# SIMMER: SIMPLE MAXIMIZATION OF ENTROPY AND RANK FOR SELF-SUPERVISED REPRESENTATION LEARNING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Consistency regularization, referring to enforcing consistency across a model's responses to different views of the same input, is widely used for self-supervised image representation learning. However, consistency regularization can be trivially achieved by collapsing the model into a constant mapping. To prevent this, existing methods often use negative pairs (contrastive learning) or ad hoc architecture constructs. Inspired by SimSiam's alternating optimization hypothesis (Chen & He, 2020), we propose a novel optimization target, SimMER, for self-supervised learning that explicitly avoids model collapse by balancing consistency (total variance minimization) and entropy of inputs' representations (entropy maximization). Combining consistency regularization with entropy maximization alone, the method can achieve performance on par with the state-of-the-art. Furthermore, we introduce an linear independence loss to further increase the performance by removing linear dependency along the feature dimension of the batch representation matrix (rank maximization), which has both anticollapsing and redundancy removal effects. With both entropy and rank maximization, our method surpasses the state-of-the-art on CIFAR-10 and Mini-ImageNet under the standard linear evaluation protocol.

## 1 INTRODUCTION

Recently, significant progress has been made in self-supervised representation learning as self-supervised models could achieve or even surpass the performance of fully supervised methods on many downstream tasks including image classification. Contrastive methods (Chen et al., 2020a;b; He et al., 2020; Chen et al., 2020c) achieved high performance by forcing negative pairs to have different representations while positive pairs to share the same feature representations. This approach is effective, but it relies on a large number of negative pairs which can be achieved using a large batch size (Chen et al., 2020a) or a large memory bank (He et al., 2020). Recent non-contrastive approaches such as Grill et al. (2020) and Chen & He (2020) learn representations by focusing on the consistency among different perturbed views of the same input. These methods are generally less memory and computation-heavy since they do not explicitly utilize negative pairs. One major problem in consistency-only approaches is the need to prevent trivial solutions where all inputs are mapped to the same vector. To prevent such model collapse, consistency regularization methods usually have an asymmetric architecture, additional layers (e.g., the predictor in Chen & He (2020)), or stop gradient operations.

To overcome the above issues in contrastive and consistency methods, we introduce SimMER, a simple yet effective consistency based self-supervised representation learning algorithm. As shown in figure 1, SimMER is a symetric method using a shared-weight neural network across all augmentation heads. The encoder network consists of a CNN backbone and an Multi Layer Perceptron (MLP) projection head (Chen et al., 2020a). SimMER is optimized through three newly proposed loss terms: total variance, rank, and entropy loss. The total variance loss ensures that the model learns a consistent representation of the same inputs under different perturbations. The entropy loss is introduced to prevent degenerate solutions where all inputs are mapped to the same representation. The rank loss is used to remove the linear dependence in the feature dimensions and forces the model to have efficient feature representations. Additionally, it also prevents the model from

collapsing. Recently, an arXiv paper, VICReg (Bardes et al., 2021) has also proposed a method that prevents model collapse without negative pairs and asymmetric architecture; however, our design and loss functions are different.

Due to the symmetric structure of SimMER over all augmentation heads, our algorithm naturally supports multiple views of the same input. Therefore, SimMER can be considered as a generalization of the Siamese architecture used in current consistency-based methods. Since SimMER learns the representations via direct optimization of the three loss terms over the $K$ views, it does not require negative pairs, large batch size, asymmetric architecture, or ad hoc architectural designs.

Our contributions are summarized as in the following. (1) We introduce a novel self-supervised representation learning algorithm, SimMER, which uses an interpretable balance between consistency regularization, entropy, and linear independence to remove the necessity of negative pairs and architecture constructs in previous self-supervised learning methods. (2) We introduce a total variance minimization loss that maintains a consistent representation of the same input under different perturbations and generalizes the previous 2-head consistency. (3) We introduce an entropy maximization loss which prevents a collapsed constant solution by maximizing the nearest neighbor distances in a batch. (4) We introduce a family of rank losses that remove redundancy in the feature space in addition to preventing model collapse by preferring linear independence in the column space of the batch representation matrix. (5) We conducted extensive experiments on CIFAR-10 and Mini-ImageNet where SimMER achieved state-of-the-art performance.

## 2 APPROACH

Within the self-supervised representation learning framework, we aim to learn useful representations of images for downstream tasks like classification by only using unlabeled images drawn from a distribution $X$. In our method (figure 1), an encoder network, $\mathcal{F}_\theta$, consisting of a standard CNN backbone attached with a projection MLP (Chen et al., 2020a), is used to encode a batch of $B$ images $(x_b)_{b=1}^B$. The projection MLP projects the output feature map of the backbone to a vector in $\mathbb{R}^D$. For each image $x_b$, we use the stochastic augmentation function $\mathcal{T}$ (e.g. `ColorJittering`, `RandomResizedCrop`) to generate $K$ views $(x_{b,k})_{k=1}^K$. Each view $x_{b,k}$ is then encoded by $\mathcal{F}_\theta$ to get the representation $z_{b,k}$. Then, an averaged representation $\bar{z}_b = \frac{1}{B}\sum_{b=1}^B z_{b,k}$ is calculated as the true representation of the image $x_b$. Inspired by SimSiam's alternating optimization hypothesis (Chen & He, 2020), we use a total variance term to encourage the consistency across different views $(x_{b,k})_{k=1}^K$ of the same image $x_b$. To avoid trivial solution of $\mathcal{F}_\theta$ like a constant function, we add an entropy maximization term to encourage the representations to contain more information. Combining total variance minimization with entropy maximization alone, the method can achieve performance on par with the state-of-the-art (see section 3.5.1). To further increase the quality of the representations, we added a rank (linear independence) maximization term to encourage all feature dimensions of the representation to be linearly independent to remove redundancy. In the following sections, we introduce the total variance minimization, entropy maximization, and rank (linear independence) maximization terms in detail.

