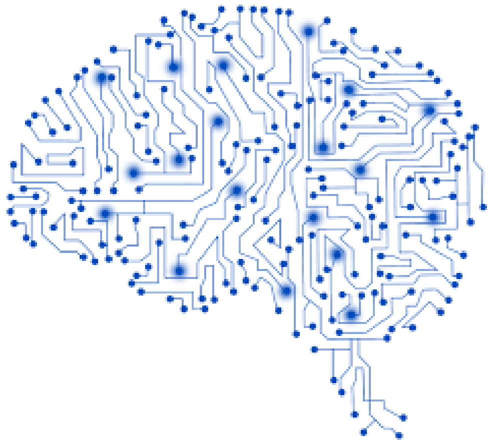
University of Minho
School of Engineering



Aprendizagem Profunda

CVAE,GANs,FCN

APP @ MEI/1º ano – 2º Semestre

Victor Alves

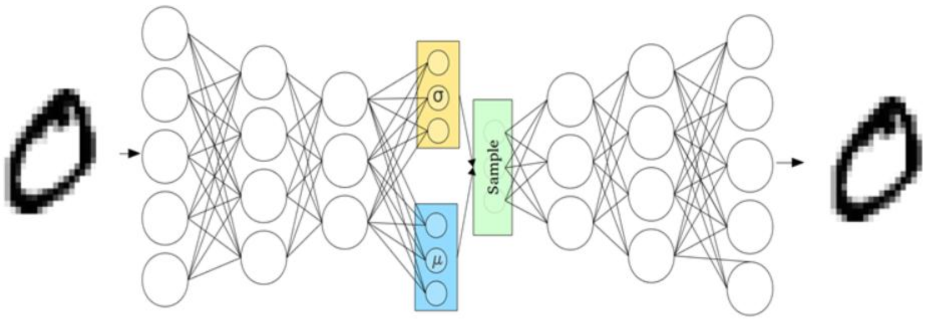
Part VIII

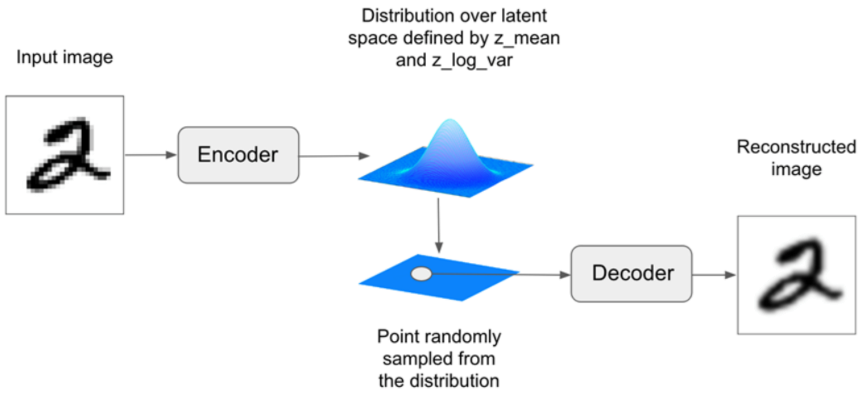
Contents

