# DP-SSL: Towards Robust Semi-supervised Learning with A Few Labeled Samples

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

The scarcity of labeled data is a critical obstacle to deep learning. Semi-supervised learning (SSL) provides a promising way to leverage unlabeled data by pseudo labels. However, when the size of labeled data is very small (say a few labeled samples per class), SSL performs poorly and unstably, possibly due to the low quality of learned pseudo labels. In this paper, we propose a new SSL method called DP-SSL that adopts an innovative data programming (DP) scheme to generate probabilistic labels for unlabeled data. Different from existing DP methods that rely on human experts to provide initial labeling functions (LFs), we develop a multiple-choice learning (MCL) based approach to automatically generate LFs from scratch in SSL style. With the noisy labels produced by the LFs, we design a label model to resolve the conflict and overlap among the noisy labels, and finally infer probabilistic labels for unlabeled samples. Extensive experiments on four standard SSL benchmarks show that DP-SSL can provide reliable labels for unlabeled data and achieve better classification performance on test sets than existing SSL methods, especially when only a small number of labeled samples are available. Concretely, for CIFAR-10 with only 40 labeled samples, DP-SSL achieves 93.82% annotation accuracy on unlabeled data and 93.46% classification accuracy on test data, which are higher than the SOTA results.

## 1 Introduction

The de-facto approaches to deep learning achieve phenomenal success with the release of huge labeled datasets. However, large manually-labeled datasets are time-consuming and expensive to acquire, especially when expert labelers are required. Nowadays, many techniques are proposed to alleviate the burden of manual labeling and help to train models from scratch, such as active learning [1], crowd-labeling [2], distant supervision [3], semi [4]/weak [5]/self-supervision [6]. Among them, semi-supervised learning (SSL) is one of the most popular techniques to cope with the scarcity of labeled data. Two major strategies of SSL are pseudo labels [7] and consistency regularization [8]. Pseudo labels (also called self-training [9]) utilize a model's predictions as the labels to train the model again, while consistency of regularization forces a model to make the same prediction under different transformations. However, when the size of labeled data is small, SSL performance degrades drastically in both accuracy and robustness. Fig. 1 shows the change of prediction error rate with the number of labeled samples of CIFAR-10. When the number of labeled samples reduces from 250 to 40, error rates of major existing SSL methods increase from 4.74% (USADTM) to 36.49% (MixMatch). One possible reason of performance deterioration is due to quality degradation of learnt pseudo labels when labeled data size is small. Therefore, in this paper we address this problem by developing sophisticated labeling techniques for unlabeled data to boost SSL even when the number of labeled samples is very small (e.g. a few labeled samples per class).

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Recently, *data programming* (DP) was proposed as a new paradigm of weak supervision [10]. In DP, human experts are required to transform the decision-making process into a series of small functions (called *labeling functions*, abbreviated as LFs), thus data can be labeled programmatically. Besides, a label model is applied to determining the correct labels based on consensus from the noisy and conflicting labels assigned by the LFs. Such a paradigm achieves considerable success in NLP tasks [11–14]. In addition, DP has also been applied to computer vision tasks [15, 16]. However, current DP methods require human experts to provide initial LFs, which is time-consuming and expensive, and it is not easy to guarantee the quality of LFs. Furthermore, LFs specifically defined for one task usually cannot be re-used for other tasks.

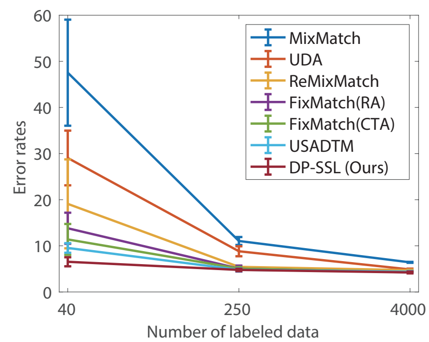


Figure 1: Error rate *vs*. #labeled samples (CIFAR-10). Results of existing methods are from the original papers. When only 40 labeled samples are given, all existing SSL methods are substantially degraded and more unstably, while our method is still effective and robust.

