

---

# Supplementary Materials: Modulate Your Spectrum in Self-Supervised Learning

---

Anonymous Author(s)

Affiliation

Address

email

## 1 A Proofs of Proposition

### 2 A.1 Proof of Proposition.1

3 **Proposition 1.** *Given  $x \in (0, 1)$ ,  $\forall T \in \mathbb{N}$  we have  $h_T(x) \in (0, 1)$  and  $h'_T(x) > 0$ .*

4 *Proof.* We know the iterative function  $f_T(x)$  satisfies

$$f_{k+1}(x) = \frac{3}{2}f_k(x) - \frac{1}{2}xf_k^3(x), k \geq 0; f_0(x) = 1 \quad (1)$$

5 We define  $h_T(x) = xf_T^2(x)$ . When  $x = 1$ , it is easy to verify  $\forall T \in \mathbb{N}$ ,  $h_T(1) = f_T(1) = 1$ . We  
6 first prove  $f_T(x) > 0$  and  $h'_T(x) > 0$  by mathematical induction.

7 (1) When  $T = 0$ , we have  $f_0(x) = 1 > 0$ , and  $h_0(x) = x$ ,  $h'_0(x) = 1 > 0$ .

8 (2) Assuming it holds when  $T = k$ , we have  $f_k(x) > 0$  and  $h'_k(x) > 0$ . Based on  $h'_k(x) =$   
9  $f_k(x)[f_k(x) + 2xf'_k(x)]$ , we have:

$$f_k(x) + 2xf'_k(x) > 0 \quad (2)$$

10 Since  $h_k(1) = 1$ ,  $h'_k(x) > 0$  and  $h_k(x)$  is continuous, we have  $\forall x \in (0, 1)$ ,  $h_k(x) < 1$ . We thus can  
11 obtain:

$$\begin{aligned} f_{k+1}(x) &= \frac{1}{2}f_k(x)[3 - xf_k^2(x)] \\ &= \frac{1}{2}f_k(x)[3 - h_k(x)] \\ &> 0 \end{aligned} \quad (3)$$

Furthermore,  $h'_{k+1}(x) = f_{k+1}(x)[f_{k+1}(x) + 2xf'_{k+1}(x)]$ , where

$$\begin{aligned} &f_{k+1}(x) + 2xf'_{k+1}(x) \\ &= \frac{3}{2}[f_k(x) + 2xf'_k(x)] - \frac{3}{2}xf_k^3(x) - 3x^2f_k^2(x)f'_k(x) \\ &= \frac{3}{2}[f_k(x) + 2xf'_k(x)] - \frac{3}{2}xf_k^2(x)[f_k(x) + 2xf'_k(x)] \\ &= \frac{3}{2}[1 - xf_k^2(x)][f_k(x) + 2xf'_k(x)] \\ &= \frac{3}{2}[1 - h_k(x)][f_k(x) + 2xf'_k(x)] \end{aligned}$$

12 So we have  $h'_{k+1}(x) = \frac{3}{2}f_{k+1}(x)[1 - h_k(x)][f_k(x) + 2xf'_k(x)] > 0$ . Combining the result in Eqn. 3,  
13 we thus have it holds when  $T = k + 1$ .

14 As a result, we have  $\forall T \in \mathbb{N}, f_T(x) > 0$  and  $h'_T(x) > 0$ , when  $x \in (0, 1)$ .  
 15 Since  $h_T(1) = 1$  and  $h_T(x)$  is continuous, we have  $h_T(x) < 1$ . Besides, we have  $h_T(x) =$   
 16  $xf_T^2(x) > 0$ , then  $h_T(x) \in (0, 1)$ .  $\square$

## 17 A.2 Proof of Proposition.2

18 **Proposition 2.** Given  $x \in (0, 1)$ ,  $\forall T \in \mathbb{N}$ , we have  $h_{T+1}(x) > h_T(x)$ .

19 *Proof.* According to proof of Proposition.1, we have that when  $x \in (0, 1)$  and  $\forall T \in \mathbb{N}, f_T(x) > 0$   
 20 and  $h_T(x) = xf_T^2(x) \in (0, 1)$ .

Therefore, we have  $h_{T+1}(x) > h_T(x) \iff f_{T+1}(x) > f_T(x)$ . It is obvious that

$$\begin{aligned} f_{k+1}(x) - f_k(x) &= \frac{3}{2}f_k(x) - \frac{1}{2}xf_k^3(x) - f_k(x) \\ &= \frac{1}{2}f_k(x) - \frac{1}{2}xf_k^3(x) \\ &= \frac{1}{2}f_k(x)[1 - xf_k^2(x)] \\ &= \frac{1}{2}f_k(x)[1 - h_k(x)] \\ &> 0 \end{aligned}$$

21 So given  $x \in (0, 1)$ ,  $\forall T \in \mathbb{N}$ , we have  $h_{T+1}(x) > h_T(x)$ .  $\square$

## 22 B Proofs of Theorem

### 23 B.1 Proof of Theorem 1.

24 **Theorem 1.** Define one-variable iterative function  $f_T(x)$ , satisfying

$$25 \quad f_{k+1}(x) = \frac{3}{2}f_k(x) - \frac{1}{2}xf_k^3(x), k \geq 0; f_0(x) = 1.$$

26 The mapping function of IterNorm is

$$27 \quad g(\lambda) = f_T\left(\frac{\lambda}{tr(\Sigma)}\right) / \sqrt{tr(\Sigma)},$$

28 so that  $\forall \lambda_i \in \lambda(\mathbf{Z})$ , IterNorm maps it to  $\hat{\lambda}_i = \frac{\lambda_i}{tr(\Sigma)} f_T^2\left(\frac{\lambda_i}{tr(\Sigma)}\right)$ .

29 *Proof.* Given  $\Sigma = \mathbf{U}\Lambda\mathbf{U}^T$ ,  $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_d)$ ,  $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_d]$ . Following the calculation  
 30 steps of IterNorm, we have

$$\Sigma_N = \Sigma / tr(\Sigma) = \sum_{i=1}^d \frac{\lambda_i}{tr(\Sigma)} \mathbf{u}_i \mathbf{u}_i^T \quad (4)$$

31 Define

$$\Phi'_T = \sum_{i=1}^d \frac{1}{\sqrt{tr(\Sigma)}} f_T\left(\frac{\lambda_i}{tr(\Sigma)}\right) \mathbf{u}_i \mathbf{u}_i^T \quad (5)$$

32 Based on  $\Phi_T = \sum_{i=1}^d g(\lambda_i) \mathbf{u}_i \mathbf{u}_i^T$ , if we can prove  $\Phi'_T = \Phi_T$ , we will have

$$33 \quad g(\lambda) = \frac{1}{\sqrt{tr(\Sigma)}} f_T\left(\frac{\lambda}{tr(\Sigma)}\right)$$

34 Define  $\mathbf{P}'_T = \sqrt{tr(\Sigma)} \Phi'_T$ , then we have  $\Phi'_T = \Phi_T \iff \mathbf{P}'_T = \mathbf{P}_T$ . We can prove  $\mathbf{P}'_T = \mathbf{P}_T$  by  
 35 mathematical induction.

