SimMatchV2: Semi-Supervised Learning with Graph Consistency

Mingkai Zheng[1] Shan You[2*]
Lang Huang[3] Chen Luo[5] Fei Wang[4] Chen Qian[2] Chang Xu[1]
[1]School of Computer Science, Faculty of Engineering, The University of Sydney
[2]SenseTime Research [3] The University of Tokyo
[4]University of Science and Technology of China
[5]State Grid Anhui Electric Power Research Institute

Abstract

Semi-Supervised image classification is one of the most fundamental problem in computer vision, which significantly reduces the need for human labor. In this paper, we introduce a new semi-supervised learning algorithm - SimMatchV2, which formulates various consistency regularizations between labeled and unlabeled data from the graph perspective. In SimMatchV2, we regard the augmented view of a sample as a node, which consists of a label and its corresponding representation. Different nodes are connected with the edges, which are measured by the similarity of the node representations. Inspired by the message passing and node classification in graph theory, we propose four types of consistencies, namely 1) node-node consistency, 2) node-edge consistency, 3) edge-edge consistency, and 4) edge-node consistency. We also uncover that a simple feature normalization can reduce the gaps of the feature norm between different augmented views, significantly improving the performance of SimMatchV2. Our SimMatchV2 has been validated on multiple semi-supervised learning benchmarks. Notably, with ResNet-50 as our backbone and 300 epochs of training, SimMatchV2 achieves 71.9% and 76.2% Top-1 Accuracy with 1% and 10% labeled examples on ImageNet, which significantly outperforms the previous methods and achieves state-of-the-art performance.

1. Introduction

In the past decades, deep learning has demonstrated its superior performance in various tasks [15, 36, 17, 59]. However, its outstanding performance is usually based on the fully-supervised setting, which requires a large amount of data and annotations. In a real-world scenario, high-quality data annotations are fairly hard to collect since it is generally expensive, time-consuming, and sometimes require expert knowledge (*e.g.* medical images). Semi-supervised learning (SSL) has recently drawn much attention to the research community due to its efficiency in dealing with few labeled data and leveraging massive unlabeled data, significantly reducing the need for human labor.

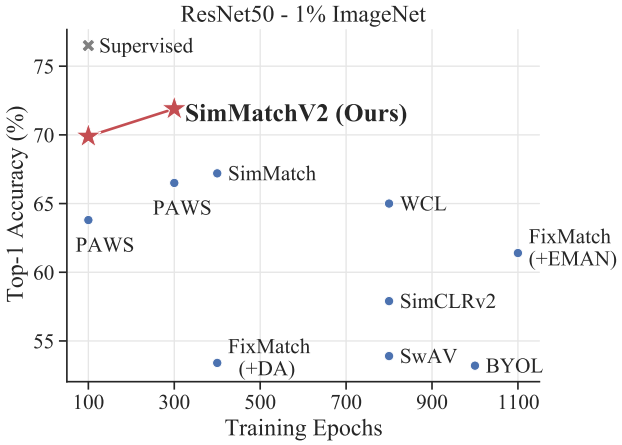


Figure 1: The Top-1 Accuracy of ResNet-50 on ImagetNet with 1% labeled data. Our proposed SimMatchV2 performs significantly better than previous methods with fewer training epochs. Most importantly, 100 epochs of SimMatchV2 training only takes 14.7 GPU days, but the performance is much higher than PAWS [1] with 300 epochs (68 GPU days).

For the typical SSL algorithms, one standard paradigm is to train a semantic classifier on the labeled data, then use the classifier to make a prediction on the unlabeled data as the pseudo-label [33]. Besides, a class-level "consistency regularization" will be applied to enforce consistency between the class predictions of a model on different perturbations of the same input, which encourages the model to learn more robust and invariant features that generalize well to unseen data. Thus, the quality of pseudo-labels plays a crucial role in consistency regularization, and many works focus on generating stable and high-quality pseudo-labels. For example, MixMatch [3], ReMixMatch [2], and FixMatch [43] are three typical "consistency regularization" based methods that significantly improve the performance of the SSL

*Correspondence to: Shan You <youshan@sensetime.com>.

tasks. However, these methods highly depend on the classifier trained on the labeled data. A limited number of labels will result in a poor classifier that generates inaccurate pseudo-labels and leads to bad performance.

With the recent success of unsupervised visual representation learning [9, 24, 10, 11, 22, 7, 12, 27, 28, 23], the pre-training and fine-tuning paradigm has gained popularity for the semi-supervised learning. In general, the model will be first trained on a large amount of unlabeled data with a proxy task (*e.g*. instance discrimination), which also imposes the consistency regularization on the instance level (*i.e*. forcing the model to give similar representation for the different perturbations of the same instance). Then, a classifier will be attached to the pre-trained model, and the whole model will be fine-tuned with the limited labeled data. Many works [29, 35, 26, 67, 68, 13, 50, 47, 30, 54] have been proposed to design a better proxy task to improve the quality of the representation. Apparently, the biggest advantage of this approach is that it does not depend on the supervised trained classifier but still yields high-quality representations for unlabelled data. Nevertheless, the drawback of this approach is also very obvious, since no labels were used during the pre-training, making the model hard to converge. Generally, the pre-training requires more than 800 epochs to converge, which is extremely time-consuming.

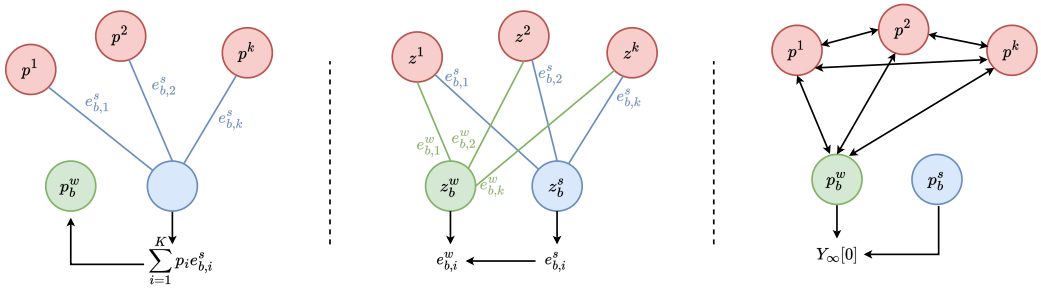
Although many different types of consistency regularizations have been proposed, including class-level [3, 2, 43], instance-level [9, 24, 22], and those that consider both [66, 43], how to establish a complete and effective consistency regularizations remains unresolved. This paper introduces a new SSL algorithm - SimMatchV2, which provides a complete consistency regularization solution for various scenarios. The overall idea is inspired by the message passing and node classification in the graph theory. In SimMatchV2, we consider the augmented view of a sample as a node in the graph which consists of a node label and its corresponding representation. Different nodes are connected with the edges, which are measured by the similarity of the node representations. Thus, the information of different nodes can be efficiently propagated through the graph. To this end, we propose four types of consistency regularization, namely 1) node-node consistency, 2) node-edge consistency, 3) edge-edge consistency, and 4) edge-node consistency. Note that the 1) node-node consistency has been proposed in previous works, but our SimMatchV2 provides a complete and unified solution to design the consistencies. Moreover, we also propose a feature normalization technique to reduce the gaps between the feature norm between different augmented views and greatly improve the performance. Our contributions can be summarized as:

- We propose SimMatchV2 - a novel semi-supervised algorithm that models various consistency regularizations from the graph perspectives and proposes four types of consistencies to provide a complete solution for the regularization in SSL.
- We show that a simple feature normalization reduces the gaps between the feature norm of different augmented views and greatly improves the performance.
- SimMatchV2 sets a new state-of-the-art performance on various SSL benchmarks. Notably, with ResNet-50 as our backbone and 300 epochs of training, SimMatchV2 achieves 71.9% and 76.2% Top-1 accuracy with 1% and 10% labeled data on ImageNet. This result is +4.7% and +0.7% better than prior arts.

2. Related work

Consistency Regularization with Pseudo-Label. Pseudo-label is a method that aims to generate labels for unlabeled data. As has been mentioned, it is often used in conjunction with consistency regularization, which constrains the model from making similar predictions on different perturbations of the same input. Most of these methods focus on generating high-quality pseudo-labels and exploring better perturbation strategies. For example, Π-model [32] propose a temporal ensemble strategy for the pseudo-label to reduce the noise in the target. Mean-Teacher [46] instead, maintain an exponential moving average (EMA) network to generate more reliable and stable pseudo-labels. Later, MixMatch [3] was proposed and greatly improved the SSL performance. Specifically, it adopts the averaged prediction of multiple strongly augmented views of the same image as the pseudo-label, then incorporates with the Mixup [63] augmentation that generates a convex combination of pairs of images and uses the corresponding mixed pseudo-label as the supervision target. ReMixMatch [2] improves MixMatch by using only a weakly augmented view to generate the pseudo-label and also introduces the distribution alignment to encourage the marginal distribution of predictions on unlabeled data to be consistent with the labeled data. FixMatch [43] abandons these complicated strategies and proposes a confidence-based threshold method to filter out the low-confidence pseudo-labels during the training.

Self-Supervised Learning for SSL. Self-supervised learning has demonstrated its outstanding performance in learning visual representations without labels. It is intuitive and natural to leverage the proxy tasks from self-supervised learning to help with SSL tasks. As mentioned, the most straightforward solution is to pre-train the model in a self-supervised manner on a large amount of unlabeled data and then fine-tune the model with the limited labeled data. In this way, the design of the self-supervised task is particularly important. For example, instance discrimination [53] is one of the most popular proxy tasks, which aims to encourage different augmented views of the same image to



(a) Node-Edge Consistency (b) Edge-Edge Consistency (c) Edge-Node Consistency

Figure 2: Illustration of various consistency in SimMatchV2. The green and blue nodes are from the weakly and strongly augmented view of the unlabeled data, the red nodes are the other samples.

be pulled closer on embedding space but pushes apart all the other images away. BYOL [22] and SimSiam [12] omit the negatives in contrastive learning and leverage an asymmetric structure to avoid representation collapse. Recently, some works (*e.g.* CoMatch [34] and SimMatch [66]) have been proposed to jointly leverage both class-level and instance-level consistencies and greatly improve the SSL performance, especially in the case when only very limited labeled data is available. There are also many works [6, 5] exploring the two-stage framework (*i.e.* pre-training followed by fine-tuning with SSL algorithms like FixMatch), which shows the importance of the self-supervised pretrain to the SSL tasks.

3. Methodology

3.1. Problem Formulation

Similar to [34, 66], we define a batch of labeled data as $\mathcal{X} = \{x_b : b \in (1, ..., B)\}$, and a batch of unlabeled data as $\mathcal{U} = \{u_b : b \in (1, ..., \mu B)\}$, where μ is the coefficient of unlabeled batch size. Then, a neural network encoder will extract the feature h from the augmented images. We use h^w_b and h^s_b to denote the feature extracted from the weakly and strongly augmented images, respectively. Then, we use $W \in$ to represent a linear layer followed by a SoftMax function that transforms the feature to a class probability *i.e.* $p = \text{SoftMax}(W^T h)$. Finally, we adopt the $g(\cdot)$ to denote a two-layer non-linear projection head which maps the feature to a low-dimensional representation vector *i.e.* $z = g(h)$. By following the common setting as in [9, 24, 66, 34], the representation z will be normalized to a unit vector by default. Note that the reason we use z as the representation instead of h is that we need to store a large number of samples as in [24]. The dimension of z is much smaller than h which makes the training more efficient. For example, h is 2048-D and z is 128-D for ResNet-50.

For labeled data, we perform standard supervised learning with the cross-entropy loss, which can be represented by Eq. (1), where y_b is the one-hot ground truth label,

$$\mathcal{L}_s = -\frac{1}{B}\sum_{b=1}^{B} y_b \log p^w_b. \tag{1}$$

3.2. Consistency Regularization with Graph

SimMatchV2 considers the augmented view of a sample as a node in the graph, which consists of a node label p (class prediction for unlabeled data and ground truth for labeled data) and the representation z. Different nodes are connected by edges measured by the similarity of the representations. We use strong/weak node to name the node from strongly/weakly augmented views.

Given a graph with K nodes, we have the node labels $\{p_k : k \in (1, ..., K)\}$ and the corresponding representations $\{z_k : k \in (1, ..., K)\}$. In our implementation, the graph nodes will be stored in a memory bank as in [24, 66, 34]. We can store both labeled and unlabeled nodes. The node label will be the ground truth for labeled data and p for unlabeled data. Suppose we have an unlabeled weak node with the representation z^w_b. We use $e^w_{b,i}$ to denote the edge between the z^w_b and the i^{th} node on the graph, which can be expressed by Eq. (2). t is a temperature parameter that controls the sharpness of the similarity distribution. We also use e^w_b to denote edges between z^w_b and all the nodes on the graph (*i.e.* $e^w_{b,i}$ is a scalar and $e^w_b \in R^{1\times K}$).

$$e^w_{b,i} = \frac{\exp(z^w_b \cdot z_i/t)}{\sum_{k=1}^{K}\exp(z^w_b \cdot z_k/t)} \tag{2}$$

Node-Node Consistency. *(Weak Node vs. Strong Node.)* The node-node consistency aims to maintain the consistency of the label between the strong node and weak node of the same instance. Such consistency has been proposed in many previous works [3, 2, 43]. In this work, we have renamed it and integrated it into our graph-based framework. Concretely, we follow the design in previous works; given unlabeled data, the node-node consistency can be expressed by Eq. (3), where τ is the confidence threshold.

