A PSEUDOCODE FOR FROSSL

```
for x in loader:
    # augment the image
    x_a, x_b = augment(x)

    # pass through network f to get embeddings
    z_a = f(x_a)
    z_b = f(x_b)
    N, d = Z_a.shape

    # center embeddings
    Z_a = Z_a - Z_a.mean(0)
    Z_b = Z_b - Z_b.mean(0)

    # normalize dimensions to sqrt(D) std.
    Z_a = (D**0.5) * (Z_a / Z_a.norm())
    Z_b = (D**0.5) * (Z_b / Z_b.norm())

    # calculate invariance (MSE) term
    invariance_loss = MSELoss(Z_a, Z_b)

    # calculate variance (Frobenius norm) term
    frobenius_a = torch.log(torch.norm(Z_a.T @ Z_a, ord='fro'))
    frobenius_b = torch.log(torch.norm(Z_b.T @ Z_b, ord='fro'))
    variance_loss = frobenius_a + frobenius_b

    # FroSSL loss
    loss = invariance_loss + variance_loss
    loss.backward()
    optimizer.step()
```

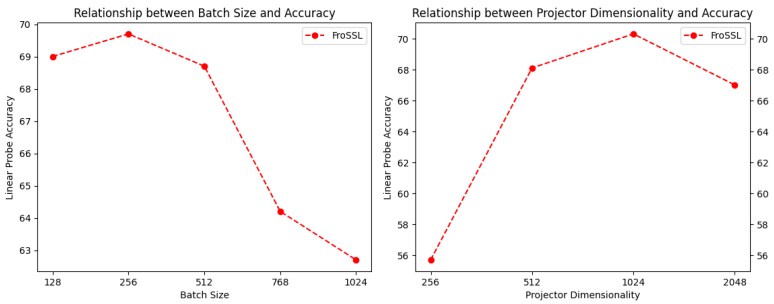
B EXPERIMENTAL DETAILS

B.1 STEPWISE CONVERGENCE EXPERIMENTAL DETAILS

We trained ResNet18 on STL10 using SGD with $lr = 0.1$ and a batch size of 256. Training occurred for only 5 epochs because we were interested in stepwise behaviors early during training.

- **Barlow Twins** We used $\lambda = 0.05$ as recommended by Zbontar et al. (2021) and $d = 1024$.
- **VICReg** We used $\lambda = 25$, $\mu = 25$, $\nu = 1$ as recommended by Bardes et al. (2022).
- **SimCLR** We used temperature $\tau = 0.2$ and $d = 256$.
- **FroSSL** We used $d = 1024$.

C ABLATION GRAPHS



D TRAINING DYNAMICS OF FROSSL

D.1 GRADIENT WITH LINEAR NETWORK

Given the empirical improvements to optimization speed observed in the previous section, it is of interest to study the theoretical training dynamics of FroSSL in comparison to other SSL criterion. Inspired by the approach taken in Simon et al. (2023), we examine how FroSSL behaves in the linear network regime.

First, we use a simplified variant of FroSSL given by:

$$\mathcal{L} = ||Z_1^T Z_1 - I_d||_F^2 + ||Z_2^T Z_2 - I_d||_F^2 + ||Z_1 - Z_2||_F^2 \tag{10}$$

As compared to Equation (6), the main changes are: the mean-squared error is replaced with an equivalent formulation in terms of Frobenius norms, the logs are removed because they lead to nonlinear ODEs later that are nontrivial to solve, and batchnorm is replaced with a distance to I_d. The variance terms of Equation (10) are similar to Barlow Twins, although they are defined on covariance matrices rather than cross-correlation matrices. Additionally, Equation (10) bears resemblance to a VICReg variant used to study optimal SSL representations through the lens of graph Laplacians (Balestriero & LeCun, 2022). However, the VICReg variant considers a covariance matrix of a batch containing both Z_1 and Z_2, rather than treating them individually as we do.

