The Edge of Orthogonality: A Simple View of What Makes BYOL Tick

Pierre H. Richemond [1] Allison Tam [1] Yunhao Tang [1] Florian Strub [1] Bilal Piot [1] Felix Hill [1]

Abstract

Self-predictive unsupervised learning methods such as BYOL (Grill et al., 2020) or SimSiam (Chen and He, 2020) have shown impressive results, and counter-intuitively, do not collapse to trivial representations. In this work, we aim at exploring the simplest possible mathematical arguments towards explaining the underlying mechanisms behind self-predictive unsupervised learning. We start with the observation that those methods crucially rely on the presence of a predictor network (and stop-gradient). With simple linear algebra, we show that when using a linear predictor, the optimal predictor is close to an orthogonal projection, and propose a general framework based on orthonormalization that enables to interpret and give intuition on why BYOL works. In addition, this framework demonstrates the crucial role of the exponential moving average and stop-gradient operator in BYOL as an efficient orthonormalization mechanism. We use these insights to propose four new *closed-form (or thereafter fixed-form) predictor* variants of BYOL to support our analysis. Our fixed-form predictors outperform standard linear trainable predictor BYOL at 100 and 300 epochs (top-1 linear accuracy on ImageNet).

1. Introduction

Modern Self-Supervised Learning (SSL) methods (Grill et al., 2020; Chen et al., 2020) have all but closed the performance gap with supervised methods in computer vision. Famously, these methods also form the basis for multimodal representation algorithms that power useful downstream applications such as generative modelling at scale (Radford et al., 2021; Jia et al., 2021; Ramesh et al., 2022; Alayrac et al., 2022). Amongst SSL methods broadly two families of algorithms coexist. First, **contrastive** methods iteratively push the representations of *positive pairs* of examples closer together, and pull *negative pairs* of examples further apart. They optimize a lower bound on the mutual information between representations and data (van den Oord et al., 2018).

Second, **self-predictive** methods rely on minimizing a least-squares objective between two network outputs with two different augmented views of the same inputs. This may involve two related or *Siamese* (Bromley et al., 1993) neural network streams with or without the same architecture (Grill et al., 2020; Chen and He, 2020; Ermolov et al., 2021). For instance, BYOL features both an *online* network, composed of an encoder and a predictor and a *target* network, composed of a similar encoder but not sharing weights. The target encoder is updated through an exponential moving average of the online encoder, whereas the online network (encoder and predictor) is updated through gradient descent by minimizing a BYOL loss between the stop-gradiented target encoder outputs and online predictor outputs. Both outputs are computed from the same inputs with different augmentations. Self-predictive learning is more memory-efficient than contrastive learning as it does not rely on extremely large batch sizes (due to negative examples) to reach good performance. However, it was initially surprising and opaque why those algorithms learn anything at all rather than collapsing to a single all-attracting constant value. Yet, several works started unveiling the underlying mechanisms behind self-predictive unsupervised learning (Tian et al., 2021; Liu et al., 2022; Halvagal et al., 2022) (see Sec. 7). Here we propose simple explanations for this phenomenon through linear algebra, and use those to derive new, *fixed-form* (closed-form or iterations) predictors for BYOL.

The crucial starting point is to notice that BYOL can be seen as matrix trace maximization, and we here refine this argument. Several fundamental algorithms in machine learning such as PCA (Pearson, 1901), spectral clustering (Cheeger, 1969; Shi and Malik, 1997), or non-negative matrix factorization (Paatero and Tapper, 1994; Lee and Seung, 1999), share this property and are also readily written as least-squares minimization objectives. Crucially they do not collapse *thanks to optimization constraints* (Kokiopoulou et al., 2011). For instance, the variational characterization of PCA implies maximization over orthogonal matrices (Horn and Johnson, 2013). In this work and as core contribution, we argue that a

[*]Equal contribution [1]Google DeepMind. Correspondence to: Pierre Harvey Richemond <richemond@deepmind.com>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

linear predictor version of BYOL is no different, but imposes predictor constraints that are both *approximate* and *implicitly enforced by its architecture*. Importantly, we justify these points using standard linear algebra, specifically, the optimal linear predictor is close to an orthogonal projection. This subsumes prior work (Tian et al., 2021; Liu et al., 2022; Halvagal et al., 2022) and enables a more general view informed by optimization on Riemannian manifolds of orthogonal matrices (Edelman et al., 1998; Absil et al., 2007; Bonnabel, 2013). We leverage this connection to propose several new self-predictive self-supervised algorithms with *fixed-form predictors*.

Contributions. Our contributions are as follows:

- We investigate the role of the linear predictor in BYOL and show its role as an approximate *orthogonal projection* as well as a spectral expansion operator for the covariance of latents.
- We interpret the BYOL update as a Riemannian gradient descent step, where the expensive *retraction* step is made computationally trivial by the predictor and EMA.
- We use those principles as inspiration to propose new self-supervised learning methods with four fixed-form predictors, summarized Table 1. We evaluate those empirically on ImageNet-1k, with a ResNet-50 encoder, at 100 and 300 epochs. Our fixed-form predictors outperform standard linear trainable predictor BYOL while only using matrix multiplications.

Besides unifying and simplifying our understanding of self-predictive unsupervised learning, these theoretical insights yield methods that can be more data- and compute-efficient, thus application-friendly.

2. Background and notations

Given batch size $b \in \mathbb{N}$, input dimension $d \in \mathbb{N}$ and latent dimension $f \in \mathbb{N}$, we consider batches of datapoints $X = (x^i)_{1 \leq i \leq b}$, where the batch X can be seen as a real matrix of shape (b, d) and each datapoint $x^i \in \mathbb{R}^d$ as a real vector. The datapoints $x^i = t^i(o^i)$ are transformations of the original raw inputs $o^i \in \mathbb{R}^d$ where the augmentations $t^i \sim \mathcal{T}$ are chosen uniformly from distribution $\mathcal{T}$. Similarly $X' = (x'^i = t'^i(o^i))_{1 \leq i \leq b}$ is augmented from distribution $\mathcal{T}'$. These are passed through the combined operation of an encoder and a projector, denoted as A_θ for the online branch and A_ξ for the target branch, resulting in latents $Z_\theta = A_\theta(X)$ and $Z'_\xi = A_\xi(X')$. The latents $Z_\theta = (z^i_\theta)_{1 \leq i \leq b}$ and $Z'_\xi = (z'^i_\xi)_{1 \leq i \leq b}$ are real matrices of shape (b, f) with respective row vectors $z^i_\theta \in \mathbb{R}^f$ and $z'^i_\xi \in \mathbb{R}^f$. The predictor is denoted by P_θ. The operator norm of a matrix X is noted $|||X|||$, its Frobenius norm $\|X\|_F$. Vector norms $\|x\|_2$ are Euclidean. The target parameter ξ is updated via exponential moving average with parameter τ and the online parameter θ is updated through gradient descent via the BYOL objective.

BYOL objective. Given a predictor P_θ and two distributions of latents Z_θ and Z'_ξ, the BYOL objective is the expected sum over the batch of the normalized L^2 distance between the target outputs z'^i_ξ and the online predictor outputs $P_\theta(z^i_\theta)$:

$$L_{\text{BYOL}}(\theta) = \mathbb{E}\left[\sum_{i=1}^{b} \left\| \frac{P_\theta(z^i_\theta)}{\|P_\theta(z^i_\theta)\|_2} - \texttt{sg}\left(\frac{z'^i_\xi}{\|z'^i_\xi\|_2}\right) \right\|_2^2\right] \tag{1}$$

where $\texttt{sg}(.)$ is a stop-gradient operator preventing backpropagation through parameters ξ. The stop-gradient notation is redundant and may be omitted, as the BYOL objective is optimized with respect to θ only.

3. A focus on the linear predictor

Despite the simplicity of the BYOL objective (1), prior work on BYOL theory has involved non-trivial matrix ordinary differential equation analysis under strong assumptions (Tian et al., 2021). We here present proofs with simpler linear algebra arguments. To do so, we make the two following simplifying assumptions from now on. First, we assume a linear predictor $P_\theta(Z_\theta) = Z_\theta P_\theta$, where $P_\theta \in \mathbb{R}^{f \times f}$ is a real square matrix. Second, we use the unnormalized L^2 distance rather than the normalized variant in our theoretical analysis. These two changes only worsen top-1 accuracy by a few percentage points as shown in Grill et al. (2020). Therefore, the BYOL objective becomes:

$$L_{\text{BYOL}}(\theta) = \mathbb{E}\left[\sum_{i=1}^{b} \|z^i_\theta P_\theta - z'^i_\xi\|_2^2\right] = \mathbb{E}\left[\|Z_\theta P_\theta - Z'_\xi\|_F^2\right], \tag{2}$$

by definitions of Z_θ, Z'_ξ and the Frobenius norm. Under this form we can also see the BYOL objective as a matrix trace:

$$L_{\text{BYOL}}(\theta) = \operatorname{tr}\left[\mathbb{E}\big((Z_\theta P_\theta - Z'_\xi)^\top (Z_\theta P_\theta - Z'_\xi)\big)\right]. \tag{3}$$

We begin with proving a crucial property of the linear optimal projector, namely, that it is an orthogonal projector (Prop. 1). We illustrate empirically that the subspace being projected upon grows throughout training, until the optimal predictor becomes the identity matrix at the end of training. This means BYOL does not collapse, and its feature space grows and, under conditions (Prop. 2), retains as much information as possible. Using these insights we propose a new fixed-form expression for a batchwise optimal predictor.

