SimMatch: Semi-supervised Learning with Similarity Matching

Mingkai Zheng\textsuperscript{1,2} \hspace{1em} Shan You\textsuperscript{2*} \hspace{1em} Lang Huang\textsuperscript{3} \hspace{1em} Fei Wang\textsuperscript{4} \hspace{1em} Chen Qian\textsuperscript{2} \hspace{1em} Chang Xu\textsuperscript{1}
\hspace{1em} \textsuperscript{1}School of Computer Science, Faculty of Engineering, The University of Sydney \hspace{1em} \textsuperscript{2}SenseTime Research \hspace{1em} \textsuperscript{3}The University of Tokyo \hspace{1em} \textsuperscript{4}University of Science and Technology of China

Abstract

Learning with few labeled data has been a longstanding problem in the computer vision and machine learning research community. In this paper, we introduced a new semi-supervised learning framework, SimMatch, which simultaneously considers semantic similarity and instance similarity. In SimMatch, the consistency regularization will be applied on both semantic-level and instance-level. The different augmented views of the same instance are encouraged to have the same class prediction and similar similarity relationship respected to other instances. Next, we instantiated a labeled memory buffer to fully leverage the ground truth labels on instance-level and bridge the gaps between the semantic and instance similarities. Finally, we proposed the unfolding and aggregation operation which allows these two similarities be isomorphically transformed with each other. In this way, the semantic and instance pseudo-labels can be mutually propagated to generate more high-quality and reliable matching targets. Extensive experimental results demonstrate that SimMatch improves the performance of semi-supervised learning tasks across different benchmark datasets and different settings. Notably, with 400 epochs of training, SimMatch achieves 67.2\%, and 74.4\% Top-1 Accuracy with 1\% and 10\% labeled examples on ImageNet, which significantly outperforms the baseline methods and is better than previous semi-supervised learning frameworks.

1. Introduction

Benefiting from the availability of large-scale annotated datasets and growing computational resources in the last decades, deep neural networks have demonstrated their success on a variety of visual tasks \{19, 20, 22, 23, 27, 38, 50, 58–60\}. However, a large volume of labeled data is very expensive to collect in a real-world scenario. Learning with few labeled data has been a longstanding problem in the computer vision and machine learning research community. Among various methods, semi-supervised learning (SSL) \{12, 46, 53, 67\} serves as an effective solution by dint of the help of massive unlabeled data, and achieves remarkable performance.

A simple but very effective semi-supervised learning method is to pretrain a model on a large-scale dataset and then transfer the learned representation by fine-tuning the pretrained model with a few labeled samples. Thanks to the recent advances in self-supervised learning \{10, 14, 15, 21, 25, 26, 30, 54\}, such pretraining and fine-tuning pipeline have demonstrated its promising performance in SSL. Most self-supervised learning frameworks focus on the design of pretext tasks. For example, instance discrimination \{55\} encourages different views of the same instance to share the same features, and different instances should have distinct features. Deep clustering based methods \{3, 9, 10\} expect different augmented views of the same instance should be classified into the same cluster. However, most of these pretext tasks are designed in a completely unsupervised manner, without considering the few labeled data at hand.

Instead of standalone two-stage pretraining and fine-
tuning, current popular methods directly involve the labeled data in a joint feature learning paradigm with pseudo-labeling [35] or consistency regularization [47]. The main idea behind these methods is to train a semantic classifier with labeled samples and use the predicted distribution as the pseudo label for the unlabeled samples. In this way, the pseudo-labels are generally produced by the weakly augmented views [5, 48] or the averaged predictions of multiple strongly augmented views [6]. The objective will be constructed by the cross entropy loss between a different strongly augmented views and the pseudo-labels. It might also be noted that the pseudo-labels will generally be sharpened or operated by argmax since every instance is expected to be classified into a category. However, when there are only very limited annotated data, the semantic classifier is no longer reliable; applying the pseudo-label method will cause the “overconfidence” issue [13, 63], which means the model will fit on the confident but wrong pseudo-labels, resulting in poor performance.

