





Supplementary Material: Towards Efficient and Effective Self-Supervised Learning of Visual Representations

Sravanti Addepalli* , Kaushal Bhogale* , Priyam Dey ,
and R.Venkatesh Babu 

Video Analytics Lab, Department of Computational and Data Sciences,
Indian Institute of Science, Bangalore

1 Background

We briefly discuss some of existing self-supervised learning approaches that have been used for the analysis in this paper.

RotNet: Rotation prediction, proposed by Gidaris et al. [11], has been one of the most successful pretext tasks for the learning of useful semantic representations. In this approach, the network is trained to predict one of the K rotations which was used for transforming the input image x_i . The authors found that $K = 4$ with $\mathcal{T} = \{0^\circ, 90^\circ, 180^\circ, 270^\circ\}$ produced the best results. Every image x_i is transformed using all four rotation transformations $x_i^{t_1}$, $x_i^{t_2}$, $x_i^{t_3}$ and $x_i^{t_4}$, and the network is trained to predict t_1 , t_2 , t_3 and t_4 , which are the rotation angles used for transforming x_i . The base encoder f_θ is trained by minimizing the following loss function \mathcal{L} :

$$\mathcal{L}_{RotNet} = \frac{1}{B} \sum_{i=0}^{B-1} \frac{1}{K} \sum_{k=0}^{K-1} \ell_{CE}(M_\theta(x_i^{t_k}), t_k) \quad (1)$$

Here, M_θ represents the network that takes as input rotated images $x_i^{t_k}$, and outputs the softmax predictions over the four possible rotation angles.

SimCLR: The work by Chen et al. [3] presents a Simple Framework for Contrastive Learning of Visual Representations (SimCLR), which utilizes existing architectures such as ResNet [13], and avoids the need for specialized architectures and memory banks. SimCLR proposed the use of multiple data augmentations, and a learnable nonlinear transformation between representations and the contrastive loss to improve the effectiveness of contrastive learning. The authors find the following augmentations to be best suited for the contrastive learning task - random crop and resize, random color jitter and random Gaussian blur. These augmentations are applied serially to every image x_i to generate two independent augmentations $x_i^{a_1}$ and $x_i^{a_2}$, which are considered as positives in the contrastive learning task. The $2(B - 1)$ augmentations of all other images in a

* Equal contribution.

Correspondence to: Sravanti Addepalli <sravantia@iisc.ac.in>

batch of size B are considered as negatives. The network is trained by minimizing the normalized temperature-scaled cross entropy loss (NT-Xent) loss with temperature T as shown in Eq.(2). The cosine similarity between two vectors a and b is denoted as $\text{sim}(a, b)$. The overall network formed by the composition of the base encoder f_θ and the projection network g_θ is represented by M_θ .

$$\mathcal{L}_{SimCLR} = -\frac{1}{2B} \sum_{i=0}^{B-1} \sum_{m=1}^2 \log \frac{\exp(\text{sim}(M_\theta(x_i^{a_1}), M_\theta(x_i^{a_2}))/T)}{\sum_{j=0}^{B-1} \sum_{l=1}^2 \mathbb{1}_{[j \neq i]} \exp(\text{sim}(M_\theta(x_i^{a_m}), M_\theta(x_j^{a_l}))/T)} \quad (2)$$

BYOL: While prior approaches relied on the use of negatives for training, Grill et al. [12] proposed Bootstrap Your Own Latent (BYOL), which could achieve state-of-the-art performance without the use of negatives. The two augmentations $x_i^{a_1}$ and $x_i^{a_2}$ are passed through two different networks - the base network M_θ , and the derived network M_ψ respectively. The weights of the base network are updated using back-propagation, while the weights of the derived network are obtained by computing a slow exponential moving average over the weights of the base network. The base network is trained such that the representation of $x_i^{a_1}$ at its output can be used to predict the representation of the $x_i^{a_2}$ at the output of the derived network, using a predictor network P_θ . The symmetric loss that is used for training the base network is shown below:

$$\mathcal{L}_{BYOL} = -\frac{1}{2B} \sum_{i=0}^{B-1} \text{sim}(P_\theta(M_\theta(x_i^{a_1})), M_\psi(x_i^{a_2})) + \text{sim}(P_\theta(M_\theta(x_i^{a_2})), M_\psi(x_i^{a_1})) \quad (3)$$

2 Eliminating false positives in self-supervised learning

As shown in Fig.1(b) of the main paper, two random augmentations of a given image may not always be similar to each other. The use of very small crops increases the likelihood of obtaining augmentations which may be unrelated to each other. This leads to false positives in instance-similarity based learning approaches. In Table-2 of the main paper, we use Grad-CAM [27] based saliency maps to select crops such that mean saliency score of the cropped image is greater than that of the full image. We describe this method in more detail below.