Second, we assume our networks are a linear mapping $W_1, W_2 \in \mathbb{R}^{dxm}$. In particular, we present a general analysis where each branch is not restricted to share weights. Because implicitly $Z_1 = f(X_1)$, we can simplify Equation (10) as

$$\mathcal{L} = ||W_1 \Gamma_1 W_1^T - I_d||_F^2 + ||W_2 \Gamma_2 W_2^T - I_d||_F^2 + ||X_1 W_1^T - X_2 W_2^T||_F^2 \tag{11}$$

where we have defined the data covariance $\Gamma_1 = X_1^T X_1 \in \mathbb{R}^{mxm}$. Because we wish to understand the training dynamics of W_1 and W_2, we define their gradients as:

$$\frac{dW_1}{dt} = \nabla_{W_1} \mathcal{L} = -2(W_1 \Gamma_1 + 2W_2 X_2^T X_1 + 2W_1 \Gamma_1 W_1^T W_1 \Gamma_1) \tag{12}$$

$$\frac{dW_2}{dt} = \nabla_{W_2} \mathcal{L} = -2(W_2 \Gamma_2 + 2W_1 X_1^T X_2 + 2W_2 \Gamma_2 W_2^T W_2 \Gamma_2) \tag{13}$$

We show how these gradients were derived in Appendix E.3. Because of the difficulty of solving Equations (12) and (13) for arbitrary choices of W, in the subsequent section we choose particular W_1 and W_2 that aids analysis.

D.2 ALIGNING W WITH THE DATA COVARIANCE

For brevity, we describe the initialization for W_1, though W_2 follows identically. Because we train with gradient descent, we parameterize W_1 in terms of time t as $W_1(t)$. As described in Simon et al. (2023), one powerful choice of $W_1(0)$ when initializing $W_1 = W_1(0)$ is setting the right singular vectors of W_1 to be the top eigenvectors of the data covariance Γ_1. This type of initialization is called "aligned initialization". One critical assumption henceforth is that for finite batch sizes the data covariances per view, Γ_1 and Γ_2, share eigenvectors but perhaps have differing eigenvalues. This is reasonable because X_1 and X_2 are drawn from the same distribution and are augmented with the same random transforms. Next, we define the eigendecompositions of Γ_1, Γ_2, and $W_1(0)$ as:

$$\Gamma_1 = V D_1 V^T \tag{14}$$

$$\Gamma_2 = V D_2 V^T \tag{15}$$

$$W_1(0) = U_1 S_1(0) \hat{V}^{\leq d} \tag{16}$$

where $U_1 \in \mathbb{R}^{dxd}$ are arbitrary orthonormal matrices, $\hat{V}^{\leq d} \in \mathbb{R}^{dxm}$ are the top d eigenvectors of Γ_1 and Γ_2, and $S_1(0) \in \mathbb{R}^{dxm}$ are diagonal matrices of singular values such that $S_1(0) = \text{diag}(s_{1,1}(0), \cdots, s_{1,d}(0))$ with $s_{1,j}(0) > 0$. The matrices U_1 and $S_1(0)$ may be thought of as our initial random parameters, with $\hat{V}^{\leq d}$ placing constraints on the final orientation of $W(0)$. The matrices $U_2(0)$ and $S_2(0)$ are defined similarly for $W_2(0)$. The training dynamics of $W_1(0)$ from Equation (12) are given by the following proposition:

Proposition D.1 (Aligned Initialization)**.** If W_1 is initialized by (16), with W_2 being initialized similarly, then the singular vectors of W_1 do not change over time. The W_1 parameterized by t given by

$$W_1(t) = U_1 S(t) \hat{V}^{\leq d} \tag{17}$$

and the singular values evolve according to

$$s_{1,j}(t) = \sqrt{\frac{2\sqrt{\lambda_j \gamma_j} - \lambda_j}{(2\lambda_j^2 - \frac{\lambda_j - 2\sqrt{\lambda_j \gamma_j}}{s_{1,j}^2(0)}) \exp\left[8t\sqrt{\lambda_j \gamma_j} - 4\lambda_j t\right] - 2\lambda_j^2}} \tag{18}$$

Proof. A full proof is given in Appendix E.4. A quick outline is that plugging (17) into (12) gives an ordinary differential equation (ODE). Solving this ODE as an initial value problem gives (18). □

Proposition D.2 (Small Initialization)**.** If W_1, W_2 have initial weights drawn from $\mathcal{N}(0, \sigma^2)$, for sufficiently small σ, then they have the same training dynamics of aligned initialization as described in Proposition D.1. This is proven in Simon et al. (2023).

