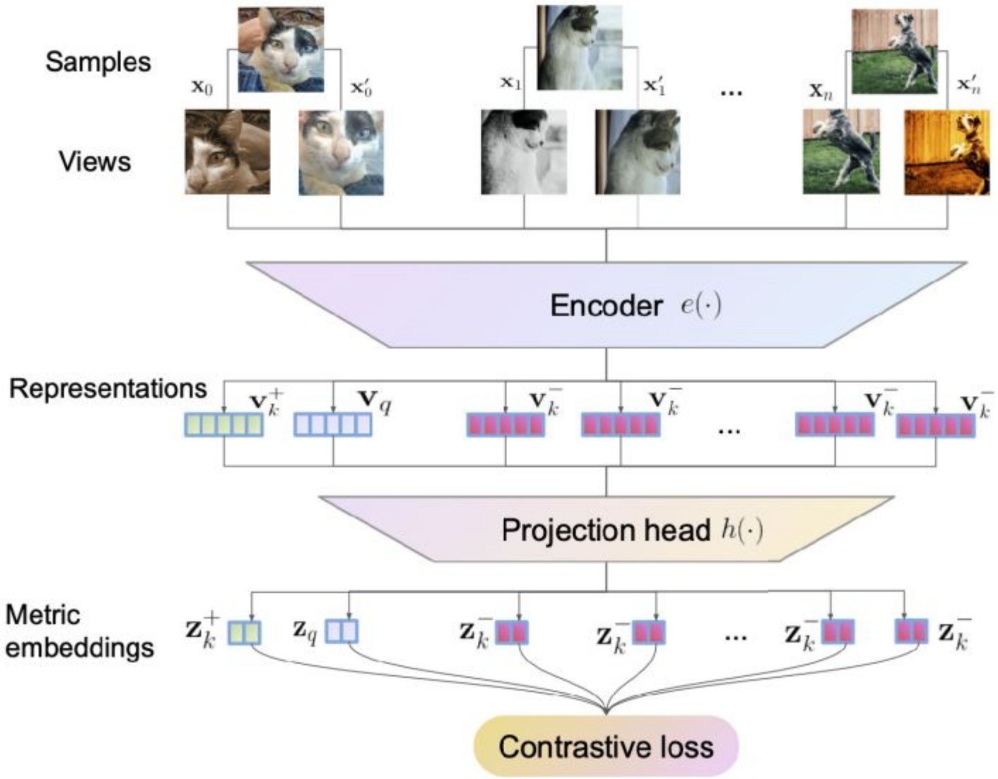
Self supervised learning: non-contrastive

SSL: Recap

- Three types of learning
  - **Supervised**: You have the data and you have the annotations/labels, you want to perform classification/regression
  - **Unsupervised**: You have the data but no label, and you want to figure out some sort of latent representation from the data e.g. clustering, dimensionality reduction
  - **Self supervised**: You have the data but no annotations/labels, and you want to perform classification/regression
- Self supervised learning (SSL) tries to take the best from both the worlds of supervised and unsupervised
  - There are often so much data but without labels
  - Labelling requires manual labor, is time consuming and is prone to errors
    - e.g. ImageNet labelling with 14M images took roughly 22 human years [1]

[1] "Self supervised learning - Pretext tasks" - https://atcold.github.io/pytorch-Deep-Learning/en/week10/10-1/

Contrastive SSL: Recap



Query	An input data point
Positive key	An augmented/transformed version of the input data point
Negative key	All other data in the dataset except the query and positive key
Encoder	Will learn an embedding space of latent representation
Projection Head	Use some form of projection, contextualization or quantization on the embedding space to learn a better representation
Contrastive loss	Minimizes the “distance” between query and positive key, while maximizing the “distance” between query and positive key pair and negative key(s)

Contrastive SSL: Recap (continued)

Contrastive Loss: The key component

$$\mathcal{L}^{self} = \sum_{i \in I} \mathcal{L}_i^{self} = -\sum_{i \in I} \log \frac{\exp\left(\boldsymbol{z}_i \bullet \boldsymbol{z}_{j(i)}/\tau\right)}{\sum_{a \in A(i)} \exp\left(\boldsymbol{z}_i \bullet \boldsymbol{z}_a/\tau\right)}$$

Here, $\boldsymbol{z}_\ell = Proj(Enc(\tilde{\boldsymbol{x}}_\ell)) \in \mathcal{R}^{D_P}$, the $\bullet$ symbol denotes the inner (dot) product, $\tau \in \mathcal{R}^+$ is a scalar temperature parameter, and $A(i) \equiv I \setminus \{i\}$. The index i is called the *anchor*, index $j(i)$ is called the *positive*, and the other $2(N-1)$ indices ($\{k \in A(i) \setminus \{j(i)\}\}$) are called the *negatives*.

Note that for each anchor i ∈ I ≡ {1...2N}, there is 1 positive pair and 2N − 2 negative pairs. The denominator has a total of 2N − 1 terms (the positive and negatives).