36 (1) When  $T = 0$ ,

$$f_0(\frac{\lambda_i}{tr(\Sigma)}) = 1, \mathbf{P}'_0 = \mathbf{P}_0 = \mathbf{I}$$

(2) When  $T \geq 1$ , assume that  $\mathbf{P}'_{T-1} = \mathbf{P}_{T-1}$ , thus

$$\begin{aligned} \mathbf{P}_T &= \frac{3}{2}\mathbf{P}_{T-1} - \frac{1}{2}\mathbf{P}_{T-1}^3 \Sigma_N \\ &= \frac{3}{2}\mathbf{P}'_{T-1} - \frac{1}{2}(\mathbf{P}'_{T-1})^3 \Sigma_N \end{aligned}$$

According to the definition of  $\mathbf{P}'_T$ ,

$$\mathbf{P}'_{T-1} = \sum_{i=1}^d f_{T-1}(\frac{\lambda_i}{tr(\Sigma)}) \mathbf{u}_i \mathbf{u}_i^T$$

Because  $\forall i, \mathbf{u}_i^T \mathbf{u}_i = 1$  and  $\forall i \neq j, \mathbf{u}_i^T \mathbf{u}_j = 0$ ,

$$\begin{aligned} \mathbf{P}_{T-1}^3 \Sigma_N &= (\mathbf{P}'_{T-1})^3 \Sigma_N \\ &= \left( \sum_{i=1}^d f_{T-1}(\frac{\lambda_i}{tr(\Sigma)}) \mathbf{u}_i \mathbf{u}_i^T \right)^3 \left( \sum_{i=1}^d \frac{\lambda_i}{tr(\Sigma)} \mathbf{u}_i \mathbf{u}_i^T \right) \\ &= \sum_{i=1}^d f_{T-1}^3(\frac{\lambda_i}{tr(\Sigma)}) \frac{\lambda_i}{tr(\Sigma)} \mathbf{u}_i \mathbf{u}_i^T \end{aligned}$$

Therefore, we have

$$\begin{aligned} \mathbf{P}_T &= \frac{3}{2}\mathbf{P}'_{T-1} - \frac{1}{2}(\mathbf{P}'_{T-1})^3 \Sigma_N \\ &= \frac{3}{2} \sum_{i=1}^d f_{T-1}(\frac{\lambda_i}{tr(\Sigma)}) \mathbf{u}_i \mathbf{u}_i^T - \frac{1}{2} \sum_{i=1}^d f_{T-1}^3(\frac{\lambda_i}{tr(\Sigma)}) \frac{\lambda_i}{tr(\Sigma)} \mathbf{u}_i \mathbf{u}_i^T \\ &= \sum_{i=1}^d \left\{ \frac{3}{2} f_{T-1}(\frac{\lambda_i}{tr(\Sigma)}) - \frac{1}{2} f_{T-1}^3(\frac{\lambda_i}{tr(\Sigma)}) \frac{\lambda_i}{tr(\Sigma)} \right\} \mathbf{u}_i \mathbf{u}_i^T \end{aligned}$$

Note

$$f_T(\frac{\lambda_i}{tr(\Sigma)}) = \frac{3}{2} f_{T-1}(\frac{\lambda_i}{tr(\Sigma)}) - \frac{1}{2} f_{T-1}^3(\frac{\lambda_i}{tr(\Sigma)}) \frac{\lambda_i}{tr(\Sigma)}$$

So that

$$\mathbf{P}_T = \sum_{i=1}^d f_T(\frac{\lambda_i}{tr(\Sigma)}) \mathbf{u}_i \mathbf{u}_i^T = \mathbf{P}'_T$$

We obtain that

$$\Phi_T = \Phi'_T = \sum_{i=1}^d \frac{1}{\sqrt{tr(\Sigma)}} f_T(\frac{\lambda_i}{tr(\Sigma)}) \mathbf{u}_i \mathbf{u}_i^T = \mathbf{U} \frac{1}{\sqrt{tr(\Sigma)}} f_T(\frac{\Lambda}{tr(\Sigma)}) \mathbf{U}^T$$

Thus, the mapping function of IterNorm is  $g(\lambda) = f_T(\frac{\lambda}{tr(\Sigma)}) / \sqrt{tr(\Sigma)}$ . The whitened output is

$\hat{\mathbf{Z}} = \Phi_T \mathbf{Z}_c = \mathbf{U} \frac{1}{\sqrt{tr(\Sigma)}} f_T(\frac{\Lambda}{tr(\Sigma)}) \mathbf{U}^T \mathbf{Z}_c$ . The covariance matrix of  $\hat{\mathbf{Z}}$  is

$$\Sigma_{\hat{\mathbf{Z}}} = \frac{1}{m} \hat{\mathbf{Z}} \hat{\mathbf{Z}}^T = \mathbf{U} \frac{\Lambda}{tr(\Sigma)} f_T^2(\frac{\Lambda}{tr(\Sigma)}) \mathbf{U}^T = \sum_{i=1}^d \frac{\lambda_i}{tr(\Sigma)} f_T^2(\frac{\lambda_i}{tr(\Sigma)}) \mathbf{u}_i \mathbf{u}_i^T$$

So that  $\forall \lambda_i \in \lambda(\mathbf{Z})$ , IterNorm maps it to  $\hat{\lambda}_i = \frac{\lambda_i}{tr(\Sigma)} f_T^2(\frac{\lambda_i}{tr(\Sigma)})$  which is a special instance of Spectral Transformation.  $\square$

## 49 B.2 Proof of Theorem 2.

50 **Theorem 2.** Let  $\mathbf{x} \in [0, 1]^d$ ,  $\forall T \in \mathbb{N}_+$ ,  $INTL(\mathbf{x})$  shown in Eqn. 7 is a strictly convex function.  
 51  $\mathbf{x}^* = [\frac{1}{d}, \dots, \frac{1}{d}]^T$  is the unique minimum point as well as the optimal solution to  $INTL(\mathbf{x})$ .

52 *Proof.* The INTL can be viewed as the following optimization problem:

$$\min_{\theta \in \Theta} INTL(\mathbf{Z}) = \sum_{j=1}^d (1 - (\Sigma_{\hat{\mathbf{Z}}})_{jj})^2 \quad (6)$$

53 where  $\mathbf{Z} = F_\theta(\cdot)$  and  $\hat{\mathbf{Z}} = IterNorm(\mathbf{Z})$ . Eqn. 6 can be viewed as a optimization problem over  $\theta$   
 54 to encourage the trace of  $\hat{\mathbf{Z}}$  to be  $d$ .

55 Let  $(x_1, \dots, x_d) = \varphi(\mathbf{Z})$ , where  $x_i = \lambda_i / tr(\Sigma)$  as defined in the submitted paper. If  $\mathbf{Z} \in \mathbb{R}^{d \times m}$ ,  
 56  $\varphi(\cdot)$  will be surjective from  $\mathbb{R}^{d \times m}$  to  $\mathbb{D}_{\mathbf{x}} = \{\mathbf{x} \in [0, 1]^d : x_1 + \dots + x_d = 1\}$ . When the range of  
 57  $F_\theta(\cdot)$  is wide enough, for example,  $F_\theta(\cdot)$  is surjective from  $\theta \in \Theta$  to  $\mathbf{Z} \in \mathbb{R}^{d \times m}$ . Here we can view  
 58  $F_\theta(\cdot)$  as a function over  $\theta$ , since the input is given and fixed. Then  $\varphi(F_\theta(\cdot))$  is surjective from  $\theta \in \Theta$   
 59 to  $\mathbf{x} \in \mathbb{D}_{\mathbf{x}}$ , meaning that if we find the optimal solution  $\mathbf{x}^*$ , we are able to get the corresponding  
 60  $\theta^* \in \Theta$ , subject to  $\mathbf{x}^* = \varphi(F_{\theta^*}(\cdot))$ . On the contrary, for any  $\theta \in \Theta$ , we can get  $\mathbf{x} = \varphi(F_\theta(\cdot)) \in \mathbb{D}_{\mathbf{x}}$ .