$$\mathcal{L}_{nn} = -\frac{1}{\mu B}\sum_{b=1}^{\mu B} \mathbb{1}(\max p^w_b > \tau)\; p^w_b \log p^s_b. \tag{3}$$

Node-Edge Consistency. *(Weak Node vs. Strong Edge.)* The purpose of the node-edge consistency is to construct an edge e_b^s by using the representation z_b^s from the strong node, then use the edge e_b^s as the weight to aggregate the node label from the graph. We expect the aggregated node label can be consistent with the label p_b^w from the weak node. Finally, the node-edge consistency will be constrained by the cross-entropy loss between p_b^w and the aggregated label, which can be expressed by Eq. (4).

$$\mathcal{L}_{ne} = -\frac{1}{\mu B}\sum_{b=1}^{\mu B} p_b^w \log(\sum_{i=1}^{K} p_i \; e_{b,i}^s) \tag{4}$$

Note that the aggregated label $\sum_{i=1}^{K} p_i e_{b,i}^s$ does not include the node label p_b^s and p_b^w from itself. The gradient will only be applied on $e_{b,i}^s$.

Edge-Edge Consistency. *(Weak Edge vs. Strong Edge.)* The edge-edge consistency can be regarded as a simple extension of the node-node consistency. Basically, the node-node consistency aims to let the model predict invariant labels for the weak and strong nodes of the same instances. Similarly, the edges should also be invariant with respect to the strong and weak nodes of the same instance. Thus, the edge-edge consistency can be formulated as the cross-entropy loss between the similarity distribution from different augmented views, which can be written as Eq. (5).

$$\mathcal{L}_{ee} = -\frac{1}{\mu B}\sum_{b=1}^{\mu B}\sum_{i=1}^{K} e_{b,i}^w \log e_{b,i}^s \tag{5}$$

Edge-Node Consistency. *(Weak Edge vs. Strong Node.)* This is somewhat similar to reversed operation to the node-edge consistency. Concretely, node-edge consistency requires the aggregated label $\sum_{i=1}^{K} p_i e_{b,i}^s$ from the strong node to be consistent with the label p_b^w from the weak node. In contrast, edge-node consistency requires the label p_b^s from the strong node to be consistent with the aggregated label $\sum_{i=1}^{K} p_i e_{b,i}^w$ from the weak node. Given an unlabeled weak node which has z_b^w and p_b^w. Then, we concatenate the unlabeled node with the graph to get $Z = [z_b^w, z_1, ..., z_k]$, $Y = [p_b^w, p_1, ..., p_k]$. Next, we use the representation matrix Z to obtain the pairwise edge matrix A (a.k.a. affinity matrix) in the same way as Eq. (2). Note that the difference between e_b and A is that e_b only calculates the edges between the unlabeled data with respect to the node in the graph ($e_b^w \in R^{1\times K}$) but A also contain the edge information within the graphs. ($A \in R^{(1+K)\times(1+K)}$). We also want to emphasize that we will make A a hollow matrix (*i.e.* $A_{i,j} = 0$ if $i == j$) before normalizing it to avoid the self-loop in the graph. Here, we adopt the transductive inference [69] for aggregation:

$$\begin{aligned} Y_\phi &= \alpha \cdot A Y_{\phi-1} + (1-\alpha)\cdot Y_0 \\ &= (\alpha\cdot A)^{\phi-1} Y_0 + (1-\alpha)\sum_{i=0}^{\phi-1}(\alpha\cdot A)^i Y_0, \end{aligned} \tag{6}$$

where ϕ is the number of iterations that the label has been propagated, Y_0 is the initial state $[p_b^w, p_1, ..., p_k]$, and α is a parameter in $(0, 1)$ controlling the weight of the aggregated label and the initial state Y_0. The node label will be propagated multiple times until it fully converges. When $\phi \to \infty$, we have $(\alpha \cdot A)^{\phi-1} = 0$ since $0 < \alpha < 1$. We can also make an approximation such that $\lim_{\phi\to\infty}\sum_{i=0}^{\phi-1}(\alpha\cdot A)^i = (I - \alpha\cdot A)^{-1}$, where I is the identity matrix. Thus, the converged Y_∞ can be derived as:

$$Y_\infty = (1-\alpha)(I-\alpha\cdot A)^{-1} Y_0 \tag{7}$$

Finally, the converged pseudo-label of the unlabeled data can be obtained from $Y_\infty[0]$ (*i.e.* the first row vector of Y_∞). Note that if we take $\alpha = 1$, then $Y_1[0]$ is identical to $\sum_{i=1}^{K} p_i e_{b,i}^w$. Since $Y_\infty[0]$ contains both information from the node label and aggregated label. Thus, we can directly use it to replace the original p_b^w in Eq. (3). Moreover, we should notice an inverse operator in Eq. (7), which is computationally very expensive for a large matrix. Thus, to reduce the complexity, we will only take the Top-N nearest node from Z and Y instead of using the whole graph.

Feature Normalization. As we have presented in Eq. (3), one of the training objectives is to minimize the predictions between the weakly and strongly augmented views. However, significant gaps exist between the feature norms of different augmented views. Also, from $p = \text{SoftMax}(W^T h)$, we can observe that the norm of h does not affect the prediction (*i.e.* $argmax(p)$). Thus, minimizing the gaps between the feature norm is unnecessary and brings additional complexity to the training. To address this issue, we simply apply a layer normalization [56] on the feature h, which can be expressed by Eq .(8). Finally, the linear layer W and non-linear projection head $g(\cdot)$ will be applied on top of the $\widehat{h}$,

$$\widehat{h} = \text{LayerNorm}(h). \tag{8}$$

Finally, the overall objective of SimMatchV2 can be expressed by Eq. (9), where λ_{nn}, λ_{ee}, and λ_{ne} are the balancing factors that control the weights of the different losses.

$$\mathcal{L} = \mathcal{L}_s + \lambda_{nn}\mathcal{L}_{nn} + \lambda_{ne}\mathcal{L}_{ne} + \lambda_{ee}\mathcal{L}_{ee} \tag{9}$$

4. Experiments

4.1. Classical SSL Benchmark

We will first conduct our experiments on the classic SSL benchmark which is widely adopted by most SSL papers.

Table 1: Error rate (%) and Rank on the classical SSL benchmark. The error rate and standard deviation are reported based on three runs. The experimental settings in this table are all based on the USB [51] code base. All results of other methods are directly copied from the USB [51] benchmark paper. The results for fully-supervised training on STL-10 is not provided since it contains an unlabeled set with unknown labels.

