Music and Image generation with Variational Auto-Encoders

Charles Dognin
charles.dognin@ensae.fr

Samuel Ritchie
samuel.ritchie@ensae.fr

Abstract

In this work, we implement a simple variational autoencoder with practical application to image and music data generation. The main goal is to learn a generative model from the data in order to allow us to efficiently sample from it for an accurate generation of new data. After presenting the basics of generative models and more specifically the theory behind VAEs, we describe our implementation and discuss our results.

1 Introduction

In statistical classification, including machine learning, the two main approaches are called the generative approach and the discriminative approach, which compute classifiers by different approaches, differing in the degree of statistical modeling. Generative models have recently been widely applied in literature, as they reveal themselves as a very powerful way of learning any kind of data distribution using unsupervised learning. It has achieved state of the art results in accurately generating data in fields such as imaging, music or text. The basic idea behind generative models is to collect a large amount of data in some domain and then train a model to generate data like it, following the intuition of Richard Feynman *"What I cannot create, I cannot understand"*. The trick behind such generative models is that the underlying neural networks have a significantly smaller dimension, forcing the model to discover and learn the most important features in order to generate it efficiently. In this sense, the problematic raised by such algorithms is strongly linked to dimensionality reduction issues. One of the most applied generative model in current litterature are the Variational Auto-Encoders (VAEs), derived from basic autoencoder artificial neural networks.
In this work, we will start by carrying out a quick review of the existing approaches regarding generative models. We will then give details on the main ideas and theoretical foundings of VAEs. In a second part, we will expose our implementation of such generative models in order to automatically generate images and music given a training set of a given image type or music genre, and eventually finish by discussing our results.

2 Why generative models ?

2.1 A quick review of generative models

Basically, a generative model comes down to describing how data is generated in terms of probabilistic model. Regarding supervised learning, a generative model estimates the joint probability distribution of data $P(X, Y)$ between the observed data X and corresponding labels Y. Another essential requirement of a generative model, which is less frequently stated, is that it provides a way to sample the data, whether we are in a supervised or unsupervised context. As opposed to discriminative models, generative ones are capable of doing more than just prediction. Indeed, by estimating $P(X, Y)$ and being able to sample data, a generative model can be used in various applications, such as imputation of missing data, dimensionality reduction, or most importantly generating unseen data. Regarding unsupervised learning, generative models address the core problem of density estimation in order to be able to accurately sample new data given a training set.

The way density is estimated splits generative models in different subtypes, as we will detail further on.
Applications of generative models are extremely wide in recent Machine Learning literature. The most original ones concern realistic sampling for artwork (new Monet paintings for example, or Jazz automatic generation), but it is also applied in fields such as reinforcement learning (simulation of time-series) or more generally in feature learning problems.

A large number of generative models exist and have been applied in literature, and one could thus define a taxonomy of the latter, depending both on the explicit or implicit structure of the density we want to estimate, and on the way we estimate it. A non exhaustive list of generative models consists in the following :

- Boltzmann machines, for explicit densities estimated by Markov Chain methods
- PixelRNN / CNN, for explicit tractable densities
- Generative Adversarial Networks (GANs) for implicit densities with direct estimation
- Variational Auto-Encoders, for explicit densities with variational bayesian estimation methods
- Generative Stochastic Networks, for implicit densities with Markov Chain estimation

Two of the most commonly used and efficient approaches are VAEs and GANs. These methods fundamentally differ in the approach for density estimation: GANs aims at achieving a Nash equilibrium between a Generator and a Discriminator, while VAEs, as we will see in part 3., aim at maximizing the lower bound of data likelihood.
In order to clearly present Variational Auto-Encoders, we will first give a few insights on basic autoencoder artificial neural networks as well as their limitations.

2.2 Auto-Encoders

Traditional Auto-Encoders are models whose goal is to learn a compressed representation of the data, similarly to the Principal Component Analysis. The autoencoder has two parts: an encoder and a decoder. The encoder encodes the input into a "code" (also called latent space), generally of lower dimension than the input in order to only keep the most important information. While the primary purpose of such models was dimensionality reduction, the rise of deep learning frameworks and interest in generating data made the autoencoder concept be widely used for learning generative models of data. In this context, the decoder decodes this code to reconstruct the output. The encoder and the decoder are generally neural network based functions which are learned with traditional gradient descent methods.

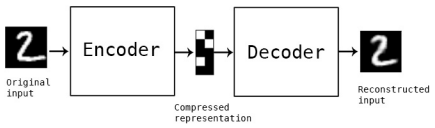

Figure 1: Functioning of a basic autoencoder

The loss function we want to minimize during training measures the gap between the original input and the reconstructed output. Many applications use autoencoder methods, among which dimensionality reduction as already mentioned, and most importantly denoising autoencoders allowing to find the relevant features in a blurred input signal. However, regarding generation of data from the learned representation, they are extremely limited. Indeed, the latent variable does not have a tractable distribution, or said in other terms the latent space may not allow easy interpolation. More precisely, one could say that autoencoders are fine for *replicating* data (thanks to clusters in the latent space), but is not good at *generating* new data because of eventual discontinuities in the latent space. Variational Auto-Encoders can help us overcome this obstacle.

