1. Training Dynamics

Fig. 1 and 2 show the training dynamics for methods compared in Tab. 1 of the main paper. All the curves are smoothed with a 0.3 moving average for a better readability (curves before smoothing are shown semi-transparent).

2. Cholesky Whitening and Backprogation

We compute W_V (Eq. 8 of the main paper) following (Siarohin et al., 2019) and using the Cholesky decomposition. The Cholesky decomposition is based on the factorization of the covariance symmetric matrix using two triangular matrices: $\Sigma_V = LL^\top$, where L is a lower triangular matrix. Once we L is computed, we compute the inverse of L, and we get: $W_V = L^{-1}$. Note that the Cholesky decomposition is fully diferentiable and it is implemented in all of the major frameworks, such as PyTorch and TensorFlow. However, for the sake of completeness, we provide below the gradient computation.

2.1. Gradient Computation

We provide here the equations for whitening differentiation as reported in (Siarohin et al., 2019). Let Z be the whitened version of the batch V, i.e., $Z = W_V(V - \boldsymbol{\mu}_V)$. The gradient $\frac{\partial L}{\partial V}$ can be computed by:

$$\frac{\partial L}{\partial V} = \frac{2}{K-1}\frac{\partial L}{\partial \Sigma}V + W_V^T\frac{\partial L}{\partial Z}. \tag{1}$$

where the partial derivative $\frac{\partial L}{\partial Z}$ is backpropogated, while $\frac{\partial L}{\partial \Sigma}$ is computed as follows:

$$\frac{\partial L}{\partial \Sigma} = -\frac{1}{2}W_V^T\left(P\circ\frac{\partial L}{\partial W_V}W_V^T + \left(P\circ\frac{\partial L}{\partial W_V}W_V^T\right)^T\right)W_V \tag{2}$$

In (2), $\circ$ is Hadamard product, while $\frac{\partial L}{\partial W_V}$ is:

$$\frac{\partial L}{\partial W_V} = \frac{\partial L}{\partial Z}V^T, \tag{3}$$

and P is:

$$P = \begin{pmatrix} \frac{1}{2} & 0 & \cdots & 0 \\ 1 & \frac{1}{2} & \ddots & 0 \\ 1 & \ddots & \ddots & 0 \\ 1 & \cdots & 1 & \frac{1}{2} \end{pmatrix}.$$

3. Training time complexity

Following (Siarohin et al., 2019), the complexity of the whitening transform is $O(k^3 + Mk^2)$, where k is the embedding dimension and M is the size of the sub-batch used in the batch slicing process. Since $k < M$ (see Sec. 3 of the main paper), the whitening transform is $O(Mk^2)$, which is basically equivalent to the forward pass of M activations in a fully-connected layer connecting two layers of k neurons each. In fact, the training time is dominated by other architectural choices which are usually more computationally demanding than the loss computation. For instance, BYOL (Grill et al., 2020) needs 4 forward passes through 2 networks for each pair of positives. Hence, to evaluate the wall-clock time, we measure the time spent for one mini-batch iteration by all the methods compared in Tab. 1 of the main paper. We use the STL-10 dataset, a ResNet-18 encoder and a server with one Nvidia Titan Xp GPU. Time of one iteration: Contrastive, 459ms; BYOL, 602ms; W-MSE 2, 478ms; W-MSE 4, 493ms. The 19ms difference between Contrastive and W-MSE 2 is due to the whitening transform. Since the factual time is mostly related to the sample forward and backward passes, the $d(d-1)$ positive comparisons in Eq. 6 of the main paper, do not significantly increase the wall-clock time of W-MSE 4 with respect to W-MSE 2.

4. Euclidean distance

Table 1. Classification accuracy (top 1) using the Euclidean distance (unnormalized embeddings) on STL-10.

Method	linear	5-nn
SimCLR (our repro.)	78.00	71.07
BYOL (our repro.)	80.83	74.94
W-MSE 2	89.91	85.56
W-MSE 4	90.40	87.09

The cosine similarity is a crucial component in most of the current self-supervised learning approaches. This is usually implemented with an L_2 normalization of the latent-space representations, which corresponds to projecting the features on the surface of the unit hypersphere. However, in our W-MSE, the whitening transform projects the representation onto a spherical distribution (intuitively, we can say on the whole unit hypersphere). Preserving the module of the features before the L_2 normalization may be useful in some applications, e.g., clustering the features after the projection head using a Gaussian mixture model. Tab. 1 shows an experiment on the STL-10 dataset where we use unnormalized embeddings for all the methods (and $\tau = 1$ for the contrastive loss). Comparing Tab. 1 with Tab. 1 of the main paper, the accuracy decrease of W-MSE is significantly smaller than in the other methods.