In this paper, we propose a new SSL method called DP-SSL that is effective and robust even when the number of labeled samples is very small. In DP-SSL, an innovative data programming (DP) scheme is developed to generate probabilistic labels for unlabeled data. Different from existing DP methods, we develop a *multiple-choice learning* (MCL) based approach to automatically generate LFs from scratch in SSL style. To remedy the over-confidence problem with existing MCL methods, we assign an additional option as abstention for each LF. After that, we design a label model to resolve the conflict and overlap among the noisy labels generated by LFs, and infer a probabilistic label for each unlabeled sample. Finally, the probabilistic labels are used to train the end model for classifying unlabeled data. Our experiments validate the effectiveness and advantage of DP-SSL. As shown in Fig. 1, DP-SSL performs best, and only 1.76% increase of error rate when the size of labeled samples decreases from 250 to 40 in CIFAR-10.

Note that the pseudo labels used in existing SSL methods is quite different from the probabilistic labels in DP-SSL, which may explain the advantage of DP-SSL over existing SSL methods. On the one hand, pseudo labels are "hard" labels that indicate an unlabeled sample belonging to a certain class or not, while probabilistic labels are "soft" labels that indicate the class distributions of unlabeled samples. Obviously, the latter should be more flexible and robust. On the other hand, pseudo labels are actually generated by a single model for all unlabeled samples, while probabilistic labels are generated from a number of diverse and specialized LFs (due to the MCL mechanism), which makes the latter more powerful in generalization as a whole.

In summary, the contributions of this paper are as follows: 1) We propose a new SSL method DP-SSL that employs an innovative data programming method to generate probabilistic labels for unlabeled data, which makes DP-SSL effective and robust even when there are only a few labeled samples per class. 2) We develop a multiple choice learning based approach to automatically generate diverse and specialized LFs from scratch for unlabeled data in SSL manner. 3) We design a label model with a novel potential and an unsupervised quality guidance regularizer to infer probabilistic labels from the noisy labels generated by LFs. 4) We conduct extensive experiments on four standard benchmarks, which show that DP-SSL outperforms the state-of-the-art methods, especially when only a small number of labeled samples are available, DP-SSL is still effective and robust.

## 2 Related Work

Here we briefly review the latest advances in multiple choice learning, semi-supervised learning, and data programming, which are related to our work. Detailed information is available in [17–20].

### 2.1 Multiple Choice Learning

Multiple choice learning (MCL) [21] was proposed to overcome the low diversity problem of models trained independently in ensemble learning. For example, stochastic multiple choice learning [22] is for training diverse deep ensemble models. However, a crucial problem with MCL is that each model tends to be overconfident. which results in poor final prediction. To solve this problem, [23] forces the predictions of non-specialized models to meet a uniform distribution, so that the final decision is summed over diverse outputs. [24] proposes an additional network to estimate the weight of each specialist's output. In this paper, we develop an improved MCL based scheme to automatically generate diverse and specialized labeling functions (LFs) from scratch in an SSL manner. These LFs are used to generate preliminary (usually noisy) labels for unlabeled data.

### 2.2 Semi-supervised Learning

Semi-supervised learning (SSL) has been extensively studied in image classification [25], object detection [26], and semantic segmentation [27]. Two popular SSL strategies for image classification are pseudo labels [7] and consistency regularization [8]. Pseudo-label methods generate artificial labels for some unlabeled images and then train the model with these artificial labels, while consistency regularization tries to obtain an artificial distribution/label and applied it as a supervision signal with other augmentations/views. These two strategies have been adopted by a number of recent SSL works [4, 8, 28–37]. For example, FixMatch [4] proposes a simple combination of pseudo labels and consistency regularization. [35] employs unsupervised learning and clustering to determine the pseudo labels. In this paper, we propose a new SSL method that is effective and robust even when the size of labeled data is very small. Our method employs an innovative data programming alike method to automatically generate probabilistic labels for unlabeled data.