3.1. Optimal linear predictor and orthogonal projection

As described in Grill et al. (2020), the optimal predictor induces the following conditional expectation:

$$P^\star(z_\theta) \triangleq \arg\min_P \; \mathbb{E}\left[\|P(z_\theta) - z'_\xi\|_2^2\right] = \mathbb{E}\left[z'_\xi \mid z_\theta\right].$$

When the predictor is both optimal and *linear*, it performs exactly a linear regression, on a batch-by-batch basis, and so is given by:

$$P^\star = \arg\min_P \quad \left\| Z_\theta P - Z'_\xi \right\|_F^2 = \left(Z_\theta^\top Z_\theta \right)^{-1} Z_\theta^\top Z'_\xi. \tag{4}$$

We refer to this as **LRP** (linear regression predictor). We can use this setting to derive an important property of the batchwise-optimal linear predictor, as per proposition below:

Proposition 1. The linear optimal predictor, batchwise, is an orthogonal projection.

Proof: Inserting the right side of equation 4 into the BYOL loss, reveals the term $Z_\theta \left(Z_\theta^\top Z_\theta \right)^{-1} Z_\theta^\top$. That term is the orthogonal projection on the subspace spanned by the batch Z_θ, we denote it by $\mathrm{Proj}_{(Z_\theta)} = Z_\theta \left(Z_\theta^\top Z_\theta \right)^{-1} Z_\theta^\top$. The identity minus that term is also an orthogonal projection, on the orthogonal complement of that space, denoted by $\mathrm{Proj}_{(Z_\theta)^\perp}$. With such notations the BYOL loss for an optimal linear predictor P^* becomes:

$$L_{\mathrm{BYOL},P^*} = \mathbb{E}\Big[\left\| \mathrm{Proj}_{(Z_\theta)^\perp}(Z'_\xi) \right\|_F^2 \Big]. \tag{5}$$

The BYOL objective for optimal linear predictor can be seen as a reconstruction loss, performing self-denoising. This is a specialization of the relationship $L_{\mathrm{BYOL},P^*} = \mathbb{E}\left[\sum_j \mathrm{Var}\left(z'_{\xi,j} \mid z_\theta \right) \right]$ that appeared in Grill et al. (2020). Our equation 5 gives additional insights.

First, it shows qualitatively how the stop-gradient operation is absolutely necessary: without stop-gradient, the projection argument Z'_ξ would be a Z'_θ, and BYOL could be incentivized to minimize its loss by moving its representation Z_θ to be orthogonal to itself, triggering collapse. Second, we also see that the batch size controls the rank of denoising approximation when it is lower than feature dimensionality.

Scaled orthogonal projection, towards the identity. An orthogonal projection P satisfies $P^2 = P$ and $P = P^\top$. We will assume that the linear predictor is close to optimality, thus it verifies both equalities approximately during training, and exactly at the end of training. Furthermore, as the original normalized L^2 distance formulation of the BYOL loss (with denominator) is invariant by strictly positive scaling of the linear predictor, we will also neglect scaling of the predictor from now on. Hence we can assume its maximal eigenvalue to be 1 at all times by spectral normalization, and all of its eigenvalues to be close either to 1 or 0 (see Figures 3a and 3c), with the evolution of the predictor rank being a main object of interest. We now show that if both network branches in BYOL fully co-adapt at the end of training, the optimal linear predictor becomes full-rank, i.e., the identity:

Proposition 2. Assume a linear additive feature noise for each batch $Z'_\xi = Z_\theta + \sigma_t \cdot G$, with G a random noise matrix whose entries we can hypothesize are i.i.d. with respect to each other and through time, and σ_t a time-dependent, scalar variance chosen so that $|||G||| = 1$. If additionally $\left|\left|\left| Z_\theta - Z'_\xi \right|\right|\right| = \sigma_t \to 0$ at convergence of BYOL training, then the linear predictor converges to the identity matrix, asymptotically.

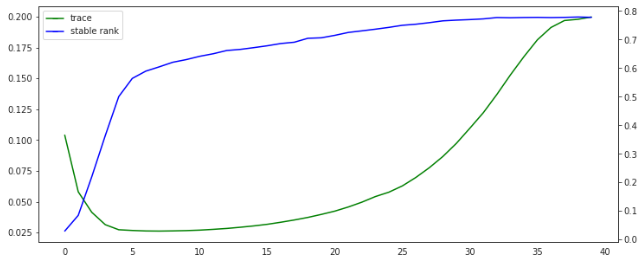


Figure 1. Normalized stable rank (blue) and trace (green) of the BYOL linear trainable predictor, throughout training on ImageNet-1k for 100 epochs. Average over 3 random seeds.

Proof: From now on, our proofs are deferred to Appendix.

3.2. Empirical evolution of predictor rank

Proposition 2 shows the predictor can reach the full-rank identity matrix at the end of training, in the idealization that the training loss effectively goes to 0. We thus seek to observe the evolution of the linear predictor rank during training. Since eigenvalues of small magnitude (compared to numerical precision) might make the rank hard to identify, we use proxy estimates. The *stable rank* (Rudelson and Vershynin, 2007) of a matrix M—a lower bound on its standard rank—is defined as $\mathrm{srank}(M) = \|M\|_F^2 / |||M|||^2$, and for an orthogonal projection P is just $\|P\|_F^2 = \mathrm{tr}(PP^\top) = \mathrm{tr}(P)$. An alternative estimate for the rank of an approximate projection is also its trace. We empirically find that both rank metrics, stable rank and trace, increase during training (Figure 1), with the stable rank generally increasing *monotonically*, showing expansion of the subspace span of the predictor. In particular, the eigenvalues of the rescaled predictor converge to either 0 or 1, with a greater fraction of the latter as training progresses. We can use this observation on the rank deficiency of the batchwise optimal predictor to derive a new fixed-form predictor.

3.3. A new fixed-form predictor

Intuitively the rank deficiency of the predictor early on in training—and potentially the associated batch covariance matrix $(Z_\theta^\top Z_\theta)$—is one reason that can make the matrix inversion in the LRP predictor (Equation 4) numerically unstable, and hinder performance. This suggests sidestepping this issue by using a Taylor-like expansion for the matrix inverse instead, leading to a new fixed-form batchwise predictor:

Proposition 3. The expression

$$P_\theta := 2 \cdot Z_\theta^\top Z_\xi' - Z_\theta^\top Z_\theta Z_\theta^\top Z_\xi' \tag{6}$$

gives another batchwise, approximately optimal predictor. As it is obtained by Neumann expansion $\left(Z_\theta^\top Z_\theta\right)^{-1} \approx 2I - Z_\theta^\top Z_\theta + \cdots$, we name this the **NE** predictor.

If the magnitude of latent elements in batches is small, e.g. with well-chosen normalization, the fourth-order term $Z_\theta^\top Z_\theta Z_\theta^\top Z_\xi'$ in Equation 6 can be considered small, so that the optimal linear predictor reduces to $2 \cdot Z_\theta^\top Z_\xi'$. Our NE predictor can therefore also be interpreted as generalizing the *DirectCopy* (Wang et al., 2021) predictor with an added correction term providing performance benefits, as we will see in Section 6.

4. A spectral view of linear BYOL

In this section we take a tour of BYOL through a spectral decomposition lens in order to characterize more precisely what it learns. We begin with showing that the optimal linear predictor, conditionally on the latents *and its own rank* k, performs k-PCA on latents, identifying the leading k eigendirections of the covariance of latents. We then prove that under additional linear features assumption, the linear encoder features are decorrelated at the end of training. This view justifies taking the matrix square root of the empirical covariance as a predictor; accordingly we propose two new predictor iterations.

4.1. Spectral view of the rank-conditioned predictor

The following proposition highlights what an optimal linear predictor learns, conditionally to its rank *and* the latents coming from the encoder and projector networks:

Proposition 4. Assume no stochastic augmentations or EMA operation ($\tau = 1, \xi = \mathrm{sg}(\theta)$). Further, assume P to be optimal and thus **exactly** an orthogonal projection batchwise. Conditionally on the latents **and** the rank of the predictor, k, the BYOL inner loss solves a k-PCA problem. The predictor's span is then given by the top-k eigenvectors of the latents' covariance matrix, $\mathbb{E}[Z_\theta^\top Z_\theta]$.

A linear optimal predictor P_θ^* in that restricted case performs $\mathrm{rank}\, P_\theta^*$-PCA on the covariance of latents, in particular finding mutually orthogonal eigenvectors. At the end of training, the predictor becomes full rank. This refines our understanding of the optimal linear predictor as an orthogonal projection, by characterizing the directions of its span.

4.2. Decorrelation of linear encoder features

The orthogonality of directions learned by the linear predictor extends to linear encoder features. At the end of training, the linear predictor is the identity (Proposition 2). This fact can be leveraged thanks to the *balancing relationship* (Tian et al., 2021), valid in the presence of a small weight decay parameter λ:

$$A_{\theta,t} A_{\theta,t}^\top = P_{\theta,t}^\top P_{\theta,t} + M_0 e^{-2\lambda t} \tag{7}$$

where $A_{\theta,t}$ and $P_{\theta,t}$ are now indexed by time t, and M_0 is a constant matrix determined at initialization. At the end of training the M_0 term has decayed exponentially and became negligible. In particular, since $P_{\theta,t} = I_f$ by then, we get that at the end of training, $A_{\theta,t} A_{\theta,t}^\top \to I_f$; encoder features are therefore decorrelated.

4.3. Covariance square-root predictors

A method for performing non-contrastive self-supervised learning is to use a predictor that directly computes a *matrix square root* of the covariance matrix of embeddings. This reflects the decorrelation of features seen above and was suggested in prior work, either using eigendecomposition (Tian et al., 2021) or Cholesky factorization towards optimal whitening (Ermolov et al., 2021). It can be explained simply in our setting. Recall our batchwise NE predictor (Equation 6) can be, when neglecting higher-order terms, made directly proportional to the covariance of online and target features, $P_\theta = Z_\theta^\top Z_\xi'$. Since for an optimal, orthogonal projection, predictor $P_\theta^2 = P_\theta$, we get $P_\theta^2 = Z_\theta^\top Z_\xi'$, justifying using a square root of the covariance of features as an alternative predictor. We show how to do this efficiently.