In this paper, we introduce a novel semi-supervised learning framework, SimMatch, which has been shown in Figure 1. In SimMatch, we bridge both sides and propose to match the similarity relationships of both semantic and instance levels simultaneously for different augmentations. Specifically, we first require the strongly augmented view to have the same semantic similarity (i.e., label prediction) with a weak augmented view; besides, we also encourage a strong augmentation to have the same instance characteristics (i.e., similarity between instances) with the weak one for more intrinsic feature matching. Moreover, different from the previous works that simply regard the predictions of the weakly augmented views as pseudo-labels. In SimMatch, the semantic and instance pseudo-labels are allowed to interact by instantiating a memory buffer that keeps all the labeled examples. In this way, these two similarities can be isomorphically transformed with each other by introducing aggregating and unfolding techniques. Thus the semantic and instance pseudo-labels can be mutually propagated to generate more high-quality and reliable matching targets. Extensive experiments demonstrate the effectiveness of SimMatch across different settings. Our contribution can be summarized as follows:

- We proposed SimMatch, a novel semi-supervised learning framework that simultaneously considers semantics similarity and instance similarity.
- To channel both similarities, we leverage a labeled memory buffer so that semantic and instance pseudo-labels can be mutually propagated with the aggregating and unfolding techniques.
- SimMatch establishes a new state-of-the-art performance for semi-supervised learning. With only 400 epochs of training, SimMatch achieves 67.2% and 74.4% Top-1 accuracy with 1% and 10% labeled examples on ImageNet.

2. Related Work

2.1. Semi-Supervised Learning

Consistency Regularization is a widely adopted method in semi-supervised learning. The main idea is to enforce the model to output a consistent prediction for the different perturbed versions of the same instance. For example, [34, 47] achieved such consistency requirement by minimizing the mean square difference between the predicted probability distribution of the two transformed views. In this case, the transformation could be either domain-specific data augmentations [3, 6, 48], or some regularization techniques in the network (e.g., drop out [49] and random max-pooling [47]). Moreover, [34] also proposed a temporal ensemble strategy to aggregate the predictions of multiple previous networks, which makes the predicted distribution more reliable. Mean Teacher [52] further extends this idea which replaced the aggregated predictions with the output of an exponential moving average (EMA) model.

MixMatch [6], ReMixMatch [5], and FixMatch [48] are three augmentation anchoring based methods that fully leverage the augmentation consistency. Specifically, MixMatch adopts a sharpened averaged prediction of multiple strongly augmented views as the pseudo label and utilizes the MixUp trick [64] to further enhance the pseudo label. ReMixMatch improved this idea by generating the pseudo label with weakly augmented views and also introduced a distribution alignment strategy that encourages the pseudo label distribution to match the marginal distribution of ground-truth class labels. FixMatch simplified these ideas, where the unlabeled images are only retained if the model produces a high-confidence pseudo label. Despite its simplicity, FixMatch achieved state-of-the-art performance among the augmentation anchoring-based methods.

2.2. Self-supervised Pretraining

Apart from the typical semi-supervised learning method, self-supervised and contrastive learning [14, 26, 55] has gained much attention in this research community since fine-tuning the pre-trained model with labeled samples has shown promising classification results, especially SimCLR v2 [15] shows that a big (deep and wide) pre-trained model is a strong semi-supervised learner. Most of the contrastive learning framework adopts the instance discrimination [55] as the pretext task, which defines different augmented views of the same instance as positive pairs, while negative pairs are formed by sampling views from different instances. However, because of the existence of similar samples, treating different instances as negative pairs will result in a class collision problem [2], which is not conducive to the down-
stream tasks (especially classification tasks). Some previous works addressed this issue by unsupervised clustering [9, 10, 37, 65], where similar samples will be clustered into the same class. There are also some other methods designed various negative free pretext tasks [17, 25, 29, 31, 66] to avoid the class collision problem. Both cluster-based methods and negative-free based methods have shown significant improvements for downstream classification tasks.

CoMatch [36] combines the idea of consistency regularization and contrastive learning, where the target similarity of two instances is measured by the similarity between two class probability distributions, which achieves the current state-of-the-art performance on semi-supervised learning. However, it is very sensitive to the hyper-parameters, the optimal temperature and threshold is different for various datasets and settings. Compared to CoMatch, SimMatch is faster, more robust, and has higher performance.

3. Method

In this section, we will first revisit the preliminary work on augmentation anchoring based semi-supervised learning frameworks; then, we will introduce our proposed method SimMatch. After that, the algorithm and the implementation details will also be explained.

3.1. Preliminaries

We define the semi-supervised image classification problem as following. Given a batch of $B$ labeled samples $\mathcal{X} = \{x_b : b \in \{1, ..., B\}\}$, we randomly apply a weak augmentation function (e.g. using only a flip and a crop) $T_w(\cdot)$ to obtain the weakly augmented samples. Then, a convolutional neural network based encoder $F(\cdot)$ is employed to extract the feature information from these samples, i.e. $\mathbf{h} = F(T(x))$. Finally, a fully connected class prediction head $\phi(\cdot)$ is utilized to map $\mathbf{h}_b$ into semantic similarities, which can be written as: $p = \phi(h)$. The labeled samples could be directly optimized by the cross entropy loss with the ground truth labels:

$$L_s = \frac{1}{B} \sum_{y, p} H(y, p) \quad (1)$$

Let us define a batch of $\mu B$ unlabeled samples $\mathcal{U} = \{u_b : b \in \{1, ..., \mu B\}\}$. By following [5, 6], we randomly apply the weak and strong augmentation $T_w(\cdot), T_s(\cdot)$ and using the same processing step as the labeled samples to get the semantic similarities for weakly augmented sample $p^w$ (pseudo label) and strongly augmented sample $p^s$. Then the unsupervised classification loss can be defined as the cross-entropy between these two predictions:

$$L_u = \frac{1}{\mu B} \sum \mathbb{1}(\max DA(p^w) > \tau)H(DA(p^w), p^s) \quad (2)$$