### 2.3 Data Programming

Data programming [10] is a weak supervision paradigm proposed to infer correct labels based on the consensus among noisy labels from labeling functions (LFs), which are modules embedded with decision-making processes for generating labels programmatically. Following the DP paradigm, Snorkel [12] and Snuba [38] were proposed as a rapid training data creation system. Their LFs are built with various weak supervision sources, like pattern regexes, heuristics, and external knowledge base etc. Recently, more works are reported in the literature [11, 13–16, 20, 39–45]. Among them, [11, 13, 14, 43–45] focus on the adaption of label model in DP. For example, [20] aims to reduce the computational cost and proposes a closed-formed solution for training the label model. [15, 16, 39–41] apply DP to computer vision. Concretely, [16, 40, 41] heavily rely on the pretrained models. [39] combines crowdsourcing, data augmentation, and DP to create weak labels for image classification. [15] presents a novel view for resolving infrequent data in scene graph prediction training datasets via image-agnostic features in LFs. However, all these methods cannot directly applied to training models from scratch with a small number of labeled samples. Thus, in this paper we extend DP by exploring both MCL and SSL to generate arbitrary labeling functions.

## 3 Method

For a $C$-class SSL classification problem, assume that all training data $X$ are divided into labeled data $X_l$ and unlabeled $X_u$, and test data are denoted as $X_t$. Following the notation in [4, 35], $\{x_l, x_l^w\} \in X_l$ are the paired labeled samples with labels $y_l \in \{1, \ldots, C\}$, and $\{x_u, x_u^w, x_u^s\} \in X_u$ are the triple unlabeled samples. Here, $x_l$ and $x_u$ represent the raw images without any transformations. $x_l^w, x_u^w$, and $x_u^s$ are the images based on the weak and strong augmentation strategies, respectively. In this paper, weak augmentation uses a standard flip-and-shift strategy, and strong augmentation is the RandAugment[46] strategy with Cutout [47] augmentation operation.

### 3.1 Framework

Fig. 2 shows the framework of our DP-SSL method, which works in three major steps as follows:

- Step 1. We employ an MCL based approach to automatically generate LFs from scratch in an SSL style. Here, each LF is trained on a subset of $C$ classes in training set based on MCL. As shown in Fig 2, the 2nd LF is trained with samples of classes "horse" and "dog", and abstains from prediction when facing monkey images.
- Step 2. A graphical model is developed as the label model to aggregate the noisy labels and produce probabilistic labels for unlabeled training data. The label model is learned in an SSL manner with an additional regularizer.

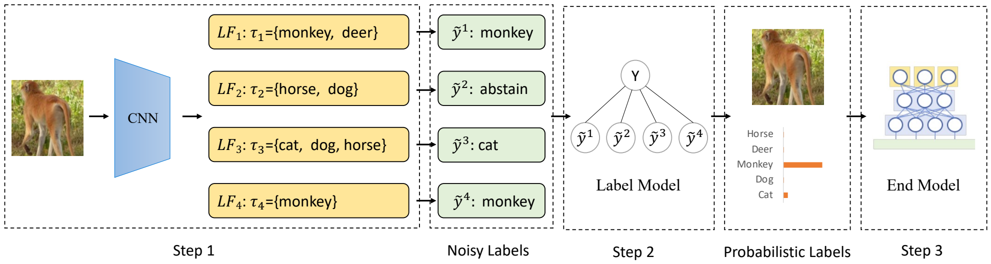


Figure 2: Framework of the DP-SSL method with four LFs.

- Step 3. The end model is trained with both provided labels and probabilistic labels generated from Step 2. Finally, we verify the performance of the end model on test data.

### 3.2 Labeling Function

In Step 1 of our method, LFs are exploited to generate noisy labels for each unlabeled image. In previous DP works for computer vision, LFs are built via external image-agnostic knowledge [15] or pretrained models [16, 40, 41]. However, it is difficult to explicitly describe the rules of image classification. Instead, here we innovatively explore MCL and SSL for automatic LF generation.

As shown in Fig. 2, we share the same backbone (Wide ResNet [48] in this paper) to extract features of images for multiple prediction heads (called LFs in this paper). To promote the diversity of LFs, we transform the features and feed each LF with different transformed features as follows:

$$\tilde{f}_k = \sum_{j=1}^{HW} \frac{e^{-\beta_k \Lambda(f_j, c_k)}}{\sum_{k'=1}^{K} e^{-\beta_{k'} \Lambda(f_j, c_{k'})}} (f_j - c_k). \tag{1}$$

In this paper, $f \in \mathbb{R}^{H \times W \times C}$ denotes the feature before global average pooling in the backbone, $f_j \in \mathbb{R}^C$ is the feature vector at position $j$ of $f$. $c_k \in \mathbb{R}^C$ is the learnable clustering center in the $k$-th LF, $\beta_k$ is the learnable variable of the $k$-th cluster, $\Lambda(A, B)$ is the distance metric for $A$ and $B$. Thus, $\tilde{f}_k$ corresponds to the feature fed to the $k$-th LF, and describes the $k$-th aggregated pattern of $f$ among $K$ centers $c_k$, it can also be considered as a learnable weighted average pooling for feature $f$. In our experiments, $e^{-\beta_k \Lambda(f_j, c_k)}$ is approximately implemented in the form as in [49].