61 Therefore, the optimization expression for minimizing INTL can be written as follows which have  
 62 the same range and optimal value as Eqn. 6:

$$(P_{INTL}) \begin{cases} \min & INTL(\mathbf{x}) = \sum_{j=1}^d \left( \sum_{i=1}^d [1 - h_T(x_i)] u_{ji}^2 \right)^2 \\ s.t. & \sum_{i=1}^d x_i = 1 \\ & x_i \geq 0, i = 1, \dots, d \end{cases} \quad (7)$$

We denote the Lagrange function of  $P_{INTL}$  is that

$$L(\mathbf{x}; \alpha, \mu) = INTL(\mathbf{x}) + \sum_{i=1}^d \alpha_i (-x_i) + \mu \left( \sum_{i=1}^d x_i - 1 \right)$$

### 63 B.2.1 Convexity and Concavity of $h_T(x)$

64 Before calculating extreme points of  $P_{INTL}$ , we first consider the convexity and concavity of  $h_T(x)$   
 65 which is critical to proof.

66 When  $T = 0$ , we have  $h_0(x) = x$ , so  $h_0''(x) = 0$ .

67 (1) When  $T = 1$ , we have  $h_1(x) = f_1^2(x) = \frac{9}{4}x - \frac{3}{2}x^2 + \frac{1}{4}x^3$ , so  $h_1''(x) = \frac{3}{2}(x - 2) < 0$ .

68 (2) Assume that when  $T = k$ ,  $h_k''(x) < 0$  holds. We can easily get following propositions by  
 69 derivation:

$$f'_{k+1}(x) = \frac{3}{2}f'_k(x) - \frac{1}{2}f_k^3(x) - \frac{3}{2}xf_k^2(x)f'_k(x) \quad (8)$$

$$f''_{k+1}(x) = \frac{3}{2}f''_k(x) - 3f_k^2(x)f'_k(x) - \frac{3}{2}xf_k^2(x)f''_k(x) - 3xf_k(x)[f'_k(x)]^2 \quad (9)$$

$$h''_{k+1}(x) = 4f_{k+1}(x)f'_{k+1}(x) + 2x[f'_{k+1}(x)]^2 + 2xf_{k+1}(x)f''_{k+1}(x) \quad (10)$$

70 For convenience in our calculation, let  $a = f_k(x)$ ,  $b = f'_k(x)$ ,  $c = f''_k(x)$ , and  $h = h_k(x) = xa^2$ .

We split Eqn. 10 into three parts and take Eqn. 8 and 9 into calculation:

$$\begin{aligned}
4f_{k+1}(x)f'_{k+1}(x) &= 4\left(\frac{3}{2}a - \frac{1}{2}ah\right)\left(\frac{3}{2}b - \frac{1}{2}a^3 - \frac{3}{2}bh\right) \\
&= a(3-h)(3b - a^3 - 3bh) \\
2x[f'_{k+1}(x)]^2 &= 2x\left(\frac{3}{2}b - \frac{1}{2}a^3 - \frac{3}{2}bh\right)^2 \\
&= \frac{1}{2}x(3b - a^3 - 3bh)^2 \\
2xf_{k+1}(x)f''_{k+1}(x) &= 2\left(\frac{3}{2}a - \frac{1}{2}ah\right)\left[\frac{3}{2}c(1-h) - 3a^2b - 3xab^2\right] \\
&= \frac{1}{2}ax(3-h)[3c(1-h) - 6a^2b - 6xab^2]
\end{aligned}$$

Considering to construct the form of  $h''_k(x) = 2(2ab + xac + xb^2)$ , we first calculate that

$$\begin{aligned}
&4f_{k+1}(x)f'_{k+1}(x) + 2xf_{k+1}(x)f''_{k+1}(x) \\
&= \frac{1}{2}(3-h)[6ab - 2a^4 - 6abh + 3xac(1-h) - 6abh - 6xb^2h] \\
&= \frac{1}{2}(3-h)[3xac(1-h) + 6ab(1-h) + 3xb^2(1-h) \\
&\quad - 3xb^2(1-h) - 2a^4 - 6abh - 6xb^2h] \\
&= \frac{3}{4}(3-h)(1-h)h''_k(x) - \frac{1}{2}(3-h)(3xb^2h + 3xb^2 + 2a^4 + 6abh)
\end{aligned}$$

Then we calculate the left part

$$\begin{aligned}
2x[f'_{k+1}(x)]^2 &= \frac{1}{2}x(3b - a^3 - 3bh)^2 \\
&= \frac{1}{2}(9xb^2 + xa^6 + 9xb^2h^2 - 6xa^3b - 18xb^2h + 6xa^3bh) \\
&= \frac{1}{2}(9xb^2 + a^4h + 9xb^2h^2 - 6abh - 18xb^2h + 6abh^2)
\end{aligned}$$

For convenience, let

$$\begin{aligned}
S &= -\frac{1}{2}(3-h)(3xb^2h + 3xb^2 + 2a^4 + 6abh) \\
&= \frac{1}{2}(3xb^2h^2 + 3xb^2h + 2a^4h + 6abh^2 - 9xb^2h - 9xb^2 - 6a^4 - 18abh)
\end{aligned}$$

Then we have

$$\begin{aligned}
2x[f'_{k+1}(x)]^2 + S &= \frac{1}{2}(3a^4h + 12xb^2h^2 - 24abh - 24xb^2h + 12abh^2 - 6a^4) \\
&= \frac{3}{2}(h-2)(a^4 + 4abh + 4xb^2h) \\
&= \frac{3}{2}(h-2)(a^4 + 4xa^3b + 4x^2a^2b^2) \\
&= \frac{3}{2}(h-2)(a^2 + 2xab)^2 \\
&= \frac{3}{2}[h_k(x) - 2][h'_k(x)]^2
\end{aligned}$$

71 Here we obtain  $h''_{k+1}(x) = \frac{3}{4}[3 - h_k(x)][1 - h_k(x)]h''_k(x) + \frac{3}{2}[h_k(x) - 2][h'_k(x)]^2$ . Based on  
72 Lemma.1, we know  $h_k(x) \in (0, 1)$ , so  $h''_{k+1}(x) < 0$ . Therefore, when  $x \in (0, 1)$ , then  $\forall T \in \mathbb{N}_+$ ,  
73  $h_T(x) = xf_T^2(x)$  is a strictly concave function that satisfies  $h''_T(x) < 0$  and  $h''_0(x) = 0$ .

## 74 B.2.2 Optimal Solution for the Lagrange Function

75 Based on Section B.2.1, when  $x \in (0, 1)$ , then  $\forall T \in \mathbb{N}_+$ ,  $h_T(x) = x f_T^2(x)$  is a strictly concave  
76 function that satisfies  $h_T''(x) < 0$ . So  $1 - h_T(x)$  is a strictly convex function.