Contrastive SSL: Recap (continued)

Caveats

- How much negative data is needed?
  - More negative data usually means that the representation will not collapse into a single cluster
  - Computationally expensive
- Quality of the negative data?
  - More careful selection of negative samples has been shown to improve the convergence rate and performance of the learned embeddings on downstream tasks
  - Consistent with hard negative and positive mining techniques
- Trade-off between quality and quantity
- There are works that do not use negative samples at all (non-contrastive), e.g. Bootstrap Your Own Latent (BYOL), Simple Siamese (SimSiam)

Non-contrastive SSL

- Get rid of the idea of “negative samples”
  - Treat every data point as part of the positive samples
  - No batch size limitations
  - No hard negative required
  - Relatively easier to train compared to contrastive approach
- Learn noise and distortion invariant representation for each data point
  - So you can feed the network a very high amount of unlabelled data during pretext task
- Shows on par results with state-of-the-art (SOTA) supervised learning techniques

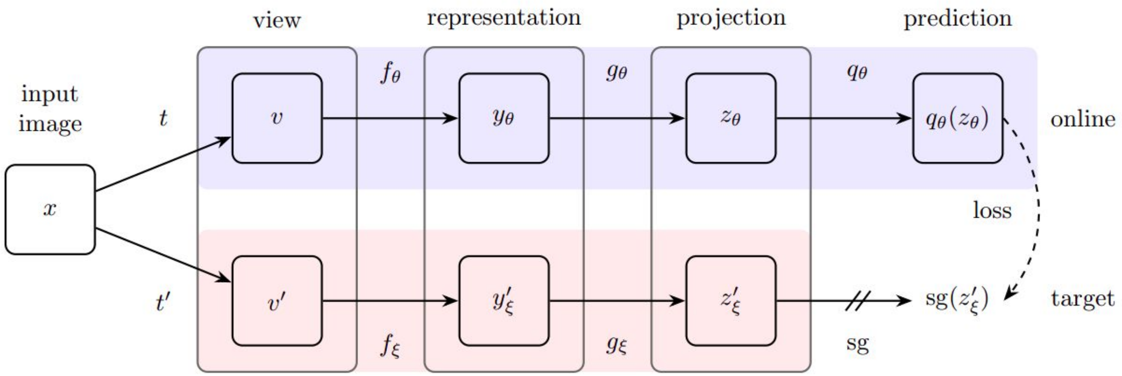
Bootstrap Your Own Latent (BYOL): A New Approach to Self-Supervised Learning

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond
DeepMind, Google
NeurIPS (2020)

Bootstrap Your Own Latent (BYOL)

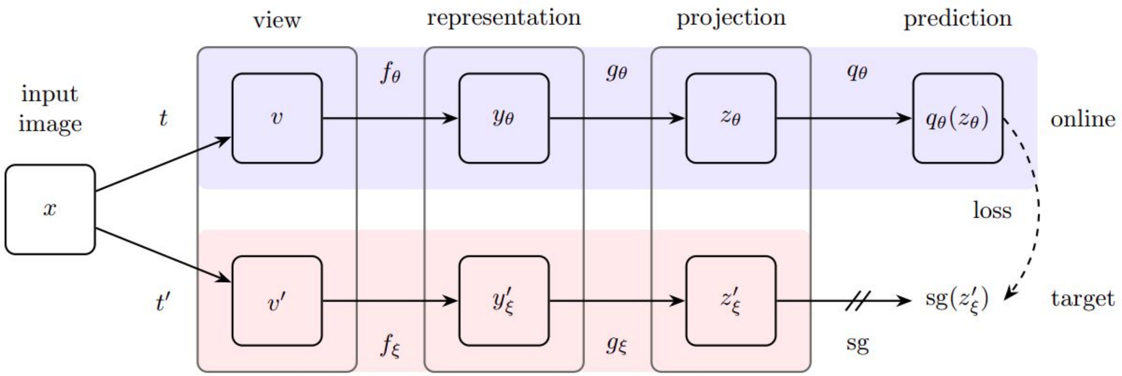
- Problem statement
  - A successful approach to SSL is to learn embeddings which are invariant to distortions of the input sample
  - How to not deal with negative pairs required in contrastive learning
    - And still perform on par with SOTA supervised and other self-supervised contrastive learning approaches?

Bootstrap Your Own Latent (continued)



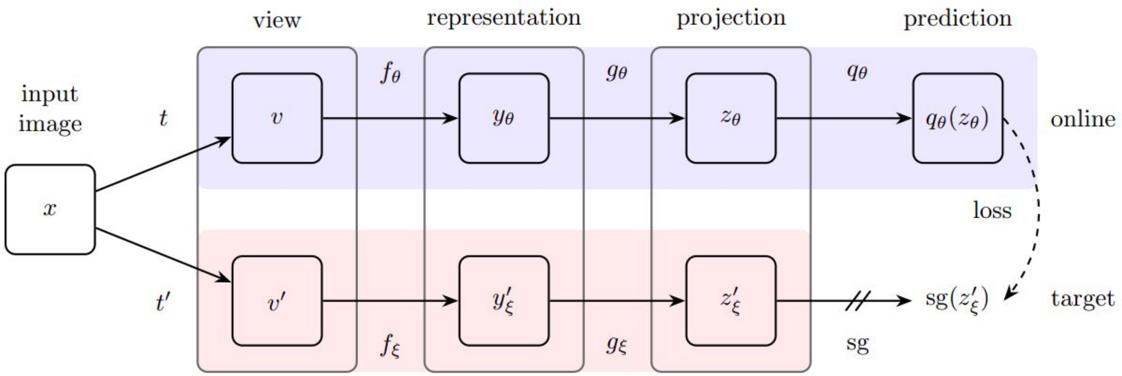
- Composed of two almost-identical networks
  - Target network
  - Predictor network → Online
- Almost-identical
  - The weights of the target network is a function of the weights of the predictor network
  - Asymmetric architecture - only online network has the predictor

