HOW DOES SIMSIAM AVOID COLLAPSE WITHOUT NEGATIVE SAMPLES? TOWARDS A UNIFIED UNDERSTANDING OF PROGRESS IN SSL

Anonymous authors
Paper under double-blind review

ABSTRACT

Towards avoiding collapse in self-supervised learning (SSL), contrastive loss is widely used but often requires a large number of negative samples. Without negative samples yet achieving competitive performance, a recent work (Chen & He, 2021) has attracted significant attention for providing a minimalist simple Siamese (SimSiam) method to avoid collapse. However, the reason for its success remains not fully clear and our investigation starts by revisiting the explanatory claims in the SimSiam. After refuting their claims, we introduce vector decomposition for analyzing the collapse based on the gradient analysis of l_2 normalized vector. This yields a unified perspective on how negative samples and SimSiam predictor alleviate collapse. Such a unified perspective comes timely for understanding the recent progress in SSL.

1 INTRODUCTION

Beyond the success of NLP (Lan et al., 2020; Radford et al., 2019; Devlin et al., 2019; Su et al., 2020; Nie et al., 2020), self-supervised learning (SSL) has also shown its potential in the field of vision tasks (Li et al., 2021; Chen* et al., 2021; El-Nouby et al., 2021). Without the ground-truth label, the core of most SSL methods lies in learning an encoder with augmentation invariant representation (Bachman et al., 2019; He et al., 2020; Chen et al., 2020a; Caron et al., 2020; Grill et al., 2020). Specifically, they often minimize the representation distance between two positive samples, *i.e.* two augmented views of the same image, based on a Siamese network architecture (Bromley et al., 1993). It is widely known that for such Siamese networks there exists a degenerate solution, *i.e.* all outputs "collapsing" to an undesired constant (Chen et al., 2020a; Chen & He, 2021). Early works have attributed the collapse to lacking a repulsive component in the optimization goal and adopted negative samples, *i.e.* views of different samples, to alleviate this problem. Introducing momentum into the target encoder, BYOL shows that Siamese architectures can be trained with only positive pairs. A technical report (Fetterman & Albrecht, 2020) has suggested the importance of batch normalization in BYOL for its success, however, a recent work Richemond et al. (2020) refutes their claim by showing BYOL works without BN.

More recently, SimSiam (Chen & He, 2021) has caught great attention by further simplifying BYOL in terms of removing momentum encoder, which can be seen as a major milestone achievement in SSL through providing a *minimalist* method for achieving competitive performance. However, the reason for the success of Simsiam remains not fully clear with a core question summarized as:

How does SimSiam avoid collapse without negative samples?

Our work attempts to answer the above question and our investigation starts with revisiting the explanatory claims in the original SimSiam paper. Notably, two components, *i.e.* stop gradient and predictor, are observed to be essential for the success of SimSiam (Chen & He, 2021). The reason has been mainly attributed to the stop gradient (Chen & He, 2021) by hypothesizing that it implicitly involves two sets of variables and SimSiam behaves like alternating between optimizing each set. Chen & He argue that the predictor h is helpful in SimSiam because h fills the gap to approximate expectation over augmentations (EOA).

Unfortunately, the above explanatory claims are found to be flawed due to reversing the two paths with and without gradient. This motivates us to find an alternative explanation, for which we in-

troduce a simple yet intuitive framework for facilitating analyzing collapse in SSL. Specifically, we propose to decompose a representation vector into center and residual components. This decomposition facilitates understanding which gradient component is beneficial for avoiding collapse. Under this framework, we attempt to understand why a Naive Siamese architecture causes collapse, and provide a unified perspective on how negative samples help prevent such collapse as well as how predictor in SimSiam achieves it *without* negative samples. For both InfoNCE with negative samples (He et al., 2019; Chen et al., 2020b;a; Tian et al., 2019; Khosla et al., 2020) and SimSiam with asymmetric predictor, their center gradient component behaves like de-centering for preventing collapse. Interestingly, we also find that for both, the residual vector component achieves de-correlation which somewhat surprisingly also prevents collapse. Overall, our work demystifies the success of InfoNCE and SimSiam, which bridges the gap between them and recent frameworks with explicit de-centering and de-correlation. Our work contributes to a unified understanding of the recent progress of different frameworks in SSL.

Finally, our work points out an important detail on whether BN, which removes the mean center is essential for preventing a collapse in BYOL or SimSiam. Specifically, we reveal that BN alone prevents collapse only when BN is adopted immediately after the MSE loss. Towards more explainable SimSiam our work investigates multiple new variants of predictors for preventing collapse. We follow prior works to set the experiment, see Appendix A.1.

2 Revisiting SimSiam and its explanatory claims

l_2 normalized vector and optimization goal. SSL trains an encoder f for learning discriminative representation and we denote such representation as a vector $\boldsymbol{z}$, *i.e.* $f(x) = \boldsymbol{z}$ where x is a certain input. For the augmentation-invariant representation, a straightforward goal is to minimize the distance between the representations of two positive samples, *i.e.* augmented views of the same image, for which mean squared error (MSE) is a default choice. To avoid scale ambiguity, the vectors are often l_2 normalized, *i.e.* $\boldsymbol{Z} = \boldsymbol{z}/||\boldsymbol{z}||_2$ (Chen & He, 2021), before calculating the MSE:

$$\mathcal{L}_{MSE} = (\boldsymbol{Z}_a - \boldsymbol{Z}_b)^2/2 - 1 = -\boldsymbol{Z}_a \cdot \boldsymbol{Z}_b = L_{cosine}, \tag{1}$$

which shows the equivalence of normalized MSE loss to cosine loss (Grill et al., 2020).

Collapse in SSL and solution of SimSiam. Based on a Siamese architecture, the loss in Eq. 1 causes the collapse, i.e. f always outputs the same constant regardless of the input. We refer this Siamese architecture with loss Eq. 1 to *Naive Siamese Model* in the remainder of paper. Towards alleviating this phenomenon, multiple works have investigated solutions, among them contrastive loss with negative samples is widely used (Chen et al., 2020a). (Chen & He, 2021) solves the collapse problem via optimizing the encoder with a symmetric loss shown as follows:

$$L_{SimSiam} = -(\boldsymbol{P}_a \cdot \text{sg}(\boldsymbol{Z}_b) + \boldsymbol{P}_b \cdot \text{sg}(\boldsymbol{Z}_a)), \tag{2}$$

where sg indicates stop gradient and $\boldsymbol{P}$ is the output of predictor h, *i.e.* $\boldsymbol{p} = h(\boldsymbol{z})$ and $\boldsymbol{P} = \boldsymbol{p}/||\boldsymbol{p}||_2$.