### 2.1 TOTAL VARIANCE MINIMIZATION

Among the recent advances of self-supervised learning algorithms (Chen & He, 2020; Grill et al., 2020; Caron et al., 2020), SimSiam by Chen & He (2020) proposes a Siamese architecture that uses consistency regularization on representations encoded by encoder network $\mathcal{F}_\theta$ to learn high quality representations without negative pairs. The key to SimSiam's success is the stop-gradient operation on one head of Siamese architecture which prevents the gradient propagation through that head. Without this operation, SimSiam can easily learn a constant mapping and the quality of the representations is degenerated (Chen & He, 2020). However, the presence of a stop-gradient operation in the SimSiam loss makes the method less interpretable. Although on the surface, SimSiam only optimizes the network parameter $\theta$, the authors theorize that SimSiam is actually performing an implicit alternating optimization (Expectation-Maximization like) algorithm on two parameters $\theta$ and $\eta$. More concretely, the target of the underlying alternating optimization is

$$\min_{\theta,\eta} \mathop{\mathbb{E}}_{\substack{\mathcal{T} \\ x\sim X}} \left[ \|\mathcal{F}_\theta(\mathcal{T}(x)) - \eta_x\|_2^2 \right], \tag{1}$$

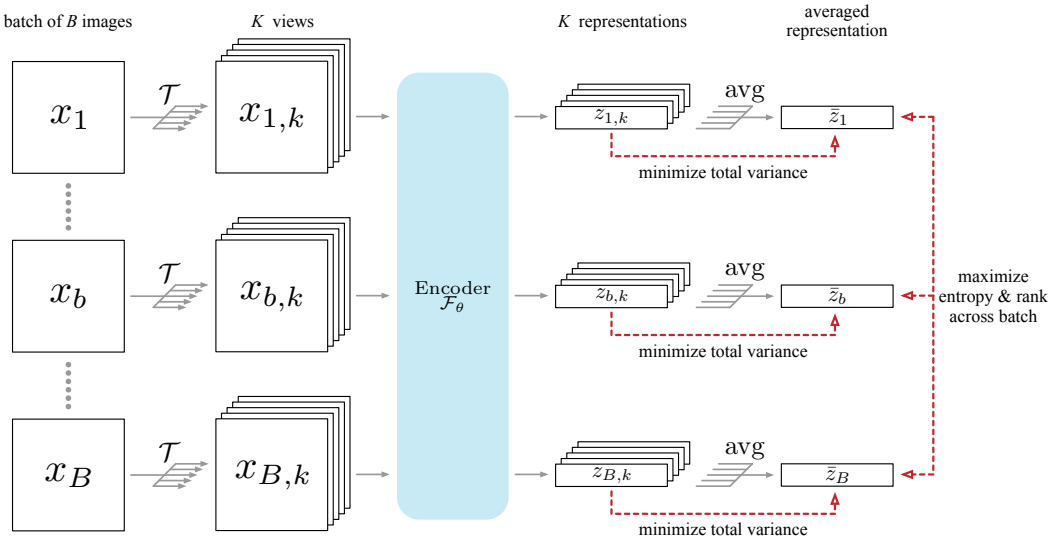


Figure 1: **SimMER architecture.** For a batch of $B$ images $(x_b)_{b=1}^B$, each image $x_b$ is augmented $K$ times through the stochastic augmentation function $\mathcal{T}$ to generate $K$ views $(x_{b,k})_{k=1}^K$. All views are encoded by the encoder network $\mathcal{F}_\theta$ (backbone plus projection MLP) to generate $K$ representations $(z_{b,k})_{k=1}^K$. To encourage consistency, we minimize the total variance across all representations of a single image. To encourage diversity and decoupling of the feature dimension, we maximize the entropy and rank (linear independence) across the averaged representations of a batch of images.

where $\eta$ is the extra set of parameter whose size is proportional to the number of images, and $\eta_x$ refers to using the image index of $x$ to access a sub-vector of $\eta$. The stop-gradient operation appears as a consequence of introducing this extra parameter $\eta$. They further theorize that during the optimization process, $\eta_x$ is assigned the average representation of $x$ over the distribution of augmentation, meaning $\eta_x = \mathbb{E}_{\mathcal{T}}[\mathcal{F}_\theta(\mathcal{T}(x))]$. Thus, equation 1 is similar to the trace of the covariance matrix of $\mathcal{F}_\theta(\mathcal{T}(x))$, or the total variance of $\mathcal{F}_\theta(\mathcal{T}(x))$. Inspired by this, we define our total variance term for consistency regularization as

$$\mathcal{L}_{TV}(\theta) = \mathop{\mathbb{E}}_{\substack{\mathcal{T} \\ x \sim X}} \left[ \left\| \mathcal{F}_\theta(\mathcal{T}(x)) - \mathop{\mathbb{E}}_{\mathcal{T}}[\mathcal{F}_\theta \mathcal{T}(x)] \right\|_2^2 \right]. \tag{2}$$