Bootstrap Your Own Latent (continued)



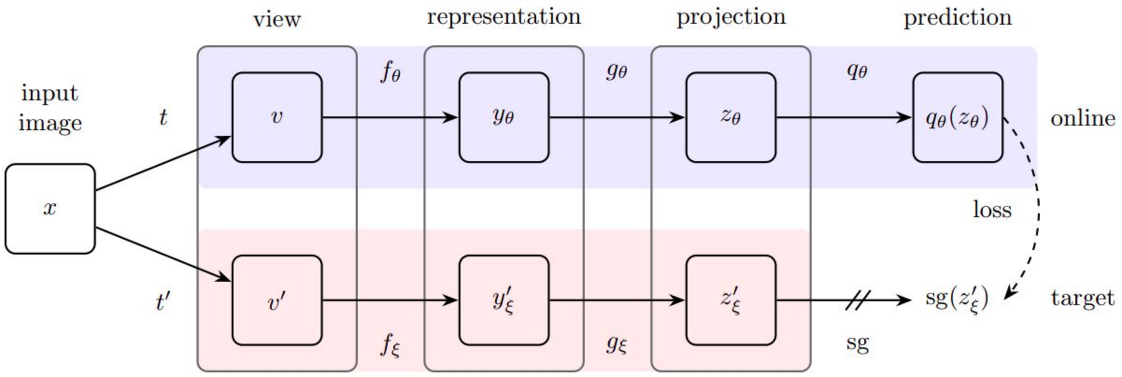
- The predictor network tries to learn an augmentation invariant representation of the input x
  - Make its own prediction match as close as possible to the output of the target network
  - The input to both networks (predictor and online) are two differently transformed versions of the same image

Bootstrap Your Own Latent (continued)



- The weights of the target network is updated after each training step
  - The weights of the target network is given by $\xi \leftarrow \tau\,\xi + (1 - \tau\,)\theta$.
  - Where, τ is a target decay between 0 and 1
- The target network uses stop gradient (sg) to prevent back propagation and update of weights, since its weights are updated by the online network.

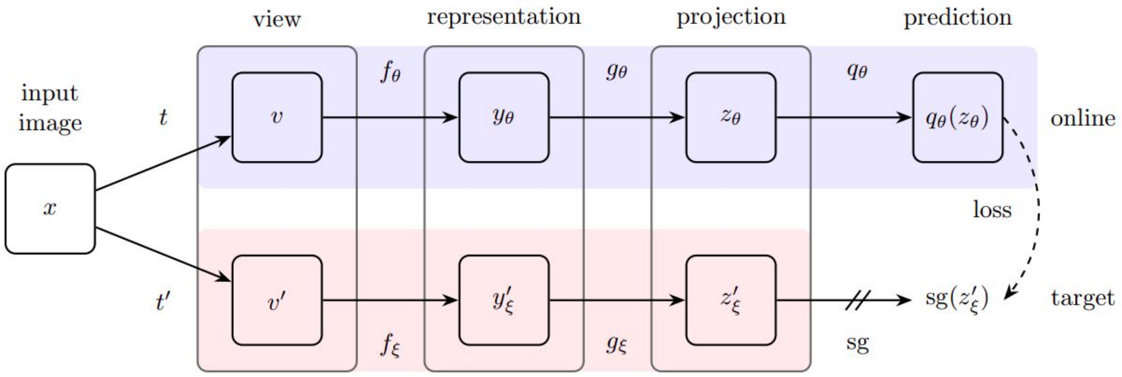
Bootstrap Your Own Latent (continued)



- The prediction z and z' are both l2 normalized.
- After each training step, minimize the MSE between normalized ***prediction*** and target ***projection*** $\mathcal{L}_{\theta,\xi}$

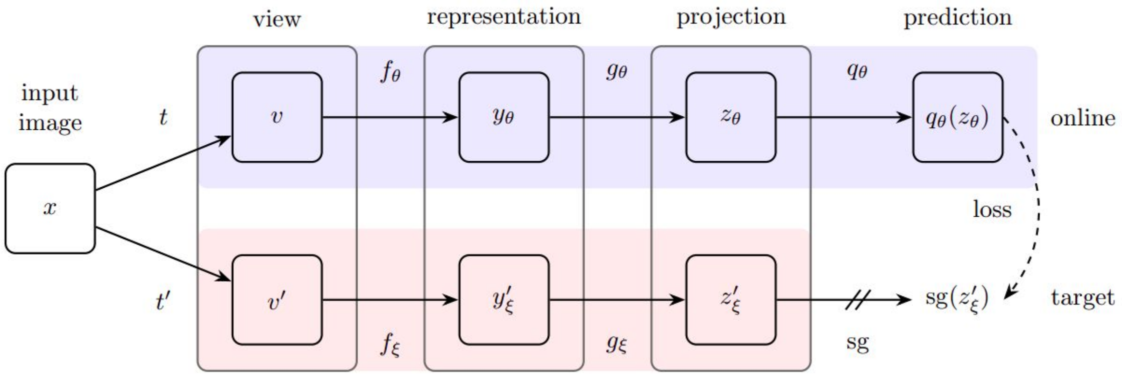
$$\mathcal{L}_{\theta,\xi} \triangleq \left\|\overline{q_\theta}(z_\theta) - \overline{z}'_\xi\right\|_2^2 = 2 - 2 \cdot \frac{\langle q_\theta(z_\theta), z'_\xi \rangle}{\left\|q_\theta(z_\theta)\right\|_2 \cdot \left\|z'_\xi\right\|_2}.$$