2.1 Revising explanatory claims in SimSiam

Overall, Chen & He have interpreted sg in Eq. 2 as an Alternating Optimization (AO) problem where h is claimed to behave as Expectation Over Augmentations (EOA). In the following, we revisit this claim in detail and point out a hidden flaw in their reasoning.

Interpreting stop gradient as AO. Chen & He hypothesize that the stop gradient in Eq. 2 is an implementation of alternating between the optimization of two sub-problems, which is defined as Alternating Optimization (AO). Specifically, with the loss considered as $\mathcal{L}(\theta, \eta) = \mathbb{E}_{x,\mathcal{T}}\Big[\big\|\mathcal{F}_\theta(\mathcal{T}(x)) - \eta_x\big\|_2^2\Big]$, the optimization objective $\min_{\theta,\eta} \mathcal{L}(\theta, \eta)$ can be solved by $\eta^t \leftarrow \arg\min_\eta \ \mathcal{L}(\theta^t, \eta)$ and $\theta^{t+1} \leftarrow \arg\min_\theta \ \mathcal{L}(\theta, \eta^t)$ alternatively. It is acknowledged in (Chen & He, 2021) that their hypothesis does not fully explain why the collapse is prevented. Nonetheless, they mainly attribute SimSiam success to the stop gradient and the interpretation that the AO problem might provide a different trajectory that makes it difficult for the alternating optimizer to approach a constant for $\forall x$.

Interpreting predictor as EOA. Their AO problem (Chen & He, 2021) is formulated independent of predictor h, for which they believe that the usage of predictor h is related to approximating Expectation Over Augmentation (EOA) for filling the gap of ignoring $\mathbb{E}_\mathcal{T}[\cdot]$ in a sub-problem of AO. The

approximation of $\mathbb{E}_{\mathcal{T}}[\cdot]$ is detailed as follows. In the sub-problem $\eta^t \leftarrow \arg\min_\eta \ \mathcal{L}(\theta^t, \eta)$, η^t indicating latent representation of images at step t is actually obtained through $\eta_x^t \leftarrow \mathbb{E}_{\mathcal{T}}\Big[\mathcal{F}_{\theta^t}(\mathcal{T}(x))\Big]$, where they in practice ignore $\mathbb{E}_{\mathcal{T}}[\cdot]$ and sample only one augmentation $\mathcal{T}'$, *i.e.* $\eta_x^t \leftarrow \mathcal{F}_{\theta^t}(\mathcal{T}'(x))$. Conceptually, Chen & He equate the role of predictor to EOA. Experimentally, they support this equivalence by proof-of-concept experiments, detailed as follows. Compared with the original model of Simsiam in Fig 1 (a), Chen & He first shows that updating η_x with a moving-average $\eta_x^t \leftarrow m * \eta_x^t + (1-m) * \mathcal{F}_{\theta^t}(\mathcal{T}'(x))$ can help the model prevent collapse even without predictor (as shown in Fig 1 (b)). Given that the training completely fails when the predictor and moving average are both removed, at first sight, their reasoning seems to be valid.

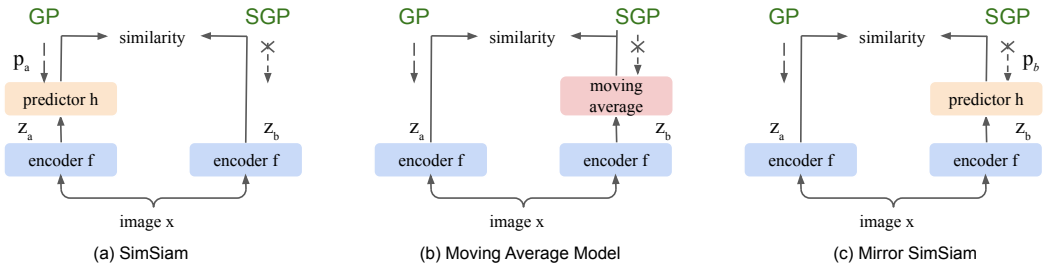


Figure 1: **Reasoning Flaw in SimSiam.** (a) The standard SimSiam architecture. (b) Moving-Average Model proposed as the proof-of-concept experiment. (c) Mirror SimSiam, which has the same model architecture with SimSiam while the reverse of GP and SGP.

2.2 Does the predictor fill the gap to approximate EOA?

Reasoning flaw. Taking the stop gradient into account, we divide SimSiam model into two sub-models with different paths and term them Gradient Path (GP) and Stop Gradient Path (SGP). For SimSiam, as shown in Fig1 (a), the sub-model with GP includes the predictor in Simsiam, while the sub-model with SGP only includes the encoder. We point out that in Simsiam, the reasoning flaw of predictor analysis lies in the *reverse of GP and SGP*. Specifically, we claim that their proof-of-concept experiment could not support the equivalence of the predictor and EOA due to the reverse of GP and SGP. It should be noted that the SGP is by default on the same side of moving-average sub-model, as shown in Fig1 (b). Conceptually, Fig1 (b) is similar to Fig1 (c) instead of Fig1 (a). It is worth mentioning that the Mirror SimSiam in Fig1 (c) is what stop gradient in the original SimSiam avoids. If the predictor of SimSiam is in the same location as the moving-average sub-model, as shown in Fig1 (b), one may conclude that there is an equivalence between predictor and EOA. Therefore, the conceptual equivalence of the predictor h (Fig1 (a)) to EOA (Fig1 (b)) does not hold and their equivalence claim has a reasoning flaw due to reversing GP and SGP .

Method	# aug	Collapse	Std	Accuracy (%)
Moving average	2	×	0.0108	46.57
Same batch	10	✓	0	1
Same batch	25	✓	0	1

Table 1: Explicit EOA in the same batch does not prevent collapse. We re-implemented the moving average experiment in the original Simsiam on CIFAR-100 dataset.

Explicit EOA does not prevent collapse. (Chen & He, 2021) points out that "in practice, it would be unrealistic to actually compute the expectation $\mathbb{E}_{\mathcal{T}}[\cdot]$. But it may be possible for a neural network (e.g., the preditor h) to learn to predict the expectation, while the sampling of $\mathcal{T}$ is implicitly distributed across multiple epochs." If implicitly sampling across multiple epochs is beneficial, explicitly sampling sufficient large N augmentations in a batch with the latest model would be more beneficial to approximate $\mathbb{E}_{\mathcal{T}}[\cdot]$. However, Table 1 showd that the collapse still occurs and suggests the success of moving average should not be interpreted from the EOA perspective.