Approximate matrix square-root. Exact matrix square root methods through eigen- or Cholesky decomposition are computationally costly, potentially unstable, and do not exploit that the linear predictor only needs to be approximately optimal, as experiments in Appendix I of Grill et al. (2020) show. We therefore propose to use approximate matrix square root iterations, applied to the empirical covariance of *online* latents $\Sigma = 1/b \cdot (Z_\theta^\top Z_\theta)$, *in lieu* of a trainable linear predictor. Given batchwise covariance matrix Σ, these methods approach $\Sigma^{1/2}$, defined as $(\Sigma^{1/2})^\top \Sigma^{1/2} = \Sigma$, as iterations increase. We choose those iterations to be inverse-free, and involve exclusively matrix products and accumulation, so as to better leverage hardware accelerators, and be numerically stable.

Visser and Newton-Schulz iterations. The two iterations we consider are the *Visser* iteration (Higham, 2008) and the *Newton-Schulz* iteration (Schulz, 1933; Denman and Beavers, 1976; Higham, 2008). We detail them below.

Proposition 5. *(Visser)* Set a maximum number n of iterations, and a positive step size η. For each online latents batch Z_θ, compute the matrix $\Sigma = \frac{1}{b} Z_\theta^\top Z_\theta$, and set $P_0 = \frac{1}{2\eta} I_f$. Then repeat n times the following *Visser* iteration for the predictor P_k:

$$P_{k+1} = P_k + \eta \cdot \left(\Sigma - P_k^2\right) \tag{8}$$

Then $P_k \underset{k}{\rightarrow} \Sigma^{1/2} = \frac{1}{\sqrt{b}}(Z_\theta^\top Z_\theta)^{1/2}$. Use $P_\theta := P_n$ as batchwise predictor. We name this the **Visser** predictor.

In practice, the Visser predictor needs several dozens iterations to achieve good results, and requires choosing an additional step size parameter η. By contrast, the *Newton-Schulz* iteration we next present does not have this requirement, and provides faster convergence as a second-order method.

Proposition 6. *(Newton-Schulz)* Set a maximum number n of iterations. For each online latents batch Z_θ, first set $\Sigma = \frac{1}{b} Z_\theta^\top Z_\theta, \quad A_0 = \Sigma / \|\Sigma\|_F, \quad B_0 = I_f$, and then repeat n times the coupled matrix iteration in (A_k, B_k)

$$\begin{cases} A_{k+1} = 1/2 \cdot A_k\big(3I_f - B_k A_k\big) & (9) \\ B_{k+1} = 1/2 \cdot \big(3I_f - B_k A_k\big) B_k & (10) \end{cases}$$

Then $A_k \underset{k}{\rightarrow} A_0^{1/2} = (Z_\theta^\top Z_\theta)^{1/2} / \sqrt{b\,\|\Sigma\|_F}$.
Use $P_\theta := A_n \cdot \sqrt{\|\Sigma\|_F}$ as batchwise predictor.
We name this the **Newton-Schulz** (**NS**) predictor.

We also note that iterate B_k converges to a multiple of $\Sigma^{-1/2}$, which will be useful later.

This iteration only requires $3 \cdot n$ matrix multiplications. Moreover, as a second-order method, it is robust to sensible choices of n. Higham (2008) discusses the theoretical and convergence properties of the two iterations in Propositions 5 and 6. In our self-supervised learning context, these fixed-form predictors recover and can even exceed the performance of a linear trainable predictor (cf. Section 6).

NS^2. Finally, given the projection relationship $P^2 \approx P$ for the approximately optimal linear predictor also implies that $P \approx P^{1/2} \approx P^{1/4} \approx \dots$, consistently with findings in Wang et al. (2021), one can chain two successive applications of the Newton-Schulz iteration and use this as a predictor, now computing an approximation to $(Z_\theta^\top Z_\theta)^{1/4}$. In practice, we found this provides small performance gains. We call this method NS^2, and evaluate it empirically in Results section 6. We however only chain Newton-Schulz iterations since we found that the chaining of two Visser iterations is prone to numerical collapse.

5. A Riemannian interpretation of `BYOL`

This section views `BYOL` and its induced orthogonality from a Riemannian geometry perspective. The optimal linear predictor is orthogonal, and can hence be seen as undergoing optimization on a *Stiefel* manifold of orthogonal matrices, whose definition we recall below. We already know that conditionally on its rank k and the latents, the optimal predictor solves a k-PCA problem (Proposition 4). In the special case $k = 1$, `BYOL` can be seen as an exact instance of a 1-PCA method called *Krasulina*'s method. Extending this equivalence to $k \geq 1$ is possible with Riemannian interpretation. We show how orthogonality constraints in the predictor emerge from the architectural combination of symmetry and linear regression, corresponding to the notion of *retraction* in Riemannian SGD. It also corresponds to the addition of negative gradients, precluding training collapse. We also derive another predictor form from those insights.

5.1. 1-PCA

Connection with Oja's and Krasulina's methods. When trying to perform whitening or PCA, for large datasets, the sample covariance of data might differ from the true, population covariance; also, data may only be available incrementally. Algorithms have been devised to deal with this streaming setting. *Oja's rule* (Oja, 1982; 1992), maintains an estimate p_t of the *leading* eigenvector of the covariance $\mathbb{E}[X^\top X]$, as given a realization of single random row vector X, it alternates between a gradient ascent update towards $X^\top X$ and a normalization update for p_t according to

$$p_t \leftarrow p_{t-1} + \eta\left(X^\top X p_{t-1}\right) \text{ and } p_t \leftarrow \frac{p_t}{\|p_t\|} \qquad (11)$$

Krasulina's method (Krasulina, 1969) performs a similar update, that folds w_t normalization into an additional term:

$$p_t \leftarrow p_{t-1} + \eta\left(X^\top X p_{t-1} - p_{t-1}\left(X\frac{p_{t-1}}{\|p_{t-1}\|}\right)^2\right) \qquad (12)$$

Both methods are known to perform 1-PCA in the limit, and their convergence properties and rates are well studied (Shamir, 2014; Jain et al., 2016). In the `BYOL` case, this gives us an analogy when the linear predictor is constrained to be exactly of rank 1, showing it learns the leading eigendirection first:

Proposition 7. Assume no stochastic augmentations or EMA operation ($\tau = 1, \xi = \operatorname{sg}\theta$). The `BYOL` predictor optimization loop conditional on latents, when constrained to operate with a rank-1 predictor, performs exactly Krasulina's update.

Proposition 7 strengthens the Proposition 4 in the special case $k = 1$. The case $k = 1$ is illuminating but begs for generalization to larger ranks. We now proceed to extend this to $k \geq 1$.

5.2. A Riemannian interpretation

The Oja and Krasulina updates can be adapted to $k \geq 1$, now maintaining a rank k matrix estimate P_t, provided the explicit p_t vector normalization steps in Equation 11 are replaced with an *orthonormalization step* for matrix P_t. This can be achieved by various means, including taking the orthogonal part in QR factorization of P_t after each update, and ensures the procedure does not collapse. Tang (2019) for instance propose a matrix version of the Krasulina procedure this way.

Riemannian SGD. This general procedure of taking gradient updates towards an empirical covariance matrix, interleaved with orthonormalization steps, can be interpreted in another way. If it is acceptable to make the orthonormalization procedure approximate, we can see this as an instance of stochastic *Riemannian gradient ascent* (Bonnabel, 2013) as follows. Enforcing orthonormality constraints in matrix optimization is equivalent to solving an unconstrained problem over a *Stiefel* matrix manifold (Edelman et al., 1998; Absil et al., 2007) defined for integers (n, p), $p \leq n$ as the manifold $\mathrm{St}(n, p)$, of matrices M such that $\left\{M \in \mathbb{R}^{n \times p} : M^\top M = I_p\right\}$. A point on the manifold is thus a set of p orthonormal vectors in $\mathbb{R}^n$ (a p-frame) and can be seen as the given of a dimension p subspace, as well as a choice of its basis vectors.

Riemannian gradient ascent proceeds by taking gradient steps, projected in the tangent space of the manifold at each step. However, because the gradient iterates remain in the tangent plane as they neglect curvature terms, they need to be systematically taken back (arbitrarily close) to the manifold itself through an operation called *retraction*, $\mathcal{R}$, after each step. This results in a sequence of matrix iterates (M_k) of the form, given step size η, and gradients of function F,

$$M_{k+1} = \mathcal{R}\big(M_k, \eta \,\mathrm{Grad}\, F(M_k)\big) \tag{13}$$

Retractions are generally computationally expensive matrix functions (Ablin and Peyré, 2022), and a large part of matrix manifold optimization consists in devising efficient retraction algorithms (Absil et al., 2007).

EMA and linear predictor as trivial retraction. Recall that the optimal linear predictor in BYOL is an orthogonal projection (Equation 5). We link this to the BYOL architecture with elementary arguments. A projection is an idempotent linear application P, such that $P^2 - P = 0$, and an orthogonal projection satisfies $PP^\top - P = 0$, so a symmetric $(P = P^\top)$ projection is also orthogonal. These properties can be made approximate given some scalar $\varepsilon \geq 0$; if defining an ε-*quasi*-projection as satisifying $\left|\!\left|\!\left|P^2 - P\right|\!\right|\!\right| \leq \varepsilon$, an ε-quasi-symmetry $\left|\!\left|\!\left|P^\top - P\right|\!\right|\!\right| \leq \varepsilon$, and an ε-*quasi*-orthogonal projection $\left|\!\left|\!\left|PP^\top - P\right|\!\right|\!\right| \leq \varepsilon$. An ε-symmetric ε-projection is thus a $(2\cdot)\varepsilon$-orthogonal projection.

