Self-supervised learning

Lesson No. 7

Patrick de Carvalho Tavares Rezende Ferreira
175480

1 Introduction

Passing a dataset with sufficient inputs and responses to a network, it will be able to learn what we want it to do in a process called supervised learning. However, a dataset with enough data often requires a few tens of thousands of samples and labeling all of them can be a slow and expensive process. There is a vastly greater amount of unlabeled data on the internet than any dataset already labeled, and it is desirable we can somehow use the information present in that unlabeled data.

Unsupervised learning is, however, a difficult process that does not always achieve the performance of supervised learning. Self-supervised learning is a proposed solution to this problem and can be seen as a form of supervised learning that produces the dataset itself following some logic, and can go through a brief adjustment process with only a few samples labeled at the end of the training, which is considerably less expensive than manually labeling an entire dataset. Self-supervised learning is based on learning the intrinsic information present in the data being entered.

2 Examples of self-supervised learning techniques

Here we will cover some examples of techniques known to achieve good results through self-supervised learning. Some of them are relatively old and known (but still effective) like autoencoders, and others are recent and also powerful like SimCLR's [1] contrasting learning.

2.1 Denoising Autoencoders

The classic architecture of an autoencoder consists of a pair of MLPs (Multilayer perceptron) that receive the input information, reduce its dimensionality in a latent variable space called a bottleneck and convert it back to the original space with the second MLP.

A denoising autoencoder (figure 1) is trained to receive an input corrupted by various types of noise and return it treated, with the highest rate of reconstruction of the input before the possible noise.

The first MLP used is known as the Encoder. Its role is to map in a space of characteristics and efficiently all the content of the entry, in such a way that it is possible to reconstruct the original data. This is where the dimensionality reduction takes place and will be discussed later on how it can be used.

The bottleneck is made up of the Encoder outputs and consists of latent variables that, after successful training, make up a space of characteristics of the original input. These variables can be used for compression, or to obtain important characteristics of the original entry such as, for example in the case of images, to evaluate the gender of the people in the photo and even their age. It is also possible to make changes to the original image using these variables.

The Decoder of this structure is the second MLP and its function is to map the coded latent variables back to the original space. It is like him that the self-supervised trained person performs, as there is no data labeling at any time, the latent variables being evaluated according to their quality according to the reconstruction capacity that the decoder has concerning the original image.

However, the data entered in the autoencoder during training is not the original data (X), but the original data corrupted by varying noise, as shown in the equation 1,

$$\tilde{X} = X + \text{ noise} \tag{1}$$

where X is the original data entry and $\tilde{X}$ is corrupted with noise. This allows you to create a structure that ignores noise from the input image and learns to remove them, reconstructing the original image.

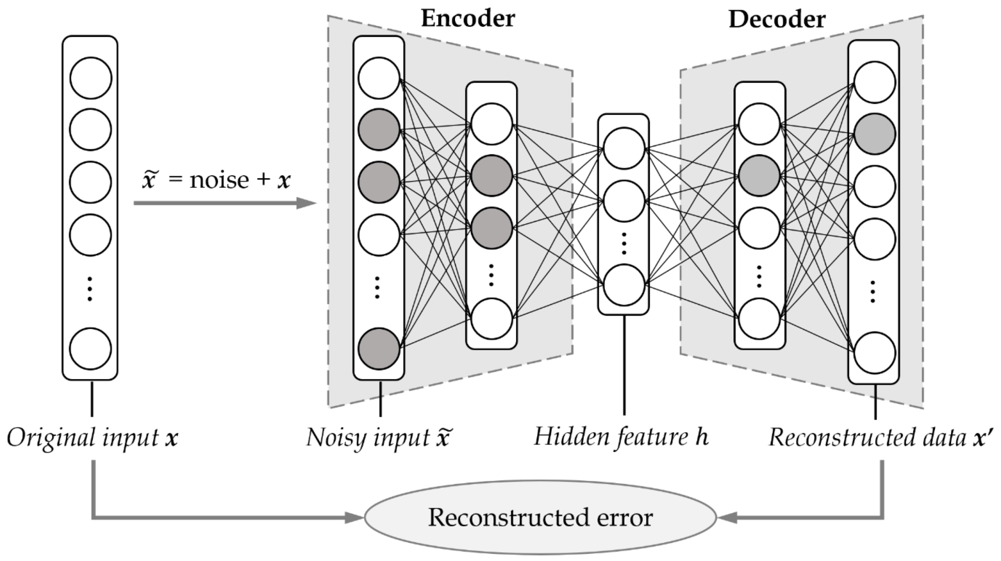

Figure 1. Denoising Autoencoders schematic [2].

2.1.1 Context Encoders

This approach consists of making a pixel prediction based on the context of the image around them, also called image inpainting. Here we deal with trained encoders to generate an image with a certain fill, with the original image having an arbitrary region removed, and the output will be an image with that same region filled by something that makes sense according to the surrounding context. .

This is a very useful process for images that have lost some information, for editing images when you want to remove an object or someone, and even for old images that have had parts damaged over time.