We discuss  $g_j(x_1, \dots, x_d) = \sum_{i=1}^d [1 - h_T(x_i)] u_{ji}^2$  first. Denote that the Hessian Matrix of  
 $g_j(x_1, \dots, x_d)$  about  $\mathbf{x}$  is

$$\nabla^2 g_j = \begin{bmatrix} -u_{j1}^2 h_T''(x_1) & & \\ & \ddots & \\ & & -u_{jd}^2 h_T''(x_d) \end{bmatrix}$$

and the Hessian Matrix of  $g_j^2(x_1, \dots, x_d)$  about  $\mathbf{x}$  is

$$\nabla^2(g_j^2) = \nabla(2g_j \nabla g_j) = 2g_j \nabla^2 g_j + 2(\nabla g_j)(\nabla g_j)^T$$

77 We denote that all eigenvalues of  $(\nabla g_j)(\nabla g_j)^T$  are  $(\nabla g_j)^T(\nabla g_j), 0, \dots, 0$ . All eigenvalues are  
78 non-negative, denoting that  $2(\nabla g_j)(\nabla g_j)^T$  is semi-positive.

Now we denote that the Hessian Matrix of  $INTL(\mathbf{x})$  is

$$\begin{aligned} \nabla^2 INTL(\mathbf{x}) &= \sum_{j=1}^d \nabla^2(g_j^2) \\ &= 2 \sum_{j=1}^d (\nabla g_j)(\nabla g_j)^T + 2 \sum_{j=1}^d g_j \nabla^2 g_j \end{aligned}$$

where

$$2 \sum_{j=1}^d g_j \nabla^2 g_j = 2 \begin{bmatrix} -\sum_{j=1}^d u_{j1}^2 h_T''(x_1) g_j & & \\ & \ddots & \\ & & -\sum_{j=1}^d u_{jd}^2 h_T''(x_d) g_j \end{bmatrix}$$

79 We denote that  $h_T''(x_i) < 0$ ,  $g_j > 0$ , and  $u_{ji}$  are not all zeros for a certain  $i$  (since  $\sum_{j=1}^d u_{ji}^2 = 1$ ).

80 Therefore,  $-\sum_{j=1}^d u_{ji}^2 h_T''(x_i) g_j > 0$  and  $2 \sum_{j=1}^d g_j \nabla^2 g_j$  must be a positive matrix.

81 Since  $2 \sum_{j=1}^d (\nabla g_j)(\nabla g_j)^T$  is semi-positive, then we can denote that  $\nabla^2 INTL$  is positive.

82 Therefore,  $INTL(\mathbf{x})$  is strictly convex about  $\mathbf{x}$  on  $(0, 1)^d$ .

83 And for  $INTL(\mathbf{x})$  is continuous, the minimum point on  $[0, 1]^d$  is the same as that on  $(0, 1)^d$ .

84 While the constraints of  $(P_{INTL})$  form a convex set,  $(P_{INTL})$  must be a convex programming, which  
85 means that the KKT point of  $(P_{INTL})$  is its unique extreme point, and the global minimum point in  
86 the same time.

We denote that the KKT conditions of  $(P_{INTL})$  is that

$$\begin{cases} \frac{\partial L}{\partial x_i} = 0, & i = 1, \dots, d \\ \alpha_i(-x_i) = 0, & i = 1, \dots, d \\ \alpha_i \geq 0, & i = 1, \dots, d \\ \sum_{i=1}^d x_i - 1 = 0 \end{cases}$$

We can identify one of the solutions to the KKT conditions is that

$$\begin{cases} \mathbf{x} = [\frac{1}{d}, \dots, \frac{1}{d}]^T \\ \boldsymbol{\alpha} = \mathbf{0} \\ \mu = -2h'_T(\frac{1}{d})[h_T(\frac{1}{d}) - 1] \end{cases}$$

It is easy to identify the last three equations in KKT conditions. As for the first equation, for all  $t = 1, \dots, d$ , we have

$$\begin{aligned} \frac{\partial L}{\partial x_t} &= 2h'_T(x_t) \sum_{i=1}^d \sum_{j=1}^d [h_T(x_i) - 1] u_{ji}^2 u_{jt}^2 - \alpha_i + \mu \\ &= 2h'_T(\frac{1}{d}) \sum_{i=1}^d \sum_{j=1}^d [h_T(\frac{1}{d}) - 1] u_{ji}^2 u_{jt}^2 + \mu \\ &= 2h'_T(\frac{1}{d}) \sum_{j=1}^d [h_T(\frac{1}{d}) - 1] \left( \sum_{i=1}^d u_{ji}^2 \right) u_{jt}^2 + \mu \\ &= 2h'_T(\frac{1}{d}) \sum_{j=1}^d [h_T(\frac{1}{d}) - 1] u_{jt}^2 + \mu \\ &= 2h'_T(\frac{1}{d}) [h_T(\frac{1}{d}) - 1] \left( \sum_{j=1}^d u_{jt}^2 \right) + \mu \\ &= 2h'_T(\frac{1}{d}) [h_T(\frac{1}{d}) - 1] + \mu \\ &= 0 \end{aligned}$$

Therefore,  $\mathbf{x}^* = [\frac{1}{d}, \dots, \frac{1}{d}]^T$  is the optimal solution to  $(P_{INTL})$ . INTL promotes the equality of all eigenvalues in the optimization process, which provides a theoretical guarantee to avoid dimensional collapse.  $\square$

## C Algorithm of INTL

The description of our paper is based on batch whitening (BW) [8, 13], and it can extend similarly for channel whitening (CW) [20], where the covariance matrix of  $\mathbf{Z}$  is calculated as  $\Sigma = \frac{1}{d} \mathbf{Z}_c^T \mathbf{Z}_c$ . We implement INTL based on CW, considering CW is more effective when the batch size  $m$  is relatively small.

Given the centralized embedding of two positive pairs  $\mathbf{Z}_c^{(v)} := (\mathbf{I} - \frac{1}{d} \mathbf{1} \cdot \mathbf{1}^T) \mathbf{Z}^{(v)}$ ,  $\mathbf{Z}^{(v)} \in \mathbb{R}^{d \times m}$  and  $v \in \{1, 2\}$ , we first calculate the covariance matrix  $\Sigma^{(v)} = \frac{1}{d} \mathbf{Z}_c^{(v)T} \mathbf{Z}_c^{(v)}$  and then use IterNorm to obtain the approximately whitened output  $\hat{\mathbf{Z}}^{(v)} = [\hat{\mathbf{z}}_1^{(v)}, \dots, \hat{\mathbf{z}}_m^{(v)}]$ . The loss functions used in our method are

$$\mathcal{L}_{MSE} = \frac{1}{m} \sum_i \left\| \frac{\hat{\mathbf{z}}_i^{(1)}}{\|\hat{\mathbf{z}}_i^{(1)}\|_2} - \frac{\hat{\mathbf{z}}_i^{(2)}}{\|\hat{\mathbf{z}}_i^{(2)}\|_2} \right\|_2^2 \quad (11)$$

$$INTL(\mathbf{Z}^{(v)}) = \sum_{i=1}^m \left( 1 - \frac{1}{d} \hat{\mathbf{z}}_i^{(v)T} \hat{\mathbf{z}}_i^{(v)} \right)^2 \quad (12)$$

$$\mathcal{L} = \mathcal{L}_{MSE} + \beta \sum_{v=1}^2 INTL(\mathbf{Z}^{(v)}), \quad (13)$$

