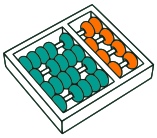

Self-supervised learning

Lesson No. 7

Levy Gurgel Chaves [*]

1 Introduction

Self-supervised learning can also be an autonomous form of supervised learning because it does not require human input in the form of data labeling. Another definition could be: a form of unsupervised learning where the data provides the supervision. In contrast to unsupervised learning, self-supervised learning does not focus on clustering that is commonly associated with unsupervised learning. In self-supervised learning, the model is pretrained through an auxiliary task exploring an intrinsic property from the current dataset. The task that we use for pretraining is known as the "pretext task". The tasks that we then use for fine tuning are known as the "downstream tasks".

The most important question that needs to be answered in order to use self-supervised learning in computer vision is: "what pretext task should you use?" It turns out that there are many you can choose from, for example, predicting colors, rotation angle. Here is a list of a few, and papers describing them, along with an image from a paper in each section showing the approach.

2 Denoising Autoencoders (DAE) [1]

Since the vanilla autoencoder learns the identity function given, the input of denoising autoencoder is partially corrupted by adding noise or masking some values of the input. Then the model is trained to recover the original input. This way, we can train the autoencoder to learn how to remove noise from pictures or signals, for example.

Autoencoders, in general, usually consist of four main parts, which are:

1. **Encoder:** genereally composed by some Convolutional/Fully connected neural networks in which the model learns how to encode (or reduce) the input dimension and compress the data into an encoded representation.
2. **Bottleneck** this layer contains the latent and compressed representation of the data given as input. The representation is usually in a much lower dimension space than the original one.
3. **Decoder:** following the inverse idea in the encoder, here the model learns how to reconstruct the data from the latent representation generated by the encoder.
4. **Loss function:** this part guides the learning of the network. This method measures how well the decoder is performing and how close to the desired output the reconstruction is. The model is trained to minimize the network's loss function.

A schema comprising the basic structures can be seen in Figure 1.
In DAE Case, the reconstruction loss is given by:

$$Loss = \frac{1}{N}\sum_{j=1}^{N} l(x^j, \hat{x}^j) \tag{1}$$

where x is the original imagem and $\hat{x}$ is the reconstructed image by the decoder.
If the input is categorical, we could use Cross-Entropy loss to calculate per sample loss, which is given by:

[*]RA: 264958 - l264958@dac.unicamp.br

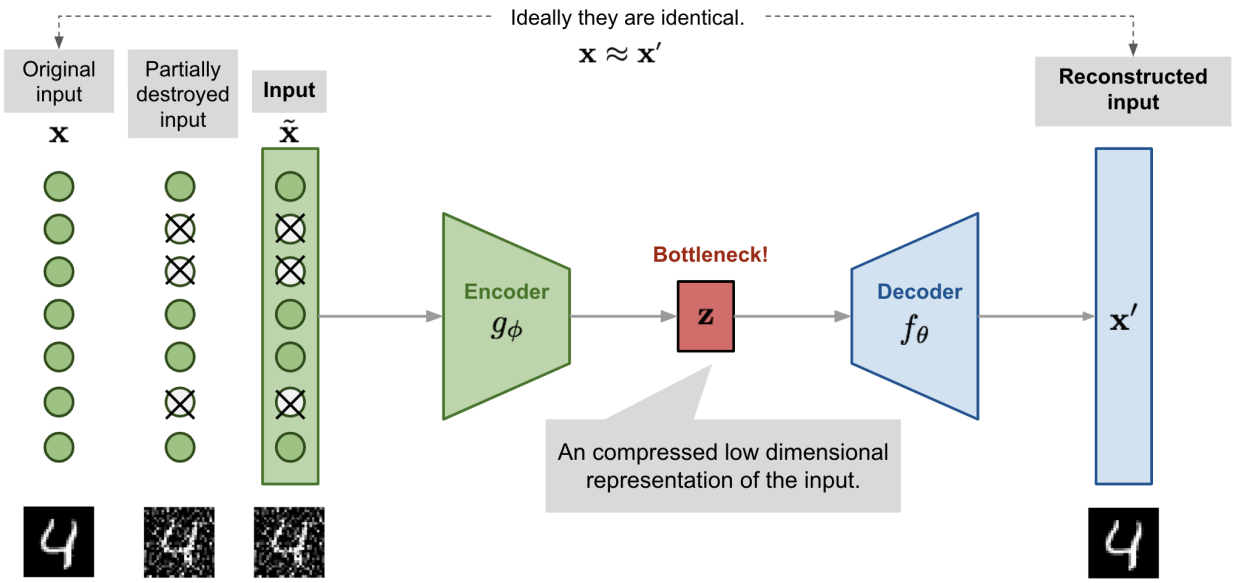

Figure 1. Denoising Autoencoder Architecture [2]

$$l(x,\hat{x}) = -\sum_{i=1}^{N}\left[x_i \log(\hat{x}_i) + (1-x_i)\log(1-\hat{x}_i)\right] \tag{2}$$

and when the input is a real value, we may possibly Mean Squared Error as loss function.

3 Context Encoder

The Context encoder [3] uses as pretext task to fill in a missing piece in the image on the basis of its surroundings – i.e. a context encoder takes in the surrounding data of the image region and tries to generate something that would fit into the image region.

The context encoder here consists of an encoder capturing the context of an image into a compact latent feature representation and a decoder which uses that representation to produce the missing image content.

In order to remove a certain region of context in a image, the authors used the idea of binary masks. In a binary mask B, where the value of pixel is 1 the pixel will maintain the original color, otherwise all pixels values in all image channels are set to zero. They defined three kinds of regions, which are:

- Central region: just set a central square patch as zero.
- Random block: the masking process is randomized, instead of choosing a single square patch as mask, a number of overlapping square mask are set up to take 1/4 of the image.
- Random region: generally uses a annotated mask as ground-truth to remove the content.

In order to train the model, we have to define a context-encoder part. The final loss function is divided into two main parts:

1. Reconstruction loss: the reconstruction loss used is a L2 norm loss function. The authors state that it helps to capture the overall structure of the missing regions and its context.

$$loss_{reconst}(x) = ||(1-B)\odot(x - E(B\odot x))||_2^2 \tag{3}$$

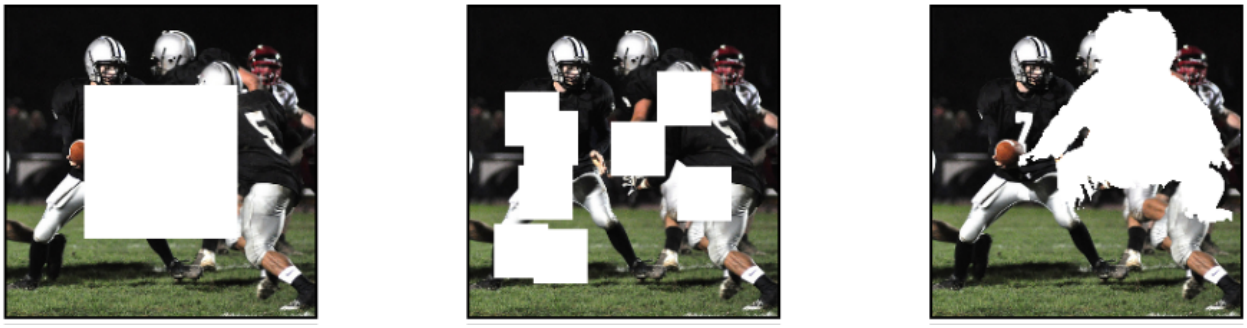

Figure 2. From left to right: 1) Central region mask; 2) random block mask; 3) random region mask

2. Adversarial loss: this loss tries to introduce more realism in the predictions and gave better results rather than only using L2-norm. This loss was introduced because L2 norm would give a blurry image, because blurry images reduces the mean pixel-wise error, which is not the effect the we want in the final output.

$$loss_{adv}(x) = \max_{D} E_x \left[\log D(x) + \log(1 - D(E(B \odot x)))\right] \tag{4}$$

where $\odot$ is the application of the given mask B. Final loss is just just both losses combined:

$$loss(x) = \lambda_r loss_{reconst}(x) + \lambda_a loss_{adv} \tag{5}$$

where λ_r and λ_a are hyperparameters. An overview of context encoders framework can be seen in the figure 3.

