Data Generation with Autoencoders

Ly Christophe - Choukarah Ahmed - Maxime Minard

November 2021

1 Introduction

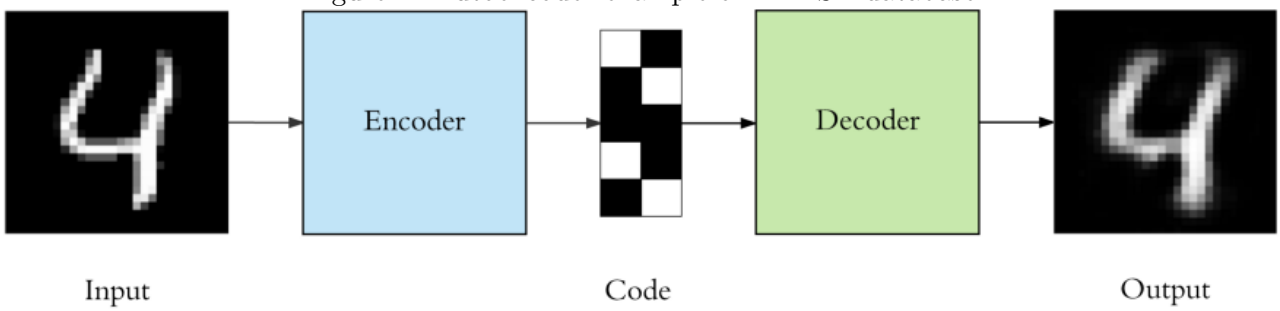
In recent years, the amount of available data has exploded. However, it is not always a good thing as a flood of un-treatable data is not something we might want. On the other side, fully labeled and ready to use data is often times expensive to obtain, that is when data generation comes in handy. The idea is to find a way to generate enough data automatically without relying on human intervention or annotation. To do so, many methods are available. In this report we will mainly focus on the Variational auto-encoders.

2 General Principles

2.1 Autoencoders

Auto-encoders are an unsupervised learning technique, where the main goal is dimensional reduction. More specifically, it is a special neural network model that has the same input and output dimensions, where the goal is to have the output be as close as possible to the input. The dimensional reduction come in practice with the existence of a hidden layer with dimension k strictly smaller than n the input dimension.

Figure 1: Autoencoder example on MNIST database



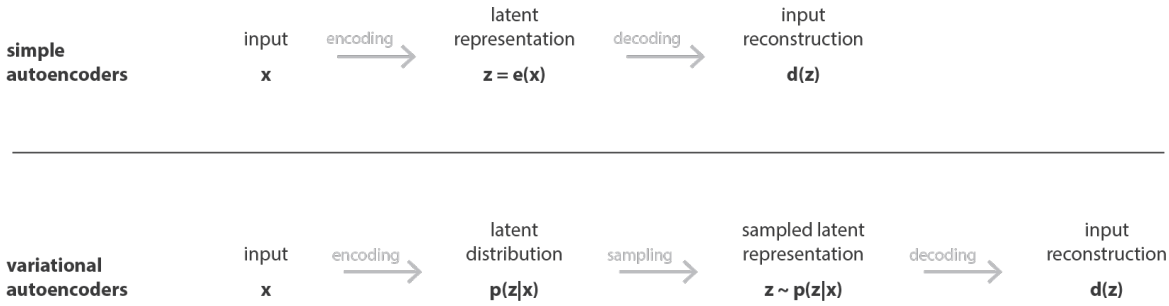
One of the advantages of auto-encoders over other dimensionality reduction methods such as PCA are that auto-encoders/neural networks are capable of learning non-linear relationships.

However, so far, encoders do not allow us to generate datapoints. We could technically sample random elements from the latent space and try to decode them, but without any guarantees on the regularity of the latent space it is not a viable option. That is why we turn to a variation of the autoencoder.

2.2 Variational Auto-encoders

Variational Auto-encoders are an evolution of normal auto-encoders. This time, instead of looking at encoding X in the latent space, we will be trying to encode it as a distribution (and in this case, a normal distribution) in the latent space. This will make it possible for us to have some regularity of the data over the latent space and be able to sample data from it.

In the variational autoencoder, this pipeline is :



Looking at the theoretical point of view, let's consider In this setting, as the goal is to infer good values of $p(z|x)$, we need this to be a good estimate for $q(z|x)$. We also use variational inference to set our estimator $p(z|x)$ in the family of gaussian distributions. As we want to get this as close as possible, we use the Kullback-Leibler divergence to characterize the "closeness" of the two distributions.

$$\begin{aligned} KL\left(q_x(z), p(z \mid x)\right) &= \left(\mathbb{E}_{z\sim q_x}\left(\log q_x(z)\right) - \mathbb{E}_{z\sim q_x}\left(\log \frac{p(x \mid z)p(z)}{p(x)}\right)\right) \\ &= \left(\mathbb{E}_{z\sim q_x}\left(\log q_x(z)\right) - \mathbb{E}_{z\sim q_x}(\log p(z)) - \mathbb{E}_{z\sim q_x}(\log p(x \mid z)) + \mathbb{E}_{z\sim q_x}(\log p(x))\right) \end{aligned}$$

This is equivalent to maximizing what we call the ELBO or the Evidence Lower bound.

$$\begin{aligned} ELBO =& \mathbb{E}_{z\sim q_x}(\log p(x \mid z)) - KL\left(q_x(z), p(z)\right) \\ =& \mathbb{E}_{z\sim q_x}\left(-\frac{\|x - f(z)\|^2}{2c}\right) - KL\left(q_x(z), p(z)\right) \end{aligned}$$

Where $f(z)$ defines the reconstructed x

The first term, called the reconstruction loss, is the difference between the input x and the output $\hat{x}$. The second term is the regularization term, where we try to regularize the latent space distribution .