2.3 ASYMMETRIC INTERPRETATION OF PREDICTOR WITH STOP GRADIENT IN SIMSIAM

Symmetric Predictor does not prevent collapse. What is the difference between Naive Siamese Model and Simsiam? Their difference lies in whether the gradient in backward propagation flows through the predictor, however, we show that this propagation helps avoid collapse only when the predictor is not included in the SGP path. With h being trained the same as Eq. 2, we optimize the encoder f through replacing the $\boldsymbol{Z}$ in Eq. 2 with $\boldsymbol{P}$. The results in Fig. 3 (b) show that it still leads to collapse. Actually, this is well expected by perceiving h to be part of the new encoder F, *i.e.* $\boldsymbol{p} = F(x) = h(f(x))$. In other words, the symmetric architectures with and without predictor h both lead to collapse.

Predictor with stop gradient is asymmetric. Clearly, the success of SimSiam lies in the asymmetric architecture, *i.e.* one path with predictor and the other without predictor. Under this asymmetric architecture, the role of stop gradient is to only allow the path with predictor to be optimized with the encoder output as target, not vice versa. In other words, the SimSiam avoids collapse by excluding Mirror SimSiam (Fig1 (c)) which has a loss (mirror-like Eq 2) as shown as:

$$\mathcal{L}_{Mirror} = -(\text{sg}(\boldsymbol{P}_a) \cdot \boldsymbol{Z}_b + \text{sg}(\boldsymbol{P}_b) \cdot \boldsymbol{Z}_a). \tag{3}$$

Predictor *vs.* inverse predictor. We interpret h as a function mapping from $\boldsymbol{z}$ to $\boldsymbol{p}$, and introduce an inverse h^{-1} for mapping from $\boldsymbol{p}$ to $\boldsymbol{z}$, *i.e.* $\boldsymbol{z} = h^{-1}(\boldsymbol{p})$. With the h^{-1}, we interpret Fig2 (b) differs from Fig1 (a) as changing the optimized target from $\boldsymbol{p}$ to $\boldsymbol{z}$, *i.e.* $h^{-1}(\boldsymbol{p})$, suggesting processing the optimized target with h^{-1} helps prevent collapse. Similarly, Fig1 (c) and Fig2 (a) both lead to success, suggesting processing the optimized target with h is not beneficial for preventing collapse. Overall, the asymmetry alone does not guarantee collapse avoidance, but also requires the optimized target to be processed by h^{-1} not h. This motivates us to explore what is the role of h and/or h^{-1}.

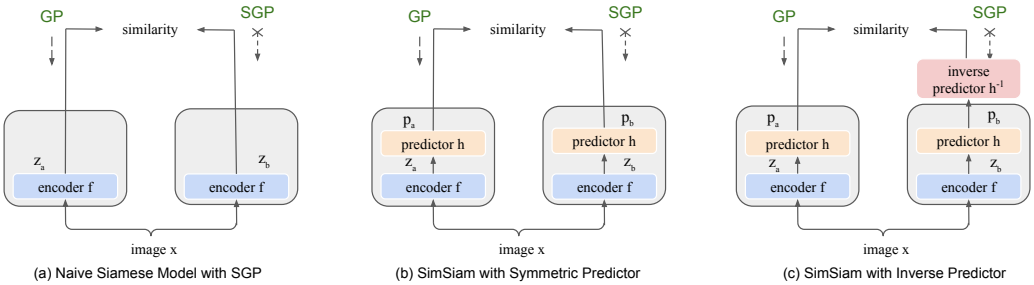


Figure 2: Different architectures of Siamese model (a), (b), and (c). Here in (c), an additional inverse predictor h^{-1} is designed that has same architecture as predictor h.

Trainable h^{-1} and its implication on EOA. In the above, we propose a *conceptual* inverse predictor h^{-1} in Fig2 (c), however, it remains yet unknown whether the model with h^{-1} (Fig2 (c)) is equivalent to the original SimSiam (Fig1 (a)) *experimentally*? In other words, is the model with h^{-1} trainable to achieve the comparable performance to SimSiam? To answer this question, we optimize h^{-1} by optimizing the $\boldsymbol{p}_a$ approaching $\boldsymbol{z}_b^*$ while simultaneously optimizing $\boldsymbol{z}_b^*$ to $\boldsymbol{z}_b$ via cosine loss, where $\boldsymbol{z}_b^*$ is the h^{-1} output. The results proves that the model with h^{-1} (Fig2 (c)) is equivalent to SimSiam since it achieves comparable performance as the original SimSiam that directly optimizes $\boldsymbol{p}_a$ approaching $\boldsymbol{z}_b$. This equivalence is another provement that the role h plays in SimSiam is not as EOA. Because theoretically the h^{-1} cannot restore a random augmentation $\mathcal{T}'$ from an expectation $\boldsymbol{p}$, when $\boldsymbol{p} = h(\boldsymbol{z}) = \mathbb{E}_{\mathcal{T}}\Big[\mathcal{F}_{\theta^t}(\mathcal{T}(x))\Big]$.

3 VECTOR DECOMPOSITION FRAMEWORK FOR UNDERSTANDING COLLAPSE

We introduce a gradient analysis framework based on vector decomposition to facilitate their investigation. By default, SimCLR (Chen et al., 2020a) and SimSiam (Chen & He, 2021) both adopt l_2 normalization in their loss for avoiding scale ambiguity. We treat the l_2 normalized vector, *i.e.* $\boldsymbol{Z}$ as the encoder output, which significantly simplifies the gradient derivation and relevant analysis.

3.1 ANALYSIS FRAMEWORK FROM THE VECTOR DECOMPOSITION PERSPECTIVE

Vector decomposition. For the purpose of analysis, we propose to decompose $\boldsymbol{Z}$ into two parts, $\boldsymbol{Z} = \boldsymbol{o} + \boldsymbol{r}$, where $\boldsymbol{o}$, $\boldsymbol{r}$ denote *center vector* and *residual vector* respectively. Specifically, the center vector $\boldsymbol{o}$ is defined as an average of $\boldsymbol{Z}$ for various samples, *i.e.* $\boldsymbol{o}_z = \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{Z}_m$, where M is the number of samples and needs to be sufficiently large. We define the residual vector $\boldsymbol{r}$ as the residual part of $\boldsymbol{Z}$, *i.e.* $\boldsymbol{r}_a = \boldsymbol{Z}_a - \boldsymbol{o}_z$.

Collapse from the vector perspective. A Naive Siamese Model is well expected to cause collapse since the loss is designed to minmize the distance between positive samples and a constant constitutes an optimal yet trivial solution to minimize such loss. When such collapse occurs, $\forall i, \boldsymbol{Z}_i = \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{Z}_m = \boldsymbol{o}_z$, where i denotes a random sample index, which shows the constant vector is $\boldsymbol{o}_z$ when collapse occurs. This interpretation only suggests a possibility that a dominant $\boldsymbol{o}$ can be one of the viable solutions, while the optimization, such as SimSiam, might still lead to a non-collapse solution. In other words, it merely describes $\boldsymbol{o}$ as the *consequence* of the collapse. On the contrary, our work is the first to investigate the *cause* of such collapse through analyzing the influence of individual gradient components.