where  $\mathcal{L}_{MSE}$  indicates MSE of  $L_2$ -normalized vectors which minimizes the distance between  $\hat{\mathbf{Z}}^{(1)}$  and  $\hat{\mathbf{Z}}^{(2)}$ . Here we simplify the expression of INTL in Eqn. 6, because off-diagonal elements of  $\Sigma_{\hat{\mathbf{Z}}}$  does not need to be calculated.  $\beta$  is the trade-off between  $\mathcal{L}_{MSE}$  and INTL. We empirically set that

```

# f: backbone + projection
# bs: batch size
# aug: random augmentation

for x in loader: # load a minibatch x with m samples
    z1, z2 = f(aug(x)), f(aug(x)) # embedding
    t1, t2 = IterNorm(z1), IterNorm(z2)
    # trade_off between MSE and INTL Loss
    trade_off = (log2(bs) - 3) * 0.01
    loss = norm_mse(t1, t2) + trade_off * (INTL(t1) + INTL(t2))
    return loss

def IterNorm(x, iters=4): # Iterative Normalization
    M, D = x.size() # x: m * d
    x = x - x.mean(dim=1).reshape(M, 1)
    sigma = (x @ x.T) / (D - 1) # covariance matrix
    trace = sigma.diagonal().sum()
    sigma_norm = sigma / trace # normalize sigma
    P = eye(M) # identity matrix: m * m
    for _ in range(iters):
        P = 1/2 * (3 * P - matrix_power(P, 3) @ sigma_norm)
    return P / trace.sqrt() @ x

def INTL(x): # Iterative Normalization with Trace Loss
    _, D = x.size()
    d = torch.pow(x, 2).sum(axis = 1) / (D - 1)
    t1 = d.add_(-1).pow_(2).sum()
    return t1

def norm_mse(x0, x1):
    x0 = normalize(x0) # L2-normalize
    x1 = normalize(x1) # L2-normalize
    return 2 - 2 * (x0 * x1).sum(dim=-1).mean()

```

(a)

```

# f: backbone + projection
# g: momentum backbone + momentum projection
# bs: batch size
# multicrop: crop each image to 6 views:
    2 x 224 + 192 + 160 + 128 + 96
# m: momentum increases from 0.996 to 1.0 as cosine
# Function IterNorm, INTL and norm_mse are totally
    the same as Algorithm 1

for x in loader: # load a minibatch x with m samples
    s = multicrop(x) # s = [x1, x2, x3, x4, x5, x6]
    update_momentum_encoder(m)

    tk = IterNorm(g(s[0])) # use x1 to be the target
    # use x2 ~ x6 to match the target
    tq = [IterNorm(f(s[i])) for i in range(1, len(s))]

    # trade_off between MSE and INTL Loss
    trade_off = (log2(bs) - 3) * 0.01

    for i in range(len(tq)):
        loss += norm_mse(tk, tq[i]) + trade_off * INTL(tq[i])
    loss /= len(tq)
    return loss

@torch.no_grad()
def update_momentum_encoder(m): # Momentum update
    for param_f, param_g in zip(f.parameters(),
                                g.parameters()):
        param_g.data = param_g.data * m +
            param_f.data * (1. - m)

```

(b)

Figure I: Algorithm of INTL, PyTorch-style Pseudocode. (a) shows training INTL with 2 views generated from each sample. (b) shows training INTL with multi-crop and exponential moving average.

104  $\beta = 0.01 * (\log_2 bs - 3)$  where  $bs$  means the batch size, and the iteration number  $T = 4$  for all of  
 105 our experiments.

106 For clarity, we describe the algorithm of INTL in PyTorch-style pseudocode, shown in Figure I(a).  
 107 Our INTL can also work well by combining with multi-crop [1] and exponential moving  
 108 average (EMA) [3, 10] (see Section E). Figure I(b) shows the algorithm of our INTL combining with multi-  
 109 crop and EMA.

## 110 D Analytical Experiments

### 111 D.1 Experiments on Synthetic 2D dataset

112 In section 3.2 of the submitted paper, we conduct experiments on the 2D dataset and report the results  
 113 on with varying  $p$ . Here, we provide the details of the experimental setup, and further show the results  
 114 of IterNorm [14] for SSL in this 2D dataset.

#### 115 D.1.1 Details of Experimental Setups

116 We synthesize a two-dimensional dataset with isotropic Gaussian blobs containing 512 sample points  
 117 as shown in Figure II(a). We construct a toy Siamese network (a simple three-layer neural network,  
 118 including three fully connected (FC) layers, with BN and ReLU appended to the first two) as the  
 119 encoder for this dataset. The dimensions of the network are  $(2 - 16) - (16 - 16) - (16 - 2)$  that  
 120 each bracket represents the input and output dimensions of each FC layer respectively. We then use  
 121 MSE as the loss function and do not normalize the features before calculating the loss function.

122 We train the model by randomly shuffling the data into mini-batches, and set the batch size to 32. We  
 123 use the stochastic gradient descent (SGD) algorithm with a learning rate of 0.1. In terms of the data



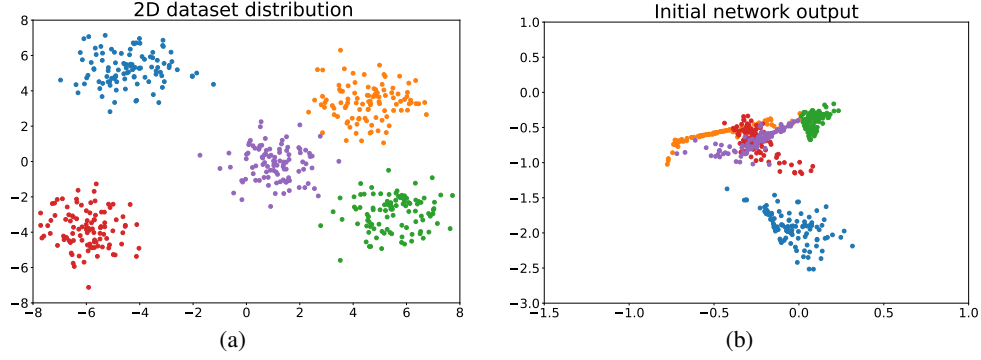


Figure II: Visualization of our synthetic 2D dataset. We show (a) the distribution of our 2D dataset; (b) the initial output of the toy Siamese network.

transformation, we only apply Gaussian noise as data augmentation and generate 2 views from each sample point in mini-batches. We visualize the output of the initialized network without training in Figure II(b). All runs are performed under the same random seed.

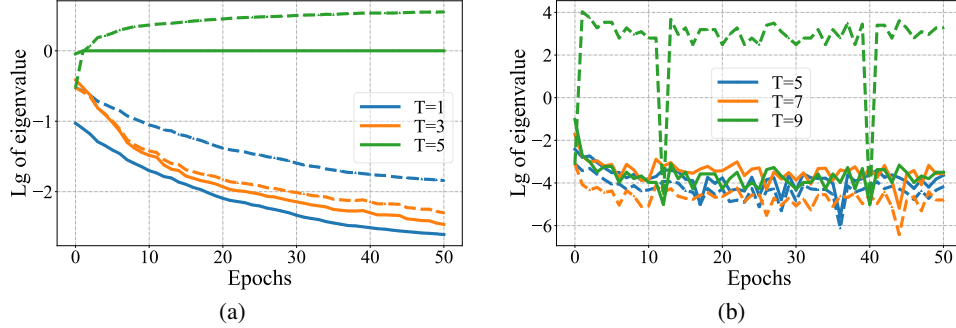


Figure III: Investigate the spectrum of transformed output  $\hat{\mathbf{Z}}$  (solid lines) and the corresponding embedding  $\mathbf{Z}$  (dashed lines) using IterNorm for SSL with different iteration numbers  $T$ . We show the evolution of eigenvalues during training on the toy 2D dataset (Note that there are only two eigenvalues and we ignore the larger one because it always remains a high value during training). In particular, (a) shows the results with a well-conditioned initial spectrum while (b) with a ill-conditioned one.