As depicted in [22], the classifiers lack diversity of prediction even trained with different protocols. Therefore, we adopt MCL to assign a subset of labeled data for each classifier automatically to improve diversity, which is formulated as

$$\begin{aligned} \mathcal{L}_1(x_l^w, y_l) &= \sum_{k=1}^{K} u^k * H(y_l, \tilde{p}_l^{w,k}) \\ s.t. \quad \sum_{k=1}^{K} u^k &= \rho * K, u^k \in \{0, 1\}, \end{aligned} \tag{2}$$

with $H(y_l, \tilde{p}_l^{w,i}) \leq H(y_l, \tilde{p}_l^{w,j})$ for $\forall i, j$ when $u^i = 1, u^j = 0$. Above, $H(y_l, \tilde{p}_l^{w,k})$ denotes the cross entropy between the ground truth label $y_l$ and the predicted distribution $\tilde{p}_l^{w,k}$ of the $k$-th LF, $u^k$ is the indicator variable in MCL to indicate whether the $k$-th LF is specialized in labeling $x_l^w$. $K$ is the number of LFs in DP and $\rho$ is the ratio of specialist LFs. When $\rho$ is equal to 1, MCL deteriorates to the basic ensemble learning, where all $K$ classifiers are trained with the same data.

Based on MCL, each LF is a specialist for some classes, so it can get high accuracy for samples in these classes. While for samples from other classes not specialized by the LF, it fails to predict due to over-confidence. Thus, we allow each LF to abstain from some samples in the dataset. Formally, we denote 0 as the abstention label, and the specialized classes of the $k$-th LF as $\tau_k \in \{\tau_k^1, \ldots, \tau_k^{|\tau_k|}\}$. Then, the output of the $k$-th LF $\tilde{y}^k$ satisfies $\tilde{y}^k \in \{0\} \cup \tau_k$. For example, the output of the 1st LF in

Fig. 2 is among "monkey", "deer" and "abstention" for its specialized category set is $\tau_1 = \{\text{monkey}, \text{deer}\}$. The objective function over labeled samples with abstention option is

$$\mathcal{L}_2(x_l^w, y_l) = \sum_{k=1}^{K} (\mathbb{1}(y_l \in \tau_k) H(y_l, \tilde{q}_l^{w,k}) + \mathbb{1}(y_l \notin \tau_k) H(0, \tilde{q}_l^{w,k})), \tag{3}$$

where $\tilde{q}_l^{w,k}$ is the normalized probability distribution of CONCAT$(\tilde{p}_l^{w,k}, \tilde{p}_{l,a}^{w,k})$ and $\tilde{p}_{l,a}^{w,k}$ is the probability of abstention. Then, for the unlabeled training data, we follow the settings in FixMatch [4], where unlabeled data are supervised by the pseudo labels $\tilde{y}_u^{w,k}$ of weak augmentation data $x_u^w$. Thus,

$$\mathcal{L}(x_u^w, x_u^s) = \sum_{k=1}^{K} \mathbb{1}(\max(\tilde{q}_u^{w,k}) \geq \epsilon)\Big(\mathbb{1}(\tilde{y}_u^{w,k} \in \tau_k) H(\tilde{y}_u^{w,k}, \tilde{q}_u^{s,k}) + \mathbb{1}(\tilde{y}_u^{w,k} \notin \tau_k) H(0, \tilde{q}_u^{s,k})\Big), \tag{4}$$

where $\tilde{y}_u^{w,k} = arg\max(\tilde{q}_u^{w,k})$. Specifically, we keep only samples whose largest probability (including the abstention option) is above the predefined threshold $\epsilon$ (0.95 in our paper), and train the model on the kept data with pseudo label $\tilde{y}_u^{w,k}$. Accordingly, the training in this step is to minimize the objective function as follows:

$$\mathcal{L}(x_l^w, y_l, x_u^w, x_u^s) = \mu_1 \mathcal{L}_1(x_l^w, y_l) + \mu_2 \mathcal{L}_2(x_l^w, y_l) + \mu_u \mathcal{L}(x_u^w, x_u^s), \tag{5}$$

where $\mu_1$, $\mu_2$ and $\mu_u$ are hyper-parameters. In our implementation, we first set $\mu_1 = 1$ and $\mu_2 = \mu_u = 0$, then adjust $\mu_1$ to 0 and $\mu_2 = \mu_u = 1$ after the convergence of $\mathcal{L}_1$.

Generally, in Step 1, MCL is expected to generate specialized class sets $\tau$ for LFs, with which samples are more easily discriminated by SSL classifiers even there are a few labeled samples. Besides, the abstention option is for addressing the over-confidence problem of samples from non-specialized sets.

### 3.3 Label Model

In Step 2 of our method, we utilize a graphical model to specify a single prediction by integrating noisy labels provided by $K$ LFs. For simplification, we assume that the $K$ LFs are independent (as shown in Fig. 2). Then, suppose that $\tilde{\mathbf{y}} \in \mathbb{R}^K$ is the vectorized form of the predictions from $K$ LFs, the joint distribution of the label model can be described as:

$$P(y, \tilde{\mathbf{y}}) = \frac{1}{Z} \prod_{k=1}^{K} \phi(y, \tilde{y}^k) \tag{6}$$

where $Z$ is the normalizer of the joint distribution, $\phi$ is the potential that couples the target $y$ and noisy label $\tilde{y}^k$. In this paper, we extend the dimension of parameters $\theta$ in label model to $K \times C$ to support multi-class classification. Set $e_{ky} := exp(\theta_{ky})$, which is the exponent of parameters $\theta_{ky}$. Now we are to construct the potential function $\phi$. Due to the specialized LFs, the potential $\phi$ should benefit the final prediction when a noisy label agrees with the target. That is, we should have $\phi(y, \tilde{y}^k) > 1$. Thus, we set $\phi$ as $1 + e_{ky}$ for this case. On the contrary, the potential $\phi$ should negatively impact the final prediction when a noisy label conflicts with the target label in the specialized category set, i.e., we should have $\phi(y, \tilde{y}^k) < 1$. Therefore, for this case we set $\phi$ to $1/(1 + e_{ky})$. For the other cases, we follow the design in [14]. In summary, the potential $\phi$ is defined as follows:

$$\phi(y, \tilde{y}^k) = \begin{cases} 1 + e_{ky}, & if\ y \in \tau_k,\ \tilde{y}^k \in \tau_k,\ \tilde{y}^k = y \\ 1/(1 + e_{ky}), & if\ y \in \tau_k,\ \tilde{y}^k \in \tau_k,\ \tilde{y}^k \neq y \\ e_{ky}, & if\ y \notin \tau_k,\ \tilde{y}^k \in \tau_k,\ \tilde{y}^k \neq y \\ 1. & otherwise \end{cases} \tag{7}$$

With the potential above, the normalizer $Z$ of the joint distribution in Eq. (6) can be obtained by summarizing over $y$ and $\tilde{y}^k$:

$$\begin{aligned} Z &= \sum_{y \in \mathcal{Y}} \prod_{k=1}^{K} \sum_{\tilde{y}^k \in \{0\} \cup \tau_k} \phi(y, \tilde{y}^k) \\ &= \sum_{y \in \mathcal{Y}} \prod_{k=1}^{K} \Big(\mathbb{1}(y \in \tau_k)(2 + e_{ky} + \frac{|\tau_k| - 1}{1 + e_{ky}}) + \mathbb{1}(y \notin \tau_k)(1 + |\tau_k| e_{ky})\Big). \end{aligned} \tag{8}$$