3 Variational Autoencoders

As mentioned earlier, the added value of VAEs compared to vanilla autoencoders lies in the fact that instead of just learning a compressed representation of the data like autoencoders, they learn the parameters of a probability distribution representing the data. As a consequence, it enables us to sample from the distribution and generate new input data samples which makes them a generative model. In this section, we will present the main theoretical background of VAEs.

Let x be the data we want to model and z the latent variable (in a lower dimension). In what follows, we will refer to the decoder network distribution modeling as $p_\theta(x|z)$ and the encoder one as $q_\Phi(z|x)$. Our goal is to learn model parameters in order to maximize the likelihood of training data :

$$p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz$$

This problem is completely untractable since it is impossible to integrate over the whole latent space. The main idea of VAEs is thus to infer $p(z)$ using $p(z|x)$. Once again, the latter quantity is impossible to tract, which is why it is approximate once more, this time thanks to the encoder network. More precisely, $p(z|x)$ is inferred using Variational Inference, one of the most popular methods with MCMC when dealing with bayesian inference. Variational Inference models the true distribution using simpler distribution that is easy to evaluate, such as Gaussian ones, and minimizes the difference between those two distribution using KL divergence metric, leading to a so-called variational lower-bound, which we derive in what follows :

$$\begin{aligned}
\log p_\theta(x) &= E_{z\sim q_\Phi}\Big[\log p_\theta(x)\Big] \\
&= E_z\left[\log \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(z|x)}\right] \\
&= E_z\left[\log \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(z|x)}\frac{q_\Phi(z|x)}{q_\Phi(z|x)}\right] \\
&= E_z\Big[\log p_\theta(x|z)\Big] - E_z\left[\log \frac{q_\Phi(z|x)}{p_\theta(z)}\right] + E_z\left[\log \frac{q_\Phi(z|x)}{p_\theta(z|x)}\right] \\
&= E_z\Big[\log p_\theta(x|z)\Big] - D_{KL}\Big(q_\Phi(z|x), p_\theta(z)\Big) + D_{KL}\Big(q_\Phi(z|x), p_\theta(z|x)\Big) \\
&\geq \mathcal{L}(x, \theta, \Phi)
\end{aligned} \tag{1}$$

where $\mathcal{L} = E_z\left[\log p_\theta(x|z)\right] - D_{KL}\left(q_\Phi(z|x), p_\theta(z)\right)$ is the variational lower bound (also called 'ELBO'). This inequality is verified for every point x of the dataset, and the VAE training consequently consists in maximizing the obtained lower bound, i.e. $\theta^*, \Phi^* = \text{argmax} \sum_{i=1}^{N} \mathcal{L}(x_i, \theta, \Phi)$.

Interestingly enough, the variational lower bound decomposes in two separate terms and that have an interpretation regarding the model :

- the first part is called the *reconstruction error* : it is the log-likelihood of the observed x given the latent feature z we have sampled. It is linked to the decoder network performances $p_{x|z}$
- the second part corresponds to the difference between distributions $p(z)$ we want to estimate and $q(z|x)$ which is used to approximate it. In practice, standard normal distributions will be used. This part checks that the proposal distribution should be like a Gaussian (or any other chosen distribution) and is often called the *regularization term*

Once this lower-bound obtained, the idea is to perform gradient descent to estimate jointly the encoder and decoder parameters Φ and θ. When taking normal distributions, these parameters are simply the mean and variance. Optimization over Φ will keep ELBO tight around p and optimization

over θ. will keep pushing the lower bound up. This is often called *black box variational inference*. In order to obtain a lower variance gradient estimator of the ELBO, the authors use a reparametrization trick which allows (under certain conditions) to express $q_\Phi(z|x)$ as a two-step process :

- sample a noise variable ϵ from a simple distribution (standard normal for example)
- apply a deterministic transformation $g_\Phi(\epsilon, x)$ that maps the random noise into a more complex distribution.

For many interesting classes of q_Φ (such as standard normal distribution), it is possible to choose a $g_\Phi(\epsilon, x)$, such that z will be distributed according to $q_\Phi(z|x)$. The main advantages of this trick are that it allows the back-propagation to take place by making the sampling operation differentiable. Also, when taking the gradient of expectations in the ELBO, we can put the gradient inside, leading to lower variance estimators and consequently enabling us to learn models that we otherwise could not have learned.
For the gaussian distribution, instead of writing $z \sim q_{\mu,\sigma}(z) = \mathcal{N}(\mu, \sigma)$ we can write $z = g_{\mu,\sigma} = \mu + \epsilon\sigma$ where $\epsilon \sim \mathcal{N}(0, 1)$.