We now observe crucially that a close-to-optimal linear regression predictor in BYOL should be close to a projection, and an exponential moving average operation on the representations with $\tau \simeq 1$ should yield a quasi-symmetric predictor. Importantly the EMA update step is done after the stop-gradient operation. The interleaved gradient updates-EMA steps in BYOL are thus analogous to gradient updates-retraction steps in Riemannian SGD. Hence,

$$\text{A } \underbrace{\text{quasi-symmetric}}_{\text{tight EMA}} \ \underbrace{\text{quasi-projection}}_{\text{linear regression}} \text{ is } \underbrace{\text{quasi-orthogonal}}_{\text{Stiefel retraction}}.$$

i.e. the coupling of the EMA and predictor make them an approximate orthonormalization step for the latents; **given a close to optimal linear predictor, the additional EMA update makes it a computationally free Stiefel retraction, on $\mathrm{St}(f, \mathrm{rank}\, P)$.**

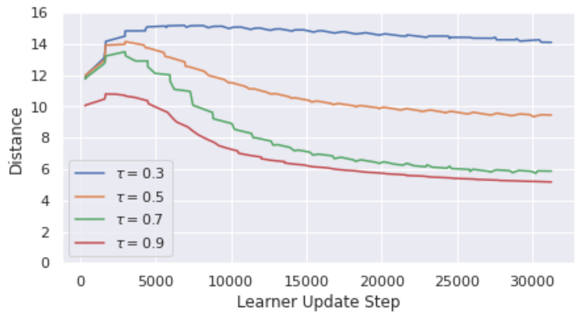


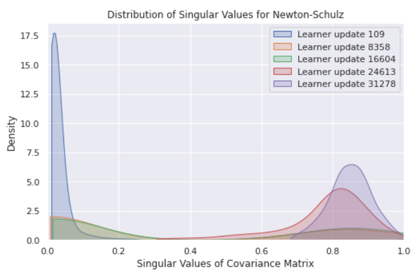
Figure 2. Distance of the linear predictor to its polar factor, as a function of the EMA parameter τ in BYOL. Smaller EMA values tend to result in higher terminal distances. ImageNet, 100 epochs.

We empirically verify this analogy by observing (Figure 2) that when the EMA parameter τ is decreased away from 1, the distance between the linear predictor P and its polar factor U in polar decomposition $P = UH$, fails to decrease throughout training. U is unitary, so this distance is representative of how far P is to the full-rank Stiefel manifold. Hence the EMA operation enables positioning close to successive Stiefel manifolds. Our view thus replaces and refines the interpretation of BYOL and SimSiam as expectation-maximization algorithms (Chen and He, 2020).

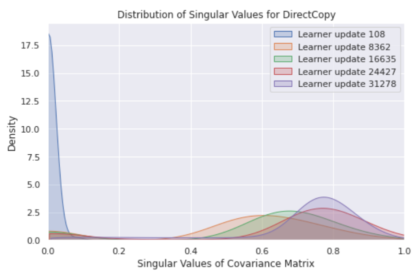
Towards a family of orthonormalizing SSL methods. This analogy would also enable us to replace the joint operation of a learnable predictor and an EMA in BYOL with any known closed-form Stiefel retraction, or outright projection, operating on the covariance of latents, deriving multiple self-predictive unsupervised learning methods at will. This uncovers a dimension of trading off complexity for number of learning epochs at given terminal performance, opening possibilities for leaner, sample-efficient self-supervised learning. We propose the simplest of these methods below.

Stiefel projection. For any matrix M (square or not) it is known that the matrix $(M^\top M)^{-1/2} M^\top$ represents the projection of M on the Stiefel manifold. We can use an approximation to this expression instead of a retraction, applied to the covariance of online features to orthogonalize it, as a fixed-form predictor. Since the sequence of matrices (B_k) in the Newton-Schulz iteration (Proposition 6) allows estimating the *inverse* matrix square root $\Sigma^{-1/2}$, we have:

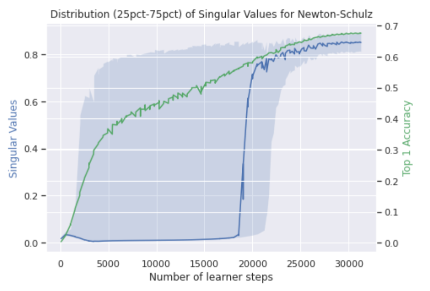
Proposition 8. Set a maximum number n of iterations. For each online latents batch Z_θ, first compute the matrix $\Sigma = \frac{1}{b} Z_\theta^\top Z_\theta$. Then compute an estimate to $(\Sigma^\top \Sigma)^{-1/2} \Sigma^\top$ using B_n after applying n Newton-Schulz iterations to $(\Sigma^\top \Sigma)$, as per Proposition 6. Use $P_n := B_n \Sigma^\top / \left\|B_n\right\|_F$ as a batchwise predictor. We name this the **Stiefel** predictor.



(a) Empirical distribution of the singular values of the covariance of features, for Newton-Schulz, throughout 100 epochs.



(b) Empirical distribution of the singular values of the covariance of features, for DirectCopy, throughout 100 epochs.



(c) *Blue:* Median, 25th and 75th percentile of empirical distribution as in leftmost Figure 3a. *Green:* Top-1 % linear accuracy.

Figure 3. Study of the (dynamically renormalized) singular values of the covariance of online latents.

Riemannian SGD can create negative gradients. Finally we show how the Riemannian perspective helps explain non-collapse in negative-free SSL. Performing Riemannian gradient descent (for the predictor) is done by first computing *Riemannian* gradients. Those are standard gradients projected onto the tangent plane to the manifold at every iterate. Riemannian gradients on the Stiefel manifold contain an additional term relative to standard gradients. The following proposition is given in Edelman et al. (1998):

Proposition 9. Let F be a function of a matrix P and denote by $F'(P)_{ij} = (\frac{\partial F}{\partial P_{ij}})$ the Jacobian (matrix gradient) of F with respect to P. The Riemannian gradient of $F(P)$ on the Stiefel manifold, $\mathrm{Grad}\, F(P)$, is given by

$$\mathrm{Grad}\, F(P) = F'(P) - PF'(P)^\top P.$$

The additional term $-PF'(P)^\top P$ can thus possibly play the same role as standard negative terms intervening explicitly in contrastive losses. This helps provide intuition on why orthonormality constraints on the predictor can prevent its collapse, and in turn, any collapse of features due to eigenalignment.

Summary of our fixed-form predictors. For clarity, we show a summary of the batchwise predictors we propose Table 1. We then proceed to evaluate these empirically.

Fixed-form batchwise predictor	Computes P_θ as
LRP (Grill et al., 2020)	$\left(Z_\theta^\top Z_\theta\right)^{-1} Z_\theta^\top Z'_\xi$
DirectPred (Tian et al., 2021)	$(Z_\theta^\top Z_\theta)^{1/2}$ (eigendecomp.)
DirectCopy (Wang et al., 2021)	$Z_\theta^\top Z_\theta$
NE (Prop. 3)	$2 \cdot Z_\theta^\top Z'_\xi - Z_\theta^\top Z_\theta Z_\theta^\top Z'_\xi$
Visser (fast iteration) (Prop. 5)	$(Z_\theta^\top Z_\theta)^{1/2}$
NS (fast iteration) (Prop. 6)	$(Z_\theta^\top Z_\theta)^{1/2}$
Stiefel (fast iteration) (Prop. 8)	$(Z_\theta^\top Z_\theta Z_\theta^\top Z_\theta)^{-1/2}(Z_\theta^\top Z_\theta)$

Table 1. Summary of fixed-form predictors.

6. Results

Experimental protocol. In this section we follow the experimental protocol of BYOL in Grill et al. (2020), except we use a linear predictor P, taken as a fixed-form expression defined batchwise (see Table 1), instead of a trainable one. Importantly, we use an EMA on the predictor to stabilize its numerics. This aside, we use the same ResNet-50 encoder with a 2-layers projector, and optimization settings (including learning rate and weight decay) as in Grill et al. (2020). We report top-1 percentage accuracy on ImageNet-1k under linear evaluation protocol, after either 100 or 300 epochs, averaged over 3 random seeds. More details and hyperparameters are presented in Appendix B.1.

Predictor Ridge. When estimating the matrix predictor through closed-form expressions or iterations, it is possible to add an extra regularization or *ridging* term. It consists in adding α times the identity matrix to the predictor. First, it provides additional numerical stability. Second, it brings the predictor *closer* to the orthogonal manifold, which we have shown to be an important property in the predictor. We report our results with α values between $+0.0$ and $+0.9$ in $+0.15$ increments (Tables 2 and 3).

BYOL baseline and main results. The top-1 accuracy reached by standard BYOL with a linear *trainable* predictor is 67.6% at 100 epochs and 71.5% at 300 epochs. Our best results achieved with fixed-form predictors (with additional ridge) are 68.8% and 72.2% respectively, with both the NS predictor (in its NS^2 variant) and the Stiefel predictor actually outperforming the linear trainable variant, by $+1.2\%$ and $+0.7\%$ respectively. At 300 epochs, the performance of the NS and Stiefel predictors remains stable across a broad range of ridge parameters α as soon as $\alpha > 0$, removing the need to tune it precisely. We show our performance at 100 epochs versus prior art in Table 4. We reach the best performance across the board with 68.8%, along with *DirectCopy* for a specific ridge value of $\alpha = 0.01$, speaking to the necessity of tuning that parameter very precisely there.

By contrast, adding a corrective term via our NE predictor is enough to improve performance in a wide α range compared to DirectCopy, and function well even without ridging, or tuning predictor setting frequency. Our NS^2 and Stiefel predictors perform the best both at 100 and 300 epochs. Overall, these positive results of our fixed-form predictors empirically validate our analyses for each form of the BYOL predictor.