Mean-saliency based cropping: We denote the saliency map of an image using $G(x)$, which is a probability map indicating the importance of each pixel in the image. In order to select rectangular crops having high saliency score, we first calculate the mean probability score $P(x)$ for an image x of dimension $W \times H$ as follows:

$$P(x) = \frac{1}{W \cdot H} \sum_{i=0}^W \sum_{j=0}^H G_{i,j}(x) \quad (4)$$

For selecting a rectangular crop from the image, we randomly sample the top left corner coordinates (l, m) , width w , and height h from the valid range. These

values can be used to obtain a rectangular crop x^{a_1} . We formulate the saliency score of the crop x^{a_1} as follows:

$$P(x^{a_1}) = \frac{1}{w \cdot h} \sum_{i=l}^{l+w} \sum_{j=m}^{m+h} G_{i,j}(x) \quad (5)$$

The sampled crop is accepted only if $P(x^{a_1}) > P(x)$. We repeatedly sample until a valid crop is found, and restrict to a maximum of 10 tries. If no valid crop is found, we use a random crop. We observe that 10 tries are sufficient to find valid crops in most cases and random cropping is used for very few images.

Computational Budget: As shown in Table-2 of the main paper, with 50 epochs of training, the accuracy on BYOL baseline is 63.64%, which increases to 66.72% with the use of supervised saliency maps. However, this method assumes the availability of a network which is pre-trained on a relevant dataset, which may not always hold true. Hence, the computational budget for training this reference network needs to be considered too. We use fully supervised network trained for 90 epochs as the reference model for generation of saliency maps. Therefore, the total budget for the BYOL baseline is 140 epochs (50 + 90). As shown in Table-6 of the main paper, the accuracy obtained by training the BYOL baselines for 100 epochs is 71.02% which is 4.3% higher than the model that is trained for 50 epochs using saliency maps, with an effective training budget of 140 epochs. This shows that while the use of saliency maps from a pre-trained network helps improve accuracy, it is not a practical option in cases where a model that is pre-trained on a related dataset is not available a priori.

3 Details on Datasets

We present our analysis and results across the following datasets: CIFAR-10, CIFAR-100 [19] and ImageNet-100 [28], which is a 100-class subset of ImageNet [7]. We do not present our main results on the full ImageNet dataset due to computational limitations. However, we show the scalability of our approach to ImageNet on a short training schedule of 30-epochs. Details of these datasets are presented below:

CIFAR-10: CIFAR-10 [19] is a 10 class dataset comprising of 50,000 images in the training set and 10,000 images in the test set. The dataset consists of RGB images of dimension 32×32 . The images in the train and test sets are equally distributed across all classes.

CIFAR-100: CIFAR-100 [19] dataset consists of 50,000 images in the training set and 10,000 images in the test set, equally distributed across 100 classes. The dimensions and number of channels of images in CIFAR-100 is the same as CIFAR-10.

ImageNet: ImageNet [7] is a 1000-class dataset consisting of around 1.2 million images in the training set and 50,000 images in the validation set. We consider the validation set as the test set, since the true test set is held private. The dataset consists of RGB images of dimension 224×224 .

ImageNet-100: ImageNet-100 is a 100-class subset of the ImageNet dataset. We consider the same 100 class subset that was used by Tian et al. [28].

4 Details on Training hyperparameters

We consider the following baselines for our experiments: SimCLR [3], BYOL [12], SimSiam [4] and SwAV [2]. Since these papers primarily demonstrate results on the ImageNet dataset, using larger architectures and longer training schedules, we perform extensive hyperparameter search to obtain strong results for the baselines on the datasets considered. We use the ResNet-18 [13] architecture for all experiments other than the ImageNet-1k runs, where ResNet-50 was used. The dimension of features before the last fully-connected classification layer is 512, which is smaller than that of ResNet-50, where the dimension is 2048. We fix the batch size to be 512 in all our experiments. We discuss details on hyperparameter tuning for obtaining strong baselines in Section-4.1, and describe the same for the proposed method in Section-4.2.