### D.1.2 Results of IterNorm for SSL

To figure out the failure of IterNorm [14] for SSL, we further conduct experiments to investigate the spectrum of the whitened output  $\hat{\mathbf{Z}}$  using IterNorm on this synthetic 2D dataset for intuitive analyses. The output dimension of the toy model is 2, so there are only two eigenvalues of the covariance matrix of the output. We then track alterations of the two eigenvalues during training. IterNorm can obtain an idealized whitened output with a small iteration number (*e.g.*,  $T=5$ , as recommend in [14]) and avoid collapse, if the embedding  $\mathbf{Z}$  has a well-conditioned spectrum<sup>1</sup> (Figure III(a)). However, if the embedding  $\mathbf{Z}$  has a ill-conditioned spectrum as shown in Figure III(b), IterNorm fails to pull the small eigenvalue to approach 1 which results in dimensional collapse.

## D.2 Experiments on CIFAR-10

In section 3 and 4 of the submitted paper, we conduct several experiments on CIFAR-10 to illustrate our analysis. We provide a brief description of the setup in the caption of Figure 1 and 2 of the submitted paper. Here, we describe the details of these experiments. All experiments are uniformly based on the following training settings, unless otherwise stated in the figures of the submitted paper.

<sup>1</sup>A well-conditioned spectrum means that the condition number  $c = \frac{\lambda_1}{\lambda_d}$  is small. Note  $\lambda_1$  is the maximum eigenvalue and  $\lambda_d$  is the minimum one.

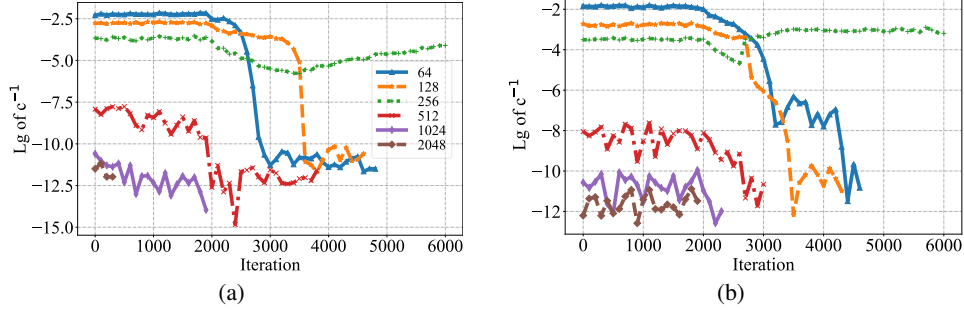


Figure IV: Investigate numerical instability of spectral transformation using power functions for SSL. The numbers in the legend represent embedding dimensions and the batch size is fixed to 512. (a) trains models on ImageNet with ResNet-50; (b) trains models on CIFAR-10 with ResNet-18; The models are trained for 6000 iterations, and we track the inverse of condition number ( $c^{-1} = \frac{\lambda_d}{\lambda_1}$ ) in logarithmic scale with base 10 to judge whether the the covariance matrix is ill-conditioned. The models that were interrupted before the end of the training indicate training crash caused by numerical instability.

**Training Settings.** We use the ResNet-18 as the encoder (the dimension of *encoding* is 512.), a two layer MLP with ReLU and BN appended as the projector (the dimension of the hidden layer and embedding are 1024 and 128 respectively). The model is trained on CIFAR-10 with a batch size of 256, using Adam optimizer [15] with a learning rate of  $3 \times 10^{-3}$ , and learning rate warm-up for the first 500 iterations and a 0.2 learning rate drop at the last 50 and 25 epochs. The weight decay is set as  $10^{-6}$ . All transformations are performed with 2 positives extracted per image with standard data argumentation (see Section E.3 for details). We use the same evaluation protocol as in *W-MSE* [8].

**Method Settings.** We use MSE of  $L_2$ -normalized vectors to be the loss function in all experiments. Specifically, in Figure 3 of the paper for the experiments of training the models with INTL, we simply set the trade-off parameter  $\beta$  between MSE and INTL as follows:  $\beta = 0.05$  for  $T = 5$ ,  $\beta = 0.5$  for  $T = 3$  and  $\beta = 5$  for  $T = 1$  without fine-tuning. The details of INTL algorithm please refer to Section C.

### D.3 Numerical Instability of Spectral Transformation Using Power Functions

One problem in using spectral transformation  $g(\lambda) = \lambda^{-p}$  ( $p$  is around 0.5) is the numerical instability, when calculating eigenvalues  $\lambda$  and eigenvectors  $\mathbf{U}$  using eigen-decomposition if the covariance matrix is ill-conditioned [18]. Here, we experimentally confirm the existence of this phenomenon during self-supervised pre-training. It is worth noting that when we set  $p$  to around 0.5, similar phenomena can be observed. We thus only display the results of the special instance  $p = 0.5$ , which is the so-called hard whitening.

To confirm that this phenomenon could occur on different scenarios, we conduct experiments on ImageNet with ResNet-50, as well as on CIFAR-10 with ResNet-18. The batch size  $m$  is fixed to 512, and we can control the shape of the covariance matrix by adjusting the embedding dimension  $d$  (The shape of the covariance matrix is  $d \times d$ ). The models are trained for 6000 iterations, and we track the inverse of condition number ( $c^{-1} = \frac{\lambda_d}{\lambda_1}$ ) to judge whether the the covariance matrix is ill-conditioned. The experimental results are shown in Figure IV and our main observations are as follows:

(a) The training will crash, when the embedding dimension is greater than the batch size (e.g.,  $d = 1024$  or  $2048$ ). In this case, the covariance matrix must be singular theoretically and the calculation of inverse of the eigenvalue will cause numerical errors. However, it is likely that the minimum eigenvalue of the covariance matrix is a very small non-zero value in practice, due to precision rounding or using an extra small constant. This situation may lead to the covariance matrix being ill-conditioned from the beginning of training. As shown in both Figure IV(a) and (b), when  $d = 1024$  or  $2048$ , the inverse of condition number is around  $10^{-12} \sim 10^{-10}$ , which demonstrates that the covariance matrix is almost ill-conditioned from the start and the training quickly breaks down.

(b) The training will probably crash, when the embedding dimension is equal to the batch size ( $d = 512$ ). In this case, it is difficult to determine whether the covariance matrix is singular. But from

the results in Figure IV, we observe that the covariance matrix is close to be ill-conditioned when  $d = 512$ . The inverse of condition number tends to decline during training, ultimately leading to the crash of the training.

(c) There is possibility that the training will crash, when the embedding dimension is less than the batch size. In this case, we observe that the covariance matrix is almost always well-conditioned during the initial training stage. However, the well-condition does not seem to be always maintained during training. We observe that the well-condition will suddenly be broken in a few iterations and the models will collapse for  $d = 64$  or  $d = 128$ . We indeed observe that the training does not crash when  $d = 256$ . This phenomenon was also mentioned slightly in [8], indicating that the training can be more stable by setting  $m = 2d$ .