3 Implementation and results

3.1 Autoencoders and Variational Autoencoders

First, we started implementing classical autoencoders. We used a network of convolutional layers, which way sufficient to work with MNIST dataset. The autoencoder we built works pretty good, and it manages to reconstruct pictures easily. However, as mentionned in the theorical part of the report, it does not work for data generation. Indeed, the results we got were too close to the data we used as input, and showed some irregularities in the "latent space".

Figure 2: Reconstruction of images with noise added between the encoder and the decoder for MNIST.

For our implementation of variational autoencoders, we used several datasets. First, we tried a simple variational autoencoder using the digit pictures from the MNIST dataset with convolutionnal layers in the network.

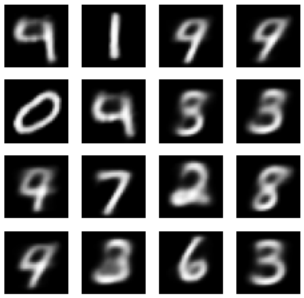


Figure 3: Pictures generated with CVAE from MNIST dataset - 20 epochs

The results we obtained on Fig.3 are pretty good since the pictures of this dataset are not very complex. It is pretty easy to distinguish the several digit numbers generated. A network of dense layers could have been enough for MNIST. To have a better understanding of what can be generated

through the latent space used on our variational autoencoder, we have also plotted several images of the latent space and the different clusters from our dataset. These representations show how images can transition from one class to another.

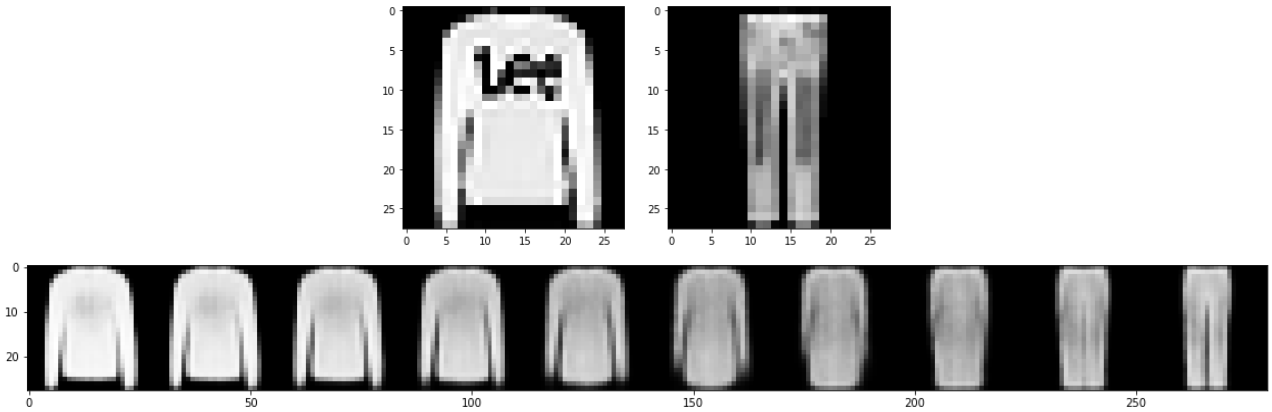


Figure 4: Pictures generated with CVAE from the Fashion MNIST dataset - 10 epochs

With a more complexe dataset such as Fashion MNIST we can see very interesting results (see Fig 4). When asked to generate images using the weighted average of two existing images on the latent space, the network create a transition. In-between similar objects such as a shirt and a trouser we can see a smooth and interesting transition. It is important to note that the transition between objects that don't share similarities (like a shoe and a shirt) give us an incoherent image.
When tested with a dataset such as Cifar 10 the results becomes very blurry due to the complexity of the images (see Fig 5). Different networks were tested every result were imprecise and didn't converged very quickly. One surprising results was that small and less complex networks worked better. Due to a lack of time we were only able to work on 10 epochs

