Clustering with a Convolutional Variational Autoencoders of Photoplethysmography Time-Series

Author: Lorenzo Barsotti

Abstract

The purpose of this project is to analyze photoplethysmography data, with a Convolutional Variational Auto-encoder Neural Network. In particular we would like to see if there are some *latent characteristics* that allow us to discriminate the biological age of a patient from its photoplethysmogram.

1 Introduction

The purpose of this project is to try to estimate the biological age of a patient, from its photoplethysmogram. A photoplethysmogram is the optically obtained signal related to the changes of volume in the blood vessels (section 1.1). In order to perform the age estimation I used a database composed of about 4770 photoplethysmograms of as many patients, and a *Convolutional Variational Auto-Encoder*. This is a particular case of *Deep Neural Network* as it will be described in the section 3. To develop the Neural Network Tensorflow 2.9.0 [1] was used.

1.1 Photoplethysmography

Photoplethysmography is a technique that use light in order to detect changes in volumes in blood vascular vessels. A *Light Emitting Diode* (LED), illuminates a high vascularized body part covered by a thin layer of skin, usually a fingertip, and a photodiode measures the amount of light transmitted or reflected. In the particular case of this project, the data were taken with camera module of smartphones: the LED illuminates a fingertip and the camera records the backscattered light. A *photoplethysmogram* (PPG)is the plot of the recorded data of the photodiode. It is a graph in which we are able to see how of the blood vessels' volume change over time. Ideally the obtained time series is very regular and the present peaks corresponds to patient's heartbeat (Figure 1).

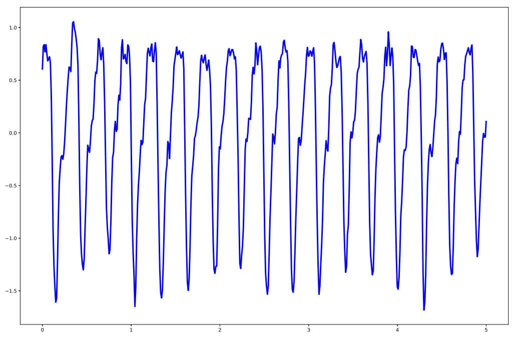

Figure 1: Example of a photoplethysmogram present in the database. The figure shows only the first part of the plot, that is much larger.

2 The database

The database contains data coming from about 3600 patients. A PPG is saved in two arrays of the same length: the first one contains the signal, and the second one the time at which each signal of the first array has been recorded. Two other important features used in this project and contained in the database, are the patient's age and a score related to the PPG quality.

The other features of the database have not been used.

2.1 Preprocessing

The signals of the PPGs may not be always perfect: there could be some *strange* peaks, or a very different behavior respect to the one in figure 1. This could be due to different factors, such as the pressure the patient push the finger with on the camera, or the position of the finger on the camera... These not *well-behaving* PPGs must be removed from the database and, for this operation, the quality index is used. In this database lower the quality index is, the better the PPG is. For this reason I set the a quality threshold at 0.005, over the which the data are removed from the database. Doing this led to a number of usable PPGs to 3256. A plate PPG has been removed manually because the quality threshold was not sufficeint to discriminate this particular PPG.

come sono presentati i dati

3 Convolutional Variational Auto-Encoders

Convolutional Variational Auto-Encorders (CVAE) is an architecture of *Deep Neural Network* (DNN). It can be used with different purposes, but in this project it has been used to perform an *unsupervised clustering*. This means that given the data, we would like to divide them in clusters depending on the age of the patients, without providing the corresponding labels during the training phase of the Neural Network.

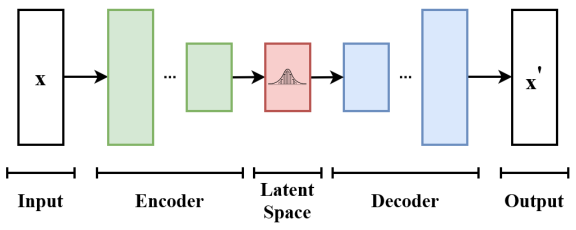

Figure 2: Structure of a VAE. It is possible to distinguish the *encoder* that, in the particular case of a CVAE, performs convolution operations, the *latent space*, in which the important information should be contained, and the *decoder*, that try to reconstruct the output starting from the latent space and provide us the Neural Network output.