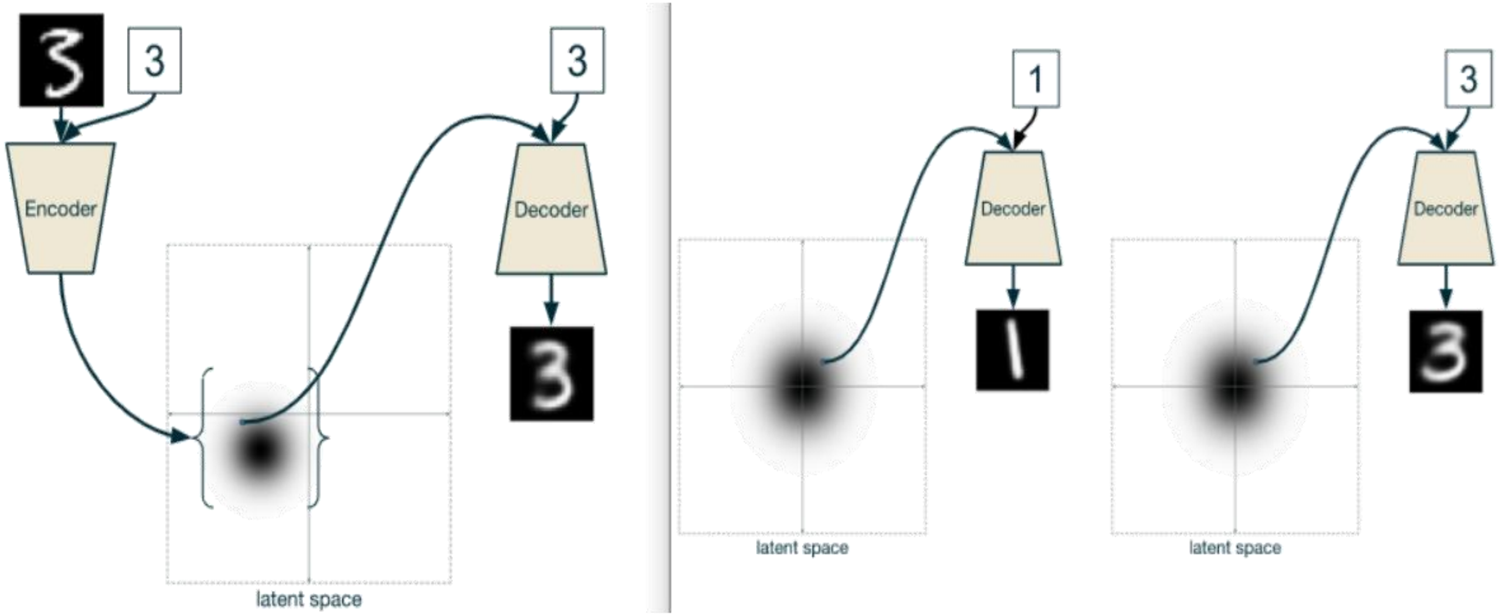
- CVAE
- GANs
- FCN

Laboratory Classes

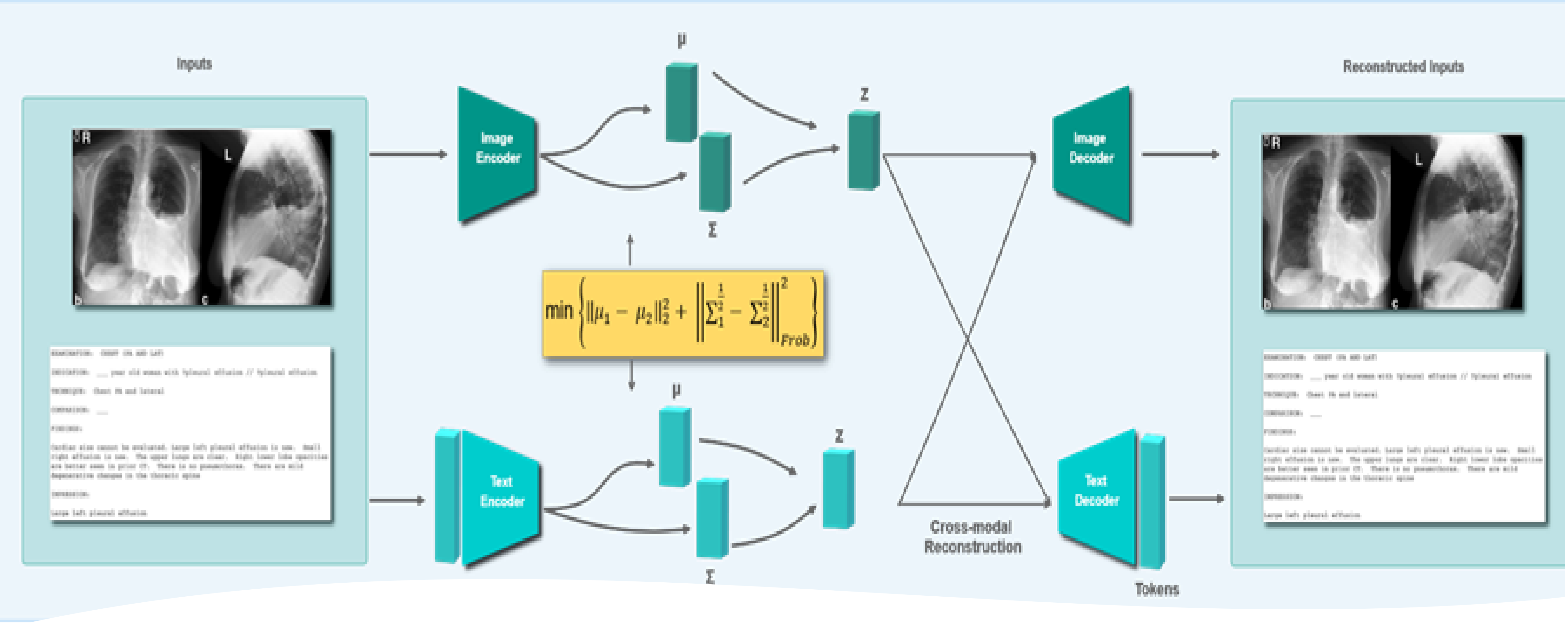
- Conditional Variational Autoencoder with MNIST dataset
  - 11_pyt_CVAE_MLP_treino_MNIST.ipynb
  - 12_pyt_CVAE_MLP_generate_MNIST.ipynb