4.1 Details on the Baseline Implementation

SimCLR: For the SimCLR [3] baseline on CIFAR-10 and CIFAR-100, we perform a hyperparameter search for the learning rate, weight decay and the temperature used in the loss. We tune the learning rate in the range of 0.1 to 1 with a step size of 0.1, and the temperature in the range of 0.1 to 0.5 with a step size of 0.1. For weight decay we search over the range $\{ 5 \times 10^{-4}, 1 \times 10^{-4}, 1 \times 10^{-5}, 1 \times 10^{-6} \}$. Finally, we use a learning rate of 0.5, weight decay of 1×10^{-4} and a temperature of 0.2 for all our experiments. Following the official implementation [3], we use cosine learning rate schedule with a warm-up of 10 epochs. For the projection head, we use a 2 layer MLP with the hidden layer consisting of 512 nodes. The output is a 128-dimensional vector. We use batch normalization layers [15] in the projection head. For ImageNet-100, we use the implementation and tuned hyperparameters from the repository *solo-learn* [6].

BYOL: For BYOL [12] baselines on CIFAR-10 and CIFAR-100, we perform a search for the learning rate and weight decay in the same manner as described in the paragraph above. Additionally we tune the momentum τ of the target network in BYOL [12] from the values $\{ 0.8, 0.85, 0.9, 0.95, 0.99, 0.995, 0.999 \}$. Finally, we use a learning rate of 0.8 and weight decay of 1×10^{-4} for CIFAR-10 and CIFAR-100. We tune the learning rate for ImageNet-100 in the range 0.4 to 0.7 with a step size of 0.1. We finally use a learning rate of 0.6 and a weight decay of 1×10^{-4} for ImageNet-100. We use τ of 0.95, 0.85 and 0.95 for CIFAR-10, CIFAR-100 and ImageNet-100 respectively.

SimSiam: For the SimSiam [4] baselines, we use the implementation from the repository [14], and perform a hyperparameter search for the learning rate, weight decay and the number of projection layers used in the loss. We tune the learning rate in the range of 0.03 to 0.1 with a step size of 0.01, and additionally try 0.2 as well. For weight decay we search over the range $\{ 6 \times 10^{-4}, 5 \times 10^{-4},$

4×10^{-4} , 3×10^{-4} , 1×10^{-4} , 1×10^{-5} , 1×10^{-6} }. For the number of projection layers, we consider two values, 2 and 3. Finally, for CIFAR10, we use a learning rate of 0.07, weight decay of 4×10^{-4} and number of projection layers as 2. For CIFAR100, we use a learning rate of 0.05, weight decay of 5×10^{-4} and number of projection layers as 3. Following the official implementation [4], we use the cosine learning rate schedule with a warm-up of 10 epochs. For the projection head, the hidden layer is set to 2048 nodes and output is a 2048-dimensional vector. For the prediction head, the hidden layer has 512 nodes and the output is again a 2048-dimensional vector. We use batch normalization layers in the projection and prediction heads similar to the official implementation [4].

SwAV: We use the code and hyperparameters from the official implementation [2]. For CIFAR-10, we search for the optimal number of prototypes over the values {10, 30, 50, 70, 90, 100, 120, 150}, ϵ over {0.01, 0.03} and queue over {0, 38, 384}. we finally set the number of prototypes to 100 without using a queue, and set ϵ to 0.03. Since CIFAR-10 images are small in size (32x32), we do not use the multi-crop strategy. We use the same settings for CIFAR-100 as well. For ImageNet-100, we scale the default number of prototypes from the official code [2] by a factor of 10 to 300, based on the scaling of number to classes from 1000 to 100. We use search for queue length in the range {0, 384, 1920, 3840} and set it to 384 finally. For the ImageNet-1k runs, we skip the use of multi-crop augmentations to speed up the training.

4.2 Details on the Proposed Implementation

We use the same hyperparameters as the respective baselines for the implementation of the proposed method, and additionally tune only the value of λ (Eq.1 of the main paper), which is the weighting factor used for the rotation loss. We use a 2 layer MLP for the rotation prediction task and use batch normalization for the hidden layer. For finding the best setting of λ , we tune for $1/(2 \cdot \lambda)$ in the range 1 to 10 with step size of 1, and for $2 \cdot \lambda$ in the range 0 to 1 with a step size of 0.1. In order to minimize computational overheads, we use the same value of λ as ImageNet-100 on ImageNet-1k as well.