Competition between $\boldsymbol{o}$ and $\boldsymbol{r}$. Complementary to the Standard Deviation (Std) (Chen & He, 2021), we introduce another indicator m_o as the ratio of $\boldsymbol{o}$ in $\boldsymbol{z}$, *i.e.* $m_o = ||\boldsymbol{o}||/||\boldsymbol{z}||$, where $||.||$ is the L_2 norm. Similarly, we define the ratio of $\boldsymbol{r}$ in $\boldsymbol{z}$, *i.e.* $m_r = ||\boldsymbol{r}||/||\boldsymbol{z}||$. Note that $\boldsymbol{Z}$ is l_2-normalized, thus the trend of m_o and m_r is expected to be opposite of each other. We interpret the cause of collapse as a competition between $\boldsymbol{o}$ and $\boldsymbol{r}$ where $\boldsymbol{o}$ dominates over $\boldsymbol{r}$, *i.e.* $m_o >> m_r$. For Eq 1, the derived negative gradient on $\boldsymbol{Z}_a$ (ignoring $\boldsymbol{Z}_b$ for simplicity due to symmetry) is shown as:

$$\mathcal{G}_{cosine} = -\frac{\partial \mathcal{L}_{MSE}}{\partial \boldsymbol{Z}_a} = \boldsymbol{Z}_b - \boldsymbol{Z}_a \iff -\frac{\partial \mathcal{L}_{cosine}}{\partial \boldsymbol{Z}_a} = \boldsymbol{Z}_b, \tag{4}$$

where the gradient component $\boldsymbol{Z}_a$ is a *dummy* term because the loss $-\boldsymbol{Z}_a * \boldsymbol{Z}_a = -1$ is a constant having zero gradient on the encoder f.

Conjecture1. We conjecture that the gradient component of $\boldsymbol{o}$ is expected to update the encoder to boost the center vector thus increase m_o (or decrease m_r), while the gradient component of $\boldsymbol{r}_a$ is expected to behave in the opposite direction to increase m_r thus decrease m_o. On the other hand, a random gradient component is expected to have relatively small influence on the trend.

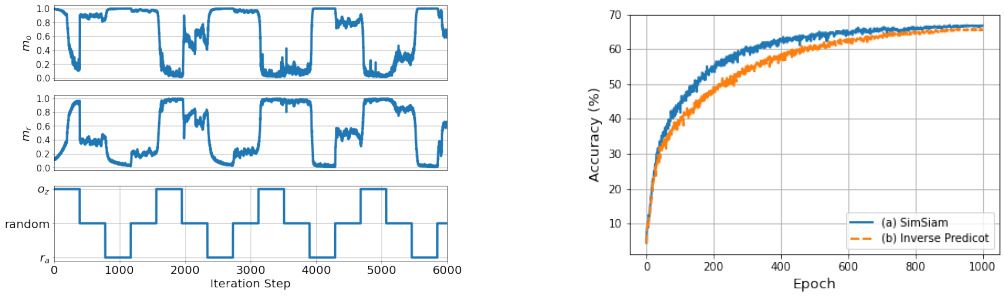
To verify the conjecture, we revisit the *dummy* gradient term $\boldsymbol{Z}_a$. We design loss $-\boldsymbol{Z}_a * \text{sg}(\boldsymbol{o}_z)$ and $-\boldsymbol{Z}_a * \text{sg}(\boldsymbol{Z}_a - \boldsymbol{o}_z)$ to show the influence of gradient component $\boldsymbol{o}$ and $\boldsymbol{r}_a$, respectively. The results in Fig. 3 (a) show that gradient component $\boldsymbol{o}$ and $\boldsymbol{r}_a$ significantly increase m_o and m_r, respectively, confirming our above conjecture. The gradient in Eq 4 can be interpreted as two equivalent forms, from which we choose $\boldsymbol{Z}_b - \boldsymbol{Z}_a = (\boldsymbol{o}_z + \boldsymbol{r}_b) - (\boldsymbol{o}_z + \boldsymbol{r}_a) = \boldsymbol{r}_b - \boldsymbol{r}_a$. Since $\boldsymbol{r}_b$ comes from the same positive sample as r_a, it is expected that $\boldsymbol{r}_b$ also increases m_r, however, this effect is expected to smaller than $\boldsymbol{r}_a$. Thus, as the results in Fig. 3 (a) confirm, the m_r is expected to decrease during the training. To prevent collapse, $\boldsymbol{r}_b$ should be sufficiently (or perfectly) close to $\boldsymbol{r}_a$ by reducing the augmentation gap between the positive sample pairs, however, the encoder relies on the augmentation gap to learn discriminative representation.

Extra gradient component. Overall, our above analysis provides an intuitive interpretation on why a symmetric Siamese architectures, such as Fig. 2 (a) and (b), cannot be trained without collapse. Thus, **Extra** gradient component, denoted as $\boldsymbol{Z}_e$, needs to be introduced to break the symmetry, which is supposed to either decrease m_o or increase m_r for preventing collapse. We decompose the $\boldsymbol{Z}_e$, into center vector and residual vector, denoted as $\boldsymbol{o}_e$ and $\boldsymbol{r}_e$, respectively.

3.2 A TOY EXAMPLE EXPERIMENT WITH NEGATIVE SAMPLE

Which repulsive component helps avoid collapse? Existing works often attribute the collapse in Naive Siamese Model to lacking a *repulsive part* during the optimization. This explanation has motivated previous works to adopt contrastive learning, *i.e.* attracting the positive samples while *repulsing* the negative samples. We experiment with contrastive learning with a simple triplet loss, $\mathcal{L}_{tri} = -\boldsymbol{Z}_a \cdot \text{sg}(\boldsymbol{Z}_b - \boldsymbol{Z}_n)$, [1] where $\boldsymbol{Z}_n$, equal to $\boldsymbol{o}_z + \boldsymbol{r}_n$, indicates the representation vector

[1] Note that the triplet loss here does not have clipping form as in Schroff et al. (2015) for simplicity.