Zero Shot Learning

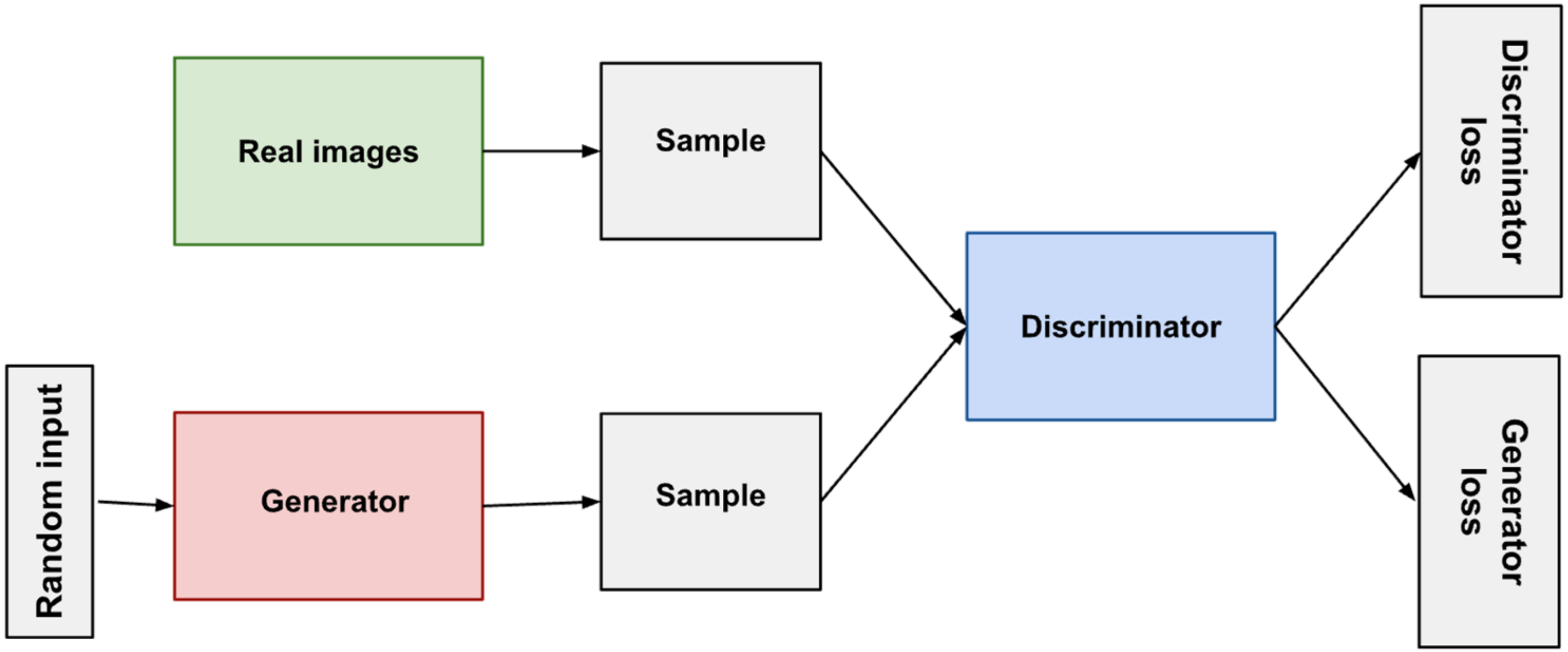


GAN - Generative Adversarial Networks

- Introduced in 2014 by Ian Goodfellow et al.
- They have been used essentially to **generate new images that are statistically similar to the original ones** (among other types of data)
- GAN includes **two models**: one that generates new examples and another that tries to discriminate whether examples are new or original (used in training), trained to optimise opposite (adversarial) functions (loss)
- An intuitive example would be a forger of paintings (or banknotes) showing his work to an art expert; as the forger gets better at creating better forgeries, the expert gets better at recognizing them