Dataset	CIFAR-10			CIFAR-100			SVHN			STL-10			Friedman	Final	Mean
# Label	40	250	4000	400	2500	10000	40	250	1000	40	250	1000	rank	rank	error rate
Fully-Supervised		4.57±0.06			18.96±0.06			2.14±0.01			-		-	-	-
Supervised	77.18±1.32	56.24±3.41	16.10±0.32	89.60±0.43	58.33±1.41	36.83±0.21	82.68±1.91	24.17±1.65	12.19±0.23	75.4±0.66	55.07±1.83	35.42±0.48	-	-	-
Π-Model [32]	76.35±1.69	48.73±1.07	13.63±0.07	87.67±0.79	56.40±0.69	36.73±0.05	80.07±1.22	13.46±0.61	6.90±0.22	74.89±0.57	52.20±2.11	31.34±0.64	14.17	15	48.20
Pseudo-Labeling [33]	75.95±1.86	51.12±2.91	15.32±0.35	88.18±0.89	55.37±0.48	36.58±0.12	75.98±5.36	16.47±0.59	9.37±0.42	74.02±0.47	51.90±1.87	30.77±0.04	14.08	14	48.42
Mean Teacher [46]	72.42±2.10	37.56±4.90	8.29±0.10	79.96±0.53	44.37±0.60	31.39±0.11	49.34±7.90	3.44±0.02	3.28±0.07	72.90±0.83	49.30±3.09	27.92±1.65	11.75	12	40.01
VAT [41]	78.58±2.78	28.87±3.62	10.90±0.16	83.60±3.21	46.20±0.80	32.14±0.31	84.12±3.18	3.38±0.12	2.87±0.18	73.33±0.47	57.78±1.47	40.98±0.96	12.92	13	45.23
MixMatch [3]	35.18±3.87	13.00±0.80	6.55±0.05	64.91±3.34	39.29±0.13	27.74±0.27	27.77±5.43	4.69±0.46	3.85±0.28	49.84±0.58	32.05±1.16	20.17±0.67	11.00	11	27.09
ReMixMatch [2]	8.13±0.58	6.34±0.22	4.65±0.09	41.60±1.48	25.72±0.07	20.04±0.13	16.43±13.77	5.65±0.35	5.36±0.58	27.87±3.85	11.14±0.52	6.44±0.15	7.75	10	14.95
UDA [55]	10.01±3.34	5.23±0.08	4.36±0.09	45.48±0.37	27.51±0.28	23.12±0.45	5.28±4.02	**1.93±0.03**	**1.94±0.02**	40.09±4.03	10.11±1.15	6.23±0.28	6.42	8	15.12
FixMatch [3]	12.66±4.49	4.95±0.10	4.26±0.01	45.38±2.07	27.71±0.42	22.06±0.10	3.37±1.01	1.97±0.01	2.02±0.03	38.19±4.76	8.64±0.84	5.82±0.06	5.29	5	14.75
Dash [57]	9.29±3.28	5.16±0.28	4.36±0.10	47.49±1.05	27.47±0.38	21.89±0.16	5.26±2.02	2.01±0.01	2.08±0.09	42.00±4.94	10.50±1.37	6.30±0.49	6.83	9	15.32
CoMatch [34]	6.51±1.18	5.35±0.14	4.27±0.12	53.41±2.36	29.78±0.11	22.11±0.22	8.20±5.32	2.16±0.04	2.01±0.04	**13.74±4.20**	7.63±0.94	5.71±0.08	5.75	6	13.41
CRMatch [18]	13.62±2.62	**4.61±0.17**	**3.65±0.04**	37.76±1.45	**24.13±0.16**	**19.89±0.23**	**2.60±0.77**	1.98±0.04	1.95±0.03	33.73±1.17	14.87±5.09	6.53±0.36	4.08	2	13.78
FlexMatch [62]	5.29±0.29	4.97±0.07	4.24±0.06	40.73±1.44	26.17±0.18	21.75±0.15	5.42±2.83	8.74±3.32	7.90±0.30	29.12±5.04	9.85±1.35	6.08±0.34	5.92	7	14.19
AdaMatch [4]	5.09±0.21	5.13±0.05	4.36±0.05	37.08±1.35	26.66±0.33	21.99±0.15	6.14±5.35	2.13±0.04	2.02±0.05	19.95±5.17	8.59±0.43	6.01±0.02	4.79	3	12.18
SimMatch [66]	5.38±0.01	5.36±0.08	4.41±0.07	39.32±0.72	26.21±0.37	21.50±0.11	7.60±2.11	2.48±0.61	2.05±0.05	16.98±4.24	8.27±0.40	5.74±0.31	5.25	4	12.10
SimMatchV2 (Ours)	**4.90±0.16**	5.04±0.09	4.33±0.16	**36.68±0.86**	26.66±0.38	21.37±0.20	7.92±2.80	2.92±0.81	2.85±0.91	15.85±2.62	**7.54±0.81**	**5.65±0.26**	**4.00**	**1**	**11.81**

There are four datasets will be used in this experiment.

- CIFAR-10 [31] consists of 60K 32x32 color images in 10 classes, with 6k images per class. The training and test set contains 5K and 1k samples, respectively. We conduct three different experiments on 40, 250, and 4,000 randomly selected images from the training set as the labeled data and use the rest of the training set as the unlabeled data.

- CIFAR-100 [31] is a more complex version of CIFAR-10, which contains 100 classes with 600 images each. There are 500 training images and 100 testing images per class. The experimental setting are similar to CIFAR-10 above except using 400, 2,500, and 10K labeled data for training.

- SVHN [42] is a real-world image dataset that contains 32x32 cropped digits from house numbers in Google Street View images. It contains 73,257 training images, with 26,032 testing images and 531,131 extra images. We merge the training and extra set and randomly select 40, 250, and 1,000 from it as the label data to conduct the experiment.

- STL-10 [14] consists of 5K labeled images, 100K unlabeled images, and 8,000 test images. We will randomly select 40, 250, and 1,000 from the 5K images; the rest images will be combined with 100K as the unlabeled data.

Implementation Details. Our implementation is mainly based on the USB [51] code base for a fair comparison. Most of the hyper-parameters are inherited from [43]. Specifically, we adopt WRN-28-2 [60] for CIFAR-10 and SVHN, WRN-28-8 for CIFAR-100, and WRN-37-2 for STL-10. We use an SGD optimizer with Nesterov momentum [45] for all experiments. The weight decay will be set to $5e^{-4}$ for most experiments except CIFAR-100, for which we set it as $1e^{-3}$. For the learning rate schedule, we use a cosine learning rate decay [39], which adjusts the learning rate to $0.03 \cdot cos(\frac{7\pi s}{16S})$ where s is the current training step, and S is the total number of training steps. The model will be optimized for 2^{20} steps. We use 64 labeled data and 448 unlabeled data for each batch. All experiments will be conducted on a single GPU. The weak and strong augmentation policies are also directly inherited from the USB codebase. Note that the final performance will be reported based on the exponential moving average of the model with a 0.999 decay rate. For SimMatchV2 specific hyper-parameters, we set $\lambda_{nn} = 1, \lambda_{ne} = 1, \lambda_{ee} = 1, t = 0.1, \alpha = 0.1, \tau = 0.95, n = 8$. ($n$ is the Top-N nearest node for Eq. (7)) These hyper-parameters are kept unchanged for all experiments. We maintain two memory banks for storing both unlabeled nodes and labeled nodes. The unlabeled bank will be used for calculating the Eq. (4) and Eq. (5). The labeled bank will be used for computing the Eq. (7).