Then, the objective function of the label model can be expressed in an SSL manner as follows:

$$\mathcal{L}(\tilde{\mathbf{y}}_l, y_l, \tilde{\mathbf{y}}_u) = \underbrace{\sum_{x_l} H(y_l, P(y, \tilde{\mathbf{y}}_l))}_{\text{labeled samples}} + \underbrace{(-\sum_{x_u} \log \sum_{y \in \mathcal{Y}} P(y, \tilde{\mathbf{y}}_u))}_{\text{unlabeled samples}} + R(\theta, \tilde{\mathbf{y}}_u), \tag{9}$$

where the first part is the cross-entropy loss, the second is the negative log marginal likelihood on the observed noisy labels $\tilde{y}_u$, and the third is a regularizer. In our method, the regularizer is utilized to guide the label model with statistical information (the accuracy of each LF). However, the accuracy of each LF on noisy labels is unavailable, while the accuracy on labeled training is almost 100% due to over-fitting. Thus, we have to estimate the accuracy of each LF with the observable noisy labels $\tilde{\mathbf{y}}$, which will be presented in Sec. 3.4. After training, the label model produces probabilistic labels $\pi$ by computing the joint distribution in Eq. (6) with the noisy labels $\tilde{\mathbf{y}}$.

### 3.4 Accuracy Estimation

Now, we formally describe our method for estimating the accuracy of LFs. We transform the multi-class problem into $C$ one-versus-all tasks. For the $i$-th one-versus-all task, we denote the unobserved true labels as $Y_i \in \{\pm 1\}$ ($+1$ is the positive label, $-1$ represents that of the other categories), noisy labels of the $k$-th LF as $\lambda_i^k \in \{\pm 1, 0\}$ (0 for abstention). Then, we can write $\mathbb{E}[\lambda_i^k Y_i]$ as

$$\begin{aligned}
\mathbb{E}[\lambda_i^k Y_i] &= P(\lambda_i^k Y_i = 1) - P(\lambda_i^k Y_i = -1) \\
&= P(\lambda_i^k Y_i = 1) - (1 - P(\lambda_i^k Y_i = 1) - P(\lambda_i^k Y_i = 0)) \\
&= 2P(\lambda_i^k = Y_i) + P(\lambda_i^k = 0) - 1.
\end{aligned} \tag{10}$$

Assume that $\lambda_i^j \perp\!\!\!\perp \lambda_i^k | Y_i$ for distinct $j$ and $k$, then

$$\mathbb{E}[\lambda_i^j \lambda_i^k] = \mathbb{E}[\lambda_i^j Y_i^2 \lambda_i^k] = \mathbb{E}[\lambda_i^j Y_i]\mathbb{E}[\lambda_i^k Y_i] \tag{11}$$

with the fact that $Y_i^2 = 1$. In Eq. (11), $\hat{\mathbb{E}}[\lambda_i^j \lambda_i^k] = \frac{1}{|x_u|}\sum_{x_u} \lambda_i^j \lambda_i^k$ is observable, which can be derived from the noisy labels of the $j$-th and $k$-th LFs, while $\mathbb{E}[\lambda_i^j Y_i]$ and $\mathbb{E}[\lambda_i^k Y_i]$ remains to be solved due to true label $Y_i$ is unavailable. Next, we introduce a third labeling result from the $l$-th LF as $\lambda_i^l$, such that $\hat{\mathbb{E}}[\lambda_i^j \lambda_i^l]$ and $\hat{\mathbb{E}}[\lambda_i^k \lambda_i^l]$ are observable. Now, $|\hat{\mathbb{E}}[\lambda_i^j Y_i]|$, $|\hat{\mathbb{E}}[\lambda_i^k Y_i]|$, $|\hat{\mathbb{E}}[\lambda_i^l Y_i]|$ can be solved by a triplet method as follows:

$$\begin{aligned}
|\hat{\mathbb{E}}[\lambda_i^j Y_i]| &= \sqrt{|\hat{\mathbb{E}}[\lambda_i^j \lambda_i^k] \cdot \hat{\mathbb{E}}[\lambda_i^j \lambda_i^l] / \hat{\mathbb{E}}[\lambda_i^k \lambda_i^l]|}, \\
|\hat{\mathbb{E}}[\lambda_i^k Y_i]| &= \sqrt{|\hat{\mathbb{E}}[\lambda_i^j \lambda_i^k] \cdot \hat{\mathbb{E}}[\lambda_i^k \lambda_i^l] / \hat{\mathbb{E}}[\lambda_i^j \lambda_i^l]|}, \\
|\hat{\mathbb{E}}[\lambda_i^l Y_i]| &= \sqrt{|\hat{\mathbb{E}}[\lambda_i^j \lambda_i^l] \cdot \hat{\mathbb{E}}[\lambda_i^k \lambda_i^l] / \hat{\mathbb{E}}[\lambda_i^j \lambda_i^k]|}.
\end{aligned} \tag{12}$$