GAN - Generative Adversarial Networks

- **Generator** - takes as input a random vector (point generated in latent space), decoding it into a new image
- **Discriminator** (adversary) - receives as input an image (real or generated) and classifies it as original (from the training data set) or one generated by the Generator



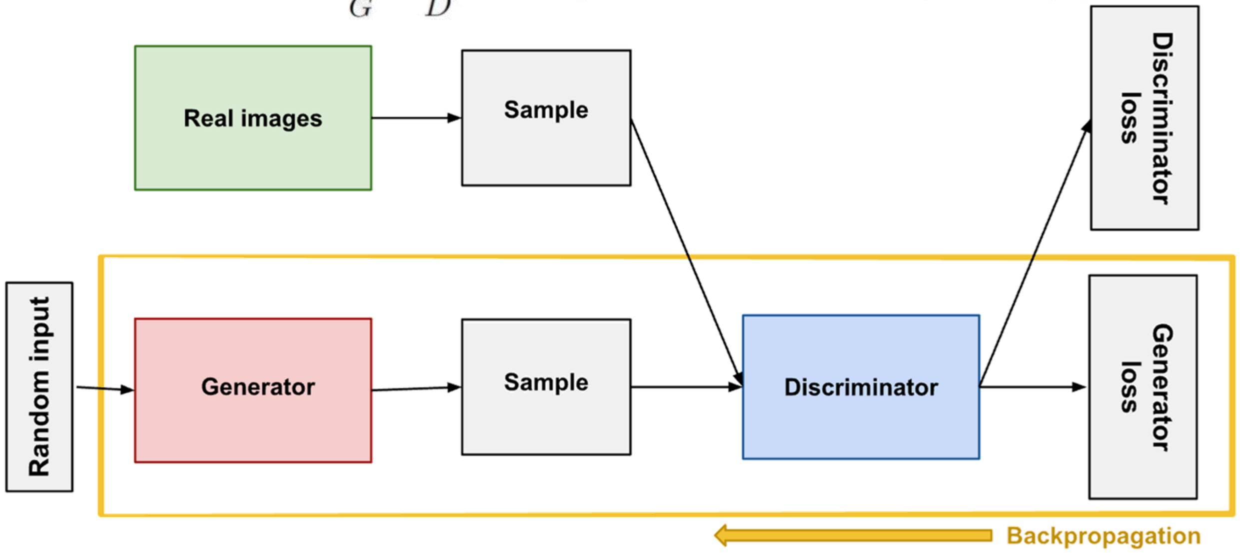
GAN - Generative Adversarial Networks

- **Generator** : receives a random value (noise) which it transforms into a new example (e.g. image); seeks to **minimise** the probability of the discriminator distinguishing the data it generates as being false.