Predictor	0.0	0.15	0.3	0.45	0.6	0.75	0.9
DirectCopy[†]	20.9	65.8	67.3	67.4	67.5	67.4	67.4
NE	61.2	<u>68.5</u>	68.3	68.1	67.7	67.4	67.0
Visser	<u>68.4</u>	68.3	68.4	68.4	68.3	68.4	68.3
NS ($n = 9$)	66.4	66.4	66.7	67.1	67.6	67.8	<u>68.0</u>
NS^2 ($n = 7$)	66.8	68.2	68.6	68.6	68.7	68.7	<u>**68.8**</u>
Stiefel ($n = 9$)	66.7	67.7	<u>**68.8**</u>	68.7	68.7	68.7	68.7

Table 2. Top-1 accuracy on ImageNet-1k (linear protocol) of our predictors, at 100 epochs, as ridge α varies. *DirectCopy* (Wang et al., 2021) is our own replication, setting predictor every batch. Best results per predictor underlined, best overall bolded. Average over 3 random seeds. Standard trainable linear predictor BYOL achieves 67.6%.

Predictor	0.0	0.15	0.3	0.45	0.6	0.75	0.9
DirectCopy[†]	48.3	71.1	71.4	71.3	71.2	71.5	71.4
NE	68.8	<u>71.8</u>	71.7	71.5	71.3	70.9	70.6
Visser	<u>71.6</u>	71.5	71.6	71.4	71.5	71.4	71.6
NS , $n = 9$	70.8	71.7	<u>72.0</u>	72.0	71.9	72.0	71.9
NS^2 , $n = 7$	70.9	72.0	72.1	72.0	<u>**72.2**</u>	72.1	72.1
Stiefel , $n = 9$	71.4	72.0	<u>72.1</u>	72.1	72.0	72.0	72.1

Table 3. Top-1 accuracy on ImageNet-1k (linear protocol) of our predictors, at 300 epochs, as ridging α varies. Average over 3 seeds. Standard trainable linear predictor BYOL achieves 71.5%.

Method	Accuracy
SimSiam (Chen and He, 2020)	68.1
Barlow Twins (Zbontar et al., 2021)	63.4
VICReg (Bardes et al., 2022)	65.3
Liu et al. (2022) (no covariance partitioning)	61.4
Liu et al. (2022) (covariance partitioning)	67.3
DirectPred (Tian et al., 2021)	68.5
DirectCopy (Wang et al., 2021), $\alpha = 0.01$	**68.8**
Neural Eigenmap (Deng et al., 2022)	68.4
Visser (ours)	68.4
NS^2 (ours)	**68.8**
Stiefel (ours)	**68.8**

Table 4. Top-1 accuracy on ImageNet-1k (linear protocol), at 100 epochs, comparing prior art (with standard, rather than higher capacity, projector network architecture) and our best results.

Performance of square-root iterations. The Visser, NS, and Stiefel predictors all perform better than the BYOL baseline. Consistently with its initialization as a large multiple of the identity matrix, the Visser iteration is insensitive to ridging, and can be used with $\alpha = 0$. We also evaluate the NS^2 variant of the Newton-Schulz iteration, chaining two of its successive applications, but with two times 7 iterations instead of 9 for the single iteration. We see performance gains from this approach (68.8% instead of 68.0% at 100 epochs) and hypothesize that they stem from better pre-conditioning of the covariance of features, yielding richer eigendirections to explore. Finally, the Stiefel predictor shows excellent performance both at 100 and 300 epochs.

Predictor singular values. We observe the empirical distribution of singular values of the latents' covariance throughout 100 epochs of training. We continously normalize those to be between 0 and 1, and observe an average of 3 seeds. We do this across two predictors: Newton-Schulz (Figures 3a and 3c), and DirectCopy (Wang et al., 2021) (Figure 3b). Empirically we observe better separation of eigenvalues with our Newton-Schulz predictor, consistently with the robustness and performance differentials highlighted above. Additional depictions for our other predictors and ablation studies on the number of iterations n for our approximate methods are provided in Appendix C.2. The robustness of the Stiefel predictor and its strong performance across epochs and α makes it the best method to use, in our view.

7. Related work and conclusion

We have sought to present minimum theoretical arguments towards understanding self-predictive unsupervised methods, using as few mathematical assumptions as possible. Previous attempts at such explanation have involved implicit contrastive properties of normalizations later disproved empirically (Richemond et al., 2020), or also relied on strong assumptions such as isotropy and multiplicative EMA (Tian et al., 2021) and on differential equation tools for their conclusions.

Instead, we focused on understanding the multiple forms of the BYOL predictor network, forming a bridge to existing methods (Table 1). The proximity of the linear predictor to an orthogonal projection (with increasing subspace span throughout training, uncovering more eigendirections of the covariance of latents until the identity matrix is reached) is a unifying insight. This helps explain existing methods with very simple algebraic arguments and allows us to derive new ones. Therefore we introduce multiple fixed-form predictor variants at once, in contrast with prior work (Bardes et al., 2022; Zbontar et al., 2021; Liu et al., 2022). Our Riemannian manifold perspective complements that of Balestriero and LeCun (2022) and also enables deriving additional methods at will, through Stiefel retractions. We operationalized our understanding with efficient implementations of our predictors that utilize fast matrix iterations, new to the self-supervised learning context. These fixed-form predictors perform strongly, offer better insights on the representations learned by self-supervised methods, and help further explain the importance of the predictor network.

References

Pierre Ablin and Gabriel Peyré. Fast and accurate optimization on the orthogonal manifold without retraction. In *International Conference on Artificial Intelligence and Statistics*, pages 5636–5657. PMLR, 2022.

Pierre-Antoine Absil, Robert E. Mahony, and Rodolphe Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2007.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a Visual Language Model for Few-Shot Learning. *ArXiv*, abs/2204.14198, 2022.

Anonymous. Towards a unified theoretical understanding of non-contrastive learning via rank differential mechanism. *OpenReview*, ICLR 2023.

Randall Balestriero and Yann LeCun. Contrastive and non-contrastive self-supervised learning recover global and local spectral embedding methods. *NeurIPS*, 2022.

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. *ICLR*, 2022.

Silvére Bonnabel. Stochastic Gradient Descent on Riemannian Manifolds. *IEEE Transactions on Automatic Control*, 58:2217–2229, 2013.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, and Skye Wanderman-Milne. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

J. Bromley, James W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, Eduard Säckinger, and R. Shah. Signature Verification Using A "Siamese" Time Delay Neural Network. In *Int. J. Pattern Recognit. Artif. Intell.*, 1993.

Jeff Cheeger. A lower bound for the smallest eigenvalue of the Laplacian. *Problems in Analysis*, 1969.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

Xinlei Chen and Kaiming He. Exploring Simple Siamese Representation Learning. *ArXiv*, abs/2011.10566, 2020.

Zhijie Deng, Jiaxin Shi, Hao Zhang, Peng Cui, Cewu Lu, and Jun Zhu. Neural eigenfunctions are structured representation learners. *ArXiv*, abs/2210.12637, 2022.

Eugene D. Denman and Alex N. Beavers. The matrix sign function and computations in systems. *Applied Mathematics and Computation*, 2:63–94, 1976.

Alan Edelman, Tomás A. Arias, and S.T. Smith. The Geometry of Algorithms with Orthogonality Constraints. *SIAM J. Matrix Anal. Appl.*, 20:303–353, 1998.

Aleksandr Ermolov, Aliaksandr Siarohin, E. Sangineto, and N. Sebe. Whitening for Self-Supervised Representation Learning. In *ICML*, 2021.

Jean-Bastien Grill, Florian Strub, Florent Altché, C. Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, B. A. Pires, Zhaohan Daniel Guo, M. G. Azar, Bilal Piot, K. Kavukcuoglu, R. Munos, and Michal Valko. Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning. In *NeurIPS*, 2020.

Manu Srinath Halvagal, Axel Laborieux, and Friedemann Zenke. An eigenspace view reveals how predictor networks and stop-grads provide implicit variance regularization. *NeurIPS Self-Supervised Learning Workshop*, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

Nicholas John Higham. *Functions of Matrices - Theory and Computation*. SIAM, 2008.

Roger A. Horn and Charles R. Johnson. Matrix analysis. In *Statistical Inference for Engineers and Data Scientists*, 2013.

Prateek Jain, Chi Jin, Sham M. Kakade, Praneeth Netrapalli, and Aaron Sidford. Streaming PCA: Matching matrix bernstein and near-optimal finite sample guarantees for oja's algorithm. In *Annual Conference Computational Learning Theory*, 2016.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, 2021.

Effrosini Kokiopoulou, Jie Chen, and Yousef Saad. Trace optimization and eigenproblems in dimension reduction methods. *Numerical Linear Algebra with Applications*, 18, 2011.

T. P. Krasulina. The method of stochastic approximation for the determination of the least eigenvalue of a symmetrical matrix. *USSR Computational Mathematics and Mathematical Physics*, 9:189–195, 1969.

Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

Kang-Jun Liu, Masanori Suganuma, and Takayuki Okatani. Bridging the gap from asymmetry tricks to decorrelation principles in non-contrastive self-supervised learning. *NeurIPS*, 2022.

Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15: 267–273, 1982.

Erkki Oja. Principal components, minor components, and linear neural networks. *Neural Networks*, 5:927–935, 1992.

Pentti Paatero and Unto Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values†. *Environmetrics*, 5:111–126, 1994.

Karl Pearson. LIII. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine Series 1*, 2:559–572, 1901.

David Pfau, Stig Petersen, Ashish Agarwal, David G. T. Barrett, and Kimberly L. Stachenfeld. Spectral Inference Networks: Unifying Deep and Spectral Learning. In *ICLR*, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In *ICML*, 2021.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *ArXiv*, abs/2204.06125, 2022.

Pierre H. Richemond, Jean-Bastien Grill, Florent Altché, C. Tallec, Florian Strub, Andrew Brock, Samuel L. Smith, Soham De, Razvan Pascanu, Bilal Piot, and Michal Valko. BYOL works even without batch statistics. *ArXiv*, abs/2010.10241, 2020.

M. Rudelson and R. Vershynin. Sampling from large matrices: An approach through geometric functional analysis. *J. ACM*, 54:21, 2007.