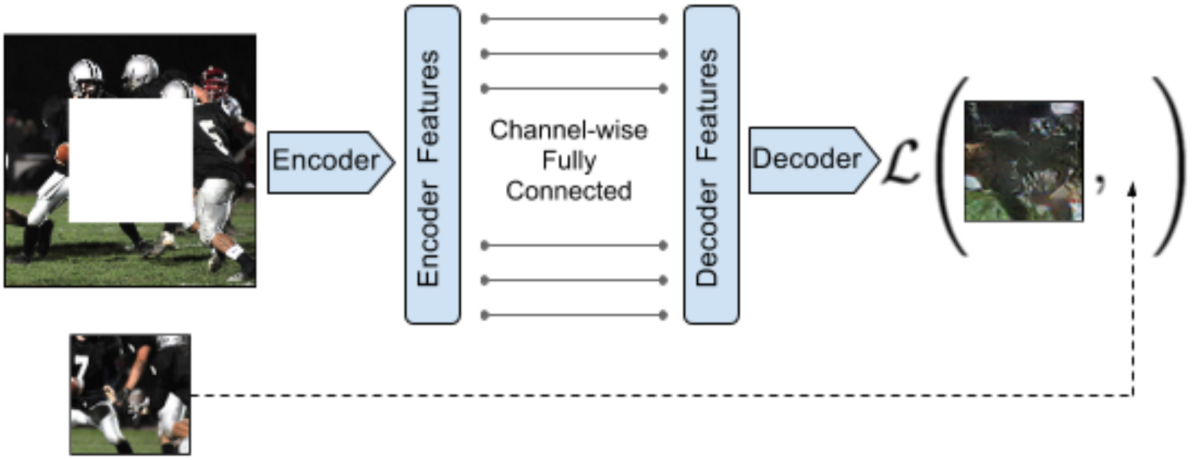

Figure 3. Context encoders framework

4 Colorization

As simple as it is, colorization can also be used as pretext task. Image colorization is a task of predicting a plausible color version of the image given a gray-scale image as input. A straight forward approach would be to use a convolutional neural network which consists of an encoder for feature extraction and a decoder for color prediction.

The network can be optimized using L2 loss between the gray-scale in its original color scheme and the output given by the decoder.

