ATIAM 2023 - ML PROJECT
MULTI-ENTITY VARIATIONAL AUTO-ENCODERS

Giovanni Bindi
`bindi@ircam.fr`

Philippe Esling
`esling@ircam.fr`

November 2023

ABSTRACT

In this project we will work with Variational Auto-Encoders (VAEs), a deep latent variable model (DLVM) introduced by [Kingma and Welling, 2013] and [Rezende et al., 2014]. We will first learn a VAE on a dataset composed of single notes and then we will make small modifications to the original VAE formulation in order to inject more structure in the encoding-decoding process. By following the Multi-Entity VAE approach described here, we will attempt to learn interpretable representations in an unsupervised way. Such representation will be also used to perform *generation by recomposition*, which is the process of generating novel data points by composing elements extracted from multiple sources.

1 Introduction

Generative models are one of the most important research lines in both statistics and deep learning. Many families of (deep) generative models exist, such as VAEs, Generative Adversarial Networks, Normalizing Flows, Diffusion Models, Autoregressive Models and Energy-Based Models. You can refer to [Bond-Taylor et al., 2022] for a recent review on some of these methods. In this project we will focus on **VAEs**, as they provide a simple yet effective playground for experimenting with probabilistic modelling. VAEs are deep latent variable models, built on the assumption that observed variables $\mathbf{x} \in \mathbb{R}^{\mathbf{n}}$ (data points) are associated with unobserved (latent) variables $\mathbf{z}$ which encode their structure, while living on a lower-dimensional manifold. You should be familiar with the VAE framework, but you can refer to [Kingma and Welling, 2019] for a comprehensive review of both VAEs and DLVMs. In the original VAE formulation the generative latent factors are assumed to be continuous, i.e. $\mathbf{z} \in \mathbb{R}^{\mathbf{d}}$ for some $d \ll n$.

2 Tasks

We will use the PyTorch framework so, if you don't already know the basics of PyTorch, go through some of the tutorials in the documentation. We suggest you to use the miniconda package manager and install the required dependencies into a separate environment (specifically created for this project). Moreover, if you are not familiar with Python programming, we suggest you to use an automatic PEP 8 style formatter (e.g. black or yapf). Finally, we suggest you to organize your code in a modular way, such that each component has its own responsibility, and upload it on a git repository[1] (e.g. GitHub).

If you already have experience with PyTorch and you want to learn a new deep learning framework you can also use JAX+Flax, it will add a functional twist to your code but at the expense of less documentation and some few possible downsides.

We suggest you to use the tensorboard platform to log your metrics, loss(es), results and so on.

[1]Remember to exclude your dataset (and any other large file) from the git.

2.1 Code a VAE on MNIST

The first step is to code a Variational Auto-Encoder. The web is full of VAE tutorials for each deep learning framework but it would be best[2] for you to re-implement the VAE from scratch. In order to do so you can follow this blog post and the references therein as a starting point.

- Use the MNIST dataset. This is the standard go-to dataset when experimenting with computer vision tasks.
- Compare your results to [Kingma and Welling, 2013] (and also check results of [Rezende et al., 2014]).
- Read [Higgins et al., 2017] and perform experiments with different values of β. You can also implement a warm-up strategy where β grows from 0 to (e.g.) 4. Can you see the impact of the β factor?

2.2 Code a VAE on audio

Let's put aside the MNIST VAE and move on the audio domain. Many components of your code should be re-usable and you should only "throw away" the model and the data-loading pipeline. From now we will rely on a dataset composed of 6 instruments[3] playing a single note, the NSynth dataset. Audio content can be represented in different ways and in this step we can experiment with two different representations, namely spectrograms and waveforms. Here, we will work with spectrograms but, if you want to work on waveforms, we can pick this (eventually more challenging) representation. In this case, it could be beneficial to read [Caillon and Esling, 2021] and the references therein, as you will have to experiment with architectural choices and hyper-parameters.

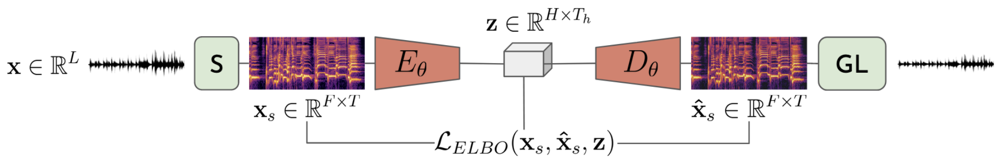
$\mathbf{x} \in \mathbb{R}^L$
S
E_θ
$\mathbf{z} \in \mathbb{R}^{H \times T_h}$
D_θ
GL
$\mathbf{x}_s \in \mathbb{R}^{F \times T}$
$\hat{\mathbf{x}}_s \in \mathbb{R}^{F \times T}$
$\mathcal{L}_{ELBO}(\mathbf{x}_s, \hat{\mathbf{x}}_s, \mathbf{z})$

Figure 1: High-level schema of the audio VAE. The blocks S and GL respectively represent the spectrogram and Griffin-Lim transforms. Then, we learn an encoder E_θ and a decoder D_θ by optimizing the ELBO.

Since the dataset is provided as 16 kHz waveforms you will have to design a pre-processing procedure in order to extract the spectrograms. Specifically, we will work with magnitude spectra obtained by the application of the short-term Fourier transform (STFT).