Performance on Classical SSL Benchmark. The overall results are reported in Table 1. We compare our SimMatchV2 against 14 SSL algorithms. As in Table 1, it is very hard for an algorithm to get the best performance on all settings. Many algorithms already meet or exceed the supervised baseline in some settings (CIFAR-10 and SVHN); the performance difference would not be significant in this case. For a more rigorous and fair comparison, we follow the steps from USB [51], which adopts the Friedman rank [19, 20] as the final evaluation metric. We also report the mean error rate for all settings. As can be seen, our SimMatchV2 achieves the best performance on 4 out of 12 settings and also has the highest Friedman rank and lowest mean error rate. CRMatch [18] also achieves a very good performance that ranked second in the final rank. It has the best performance on 5 out of the 12 settings, but it is less stable when the number of labeled data is limited, and the mean error rate is much worse than SimMatchV2. AdaMatch [4] and SimMatch [66] has a more stable performance under various conditions and they have a very close mean error rate compared to SimMatchV2. However, these two methods did not achieve any best results in this experiment, leading to a worse Friedman rank. Overall, the highest Friedman rank and lowest mean error rate for SimMatchV2 should demonstrate superiority in terms of both performance and stability.

4.2. Experiments on ImageNet

Next, we evaluate our SimMatchV2 on the large-scaled ImageNet dataset [15], which contains 1,000 categories

Table 2: Top-1 an Top-5 Accuracy with ResNet-50 on ImageNet with 1% and 10% settings. Note that the underline in the parameters - inference columns denotes the methods which require more parameters than standard ResNet-50 during the inference stage.

Pre-training Algorithm	Semi-supervised Algorithm	Pre-training Epochs	Semi-supervised Epochs	Paramters train	Paramters inference	Top-1 Label fraction 1%	Top-1 Label fraction 10%	Top-5 Label fraction 1%	Top-5 Label fraction 10%
None	VAT+EntMin. [41, 21, 61]	-	∼100	25.6M	25.6M	-	-	47.0	83.4
	S4L-Rotation [61]	-	∼200	25.6M	25.6M	-	-	53.4	83.8
	UDA [55]	-	∼530	25.6M	25.6M	-	68.8	-	88.8
	FixMatch [43]	-	∼300	25.6M	25.6M	-	71.5	-	89.1
	FixMatch-EMAN [6]	-	∼300	25.6M	25.6M	-	72.8	-	90.3
	SsCL [64]	-	∼800	28.0M	25.6M	60.2	72.1	82.8	90.9
	CoMatch [34]	-	∼400	28.0M	25.6M	66.0	73.6	86.4	91.6
	SimMatch [66]	-	∼400	28.0M	25.6M	67.2	74.4	87.1	91.7
PCL [35]	Fine-tune	∼200	∼20	23.8M	25.6M	-	-	75.3	85.6
SimCLR [9]		∼1000	∼60	28.0M	25.6M	48.3	65.6	75.5	87.8
SimCLR V2 [10]		∼800	∼60	32.2M	29.8M	57.9	68.4	82.5	89.2
BYOL [22]		∼1000	∼50	35.1M	25.6M	53.2	68.8	78.4	89.0
SwAV [7]		∼800	∼20	28.0M	25.6M	53.9	70.2	78.5	89.9
WCL [65]		∼800	∼60	32.2M	29.8M	65.0	72.0	86.3	91.2
MoCo V2 [11]	Fine-tune	∼800	∼20	28.0M	25.6M	49.8	66.1	77.2	87.9
	FixMatch-EMAN [6]	∼800	∼300	28.0M	25.6M	61.4	73.9	82.1	91.0
PAWS [7]	Fine-tune	∼300	∼50	36.1M	29.8M	66.5	75.5	-	-
None	**SimMatchV2 (Ours)**	-	**∼100**	**28.0M**	**25.6M**	**69.9**	**74.8**	**88.7**	**91.7**
		-	**∼300**	**28.0M**	**25.6M**	**71.9**	**76.2**	**90.0**	**92.2**

with 1.2M training images and 50K validation images. Concretely, we follow previous works [34, 66, 1, 6] to take 1% and 10% samples from the training images as our labeled data and use the rest samples as the unlabeled data. To avoid the effects of different selections on the labeled data, we follow [9, 10, 6, 1, 65] to adopt a pre-defined labeled / unlabeled splits [1] that has been adopted in most SSL papers.

Implementation Details. For ImageNet experiments, most of our hyper-parameter and settings are directly inherited from [66]. Specifically, we use ResNet-50 [25] as our default backbone and use the SGD with Nesterov momentum as the optimizer. We set the weight decay to 0.0001 during the training. For the learning rate, we warm up the model for 5 epochs until it reaches 0.03, and it will be decayed to 0 with the cosine scheduler [39]. By following [6, 66], we use 64 labeled data and 320 unlabeled data in each batch; the data will be evenly distributed on 8 GPUs. For the hyper-parameter settings, we set $\lambda_{nn} = 10, \lambda_{ne} = 10, \lambda_{ee} = 10, t = 0.1, \alpha = 0.1, \tau = 0.7, n = 128$ for 1% setting, and we use $\lambda_{nn} = 5, \lambda_{ne} = 5$ for 10% since we found the performance is slightly better. For strong augmentation, we adopt a multi-crop strategy as in [68], which contains five crops with different sizes (224x224, 192x192, 160x160, 128x128, and 96x96); these crops will be augmented with the policy in [8]. For weak augmentation, we follow the same strategy in [66, 34].

Performance on ImageNet. The performance has been shown in Table 2. With 100 epochs of training, SimMatchV2 achieves 70.4% and 74.8% Top-1 accuracy with 1% and 10% labeled data, which already surpass the performance of CoMatch [34], and SimMatch [66] with 400 epochs. With 300 epochs of training, SimMatchV2 achieves 72.1% and 76.2% Top-1 accuracy, which improves absolute +5.2% and +0.7% over PAWS [1]. As we can observe that the improvement of 10% setting is much lesser than 1% setting. We doubt that this is because the performance of 76.2% is already very close to the fully supervised setting (76.5%), which should be the upper bound of the SSL tasks. Most importantly, our SimMatchV2 established a new state-of-the-art performance without any pre-training.

[1] https://www.tensorflow.org/datasets/catalog/imagenet2012_subset

Table 3: Runtime Comparison. MC denotes whether we use the multi-crop strategy. The experiments are performed on 1% ImageNet with 100 epochs of training. ↓ indicates lower is better and ↑ means higher is better.