E Proofs

E.1 Proof of Proposition 3.1

We start with rewriting the $\arg\min$ of Equation (6) as such:

$$\begin{aligned}
\arg\min_{Z_1, Z_2} \mathcal{L}_{\text{FroSSL}} &= \arg\min_{Z_1, Z_2} \left[\log(||Z_1^T Z_1||_F^2) + \log(||Z_2^T Z_2||_F^2) + \mathcal{L}_{\text{MSE}}(Z_1, Z_2)\right] \\
&= \arg\min_{Z_1, Z_2} \left[||Z_1^T Z_1||_F^2 + ||Z_2^T Z_2||_F^2 + \mathcal{L}_{\text{MSE}}(Z_1, Z_2)\right]
\end{aligned}$$

Without loss of generality, assume that each dimension has unit variance. Then both covariance matrices have 1 in each diagonal element.

$$\begin{aligned}
&= \arg\min_{Z_1, Z_2} \left[||Z_1^T Z_1 - \text{diag}(Z_1^T Z_1)||_F^2 + ||Z_2^T Z_2 - \text{diag}(Z_2^T Z_2)||_F^2 + 2D + \mathcal{L}_{\text{MSE}}(Z_1, Z_2)\right] \\
&= \arg\min_{Z_1, Z_2} \left[\mathcal{L}_{nc}(Z_1) + \mathcal{L}_{nc}(Z_2) + 2D + \mathcal{L}_{\text{MSE}}(Z_1, Z_2)\right]
\end{aligned}$$

Thus we have that the embeddings that minimize FroSSL also minimize the non-contrastive losses $\mathcal{L}_{nc}$ for both views.

E.2 Proof of Proposition 3.2

With Property 2, we rewrite Equation (6) to use Gram matrices rather than covariance matrices:

$$\begin{aligned}
\mathcal{L}_{\text{FroSSL}} &= \log(||Z_1^T Z_1||_F^2) + \log(||Z_2^T Z_2||_F^2) + \mathcal{L}_{\text{MSE}}(Z_1, Z_2) \\
&= \log(||Z_1 Z_1^T||_F^2) + \log(||Z_2 Z_2^T||_F^2) + \mathcal{L}_{\text{MSE}}(Z_1, Z_2)
\end{aligned}$$

Assuming that each embedding is normalized to have unit norm, then both Gram matrices have 1 in each diagonal element. Then the rest of the proof then follows similarly to Proposition 3.1.

E.3 Derivation of FroSSL Variant Gradient

We start with Equation (11) and derive each term individually.

Term 3: The third term of (11), which corresponds to the MSE invariance term of (6), can be rewritten as:

$$||X_1 W_1^T - X_2 W_2^T||_F^2 = \text{trace}((X_1 W_1^T - X_2 W_2^T)^T (X_1 W_1^T - X_2 W_2^T)) \tag{19}$$

$$= \text{trace}(W_1 X_1^T X_1 W_1^T + W_2 X_2^T X_2 W_2^T - 2 W_2 X_2^T X_1 W_1^T) \tag{20}$$

$$= \text{trace}(W_1 X_1^T X_1 W_1^T) + \text{trace}(W_2 X_2^T X_2 W_2^T) - 2\text{trace}(W_2 X_2^T X_1 W_1^T) \tag{21}$$

Using Equations 102 and 111 in the Matrix Cookbook (Petersen et al., 2008), we get the gradient as

$$\begin{aligned}\nabla_{W_1}||X_1W_1^T - X_2W_2^T||_F^2 &= 2W_1X_1^TX_1 - 2W_2X_2^TX_1 \\ &= 2W_1\Gamma_1 - 2W_2X_2^TX_1\end{aligned} \tag{22}$$

$$\begin{aligned}\nabla_{W_2}||X_1W_1^T - X_2W_2^T||_F^2 &= 2W_2X_2^TX_2 - 2W_1X_1^TX_2 \\ &= 2W_2\Gamma_2 - 2W_1X_1^TX_2\end{aligned} \tag{23}$$

Term 1: The first term of (11), which corresponds to the argument of the View 1 logarithm of (6), is derived using Equation 6 of Simon et al. (2023). In particular, we get

$$\nabla_{W_1}||W_1\Gamma_1W_1^T - I_d||_F^2 = -4(W_1\Gamma_1W_1^T - I_d)W_1\Gamma_1 \tag{24}$$

$$\nabla_{W_2}||W_1\Gamma_1W_1^T - I_d||_F^2 = 0 \tag{25}$$

Term 2: The second term of (11) follows similarly to the first term.