Bootstrap Your Own Latent (continued)



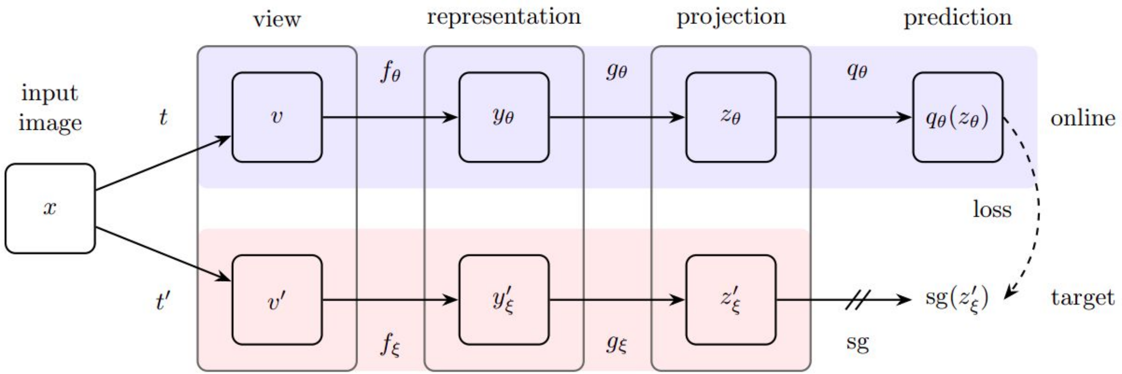
- The loss is symmetrized by reversing the inputs
  - v' is fed to the online network
  - v is fed to the target network
  - Compute $\widetilde{\mathcal{L}}_{\theta,\xi}$

Bootstrap Your Own Latent (continued)



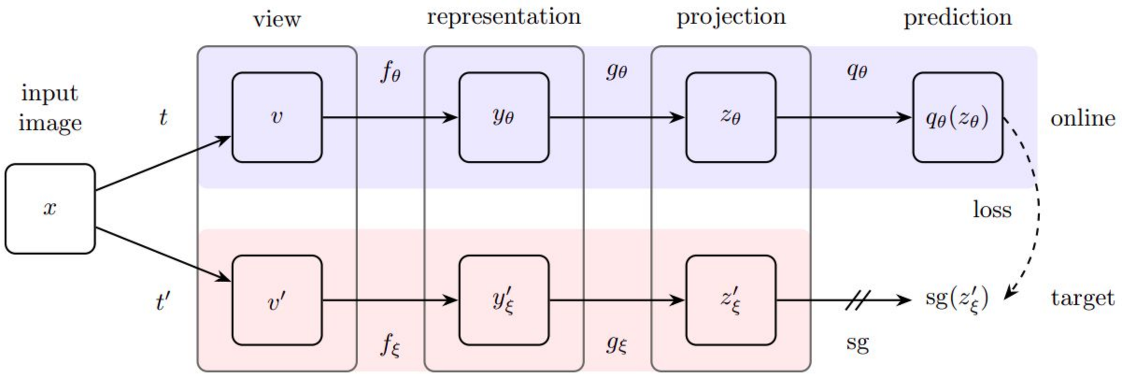
- The actual loss therefore is $\mathcal{L}^{\mathtt{BYOL}}_{\theta,\xi} = \mathcal{L}_{\theta,\xi} + \widetilde{\mathcal{L}}_{\theta,\xi}$
  - Perform a stochastic optimization step on this loss

Bootstrap Your Own Latent (continued)



- The loss **L** is not simply a gradient descent over ξ and θ
  - Similar to GANs, where there is no loss that is jointly minimized with respect to both the discriminator and predictor parameters
- Not copying the weights over to the target network makes it resistant to sudden changes in the predictor network

Bootstrap Your Own Latent (continued)



- ResNet with residual CNN with 50 layers used as the basic encoder
  - They have also used deeper and wider residual CNNs
  - The final output has 2048 dimensions
- The projector is an MLP
  - Final output size is 256
- LARS optimizer is used

Barlow Twins: Self-Supervised Learning via Redundancy Reduction

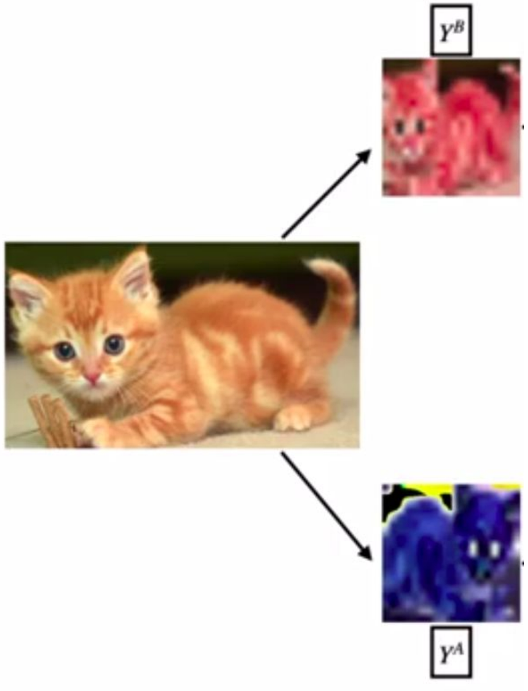
Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, Stephane Deny
Facebook AI Research
ICML (2021)