(a) (b)

Figure 3: a) We change the optimization target every 2 epochs (392 iteration steps), and reports the level of m_o and m_r when use different loss in the first and second row. We give the loss status in the last row. Three losses are used in this experiment: o_z corresponding to $-\boldsymbol{Z}_a \cdot \text{sg}(\mathbb{E}(\boldsymbol{Z}_b))$, r_a corresponding to $-\boldsymbol{Z}_a \cdot \text{sg}(\boldsymbol{Z}_b - \mathbb{E}(\boldsymbol{Z}_b)$, and "random" means we generate a random vector as target. b) Line (I) is SimSiam baseline model, (II) inverse predictor h^{-1}.

of a random negative sample. The derived negative gradient is $\boldsymbol{Z}_b + \boldsymbol{Z}_e$, where the extra gradient component $\boldsymbol{Z}_e = -\boldsymbol{Z}_n$ and is decomposed to $\boldsymbol{o}_e$ and $\boldsymbol{r}_e$. The difference between a positive sample and a negative one lies in the component $\boldsymbol{r}$, *thus it might be tempting to guess that* $\boldsymbol{r}_e$ *is the key component of repulsive part that avoids the collapse*. We prove by a toy example experiment that the component $\boldsymbol{o}_e$ instead of $\boldsymbol{r}_e$ inside $\boldsymbol{Z}_e$ is beneficial for preventing collapse.

Method	$\mathcal{L}_{triplet}$	Std	m_o	m_r	Collapse	Accuracy (%)
Baseline	$-\boldsymbol{Z}_a \cdot \text{sg}(\boldsymbol{Z}_b + \boldsymbol{Z}_e)$	0.020	0.026	0.99	$\times$	36.21
Removing $\boldsymbol{r}_e$	$-\boldsymbol{Z}_a \cdot \text{sg}(\boldsymbol{Z}_b + \boldsymbol{o}_e)$	0.02005	0.026	0.99	$\times$	47.41
Removing $\boldsymbol{o}_e$	$-\boldsymbol{Z}_a \cdot \text{sg}(\boldsymbol{Z}_b + \boldsymbol{r}_e)$	0	1	0	$\checkmark$	1

Table 2: Gradient component analysis with a random negative sample.

To explore the individual influence of $\boldsymbol{o}_e$ and $\boldsymbol{r}_e$ in the $\boldsymbol{Z}_e$, we design two experiments by removing one component while keeping the other one. In the first experiment, we remove the $\boldsymbol{r}_e$ in $\boldsymbol{Z}_e$ while keeping the $\boldsymbol{o}_e$. In contrast, the $\boldsymbol{o}_e$ is removed while keeping the $\boldsymbol{r}_e$ in the second experiment. The results are shown in Table 2. In contrast to what existing explanations may expect, we find that the gradient component $\boldsymbol{o}_e$ prevents collapses. Actually, $\boldsymbol{o}_e = -\boldsymbol{o}_z$ in this setup, thus $\boldsymbol{o}_e$ has the de-centering role and prevents collapse. On the contrary, $\boldsymbol{r}_e$ does not prevent collapse and it even decreases the performance when the collapse does not occur. Since the negative sample is randomly chosen, r_e just behaves like a random noise on the optimization.

3.3 Decomposed gradient analysis for SimSiam

It is worth mentioning that we cannot derive the gradient on $\boldsymbol{z}_a$ in SimSiam because the predictor is a nonlinear MLP module. The negative gradient on $\boldsymbol{P}_a$ for $L_{SimSiam}$ in Eq. 2 is derived as:

$$\mathcal{G}_{SimSiam} = -\frac{\partial \mathcal{L}_{SimSiam}}{\partial \boldsymbol{P}_a} = \boldsymbol{Z}_b = \boldsymbol{P}_b + (\boldsymbol{Z}_b - \boldsymbol{P}_b) = \boldsymbol{P}_b + \boldsymbol{Z}_e \tag{5}$$

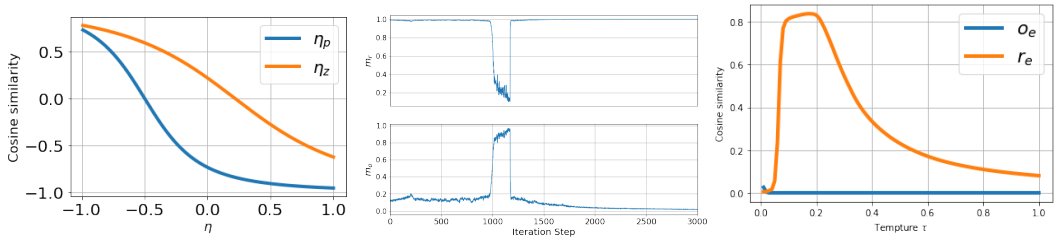
where $\boldsymbol{Z}_e$ indicates the aforementioned **Extra** gradient component. Here, we investigate the influence of its two individual components, *i.e.* $\boldsymbol{o}_e$ or $\boldsymbol{r}_e$, on the collapse. Similar to the analysis with the toy example experiment in Sec. 3.2, we design the experiment by removing one component while keeping the other one.

The results are reported in Table 3. As expected, the model collapses when both components in $\boldsymbol{Z}_e$ are removed and the best performance is achieved when both components are kept. Interestingly, the model does not collapse when either of the two components is kept.

How o_e prevents collapse in SimSiam. Here, $\boldsymbol{o}_z$ and $\boldsymbol{o}_p$ are used to denote the center vector of $\boldsymbol{Z}$ and $\boldsymbol{P}$, respectively, for differentiation. *With the Conjecture1, it is well expected that $\boldsymbol{o}_e$ helps prevent collapse if $\boldsymbol{o}_e$ contains negative $\boldsymbol{o}_p$ since the analyzed vector is $\boldsymbol{P}_a$.* To determine the amount of component of $\boldsymbol{o}_p$ existing in $\boldsymbol{o}_e$, we measure the cosine similarity between $\boldsymbol{o}_e - \eta_p \boldsymbol{o}_p$ and $\boldsymbol{o}_p$ for a wide range of η_p, where $\boldsymbol{o}_e = \boldsymbol{o}_z - \boldsymbol{o}_p$. The results show that when η_p is around -0.5 for their cosine similarity being close to zero, suggesting $\boldsymbol{o}_e$ has $-0.5\boldsymbol{o}_p$. Overall, the $\boldsymbol{o}_e$ indeed contains negative o_p, thus naturally explaining why the gradient component $\boldsymbol{o}_e$ helps prevent collapse. Similarly, we also evaluate the amount of component $\boldsymbol{o}_z$ existing in the negative $\boldsymbol{o}_e$, *i.e.* = $\boldsymbol{o}_p - \boldsymbol{o}_z$, via reporting the similarity between $-\boldsymbol{o}_e - \eta_z \boldsymbol{o}_z$ and $\boldsymbol{o}_z$ and the results show that their cosine similarity is zero when η_z is set to around 0.2. This positive η_z explains why Fig. 2(c) causes collapse. In other words, processing the target vector $\boldsymbol{Z}_b$ with h^{-1}, as in Fig. 2 (c), alleviates collapse ($\eta_p = -0.5$), while processing it with h, as in Fig. 1(c), actually strengthens the collapse ($\eta_z = 0.2$). Note that a negative η has the de-centering role for preventing collapse.