Günther Schulz. Iterative berechung der reziproken matrix. *Zamm-zeitschrift Fur Angewandte Mathematik Und Mechanik*, 13:57–59, 1933.

Uri Shaham, Kelly Stanton, Haochao Li, Boaz Nadler, Ronen Basri, and Yuval Kluger. SpectralNet: Spectral Clustering using Deep Neural Networks. *ArXiv*, abs/1801.01587, 2018.

Ohad Shamir. A stochastic PCA and SVD algorithm with an exponential convergence rate. *ArXiv*, abs/1409.2848, 2014.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 731–737, 1997.

Cheng Tang. Exponentially convergent stochastic k-PCA without variance reduction. In *NeurIPS*, 2019.

Yuandong Tian, Xinlei Chen, and S. Ganguli. Understanding Self-Supervised Learning Dynamics without Contrastive Pairs. *ICML*, 2021.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding. *arXiv preprint arXiv:1807.03748*, 2018.

Xiang Wang, Xinlei Chen, Simon Shaolei Du, and Yuandong Tian. Towards demystifying representation learning with non-contrastive self-supervision. *ArXiv*, abs/2110.04947, 2021.

Xi Weng, Lei Huang, Lei Zhao, Rao Muhammad Anwer, Salman Khan, and Fahad Shahbaz Khan. An investigation into whitening loss for self-supervised learning. *arXiv preprint arXiv:2210.03586*, 2022.

Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv: Computer Vision and Pattern Recognition*, 2017.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow Twins: Self-supervised learning via redundancy reduction. In *ICML*, 2021.

Shaofeng Zhang, Feng Zhu, Junchi Yan, Rui Zhao, and Xiaokang Yang. Zero-CL: Instance and feature decorrelation for negative-free symmetric contrastive learning. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=RAW9tCdVxLj`.

Dengkui Zhu, Boyu Li, and Ping Liang. On the matrix inversion approximation based on Neumann series in massive MIMO systems. *2015 IEEE International Conference on Communications (ICC)*, pages 1763–1769, 2015.

Supplementary Material

A. Proof of theoretical results

Proposition 1. The linear optimal predictor, batchwise, is an orthogonal projection.

Proof. Recall that we have on each batch

$$P^\star = \underset{P}{\arg\min} \quad \left\|Z_\theta P - Z'_\xi\right\|_F^2 = \left(Z_\theta^\top Z_\theta\right)^{-1} Z_\theta^\top Z'_\xi \tag{14}$$

We can just insert the value of the optimum in the BYOL objective (right and left term of equation above) yielding

$$\left\|Z_\theta \left(Z_\theta^\top Z_\theta\right)^{-1} Z_\theta^\top Z'_\xi - Z'_\xi\right\|_F^2 = \left\|\left(Z_\theta \left(Z_\theta^\top Z_\theta\right)^{-1} Z_\theta^\top - I_f\right) Z'_\xi\right\|_F^2$$

Here $Z_\theta \left(Z_\theta^\top Z_\theta\right)^{-1} Z_\theta^\top$ is recognized as the linear, orthogonal projection application on the vector space spanned by the rows of batch Z_θ, which is of dimension at most $\min(b, f)$; we denote this projection by $\mathrm{Proj}_{(Z_\theta)}$. Its complement to the identity is also an orthogonal projection, $\mathrm{Proj}_{(Z_\theta)^\perp}$. Thus for an *optimal* linear predictor, on each batch,

$$L_{\text{BYOL},P^*} = \mathbb{E}\left[\left\|\mathrm{Proj}_{(Z_\theta)^\perp}(Z'_\xi)\right\|_F^2\right] \tag{15}$$

□

One added benefit of the orthogonal projector representation, see Eq. (5), is that it enables identifying two sources of noise intervening in BYOL thanks to decomposing Z'_ξ. We additionally have the proposition:

Proposition. The BYOL loss with optimal linear predictor decomposes into two noise terms: resampling noise and EMA noise.

$$L_{\text{BYOL},P^*} = \mathbb{E}\left[\left\|\mathrm{Proj}_{(Z_\theta)^\perp}\left[\underbrace{(Z'_\xi - Z_\xi)}_{\text{resampling noise}} + \underbrace{(Z_\xi - Z_\theta)}_{\text{EMA noise}}\right]\right\|_F^2\right].$$

Proof. By rewriting $Z'_\xi = (Z'_\xi - Z_\xi) + (Z_\xi - Z_\theta) + Z_\theta$ and substituting this expression into Equation 15, where Z_θ then vanishes. $Z'_\xi - Z_\xi$ can be interpreted as resampling noise due to stochastic augmentations, while the difference $(Z_\xi - Z_\theta) + Z_\theta$ can be seen as exponential moving average noise.

This decomposition may help explain why the performance of tied-weights (that is, EMA-less, with $\tau = 1$) versions of BYOL (Grill et al., 2020; Chen and He, 2020) is suboptimal: the vanishing of the second term providing an additional source of gradient for learning. □

Proposition 2. Assume a linear additive feature noise for each batch $Z'_\xi = Z_\theta + \sigma_t \cdot G$, with G a random noise matrix whose entries we can hypothesize are i.i.d. with respect to each other and through time, and σ_t a time-dependent, scalar variance chosen so that $|||G||| = 1$. If additionally $\left|\left|\left|Z_\theta - Z'_\xi\right|\right|\right| = \sigma_t \to 0$ at convergence of BYOL training, then the linear predictor converges to the identity matrix, asymptotically.

Proof. Substituting $Z'_\xi = Z_\theta + \sigma_t \cdot G$ into $P^*_\theta = \left(Z_\theta^\top Z_\theta\right)^{-1} Z_\theta^\top Z'_\xi$ (Equation 4) immediately yields:

$$P^*_\theta = I_f + \sigma_t \left(Z_\theta^\top Z_\theta\right)^{-1} Z_\theta^\top G.$$

and then by assumption $\sigma_t \to 0$ which makes the second term vanish (provided for instance features Z_θ are bounded, which is realized in practice) and gets the result we want of $P^*_\theta \to I_f$.

We can prove a slightly stronger result in this model. In this idealized setting P_θ^* cannot be exactly an orthogonal projector batchwise (as $P_\theta^{*2} \neq P_\theta^*$), but we get a formal control on how much it deviates from that property by upper-bounding $P_\theta^* P_\theta^{*\top} - P_\theta^*$ and writing

$$\frac{\left\lvert\!\left\lvert\!\left\lvert P_\theta^* P_\theta^{*\top} - P_\theta^* \right\rvert\!\right\rvert\!\right\rvert}{\lvert\!\lvert\!\lvert P_\theta^* \rvert\!\rvert\!\rvert} \leq \sigma_t \lvert\!\lvert\!\lvert Z_\theta \rvert\!\rvert\!\rvert^{-1} \to 0 \quad \text{if } Z_\theta \text{ bounded.}$$

□

Proposition 3. The expression

$$P_\theta := 2 \cdot Z_\theta^\top Z'_\xi - Z_\theta^\top Z_\theta Z_\theta^\top Z'_\xi \tag{6}$$

gives another batchwise, approximately optimal predictor. As it is obtained by Neumann expansion $\left(Z_\theta^\top Z_\theta\right)^{-1} \approx 2I - Z_\theta^\top Z_\theta + \cdots$, we name this the **NE** predictor.

Proof. We obtain this thanks to $\left(Z_\theta^\top Z_\theta\right)^{-1} \approx 2I - Z_\theta^\top Z_\theta + \cdots$, the leading term of a *Neumann expansion* (Zhu et al., 2015). This is simply substitued into the optimal linear regression predictor $\left(Z_\theta^\top Z_\theta\right)^{-1} Z_\theta^\top Z'_\xi$ (Equation 4). Using only the first term is not only justified at the end of training (when $\left(Z_\theta^\top Z_\theta\right)^{-1} \to I_f$) but also if we can scale the norm of batches Z_θ to be small. This generalizes the usage of $P_\theta^* \approx \mathbb{E}\left[Z_\theta^\top Z'_\xi\right]$, which was separately named *DirectCopy* in (Wang et al., 2021). □

Proposition 4. Assume no stochastic augmentations or EMA operation ($\tau = 1, \xi = \mathrm{sg}(\theta)$). Further, assume P to be optimal and thus **exactly** an orthogonal projection batchwise. Conditionally on the latents **and** the rank of the predictor, k, the BYOL inner loss solves a k-PCA problem. The predictor's span is then given by the top-k eigenvectors of the latents' covariance matrix, $\mathbb{E}[Z_\theta^\top Z_\theta]$.

Proof. With a slight abuse of notation, we identify P and any matrix in $\mathbb{R}^{k \times f}$ with k non-null row vectors spanning $\operatorname{Im} P$. Since $P = P^\top P$ as it is exactly an orthogonal projection, we can rewrite the BYOL loss minimization as

$$\min_{P, PP^\top = I_k} \mathbb{E} \left\| Z_\theta - P Z_{\mathrm{sg}(\theta)} \right\|^2 = \min_{P, PP^\top = I_k} \mathbb{E} \left\| Z_\theta - P^\top P Z_{\mathrm{sg}(\theta)} \right\|^2 \tag{16}$$

$$= \min_{P, PP^\top = I_k} \operatorname{tr}\left((\mathrm{I}_d - \mathrm{P}) \, \mathbb{E}\left[\mathrm{Z}_\theta \mathrm{Z}_{\mathrm{sg}(\theta)}^\top \right] \right) \tag{17}$$

which directly reflects the variational characterization of k-PCA.

We also note that this last form sheds light on a surprising empirical observation that parametrizing the linear predictor to be *residual* (He et al., 2016), effectively initializing P at the identity, precludes all learning and makes no progress whatsoever, despite it being mathematically equivalent to the standard formulation of a neural predictor. □

Proposition 7. Assume no stochastic augmentations or EMA operation ($\tau = 1, \xi = \mathrm{sg}\,\theta$). The BYOL predictor optimization loop conditional on latents, when constrained to operate with a rank-1 predictor, performs exactly Krasulina's update.