Barlow Twins

- Problem statement
  - A successful approach to SSL is to learn embeddings which are invariant to distortions of the input sample
  - A recurring issue with this approach is the existence of trivial constant solutions
    - The embedding space keeps outputting constant vectors i.e. failing to learn the latent representations of the input
    - Known as the collapse problem
  - Contrastive learning is heavily dependent on batch sizes and hard negative mining
  - Can this be done in an easier way?
  - No requirements on:
    - large batches
    - asymmetry between the network twins such as a predictor network
    - gradient stopping
    - a moving average on the weight updates

Barlow Twins (continued)

- Based on the redundancy-reduction principle introduced by Barlow [10] in the field of neurosciences
  - Similar to how humans learn differences between entities
- Take a sample image
- Apply two transformations on it
  - The transformations are chosen from a probabilistic distribution
  - Includes:
    - Random crop
    - Random rotation
    - Color adjustments
    - Jittering
    - Blurring
- Try to minimize the differences in representations between the transformed images

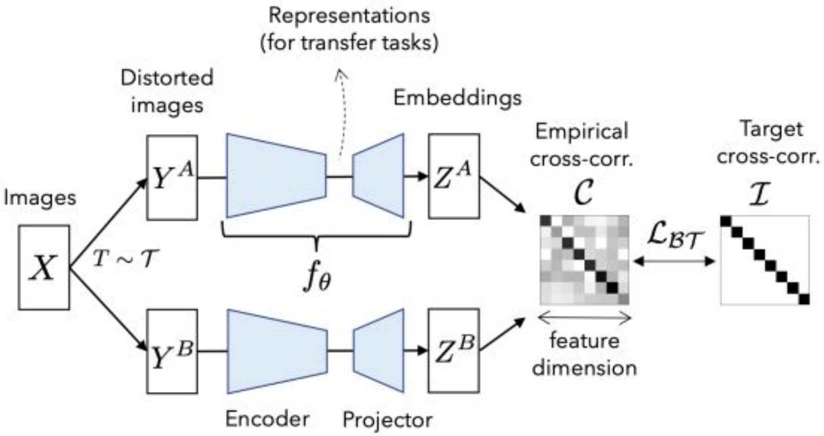


[10] Possible Principles Underlying the Transformations of Sensory Messages, Barlow, H. B. (1961)

Barlow Twins (continued)

Based on joint embedding learning with siamese networks

- Neural networks that have two or more identical subnetworks with shared weights and same configuration/parameters
- Parameter updating is mirrored across sub-networks



To simplify notations, Z^A and Z^B are assumed to be mean-centered along the batch dimension, such that each unit has mean output 0 over the batch.

Barlow Twins (continued)

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

Where,
λ is a positive constant trading off the importance of the first and second terms of the loss and
C is the square cross-correlation matrix

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

Where,
b indexes batch samples
i, j index the vector dimension of the networks' outputs
C is a cross-correlation matrix

Barlow Twins (continued)

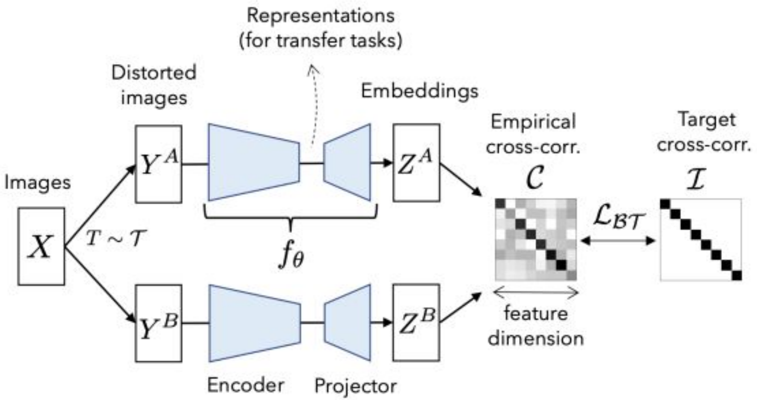
$$\mathcal{L}_{\mathcal{BT}} \triangleq \underbrace{\sum_i (1 - \mathcal{C}_{ii})^2}_{\text{invariance term}} + \lambda \underbrace{\sum_i \sum_{j \neq i} {\mathcal{C}_{ij}}^2}_{\text{redundancy reduction term}}$$

The goal is:

- Get the diagonal elements of C as close to 1 as possible
  - Makes the learnt representations invariant to distortions
- Get the off-diagonal elements as close to 0 as possible
  - De-correlate the vector components of the embedding

So, we want a representation that is invariant to distortions and noise, while also preserving maximum information of the entity we are trying to represent

Barlow Twins (continued)