$$\arg\min_{G} \ \mathbb{E}_{\mathbf{z},\mathbf{x}}[\ \log D(G(\mathbf{z})) + \log(1 - D(\mathbf{x}))\ ]$$

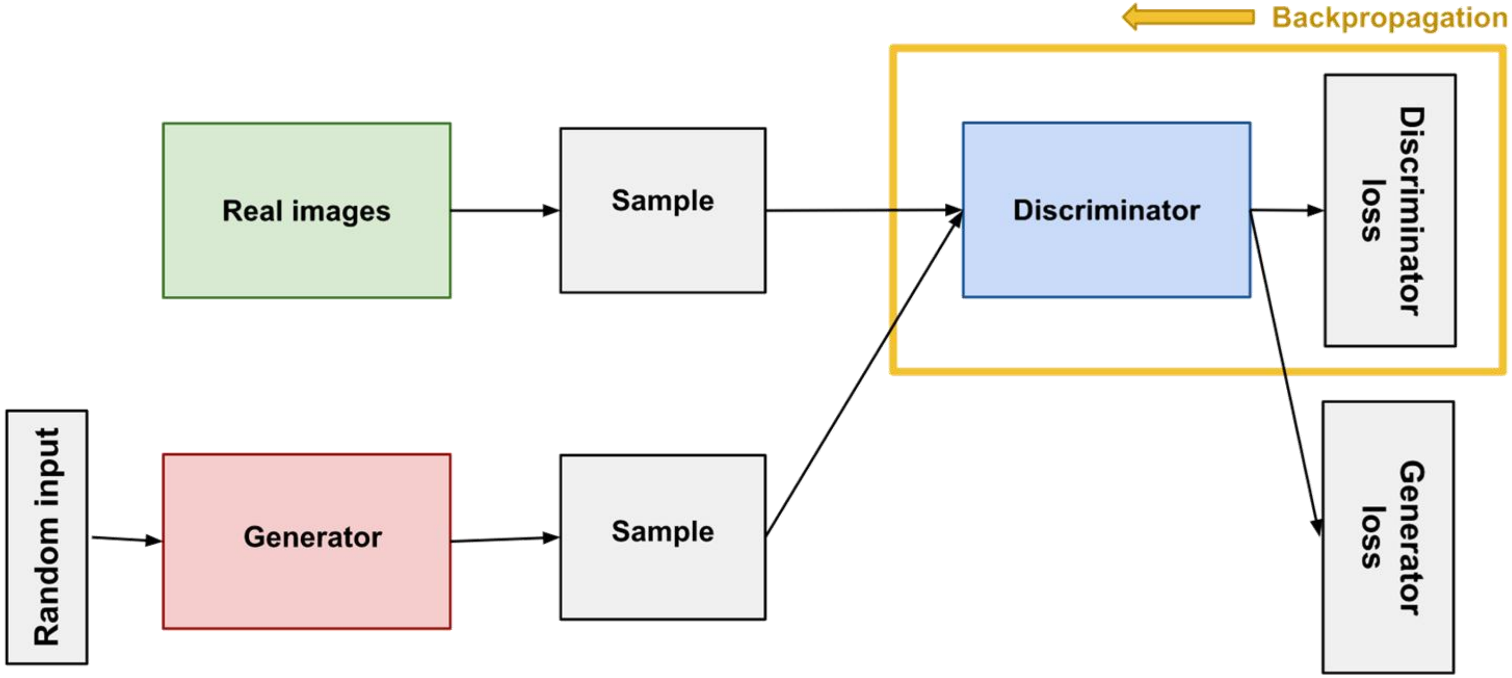
- The ultimate end of the generator is to generate false data that deceives the discriminator:

$$\arg\min_{G}\max_{D} \ \mathbb{E}_{\mathbf{z},\mathbf{x}}[\ \log D(G(\mathbf{z})) + \log(1 - D(\mathbf{x}))\ ]$$