Method	$\boldsymbol{o}_e$	$\boldsymbol{r}_e$	Collapse	Accuracy (%)
Simsiam	✓	✓	×	66.62
	✓	×	×	48.08
	×	✓	×	66.15
	×	×	✓	1

Table 3: Gradient component analysis for SimSiam.



(a) (b) (c)

Figure 4: (a)Investigating the amount of $\boldsymbol{o}_p$ excising in $\boldsymbol{o}_z - \boldsymbol{o}_p$ and the amount of $\boldsymbol{o}_z$ excising in $\boldsymbol{o}_p - \boldsymbol{o}_z$. (b) Normally train the model as SimSiam for 5 epochs, then using collapsing loss for 1 epoch to reduce residual ratio, followed by a correlation regularization loss.

Interesting phenomenon for r_e. In the toy example experiment in Section 3.2, $\boldsymbol{r}_n$ is found to be *not* beneficial for preventing collapse and even decreases the performance. Interestingly, as show in Table 3, we find that $\boldsymbol{r}_e$ alone is sufficient for preventing collapse and even achieves comparable performance as Z_e. We leave relevant explanation to Section 3.4.

3.4 Beyond the center vector to prevent collapse

Motivated from preventing the collapse to a constant, multiple basic works, such as W-MSE (Ermolov et al., 2021), Barlow-twins (Zbontar et al., 2021), DINO (Caron et al., 2021), explicitly adopt de-centering to prevent collapse. They all attribute their reason for not collapsing to de-centering, *i.e.* introducing an o_e that contains negative center vector. The success of this practice and their motivation align well with our conjecture1 as well as as above emperical results. To our knowledge, we are the first to show orthogonal to o_e, the r_e can also prevent collapse.

Predictor in SimSiam encourages dimentional de-correlation. Assuming the h only has a single FC layer to exclude the influence of o_e, the weights in FC are expected to learn the correlation between different dimensions. This interpretation aligns well with the finding that the eigenspace of h weight align well with that of correlation matrix (Tian et al., 2021). In essence, the h is trained to minimize the cosine similarity between $h(\boldsymbol{z}_a)$ and $I(\boldsymbol{z}_b)$, where I is identity mapping. Thus, h is optimized close to I, which is conceptually equivalent to optimize the de-correlation for Z.

Conjecture 2 and motivation. We conjecture that dimentional de-correlation increases m_r for preventing collapse. The motivation is straightforward as follows. The dimensional correlation would be minimum if only a single dimension has a very high value for every individual class and the dimension changes for different classes. In another extreme case, when all the dimensions have

the same values, equivalent to have a single dimension, which already collapses by itself in the sense of losing representation capacity. Conceptually, r_e has no direct influence on the center vector, thus we interpret that r_e prevents collapse through increasing m_r.

To verify the above conjecture, we train SimSiam normally with the loss in Eq 2 and train with several epochs with the loss in Eq 1 for intentionally decreasing the m_r to close to zero. Then, we train the loss with only a correlation regularization term. The results in Fig. 4b this regularization term increases the m_r at a very fast rate. The regularization term is detailed in the Appendix A.3.

Conceptual explanation of de-correlation in Info-NCE. Contrastive learning often requires a large number of negative samples and InfoNCE loss is a default choice for multiple seminal works (Sohn, 2016; Wu et al., 2018; Oord et al., 2018) . The derived negative gradient of $\mathcal{L}_{InfoNCE}$ on $\boldsymbol{Z}_a$ is proportional to:

$$\mathcal{G}_{InfoNCE} = -\frac{\partial \mathcal{L}_{InfoNCE}}{\partial \boldsymbol{Z}_a} = \boldsymbol{Z}_b + \sum_{i=0}^{N} -\lambda_i \boldsymbol{Z}_i \tag{6}$$

where $\lambda_i = \frac{\exp(\boldsymbol{Z}_a \cdot \boldsymbol{Z}_i/\tau)}{\sum_{i=0}^{N} \exp(\boldsymbol{Z}_a \cdot \boldsymbol{Z}_i/\tau)}$, and $\boldsymbol{Z}_0$=$\boldsymbol{Z}_b$ for notation simplicity. The detailed derivation is given in the Appendix A.2. λ_i is large when $\boldsymbol{Z}_i$ is more similar to $\boldsymbol{Z}_a$. The extra gradient component $\boldsymbol{Z}_e = \sum_{i=0}^{N} -\lambda_i \boldsymbol{Z}_i = -\sum_{i=0}^{N} \lambda_i(\boldsymbol{o}_Z + \boldsymbol{r}_i) = -\boldsymbol{o}_z - \sum_{i=0}^{N} \lambda_i \boldsymbol{r}_i$, for which it is straightforward that $\boldsymbol{o}_e = \frac{1}{M}\sum_{m=1}^{M}(\boldsymbol{Z}_e)_m$ whose expectation is $-\boldsymbol{o}_z$, *i.e.* $\boldsymbol{o}_e = -\boldsymbol{o}_z$. Clearly, the gradient component o_e contains negative $\boldsymbol{o}_z$ as de-centering to avoid collapse, and here we investigate the role of $\boldsymbol{r}_e$, derived as $-\sum_{i=0}^{N} \lambda_i \boldsymbol{r}_i = -\sum_{i=0}^{N}(\lambda_i - \frac{1}{N})\boldsymbol{r}_i$, where $\frac{1}{N}\sum_{i=0}^{N} \boldsymbol{r}_i = 0$ by the definition of residual vector. Here, $-\sum_{i=0}^{N} \lambda_i = 1$ and λ_i is proportional to $\boldsymbol{Z}_a \cdot \boldsymbol{Z}_i$ which indicates the cosine similarity between $\boldsymbol{Z}_a$ and $\boldsymbol{Z}_i$. Intuitively, $\lambda_i - \frac{1}{N}$ is positive (negative) when the their cosine similarity is high (low). For conceptual illustration, the sample for $\boldsymbol{z}_a$ has a cat inside it, and the highest in $\boldsymbol{z}_a$ is the m-th dimension. A negative sample for $\boldsymbol{z}_i$ has dog inside it whose highest logit is the n-th logit in $\boldsymbol{z}_i^n$. We assume cat and dog have similar features thus it is expected that $m-th$ and $n-th$ dimensions have a high positive correlation. Since the λ_i has a high value, thus the component $\lambda_i \boldsymbol{r}_i$ has the tendency to decrease the value at $n-th$ dimention in $\boldsymbol{z}_a$, which reduces the correlation between m-th and n-th in $\boldsymbol{z}$ during the training. To verify our interpretation, we measure the cosine similarity between $\boldsymbol{r}_e$ and the gradient on $\boldsymbol{Z}_a$ induced by a correlation regularization loss. The results are shown in Fig. 4c. The gradient similarity is positive for a wide range of temperature value τ. When τ is very high, the λ_i tends to be equal to each other and thus the decorrelation effect vanishes. Specifically, when τ is sufficiently large, $\sum_{i=1}^{N} \lambda_i \boldsymbol{r}_i$ is equivalent to $\sum_{i=1}^{N} \frac{1}{N}\boldsymbol{r}_i$, *i.e.* no de-correlation. Empirically, we find that when τ is around 0.2, $\boldsymbol{r}_e$ achieves high similarity with the gradient induced by the correlation regularization loss.