In practice, we use $K$ different views of $x$ to estimate $\mathbb{E}_{\mathcal{T}}[\mathcal{F}_\theta \mathcal{T}(x)]$. Therefore, in a batch of $B$ images, $\mathcal{L}_{TV}$ is calculated as $\frac{1}{B}\sum_{b=1}^B \|z_{b,k} - \bar{z_b}\|_2^2$, where $z_{b,k} = \mathcal{F}_\theta(\mathcal{T}(x_b))$ is the representation of each view, and $\bar{z_b} = \frac{1}{K}\sum_{k=1}^K z_{b,k}$ is the averaged representation of each image. This conceptually forms a completely symmetric $K$-head architecture and can be viewed as a generalization of the existing two-head Siamese architecture.

### 2.2 Entropy Maximization

Minimizing the total variance term directly will lead to inevitable collapse of the model as it will quickly degenerate into a constant mapping (see section 3.5.1). In prior works, negative pairs or ad hoc architecture constructs (e.g., predictor head, stop-gradient, momentum encoder) are used to prevent such model collapse. However, a large batch size is required to sample enough negative pairs and architecture constructs often lack interpretability. Inspired by information theory, we introduce an entropy maximization term, $\mathcal{L}_H$, in the loss to prevent model collapse.

We can view the representation encoded by the encoder network as a random variable $Z$, where $Z = \mathcal{F}_\theta(\mathcal{T}(X))$. Since $Z$ is continuous, the classic definition of discrete entropy does not apply. To measure the entropy of $Z$, we need to use the notation of differential entropy. Differential entropy is also related to the shortest description length and is similar in many ways to the entropy of a discrete

random variable (Cover & Thomas, 2006). More concretely, the differential entropy of $Z$ is defined as

$$h(Z) = -\int f(z) \log f(z) dz, \tag{3}$$

where $f$ is the probability density function of $Z$. Since access to the probability density function of $Z$ requires access to the probability density function of $X$, which is not feasible. Instead, we estimate the differential entropy from the samples directly. In particular, we find the classic entropy estimator based on nearest neighbor distances by Kozachenko & Leonenko (1987) performed well in practice and offers additional insights to our method. Given a batch of $B$ random representation vectors $(z_b)_{i=1}^B$, the nearest neighbor estimate is defined as

$$h(Z) \approx \hat{h}\left((z_b)_{i=1}^B\right) = \frac{1}{B}\sum_{b=1}^{B} \log\left(B \min_{i\in[B], i\neq b} \|z_b - z_i\|_2\right) + \ln 2 + C_E, \tag{4}$$

where $C_E = -\int_0^\infty e^{-t}\log t dt$ is the Euler constant (Beirlant et al.). Under some mild conditions, Kozachenko & Leonenko (1987) prove the mean square consistency of $\hat{h}$ (Beirlant et al.). Removing the constant terms from $\hat{h}$, we obtain the entropy maximization loss of SimMER:

$$\mathcal{L}_H(\theta) = -\frac{1}{B}\sum_{b=1}^{B} \log\left(\min_{i\in[B], i\neq b} \|\bar{z}_b - \bar{z}_i\|_2\right). \tag{5}$$

We keep the $\log$ term to increase numerical stability during optimization. Intuitively, this loss maximizes the distance between any two nearest representations in the batch, encouraging all representations to evenly spread out in the feature space. Additionally, we can also view the nearest neighbor as a negative sample, comparing with prior contrastive learning methods (He et al., 2020; Chen et al., 2020a) which consider every other sample as the negative sample. Noticeably, with $\mathcal{L}_{TV}$ and $\mathcal{L}_H$, SimMER can already achieve linear evaluation performance on par with state of the art (see section 3.5.1).

### 2.3 Rank (Linear Independence) Maximization

We now introduce a rank maximization term, $\mathcal{L}_R$, to increase the linear independence on the representation dimension, which reduces redundancy in the learned representations and also prevents model collapse. To be more specific, given a batch of $B$ representation $(z_b)_{b=1}^B$ and $z_b \in \mathbb{R}^D$, we want to maximize $\mathrm{rank}([z_b^\intercal]_{b=1}^B)$, where $[z_b^\intercal]_{b=1}^B$ is a matrix of shape $B \times D$ whose $b$-th row is $z_b^\intercal$. To illustrate the effect of maximizing $\mathrm{rank}([z_b^\intercal]_{b=1}^B)$, we use the following toy example. Suppose that the encoder network $\mathcal{F}_\theta$ outputs vectors of the form $z = (z_{(1)}, z_{(2)}, z_{(3)}) \in \mathbb{R}^3$. If it happens that for all the $z$ outputted by $\mathcal{F}_\theta$, $z_{(3)} = z_{(1)} + z_{(2)}$, then information encoded in $z_{(3)}$ is redundant. For example, performing linear classification on a truncated vector $(z_{(1)}, z_{(2)})$ will generate the same result with the original vector $z = (z_{(1)}, z_{(2)}, z_{(3)})$ (notice, the actual linear evaluation is not performed on $z$ directly as we remove the MLP projection head after self-supervised training). Additionally, increasing linear independence can also be viewed as strengthening the effect of entropy maximization term by preventing the representation from concentrating into in a linear subspace of the $\mathbb{R}^d$. If we maximize $\mathcal{L}_H$ term alone, it is theoretically possible that our learned representations are only spread out in a linear subspace of $\mathbb{R}^d$.