Where $\tau$ is the confidence threshold. Following [48], we only retain the unlabeled samples whose largest class probability in the pseudo-labels are larger than $\tau$. $DA(\cdot)$ stands for the distribution alignment strategy from [5] which balanced the pseudo-labels distribution. We simply follow the implementation from [36] where we maintain a moving-average of $p_{avg}$ and adjust the current $p^w$ with $\text{Normalize}(p^w / p_{avg})$. Please also noted that we do not take the sharpened or one-hot version of $p^w$, $DA(p^w)$ will be directly served as the pseudo-label.

3.2. Instance Similarity Matching

In SimMatch, we also consider the instance-level similarity as we have discussed previously. Concretely, we encourage the strongly augmented view to have a similar similarity distribution with the weakly augmented view. Suppose we have a non-linear projection head $g(\cdot)$ which maps the representation $h$ to a low-dimensional embedding
\( z_b = g(\mathbf{h}_b) \). Following the anchoring based method, we use \( z^w_b \) and \( z^s_b \) to denote the embedding from the weakly and strongly augmented view. Now, let us assume we have \( K \) weakly augmented embeddings for a bunch of different samples \( \{ z_b : k \in (1, ..., K) \} \), we calculate the similarities between \( z^w \) and \( i \)-th instance by using a similarity function \( \text{sim}(\cdot, \cdot) \), which represents the dot product between \( L_2 \) normalized vectors \( \text{sim}(\mathbf{u}, \mathbf{v}) = \mathbf{u}^T \mathbf{v} / \| \mathbf{u} \| \| \mathbf{v} \| \). A softmax layer can be adopted to process the calculated similarities, which then produces a distribution:

\[
q^w_i = \frac{\exp(\text{sim}(z^w_i, z_b)/t)}{\sum_{k=1}^K \exp(\text{sim}(z^w_i, z_k)/t)}
\]  (3)

Where \( t \) is the temperature parameter that controls the sharpness of the distribution. On the other hand, we can calculate the similarities between the strongly augmented view \( z^s \) and \( z_i \) as \( \text{sim}(z^s_i, z_i) \). The resulting similarity distribution can be written as:

\[
q^s_i = \frac{\exp(\text{sim}(z^s_i, z_i)/t)}{\sum_{k=1}^K \exp(\text{sim}(z^s_i, z_k)/t)}
\]  (4)

Finally, the consistency regularization can be achieved by minimizing the difference between \( q^s \) and \( q^w \). Here, we adopt the cross entropy loss, which can be formulated as:

\[
L_{in} = \frac{1}{\mu B} \sum_{q^w, q^s} H(q^w, q^s)
\]  (5)

Please noted that the instance consistency regularization will only be applied on the unlabeled examples. The overall training objective for our model will be:

\[
L_{overall} = L_s + \lambda_u L_u + \lambda_{in} L_{in}
\]  (6)

where \( \lambda_u \) and \( \lambda_{in} \) are the balancing factors that control the weights of the two losses.

### 3.3. Label Propagation through SimMatch

Although our overall training objective also considers the consistency regularization on instance-level, however, the generation of the instance pseudo-labels \( q^w \) are still in a fully unsupervised manner, which is absolutely a waste of the labeled information. To improve the quality of the pseudo-labels, in this section, we will illustrate how to leverage the labeled information on instance-level and also introduce a way that allows semantic similarity and instance similarity to interact with each other.

We instantiated a labeled memory buffer to keep all the annotated examples as we have shown in Figure 2 (red branch). In this way, each \( z_b \) that we used in Eq.(3) and Eq.(4) could be assigned to a specific class. If we interpret the vectors in \( \phi \) as the “centered” class references, the embeddings in our labeled memory buffer can be viewed as a set of “individual” class references.