1. Using torchaudio, implement the *waveform* $\rightarrow$ *magnitude spectrogram* transformation. You will also have to implement the inversion *spectrogram* $\rightarrow$ *waveform* using the Griffin-Lim algorithm. You can also use Mel-Spectrograms.
2. Implement the data-loading procedures to import and iterate over both the spectrograms (or the waveforms).
3. Now that you have a functional data-loading process, start coding the VAE model. Here you will have to make design choices on the neural networks (e.g. number & type of layers, number of channels) and the model itself (latent space dimensionality d, regularizer value β). You will have to perform many experiments in order to find an acceptable set of hyper-parameters that will perform well on the task. Also keep in mind that the Griffin-Lim inversion algorithm will lead to poor audio quality: test its behaviour on the original data to understand its impact on the output. Once that you have a functioning model, remember to save it.
4. Experiment with latent space exploration: decode random points, perform (linear and/or spherical) interpolations between two points.
5. **Bonus**. You can add conditioning information (e.g. pitch, instrument class) in order to build a conditional model. You can read [Sohn et al., 2015] or this blog post to get acquainted with such type of models.

2.3 Modifications

Now that we have our vanilla VAE, let's apply some modifications to the original formulation in order to learn different representations. We will follow this paper and first learn object-centric representations. We will then apply this idea to the audio domain.

[2]Looking at others people code is absolutely fine (and recommended) but your first VAE should not be a copy-paste of some random code found on the internet.

[3]Bass, brass, flute, guitar, keyboard and vocal.

We aim to disentangle representations that correspond to specific entities in the original data (e.g. objects in images, instruments in audio). We will then impose an a-priori number N of entities that we will attempt to disentangle and then use N latent variables to represent them. These latent variables will be obtained initially by following the Multi-Entity VAE proposal. We will be following a strategy that enforces disentanglement by decomposing these entities into separate representations that are processed independently and re-composed together by a (non learned) composition operator.

This is the general idea behing Multi-Entity VAE (MVAE):

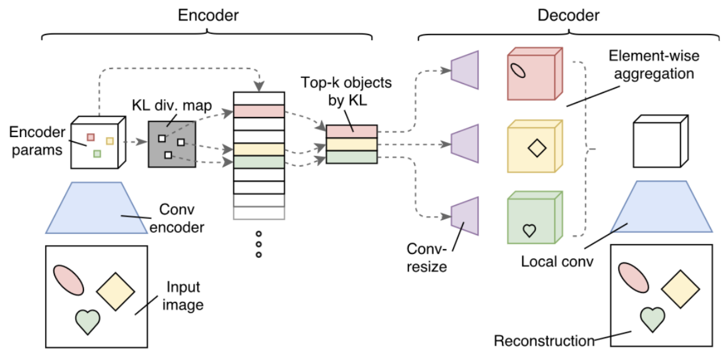

Figure 2: High-level schema of the MVAE. The encoder takes input images and produces a spatial map of posterior parameters $\mu, \mathbf{\Sigma} \in \mathbb{R}^{P \times H_l \times W_l}$. The KL-divergence of the posterior distribution against the prior $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$ is computed in each spatial location and the top-N posterior parameters by KL-divergence are selected. These parameters are then used to parameterize the Gaussian distributions that we will use to model our priors. These distributions are then sampled, and the samples are passed *independently* through a decoder network. Finally, the resulting objects are aggregated using an element-wise operation such as the sum.

In order to experiment with this model you will have to adapt your original VAE code to incorporate this latent variable selection scheme based on an information maximisation principle (i.e. finding the variables that maximise the KL divergence with the prior).

We will then proceed as following:

- Implement the MVAE model on a classic disentanglement dataset in computer vision such as CLEVR (which will be provided by us).
- Implement the code to perform:
  1. Single latent decomposition (in order to show if you are able to learn disentangled object-centric representations).
  2. Latent space re-composition (in order to be able to generate novel images by decoding and re-composing latent representations coming from different sources).
  3. KL-divergence map visualization.
- Experiment with different number of latent variables N and element-wise re-composition operators such as summation, maximum, minimum and mean. Can you tell the impact of the operator on the learned representations?
- Experiment with different architectural choices such as when to perform the re-composition. The original MVAE paper proposes to re-compose the decoded representation at an intermediate level in the decoding phase. What if you decode each representation separately and then re-compose them in the data domain?
- **Bonus.** Port this model to audio. Once that you are able to perform the previous experiments on images you can move to the audio domain. You can use the NSynth dataset to debug and make small modifications to the code. We will then provide a more structured dataset where two instruments are playing at the same time and the goal will be to use the MVAE model to (try to) learn disentangled instrument-centric representations.
- **Bonus.** Think about how to go beyond the independence assumption on the latent variables. This is an open research problems and we could think together on how to modify this model to better account for statistical relationships between the elements.

Global instruction

Generalities

Deadline	January 2024
Organization	Group from 4 to 5 students
Deposit	`Github`

Project folders

`code/` with your code following the PEP8 coding style and organized in modules	
`report/` in PDF format	

Report

Redaction	LaTeX with a **given format style** that you will receive
Language	English
Maximum number of pages	8 pages following the scientific papers

Evaluation grid

Report - Including content, results and style	**7 pts**
Code - Accuracy, evaluation and coding style	**13 pts**

References

[Bond-Taylor et al., 2022] Bond-Taylor, S., Leach, A., Long, Y., and Willcocks, C. G. (2022). Deep generative modelling: A comparative review of VAEs, GANs, normalizing flows, energy-based and autoregressive models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7327–7347.

[Caillon and Esling, 2021] Caillon, A. and Esling, P. (2021). Rave: A variational autoencoder for fast and high-quality neural audio synthesis.

[Higgins et al., 2017] Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017). beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*.

[Kingma and Welling, 2013] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes.

[Kingma and Welling, 2019] Kingma, D. P. and Welling, M. (2019). An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392.

[Rezende et al., 2014] Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models.

[Sohn et al., 2015] Sohn, K., Lee, H., and Yan, X. (2015). Learning structured output representation using deep conditional generative models. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.