Proof. Simply recovered by writing the BYOL inner objective (with rank-1 predictor parametrized by row vector p, L^2 normalized without loss of generality), so that $P = \frac{p^\top p}{\|p\|^2}$:

$$\mathcal{L}_{\mathrm{BYOL}} = \mathbb{E} \left\| Z_\theta - \frac{p^\top p}{\|p\|^2} Z_{\mathrm{sg}(\theta)} \right\|^2 \tag{18}$$

and then computing $\nabla_p \mathcal{L}_{\mathrm{BYOL}}$ for this expression, then taking a gradient ascent step $w_t \leftarrow w_{t-1} + \epsilon \cdot \nabla_p \mathcal{L}_{\mathrm{BYOL}}$ (see Tang (2019)). □

B. Details of algorithms and pseudocode

B.1. Hyperparameters

Our data augmentations and evaluation protocols rigorously follow the standards described in (Grill et al., 2020; Chen et al., 2020). Similarly, our hyperparameters are the same as in Grill et al. (2020), except for two differences. The first one is the learning rate and weight decay combination we employ - given shorter training compared to the original 1000 epochs used in BYOL, we choose a LARS (You et al., 2017) optimizer initial learning rate of 0.45 at 100 epochs and 0.3 at 300 epochs, both in conjunction with a weight decay parameter of 1.0×10^{-6} and an initial target decay rate τ set to 0.99.

The second is the addition of an exponential moving average operation applied to the predictor matrix after computation of the relevant predictor formula (and potential extra ridging operation). We found empirically that this EMA operation stabilizes training. The formula we use to update (previous) predictor P given current batchwise iterate P_n is

$$P \leftarrow \rho P + (1-\rho) P_n \tag{19}$$

The value of constant covariance EMA parameter ρ is predictor-dependent, and picked at 0.8 for the LRP predictor, 0.99 for the NE, Visser, NS and NS^2 predictors, and 0.999 for the Stiefel predictor. We did not seek to optimize for a specific temporal schedule for ρ, which might enable unifying its value across all predictors.

Finally, we set the Visser step size (η in Proposition 5) to 0.001.

B.2. Predictors pseudo-code

For ease of reproducibility and clarity of understanding, we below give Python 3 and specifically JAX (Bradbury et al., 2018) pseudo-code that can be used to compute our predictors, reflecting main text equations, as well as any additional scaling factors that we empirically found could help numerical stability (at the margin). Notation-wise, the two lines

```
z_theta = online_out['projection_view1']
z_xi = target_out['projection_view2']
```

correspond to the computation of batches of latents Z_θ and Z'_ξ, respectively. We begin with our 4 proposed formulaic predictors, before also giving, for convenience, a comparable way of computing the LRP predictor from (Grill et al., 2020).

B.2.1. NE PREDICTOR PSEUDO-CODE

```
def ne_predictor(zt, zx):

  zttzx = jax.numpy.matmul(jax.numpy.transpose(zt), zx)
  p = 2.0 * zttzx
  p -= jax.numpy.matmul( jax.numpy.transpose(zt), jax.numpy.matmul(zt, zttxt))

  return p

z_theta = online_out['projection_view1']
z_xi = target_out['projection_view2']

z_theta /= jax.numpy.linalg.norm(z_theta, ord=2)
z_xi   /= jax.numpy.linalg.norm(z_xi, ord=2)

pred = ne_predictor(z_theta, z_xi)
pred /= jax.numpy.linalg.norm(pred, ord=2)
```

Listing 1. Pseudo-code for the NE predictor.

B.2.2. Visser predictor pseudo-code

```python
def visser(m, n, eta):

  p = 1/(2.0 * eta) * jax.numpy.eye(m.shape[0])
  for _ in range(n):
    p += eta * (m - jax.numpy.matmul(p, p))

  return p

z_theta = online_out['projection_view1']
sigma = jax.numpy.matmul(jax.numpy.transpose(z_theta), z_theta)

pred = visser(sigma, n_iters, epsilon)
```

Listing 2. Pseudo-code for Visser predictor.

B.2.3. Newton-Schulz predictor pseudo-code

```python
def newtonschulz(m, n):

  a = m
  fronorm_a = jax.numpy.linalg.norm(a, 'fro')
  a /= fronorm_a

  b = jax.numpy.eye(m.shape[0])
  c = 3.0 * b
  for _ in range(n):
    ba = jax.numpy.matmul(b, a)
    a = 0.5 * jax.numpy.matmul(a, c - ba)
    b = 0.5 * jax.numpy.matmul(c - ba, b)

  a *= jax.numpy.sqrt(fronorm_a)
  b /= jax.numpy.sqrt(fronorm_a)

  return a, b

z_theta = online_out['projection_view1']
sigma = jax.numpy.matmul(jax.numpy.transpose(z_theta), z_theta)

pred = newtonschulz(sigma, n_iter)[0]
```

Listing 3. Pseudo-code for Newton-Schulz predictor.

B.2.4. STIEFEL PREDICTOR PSEUDO-CODE

```
def stiefel(m, n):
  x = m.transpose()
  xxt = jax.numpy.matmul(x, m)
  xxt /= m.shape[0]

  x_inv_sqrt = newtonschulz(xxt, n)[1]
  x_inv_sqrt /= jax.numpy.linalg.norm(x_inv_sqrt, ord='fro')

  p = jax.numpy.matmul(x_inv_sqrt, x)

  return p

z_theta = online_out['projection_view1']
sigma = jax.numpy.matmul(jax.numpy.transpose(z_theta), z_theta)

pred = stiefel(sigma, n_iter)
```

Listing 4. Pseudo-code for Stiefel predictor.

B.2.5. LINEAR REGRESSION PREDICTOR PSEUDO-CODE

```
def lrp(zt, zx, safe_eps=1e-12):
    normfactor = 0.5 * jax.numpy.linalg.norm(zt, ord='fro')
    normfactor += 0.5 * jax.numpy.linalg.norm(zx, ord='fro') + safe_eps

    p = jax.numpy.matmul(jax.numpy.linalg.pinv(zt/normfactor), zx/normfactor)
    return p

z_theta = online_out['projection_view1']
z_xi = target_out['projection_view2']
pred = lrp(z_theta, z_xi)
```

Listing 5. Pseudo-code for LRP predictor.

C. Additional quantitative results

C.1. Top-5 evaluation of predictors

Predictor	0.0	0.15	0.3	0.45	0.6	0.75	0.9
DirectCopy†	43.6	87.4	88.1	<u>88.3</u>	88.3	88.2	88.2
NE	84.4	<u>88.9</u>	88.7	88.6	88.4	88.2	88.0
Visser	<u>88.8</u>	88.7	88.8	88.8	88.8	88.8	88.8
NS , $n = 9$	87.6	87.7	87.8	88.1	88.4	88.4	<u>88.5</u>
NS^2 , $n = 7$	87.9	88.8	88.9	89.0	88.9	89.0	<u>89.0</u>
Stiefel , $n = 9$	87.8	88.4	89.0	89.0	89.0	**<u>89.1</u>**	89.0

Table 5. Top-5 accuracy on ImageNet-1k (linear protocol) of our predictors, at 100 epochs, as ridging α varies. *DirectCopy* (Wang et al., 2021) is our own replication. Average over 3 random seeds. Standard trainable linear predictor `BYOL` achieves 88.5%.

Predictor	0.0	0.15	0.3	0.45	0.6	0.75	0.9
DirectCopy†	74.4	90.5	90.6	90.6	90.6	<u>90.7</u>	90.6
NE	89.2	<u>90.8</u>	90.6	90.6	90.4	90.3	90.2
Visser	<u>90.6</u>	90.6	90.6	90.6	90.5	90.6	90.5
NS , $n = 9$	90.2	90.8	<u>90.9</u>	90.9	90.8	90.9	90.9
NS^2 , $n = 7$	90.3	90.9	<u>91.0</u>	91.0	90.9	90.9	90.9
Stiefel , $n = 9$	90.5	**<u>91.1</u>**	90.9	90.9	91.0	90.9	90.9

Table 6. Top-5 accuracy on ImageNet-1k (linear protocol) of our predictors, at 300 epochs, as ridging α varies. *DirectCopy* (Wang et al., 2021) is our own replication. Average over 3 random seeds. Standard trainable linear predictor `BYOL` achieves 91.0%.

C.2. Ablation on the number of iterations in approximate methods

Ablation on the number of iterations. It is instructive to vary the number of iterations for our approximate square-root methods. Our baseline Newton-Schulz predictor uses 9 iterations. In Table 8, we compute the performance as a function of variable number of iterations n. We see it remains robust both at 100 and 300 epochs as long as $n \geq 5$, and that variations in performance remain minimal, in line with intuition for a second-order optimization method. Good results are obtained with as few as 3 iterations. In Table 7, we perform the same analysis with the number of necessary Visser iterations. We find that at 100 epochs, which is the case we're most interested in, at least 50 iterations are required for good performance, and more iterations doesn't hurt performance. No such clear pattern regarding performance emerges at 300 epochs.

Visser performance	12	25	50	100
100 epochs	67.2	68.2	**68.4**	68.4
300 epochs	**71.8**	70.6	71.6	69.2

Table 7. Top-1 accuracy on ImageNet-1k (linear protocol) of the Visser predictor, as a function of its number of iterations, for best ridging parameter $\alpha = 0.0$. Average over 3 random seeds.

NS performance	3	5	7	9
100 epochs	68.0	**68.1**	**68.1**	68.0
300 epochs	71.8	**72.1**	72.0	71.9

Table 8. Top-1 accuracy on ImageNet-1k (linear protocol) of the NS predictor, as a function of its number of iterations, for best ridging parameter $\alpha = 0.9$. Average over 3 random seeds.