For SimCLR, we use $2 \cdot \lambda$ as 1 for CIFAR-10 and CIFAR-100, and 0.1 for ImageNet-100. For BYOL, we use $1/(2 \cdot \lambda)$ as 5 for CIFAR-10 and CIFAR-100, and 6 for ImageNet-100. For SimSiam, we set the value of $2 \cdot \lambda$ to 0.1 for CIFAR-10 and 0.2 for CIFAR-100. For SwAV, we set the value of $2 \cdot \lambda$ to 0.5 for CIFAR-10 and CIFAR-100, and 0.1 for ImageNet-100 and ImageNet-1k.

4.3 Training Details of Linear Evaluation

The linear evaluation stage consists of training a linear classification layer on top of the frozen backbone network. We do not update the batch statistics in this stage. For linear evaluation on CIFAR-10 and CIFAR-100, we do not apply any spatial augmentations to the images during training. We use the SGD optimizer with momentum of 0.9. We train for 100 epochs with a batch size of 512. We use a learning rate of 1.0 which is the best setting chosen from the range { 0.1,

0.5, 1.0, 1.5, 2.0 }. The same settings are used for ImageNet-100 BYOL linear evaluation as well.

For SimSiam linear evaluation, we apply Random cropping and horizontal flipping. We use the SGD optimizer with momentum over 100 epochs using a batch size of 512, learning rate of 30.0 and momentum of 0.9, as recommended by the authors [4]. Cosine scheduler with decay is employed without any warmup for the training.

On ImageNet-100, we use the settings from the repository *solo-learn* [6] for the linear evaluation of SimCLR [3]. For linear evaluation of SwAV models on ImageNet-100 and ImageNet-1k, we use the settings from their official repository [2], and use 30 epochs of training on ImageNet-1k.

We use the same hyperparameters for the linear evaluation of the proposed approach and the respective baselines.

4.4 Training Details of Semi-supervised learning

We follow the semi-supervised training settings from [2, 20] for both 1% and 10% labels. Specifically, we train for 20 epochs with a batch size of 256. For the setting of 1% labels, we use a learning rate of 0.02 for the backbone and 5.0 for the linear layer. For the setting of 10% labels, we use a learning rate of 0.01 for the backbone and 0.2 for the linear layer. We decay the learning rates by a factor of 0.2 at epochs 12 and 16 in both the settings. We do not use weight decay during the training.

5 Ablation Experiments

In this section, we present additional experiments and results to highlight the significance of various aspects of the proposed method.

Table 1: **Rotation Angles:** Ablation experiments to show the impact of the rotation set (\mathcal{T}) used in the proposed approach. K-Nearest Neighbor (KNN) classification accuracy (%) with $K=200$ and Linear evaluation accuracy (%) on the CIFAR-100 dataset are reported for the baseline (BYOL [12]) and variations in the proposed approach (BYOL + rotation).

Rotation Set (\mathcal{T})	$ \mathcal{T} $	KNN	Linear
ϕ (BYOL [12])	0	54.37	60.67
$\{0^\circ, 180^\circ\}$	2	58.03	66.21
$\{90^\circ, 270^\circ\}$	2	53.86	62.96
$\{0^\circ, 90^\circ\}$	2	56.41	65.24
$\{0^\circ, 270^\circ\}$	2	56.29	65.04
$\{0^\circ, 90^\circ, 180^\circ, 270^\circ\}$	4	58.41	67.03
$\{45^\circ, 135^\circ, 225^\circ, 315^\circ\}$	4	57.60	65.50
$\{0^\circ, 45^\circ, \dots, 270^\circ, 315^\circ\}$	8	57.54	67.25
$\{0^\circ, 30^\circ, \dots, 300^\circ, 330^\circ\}$	12	55.43	63.61

5.1 Impact of Variation in Rotation Angles

In the proposed method, we transform every input image using a rotation transformation $t(\cdot)$ which is randomly sampled from the set $\mathcal{T} = \{0^\circ, 90^\circ, 180^\circ, 270^\circ\}$. We present results by varying the number of rotation angles in the set \mathcal{T} with BYOL [12] as the base approach in Table-1. While the use of 8 rotation angles results in the best results, we use 4 rotation angles (which results in marginally lower accuracy after linear evaluation) due to the simplicity of implementation, since rotation by multiples of 90° does not require additional transformations such as cropping and resizing. The use of two rotation angles with $\mathcal{T} = \{0^\circ, 180^\circ\}$ leads to a drop of 0.82% in linear evaluation accuracy when compared to the proposed method of using 4 rotation angles. However, this setting is still 5.54% better than the BYOL baseline. Therefore, the surprisingly simple task of predicting whether an image is in the correct orientation, or turned upside down is sufficient to boost the performance of the baseline method significantly. In the two-angle prediction task, excluding the 0° rotation angle with $\mathcal{T} = \{90^\circ, 270^\circ\}$ leads to a significant drop of 3.25% when compared to using $\mathcal{T} = \{0^\circ, 180^\circ\}$. We further note that using rotation transformations that are uniformly spaced ($\mathcal{T} = \{0^\circ, 180^\circ\}$) leads to better performance when compared to the use of $\mathcal{T} = \{0^\circ, 90^\circ\}$ or $\mathcal{T} = \{0^\circ, 270^\circ\}$.