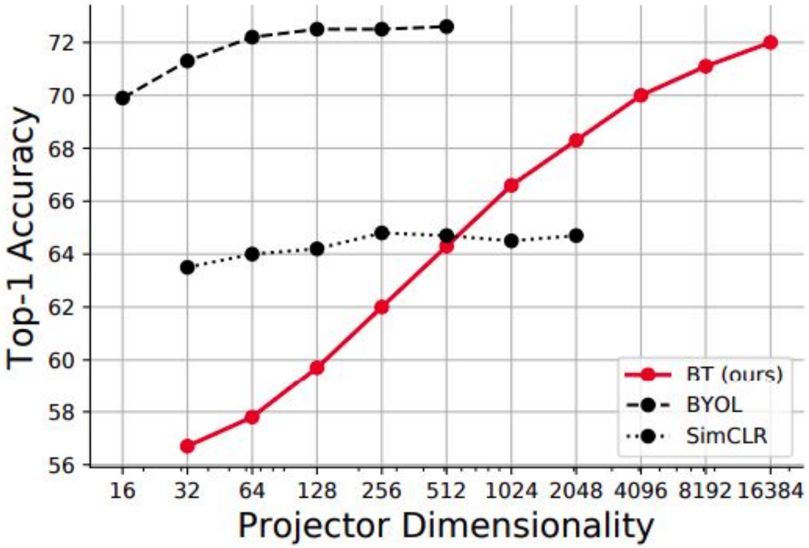


Architecture

- **Encoder: ResNet-50 network**
  - without the final classification layer
  - 2048 output units
- **Projector: 3 linear layers**
  - each with 8192 output units.
  - The first two layers of the projector are followed by a batch normalization layer and rectified linear units
- LARS optimizer

Barlow Twins (continued)

A very interesting yet unexplained outcome of the study



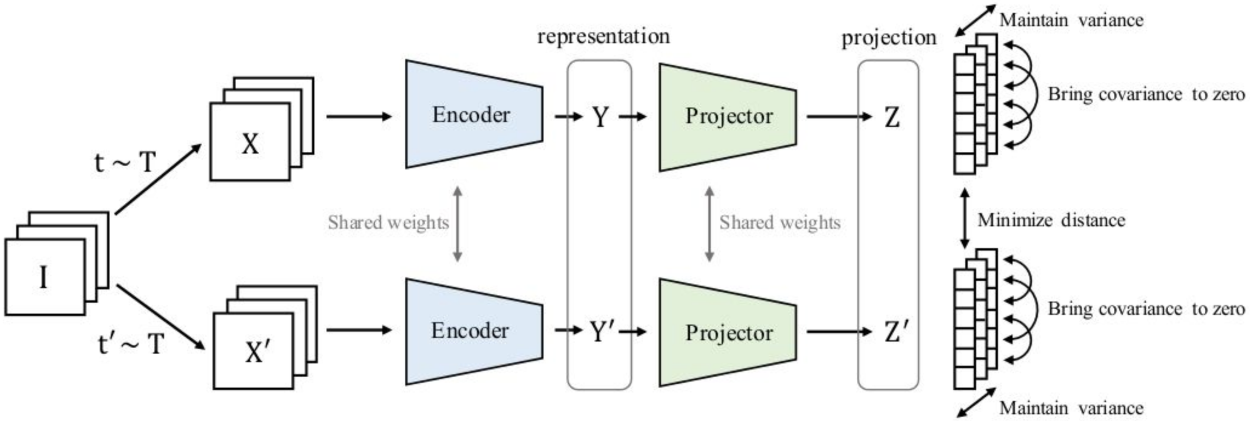
VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning

Adrien Bardes, Jean Ponce, Yann LeCun
Facebook AI Research
ICPS (2021)

Problem statement

- A problem in SSL
  - The encoder can end up outputting constant vectors, i.e. a vector of just 1s
  - Known as the collapse problem
- Some form of regularization is needed to solve the collapse problem
- Free from "architectural tricks"
  - BYOL depends on stop gradients and asymmetric networks
- Does not need normalization of projections/embeddings
  - e.g. Barlow Twins
- Achieves results on par with the state of the art on several downstream tasks
- Three simple principles: variance, invariance and covariance

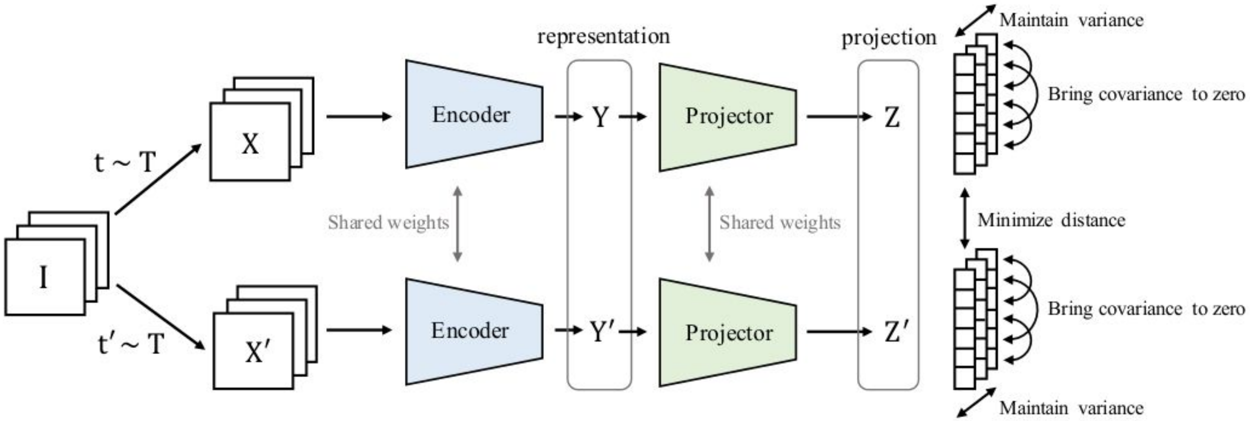
VICReg: Variance-Invariance-Covariance Regularization



Variance principle

- constraints the variance of the embeddings along each dimension independently
- Use a hinge loss which constrains the standard deviation computed along the batch dimension of the embeddings to reach a fixed target