GAN - Generative Adversarial Networks

- **Discriminator**: maximizes the probability of identifying false data as false.



$$\arg\max_{D} \mathbb{E}_{\mathbf{z},\mathbf{x}}[\underbrace{\log D(G(\mathbf{z}))}_{\textbf{Fake}} + \underbrace{\log(1 - D(\mathbf{x}))}_{\textbf{Real}}]$$

GAN - Generative Adversarial Networks

Training - must alternate between Generator and Discriminator training, one or more epochs each (while training one model, the other remains unchanged)

Generator training:

- Generate fake images
- Compute the discriminator loss on fake images
- Perform backpropagation + an optimization step to update the generator’s weights

Discriminator training:

- Compute the discriminator loss on real, training images
- Generate fake images
- Compute the discriminator loss on fake, generated images
- Add up the real and fake loss
- Perform backpropagation + an optimization step to update the discriminator’s weights

GAN - Generative Adversarial Networks

- **Convergence** is difficult to identify because the loss functions are opposite (adversarial) and therefore there is no absolute improvement
- Training of GANs is quite difficult in practice - several "**tricks**" needed, e.g:
  - Use tanh as the last activation in the generator, instead of sigmoid;
  - Sample latent space points using a normal distribution (Gaussian distribution) instead of a uniform distribution.
  - Introduce randomness by using dropout in the discriminator or by adding random noise to the discriminator labels.

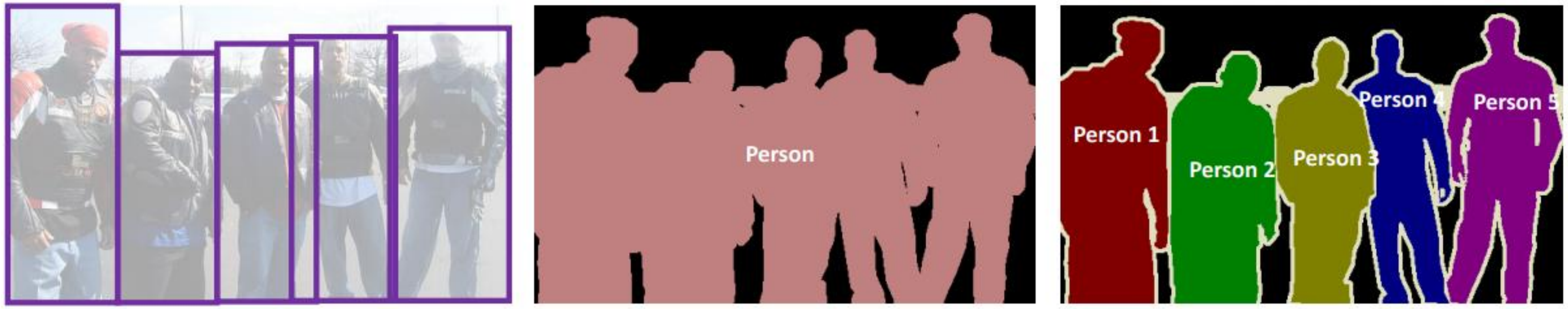
Other GANs / applications

- https://developers.google.com/machine-learning/gan/applications
- **Progressive GANs** – higher resolution images
- **Conditional GANs** – create constraints for the images to be generated
- Image to image translation
- Text to image synthesis
- Text to speech
- **Cycle GANs** – image processing