3.5 Towards a unified understanding of recent progress in SSL

Unifying InfoNCE and SimSiam. At first sight, there is no conceptual similarity between InfoNCE and SimSiam, and this is why the community is intrigued by the success of SimSiam without negative samples. Through decomposing the $\boldsymbol{Z}_e$ into $\boldsymbol{o}_e$ and $\boldsymbol{r}_e$, we find that for both, their $\boldsymbol{o}_e$ plays the role of de-centering and their $\boldsymbol{r}_e$ behaves like de-correlation. In this sense, we bring the two seemingly irrelevant frameworks into a unified perspective in the sense of de-centering and de-correlation.
Beyond InfoNCE and SimSiam. A literature survey on the recent progress, in SSL shows that there is a trend of performing *explicit* manipulation of de-centering and de-correlation. Three representative works are W-MSE (Ermolov et al., 2021), Barlow-twins (Zbontar et al., 2021), DINO (Caron et al., 2021) and that they often achieve performance comparable to those with InfoNCE or SimSiam. Towards a unified understanding of recent progress in SSL, our work is most similar to a concurrent work Bardes et al. (2021). Their work is mainly inspired by Barlow-twins (Zbontar et al., 2021) but decomposes them into three components in a more explicit manner. On the contrary, our work is motivated to answer the question how SimSiam prevents collapse without negative samples. The take-way is also different in the sense that we find that de-correlation itself is sufficient for preventing collapse, while their work claims that variance component (equivalent to de-centering) is an indispensable component for preventing collapse. Overall, through demystifying InfoNCE and SimSiam, our work bridges the understanding gap between those frameworks that implicitly and explicitly exploit de-centering and de-correlation for achieving competitive performance. We believe

that their marginal performance gap can be often ascribed to hyper-parameter tuning effect because they in essence achieve the same de-centering and de-correlation goals.

4 SIMSIAM++: ADDITIONAL INVESTIGATION OF SIMSIAM

Two important details: BN and l_2-normalization. Regarding the reason for SimSiam's success without negative samples, to our knowledge, our work is the first to revisit and refute the explanatory claims in (Chen & He, 2021). Several works, however, have attempted to demystify the success of BYOL Grill et al. (2020), a close variant of SimSiam. The success has been ascribed to BN in (Fetterman & Albrecht, 2020), however, Richemond et al. (2020) refutes the claim and (Chen & He, 2021) also claims that there is no evidence that BN itself is beneficial for preventing collapse. Here, we point out a critical implementation detail that determines whether BN helps prevent collapse. Since the the role of intermediate BNs is ascribed to stabilize training Richemond et al. (2020); Chen & He (2021), we only investigate the final BN in the encoder. Specifically, BN alone, *i.e*. using the Naive Siamese Model as shown in Fig 2, prevents collapse only if it is applied immediately before the MSE loss. For example, the cosine loss adopts l_2-normalization before the MSE loss, for which the BN needs to be applied after the l_2 normalization. The rationale is that, as our conjecture1 suggests, de-centering is the key for preventing collapse. In other words, BN that removes the mean is beneficial for preventing collapse. However after the l_2 normalzation, a new mean is generated. Moreover, we emperically find that unnormalzied MSE loss works poorly for SimSiam, thus our further investigation still adopts the default cosine loss.

Method	Predictor component	Accuracy (%)
SimSiam	MLP	66.9
One bias	Bias	49.82
One FC	Tanh(FC)	64.82
Two FC	FC+FC+Bias	66.7

Table 4: Linear evaluation on CIFAR-100.

SimSiam with Explainable predictor. Our above investigation shows that the predictor has two role of de-centering and de-correlation. Since the original predictor involves a nonlinear MLP, it is hard to understand how the predictor actually achieves it. Towards more explainable predictor, we attempt to train a a bias vector for achieving de-centering and a FC layer to achieve de-correlation. With the current SimSiam encoder, we find that the above goals cannot be achieved. However, attaching a l_2 normalization layer at the end of the encoder, *i.e*. the input of the predictor facilitates the investigation. A predictor with a single bias layer can be trained, where the trainable vector has a very high cosine similarity to the center vector. Theoretically, a FC layer is sufficient to learn the dimensional correlation, in practice, we find that a single FC layer can be difficult to to train. Inspired by Bell-Kligler et al. (2019), we adopt two consecutive FC layers even though two FC layers are mathematically equivalent to a single FC layer. multiple FC layers have the property to make the training more stable. This predictore implementation is equivalent to removing the BN and ReLU in the original predictor. Moreover, to train a single FC layer, we apply a Tanh layer to constrain the FC weight values after every iteration to make the training more stable. The results in Table 4 show that they achieve comparable perfromance as original SimSiam except a bias layer.

5 CONCLUSION

Our work has investigated how SimSiam prevents collapse without negative samples. Our investigation starts with revisiting the explanatory claims in SimSiam and point out a hidden flaw in their reasoning. Afte refuting their claims, our proposes to decompose the representation vector and analyze the decomposed gradients. We find that for both InfoNCE and SimSiam, the center vector gradient has the de-centering effect and the residual gradient vector has the de-correlation effect. Both de-centering and de-correlation effects are found to be beneficial for preventing collapse. We also investigate SimSiam with more explainable predictors.

REFERENCES

Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *arXiv preprint arXiv:1906.00910*, 2019.

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

Sefi Bell-Kligler, Assaf Shocher, and Michal Irani. Blind super-resolution kernel estimation using an internal-gan. *NeurIPS*, 2019.

Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 1993.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *arXiv preprint arXiv:2104.14294*, 2021.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020a.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *CVPR*, 2021.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020b.

Xinlei Chen*, Saining Xie*, and Kaiming He. An empirical study of training self-supervised vision transformers. *arXiv preprint arXiv:2104.02057*, 2021.

Victor G. Turrisi da Costa, Enrico Fini, Moin Nabi, Nicu Sebe, and Elisa Ricci. Solo-learn: A library of self-supervised methods for visual representation learning, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423.