By given a weakly augmented sample, we first compute the semantic similarity \( p^w_i \in \mathbb{R}^{1 \times K} \) and instance similarity \( q^w \in \mathbb{R}^{1 \times K} \). (Noted that \( L \) is generally much smaller than \( K \) since we need at least one sample for each class.) To calibrate \( q^w \) with \( p^w \), we need to unfold \( p^w \) into \( K \) dimensional space which we denote it as \( p^{w\text{fold}}_i \). We achieved this by matching the corresponding semantic similarity for each labeled embedding:

\[
p_i^{w\text{fold}} = p_j^w, \quad \text{where Class}(q^w_j) = \text{Class}(p^w_i)
\]  (7)

where \( \text{class}(\cdot) \) is the function that returns the ground truth class. Specifically, \( \text{class}(q^w_j) \) represent the label for the \( j \)-th element in memory buffer and \( \text{class}(p^w_i) \) means the \( i \)-th class. Now, we regenerate the calibrated instance pseudo-labels by scaling \( q^w \) with \( p^{w\text{fold}} \), which can be expressed as the following:

\[
\hat{q}^w_i = \frac{q^w_i p^{w\text{fold}}_i}{\sum_{k=1}^K q^w_k p^{w\text{fold}}_k}
\]  (8)

The calibrated instance pseudo-label \( \hat{q} \) will be served as a new target and replace the old one \( q^w \) in Eq.(5). On the other hand, we can also use the instance similarity to adjust the semantic similarity. To do this, we first need to aggregate \( q \) into \( L \) dimensional space which we denote it as \( q^{agg} \). We achieved this by sum over the instance similarities that share the same ground truth labels:

\[
q^{agg}_i = \sum_{j=0}^K \mathbb{1}([\text{class}(p_i^w) = \text{class}(q_j^w)]) q^w_j
\]  (9)

Now, we regenerate the adjusted semantic pseudo-label by smoothing \( p^w \) with \( q^{agg} \), which can be written as:

\[
\hat{p}_i = \alpha p_i^w + (1 - \alpha) q^{agg}_i
\]  (10)

where \( \alpha \) is the hyper-parameter that controls the weight of the semantic and instance information. Similarly, The adjusted semantic pseudo-label will replace the old one \( p_i^w \) in Eq.(2). In this way, the pseudo-label \( \hat{p} \) and \( \hat{q} \) will both contain the semantic-level and instance-level information. As we have shown in Figure 3, when semantic and instance similarities are similar, which means these two distributions agree with the prediction of each other, then the result pseudo-label will be much sharper and produce high confidence for some classes. On the other hand, if these two similarities are different, the result pseudo-label will be much flatter and not contain high probability values. In SimMatch, we adopt the scaling and smoothing strategy for \( \hat{q} \) and \( \hat{p} \) respectively, we also have tried different combination for these two strategies, please see more details in our ablation study section.
augmented samples will be passed into $F_t$. To leverage a student-teacher based framework, we denote it as $F_t$ and $F_s$. When $K$ is large, we follow MoCo [26] to leverage a student-teacher based framework, we denote it as $F_s$ and $F_t$. In this case, the labeled examples and strongly augmented samples will be passed into $F_s$, the weakly augmented samples will be feed into $F_t$ to generate the pseudo-labels. The parameters of $F_t$ will be updated by:

$$F_t \leftarrow mF_t + (1-m)F_s$$

(11)

On the other hand, when $K$ is small, maintain a teacher network is not necessary. We simply adopt the temporal ensemble strategy [34, 55] to smooth the features in memory buffer, which can be written as:

$$z_t \leftarrow mz_{t-1} + (1-m)z_t$$

(12)

In this case, all the samples will be directly pass into the same encoder. The student-teacher version of SimMatch has been illustrated in Algorithm 1.

**Algorithm 1: SimMatch (Student-Teacher)**

**Input:** $x_t$ and $x_u$, a batch of labeled and unlabeled samples. $T_w(\cdot)$ and $T_s(\cdot)$: Weak and strong augmentation function. $F_t$ and $F_s$: Teacher and student encoder. $\phi_t$ and $\phi_s$: teacher and student classifier. $g_t$ and $g_s$: teacher and student projection head. $Q_f$ and $Q_l$: The feature and label memory buffer.

**while** network not converge do

for $i = 1$ to $s$ do

$\mathbf{h}^w = F_t(T_w(x_u))$ \hspace{0.5cm} $\mathbf{h}^s = F_s(T_s(x_u))$

$p^w = DA(\phi_t(\mathbf{h}^w))$ \hspace{0.5cm} $p^s = \phi_s(\mathbf{h}^s)$

$\mathbf{z}^w = g_t(\mathbf{h}^w)$ \hspace{0.5cm} $\mathbf{z}^s = g_s(\mathbf{h}^s)$

$\mathbf{h}_t^i = F_t(T_u(x_t))$ \hspace{0.5cm} $\mathbf{h}_s^i = F_s(T_u(x_t))$

$p^f = \phi_s(\mathbf{h}_s^i)$ \hspace{0.5cm} $z^f = g_t(\mathbf{h}_t^i)$

Compute $q^w$ and $q^s$ by Eq.(3) Eq.(4)