A *Variational Auto-Encoders* (VAE) can be represented by the scheme in Figure 2, where we can see that it is composed by several *layers*. The data are provided to the input layer in the form of tensor, with a shape eqaul to (number of inputs, input height, input width, number of channel). This initial layer pass the data to the *encoder*. The *encoder* reduce the original information to simple distributions that should contain the same information of the original data (*latent space*). Then a decoder is applied to the *latent space* in order to reconstruct an output as much as similar to the input data.

A CVAE is a VAE in which, the encoder consist in perform convolution operations. A convolutional layer applies convolutions on the input and pass the result to the next layer. If the input of a convolutional layer are images,so if we consider a bi-dimensional input, the operation of convolution consist in bring together the information of a small number of pixel in an unique pixel. The pixel on which the convolution is performed, is determined by the *kernel matrix*. Usually the *kernel* is a square matrix with an odd-number of pixel per side. All the pixel in the original image are convoluted, with a *kernel matrix* in a new single pixel. Once the boundary condition (*padding*) and the *stride* [1] are fixed, the process must be repeated for every pixel of the original image. An example can be seen in the Figure 3. The *encoder*, encodes the original image in a Gaussian distribution, so its output is a vector, containing the mean and the variance of the distribution. So with a bi-dimensional input, we would have a latent space composed by two vectors of this type, one for each dimension.

[1]The stride parameter fixes how often we have to perform a convolution with the *kernel matrix* respect to the length of the image. if the stride is 1, than the I repeat the convolution moving the source matrix of 1 pixel every time, if it is 2, I move the source matrix of two pixel, ...

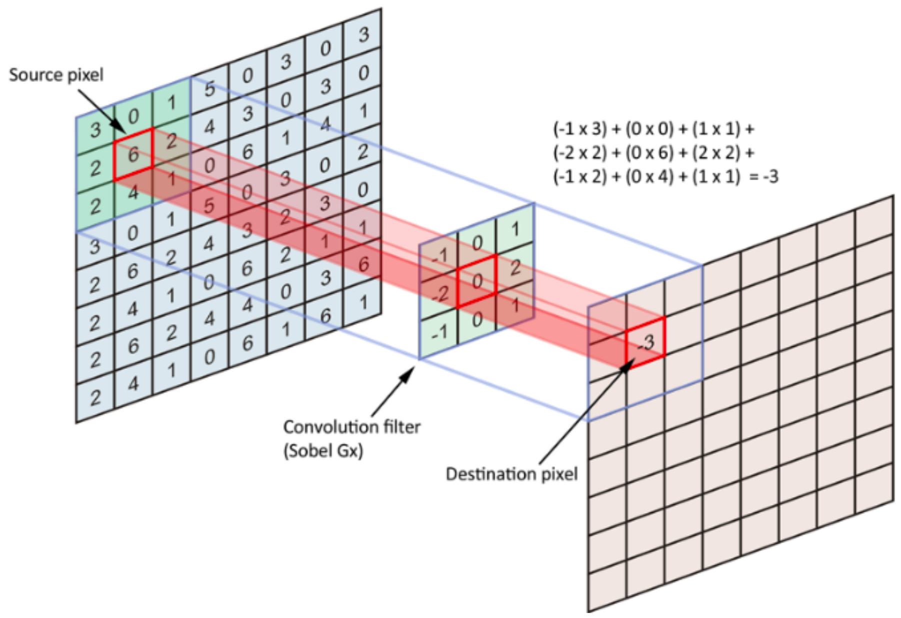
Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
(Sobel Gx)
Destination pixel

Figure 3: Example of how a *kernel* matrix convolution works. The source matrix is convoluted with the *kernel* matrix. This lead to a number, that become the output value of a pixel

References

[1] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.