These experiments show that the level of difficulty of the auxiliary task plays a crucial role in the representations learned. The task should neither be too difficult (12 rotation angles), nor should it be too easy (2 rotation angles). Moreover, since the test images would have 0° rotation angle, it helps to include this as one of the classes in \mathcal{T} .

Table 2: **Effect of number of layers shared with the Rotation Task:** Ablation experiments to show the impact of number of layers shared with the rotation task in the proposed approach. K-Nearest Neighbor (KNN) classification accuracy (%) with $K=200$ and Linear evaluation accuracy (%) on the CIFAR-100 dataset are reported for the baseline (BYOL [12]) and variations in the proposed approach (BYOL + rotation).

Layers shared with Rotation Task	KNN	Linear
None (BYOL [12] baseline)	54.37	60.67
First Convolutional layer (f_θ)	50.36	52.50
+ Block - 1 (f_θ)	50.98	52.84
+ Block - 2 (f_θ)	51.75	54.85
+ Block - 3 (f_θ)	52.77	58.31
+ Block - 4 (f_θ)	58.29	66.06
+ Projection network (g_θ)	58.41	67.03

5.2 Impact of Number of Shared Layers across Tasks

In the proposed approach, we share the base encoder f_θ and the Projection network g_θ between the instance-similarity task and the rotation task. We perform

experiments to study the impact of varying the number of shared layers between the two tasks. The results of these experiments on the CIFAR-100 dataset with BYOL [12] as the base method are presented in Table 2. The ResNet-18 architecture consists of a convolutional layer followed by 4 residual blocks. As an example, for the case where only Block-1 is shared between the two tasks, we replicate the remaining part of f_θ and g_θ separately for the rotation task. Thus in this case, the rotation task only impacts Block-1 of the final base encoder f_θ . As shown in Table 2, increasing the number of shared blocks results in better performance. In fact, sharing only the first few layers leads to a degradation in performance when compared to the BYOL baseline. This indicates that the rotation task indeed helps improve the convergence of the overall network, and is not merely helping with learning better filters in the initial layers, as was the case in RotNet [11] training.

Table 3: **Robustness to Image Augmentations:** Ablation experiments to show the impact of color jitter augmentation on the baseline (BYOL [12]) and proposed method (BYOL + Rotation). K-Nearest Neighbor (KNN) classification accuracy (%) with K=200 and Linear evaluation accuracy (%) on the CIFAR-10 dataset are reported. The proposed method is significantly more robust to the absence of color jitter augmentation.

	KNN	Linear
BYOL [12]	86.56	89.30
BYOL (without Color Jitter)	82.21 _{-4.35}	85.90 _{-3.40}
BYOL + Rotation	89.80	91.89
BYOL + Rotation (without Color Jitter)	88.52 _{-1.28}	91.28 _{-0.61}

5.3 Robustness to Image Augmentations

BYOL [12] is known to be more robust to image augmentations when compared to contrastive learning methods such as SimCLR [3]. The authors claim that although color histograms are sufficient for the instance-similarity task, BYOL is still able to learn additional semantic features for the image even without color jitter. We compare the impact of removing the color jitter augmentation on the baseline (BYOL) and the proposed approach (BYOL + Rotation) on CIFAR-10 dataset in Table-3. We observe that addition of rotation task boosts the robustness to such augmentations even further. The absence of color jitter leads to a drop of 3.4% in linear evaluation accuracy of BYOL, whereas the drop in accuracy for the proposed method without color jitter is only 0.61%, which is significantly lower. This makes the proposed method suitable for fine-grained image classification tasks as well, where the network needs to rely on color information for achieving good performance.

5.4 Exploring Different Loss Formulations for the Rotation Task

The proposed approach combines Cross-Entropy (CE) loss for the rotation task with various instance-similarity based tasks as shown in Eq.1 of the main pa-

Table 4: **Exploring Different Loss Formulations for the Rotation Task:** Ablation experiments to show the impact of different loss formulations on the rotation task. K-Nearest Neighbor (KNN) classification accuracy (%) with K=200 and Linear evaluation accuracy (%) on the CIFAR-10 dataset are reported. We additionally report the Rotation Task Accuracy (%) obtained by freezing the base encoder f_θ and training a 2-layer MLP for the rotation classification task.