We can obtain the estimated accuracy of each LF by resolving the sign of $\mathbb{E}[\lambda_i^k Y_i]$. Let $\hat{a}_i^k := \hat{P}(\lambda_i^k = Y_i | \lambda_i^k \neq 0)$ be the estimated accuracy of the $k$-th LF on the $i$-th category. Therefore, the regularizer of $R(\theta, \tilde{y}_u)$ can be formulated as

$$R(\theta, \tilde{\mathbf{y}}_u) = \sum_{i=1}^{C} \sum_{k}^{K} \hat{a}_i^k \log P_\theta(\lambda_i^k = Y_i | \tilde{y}_u^k \neq 0) + (1 - \hat{a}_i^k) \log(1 - P_\theta(\lambda_i^k = Y_i | \tilde{y}_u^k \neq 0)) \tag{13}$$

where $P_\theta(\lambda_i^k = Y_i | \tilde{y}_u^k \neq 0)$ can be computed in closed form by marginalizing over all the other variables in the model in Eq. (6) without noisy labels $\tilde{y}$. Details of $P_\theta$ can be referred to **Appendix**.

### 3.5 End Model

In Step 3, probabilistic labels are used to train an end model under any network architecture. We utilize noise-aware empirical risk expectation as the objective function to take annotation errors into account. Accordingly, the final objective function is as follows:

$$\mathcal{L}(x_l, y_l, x_u, \pi) = \underbrace{\sum_{x_l} H(y_l, p_l)}_{\text{labeled samples}} + \underbrace{\sum_{x_u} \mathbb{E}_{y \sim \pi} H(y, p_u)}_{\text{unlabeled samples with probabilistic label}} \tag{14}$$

where $p_l$ and $p_u$ are the predicted distributions of $x_l$ and $x_u$, $\pi$ is the distribution produced by the label model in Sec. 3.3. Actually, $H(y_l, p_l)$ can also be rewritten as $\mathbb{E}_{y\sim P(y_l)}H(y, p_l)$ where $P(y_l)$ is the one-hot distribution of $y_l$.

# 4 Experiments

## 4.1 Implementation Details

In the training phase, we follow the settings of previous works [4, 34, 35], augment data in weak (a standard flip-and-shift strategy) and strong forms (RandAugment [46] followed by Cutout [47] operation), and utilize a Wide ResNet as the end model for a fair comparison. In our framework, the batch size for labeled data and unlabeled data is set to 64 and 448, respectively. Besides, we use the same hyperparameters ($K = 50$, $\rho = 0.2$, $\epsilon = 0.95$) for all datasets. We compare DP-SSL with major existing methods on CIFAR-10 [50], CIFAR-100 [50], SVHN [51] and STL-10 [52]. We also analyze the effect of annotation and conduct ablation study in Sec. 4.4 and Sec. 4.5 respectively. All experiments are implemented in Pytorch v1.7 and conducted on 16 NVIDIA RTX3090s.

## 4.2 Datasets

**CIFAR-10 and CIFAR-100** [50] contain 50,000 training examples and 10,000 validation examples. All images are of 32x32 pixel size and fall in 10 or 100 classes, respectively.

**SVHN** [51] is a digital image dataset that consists of 73,257, 26,032 and 531,131 samples in the train, test, and extra folders. It has the same image resolution and category number as CIFAR-10.

**STL-10** [52] is a dataset for evaluating unsupervised and semi-supervised learning. It consists of 5000 labeled images and 8000 validation samples of 96x96 size from 10 classes. Besides, there are 100,000 unlabeled images available, including odd samples.