Unfortunately, for general matrices, calculating the matrix rank is an NP-hard problem, and the rank cannot be maximized directly. However, multiple values related to the singular values of the matrix can be used to measure the level of linear independence and can be optimized easily with gradient methods. To introduce the four variants of rank maximization loss, we use the following standard notation: For a rectangular matrix $A \in \mathbb{R}^{m\times n}$, we use $\sigma_i(A)$ to denote the $i$-th largest singular value of $A$ with $1 \leq i \leq \min(m, n)$. As a shorthand, we use $r$ to denote $\mathrm{rank}(A)$. Notice for $i > r$, we have $\sigma_i(A) = 0$.

**Total Least Squares Minimization.** The total least squares minimization seeks to find a vector $u \in \mathbb{R}^n$ on the unit ball that minimizes the 2-norm of vector $Au$. The value of $\|Au\|_2$ can be used to measure the level of linear independence of the columns of $A$. For example, if there are linearly dependent columns in $A$, the value of $\|Au\|_2$ will become 0. A higher value of the total least squares

minimum indicates that any linear combination of the columns of $A$ is still far from the zero vector, thus more linearly independent. Formally, we define the total least squares minimum of a matrix $A$ as

$$\text{tlsm}(A) = \min_{u} \quad \|Au\|_2 \tag{6}$$

$$\text{subj. to} \quad \|u\|_2 = 1. \tag{7}$$

It can be shown that the total least square value of $A$ is the smallest singular value, i.e., $\text{tlsm}(A) = \sigma_{\min(m,n)}(A)$. Here, we define the first variant of $\mathcal{L}_R$ as

$$\mathcal{L}_{R1}(\theta) = -\text{tlsm}([z_b^\intercal]_{b=1}^B). \tag{8}$$

**Nuclear Norm.** The nuclear norm of $A$, denoted as $\|A\|_*$, is equal to the sum of all its singular values, i.e., $\|A\|_* = \sum_{i=1}^r \sigma_i(A)$. Fazel Sarjoui (2002) shows that on the unit ball, the convex envelope of the rank of a matrix is the nuclear norm of a matrix. Here, we define the second variant of $\mathcal{L}_R$ as

$$\mathcal{L}_{R2}(\theta) = -\|[z_b^\intercal]_{b=1}^B\|_*. \tag{9}$$

**Normalized Nuclear Norm.** We can directly bound the rank with the nuclear norm by normalizing it with the induced 2-norm or Frobenius norm. The induced 2-norm of $A$, denoted as $\|A\|_2$, is defined as

$$\|A\|_2 = \max_{u} \quad \|Au\|_2 \tag{10}$$

$$\text{subj. to} \quad \|u\|_2 = 1. \tag{11}$$

It is well known that the induced 2-norm is equal to the largest singular value, i.e., $\|A\|_2 = \sigma_1(A)$. Moreover, the Frobenius norm, denoted as $\|A\|_F$, is equal to the $l^2$ norm of the singular value vector, i.e., $\|A\|_F = (\sum_{i=1}^r \sigma_i(A))^{\frac{1}{2}}$. We can obtain the following inequality with Cauchy–Schwarz inequality:

$$\|A\|_* \leq \sqrt{r}\|A\|_F \leq r\|A\|_2. \tag{12}$$

Therefore, we have $r \geq \frac{\|A\|_*}{\|A\|_2}$ and $r \geq \left(\frac{\|A\|_*}{\|A\|_F}\right)^2$. We can use these to define our third and fourth variants of $\mathcal{L}_R$:

$$\mathcal{L}_{R3}(\theta) = -\frac{1}{\min(B,D)} \frac{\|[z_b^\intercal]_{b=1}^B\|_*}{\|[z_b^\intercal]_{b=1}^B\|_2} \tag{13}$$

$$\mathcal{L}_{R4}(\theta) = -\frac{1}{\min(B,D)} \left(\frac{\|[z_b^\intercal]_{b=1}^B\|_*}{\|[z_b^\intercal]_{b=1}^B\|_F}\right)^2. \tag{14}$$

The term $\frac{1}{\min(B,D)}$ is added to normalize the loss to $[0, 1]$, as $\min(B, D)$ is the maximum possible rank. In practice, we find that all four variants can further improve the performance of our method. $\mathcal{L}_{R1}$, $\mathcal{L}_{R3}$, and $\mathcal{L}_{R4}$ have a theoretical lead as the total least squares minimum measures the level of linear independence directly, and the normalized nuclear norm provides a lower bound of the rank. However, we choose to use the nuclear norm variant $\mathcal{L}_{R2}$ (equation 9) as we observe it works better in practice. Additionally, a bias column can be horizontally concatenated to $[z_b^\intercal]_{b=1}^B$ to ensure that the learned representation does not collapse into an affine subspace instead of a linear subspace. More specifically, we can maximize $\mathcal{L}_R$ on $[z_b^\intercal, 1]_{b=1}^B$ instead of $[z_b^\intercal]_{b=1}^B$, where the $b$-th row of $[z_b^\intercal, 1]_{b=1}^B$ is $z_b^\intercal$ concatenated with 1. A full ablation study on these variants is in section 3.5.2.