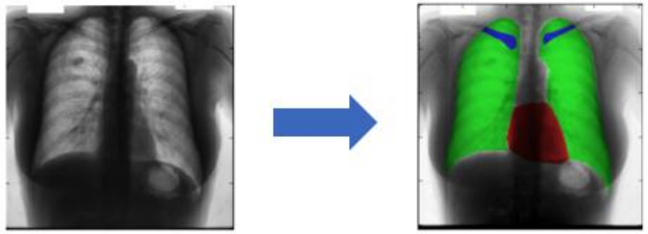
Laboratory Classes

- Generative Adversarial Networks with medMNIST dataset
  - 13_pyt_GAN_medNIST.ipynb

FCN - Fully Convolutional Network



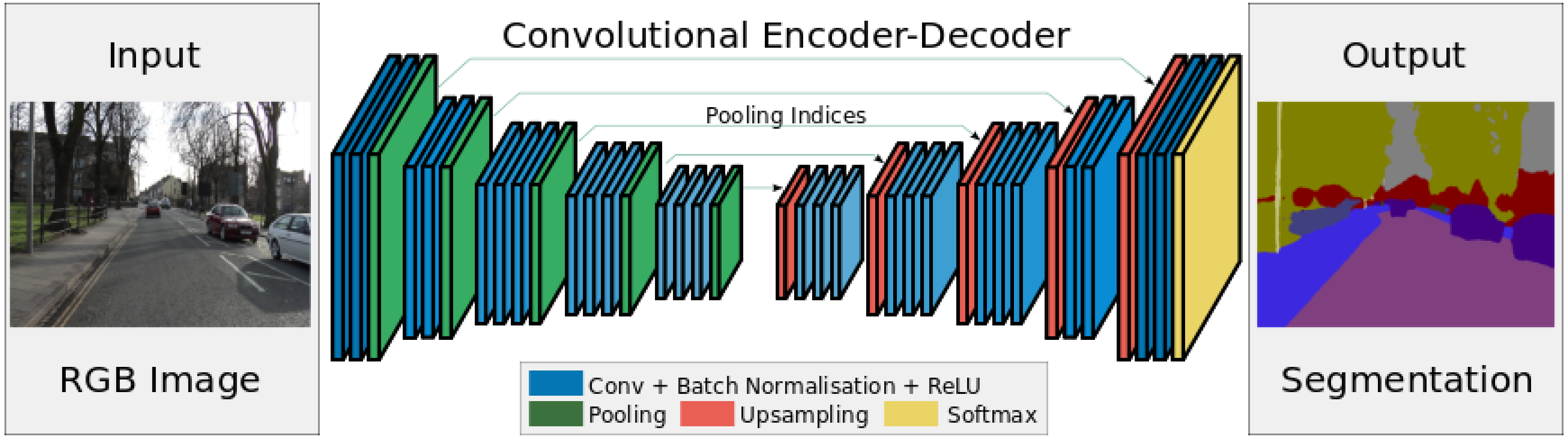
Object Detection | Semantic Segmentation | **Instance Segmentation**



