Exploring generative models with Python

Javier Jorge Cano

PyCon 2018

Problem definition

- Several of the successful applications of machine learning techniques are based on the amount of data currently available.
- But sometimes data is scarce, i.e: difficult processes to collect it (i.e: medical imaging).
- We can obtain more data performing controlled distortions that do not modify the true nature of the sample, this is called **data augmentation**.

Figure 1: Left to right: original image and different alterations.

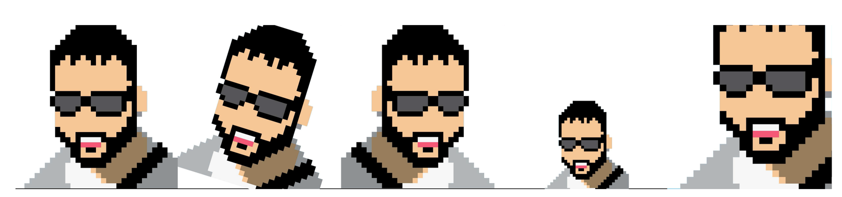

Figure 2: Another example of simulating different positions and rotations with different transformations.

- These transformations are hand-crafted and problem-dependent. Could we provide a domain-agnostic approach to do this?

Generative models

- Neural Networks are good at classifying, meaning that they learn a "mapping" between the input and the output.

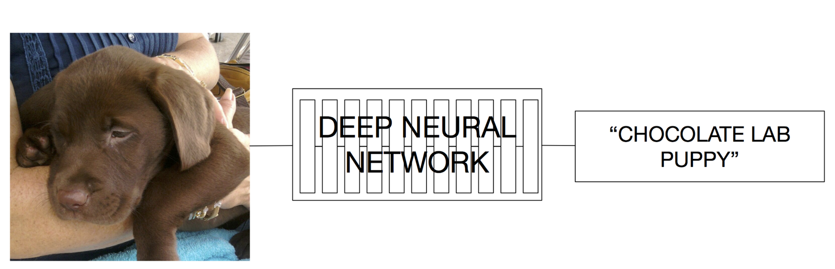

Figure 3: Example of DNN-based classifier.

- We can force this map to mirror the input, so we end up with a model that can reconstruct the sample.

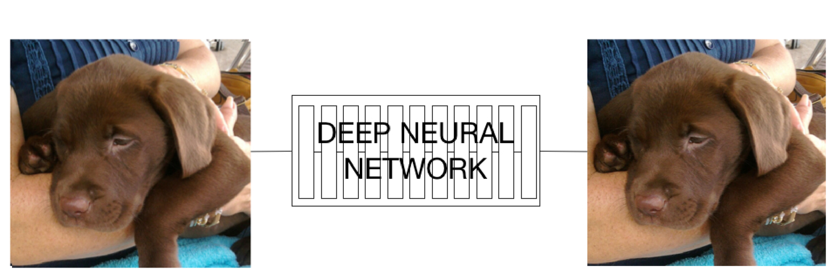

Figure 4: Example of DNN-based reconstruction model.

- These models usually encode the input in some manifold, and then decode it to recover the original sample.
- This kind of models are known as **Generative models**.
- Could we use this model's "**inner representation**" to create new outputs?
- Could these artificial samples be useful? (i.e: for training classifiers)

- Two techniques dominate recent approaches to deal with these problems. One line of research is based on **Generative Adversarial Networks** (GAN) and the other one is based on **Variational Autoencoders** (VAE).
- GAN: Based on the mixture of Game Theory and Machine Learning, where two networks (Generator and Discriminator) are competing in a game where one wants to fool the other.

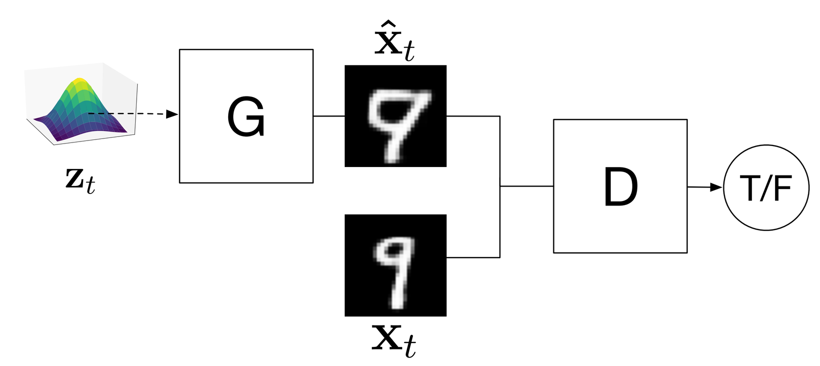

Figure 5: Discriminator learns to decide if a sample is true or fake, while Generator aims to fool the discriminator.