Method	MC	GPU Memory ↓	GPU days ↓	1% Top-1 ↑
CoMatch [34]	✗	81G	10.3	61.1
SimMatch [66]	✗	40G	7.7	61.2
SimMatchV2	✗	**42G**	**7.8**	**63.7**
PAWS [1]	✓	1048G	22.6	63.8
SimMatchV2	✓	**83G**	**14.7**	**69.9**

Efficiency Comparison. Next, We also compare the training efficiency for different methods. The experiments are conducted with 8 Nvidia GPUs except for PAWS [1], which requires at least 64 GPUs to run. We use mixed precision training for all methods. Note that CoMatch [34] and SimMatch [66] have their own labeled / unlabeled splits, which makes the comparison a bit unfair. In this experiment, we adopt their officially released codebase to reproduce the performance on the same splits.

We show the training statistics in Table 3. When we omit the multi-crops, the computational cost of SimMatchV2 is slightly higher compared to SimMatch [66] (*i.e.* +2G memory, +0.1 GPU days), and it is much more efficient than CoMatch [34]. We can also observe that SimMatchV2 has much better performance (+2.5%) than CoMatch [34] and SimMatch [66] even without the multi-crops. Moreover, when the multi-crops are included, the training time of SimMatchV2 is nearly doubled (7.8 days → 14.7 days), but it is still only 65% of PAWS [1] (14.7 days / 22.6 days). The extra training cost of PAWS [1] is from the support

set containing 960x7 labeled data per batch. Most importantly, our SimMatchV2 achieves similar performance with PAWS (63.7% vs. 63.8%) [1] with only 35% training time (7.8 days / 22.6 days), and has an absolute improvement of +6.6% with 65% training time.

Table 4: More experiments on different architectures. Note that we do not use any pre-training in this experiment.

Method	Arch	Params	Epochs	1%	10%
Semiformer [52]	ViT-S + Conv	40M	600	-	75.5
ProbPseudo Mixup [5]	ViT-Small	22M	100	-	70.9
SimMatchV2	**ViT-Small**	**22M**	**100**	**63.7**	**75.9**
ProbPseudo Mixup [5]	ViT-Base	86M	100	-	73.5
SimMatchV2	**ViT-Base**	**86M**	**100**	**65.3**	**76.8**
ProbPseudo Mixup [5]	ConvNext-T	28M	100	-	74.1
SimMatchV2	**ConvNext-T**	**28M**	**100**	**71.3**	**77.8**
ProbPseudo Mixup [5]	ConvNext-S	50M	100	-	75.1
SimMatchV2	**ConvNext-S**	**50M**	**100**	**72.3**	**78.4**

Working with More Advanced Architectures. With the recent successes in model architecture design, many more powerful models have been proposed, especially the vision transformer and its variant [16, 48, 49, 38, 37]. Thus, we further apply our SimMatchV2 on ViT-Small [48], ViT-Base [16], ConvNext-T [38], and ConvNext-S [38] to show the generality on different architectures. In this experiment, we follow [5] to use 128 labeled data and 640 unlabeled per batch. We also adopt the AdamW [40] optimizer with the learning rate of $1.25e^{-4}$ and weight decay of 0.1. We present the results in Table 4. As we can see, SimMatchV2 with a better model can generally produce a better result. The exploration of cooperating with more advanced SSL pipeline [5] and stronger data augmentations (*e.g.* ProbPseudo Mixup [5]) will be left as a future work since it requires more hyper-parameter turnings and more sophisticated designs, which is not the focus of this paper.

4.3. Experiments on Semi-iNat 2021

Table 5: Accuracy for real-world dataset - Semi-iNat 2021.

Method	Semi-iNat 2021			
	With Random Init		With MoCo Init	
	Top1	Top5	Top1	Top5
Supervised	19.09	35.85	34.96	57.11
MixMatch [3]	16.89	30.83	-	-
+CCSSL [58]	19.65	35.09	-	-
FixMatch [43]	21.41	37.65	40.30	60.05
+CCSSL [58]	31.21	52.25	41.28	64.30
CoMatch [34]	20.94	38.96	38.94	61.85
+CCSSL [58]	24.12	43.23	39.85	63.68
SimMatchV2 (Ours)	**43.05**	**63.81**	**53.13**	**73.40**

Finally, we also test the performance of SimMatchV2 on Semi-iNat 2021 [44], which contains 810 different categories with 9,721 labeled data, 313,248 unlabeled data, and 4,050 validation data. Unlike the dataset we used in previous experiments, Semi-iNat 2021 is designed to expose some challenges encountered in a real-world scenario, which contains highly similar classes, class imbalance, and domain mismatch between the labeled and unlabeled data. Such a more realistic dataset should be a better benchmark to reflect the performance of the different methods.

In this experiment, most of the hyper-parameters and implementation details are the same as our ImageNet setting, except that we do not use the multi-crops strategy for fair comparison. The model will be optimized for 512×1024 steps as in [58]. We also follow the setting in [58] to conduct two experiments to report the training from scratch performance and training from pre-trained model performance.

We show the results in Table 5. Our SimMatchV2 achieves 43.05% Top-1 accuracy when training from scratch, which is +11.84% better than the state-of-the-art method (FixMatch + CCSSL). When the MoCo pre-trained model is adopted, our SimMatchV2 achieves a 53.13% Top-1 accuracy, which is +10.08% better than the from-scratch performance, and +11.85% better than FixMatch + CCSSL. We believe this experiment will further demonstrate the superiority of SimMatchV2.

5. Ablation study

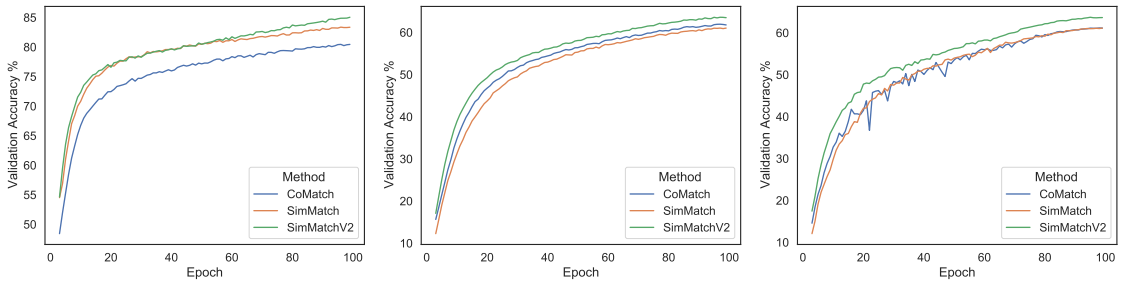
In this section, we will empirically study our SimMatchV2 based on various conditions and show the effect of each component and hyper-parameter sensitivities of our methods. For all experiments in this section, we adopt the ResNet-50 as our backbone and train the model on ImageNet for 100 epochs on the 1% labeled data. The multi-crops are not used by default unless we mentioned.

