ATIAM 2021 - ML Project
Audio networks hacking : novelty search through active divergence

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
{chemla, esling}@ircam.fr

November 2021

Abstract

Variational auto-encoders are generative 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. Most of these models are generally used for representation or interpolation, but some of these models are also able to generate new contents that to do not belong the original data. The aim of this project is to inspect the innovative abilities of these models, and to chase where the model generates creative content through diverse "hacking" methods.

1 Introduction

The development of machine learning-based generative systems that occurred within the last decade provided a flourishing diversity of models, applicable to different domains including image processing, audio processing, 3d processing, language processing, and many more.
Among them, the *variational auto-encoder* (VAE) is a versatile model that has shown simultaneously great generalization properties and satisfactory reconstruction abilities, despite its light structure. VAEs are based on two processes : an *encoding* process, projecting input data to an abstract representation called the *latent space*, and a *decoding* process, giving back the data corresponding a given latent position. This latent space can thus be understood as a compressed representation of the learned database, whose reduced number of dimensions can then be navigated freely to generate data corresponding with the underlying structure of data.

However, while such properties are typically desirable for a task-oriented system, the use of this model is constrained to an *interpolation* regime, where novel samples are generally non-linear interpolations between training examples. The purpose of this project is to evaluate the ability of this generative model to produce new artifacts, and to develop several procedures to lead it to perform *out-of-domain* generation. This novel axis of research, called *active divergence* by the literature, is proposes several procedures to whether explore the "dark corners" of generation abilities of the model, or to enforce them to get out of their former distribution by fine-tuning or weight corrpution. We will adapt these methods to the audio domain, taking a dataset of 11 orchestral instruments, and aim to generate spectro-morphologies that may produce new timbres and sounds that differ from their origin.

2 Variational Auto-Encoder

In this project we will use PyTorch [1], a versatile machine-learning Python framework that you will use to code your Variational Auto-Encoder (VAE).

[1] https://pytorch.org/

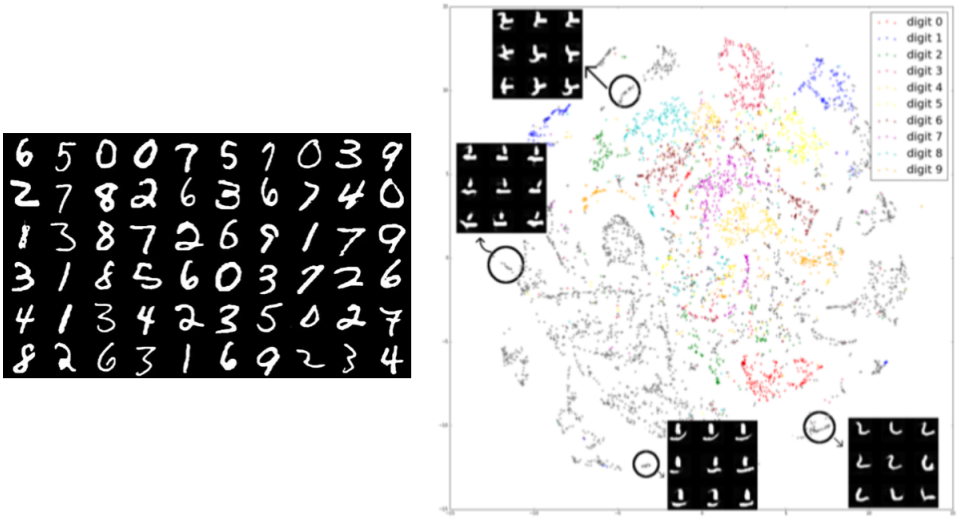

Figure 1: Novelty search in internal representations of deep learning models may yield to unknown contents (right) that are not contained in the base distribution (left). This project aims to explore new content of a generative model trained on acoustic instruments, hence retrieving new spectro-morphologies induced by the model.

Exercise 0 : First bibliography

1. Put your glasses and read this nice tutorial on variational auto-encoding of Blei, Kucukelbir, and McAuliffe, "Variational Inference: A Review for Statisticians", and then some articles grounding the topic Kingma and Welling, "Auto-Encoding Variational Bayes"; Higgins et al., "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework".

2. You can also read Broad, Berns, et al., "Active Divergence with Generative Deep Learning–A Survey and Taxonomy" to get an overview of *active divergence* techniques (this is not required in what's next though).

Exercise 1 : Install and learn Pytorch

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

1. Create the github repository that will host your wonderful code.

2. We advise you to used 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. It is also advised you structure your code as a *package* [3].

3. Install PyTorch on your device if it isn't already done. Follow 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` library.

4. Follow additional tutorials, that will help you for whats next : convolutional image classification, CUDA semantics and torchaudio documentation.

[2] `https://docs.conda.io/projects/conda/en/latest/user-guide/getting-started.html`
[3] `https://docs.python-guide.org/writing/structure/`

Exercise 2 : Code the MNIST VAE

You will know code your own VAE (finally). First, you will train on the MNIST dataset (the most classical dataset for image processing), that you should be able to train on your computers. You will first look at the main article's implementation, and then implement it on Pytorch guided by 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.
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 models to the VAE results from Kingma and Welling, "Auto-Encoding Variational Bayes".
4. Implement the warm-up procedure for the ELBO (the regularization coefficient β varying from 0 to 1) during N epochs. What is it made for?.
5. (Bonus) Perform a grid search on the most important parameters (latent space dimensionality, neural network capacity, regularization coefficient), and keep that apart for what follows.