- VAE: Based on the principle of encoding-decoding, these models aim to minimize the reconstruction error (i.e: Mean squared error) while constraining the inner representation to be similar to a simple distribution (i.e: Gaussian).

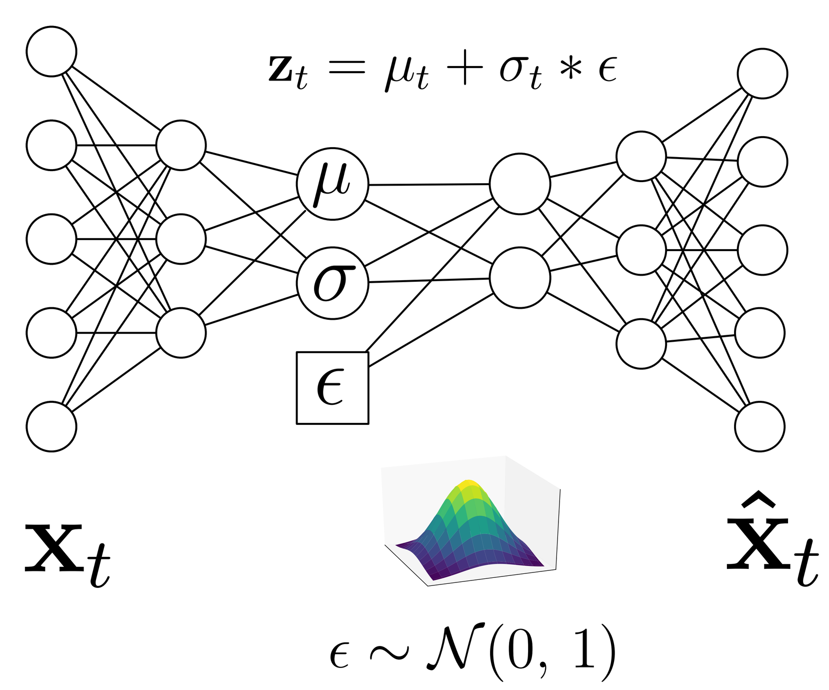

Figure 6: VAE architecture, the input $\mathbf{x}_t$ is projected in the manifold, obtaining the vector $\mathbf{z}_t$. The decoder recovers this vector getting the reconstructed input $\hat{\mathbf{x}}_t$.

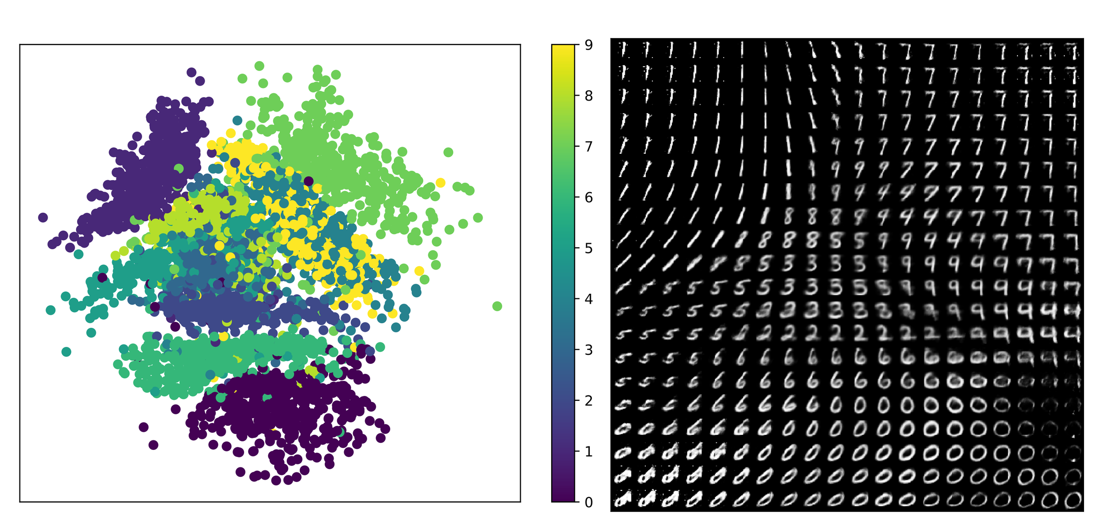

Figure 7: Left: 2D Latent space provided by the model, trained with MNIST dataset. Right: Sampling the 2D manifold and getting the images.