Training Dynamics. We first show the training dynamics of SimMatchV2 compared to SimMatch [66] and CoMatch [34]. We visualize the pseudo-labels accuracy, unlabeled data accuracy, and validation accuracy in Figure 3. As we can observe that the pseudo-labels from SimMatchV2 have a higher quality than SimMatch [66], especially in the late stage of the training. Both of these two methods generate better pseudo-labels than CoMatch [34]. We can also notice that SimMatchV2 consistently has better accuracy for the unlabeled data and validation data, which shows the advantages of our method.

Table 6: Effect of each component in SimMatchV2. N and E denote the node and edge, respectively. N-E means the node-edge consistency and vice versa. Feat norm means feature normalization.

N-N	N-E	E-E	E-N	Feat Norm	Top-1	Improvement
✓					52.9	-
✓	✓				60.0	+7.1
✓	✓	✓			60.7	+0.7
✓	✓	✓	✓		62.2	+1.5
✓	✓	✓	✓	✓	63.7	+1.5

Effect of Each Components. In table 6, we show the effect of each design in SimMatchV2. Basically, if only node-node consistency is adopted, the setting is equivalent to the FixMatch [43] with distribution alignment [2], which is our baseline in the first row of the table; Next, we can see that the node-edge, edge-node, and edge-node consistencies give +7.1%, +0.7%, +1.5% improvements, respectively. Finally, the feature normalization also boosts the performance



(a) Pseudo-Labels Accuracy (b) Unlabeled Data Accuracy (c) Validation Accuracy

Figure 3: In this Figure, we compare the training dynamics of SimMatchV2 with SimMatch [66], and CoMatch [34] from 3 different perspectives. a) The accuracy of p^w that pass the threshold τ. b) The accuracy of p^w regardless of the threshold. c) The accuracy on the validation data. All experiments are in 1% ImageNet setting with 100 epochs training.

+1.5%. This experiment should demonstrate that these different designs are all very effective for SSL tasks.

Table 7: Effect of memory bank size on the performance. 5k - 163K are setting that we use unlabeled bank; labeled bank means we use all labeled bank (All labeled data 12K for 1% ImageNet).

Bank Size	5K	10K	40K	82K	163K	Labeled Bank (12K)
Top-1	63.1	63.2	63.5	**63.7**	63.6	63.2

Node-Edge and Edge-Edge Consistency. We will study the node-edge and edge-edge consistency together since the memory bank size is their primary factor. The experimental results are shown in Table 7. We can observe that when unlabeled data is used, the performance will be slightly increased with a larger memory bank, and the best memory bank size is 82K. We can also notice that we get slightly lower performance with 12K labeled data. Therefore, we decide to use the unlabeled data for node-edge and edge-edge consistency.

Table 8: Effect of Top-N to Edge-Node Consistency. (We use labeled bank for this experiment.)

N	8	16	32	64	128	256
Top-1	63.4	63.5	63.5	63.5	**63.7**	63.6

Table 9: Effect of labeled / unlabeled bank and ϕ in Eq. (6) to Edge-Node Consistency.

	Labeled Y_∞	Labeled Y_1	Unlabeled Y_∞	Unlabeled Y_1
Top-1	**63.7**	63.3	62.4	62.2

Edge-Node Consistency. As we have mentioned, it is computationally very expensive to involve the whole memory bank for the edge-node consistency calculation. Thus, we only adopt the Top-N nearest neighbor to propagate the label. We show this ablation study in Table 8. The performance is very robust with the selection of the number of Top-N, the best results come from $N = 128$, and the performance is only slightly decreased -0.3% in the worst case. Note that all experiments in Table 8 are performed with the labeled bank; we also have the ablation study to test the effect of the labeled / unlabeled bank and the ϕ in Eq. (6). In Table 9, we can see that the performance with the labeled bank is much better than the unlabeled bank. Also, the performance with the fully converged Y_∞ is +0.4% better than Y_1, which shows the necessity of our design.

Table 10: Effect of Feature Normalization

	Top-1	Norm Gap ↓	CE Loss ↓
W/o Feat Norm	62.2	1.83	2.75
W/ Feat Norm	**63.7**	**1.31**	**2.63**

Feature Normalization. Finally, we show the effect of feature normalization in Table 10. As we mentioned, feature normalization aims to reduce the norm gaps between the weak and strong augmented views. Thus, to validate this, we calculate the mean of the absolute value of $\|h^s\| - \|h^w\|$ across the whole training process. We name it "norm gaps" in Table 10. With the feature normalization, the norm gaps can be reduced from 1.83 $\rightarrow$ 1.31. We also calculate the mean of the CE loss Eq. (3) (regardless of the threshold, *i.e.* $\tau = 0$) across the training process.

6. Conclusion

In this work, we propose a new semi-supervised learning algorithm - SimMatchV2, which models consistency regularization from the graph perspective. SimMatchV2 provides four different types of consistency by leveraging the idea of message passing in graph theory and also proposes a simple feature normalization to reduce the gaps between the feature norm of different augmented views. The experiments on different benchmarks and various architectures demonstrate state-of-the-art performance. The limitation of this work is the missing exploration of cooperating with the more advanced SSL training pipelines and stronger data augmentations [5] since it needs more experiments to find the optimal setting. We plan to investigate the pre-training setting in our future work.

Acknowledgment

This work was supported in part by the Australian Research Council under Project DP210101859 and the University of Sydney Research Accelerator (SOAR) Prize.

References

[1] Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Armand Joulin, Nicolas Ballas, and Michael Rabbat. Semi-supervised learning of visual features by non-parametrically predicting view assignments with support samples. *arXiv preprint arXiv:2104.13963*, 2021. 1, 6, 7

[2] David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. *arXiv preprint arXiv:1911.09785*, 2019. 1, 2, 3, 5, 7

[3] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. *arXiv preprint arXiv:1905.02249*, 2019. 1, 2, 3, 5, 7

[4] David Berthelot, Rebecca Roelofs, Kihyuk Sohn, Nicholas Carlini, and Alex Kurakin. Adamatch: A unified approach to semi-supervised learning and domain adaptation. *arXiv preprint arXiv:2106.04732*, 2021. 5

[5] Zhaowei Cai, Avinash Ravichandran, Paolo Favaro, Manchen Wang, Davide Modolo, Rahul Bhotika, Zhuowen Tu, and Stefano Soatto. Semi-supervised vision transformers at scale. In *NeurIPS*, 2022. 3, 7, 8

[6] Zhaowei Cai, Avinash Ravichandran, Subhransu Maji, Charless Fowlkes, Zhuowen Tu, and Stefano Soatto. Exponential moving average normalization for self-supervised and semi-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 194–203, 2021. 3, 6

[7] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. 2020. 2, 6

[8] 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. 6

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020. 2, 3, 6

[10] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020. 2, 6

[11] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 2, 6

[12] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021. 2, 3

[13] Ching-Yao Chuang, Joshua Robinson, Lin Yen-Chen, Antonio Torralba, and Stefanie Jegelka. Debiased contrastive learning. In *Advances in neural information processing systems*, 2020. 2

