: compute mel-spectrograms from raw waveforms,
    - `def invert(spec, **kwargs)`: invert mel-spectrograms and return waveforms.

2. Implement the torch dataset and dataloader to train your model using the `AudioTransform` class.
   NB: Do not forget to normalize your spectrograms!

4.2 Model - Implement your Temporal Spectral VAE on mel-spectrograms

You will now adapt your VAE to a fully convolutional neural network to generate mel-spectrograms using the musical electronic loops dataset we provided. Design and train your Temporal Spectral VAE using 1D-convolutions to encode each frame. Find the architecture that provides the best reconstruction.

5 Latent space exploration

It's time for exploration ! For this last section, we will provide you with a RAVE model pre-trained on the same dataset so that you can compare both models.

5.1 Qualitative evaluation and visualisation of informative latent dimensions

Apply the post-training latent analysis introduced in the RAVE paper [1] on your model to extract the most informative latent dimensions. Project the audio dataset on the most informative dimensions and plot the results (PCA). Try the UMAP algorithm to identify some clusters. Sample along the 3 most informative latent dimensions individually and listen to the results. Compute some correlation scores between these dimensions and acoustic descriptors (such as loudness, centroid, etc.).

Randomly sample a smooth trajectory inside your latent space, and listen to your generative model gradually evolve. You can also try 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.

5.2 Use the Hand motion dataset to sample latent trajectories and asses the quality and coverage of your generative model

References

[1] Antoine Caillon and Philippe Esling. Rave: A variational autoencoder for fast and high-quality neural audio synthesis. *arXiv preprint arXiv:2111.05011*, 2021.

[2] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. Gansynth: Adversarial neural audio synthesis. *arXiv preprint arXiv:1902.08710*, 2019.

[3] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts. Ddsp: Differentiable digital signal processing. *arXiv preprint arXiv:2001.04643*, 2020.

[4] Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew M Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

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