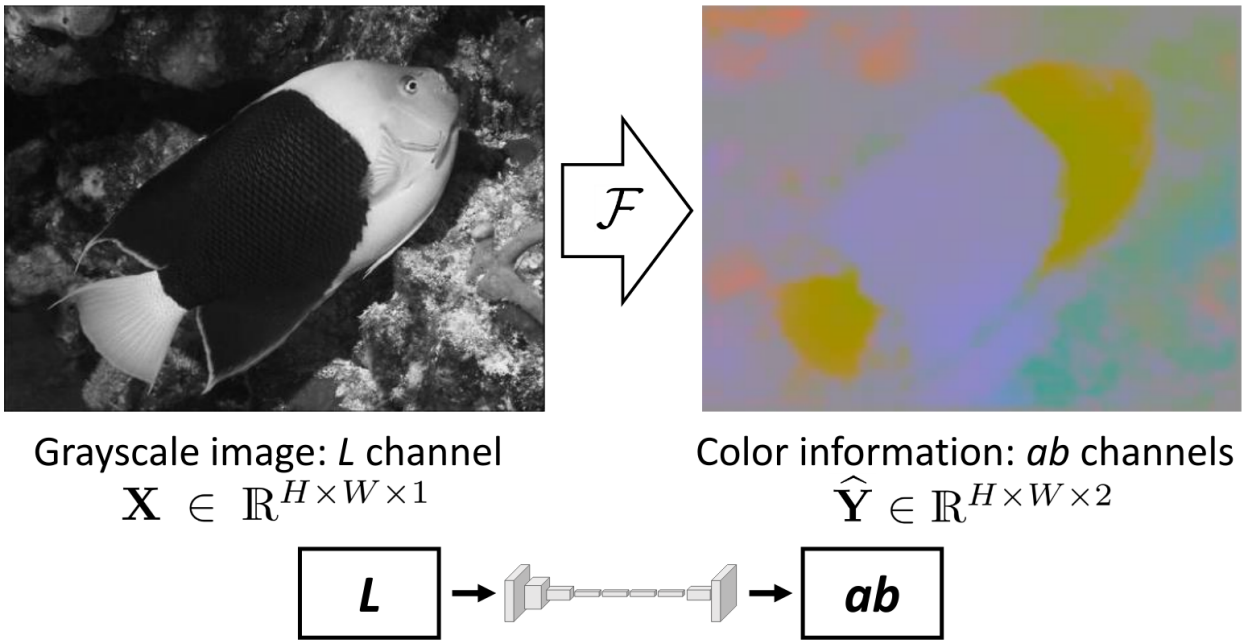

Figure 4. Illustration of method proposed by [4]

For example, in [4], the model outputs colors in the *CIE Lab* color space*. Due to the large search color space and multimodal nature of the colorization problem, cross-entropy loss of predicted probability distribution over binned color values works better than L2 loss of the raw color values. So, the authors model the loss function as follows: for every pixel $X_{h,w}$ of an output image $\hat{X}$ we can find the nearest **ab** bin and represent it as a one-hot encoding vector, in which we assign 1 to nearest **ab** bin an 0 to all other bins (Q). Mathematically:

$$Loss(\hat{X}, X) = -\frac{1}{HW} \sum_{h,w} \sum_{q}^{Q} X_{h,w,q} \log(\hat{X}_{h,w,q}) \tag{6}$$

Using the above loss function, unfortunately, the authors state that the model produces very dull colors.

To fix the loss function and make the algorithm to produce vibrant colors, the authors changed the loss function to:

$$Loss(\hat{X}, X) = -\frac{1}{HW} \sum_{h,w} v(X_{h,w}) \sum_{q}^{Q} X_{h,w,q} \log(\hat{X}_{h,w,q}) \tag{7}$$

where they introduced the color rebalancing term, $v(.)$ which is used to rebalance the loss based on the rarity of the color class. This contributes towards getting more vibrant and saturated colors in the output.

5 SimCLR [5]

SimCLR is a simple framework that first learns representations of images on an unlabeled dataset, and so it may be fine-tuned with a little amount of labeled images to get good performance for a given classification task. The representations are learned by simultaneously maximizing agreement between differently views – i.e. augmentations – of the identical image and minimizing agreement between transformed views of various images, following a contrastive learning. Updating the parameters of a neural network using this contrastive objective causes representations of corresponding views to “attract” one another, while representations of non-corresponding views “repel” each other [5].