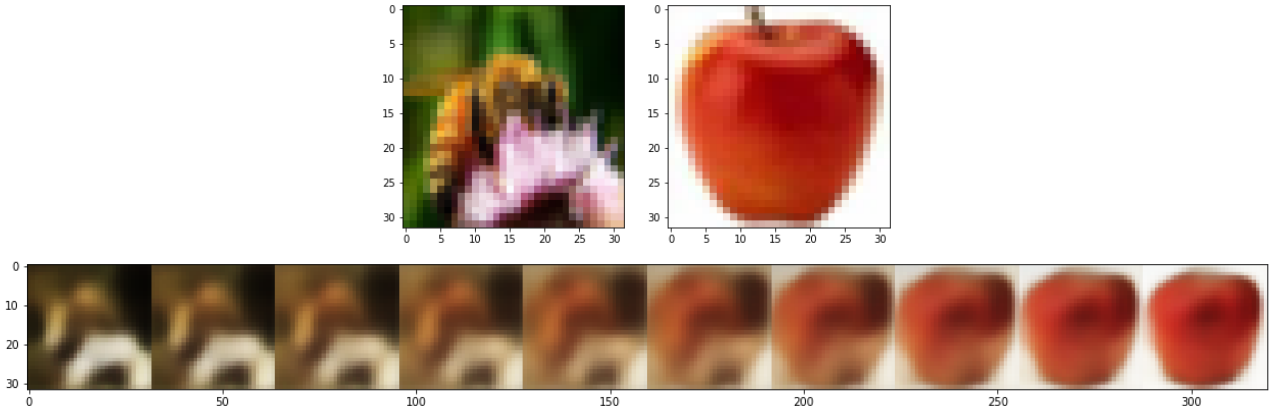


Figure 5: Pictures generated with CVAE from the CIFAR10 dataset - 10 epochs

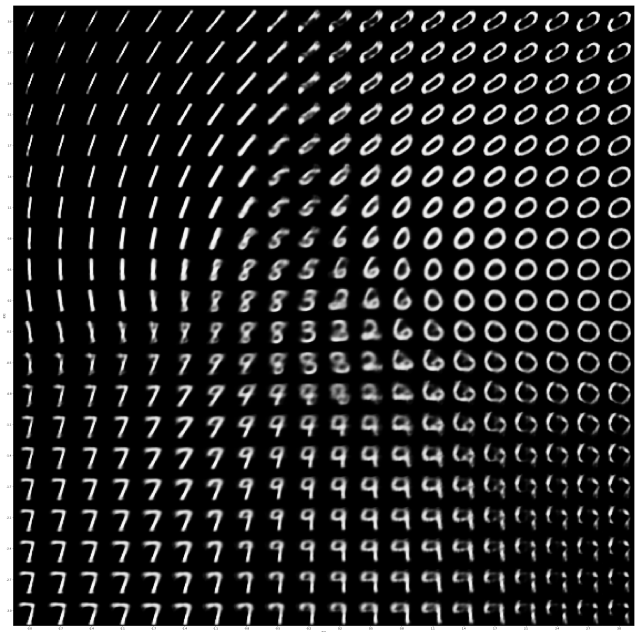


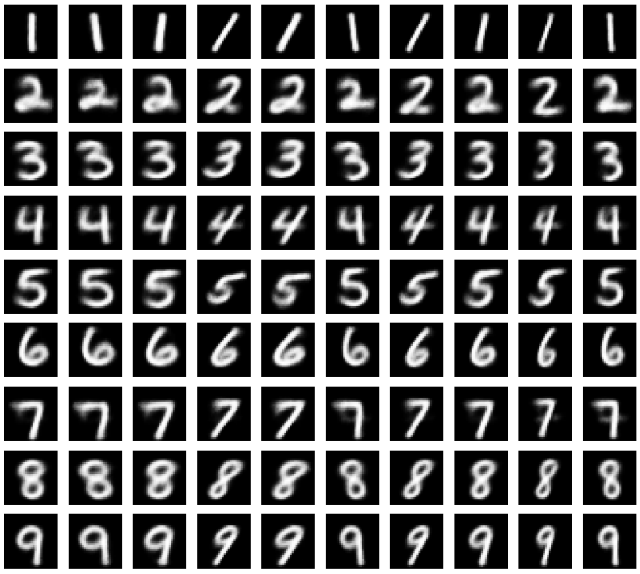
Figure 6: Sample of the latent space for CVAE.

3.2 Conditional Variational Autoencoders

While autoencoders remain a solid option for data generation, there is still the issue that we cannot choose exactly which type of data to generate, and in the MNIST case, which number to generate.

That is why we introduce the C-VAE, where we parametrize the entire network 2.2 with the label. This allows us to first separate the different latent spaces according to each number and to also choose which latent space to generate from.

With that in mind, our implementation used a very basic encoder/decoder architecture (a few dense layers) as convolutional networks are not possible anymore, here are the results obtained :



In This figure, we can see that we can generate from same space in the latent space but with different conditions different numbers. Moreover, we can see that these points in the latent space now represent characteristics of the handwriting such as thickness or angle.

4 Conclusion

Overall, while the variational autoencoders provide us with a fitting method to generate data, it does fail to produce good results when the problem becomes too complex such as in the CIFAR-10 Database. To conclude, we managed to have a good overview of what autoencoders are capable of, in terms of image reconstruction and image generation. It was a good opportunity for us to discover different kinds of autoencoders with the variational ones.

References

[1] Max Welling Diederik P. Kingma. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning: Vol. 12 (2019): No. 4, pp 307-392*, 2019.

[2] Joseph Rocca. Understanding variational autoencoders, September 2019. https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73.