Compute $p^{\text{aug}, \text{fold}}$ and $q^{\text{aug}}$ by Eq.(7) Eq.(9)

Compute $\hat{q}$ and $\hat{p}$ by Eq.(8) Eq.(10)

$L_s = \frac{1}{B} \sum H(y, p_f)$

$L_u = \frac{1}{B} \sum 1(\max \hat{p} > \tau) H(\hat{p}, p^u)$

$L_{in} = \frac{1}{B} \sum H(\hat{q}, q^u)$

$L_{\text{overall}} = L_s + \lambda_u L_u + \lambda_{in} L_{in}$

Optimize $F_s$, $g_s$ and $\phi_s$ by $L_{\text{overall}}$

Momentum Update $F_t$, $g_t$ and $\phi_t$

Update $Q_f$ and $Q_l$ with $z^f$ and $y$

end

**Output:** The well trained model $F_s$ and $g_s$

4. **Experiments**

In this section, we will first test SimMatch on various dataset and settings to shows its superiority, then we will ablation each component to validate the effectiveness of each component in our framework.

4.1. **CIFAR-10 and CIFAR-100**

We first evaluation SimMatch on CIFAR-10 and CIFAR-100 [33] datasets. CIFAR-10 consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. CIFAR-100 is just like the CIFAR-10, except it has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. For CIFAR-10, we randomly sample 4, 25, and 400 samples from the training set as the labeled data and use the rest of the training set as the unlabeled data. For CIFAR-100, we perform the same experiments but use 4, 25, and 100 samples per class.

**Implementation Details.** Most our implementations follows [48]. Specifically, we adopt WRN28-2, and WRN28-8 [61] for CIFAR-10 and CIFAR-100 respectively. We use a standard SGD optimizer with Nesterov momen-
then cosine decay it to 0. We use the same set of hyper-epochs until it reaches the initial learning rate 0.03 and Nesterov momentum. We warm up the model for five ResNet-50 [28] and use a standard SGD optimizer with selected per class for 1% and 10% settings respectively. CoMatch [36], where 13 and 128 labeled samples will be specifically, we test our algorithm on 1% and 10% settings.

**4.2. ImageNet-1k**

We also performed SimMatch on the large-scale ImageNet-1k dataset [20] to show the the superiority. Specifically, we test our algorithm on 1% and 10% settings. We follow the same label generation process as in CoMatch [36], where 13 and 128 labeled samples will be selected per class for 1% and 10% settings respectively.

**Implementation Details.** For ImageNet-1k, we adopt ResNet-50 [28] and use a standard SGD optimizer with Nesterov momentum. We warm up the model for five epochs until it reaches the initial learning rate 0.03 and then cosine decay it to 0. We use the same set of hyper-parameters for both 1% and 10% settings ($\lambda_u = 10, \lambda_{in} = 5, t = 0.1, \alpha = 0.9, \tau = 0.95, \mu = 7, m = 0.7, B = 64, S = 2^{20}$). For distribution alignment, we accumulate the past 32 steps $p^w$ for calculating the moving average $p^w_{avg}$. We adopt the temporal ensemble memory buffer [34] since most settings for these two datasets have a relatively small $K$. For the implementations of the strong and weak augmentations, we strictly follow the FixMatch [48].

**Results.** The result has been reported in Table 1. For baseline, we mainly consider methods TI-Model [34], Pseudo-Labeling [35], Mean Teacher [52], UDA [56], MixMatch [6], ReMixMatch [5], FixMatch [48], and CoMatch [36]. We compute the mean and variance of accuracy when training on 5 different “folds” of labeled data. As we can see that the SimMatch achieves state-of-the-art performance on various settings, especially on CIFAR-100. For CIFAR-10, SimMatch has a large performance gain on 40 labels setting, but the improvements for 250 and 4000 is relatively small. We doubt that this is due to the accuracy of 95% ~ 96% being already quite close to the supervised performance.