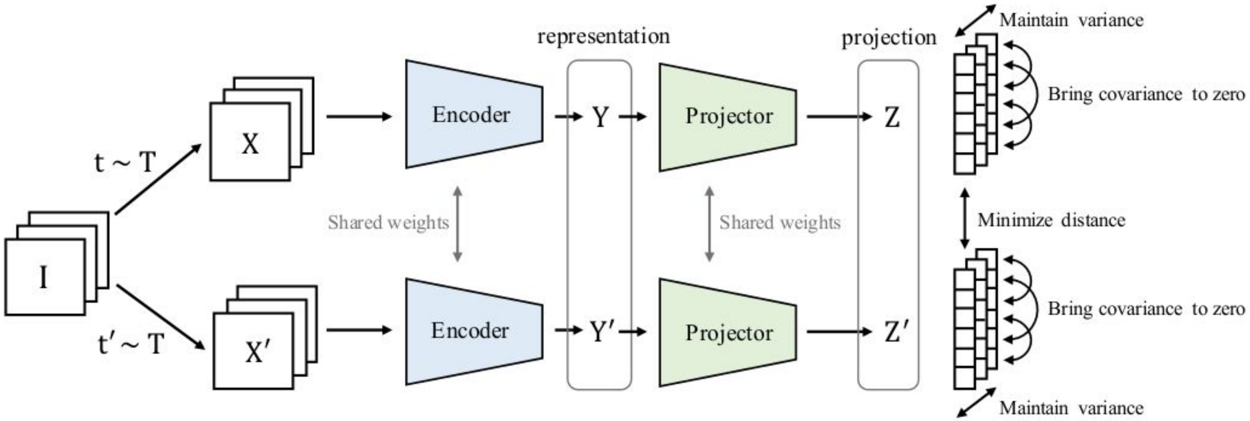
VICReg: Variance-Invariance-Covariance Regularization



Invariance principle

- uses a standard mean-squared euclidean distance to learn invariance to multiple views of an image

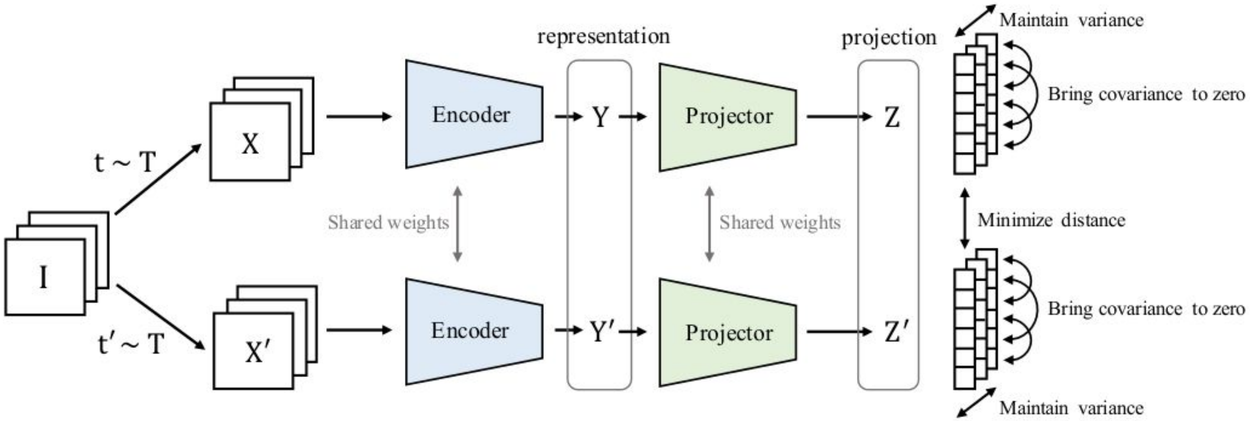
VICReg: Variance-Invariance-Covariance Regularization



Covariance principle

- Prevent different dimensions of the same projection from encoding the same information
- Inspired by Barlow Twins

VICReg: Variance-Invariance-Covariance Regularization



The authors used a limited number of transformations in T

- random crops of the image
- color distortions

VICReg: Variance-Invariance-Covariance Regularization

- Variance regularization term *v*

$$v(Z) = \frac{1}{d}\sum_{j=1}^{d} \max(0, \gamma - \sqrt{\mathrm{Var}(Z_{:,j}) + \epsilon})$$

$$\mathrm{Var}(x) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

i = an image sampled from a dataset D, i ~ D

γ = Target value of standard deviation, fixed to 1

Z = Batch of n vectors of dimension d

$Z_{:,j}$ = the vector composed of each value at dimension j in all vectors in Z

ϵ = A small scalar protecting from numerical instabilities

$\bar{x}$ is the mean of x, n is the size of x

VICReg: Variance-Invariance-Covariance Regularization

- Variance regularization term *v*

$$v(Z) = \frac{1}{d} \sum_{j=1}^{d} \max(0, \gamma - \sqrt{\mathrm{Var}(Z_{:,j}) + \epsilon})$$

$$\mathrm{Var}(x) = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2$$

Force variance to be γ along the batch dimension

Use of standard deviation instead of variance in hinge loss because the gradient of the variance can become 0 when input vector is close to the mean vector in the batch