### 2.4 SIMMER

With all three components, $\mathcal{L}_{TV}, \mathcal{L}_H, \mathcal{L}_R$, we now introduce our complete loss function:

$$\mathcal{L}_{\text{SimMER}}(\theta) = \mathcal{L}_{TV}(\theta) + \alpha_H \mathcal{L}_H(\theta) + \alpha_R \mathcal{L}_R(\theta). \tag{15}$$

SimMER loss imposes consistency across different views of the same image by minimizing $\mathcal{L}_{TV}$. Minimizing $\mathcal{L}_H$ and $\mathcal{L}_R$ raises the level of entropy and linear independence to prevent the output representations from collapsing to a single point or a linear subspace of $\mathbb{R}^D$. Lastly, $\mathcal{L}_R$ also removes the redundancy in the representations by enforcing feature dimensions to be not linearly dependent.

We use two hyperparameters $\alpha_H$ and $\alpha_R$ to control the balance between the three terms. The pseudo-code of calculating $\mathcal{L}_{\text{SimMER}}$ is provided in algorithm 1.

**Algorithm 1:** SimMER Loss

**input :** a batch of $B$ unlabeled images $(x_i)_{i=1}^B$, number of views $K$, weight for the entropy maximization loss $\alpha_H$, weight for the rank maximization loss $\alpha_R$

**for** $b \leftarrow 1$ **to** $B$ **do**
  **for** $k \leftarrow 1$ **to** $K$ **do**
    $z_{b,k} = \mathcal{F}_\theta(\mathcal{T}(x_b))$ ▷get representations of $K$ views of $B$ images
  $\bar{z}_b = \frac{1}{K}\sum_{k=1}^K z_{b,k}$ ▷get the averaged representation
$\mathcal{L}_{TV} = \frac{1}{B}\sum_{b=1}^B \|z_{b,k} - \bar{z}_b\|_2^2$
$\mathcal{L}_H = -\frac{1}{B}\sum_{b=1}^B \log\left(\min_{i\in[B],i\neq b} \|\bar{z}_b - \bar{z}_i\|_2\right)$ ▷$[\bar{z}_b{}^\intercal]_{b=1}^B$ denotes a matrix whose $b$-th rows is $\bar{z}_b{}^\intercal$
$\mathcal{L}_R = -\|[\bar{z}_b{}^\intercal]_{b=1}^B\|_*$

**output:** $\mathcal{L}_{TV} + \alpha_H\mathcal{L}_H + \alpha_R\mathcal{L}_R$

# 3 Experiments

## 3.1 Implementation Details

We use ResNet-18 and ResNet-50 (He et al., 2016) as our backbone. Our projection MLP follows the implementation of SimSiam (Chen & He, 2020). During self-supervised training, the projection MLP is attached to the output of the final pooling layer of the backbone. Unless specified otherwise, the following are the default configuration used in all experiments, which are mainly derived from SimSiam (Chen & He, 2020). We use Adam (Kingma & Ba, 2015) with weight decay as the optimizer ($base\_lr = 0.001, weight\_decay = 1 \times 10^{-6}$) to train on the full training set for 800 epochs with batch size $B = 512$ (Chen & He, 2020). For hyperparameters, we choose $\alpha_H = 0.1, \alpha_R = 2 \times 10^{-4}, K = 4$. We also follow Goyal et al. (2018)'s linear learning rate scaling scheme, i.e., $lr = base\_lr \times B/256$. During self-supervised training, linear evaluation, and semi-supervised finetuning stages, we use a cosine learning rate scheduler to decay the learning rate (Loshchilov & Hutter, 2016). To be consistent with SimSiam, we use a 3 layer MLP on ResNet-50 and 2 layer MLP on ResNet-18. We use the same type of stochastic data augmentation as that of SimSiam. We use linear evaluation to evaluate the quality of the representation learned by SimMER. Additionally, we use a partially labeled dataset to finetune the model to draw comparisons with semi-supervised methods. Unlike many baselines (Chen & He, 2020; Caron et al., 2020; Grill et al., 2020), we do not apply any normalization to our representations.

## 3.2 Datasets

**CIFAR-10.** CIFAR-10 is a commonly used dataset that contains 60K images of size $32 \times 32$. The training set and the test set each contain 50K and 10K images uniformly distributed in 10 classes. We use the full training set for self-supervised training.

**Mini-ImageNet.** Mini-ImageNet has a total of 60K images of shape $84 \times 84$ with 100 classes (600 images per class). Mini-ImageNet was originally proposed by Vinyals et al. (2016). As a resized subset of the full ILSVRC2012 ImageNet dataset (Russakovsky et al., 2015), it has a high level of complexity while training on Mini-ImageNet requires less computation power than the full ImageNet. Although it is originally designed for meta-learning, recently, there is a rising trend for using this ImageNet subset as a benchmark in the weakly-supervised learning literature (Hu et al., 2021; Kuo et al., 2020; Iscen et al., 2019) due to its high complexity. We follow the protocol of Iscen et al. (2019), in which 500 images are selected from each class to form the training set, and the remaining 100 images of each class are used for testing. We use the full training set for self-supervised training.