Input Image | Segmented Image

FCN - Fully Convolutional Network

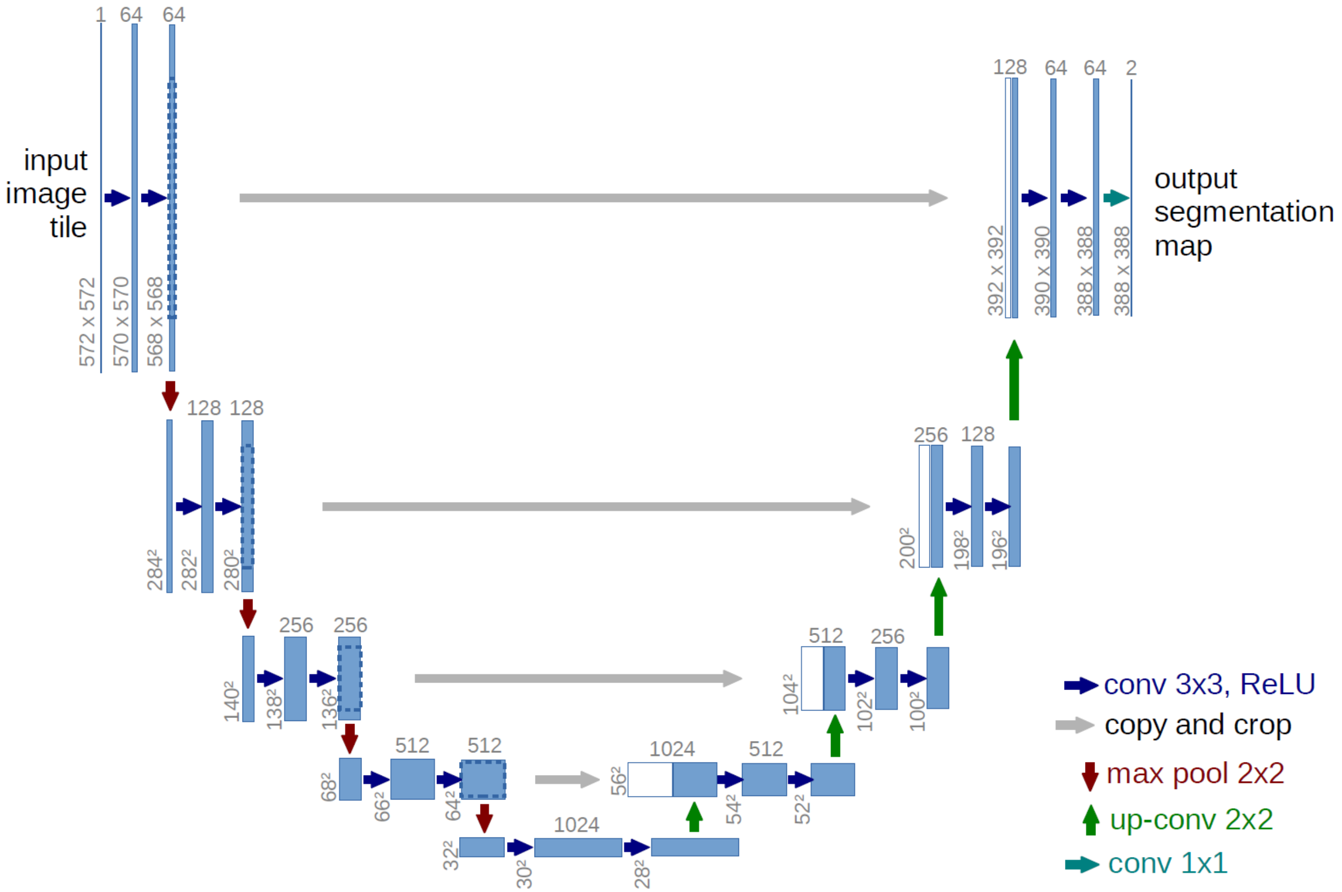
SegNet



https://arxiv.org/abs/1511.00561

FCN - Fully Convolutional Network

- **U-net**



https://arxiv.org/abs/1505.04597

Laboratory Classes

- Segmentation Exercise
  - 16.1_pyt_exercicio_segmentacao.ipynb

In this exercise we will segment the left ventricle of the heart in relatively small images using neural nets.

The code for a segmentation net and its training is presented. The network is not very good, so t**he exercise is to improve the quality of segmentation by improving the network and/or the training scheme, including data loading efficiency and data augmentation**.

The data used here are derived from Sunnybrook Cardiac Dataset cardiac MR images, filtered to contain only segmentations of the left ventricular myocardium and reduced in XY dimensions.

Data extracted from:
https://github.com/ericspod/VPHSummerSchool2019/raw/master/scd_lvsegs.npz

Occlusion

Sensitivity to Occlusion

One method for trying to visualise why the net made a particular prediction is occlusion sensitivity. We occlude part of the image and see how the probability of a particular prediction changes. We then iterate over the image, moving the occluded part as we go, and in doing so build a sensitivity map detailing which areas were the most important in the decision making.