$$\nabla_{W_1}||W_2\Gamma_2W_2^T - I_d||_F^2 = 0 \tag{26}$$

$$\nabla_{W_2}||W_2\Gamma_2W_2^T - I_d||_F^2 = -4(W_2\Gamma_1W_2^T - I_d)W_2\Gamma_2 \tag{27}$$

Combining Everything We can now combine all of our gradients to find (12) and (13).

$$\begin{aligned}\nabla_{W_1}\mathcal{L} &= 2W_1\Gamma_1 - 2W_2X_2^TX_1 - 4(W_1\Gamma_1W_1^T - I_d)W_1\Gamma_1 \\ &= -2(W_1\Gamma_1 + 2W_2X_2^TX_1 + 2W_1\Gamma_1W_1^TW_1\Gamma_1)\end{aligned} \tag{28}$$

$$\begin{aligned}\nabla_{W_2}\mathcal{L} &= 2W_2\Gamma_2 - 2W_1X_1^TX_2 - 4(W_2\Gamma_2W_2^T - I_d)W_2\Gamma_2 \\ &= -2(W_2\Gamma_2 + 2W_1X_1^TX_2 + 2W_2\Gamma_2W_2^TW_2\Gamma_2)\end{aligned} \tag{29}$$

E.4 Derivation of Linear layer Gradient w.r.t Time

Plugging (17) into (12), we immediately get

$$\begin{aligned}\frac{dW_1}{dt} = 2US_1(t)D_1\hat{V}^{\odot\leq d} - 4US_1(t)(D_1^{\odot\frac{1}{2}} \odot D_2^{\odot\frac{1}{2}})\hat{V}^{\leq d} \\ - 4US_1^3(t)D_1^2\hat{V}^{\leq d}\end{aligned} \tag{30}$$

Next, one should recognize that all terms are left-multiplied by U and right-multiplied by $\hat{V}^{\leq d}$. This lets us combine everything into a more compact form.

$$\frac{dW}{dt} = 2U\left[S_1(t)\left(D_1 - 2(D_1^{\odot\frac{1}{2}} \odot D_2^{\odot\frac{1}{2}}) - 2S_1^2(t)D_1^2\right)\right]\hat{V}^{\leq d} \tag{31}$$

It can be seen in (31) that the singular vectors of W remain unchanged over time. The only changes are the singular values which are given in the brackets. The dynamics of the singular values over time constitute an ODE given by

$$s_{1,j}'(t) = 2s_{1,j}(t)\left(\lambda_j - 2\sqrt{\lambda_j\gamma_j} - 2s_{1,j}^2(t)\lambda_j^2\right) \tag{32}$$

where λ_j, γ_j are the j-th largest eigenvalues of D_1, D_2, respectively. One can find the general solution to this ODE using their favorite ODE solver. It gives a solution in the form:

$$s_{1,j}(t) = \sqrt{\frac{2\sqrt{\lambda_j\gamma_j} - \lambda_j}{\exp\left[2(2\sqrt{\lambda_j\gamma_j} - \lambda_j)(c_1 + 2t)\right] - 2\lambda_j^2}} \tag{33}$$

where c_1 is some constant. We can find c_1 by solving the initial value problem given by our initial S_1 matrix. Thus (33) can be rewritten as:

$$s_{1,j}(t) = \sqrt{\frac{2\sqrt{\lambda_j\gamma_j} - \lambda_j}{(2\lambda_j^2 - \frac{\lambda_j - 2\sqrt{\lambda_j\gamma_j}}{s_{1,j}^2(0)})\exp\left[8t\sqrt{\lambda_j\gamma_j} - 4\lambda_j t\right] - 2\lambda_j^2}} \tag{34}$$

REFERENCES

Randall Balestriero and Yann LeCun. Contrastive and non-contrastive self-supervised learning recover global and local spectral embedding methods. *Advances in Neural Information Processing Systems*, 35:26671–26685, 2022. 2

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations*, 2022. 1

Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008. 4

James B. Simon, Maksis Knutins, Ziyin Liu, Daniel Geisz, Abraham J. Fetterman, and Joshua Albrecht. On the stepwise nature of self-supervised learning. In *International Conference on Machine Learning*, 2023. 2, 3, 4

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pp. 12310–12320. PMLR, 2021. 1