We show that numerical instability indeed exists when using hard whitening [8], from the above analysis. Although one can alleviate this numerical instability by using an empirical setting with  $m = 2d$ , we observe training crashes caused by numerical instability can still occur at any stage of training through our experiments (we run 10 random seeds by setting  $m = 2d$  with longer training iterations, and the numerical problems may occur 3 – 4 times in the early, mid, or even towards the end of training.). Even though one can continue the training by using the saved checkpoints if training crashes in practice, it greatly limits the practical application in long-term pre-training.

## E Details of Experiments on Standard SSL Benchmark

In this section, we provide the details of implementation and training protocol for the experiments on large-scale ImageNet [7], medium-scale ImageNet-100 [19] and small-scale CIFAR-10/100 [16] classification as well as transfer learning to COCO [17] object detection and instance segmentation. We also provide computational overhead of INTL pre-training on ImageNet.

### E.1 Datasets

- CIFAR-10 and CIFAR-100 [16], two small-scale datasets composed of  $32 \times 32$  images with 10 and 100 classes, respectively.
- ImageNet-100 [19], a random 100-class subset of ImageNet [7].
- ImageNet [7], the well-known largescale dataset with about 1.3M training images and 50K test images, spanning over 1000 classes.
- COCO2017 [17], a large-scale object detection, segmentation, and captioning dataset with 330K images containing 1.5 million object instances.

### E.2 Experiment on ImageNet

In section 5.1 of the paper, we compare our INTL to the state-of-the-art SSL methods on large-scale ImageNet classification. Here, we describe the training details of these experiments.

**Backbone and Projection.** We use the ResNet-50 [12] as the backbone and the output dimension is 2048. We use a 3-layers MLP as the projection: two hidden layers with BN and ReLU applied to it and a linear layer as output. The dimensions of the hidden layer and embedding are all 8192.

**Image Transformation Details.** In image transformation, We use the same augmentation parameters as BYOL [10]. Each input image is transformed twice to produce the two distorted views. The image augmentation pipeline consists of the following transformations: random cropping, resizing to  $224 \times 224$ , horizontal flipping, color jittering, converting to grayscale, Gaussian blurring, and solarization. The details of parameters are shown in Table A.

**Optimizer and Learning Rate Schedule.** We apply the SGD optimizer, using a learning rate of  $base-lr \times BatchSize / 256$  and cosine decay schedule.

Table A: Parameters used for image augmentations on ImageNet and ImageNet-100.

Parameter	s	
	$T_1$	$T_2$
crop size	$224 \times 224$	$224 \times 224$
maximum scale of crops	1.0	1.0
minimum scale of crops	0.08	0.08
brightness	0.4	0.4
contrast	0.4	0.4
saturation	0.2	0.2
hue	0.1	0.1
color jitter prob	0.8	0.8
horizontal flip prob	0.5	0.5
gaussian prob	1.0	0.1
solarization prob	0.0	0.2

Table B: Parameters used for multi-crop of INTL on ImageNet.

Parameter	$T_1$	$T_2$	$T_3$	$T_4$	$T_5$	$T_6$
crop size	$224 \times 224$	$224 \times 224$	$192 \times 192$	$160 \times 160$	$128 \times 128$	$96 \times 96$
maximum scale of crops	1.0	1.0	0.857	0.714	0.571	0.429
minimum scale of crops	0.2	0.2	0.171	0.143	0.114	0.086
brightness	0.4	0.4	0.4	0.4	0.4	0.4
contrast	0.4	0.4	0.4	0.4	0.4	0.4
saturation	0.2	0.2	0.2	0.2	0.2	0.2
hue	0.1	0.1	0.1	0.1	0.1	0.1
color jitter prob	0.8	0.8	0.8	0.8	0.8	0.8
horizontal flip prob	0.5	0.5	0.5	0.5	0.5	0.5
gaussian prob	0.5	0.5	0.5	0.5	0.5	0.5
solarization prob	0.1	0.1	0.1	0.1	0.1	0.1

Table D: Comparisons on ImageNet linear classification with various training epochs. All are based on ResNet-50 backbone. The table is mostly inherited from [4].

Method	Bs	EMA	Multi-Crop	100 eps	200 eps	400 eps	800 eps
SimCLR	4096	×	×	66.5	68.3	69.8	70.4
MoCo v2	256	✓	×	67.4	69.9	71.0	72.2
BYOL	4096	✓	×	66.5	70.6	73.2	74.3
SwAV	4096	×	×	66.5	69.1	70.7	71.8
	4096	×	✓	72.1	73.9	74.6	75.3
	256	×	✓	-	72.7	74.3	-
SimSiam	256	×	×	68.1	70.0	70.8	71.3
W-MSE	4096	×	✓	69.4	-	72.6	-
CW-RGP	512	×	✓	69.7	71.0	-	-
<b>INTL (ours)</b>	512	×	×	69.5	71.1	72.4	73.1
	256	✓	×	69.2	71.5	-	74.3
	256	×	✓	72.4	74.3	74.9	-
	256	✓	✓	<b>73.5</b>	<b>75.2</b>	<b>76.1</b>	<b>76.6</b>

225 The *base-lr* for 100-epoch pre-training is 0.5, for 200(400)-epoch is 0.4 and for 800-epoch is 0.3. The  
226 weight decay is  $10^{-5}$  and the SGD momentum is 0.9. In addition, we use learning rate warm-up for  
227 the first 2 epochs of the optimizer.

228 **Evaluation Protocol.** For linear classification, we  
229 train the *linear classifier* for 100 epochs with SGD  
230 optimizer (using a learning rate of *base-lr*  $\times$  Batch-  
231 Size / 256 with a *base-lr* of 0.2) and using *Multi-  
232 StepLR* scheduler with  $\gamma = 0.1$  dropping at the last  
233 40 and 20 epochs. While for semi-supervised clas-  
234 sification, we fine-tune our pre-trained INTL back-  
235 bone and train the *linear classifier* for 20 epochs.  
236 We use SGD optimizer (*base-lr* for backbone is  
237 0.006 and for classifier is 0.2) and cosine decay  
238 schedule. The batch size and weight decay for both  
239 are 256 and 0 respectively.

#### 240 Multi-Crop and Exponential Moving Average.

241 Note that multi-crop [1] and exponential moving  
242 average (EMA) [3, 10] are commonly acknowledged strategies that can improve the performance  
243 of SSL methods. *e.g.*, BYOL achieves a high Top-1 accuracy of 74.3% by applying EMA and  
244 SwAV achieves 75.3% with multi-crop. We thus also experiment with INTL that uses these two  
245 strategies. We propose an efficient multi-crops variety that crops each image to 6 views with the  
246 size of  $2 \times 224 + 192 + 160 + 128 + 96$  (details of parameters are shown in Table B). Meanwhile,

Table C: Parameters used for INTL pre-training on ImageNet-100.