[14] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. volume 15 of *Proceedings of Machine Learning Research*, pages 215–223, Fort Lauderdale, FL, USA, 11–13 Apr 2011. JMLR Workshop and Conference Proceedings. 5

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 1, 5

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 7

[17] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 1

[18] Yue Fan, Anna Kukleva, Dengxin Dai, and Bernt Schiele. Revisiting consistency regularization for semi-supervised learning. *International Journal of Computer Vision*, pages 1–18, 2022. 5

[19] Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, 32(200):675–701, 1937. 5

[20] Milton Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940. 5

[21] Yves Grandvalet, Yoshua Bengio, et al. Semi-supervised learning by entropy minimization. *CAP*, 367:281–296, 2005. 6

[22] 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. 2, 3, 6

[23] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022. 2

[24] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019. 2, 3

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 6

[26] Qianjiang Hu, Xiao Wang, Wei Hu, and Guo-Jun Qi. Adco: Adversarial contrast for efficient learning of unsupervised representations from self-trained negative adversaries. *arXiv preprint arXiv:2011.08435*, 2020. 2

[27] Lang Huang, Shan You, Mingkai Zheng, Fei Wang, Chen Qian, and Toshihiko Yamasaki. Green hierarchical vision transformer for masked image modeling. *Advances in Neural Information Processing Systems*, 35:19997–20010, 2022. 2

[28] Lang Huang, Shan You, Mingkai Zheng, Fei Wang, Chen Qian, and Toshihiko Yamasaki. Learning where to learn in cross-view self-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14451–14460, 2022. 2

[29] Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning. In *Neural Information Processing Systems (NeurIPS)*, 2020. 2

[30] Soroush Abbasi Koohpayegani, Ajinkya Tejankar, and Hamed Pirsiavash. Mean shift for self-supervised learning. *arXiv preprint arXiv:2105.07269*, 2021. 2

[31] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009. 5

[32] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016. 2, 5

[33] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896, 2013. 1, 5

[34] Junnan Li, Caiming Xiong, and Steven CH Hoi. Comatch: Semi-supervised learning with contrastive graph regularization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9475–9484, 2021. 3, 5, 6, 7, 8

[35] Junnan Li, Pan Zhou, Caiming Xiong, and Steven Hoi. Prototypical contrastive learning of unsupervised representations. In *International Conference on Learning Representations*, 2021. 2, 6

[36] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 1

[37] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 7

[38] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022. 7

[39] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 5, 6

[40] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 7

[41] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018. 5, 6

[42] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 5

[43] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*, 2020. 1, 2, 3, 5, 6, 7

[44] Jong-Chyi Su and Subhransu Maji. The semi-supervised inaturalist challenge at the fgvc8 workshop, 2021. 7

[45] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013. 5

[46] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv preprint arXiv:1703.01780*, 2017. 2, 5

[47] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? *arXiv preprint arXiv:2005.10243*, 2020. 2

[48] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021. 7

[49] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 32–42, 2021. 7

[50] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. *arXiv preprint arXiv:2005.10242*, 2020. 2

[51] Yidong Wang, Hao Chen, Yue Fan, Wang Sun, Ran Tao, Wenxin Hou, Renjie Wang, Linyi Yang, Zhi Zhou, Lan-Zhe Guo, Heli Qi, Zhen Wu, Yu-Feng Li, Satoshi Nakamura, Wei Ye, Marios Savvides, Bhiksha Raj, Takahiro Shinozaki, Bernt Schiele, Jindong Wang, Xing Xie, and Yue Zhang. Usb: A unified semi-supervised learning benchmark for classification. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. 5

[52] Zejia Weng, Xitong Yang, Ang Li, Zuxuan Wu, and Yu-Gang Jiang. Semi-supervised vision transformers. In *ECCV*, 2022. 7

[53] Zhirong Wu, Yuanjun Xiong, X Yu Stella, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2

[54] Haoyu Xie, Changqi Wang, Mingkai Zheng, Minjing Dong, Shan You, Chong Fu, and Chang Xu. Boosting semi-supervised semantic segmentation with probabilistic representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 2938–2946, 2023. 2

[55] Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. Unsupervised data augmentation for consistency training. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information*

Processing Systems, volume 33, pages 6256–6268. Curran Associates, Inc., 2020. 5, 6

[56] Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin. Understanding and improving layer normalization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 4

[57] Yi Xu, Lei Shang, Jinxing Ye, Qi Qian, Yu-Feng Li, Baigui Sun, Hao Li, and Rong Jin. Dash: Semi-supervised learning with dynamic thresholding. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11525–11536. PMLR, 18–24 Jul 2021. 5

[58] Fan Yang, Kai Wu, Shuyi Zhang, Guannan Jiang, Yong Liu, Feng Zheng, Wei Zhang, Chengjie Wang, and Long Zeng. Class-aware contrastive semi-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14421–14430, June 2022. 7

[59] Xingyi Yang, Daquan Zhou, Songhua Liu, Jingwen Ye, and Xinchao Wang. Deep model reassembly. In *Advances in Neural Information Processing Systems*, 2022. 1

[60] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 5

[61] Xiaohua Zhai, A. Oliver, A. Kolesnikov, and Lucas Beyer. S4l: Self-supervised semi-supervised learning. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1476–1485, 2019. 6

[62] Bowen Zhang, Yidong Wang, Wenxin Hou, HAO WU, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 18408–18419. Curran Associates, Inc., 2021. 5

[63] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 2

[64] Yuhang Zhang, Xiaopeng Zhang, Robert Qiu, Jie Li, Haohang Xu, Qi Tian, et al. Semi-supervised contrastive learning with similarity co-calibration. *arXiv preprint arXiv:2105.07387*, 2021. 6

[65] Mingkai Zheng, Fei Wang, Shan You, Chen Qian, Changshui Zhang, Xiaogang Wang, and Chang Xu. Weakly supervised contrastive learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10042–10051, October 2021. 6

[66] Mingkai Zheng, Shan You, Lang Huang, Fei Wang, Chen Qian, and Chang Xu. Simmatch: Semi-supervised learning with similarity matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14471–14481, June 2022. 2, 3, 5, 6, 7, 8

[67] Mingkai Zheng, Shan You, Fei Wang, Chen Qian, Changshui Zhang, Xiaogang Wang, and Chang Xu. Ressl: Relational self-supervised learning with weak augmentation. *arXiv preprint arXiv:2107.09282*, 2021. 2

[68] Mingkai Zheng, Shan You, Fei Wang, Chen Qian, Changshui Zhang, Xiaogang Wang, and Chang Xu. Relational self-supervised learning. *arXiv preprint arXiv:2203.08717*, 2022. 2, 6

[69] Dengyong Zhou, Olivier Bousquet, Thomas Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2003. 4