	KNN	Linear	Rotation Acc (f_θ)
BYOL Baseline [12]	86.56	89.30	73.40
Ours (Classification with CE Loss)	89.80	91.89	93.73
Classification with SupCon [16] Loss	88.05	90.19	81.86
Minimizing cosine similarity between Rotation Augmentations	86.84	88.95	77.27
BYOL + Rotation Augmentation	74.32	79.70	66.61
Ours (BYOL + Rotation) + Rotation Augmentation	84.49	87.75	94.24

per. We explore the use of different loss formulations for the rotation task with BYOL [12] as the base method on the CIFAR-10 dataset in Table-4. We first replace the CE loss for rotation with SupCon [16] loss, where all images with a similar rotation angle are treated as positives, while the remaining images in the batch are treated as negatives. This results in a significant drop of 1.7% in the Linear evaluation accuracy. We observe a larger drop of 2.94% when the CE loss is replaced with cosine similarity between two unique rotation augmentations sampled from the transformation set $\mathcal{T} = \{0^\circ, 90^\circ, 180^\circ, 270^\circ\}$. While the three approaches of minimizing CE loss, SupCon loss and cosine similarity between rotation augmentations seek to cluster similarly rotated images together and repel others, we find large differences in the representations learned. This shows that explicitly enforcing fixed categories in the auxiliary task helps in building a global semantic representation which is reinforced across training batches. This is exclusively achieved in the minimization of CE loss since it considers specific rotation based categories.

We study the impact of adding rotations from the set $\mathcal{T} = \{0^\circ, 90^\circ, 180^\circ, 270^\circ\}$ as augmentations in the BYOL training pipeline. Contrary to the proposed approach, this would encourage representations that are invariant to rotation. This leads to a large drop of 9.6% when compared to the BYOL baseline. This is consistent with the observations by Chen et al. [3] that rotation as an augmentation is not helpful in learning good representations. By including the rotation classification task in addition to this in the training objective, the accuracy improves by 8.05%, although it is still lower than the BYOL baseline due to the inclusion of rotation as augmentations which is contrasting to the rotation classification objective.

We further compare the rotation sensitivity of representations at the output of the base encoder f_θ . Similar to the experiments in Section-5.3 of the main paper, we freeze the network till the f_θ and train a rotation task classifier over this using a 2-layer MLP head. We measure the rotation task accuracy, which serves as an indication to the rotation sensitivity of the base network. We observe that the trend in accuracy on the linear evaluation task is similar to the rotation

task accuracy, indicating that rotation-covariant representations are better for downstream tasks. While the use of rotation augmentation along with rotation task prediction achieves a very high rotation accuracy, its performance on the contrastive task is only 64.34%, which is significantly lower than the baseline and the proposed methods (Ref. Table-3 in the main paper). Therefore, the accuracy on linear evaluation task is also lower than these methods.

6 Reduction in Noise during training

To further demonstrate how the well-posedness of the rotation task reduces noise during the training, we plot the Signal-to-Noise ratio (SNR) of the gradients. For this, we follow Mitrovic et al. [23] and at each iteration, we compute the ratio of the exponential moving average of mean and variance of the gradients, and average it across all the parameters to obtain the SNR. Fig-1 shows the progression of SNR during training on CIFAR-10 with SimSiam [4] as the base method. We find that our method indeed improves the SNR during training by providing a *noise-free* supervisory signal and hence facilitates the learning of representations in an efficient and effective way.

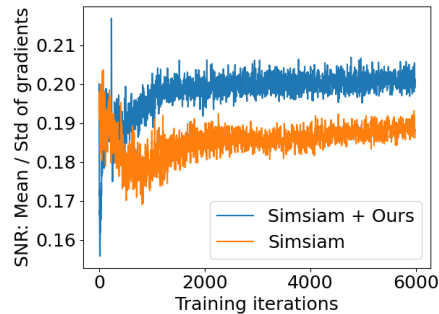


Fig. 1: Plot of SNR during SimSiam training on CIFAR-10 dataset. The use of rotation task (SimSiam+Ours) reduces the noise in the gradients, leading to faster convergence.