Parameter	Value
max epoch	400
backbone	ResNet-18
projection layers	3
projection hidden dimension	4096
projection output dimension	4096
optimizer	SGD
SGD momentum	0.9
learning rate	0.5
learning rate warm-up	2 epochs
learning rate schedule	cosine decay
weight decay	$2.5e-5$
batch size	128

EMA we used is asymmetric as MoCo that reduces memory overhead and accelerates training speed. We set the base coefficient for momentum updating to 0.996 for all-epoch training. The momentum coefficient follows a cosine increasing schedule with final value of 1.0 as BYOL [10]. Note that for linear classification, the *base-lr* is 0.4 and for semi-supervised classification, the *base-lr* for backbone is 0.004. The other settings are the same as the baseline. Benefiting from these two strategies, our INTL achieves a Top-1 accuracy of 75.2% with only 200-epoch pre-training. For long-term training of 800 epochs, our INTL achieves a Top-1 accuracy of 76.6% which exceeds the performance of the supervised baseline [2] and other SSL methods. We also provide the results using various epochs in Table D, from which we observe that INTL improves the performance steadily as the training epoch increases.

### E.3 Experiments for Small and Medium Size Datasets

In section 5.1 of the paper, we provide the classification results of INTL pre-training on small and medium size datasets such as CIFAR-10, CIFAR-100 and ImageNet-100. Here, We describe the details of implementation and training protocol for the experiments on these datasets as follows. For fairness, most of hyper-parameters we used such as batch size, projection settings, data augmentation and so on are consistent with solo-learn [6]. For these datasets, we use the basic INTL shown in Algorithm I(a).

**Experimental setup on ImageNet-100.** Details of implementation and training protocol for INTL pre-training on ImageNet-100 are shown in Table C. The image transformation and evaluation protocol are the same as ones on ImageNet.

**Experimental setup on CIFAR-10/100.** Then Details of implementation and training protocol for INTL pre-training on CIFAR-10/100 are shown in Table E. The details of image transformation are shown in Table F. For evaluation, we use the same setup of protocol as in *W-MSE* [8]: training the linear classifier for 500 epochs using the Adam optimizer and the labeled training set of each specific dataset, without data augmentation; the learning rate is exponentially decayed from  $10^{-2}$  to  $10^{-6}$  and the weight decay is  $5 \times 10^{-6}$ .

In addition, we also evaluate the accuracy of a k-nearest neighbors classifier (k-NN,  $k = 5$ ) in these experiments. For other methods, we evaluate the models provided by [6] to obtain k-NN accuracy which does not require additional parameters and training.

Table E: Parameters used for INTL pre-training on CIFAR-10/100.

Parameter	Value
max epoch	1000
backbone	ResNet-18
projection layers	3
projection hidden dimension	2048
projection output dimension	2048
optimizer	SGD
SGD momentum	0.9
learning rate	0.3
learning rate warm-up	2 epochs
learning rate schedule	cosine decay
weight decay	1e-4
batch size	256

### E.4 Experiments for Transfer Learning

In this part, we describe the training details of experiments for transfer learning. Our implementation is based on the released codebase of MoCo [11]<sup>2</sup> for transfer learning to object detection and instance segmentation tasks. We use the default hyper-parameter configurations from the training scripts provided by the codebase for INTL, using our 200-epoch and 800-epoch pre-trained model on ImageNet.

For the experiments of *COCO detection* and *COCO instance segmentation*, we use Mask R-CNN (1x schedule) fine-tuned in COCO 2017 train, evaluated in COCO 2017 val. The Mask R-CNN model is with the C4-backbone. Our INTL is performed with 3 random seeds, with mean and standard deviation reported.

### E.5 Computational Overhead

In Table G, we report compute and GPU memory requirements based on our implementation for different settings. We train each model with 2 A100-PCIE-40GB GPUs, using mixed precision

<sup>2</sup><https://github.com/facebookresearch/moco/tree/main/detection> under the CC-BY-NC 4.0 license.

Table F: Parameters used for image augmentations on CIFAR-10/100.

Parameter	$T_1$	$T_2$
crop size	$32 \times 32$	$32 \times 32$
maximum scale of crops	1.0	1.0
minimum scale of crops	0.08	0.08
brightness	0.4	0.4
contrast	0.4	0.4
saturation	0.2	0.2
hue	0.1	0.1
color jitter prob	0.8	0.8
horizontal flip prob	0.5	0.5
gaussian prob	0	0
solarization prob	0.0	0.2

and py-torch optimized version of synchronized batch-normalization layers. We report results with ResNet-50 and a batch size of 256.

Table G: Computational cost. We report time and GPU memory requirements of our implementation for INTL trained per epoch.

Method	EMA	Multi-Crop	time / 1 epoch	peak memory / GPU
<b>INTL</b>	✗	✗	29min11	16.0 G
	✓	✗	24min46	11.8 G
	✗	✓	57min33	25.9 G
	✓	✓	50min52	21.2 G

## F Licenses of Datasets

ImageNet [7] is subject to the ImageNet terms of access: [5]

COCO [17]. The annotations are under the Creative Commons Attribution 4.0 License. The images are subject to the Flickr terms of use [9].

## References

- [1] Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., Joulin, A.: Unsupervised learning of visual features by contrasting cluster assignments. In: NeurIPS (2020)
- [2] Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: ICML (2020)
- [3] Chen, X., Fan, H., Girshick, R., He, K.: Improved baselines with momentum contrastive learning. arXiv preprint arXiv:2003.04297 (2020)
- [4] Chen, X., He, K.: Exploring simple siamese representation learning. In: CVPR (2021)
- [5] contributors, I.: Imagenet terms of access (2020), <https://image-net.org/download>
- [6] da Costa, V.G.T., Fini, E., Nabi, M., Sebe, N., Ricci, E.: solo-learn: A library of self-supervised methods for visual representation learning. Journal of Machine Learning Research **23**(56), 1–6 (2022), <http://jmlr.org/papers/v23/21-1155.html>
- [7] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR (2009)
- [8] Ermolov, A., Siarohin, A., Sangineto, E., Sebe, N.: Whitening for self-supervised representation learning. In: ICML (2021)
- [9] Flickr, I.: Flickr terms and conditions of use (2020), <http://aiweb.techfak.uni-bielefeld.de/content/bworld-robot-control-software/>

- 320 [10] Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C.,  
321 Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., Piot, B., kavukcuoglu, k., Munos, R., Valko, M.:  
322 Bootstrap your own latent - a new approach to self-supervised learning. In: NeuralIPS (2020)
- 323 [11] He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual  
324 representation learning. In: CVPR (2020)
- 325 [12] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR  
326 (2016)
- 327 [13] Hua, T., Wang, W., Xue, Z., Ren, S., Wang, Y., Zhao, H.: On feature decorrelation in self-  
328 supervised learning. In: ICCV (2021)
- 329 [14] Huang, L., Zhou, Y., Zhu, F., Liu, L., Shao, L.: Iterative normalization: Beyond standardization  
330 towards efficient whitening. In: CVPR (2019)
- 331 [15] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980**  
332 (2014)
- 333 [16] Krizhevsky, A.: Learning multiple layers of features from tiny images. Tech. rep. (2009)
- 334 [17] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollar, P., Zitnick, L.:  
335 Microsoft coco: Common objects in context. In: ECCV (2014)
- 336 [18] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z.,  
337 Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning  
338 library. Advances in neural information processing systems **32** (2019)
- 339 [19] Tian, Y., Krishnan, D., Isola, P.: Contrastive multiview coding. In: European conference on  
340 computer vision (2020)
- 341 [20] Weng, X., Huang, L., Zhao, L., Anwer, R.M., Khan, S., Khan, F.: An investigation into  
342 whitening loss for self-supervised learning. In: NeurIPS (2022)