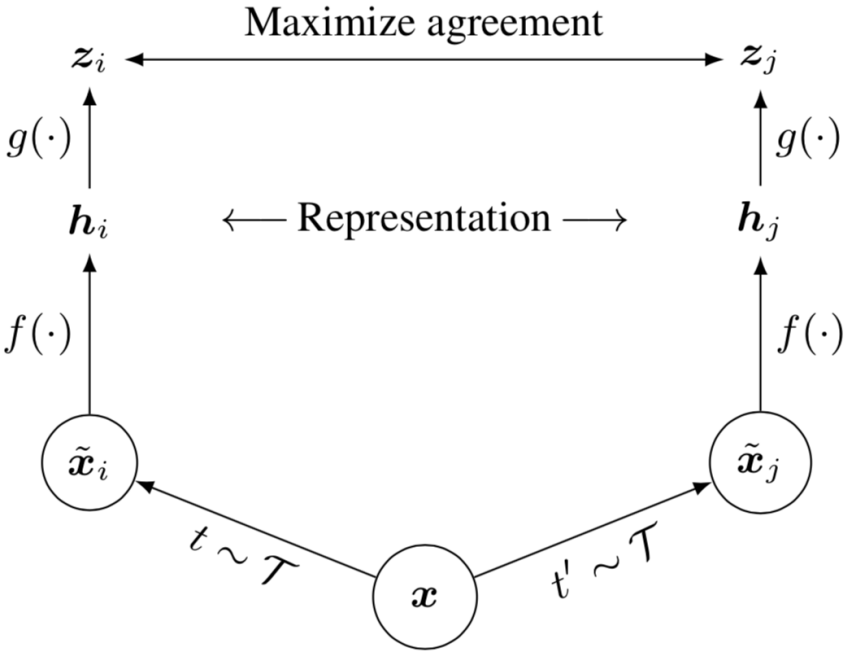

Figure 5. SimCLR process

The first component is data augmentation. It is a module that transforms any given data example randomly resulting in two correlated views of the same example, denoted $\tilde{x}_i$ and $\tilde{x}_j$, which it is considered as a positive pair.

The two augmentations are sampled from a set of augmentations (random cropping, random color distortion, and Gaussian blur). The idea behind these augmentations transformations of individual images is to encourage "consistent" representation of the same image under transformations.

SimCLR then computes the corresponding representation using a ResNet variant architecture. After that, SimCLR computes a non-linear projection of the image representation using a fully-connected network, which amplifies the invariant features and maximizes the ability of the network to identify different transformations of the same image. They use stochastic gradient descent to update both CNN and MLP in order to minimize the loss function of the contrastive objective. To train the network, SimCLR uses the contrastive loss function defined for a contrastive prediction task.

Given a set $\{\tilde{x}_k\}$ including a positive pair of examples $\tilde{x}_i$ and $\tilde{x}_j$, the contrastive prediction task aims to identify $\tilde{x}_j$ in $\{\tilde{x}_k\}_{k\neq i}$ for a given $\tilde{x}_i$.

To do so, we have firstly to define the matrix S as:

$$S_{i,j} = sim(z_i, z_j) = \frac{z_i^\top z_j}{||z_i|| \cdot ||z_j||} \tag{8}$$

Where z_i is the generated projection head for $i \in \{1, 2, ..., 2N\}$ (N is the batch size of images), and the similarity measure that is applied is the *cosine similarity*.

With that, we define L, using Equation 9, where τ is the temperature parameter and $\mathbb{I}_{[k\neq i]}$ is the indicator function equal to 1 when $k \neq i$ and 0 otherwise. The matrix is considered as a loss function between two vectors z.

$$L_{i,j} = -log\left(\frac{exp(S_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{I}_{[k\neq i]} exp(S_{i,k}/\tau)}\right) \tag{9}$$

Finally, with that we have the loss function for all $2N$ vectors z. The result will be a single value and it is defined in Equation 10. The network $f(\cdot)$ and $g(\cdot)$ are updated to minimize this value.

$$\mathscr{L} = \frac{1}{2N} \sum_{k=1}^{N} [L_{2k-1,2k} + L_{2k,2k-1}] \tag{10}$$

After pretraining on the unlabeled images guided by the pretext task, the method is able to either directly use the output of the ResNet as the final representation or fine-tune it with labeled images to achieve good performance for downstream tasks.

References

[1] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1096–1103.

[2] L. Weng, "From autoencoder to beta-vae," *lilianweng.github.io/lil-log*, 2018. [Online]. Available: http://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html.

[3] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2536–2544.

[4] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," *CoRR*, vol. abs/1603.08511, 2016. arXiv: 1603.08511. [Online]. Available: http://arxiv.org/abs/1603.08511.

[5] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," *arXiv preprint arXiv:2002.05709*, 2020.