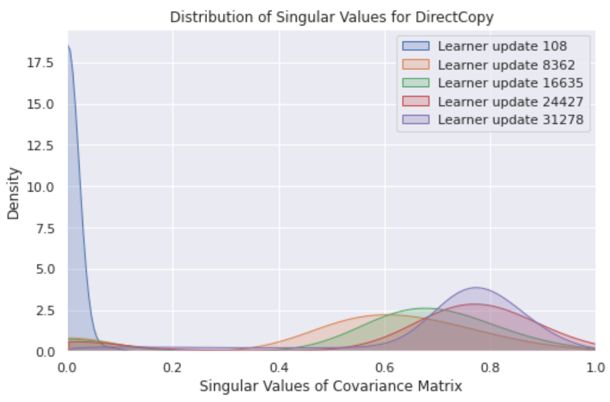
When all three spectrums of epochs, number of iterations as we've just seen Table 9, and ridging parameter α are considered, the Stiefel predictor performance is best across the board.

D. Additional graphs

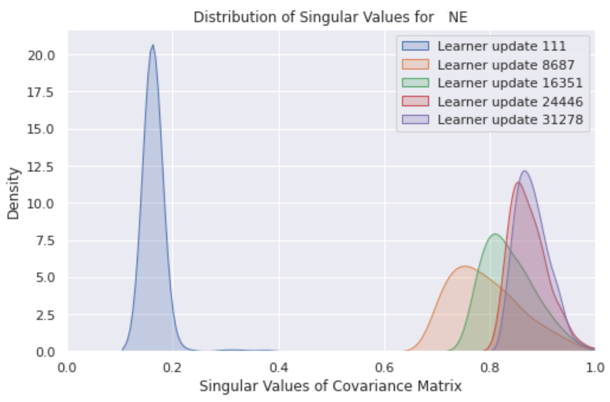
Figures 4 and 5 extend Figure 3. For four different predictors (*DirectCopy*, and our NE, NS and Stiefel predictors), they depict the evolution throughout training of the empirical distribution of singular values of the online latents' covariance, as

Stiefel performance	3	5	7	9
100 epochs	66.8	68.0	68.6	**68.8**
300 epochs	71.8	71.9	71.9	**72.1**

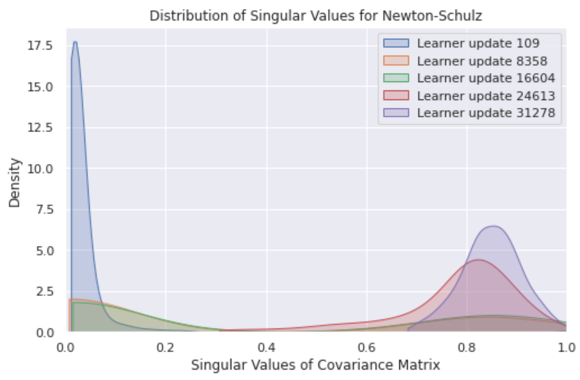
Table 9. Top-1 accuracy on ImageNet-1k (linear protocol) of the Stiefel predictor, as a function of its number of iterations, for best ridging parameter $\alpha = 0.3$. Average over 3 random seeds.



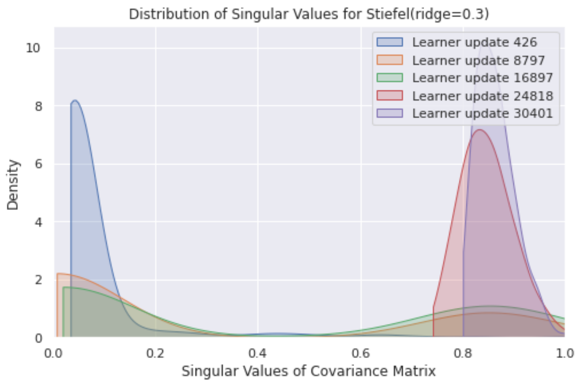
(a) *DirectCopy* predictor



(b) NE predictor



(c) Newton-Schulz predictor

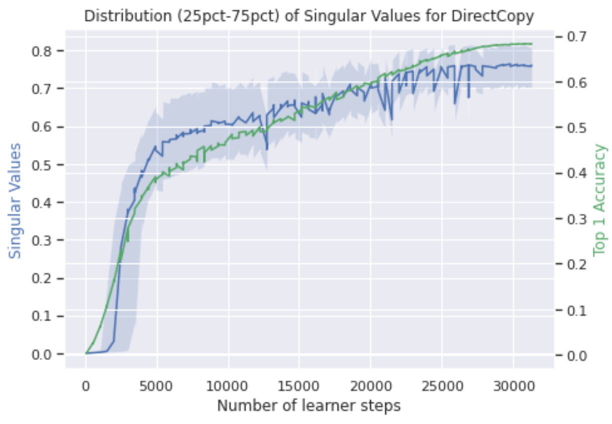


(d) Stiefel predictor

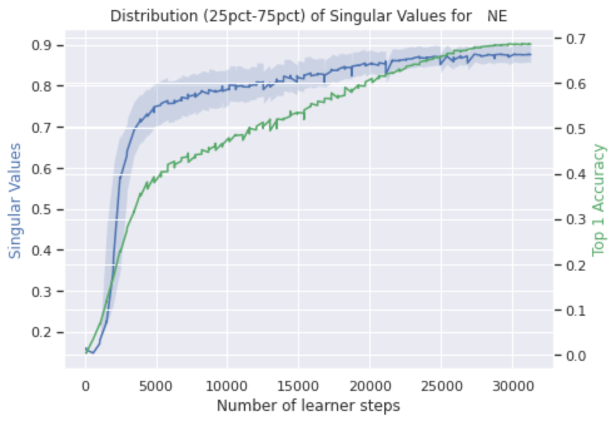
Figure 4. Empirical distribution of the singular values of the covariance of features, for various predictors. Plot throughout 100 epochs of training, on ImageNet-$1k$, averaged over 3 random seeds, extending Figure 3.

those are dynamically renormalized between 0 and 1 throughout training. These distributions are averaged across 3 ImageNet trials at 100 training epochs. Starting from a unimodal distribution with a very concentrated peak near 0, these distributions become bimodal with local maxima near 0 and 1 throughout training. This is consistent with the expected behaviour of a matrix getting closer and closer to idempotence ($P^2 = P$).

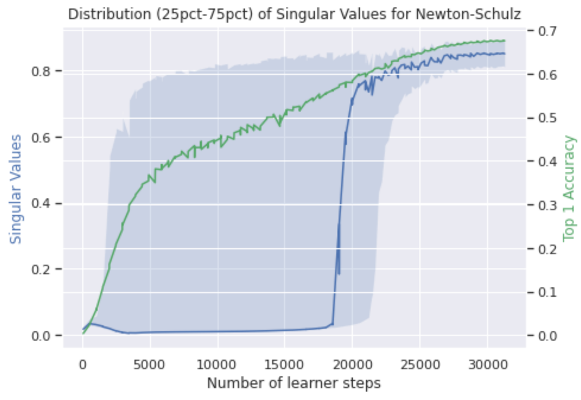
We observe first Figure 4 that a narrow peak of eigenvalues close to 1, indicative of a predictor almost perfectly an orthogonal projection, seems to loosely correlate with the final performance reached by that predictor. Second, we see Figure 5 that there are two distinct temporal profiles for the predictors we depict, and the distributions of singular values they induce. The top 2 (NE and *DirectCopy*) latch onto a few key singular directions early in training, before making steady progress afterwards. By contrast, the two bottom predictors depicted in that figure, NS and Stiefel, both based on the Newton-Schulz iteration, display a clear pattern of wider selection of singular directions (meaning that the span of these predictors is slower to grow, until midway in training where the behaviour changes abruptly). This behaviour is also associated with slightly better terminal accuracy under linear evaluation.



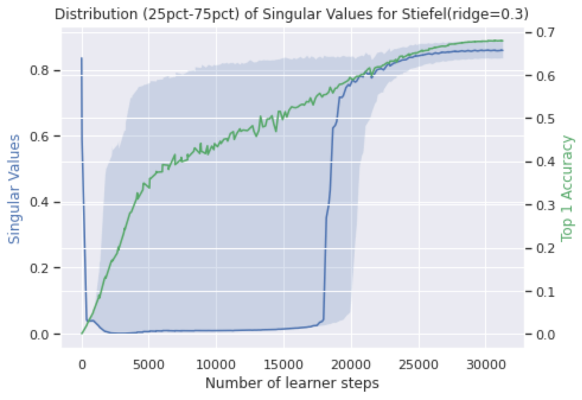
(a) *DirectCopy* predictor



(b) NE predictor



(c) Newton-Schulz predictor



(d) Stiefel predictor

Figure 5. Blue (left legend): Median, 25th and 75th percentile of empirical distribution of the singular values of the empirical covariance of latents, for four different predictors. *Green* (right legend): Evolution of their Top-1 % linear accuracy. Plot throughout 100 epochs of training, on ImageNet-$1k$, averaged over 3 random seeds, extending Figure 3.

E. Compute costs

We run our experiments on tranches of 128 Cloud TPUv3 cores. With this number of devices, each single run usually lasts a few hours, depending on its number of epochs. For instance, a single 300 epochs run can last around 6 hours, excluding hardware pre-emptions. A typical result with 3 random seeds over 300 epochs each requires less than a day of compute in this setting. As we use the LARS optimizer (You et al., 2017), this configuration is robust to lowering the total number of devices without much retuning of the base learning rate, although this comes at a proportional cost in terms of elapsed compute time. In our experiments, we did not see extremely material wall clock overhead from using fixed-form predictors instead of trainable ones, *except* for the case of the Visser iteration, which can require on the order of magnitude of 100 steps or more (as it is a first-order method) and is thus able to introduce significant slowdowns. The reader interested in both speed and performance will therefore prefer a Newton-Schulz or Stiefel iteration for predictor.