ATIAM 2019 - ML Project
Granular synthesis using variational learning

Constance Douwes[1], Axel Chemla–Romeu-Santos[1] Philippe Esling[1]
[1] Institut de Recherche et Coordination Acoustique Musique (IRCAM)
UMPC - CNRS UMR 9912 - 1, Place Igor Stravinsky, F-75004 Paris
{douwes, chemla, esling}@ircam.fr

November 2019

Abstract

Generative systems are machine-learning models whose training is based on two simultaneous optimization tasks. The first is to build a latent space, that provides a low-dimensional representation of the data, eventually subject to various regularizations and constraints. The second is the reconstruction of the original data through the sampling of this latent space. These systems are very promising because their space is a high-level, "over-compressed" representation that can be used as an intermediate space for several tasks, such as visualization, measurements, or classification. The main goal of this project is to use variational models for raw audio in order to create a granular synthesizer based on latent sampling.

1 Introduction

Among recent generative systems found in the literature, several have had a large success in the machine learning community. Among them, we find the variational auto-encoder (VAE) which has shown great generalization properties and good reconstruction capabilities despite its light structure. VAEs are based on two structures : first an encoder, that projects input data to an abstract space called the *latent space*, and a decoder, that gives back the corresponding data from the latent position. Latent space can thus be understood as a representation for the learned database, that can be navigated freely to generate coherent data with the original data space.

The idea behind this project is to use VAE to produce audio grains that could serve a granular synthesizer. The first task in hence to train a VAE to reproduce audio grains given a selected database. Then the second task is to sample the latent space to generate audio grains, material of the granular synthesizer.

2 Variational Auto-Encoder

First, you will have to learn PyTorch, a machine-learning Python framework that you will use to code your Variational Auto-Encoder (VAE).

Exercise 0 : Bibliography

1. Put your glaces and read the nice tutorial on variational auto-encoding of Blei [1], and the main articles of variational auto-encoders [3, 2]
2. If you feel it, read other papers

Exercise 1 : Install and learn Pytorch

This exercise only includes basic programming skills that you will need for the following work. If we already have enough expertise, you can skip it but I'd rather recommend you to have a look.

1. Install PyTorch on your device if it isn't already done.
2. Follow the basic tutorials and learn how it works `http://pytorch.org/tutorials/`.
3. Learn how to implement basic models, therefore use `torch.nn` library.

Exercice 2 : Code the VAE

You will know code your own VAE (finally). You can either look at the main article's implementation, or rely on this nice tutorial `https://wiseodd.github.io/techblog/2016/12/10/variational-autoencoder/` with nice mathematical explanations, programmed in Keras (otherwise it's too simple).

1. Based on the tutorial or another source you will find, develop your very own VAE.
2. Test your model on the MNIST dataset. As the output is binary, formulate what should be its loss function.
3. Compare your models to the VAE results from Kingma `&` Welling [3].
4. Implement a warm-up. For example β varying from 0 to 1.

3 Audio grain generator

Exercise 3 : Datasets

1. Choose a database you want to work with, it can be drum samples, synthetic samples (from DIVA synthesizer) or classical instrument samples (violin / piano /flute ...). I have all these types of datasets which can save you time (precious at that point of the project).

2. Create your own one if you feel like it (waveform only) but don't spend too much time on it.

Exercise 4: Grain-VAE training

1. Now that you have a functional VAE, change it for continuous outputs (instead of binary ones before). Be careful you have now to output a Gaussian distribution, once again formulate what should be its loss function.

2. Fix a grain length (for example 1024 points sampled at 22,05k). Then create a convolutional encoder and decoder with several 1D convolutional layers (and transposed ones).

3. Don't forget to add saving and plotting functions.

4. Verify your code works, and send it to me so I can run them on our sweet GPUs.

4 Granular synthesizer

Exercise 5 : Simple Granular Synthesizer

1. Do some tutorials on Max (or on PureData) to be comfortable with `groove~` ,`cycle~` ,`buffer~` ,`kslider` objects. The max helper is essential and very useful.

2. Create either on Max (or on PureData) a granular synthesizer based on sample sampling

Exercise 6 : Variational Granular Synthesizer

Comming very soooooon

5 Have fun

1. We now have a variational granular synthesizer. Now that the main core is developed, how could you improve the interaction of a user with this synthesizer (additional knobs, additional options, you are free about almost anything...!) ?

2. Make an article..?

3. Max for Live ;)

References

[1] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, Feb 2017.

[2] 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.

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