- We are going to use VAE as they can provide the manifold directly. With this manifold we can perform different modifications over the original data.

VAE for data augmentation

1. Train VAE with your training set and use a validation set in order to check the model's evolution.
2. Project your training set using the model, obtaining the projected versions $\mathbf{z}_t$ for each sample.
3. Alter these $\mathbf{z}_t$ vectors, and then reconstruct them.

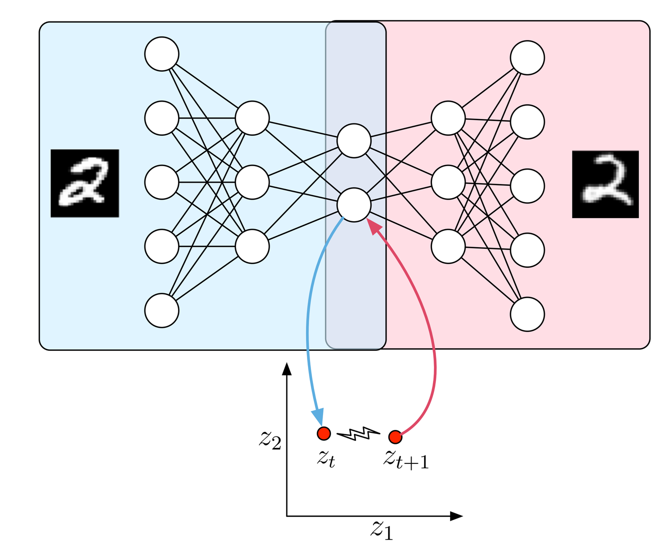

Figure 8: Encoding the input, modifying it and then reconstructing the altered sample with the decoder.

Different methods to alter the projected samples:

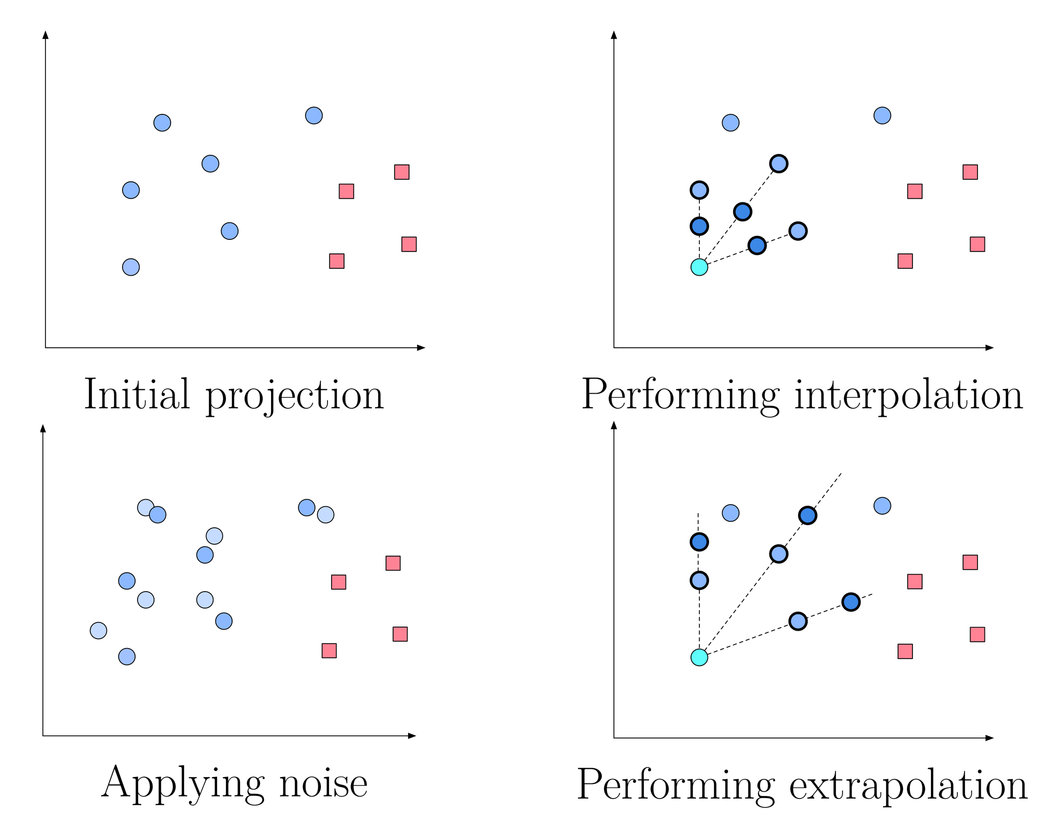

Figure 9: 2D example manifold's transformations.