Alaaeldin El-Nouby, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, et al. Xcit: Cross-covariance image transformers. *arXiv preprint arXiv:2106.09681*, 2021.

Aleksandr Ermolov, Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening for self-supervised representation learning. In *ICML*. PMLR, 2021.

Abe Fetterman and Josh Albrecht. Understanding self-supervised and contrastive learning with "bootstrap your own latent" (byol), 2020.
`https://untitled-ai.github.io/understanding-self-supervised-contrastive-learning.html`.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2020.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

Chunyuan Li, Jianwei Yang, Pengchuan Zhang, Mei Gao, Bin Xiao, Xiyang Dai, Lu Yuan, and Jianfeng Gao. Efficient self-supervised vision transformers for representation learning. *arXiv preprint arXiv:2106.09785*, 2021.

Ping Nie, Yuyu Zhang, Xiubo Geng, Arun Ramamurthy, Le Song, and Daxin Jiang. Dc-bert: Decoupling question and document for efficient contextual encoding. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1829–1832, 2020.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.

Pierre H Richemond, Jean-Bastien Grill, Florent Altché, Corentin Tallec, Florian Strub, Andrew Brock, Samuel Smith, Soham De, Razvan Pascanu, Bilal Piot, et al. Byol works even without batch statistics. *arXiv preprint arXiv:2010.10241*, 2020.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815–823, 2015.

Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *NeurIPS*, 2016.

Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. {VL}-{bert}: Pre-training of generic visual-linguistic representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SygXPaEYvH.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019.

Yuandong Tian, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning dynamics without contrastive pairs. *arXiv preprint arXiv:2102.06810*, 2021.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, 2018.

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

A Appendix

A.1 Experimental settings

Unsupervised pre-training: Below are settings for self-supervised pre-training stage. All experiments setting are used as default from the solo-learn library (da Costa et al., 2021).

Data augmentation: We use the series of transformations as follows, notations are in Pytorch style: *RandomResizedCrop* with scale [0.2, 1.0], interpolation is bicubic. Randomly apply with probability of 0.8 for *ColorJitter* (brightness (0.4), contrast (0.4), saturation (0.4), hue (0.1)) with strength accordingly. Random gray scale *RandomGrayscale* apply with $p = 0.2$ Random apply horizontal flip with $p = 0.5$ Normalize image to mean and std as (0.4914, 0.4822, 0.4465) and (0.247, 0.243, 0.261), respectively.

Initialization: All layers of the network are initialized by default of Pytorch.
Network architecture: Network backbone **B** is Resnet-18. Projection MLP head contains three fully-connected (FC) layers follow by Batch Norm (BN) and ReLU. We remove ReLU in the final FC layer: $FC_1 + BN + ReLU + FC_2 + BN + ReLU + FC_3 + BN$. All FC layers have 2048 neurons for both input, output and hidden dimensions. Predictor MLP head includes two FC layers as follows: $FC_1 + BN + ReLU + FC_2$. Input and output of predictor both have dimension of 2048, while hidden dimension is 512.

Optimizer: SGD optimizer is used for unsupervised pre-training. Batch size is 256 and learning rate is linearly scaling by formula $lr \times Bsz/256$ from the base learning rate of 0.5. The schedule for learning rate is cosine decay as Simsiam. Momentum 0.9 and weight decay 1.0×10^{-5} are used for SGD. We use one GPU for each pre-training experiment. The learning rate of predictor is fixed (not cosine decay) same as Simsiam trick. We use a warmup training for first 10-epochs.

Online linear evaluation: The frozen features (2048 dimensions) from the training set are extracted (from the self-supervised pre-trained model) to feed into train a supervised linear classifier (1 FC layer with input 2048 and output is 10 classes). The test is done on the validation set. Learning rate for linear classifier is 0.1. All online evaluation reports are done using the solo-learn library for self-supervised learning (da Costa et al., 2021).

A.2 Derivative of InfoNCE

The derived negative gradient of $\mathcal{L}_{InfoNCE}$ on $\boldsymbol{Z}_a$ is proportional to:

$$\begin{aligned}
\mathcal{G}_{InfoNCE} &= -\frac{\partial \mathcal{L}_{InfoNCE}}{\partial \boldsymbol{Z}_a} \\
&= \boldsymbol{Z}_b + \sum_{i=0}^{N} -\lambda_i \boldsymbol{Z}_i \\
&= \frac{1}{\tau}(\boldsymbol{Z}_b - \sum_{i=0}^{N} \lambda_i (\boldsymbol{o}_Z + \boldsymbol{r}_i)) \\
&= \frac{1}{\tau}(\boldsymbol{Z}_b + (-\boldsymbol{o} - \sum_{i=0}^{N} \lambda_i \boldsymbol{r}_i))
\end{aligned} \tag{7}$$

where $\lambda_i = \frac{\exp(\boldsymbol{Z}_a \cdot \boldsymbol{Z}_i/\tau)}{\sum_{i=0}^{N} \exp(\boldsymbol{Z}_a \cdot \boldsymbol{Z}_i/\tau)}$, and $\boldsymbol{Z}_0$=$\boldsymbol{Z}_b$ for notation simplicity. Here, λ_i is proportional to $\boldsymbol{Z}_a \cdot \boldsymbol{Z}_i$ which indicates the cosine similarity between $\boldsymbol{Z}_a$ and $\boldsymbol{Z}_i$. Thus, λ_i is large when $\boldsymbol{Z}_i$ is more similar to $\boldsymbol{Z}_a$.

A.3 Regularization loss in Conjecture 2

We compute auto-correlation matrix of encoder output along the mini-batch. Recently, Dimensional Correlation in SSL has attached great attention to prevent collapse(Zbontar et al., 2021; Ermolov et al., 2021). Following the (Zbontar et al., 2021), we define $\mathcal{C}$ as the dimensional correlation matrix computed among the representations in a batch. Suppose the dimension of representation vector of a single sample Z is d, $\mathcal{C}$ is matrix with the dimension of $d \times d$:

$$\mathcal{C}_{ij} = \frac{\sum_b Z^A_{b,i} Z^B_{b,j}}{\sqrt{\sum_b (Z^A_{b,i})^2} \sqrt{\sum_b (Z^B_{b,j})^2}} \tag{8}$$

where b indicates the sample index in a batch and i, j indicates dimension indexes in Z. Here we use $Z^A_{b,i} = Z^B_{b,j} = Z_1$. The correlation loss calculation in pytorh style as follows:

```
corr = torch.einsum("bi, bj -> ij", Z1, Z1) / N
diag = torch.eye(D, device=corr.device)
cdif = (corr - diag).pow(2)
loss = scale_loss * cdif.sum()
```