**Table 1. Top-1 Accuracy comparison (mean and std over 5 runs) on CIFAR-10 and CIFAR-100 with varying labeled set sizes.**

<table>
<thead>
<tr>
<th>Method</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>40 labels</td>
<td>250 labels</td>
</tr>
<tr>
<td>TI-Model [34]</td>
<td>45.74±3.97</td>
<td>58.99±0.38</td>
</tr>
<tr>
<td>Pseudo-Labeling [35]</td>
<td>-</td>
<td>83.91±0.28</td>
</tr>
<tr>
<td>Mean Teacher [32]</td>
<td>70.95±5.93</td>
<td>95.12±0.18</td>
</tr>
<tr>
<td>UDA [56]</td>
<td>52.46±11.50</td>
<td>93.58±10.0</td>
</tr>
<tr>
<td>MixMatch [6]</td>
<td>80.90±9.64</td>
<td>95.28±0.13</td>
</tr>
<tr>
<td>ReMixMatch [5]</td>
<td>86.19±3.37</td>
<td>95.74±0.05</td>
</tr>
<tr>
<td>FixMatch [48]</td>
<td>86.78±3.75</td>
<td>95.44±0.13</td>
</tr>
<tr>
<td>Dash [57]</td>
<td>93.09±1.39</td>
<td>95.09±0.33</td>
</tr>
<tr>
<td>CoMatch [36]</td>
<td>94.40±1.37</td>
<td>95.16±0.39</td>
</tr>
<tr>
<td>SimMatch (Ours)</td>
<td>62.19±2.21</td>
<td>74.93±0.32</td>
</tr>
</tbody>
</table>

*Note that we use an identical set of hyper-parameters for the learning rate schedule, we use a cosine learning rate decay [40] which adjust the learning rate to $0.03 \cdot \cos(\frac{2 \pi s}{T})$ where $s$ is the current training step, and $T$ is the total number of training steps. We also report the final performance using an exponential moving average of model parameters. Note that we use an identical set of hyper-parameters for both datasets ($\lambda_u = 1, \lambda_{in} = 1, t = 0.1, \alpha = 0.9, \tau = 0.95, \mu = 7, m = 0.7, B = 64, S = 2^{20}$). For distribution alignment, we accumulate the past 32 steps $p^w$ for calculating the moving average $p^w_{avg}$. We adopt the temporal ensemble memory buffer [34] since most settings for these two datasets have a relatively small $K$. For the implementations of the strong and weak augmentations, we strictly follow the FixMatch [48].

**Transfer Learning.** We also evaluate the learned representations on multiple downstream classification tasks. We follow the linear evaluation setup described in [14, 25]. Specifically, we trained an L2-regularized multinomial logistic regression classifier on features extracted from the frozen pretrained network (400 epochs 10% SimMatch), then we used L-BFGS [39] to optimize the softmax cross-entropy objective, and we did not apply data augmentation. We selected the best L2-regularization parameter and learning rate from validation splits and applied it to the test sets. The datasets used in this benchmark are as follows: CIFAR-10 [33], CIFAR-100 [33], Food101 [7], Cars [32], DTD [18], Pets [43], Flowers [42]. The results have been shown in Table 3. As we can see, with only 400 epochs
Table 2. Experimental results on ImageNet with 1% and 10% labeled examples.

<table>
<thead>
<tr>
<th>Self-supervised Pre-training</th>
<th>Method</th>
<th>Epochs</th>
<th>Parameters (train/test)</th>
<th>Top-1 Label fraction</th>
<th>Top-5 Label fraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Pseudo-label [35, 62]</td>
<td>~100</td>
<td>25.6M / 25.6M</td>
<td>-</td>
<td>51.6 / 82.4</td>
</tr>
<tr>
<td></td>
<td>VAT+EntMin. [24, 41, 62]</td>
<td>-</td>
<td>25.6M / 25.6M</td>
<td>-</td>
<td>- / 88.5</td>
</tr>
<tr>
<td></td>
<td>S4L-Rotation [62]</td>
<td>~200</td>
<td>25.6M / 25.6M</td>
<td>-</td>
<td>35.4 / 83.8</td>
</tr>
<tr>
<td></td>
<td>UDA [56]</td>
<td>-</td>
<td>25.6M / 25.6M</td>
<td>-</td>
<td>68.8 / 88.5</td>
</tr>
<tr>
<td></td>
<td>FixMatch [48]</td>
<td>~300</td>
<td>25.6M / 25.6M</td>
<td>-</td>
<td>71.5 / 89.1</td>
</tr>
<tr>
<td></td>
<td>CoMatch [36]</td>
<td>~400</td>
<td>30.0M / 25.6M</td>
<td>66.0 / 73.6</td>
<td>86.4 / 91.6</td>
</tr>
<tr>
<td>PCL [37]</td>
<td>Fine-tune</td>
<td>~200</td>
<td>25.8M / 25.6M</td>
<td>-</td>
<td>75.3 / 85.6</td>
</tr>
<tr>
<td>SimCLR [14]</td>
<td></td>
<td>~1000</td>
<td>30.0M / 25.6M</td>
<td>48.3 / 65.6</td>
<td>75.5 / 87.8</td>
</tr>
<tr>
<td>SimCLR V2 [15]</td>
<td></td>
<td>~800</td>
<td>34.2M / 25.6M</td>
<td>57.9 / 68.4</td>
<td>82.5 / 89.2</td>
</tr>
<tr>
<td>BYOL [25]</td>
<td></td>
<td>~1000</td>
<td>37.1M / 25.6M</td>
<td>53.2 / 68.8</td>
<td>78.4 / 89.0</td>
</tr>
<tr>
<td>SwAV [10]</td>
<td></td>
<td>~800</td>
<td>30.4M / 25.6M</td>
<td>53.9 / 70.2</td>
<td>78.5 / 89.9</td>
</tr>
<tr>
<td>WCL [65]</td>
<td></td>
<td>~800</td>
<td>34.2M / 25.6M</td>
<td>65.0 / 72.0</td>
<td>86.3 / 91.2</td>
</tr>
<tr>
<td>MoCo V2 [16]</td>
<td>Fine-tune</td>
<td>~800</td>
<td>30.0M / 25.6M</td>
<td>49.8 / 66.1</td>
<td>77.2 / 87.9</td>
</tr>
<tr>
<td></td>
<td>CoMatch [36]</td>
<td>~1200</td>
<td>30.0M / 25.6M</td>
<td>67.1 / 73.7</td>
<td>87.1 / 91.4</td>
</tr>
<tr>
<td>MoCo-EMAN [8]</td>
<td>FixMatch-EMAN [8]</td>
<td>~1100</td>
<td>30.0M / 25.6M</td>
<td>63.0 / 74.0</td>
<td>83.4 / 90.9</td>
</tr>
<tr>
<td>None</td>
<td>SimMatch (Ours)</td>
<td>~400</td>
<td>30.0M / 25.6M</td>
<td>67.2 / 74.4</td>
<td>87.1 / 91.6</td>
</tr>
</tbody>
</table>