## 3.3 Linear Evaluation

We closely follow SimSiam's linear evaluation protocol. During linear evaluation, we freeze the backbone and replace the MLP projection head with a linear classifier. We train the linear classifier

alone on the full training set for 90 epochs with LARS (You et al., 2017) optimizer. The batch size is 4096 and we linearly scale the base learning rate to be $lr = 0.1 \times 4096/256$. The weight decay is set to be 0. The final test accuracy is calculated on the entire test set. We use SimSiam (Chen & He, 2020), SimCLR (Chen et al., 2020a), MoCo (He et al., 2020), MoCo v2 (Chen et al., 2020c), SwAV (Caron et al., 2020), and BYOL (Grill et al., 2020) as our baselines. We are unable to provide a comparison with VICReg (Bardes et al., 2021) due to unavailability of the code.

**CIFAR-10.** Following SimSiam and MoCo, we use ResNet-18 CIFAR variant for all CIFAR-10 experiments. It uses a smaller kernel on the first convolution layer and turns off the first max-pooling layer. Table 1 shows that our method outperforms all baselines. Both SimSiam and SimCLR scores are reported in the SimSiam paper (Chen & He, 2020), and the MoCo score is reported in MoCo's official code repository (He et al., 2021). We use 10 views in this experiment. Since we believe the model is saturated on CIFAR-10, any gain of accuracy on CIFAR-10 is difficult.

Table 1: CIFAR-10 top-1 linear evaluation test accuracy. All experiments use ResNet-18. † reported in SimSiam. * reported in MoCo's original code repository.

| Method | Accuracy (%) | Negative Pair | Stop-gradient | Predictor | Memory Bank | Momentum Encoder |
|---|---|---|---|---|---|---|
| MoCo* | 90.70 | ✓ | | | ✓ | ✓ |
| SimCLR† | 91.10 | ✓ | | | | |
| SimSiam† | 91.80 | | ✓ | ✓ | | |
| **SimMER** | **92.61** | | | | | |

**Mini-ImageNet.** We used ResNet-50 for all Mini-ImageNet experiments. Since the baseline methods do not have Mini-ImageNet results, we used the official implementation of MoCo, MoCo v2, SwAV, and SimSiam to conduct the experiments (He et al., 2020; Chen & He, 2020; Caron et al., 2020). A popular deep learning research library, PyTorch Lightning (Falcon & The PyTorch Lightning team, 2019), provides the implementation for BYOL (Grill et al., 2020). We also used the default ResNet-50 hyperparameters provided in each baseline method to obtain the results. As shown in Table 2, SimMER significantly improves all baseline methods on Mini-ImageNet.

Table 2: Mini-ImageNet top-1 linear evaluation test accuracy. All experiments use ResNet-50. * use original implementation. † uses implementation provided by Pytorch-Lightning.

| Method | Accuracy (%) | Negative Pair | Stop-gradient | Predictor | Momentum Encoder | Memory Bank |
|---|---|---|---|---|---|---|
| MoCo v2* | 55.30 | ✓ | | | ✓ | ✓ |
| SwAV* | 42.70 | | ✓ | | | |
| BYOL† | 59.46 | | ✓ | ✓ | ✓ | |
| SimSiam* | 58.14 | | ✓ | ✓ | | |
| **SimMER** | **62.62** | | | | | |

### 3.4 Semi-supervised Experiments

To compare with semi-supervised learning methods, we finetune our self-supervised pretrained model on a fraction of the labeled train set and evaluate the classification accuracy on the test set. In particular, after replacing the MLP projection head with a linear classifier, we train the backbone and linear classifier together with separate learning rates. The base learning rate for the classifier is $0.1$, and the backbone's base learning rate is $0.001$. Linear learning rate scaling still applies. We use SGD with momentum $(0.9)$ and no weight decay as the optimizer. We turn off the first max-pooling layer of the backbone. We use MeanTeacher (Tarvainen & Valpola, 2017), Label Propagation (Iscen et al., 2019), SimPLE (Hu et al., 2021), FeatMatch (Kuo et al., 2020) as our baselines. We finetune our model with a batch size of $512$ on the exact 40-shot train split (40 images for each class, 4000 images in total) used by Hu et al. (2021) for $40$ epochs. Results are in table 3. Although SimMER is not specifically designed for the semi-supervised learning task, it achieves accuracy on par with state-of-the-art methods.

Table 3: Mini-ImageNet 40 shot semi-supervised top-1 test accuracy. All experiments use ResNet-18. * are reported in Label Propagation. † are reported in their own original publications.

| **Method** | MeanTeacher* | Label Propagation* | SimPLE† | FeatMatch† | **SimMER** |
|---|---|---|---|---|---|
| **Accuracy (%)** | 27.49 | 29.71 | 49.39 | **60.95** | 60.94 |

### 3.5 Ablation Study

We conducted our ablation study with the ResNet-18 CIFAR variant (Chen & He, 2020; He et al., 2020) on CIFAR-10. The quality of the learned representation is evaluated by the linear evaluation protocol introduced in Section 3.3.

#### 3.5.1 Effect of Hyperparameters

We conducted an extensive study on the effect of hyperparameters. The results are available in Table 4. We found that the method is not sensitive to changes in hyperparameters. Notice, when $\alpha_H = 0$, the rank maximization loss $\mathcal{L}_R$ alone can stop the model from collapsing though suffering a loss of performance. With $\alpha_R = 0$, the entropy maximization loss, $\mathcal{L}_H$, alone can achieve performance on par with state-of-the-art methods. The performance gain of enabling $\mathcal{L}_R$ is observable but not significant. Increasing the number of views, $K$, generally increases the performance. Our method works on a small batch size of $B = 64$ or $B = 128$, showing the calculation of $\mathcal{L}_H$ and $\mathcal{L}_R$ is not heavily dependent on a large batch size. Lastly, when both $\alpha_H = 0$ and $\alpha_R = 0$, the model collapses as we expected.