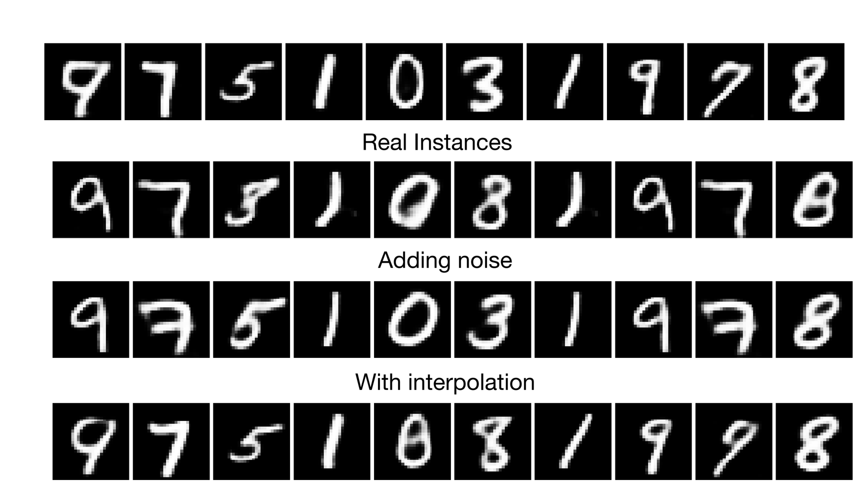

Figure 10: Original input and the different outputs depending on the modification, each column corresponds to the same digit.

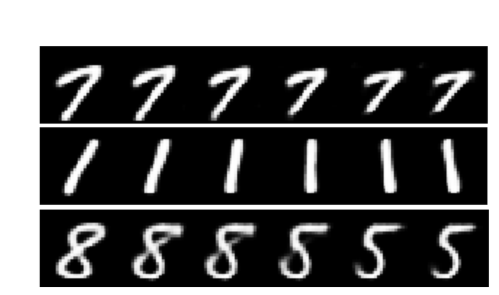

Figure 11: Smooth transformations that capture changes in the scale and rotation, even changes among classes.

Case use: UJIPEN Dataset

Labels	Dim.	Tr.	Val.	Ts.
97	4,9k (70x70 px)	5,820	582	4,656

Table 1: Dataset distribution.

Figure 12: Some samples from UJIPEN.

Classifier	Baseline	Generation methods		
		Noise	Interpolation	Extrapolation
Support vector machines	54.79	59.09	63.48	63.19
K-nearest neighbors	31.86	39.29	53.15	46.74
Neural Network	63.99	57.20	**65.60**	64.09

Table 2: UJIPEN results, accuracy on test set (%)[1].

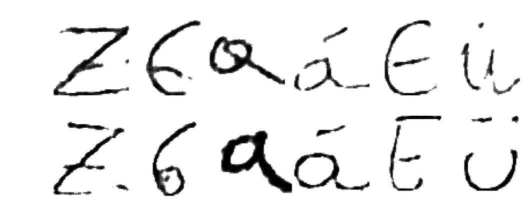

Figure 13: Top row: Samples after interpolating. Bottom row: Samples after extrapolating.

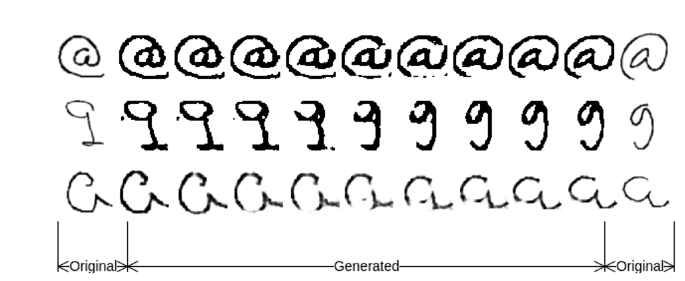

Figure 14: Linear interpolation between samples from the same class.

Conclusions

- We used a generative model, a VAE in particular, to create synthetic samples.
- We performed different alterations in the inner representation that helped the selected classifiers.
- For future work, we want to use this model in different data: text, speech, feature vectors, etc.

Python For Science: All the experiments were performed with:

- Numpy.
- TensorFlow.
- Matplotlib.

Reference

[1] J. Jorge, J. Vieco, R. Paredes, J. A. Sanchez, and J. M. Benedí. Empirical evaluation of variational autoencoders for data augmentation. In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP*, pages 96–104. INSTICC, SciTePress, 2018.