A. Connection between BARLOW TWINS and the Information Bottleneck Principle

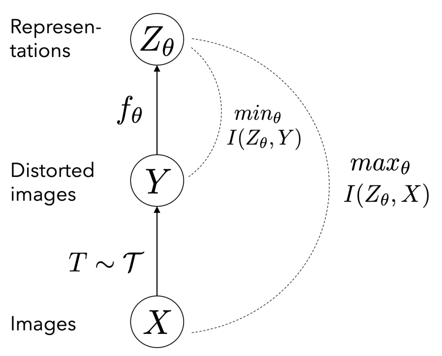

Figure 6. The information bottleneck principle applied to self-supervised learning (SSL) posits that the objective of SSL is to learn a representation Z_θ which is informative about the image sample, but invariant (i.e. uninformative) to the specific distortions that are applied to this sample. BARLOW TWINS can be viewed as a specific instanciation of the information bottleneck objective.

We explore in this appendix the connection between BARLOW TWINS' loss function and the *Information Bottleneck* (IB) principle (Tishby & Zaslavsky, 2015; Tishby et al., 2000).

As a reminder, BARLOW TWINS' loss function is given by:

$$\mathcal{L_{BT}} \triangleq \underbrace{\sum_i (1 - \mathcal{C}_{ii})^2}_{\text{invariance term}} + \lambda \underbrace{\sum_i \sum_{j \neq i} {\mathcal{C}_{ij}}^2}_{\text{redundancy reduction term}} \tag{3}$$

where λ is a positive constant trading off the importance of the first and second terms of the loss, and where $\mathcal{C}$ is the cross-correlation matrix computed between the outputs of the two identical networks along the batch dimension :

$$\mathcal{C}_{ij} \triangleq \frac{\sum_b z^A_{b,i} z^B_{b,j}}{\sqrt{\sum_b \left(z^A_{b,i}\right)^2} \sqrt{\sum_b \left(z^B_{b,j}\right)^2}} \tag{4}$$

where b indexes batch samples and i, j index the vector dimension of the networks' outputs. $\mathcal{C}$ is a square matrix with size the dimensionality of the network's output, and with values comprised between -1 (i.e. perfect anti-correlation) and 1 (i.e. perfect correlation).

Applied to self-supervised learning, the IB principle posits that a desirable representation should be as informative as possible about the sample represented while being as invariant (i.e. non-informative) as possible to distortions of that sample (here the data augmentations used) (Fig. 6). This trade-off is captured by the following loss function:

$$\mathcal{IB}_\theta \triangleq I(Z_\theta, Y) - \beta I(Z_\theta, X) \tag{5}$$

where $I(.,.)$ denotes mutual information and β is a positive scalar trading off the desideratas of preserving information and being invariant to distortions.

Using a classical identity for mutual information, we can rewrite equation 5 as:

$$\mathcal{IB}_\theta = [H(Z_\theta) - \cancelto{0}{H(Z_\theta|Y)}] - \beta[H(Z_\theta) - H(Z_\theta|X)] \tag{6}$$

where $H(.)$ denotes entropy. The conditional entropy $H(Z_\theta|Y)$ —the entropy of the representation conditioned on a specific distorted sample— cancels to 0 because the function f_θ is deterministic, and so the representation Z_θ conditioned on the input sample Y is perfectly known and has zero entropy. Since the overall scaling factor of the loss function is not important, we can rearrange equation 6 as:

$$\mathcal{IB}_\theta = H(Z_\theta|X) + \frac{1-\beta}{\beta} H(Z_\theta) \tag{7}$$

Measuring the entropy of a high-dimensional signal generally requires vast amounts of data, much larger than the size of a single batch. In order to circumvent this difficulty, we make the simplifying assumption that the representation Z is distributed as a Gaussian. The entropy of a Gaussian distribution is simply given by the logarithm of the determinant of its covariance function (up to a constant corresponding to the assumed discretization level that we ignore here) (Cai et al., 2015). The loss function becomes:

$$\mathcal{IB}_\theta = \mathbb{E}_X log\, |\mathcal{C}_{Z_\theta|X}| + \frac{1-\beta}{\beta} log\, |\mathcal{C}_{Z_\theta}| \tag{8}$$

This equation is still not exactly the one we optimize for in practice (see eqn. 3 and 4). Indeed, our loss function is only connected to the IB loss given by eqn. 8 through the following simplifications and approximations:

- In the case where $\beta <= 1$, it is easy to see from eqn. 8 that the best solution to the IB trade-off is to set the representation to a constant that does not depend on the input. This trade-off thus does not lead to interesting representations and can be ignored. When $\beta > 1$, we note that the second term of eqn. 8 is preceded by a negative constant. We can thus simply replace $\frac{1-\beta}{\beta}$ by a new positive constant λ, preceded by a negative sign.