Table 4: Ablation study on CIFAR-10. All experiments use ResNet-18. We study the effect of $\alpha_H$, $\alpha_R$, number of views $K$, and batch size $B$. The first row serves as the basis for comparison, and shaded areas indicate changes of hyperparameters.

| $\alpha_H$ | $\alpha_R$ | **Num. Views** ($K$) | **Batch Size** ($B$) | **Accuracy (%)** |
|---|---|---|---|---|
| 0.1 | $2 \times 10^{-4}$ | 4 | 512 | 92.44 |
| 0 | $2 \times 10^{-4}$ | 4 | 512 | 82.50 |
| 1 | $2 \times 10^{-4}$ | 4 | 512 | 91.20 |
| 10 | $2 \times 10^{-4}$ | 4 | 512 | 90.01 |
| 0.1 | 0 | 4 | 512 | 91.53 |
| 0.1 | $2 \times 10^{-5}$ | 4 | 512 | 91.77 |
| 0.1 | $2 \times 10^{-2}$ | 4 | 512 | 90.32 |
| 0.1 | $2 \times 10^{-4}$ | 10 | 512 | **92.61** |
| 0.1 | $2 \times 10^{-4}$ | 3 | 512 | 91.96 |
| 0.1 | $2 \times 10^{-4}$ | 4 | 64 | 91.44 |
| 0.1 | $2 \times 10^{-4}$ | 4 | 128 | 92.07 |
| 0 | 0 | 4 | 512 | 10.00 |

#### 3.5.2 Effect of Rank Maximization Loss Variant

We conducted experiments on the four variants of the rank maximization loss, $\mathcal{L}_{\mathcal{R}}$. The four variants are total least square minimum ($\mathcal{L}_{R1}$), nuclear norm ($\mathcal{L}_{R2}$), nuclear norm normalized by induced norm ($\mathcal{L}_{R3}$), and nuclear norm normalized by Frobenius norm ($\mathcal{L}_{R4}$). Additionally, we study the effect of attaching a bias column to the batch representation matrix $[z_b^\intercal]_{b=1}^B$, which prevents the representations from collapsing into affine subspaces instead of linear subspaces. Although $\mathcal{L}_{R1}$, $\mathcal{L}_{R3}$, and $\mathcal{L}_{R4}$) seem to be a better measurement of linear independence in theory, we found simple nuclear norm $\mathcal{L}_{R2}$ works best. Although we observe an increment in performance with the bias column attached, we decide not to use it in the main experiments for simplicity.

Table 5: Ablation study on the effect of different variants of $\mathcal{L}_R$ conducted on CIFAR-10. All experiments use ResNet-18. By default, we use $\mathcal{L}_{R2}$, the nuclear norm, as $\mathcal{L}_R$. The index in the subscript of loss weight $\alpha_R$ indicates which variation the weight is acting on.

| $\alpha_{R2}$ | $\alpha_{R1}$ | $\alpha_{R3}$ | $\alpha_{R4}$ | **Use Bias** | **Accuracy (%)** |
|---|---|---|---|---|---|
| $2 \times 10^{-4}$ | 0 | 0 | 0 | | 92.44 |
| 0 | $2 \times 10^{-3}$ | 0 | 0 | | 91.64 |
| 0 | 0 | $2 \times 10^{-3}$ | 0 | | 91.81 |
| 0 | 0 | 0 | $2 \times 10^{-3}$ | | 91.82 |
| $2 \times 10^{-4}$ | 0 | 0 | 0 | ✓ | 92.52 |

## 4 Related Work

**Contrastive learning.** Contrastive learning methods learn effective features by pushing away the representations of negative pairs and force positive pairs to have the same representations. Positive pairs are usually constructed from differently augmented versions of the same input, while negative pairs are all other inputs in a mini-batch. In other words, each input sample is considered as a class in the contrastive learning paradigm. Contrastive methods (Chen et al., 2020a; He et al., 2020; Chen et al., 2020b;c) have achieved state-of-the-art performance in many tasks. The success of contrastive methods, however, depends on a large number of negative samples. This can be achieved by either using a large batch size (Chen et al., 2020a;b) or a memory bank (He et al., 2020; Chen et al., 2020c). Both approaches require a large amount of memory and are computationally expensive.

**Consistency Regularization.** Similar to contrastive methods, consistency regularization based algorithms (Grill et al., 2020; Chen & He, 2020; Caron et al., 2020) also force the model to produce the similar representation for different views of the same input; however, consistency based methods do not use negative pairs. Without the need for a large batch size, they are generally less computationally expensive and have a smaller memory footprint. On the other hand, not using negative examples could lead to trivial solutions such as a constant mapping where all inputs are mapped to the same value. Therefore, the most important problem in consistency based methods is to add a constraint so that the model does not learn a degenerate mapping. Common approaches to prevent collapsing are asymmetric architecture, ad hoc architecture design, and stop gradient operations. Bardes et al. (2021) use an invariance loss for consistency regularization, a variance loss to prevent model collapsing, and a covariance loss to decorrelate the features.