Exercise 3 : Code the audio VAE

Keep your MNIST VAE aside and now adapt your VAE in order to learn the spectral content of 11 classical instruments (*acidsInstruments*). To this end we will learn the spectral magnitude of these sounds using short-term Fourier transform (STFT) or a more powerful one, called *non-stationary Gabor transform*.

1. Based on the `torchaudio` package, implement the procedure to import the magnitude content of an STFT. You will also have to implement the inversion from magnitude spectra to waveform using Griffin-Lim phase reconstruction.
2. Using the nsgt package of Pyhton, also implement an AudioTransform for NSGT compatible with your VAE.
3. Train your VAE on the acidsInstruments dataset, finding the architecture that provides the best reconstruction. As the data here is not binary but continuous, what will we us as output distribution?
4. Implement basic latent exploration functions such as linear / spherical random trajectories, encoding / decoding arbitrary sounds...
5. (Bonus) If you got time, perform a grid search on the most important parameters (latent space dimensionality, neural network capacity, regularization coefficients), and keep that apart for what follows.

3 Active divergence with generative models.

Latent space exploration, both with models trained on MNIST and acidsInstruments, often yield generations that are mix-ups of several data examples. While this smoothness is a great property of VAEs, this can be quite disappointing when looking for the novel data that has been inferred by your system. Hence, we will try three different *active divergence* methods, both on image and audio, that will

- enforce the system to generate new content through random corruption and loss hacking methods
- chase the novel samples through latent space analysis and generative feedback
- use an external classifier that will enforce the system to be unpredictable given some classes.

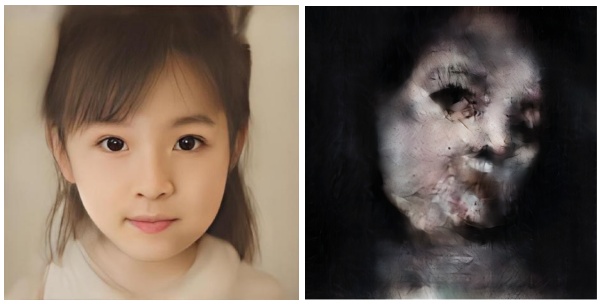

Figure 2: Some artists generate new artifacts by destroying some parts of generative models, as above with a BigGAN trained on faces (left) and then amputated to generate corrupted artifacts (right).

Exercise 4 : Destroy your own model.

- Which latent regions of your model are the most likely to generate novel examples? Try to generate some.
- Try to apply some random corruption to your weights, and check what comes out with various amounts of noise. Can you infer the qualitative effect of such procedure in the generative results of your algorithm?
- Could you find a way to freeze the generator and automatically find the latent positions that maximize the *unlikelihood* of your model? (you can take some inspiration from Broad, Leymarie, and Grierson, "Amplifying the uncanny").

Exercise 5 : Latent space exploration for novelty search.

- Read Kégl, Cherti, and Kazakçı, "Spurious samples in deep generative models: bug or feature?" to get more insights about trade-offs between interpolation/extrapolation of generative models. If you can, evaluate your different models with the proposed measures (you will be provided a specific dataset, *acidsInstruments-test*, with instruments unseen by the algorithm).
- Implement with your MNIST VAE the procedure described in Kazakçı, Mehdi, and Kégl, "Digits that are not: Generating new types through deep neural nets".
- Implement the same procedure with your audio VAE, and chase the unknown sounds coming from your VAE.

Exercice 5 : Fool your model with a classifier

- Train a simple classifier based on the original instrumental samples.
- Fine-tune the model to discriminate accurately the generations of your VAEs, including unseen samples.
- Freeze your classifier, and train to fool it with your generator to produce artifacts that are classified in a an uniform manner across every classes (you can take inspiration from Elgammal et al., "Can: Creative adversarial networks, generating" art" by learning about styles and deviating from style norms"). You can also train the classifier, implementing an adversarial loss. Evaluate how the system react.

References

Blei, David M., Alp Kucukelbir, and Jon D. McAuliffe. "Variational Inference: A Review for Statisticians". In: *Journal of the American Statistical Association* 112.518 (Feb. 2017), pp. 859–877. ISSN: 1537-274X. DOI: 10.1080/01621459.2017.1285773. URL: http://dx.doi.org/10.1080/01621459.2017.1285773.

Broad, Terence, Sebastian Berns, et al. "Active Divergence with Generative Deep Learning–A Survey and Taxonomy". In: *arXiv preprint arXiv:2107.05599* (2021).

Broad, Terence, Frederic Fol Leymarie, and Mick Grierson. “Amplifying the uncanny”. In: *arXiv preprint arXiv:2002.06890* (2020).

Elgammal, Ahmed et al. “Can: Creative adversarial networks, generating” art” by learning about styles and deviating from style norms”. In: *arXiv preprint arXiv:1706.07068* (2017).

Higgins, Irina et al. “beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework”. In: *ICLR*. 2017.

Kazakçı, Akın, Cherti Mehdi, and Balázs Kégl. “Digits that are not: Generating new types through deep neural nets”. In: *International Conference on Computational Creativity*. 2016.

Kégl, Balázs, Mehdi Cherti, and Akın Kazakçı. “Spurious samples in deep generative models: bug or feature?” In: *arXiv preprint arXiv:1810.01876* (2018).

Kingma, Diederik P and Max Welling. “Auto-Encoding Variational Bayes”. In: *arXiv:1312.6114* (2013). arXiv: 1312.6114 [stat.ML].