The authors claim to have performed the training of the network using standard pixel-wise reconstruction loss and also with reconstruction plus an adversarial loss, with the second form producing better results, since it can better handle multiple modes in the output.

2.1.2 Stacked Autoencoders

A final application for this model is for training for deep nets. The process consists of training encoders so that the information from the data is previously extracted in a pretraining process focused only on finding relevant information.

With the latent variables of 1 or more stacked encoders, it becomes easier to obtain successful training when using a large neural network that receives this information as input instead of raw data.

2.2 Contrastive Learning

As an example, we have SimCLR, which is a generic framework for contrasting learning. It uses an unsupervised training system that does not require labeled data and has a customized loss (NT-Xent, equation 2) that facilitates such a task. At the end of the training, a fine-tuning process with little labeled data allows you to perform image classification.

$$l_{i,j} = -\log \frac{\exp(\mathrm{sim}(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{k \neq i} * \exp(\mathrm{sim}(z_i, z_k)/\tau)} \tag{2}$$

It is based on building a convolutional feature extractor - which in the original paper is a ResNet [3] - to obtain an encoder that extracts latent variables into a feature space from the input data. Then, we add an MLP over this layer to be able to make important achievements about the characteristics present in the input data and compare it with the other images being worked on (shown in figure 2).

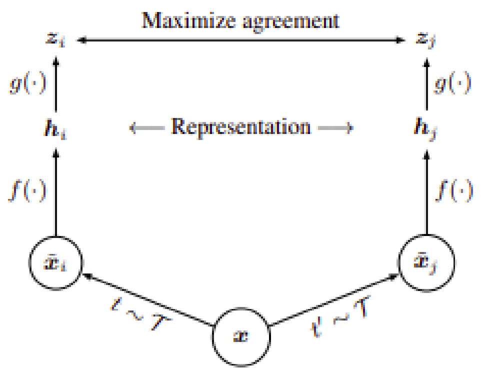

Figure 2. SimCLR proposed structure [1].

How this framework performs unsupervised learning and learns the similarities between the images is to minimize the difference in output for two equivalent images corrupted by different types of distortions, in a process called data augmentation.

After this process of contrasting training, a brief fine-tuning process with labeled data (less than 5 % of the original dataset) can adjust a linear classifier working with the outputs of the convolutional stage of this structure. The results obtained with this network are close to those that consist of the state of the art in supervised learning.

2.2.1 Data Augmentation

Bearing in mind that we are dealing with non-rotated images and we want the network to learn the similarity between equivalent images in a way that the script knows that it is dealing with equivalent images, one way we can do this is the data augmentation process.

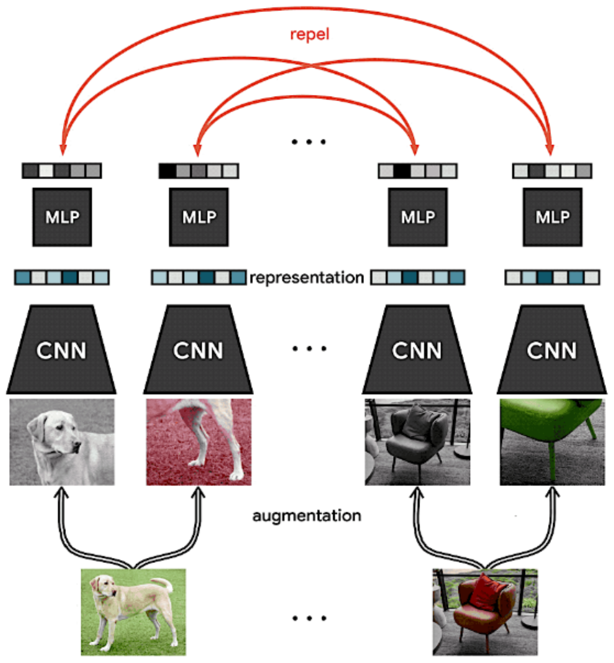

Figure 3. Data augmentation comparison [1].

For this, we take the original dataset with the unlabeled images and double the number of images, corroding them at least in terms of the color spectrum and different cutouts. With that, we will have images with equivalent characteristics, but that the network will not be able to identify as being the same from its shape and its color, it will have to learn to identify the characteristics present in the image, extract these attributes and compare to see if make the equivalent of the image (figure 3).

References

[1] *[2002.05709] A Simple Framework for Contrastive Learning of Visual Representations.* [Online]. Available: https://arxiv.org/abs/2002.05709 (visited on 06/21/2020).

[2] S. Park, M.-S. Gil, H. Im, and Y.-S. Moon, "Measurement Noise Recommendation for Efficient Kalman Filtering over a Large Amount of Sensor Data," en, *Sensors*, vol. 19, no. 5, p. 1168, Jan. 2019, Number: 5 Publisher: Multidisciplinary Digital Publishing Institute. DOI: 10.3390/s19051168. [Online]. Available: https://www.mdpi.com/1424-8220/19/5/1168 (visited on 07/04/2020).

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv:1512.03385 [cs]*, Dec. 2015, arXiv: 1512.03385. [Online]. Available: http://arxiv.org/abs/1512.03385 (visited on 06/21/2020).