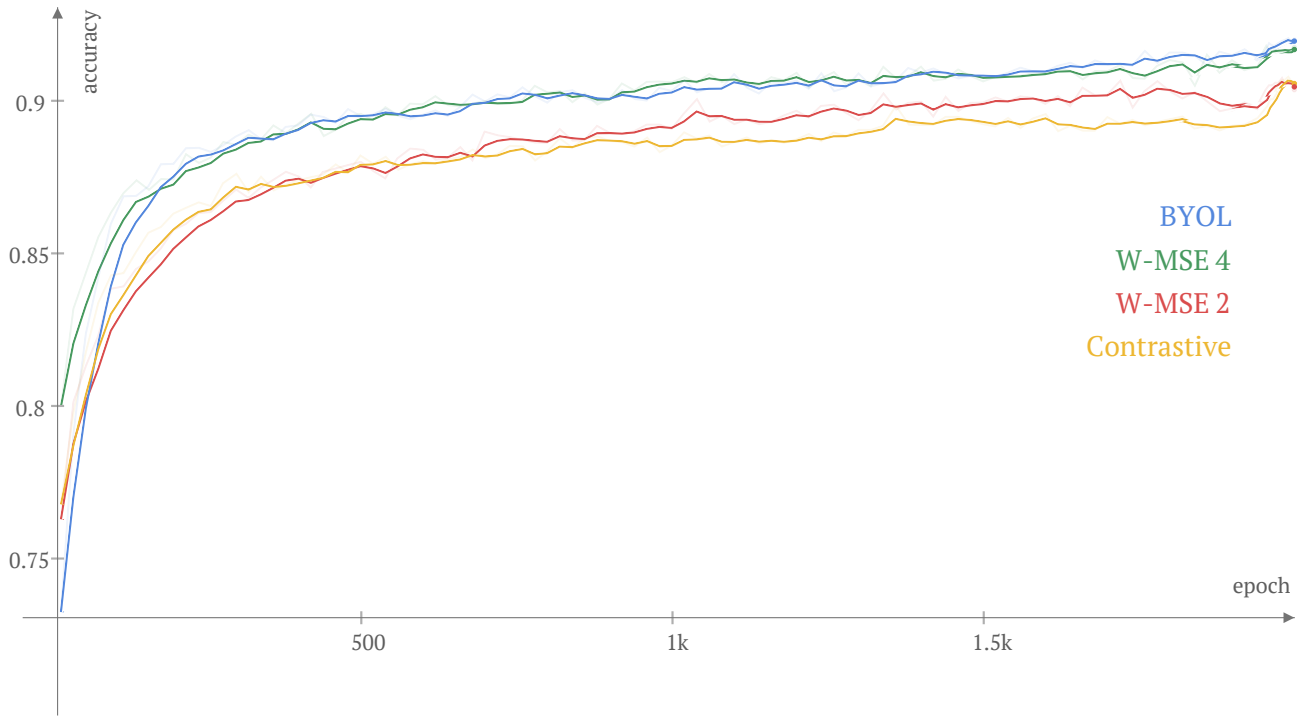

Figure 1. Training dynamics on the STL-10 dataset (linear-classifier based evaluation).

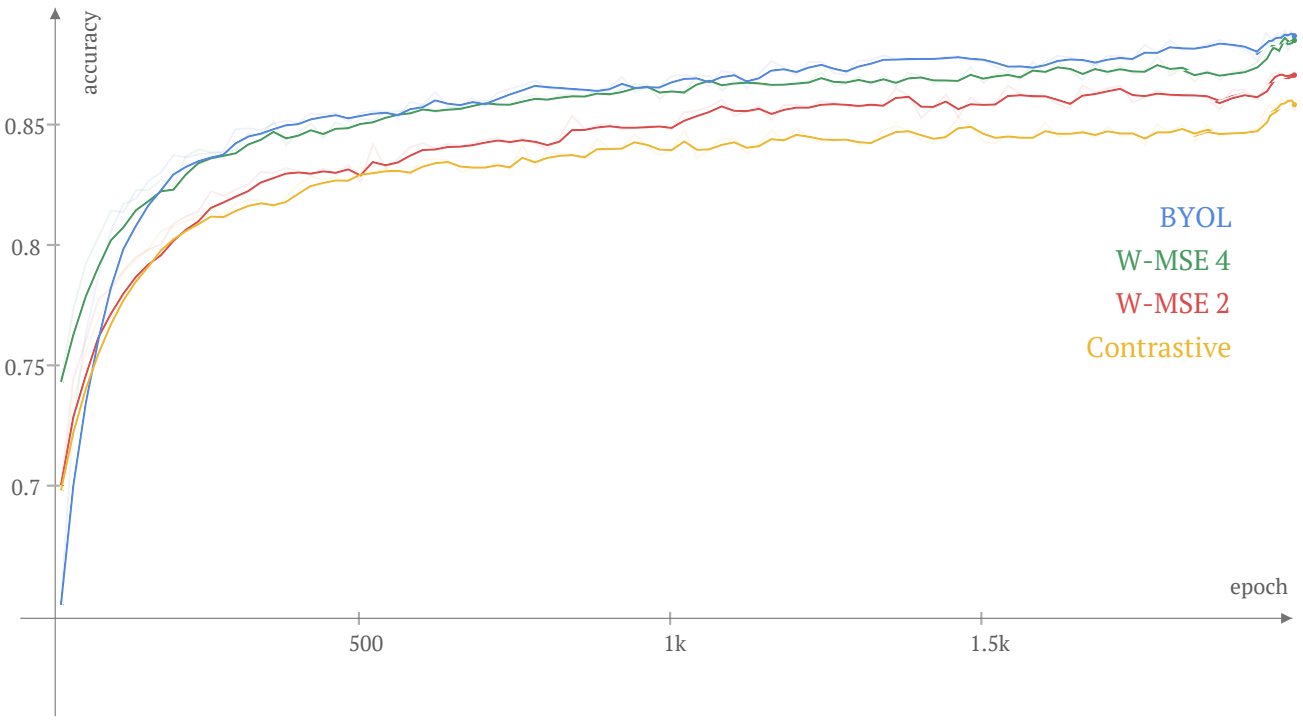

Figure 2. Training dynamics on the STL-10 dataset (5-nn classifier based evaluation).

References

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z. D., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., and Valko, M. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv:2006.07733*, 2020.

Siarohin, A., Sangineto, E., and Sebe, N. Whitening and coloring transform for GANs. In *International Conference on Learning Representations*, 2019.