7 Transfer Learning

In this section, we report results using a ResNet-50 architecture with a 30-epoch training schedule. We perform the pretraining across 4 Nvidia Tesla V100 GPUs. We do not use multi-crop strategy in order to reduce the computational overheads. For all the ImageNet-1k runs, we do not perform additional hyperparameter tuning for the proposed approach, and use the same value of λ that was best in the SwAV ImageNet-100 runs ($2 \cdot \lambda = 0.1$). Using the linear evaluation training code and hyperparameters from the official SwAV repository for 30 epochs on the ImageNet-1k dataset, we achieve 54.9% accuracy using the SwAV baseline, and 57.3% accuracy using the proposed method, resulting in a

gain of 2.4% (Table-5). This shows that the proposed approach generalizes well to large-scale datasets and larger model capacities as well.

Table 5: **Transfer Learning (Classification):** Performance (%) after linear evaluation on different datasets with a ResNet-50 backbone trained using SwAV [2] and the proposed approach.

	ImageNet	CIFAR-10	CIFAR-100	Flowers	Caltech	Aircraft	DTD	Cars	Food	Pets	SUN	VOC	Avg
SwAV [2]	54.90	86.22	64.18	83.53	80.91	38.78	69.79	31.65	59.41	70.73	52.48	76.33	64.08
SwAV + Ours	57.30	87.85	66.94	85.78	84.18	42.09	69.68	32.52	59.46	71.27	53.25	76.70	65.59

Classification: We evaluate the generalization of the learned representations to other datasets by training a linear classifier on the pretrained backbone after freezing the weights of the backbone, as reported by Caron et al. [2]. We report transfer learning results on CIFAR-10 [19], CIFAR-100 [19], Oxford 102 Flowers [24], Caltech-101 [10], FGVC Aircraft [22], DTD [5], Stanford Cars [18], Food-101 [1], Oxford-IIIT Pets [25], SUN397 [30] and Pascal VOC2007 [9] datasets, as is common in literature [17, 3, 8]. We use the code, hyperparameter tuning strategy and validation splits from the official repository of Ericsson et al. [8] for obtaining results on the SwAV baseline. For the evaluation of the proposed method, we use the best hyperparameters obtained for baselines, in order to highlight the gains obtained using the proposed approach more clearly. We achieve better performance across most of the datasets, and similar performance as the baseline on the DTD dataset [5]. This is possibly because the DTD dataset is composed of textures only, and the images are rotation invariant. Therefore, learning representations that are covariant to rotation does not help in this case. Overall, we obtain an average improvement of 1.51% across all datasets.

Table 6: **Transfer Learning (Object Detection):** Performance (AP, AP50 and AP75) on Pascal VOC [9] dataset for the task of Object Detection using Faster RCNN [26] FPN [21] with a ResNet-50 backbone that is pretrained using SwAV [2] and the proposed approach. Pascal VOC07+12 trainval dataset is used for training and VOC07 test is used for evaluation. We consider two settings for evaluation: first with the ResNet-50 backbone being frozen, and second with the backbone being updated during training (Finetune).

Method	VOC (Frozen)			VOC (Finetune)		
	AP	AP50	AP75	AP	AP50	AP75
SwAV [2]	44.10	74.54	45.00	43.80	74.46	45.07
SwAV + Ours	45.12	75.37	46.67	45.19	75.17	46.67

Object Detection: We evaluate the generalization of the learned representations to the task of Object Detection on the Pascal VOC dataset [9] using

Faster RCNN [26] with Feature Pyramid Network [21] as the backbone. Pascal VOC07+12 trainval dataset is used for training and VOC07 test is used for evaluation. We consider two settings for evaluation: first with the ResNet-50 backbone being frozen, and second with the backbone being updated during training (Finetune). The training is done using the detectron2 framework [29] and their hyperparameters, as used by Ericsson et al. [8]. As shown in Table-6, we obtain consistent gains across the metrics AP, AP50 and AP75 in both evaluation settings.

8 Additional Results on ImageNet

In this section, we show additional results of the proposed method on ImageNet using the ResNet-50 architecture. Table-7 shows the model performance for longer training epochs, highlighting that the proposed approach can indeed scale to a longer training regime as well. We also achieve consistent gains of around 2.5% over the SwAV baselines for 30 and 50 epoch runs with and without multicrop, and across different batch sizes as shown in Table-8.