Table 3. Transfer learning performance using ResNet-50 pretrained with ImageNet. Following the evaluation protocol from [14, 25], we report Top-1 classification accuracy except Pets and Flowers for which we report mean per-class accuracy.

<table>
<thead>
<tr>
<th>Method</th>
<th>Epochs</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
<th>Food-101</th>
<th>Cars</th>
<th>DTD</th>
<th>Pets</th>
<th>Flowers</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised</td>
<td>-</td>
<td>93.6</td>
<td>78.3</td>
<td>72.3</td>
<td>66.7</td>
<td>74.9</td>
<td>91.5</td>
<td>94.7</td>
<td>81.7</td>
</tr>
<tr>
<td>SimCLR [14]</td>
<td>1000</td>
<td>90.5</td>
<td>74.4</td>
<td>72.8</td>
<td>49.3</td>
<td>75.7</td>
<td>84.6</td>
<td>92.6</td>
<td>77.1</td>
</tr>
<tr>
<td>MoCo V2 [16]</td>
<td>800</td>
<td>92.2</td>
<td>74.6</td>
<td>72.5</td>
<td>50.5</td>
<td>74.4</td>
<td>84.6</td>
<td>90.5</td>
<td>77.0</td>
</tr>
<tr>
<td>BYOL [25]</td>
<td>1000</td>
<td>91.3</td>
<td>78.4</td>
<td>75.3</td>
<td>67.8</td>
<td>75.5</td>
<td>90.4</td>
<td>96.1</td>
<td>82.1</td>
</tr>
<tr>
<td>SimMatch (10%)</td>
<td>400</td>
<td>93.6</td>
<td>78.4</td>
<td>71.7</td>
<td>69.7</td>
<td>75.1</td>
<td>92.8</td>
<td>93.2</td>
<td>82.1</td>
</tr>
</tbody>
</table>

Table 4. GPU hours per epoch for different methods. The speed is tested on 8 NVIDIA V100 GPUs.

<table>
<thead>
<tr>
<th>Method</th>
<th>FixMatch</th>
<th>CoMatch</th>
<th>SimMatch (Ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU (Hours)</td>
<td>2.77</td>
<td>2.81</td>
<td>2.34</td>
</tr>
</tbody>
</table>

of training, SimMatch achieves the best performance on CIFAR-10, CIFAR-100, Cars and Flowers datasets which is comparable with BYOL and significantly better than SimCLR, MoCo V2, and supervised baseline. These results further validate the representation quality of SimMatch for classification tasks.

Training Efficiency. Next, we test the actual training speed for FixMatch, CoMatch, and SimMatch. The results is shown in Table 4, SimMatch is nearly 17% faster than FixMatch and CoMatch. In FixMatch, the weakly augmented $U$ will be passed into the online network, which consumes more resources for the extra computational graphs. But in SimMatch, $U$ only needs to be passed into the EMA network, so the computational graph does not need to be retained. Compared with CoMatch which requires two forward passes (strongly and weakly augmented $U$) for the EMA network, SimMatch only requires one pass. Moreover, CoMatch adopts 4 memory banks (258M Memory) to compute the pseudo-label; SimMatch only needs 2 memory banks with 6.4M / 64M Memory for 1% and 10% labels, thus the pseudo-label generation will also be faster.