**Information Theory.** Information theory has been widely utilized in self-supervised representation learning. For discrete representations, Hu et al. (2017) proposed the idea of maximizing the mutual information between input and representation while regularizing the network with consistency. Hjelm et al. (2018) also encourages the idea of maximizing mutual information between input and representations. Recently, more works start to focus on learning minimal sufficient representation. Tian et al. (2020) explore the importance of the augmentation in the lens of information theory and argues that mutual information between views should be minimized while keeping the task-relevant information intact. Tsai et al. (2021) further study the minimal sufficient representation problem and connect prior contrastive and predictive learning objectives to information theory.

## 5 Conclusion

We propose SimMER, a self-supervised learning algorithm. SimMER improves on previous works (Hu et al., 2021; He et al., 2020; Chen et al., 2020a; Grill et al., 2020; Caron et al., 2020) by replacing negative pairs and ad hoc architecture constructs with an interpretable trade-off between consistency regularization, entropy, and linear independence. Extensive experiments over standard datasets demonstrate the effectiveness of the SimMER. SimMER shows significant performance gains over previous state-of-the-art self-supervised learning methods on CIFAR-10 and Mini-ImageNet during linear evaluation. Furthermore, although not designed specifically for semi-supervised learning tasks, SimMER achieves performance on par with state-of-the-art semi-supervised methods.

## REFERENCES


Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning. *arXiv:2105.04906 [cs]*, May 2021. URL http://arxiv.org/abs/2105.04906. arXiv: 2105.04906.

J Beirlant, E J Dudewicz, and L Gyorfi. Nonparametric entropy estimation: an overview. pp. 14.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. *arXiv: Computer Vision and Pattern Recognition*, June 2020.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *International Conference on Machine Learning*, July 2020a.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E. Hinton. Big Self-Supervised Models are Strong Semi-Supervised Learners. In *Neural Information Processing Systems*, volume 33, June 2020b.

Xinlei Chen and Kaiming He. Exploring Simple Siamese Representation Learning. *arXiv: Computer Vision and Pattern Recognition*, November 2020.

Xinlei Chen, Haoqi Fan, Ross B. Girshick, and Kaiming He. Improved Baselines with Momentum Contrastive Learning. *arXiv: Computer Vision and Pattern Recognition*, March 2020c.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory 2nd Edition*. Wiley-Interscience, Hoboken, N.J, 2nd edition edition, July 2006. ISBN 978-0-471-24195-9.

William Falcon and The PyTorch Lightning team. PyTorch Lightning, 3 2019. URL https://github.com/PyTorchLightning/pytorch-lightning.

Maryam Fazel Sarjoui. *Matrix rank minimization with applications*. PhD thesis, ProQuest Dissertations Publishing, 2002. URL https://search.proquest.com/docview/305537461?pq-origsite=primo. ISBN: 9780493628547.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour, 2018.

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, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning. In *Neural Information Processing Systems*, volume 33, June 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Computer Vision and Pattern Recognition*, pp. 770–778, June 2016. doi: 10.1109/CVPR.2016.90.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. In *Computer Vision and Pattern Recognition*, pp. 9729–9738, June 2020. doi: 10.1109/CVPR42600.2020.00975.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. facebookresearch/moco, October 2021. URL https://github.com/facebookresearch/moco/blob/f7364915bc4e9161427d7c6bb4f454fa0f171337/colab/moco_cifar10_demo.ipynb. original-date: 2020-03-17T22:42:11Z.

R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Philip Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, August 2018.

Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximizing self-augmented training. In *International Conference on Machine Learning*, pp. 1558–1567, August 2017.

Zijian Hu, Zhengyu Yang, Xuefeng Hu, and Ram Nevatia. SimPLE: Similar Pseudo Label Exploitation for Semi-Supervised Classification. March 2021. URL http://arxiv.org/abs/2103.16725. arXiv: 2103.16725.

Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label Propagation for Deep Semi-Supervised Learning. pp. 5070–5079, 2019. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Iscen_Label_Propagation_for_Deep_Semi-Supervised_Learning_CVPR_2019_paper.html.

Diederik P. Kingma and Jimmy Lei Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, January 2015.

L F Kozachenko and N N Leonenko. Sample Estimate of the Entropy of a Random Vector. *Problems of Information Transmission*, 23(2):9, 1987. URL http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=ppi&paperid=797&option_lang=eng.

Chia-Wen Kuo, Chih-Yao Ma, Jia-Bin Huang, and Zsolt Kira. FeatMatch: Feature-Based Augmentation for Semi-supervised Learning. In *European Conference on Computer Vision*, pp. 479–495, August 2020. doi: 10.1007/978-3-030-58523-5_28.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts. In *International Conference on Learning Representations*, August 2016.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, December 2015. doi: 10.1007/S11263-015-0816-Y.

Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *International Conference on Learning Representations*, January 2017.

Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What Makes for Good Views for Contrastive Learning? *arXiv:2005.10243 [cs]*, December 2020. URL http://arxiv.org/abs/2005.10243. arXiv: 2005.10243.

Yao-Hung Hubert Tsai, Yue Wu, Ruslan Salakhutdinov, and Louis-Philippe Morency. Self-supervised Learning from a Multi-view Perspective. In *International Conference on Learning Representations*, May 2021.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29, pp. 3630–3638. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/90e1357833654983612fb05e3ec9148c-Paper.pdf.

Yang You, Igor Gitman, and Boris Ginsburg. Large Batch Training of Convolutional Networks. *arXiv: Computer Vision and Pattern Recognition*, August 2017.