Table 7: IN-1K: Improvements obtained on longer training epochs

#Epochs	Method	Linear Acc (%)
35	SwAV + Ours	59.5
50	SwAV + Ours	61.9
100	SwAV + Ours	64.3

Table 8: IN-1K: Performance across different settings

Method	30 ep, B256 w/o Multicrop	50 ep, B1024 with Multicrop
SwAV	54.9	65.8
SwAV + Ours	57.3	68.3

Bibliography

- [1] Bossard, L., Guillaumin, M., Van Gool, L.: Food-101—mining discriminative components with random forests. In: European Conference on Computer Vision (ECCV) (2014) [11](#)
- [2] Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., Joulin, A.: Un-supervised learning of visual features by contrasting cluster assignments. In: Advances in Neural Information Processing Systems (NeurIPS) (2020) [4](#), [5](#), [6](#), [11](#)
- [3] Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: Proceedings of the 37th International Conference on Machine Learning (ICML) (2020) [1](#), [4](#), [6](#), [8](#), [9](#), [11](#)
- [4] Chen, X., He, K.: Exploring simple siamese representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021) [4](#), [5](#), [6](#), [10](#)
- [5] Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., Vedaldi, A.: Describing textures in the wild. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014) [11](#)
- [6] da Costa, V.G.T., Fini, E., Nabi, M., Sebe, N., Ricci, E.: Solo-learn: A library of self-supervised methods for visual representation learning (2021), <https://github.com/vturrisi/solo-learn> [4](#), [6](#)
- [7] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2009) [3](#)
- [8] Ericsson, L., Gouk, H., Hospedales, T.M.: How well do self-supervised models transfer? In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021) [11](#), [12](#)
- [9] Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International journal of computer vision (IJCV) (2010) [11](#)
- [10] Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In: Conference on Computer Vision and Pattern Recognition Workshop (2004) [11](#)
- [11] Gidaris, S., Singh, P., Komodakis, N.: Unsupervised representation learning by predicting image rotations. In: International Conference on Learning Representations (ICLR) (2018) [1](#), [8](#)
- [12] Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., Piot, B., kavukcuoglu, k., Munos, R., Valko, M.: Bootstrap your own latent - a new approach to self-supervised learning. In: Advances in Neural Information Processing Systems (NeurIPS) (2020) [2](#), [4](#), [6](#), [7](#), [8](#), [9](#)

- [13] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) **1, 4**
- [14] Hua, T.: A pytorch implementation for paper, exploring simple siamese representation learning (2021), <https://github.com/PatrickHua/SimSiam> **4**
- [15] Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning (ICML) (2015) **4**
- [16] Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., Krishnan, D.: Supervised contrastive learning. In: Advances in Neural Information Processing Systems (NeurIPS) (2020) **9**
- [17] Kornblith, S., Shlens, J., Le, Q.V.: Do better imagenet models transfer better? In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019) **11**
- [18] Krause, J., Deng, J., Stark, M., Fei-Fei, L.: Collecting a large-scale dataset of fine-grained cars (2013) **11**
- [19] Krizhevsky, A., et al.: Learning multiple layers of features from tiny images (2009) **3, 11**
- [20] Li, J., Zhou, P., Xiong, C., Socher, R., Hoi, S.C.: Prototypical contrastive learning of unsupervised representations. arXiv preprint arXiv:2005.04966 (2020) **6**
- [21] Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR) (2017) **11, 12**
- [22] Maji, S., Rahtu, E., Kannala, J., Blaschko, M., Vedaldi, A.: Fine-grained visual classification of aircraft. arXiv preprint arXiv:1306.5151 (2013) **11**
- [23] Mitrovic, J., McWilliams, B., Rey, M.: Less can be more in contrastive learning. In: "I Can't Believe It's Not Better!" NeurIPS Workshop (2020) **10**
- [24] Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number of classes. In: 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing (ICVGIP) (2008) **11**
- [25] Parkhi, O.M., Vedaldi, A., Zisserman, A., Jawahar, C.: Cats and dogs. In: 2012 IEEE conference on computer vision and pattern recognition. pp. 3498–3505. IEEE (2012) **11**
- [26] Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems (NeurIPS) (2015) **11, 12**
- [27] Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE international conference on computer vision (ICCV) (2017) **2**
- [28] Tian, Y., Krishnan, D., Isola, P.: Contrastive multiview coding. In: European conference on computer vision (ECCV) (2020) **3, 4**

- [29] Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R.: Detectron2. <https://github.com/facebookresearch/detectron2> (2019) 12
- [30] Xiao, J., Hays, J., Ehinger, K.A., Oliva, A., Torralba, A.: Sun database: Large-scale scene recognition from abbey to zoo. In: 2010 IEEE computer society conference on computer vision and pattern recognition (CVPR) (2010) 11