4.3. Ablation Study

Pseudo-Label Accuracy. Firstly, we would like to show the pseudo-label accuracy of SimMatch. In Figure 4, we visualized the training progress of FixMatch and our method. SimMatch can always generate high-quality pseudo-labels and consistently has higher performance on both unlabeled samples and validation sets.

Temperature. The temperature $t$ in Eq. (4) and Eq. (3) controls the sharpness of the instance distribution. (Noted that $t = 0$ is equivalent to the argmax operation). We present the result of varying different $t$ value in Figure 5a. As can be seen, the best Top-1 accuracy comes from $t = 0.1$, and slightly decreased when $t = 0.07$. This is consist with the most recent works in contrastive learning where $t = 0.1$ is generally the best temperature [10, 11, 14, 15].

Smooth Parameter. We also show the effective of different smooth parameter $\alpha$ Eq. (10) in Figure 5b. Specifically, we sweep over [0.8, 0.9, 0.95, 1.0] for $\alpha$, it clear to see that $\alpha = 0.9$ achieves the best result. Noted that $\alpha = 1.0$ is equivalent to directly take the original pseudo-labels $p^w$ for Eq. (2), which result in 1.8% performance drop.

Label Propagation. Next, we would like to verify the effectiveness of the label propagation. The results has been shown in Table 5. When we remove $\hat{p}$, this is the same case with $\alpha = 1.0$, so we will not discuss this setting further. If we remove $\hat{q}$, which means the projection head will
(b) Unlabeled Samples Accuracy

Figure 4. Visualization of (a) pseudo-labels accuracy - the accuracy of $\hat{p}$ that has higher confidence than threshold, (b) unlabeled samples accuracy - the accuracy of all $\hat{p}$ regardless of the threshold, (c) validation accuracy for FixMatch and SimMatch on 1% and 10% setting.

Table 5. Results of removing scaling and smoothing strategy. (ImageNet-1k 1% - 100 ep)

<table>
<thead>
<tr>
<th>Method</th>
<th>w/o $\hat{p}$</th>
<th>w/o $\hat{q}$</th>
<th>Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top-1</td>
<td>59.9</td>
<td>52.3</td>
<td>61.7</td>
</tr>
</tbody>
</table>

Table 6. Results of different combinations for scaling and smoothing strategy. (ImageNet-1k 1% - 100 ep)

<table>
<thead>
<tr>
<th>$\hat{p}$</th>
<th>$\hat{q}$</th>
<th>Scaling</th>
<th>Smoothing</th>
</tr>
</thead>
<tbody>
<tr>
<td>56.6</td>
<td>59.9</td>
<td>61.7</td>
<td>61.5</td>
</tr>
</tbody>
</table>

Table 7. Results of replacing $L_{in}$ with InfoNCE and SwAV. (ImageNet-1k 1% - 100 ep)

<table>
<thead>
<tr>
<th>Method</th>
<th>InfoNCE</th>
<th>SwAV</th>
<th>SimMatch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top-1</td>
<td>53.5</td>
<td>49.7</td>
<td>61.7</td>
</tr>
</tbody>
</table>

The classification problem and InfoNCE objective. To be specific, the classification problem aims to group similar samples together, but InfoNCE aims to distinguish every instance. When working with SwAV, we tried to set the number of prototypes to 1000, 3000, and 10000. Finally, the best result we can get is 49.7%, which is 12% lower than SimMatch. SwAV aims to distribute the samples equally to each prototype, preventing the model from collapsing. However, distribution alignment has a similar objective, which is adopted by SimMatch in Eq (2). Moreover, the SwAV loss will be trained in a completely unsupervised manner, which will lose the power of the labels. The advantage of $L_{in}$ is that the label information can easily cooperate with instance similarities.

5. Conclusion

In this paper, we proposed a new semi-supervised learning framework SimMatch, which considers the consistency regularization on both semantic-level and instance-level. We also introduced a labeled memory buffer to fully leverage the data annotations on instance-level. Finally, our defined unfolding and aggregation operation allows the label to propagate between semantic-level and instance-level information. Extensive experiment shows the effectiveness of each component in our framework. The results on ImageNet-1K demonstrate the state-of-the-art performance for semi-supervised learning.

Acknowledgment

This work is funded by the National Key Research and Development Program of China (No. 2018AAA0100701) and the NSFC 61876095. Chang Xu was supported in part by the Australian Research Council under Projects DE180101438 and DP210101859. Shan You is supported by Beijing Postdoctoral Research Foundation.
References


[56] 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.