The below graph resumes the different steps data goes through in VAEs.

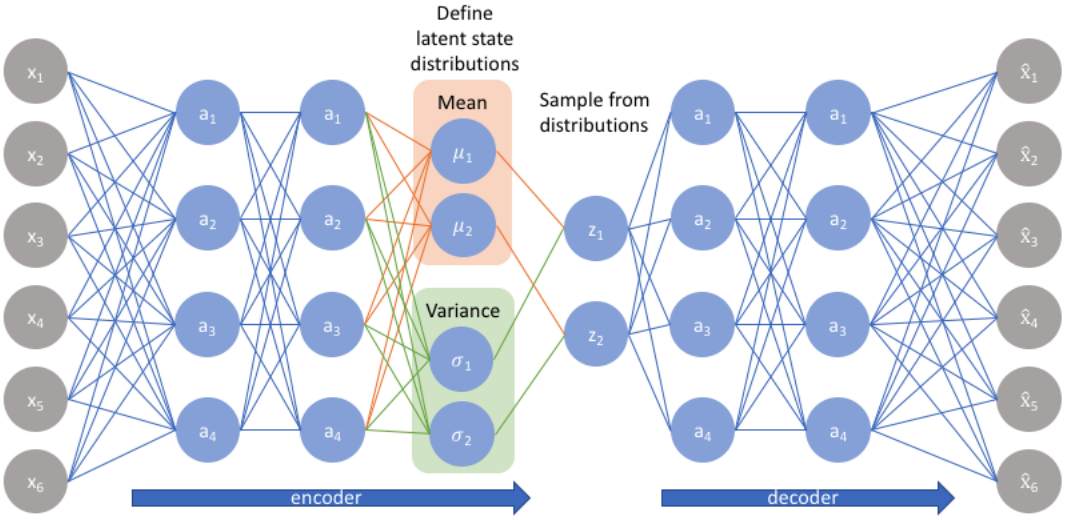

Figure 2: Functioning of a Variational Auto-Encoder

4 Implementation and results

Implementation was led on Python3 with the Keras deeplearning framework. The corresponding data and scripts can be found in the main Notebook of the project on the GitHub repository `https://github.com/charlesdognin/VAE_Based_Music_Generation`.

Regarding image generation, we obtained encouraging results when training on a set of 94 celebrity images. Two examples of generated images after 150 epochs are shown below. Details regarding the activation functions used as well as the neural networks used for encoding and decoding can be found in the Notebook.

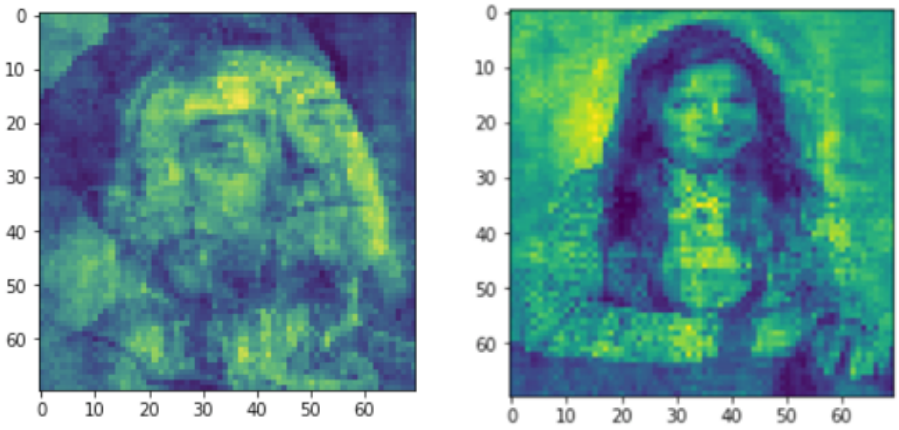

Figure 3: Images generated by VAE training on 90 celebrity faces (150 epochs)

We also obtained satisfying results regarding regarding music generation. The first step was finding a large amount of .wav files (high quality uncompressed musical audio format) of the same style. We used the scipy.io.wavfile.read function to turn the sounds into numpy vectors that could be fed to the neural network. We trained the VAE on 10 drum loops of approximately 5 seconds. With our personal laptops it was difficult to take longer musics, sound taking a considerable place compared to image (a 3 seconds song is a (30000,1) vector). We ran the network with 100 and 300 epochs. The generated sounds can also be found on the Github.

References

[1] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

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

[3] Adam Roberts, Jesse Engel, and Douglas Eck. Hierarchical variational autoencoders for music. 2017.