VICReg: Variance-Invariance-Covariance Regularization

- The covariance matrix of Z

$$C(Z) = \frac{1}{n-1}\sum_{i=1}^{n}(Z_i - \bar{Z})(Z_i - \bar{Z})^T, \quad \text{where} \quad \bar{Z} = \frac{1}{n}\sum_{i=1}^{n} Z_i.$$

- Z_i = i-th vector in Z
- The covariance regularization term *c*

$$c(Z) = \frac{1}{d}\sum_{i \neq j} C(Z)^2_{i,j}$$

VICReg: Variance-Invariance-Covariance Regularization

- The invariance criterion *s* between Z and Z’

$$s(Z, Z') = \frac{1}{n} \sum_i \|Z_i - Z'_i\|_2^2.$$

Z’ is the output of the other subnetwork of the siamese network. Neither Z nor Z’ is normalized via standardization or projection onto unit sphere

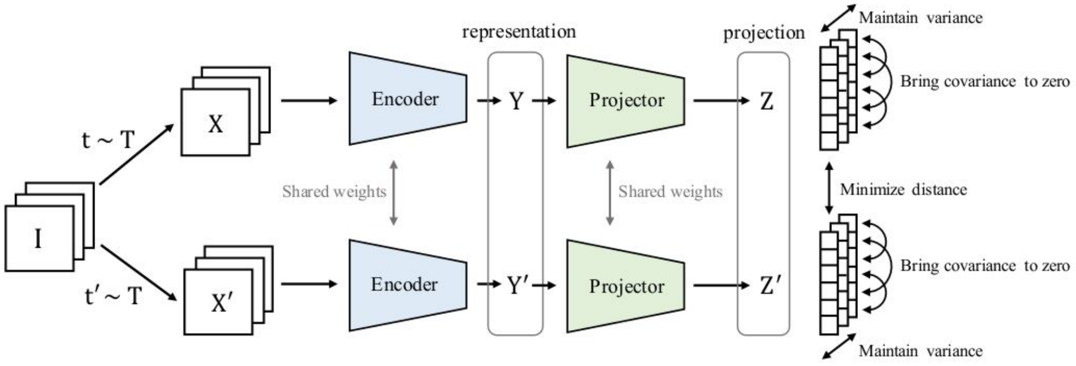
- The final loss function is therefore

$$\ell(Z, Z') = \lambda s(Z, Z') + \mu\{v(Z) + v(Z')\} + \nu\{c(Z) + c(Z')\},$$

λ, μ and ν are the hyperparameters controlling importance of each term.

They are found by grid search as stated by the authors

VICReg: Variance-Invariance-Covariance Regularization



- The final loss over the entire dataset is therefore

$$\mathcal{L} = \sum_{I \in \mathcal{D}} \sum_{t,t' \sim \mathcal{T}} \ell(Z^I, Z'^I),$$

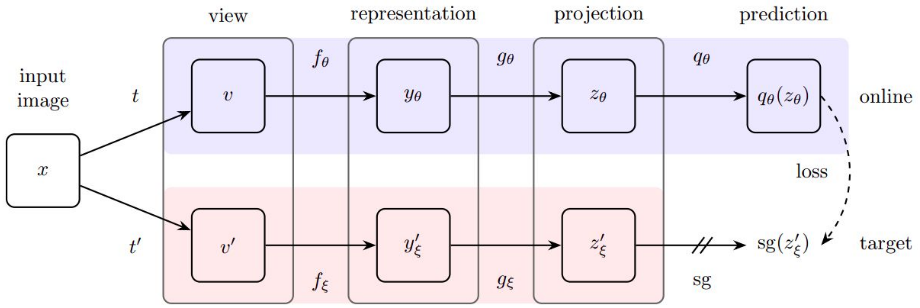
where Z^I and Z'^I are the batches of projection vectors corresponding to the batch of images I transformed by t and t', and is minimized over the encoder parameters θ and projector parameters ϕ.

Further study

- Simple Siamese Networks (SimSiam)[1]
  - Published in a similar timeframe as BYOL
  - Two differences with BYOL:
    - The weights of the encoders are shared
    - The loss function is the symmetrized negative cosine similarity instead of MSE.
- There are a number of factors that the authors of BYOL and Barlow Twins have mentioned which helps them avoid the collapse problem (empirically proven)
  - Weights of target network is an EMA of the weights of the target network
  - Use of weight decay
  - Use of relatively higher learning rate in the predictor network compared to the target network
- But why exactly do these work?
  - Explained in the "Understanding Self-Supervised Learning Dynamics without Contrastive Pairs"[2] paper

[1] Exploring Simple Siamese Representation Learning, CVPR (2021)
[2] Understanding Self-Supervised Learning Dynamics without Contrastive Pairs, ICML (2021)

Further study (continued)



- Understanding Self-Supervised Learning Dynamics without Contrastive Pairs[2]
  - Two-fold contribution
    - Explain why the mentioned constraints on the target and predictor networks work mathematically
    - DirectPred
  - Discover an important relation between weights of the encoder and the predictor
    - Eigenspace alignment [2] [3]
  - This discovery leads to an important conclusion
    - The weights of the predictor network can be directly found out using the correlation matrix of the inputs
    - Dubbed as DirectPred in the paper
    - No need for gradient descent
  - DirectPred performs on par with BYOL, Barlow Twins with a significantly simpler network structure

[2] Facebook AI Research, Understanding Self-Supervised Learning Dynamics without Contrastive Pairs (2021)
[3] Facebook AI Research Blog, Demystifying a key self-supervised learning technique: Non-contrastive learning (2021) - link