- In practice, we find that directly optimizing the determinant of the covariance matrices does not lead to SoTA solutions. Instead, we replace the second term of the loss in eqn. 8 (maximizing the information about samples), by the proxy which consist in simply minimizing the Frobenius norm of the cross-correlation matrix. If the representations are assumed to be re-scaled to 1 along the batch dimension before entering the loss (an assumption we are free to make since the cross-correlation matrix is invariant to this re-scaling), this minimization only affects the off-diagonal terms of the covariance matrix (the diagonal terms being fixed to 1 by the re-scaling) and encourages them to be as close to 0 as possible. It is clear that this surrogate objective, which consists in decorrelating all output units, has the same global optimum than the original information maximization objective.

- For consistency with eqn. 8, the second term in BARLOW TWINS' loss should be computed from the auto-correlation matrix of one of the twin networks, instead of the cross-correlation matrix between twin networks. In practice, we do not see a strong difference in performance between these alternatives.

- Similarly, it can easily be shown that the first term of eqn. 8 (minimizing the information the representation contains about the distortions) has the same global optimum than the first term of eqn. 3, which maximizes the alignment between representations of pairs of distorted samples.

B. Evaluations on ImageNet

B.1. Linear evaluation on ImageNet

The linear classifier is trained for 100 epochs with a learning rate of 0.3 and a cosine learning rate schedule. We minimize the cross-entropy loss with the SGD optimizer with momentum and weight decay of 10^{-6}. We use a batch size of 256. At training time we augment an input image by taking a random crop, resizing it to 224×224, and optionally flipping the image horizontally. At test time we resize the image to 256×256 and center-crop it to a size of 224×224.

B.2. Semi-supervised training on ImageNet

We train for 20 epochs with a learning rate of 0.002 for the ResNet-50 and 0.5 for the final classification layer. The learning rate is multiplied by a factor of 0.2 after the 12th and 16th epoch. We minimize the cross-entropy loss with the SGD optimizer with momentum and do not use weight decay. We use a batch size of 256. The image augmentations are the same as in the linear evaluation setting.

C. Transfer Learning

C.1. Linear evaluation

We follow the exact settings from PIRL (Misra & van der Maaten, 2019) for evaluating linear classifiers on the Places-205, VOC07 and iNaturalist2018 datasets. For Places-205 and iNaturalist2018 we train a linear classifier with SGD (14 epochs on Places-205, 84 epochs on iNaturalist2018) with a learning rate of 0.01 reduced by a factor of 10 at two equally spaced intervals, a weight decay of 5×10^{-4} and SGD momentum of 0.9. We train SVM classifiers on the VOC07 dataset where the C values are computed using cross-validation.

C.2. Object Detection and Instance Segmentation

We use the detectron2 library (Wu et al., 2019) for training the detection models and closely follow the evaluation settings from (He et al., 2019). The backbone ResNet50 network for Faster R-CNN (Ren et al., 2015) and Mask R-CNN (He et al., 2017) is initialized using our BARLOW TWINS pretrained model.

VOC07+12 We use the VOC07+12 `trainval` set of $16K$ images for training a Faster R-CNN (Ren et al., 2015) C-4 backbone for $24K$ iterations using a batch size of 16 across 8 GPUs using SyncBatchNorm. The initial learning rate for the model is 0.1 which is reduced by a factor of 10 after $18K$ and $22K$ iterations. We use linear warmup (Goyal et al., 2017) with a slope of 0.333 for 1000 iterations.

COCO We train Mask R-CNN (He et al., 2017) C-4 backbone on the COCO 2017 `train` split and report results on the `val` split. We use a learning rate of 0.03 and keep the other parameters the same as in the $1\times$ schedule in detectron2.