## 4.3 Comparison with Existing SSL Methods

For a fair comparison, we conduct experiments with the codebase of FixMatch and cite the results on CIFAR-10, CIFAR-100, SVHN and STL-10 from [4, 35]. We utilize the same network architecture (a Wide ResNet-28-2 for CIFAR-10 and SVHN, WRN-28-8 for CIFAR-100, and WRN-37-2 for STL-10) and training protocol of FixMatch, such as optimizer and learning rate schedule. Unlabeled data are generated by the scripts in FixMatch. Results of DP-SSL and existing methods in Tab. 1 and Tab. 2 are presented with the mean and standard deviation (STD) of accuracy on 5 pre-defined folds.

As shown in Tab. 1, our method achieves the best performance in most cases, especially when there are only 4 labeled samples per class. Specifically, our method achieves a 93.46% accuracy on CIFAR-10 with 4 labeled samples per category, which is 3.3% higher than that of USADTM — the state-of-the-art method. Again on STL-10, our method surpasses USADTM and achieves the best performance when there are 4 and 25 labeled samples per class.

On CIFAR-100, our method performs the best for 40 labels case and the 2nd for 2500 and 10,000 labels cases. We also notice that DP-SSL has relatively large STDs for 2500 and 10,000 labels cases, which is due to the coarse accuracy estimation. In fact, even if triplet mean is adopted in estimation, the triplet selection in Eq. (12) still impacts accuracy estimation and regularizer a lot, especially when $\mathbb{E}[\lambda_i^k Y_i]$ is close to 0 or sign recovery of $\mathbb{E}[\lambda_i^k Y_i]$ is wrong. Actually, there are some advanced approaches to unsupervised accuracy estimation [53–55] that can replace the naive triplet mean estimation. Ideally, if we can obtain the exact accuracy of each class $\hat{b}_i^k := \hat{P}(\lambda_i^k = Y_i|\lambda_i^k = 1)$ and regularize it as $R(\theta, \tilde{\mathbf{y}}_u) = \sum_{i=1}^{C}\sum_{k}^{K} \hat{b}_i^k \log P_\theta(\lambda_i^k = Y_i|\tilde{y}_u^k = i) + (1 - \hat{b}_i^k)\log(1 - P_\theta(\lambda_i^k = Y_i|\tilde{y}_u^k = i))$, we will get an end model with ($27.92 \pm$ **0.23**)% error rate for 2500 labeled samples.

Comparing with USADTM, our method does not perform well enough when more labeled data available. For USADTM, apart from the proxy label generator, unsupervised representation learning contributes a lot for its performance. As shown in the ablation study of [35], USADTM without unsupervised representation learning achieves around 5.73% and 4.99% error rate for 250 and 4000 labeled samples in CIFAR-10, while our method DP-SSL obtains 4.78% and 4.23% error rate.

Table 1: Results of error rate on CIFAR-10, CIFAR-100 and SVHN for different existing SSL methods (Π-Model [28], Pseudo-Labeling [7], Mean Teacher [31], MixMatch [30], UDA [33], ReMixMatch [34], FixMatch [4] and USADTM [35]) and our DP-SSL method.

| | CIFAR-10 | | | CIFAR-100 | | | SVHN | | |
|---|---|---|---|---|---|---|---|---|---|
| Method | 40 labels | 250 labels | 4000 labels | 400 labels | 2500 labels | 10000 labels | 40 labels | 250 labels | 1000 labels |
| Π -Model | - | 54.26$\pm$3.97 | 14.01$\pm$0.38 | - | 57.25$\pm$0.48 | 37.88$\pm$0.11 | - | 18.96$\pm$1.92 | 7.54$\pm$0.36 |
| Pseudo-Labeling | - | 49.78$\pm$0.43 | 16.09$\pm$0.28 | - | 57.38$\pm$0.46 | 36.21$\pm$0.19 | - | 20.21$\pm$1.09 | 9.94$\pm$0.61 |
| Mean Teacher | - | 32.32$\pm$2.30 | 9.19$\pm$0.19 | - | 53.91$\pm$0.57 | 35.83$\pm$0.24 | - | 3.57$\pm$0.11 | 3.42$\pm$0.07 |
| MixMatch | 47.54$\pm$11.50 | 11.05$\pm$0.86 | 6.42$\pm$0.10 | 67.61$\pm$1.32 | 39.94$\pm$0.37 | 28.31$\pm$0.33 | 42.55$\pm$14.53 | 3.98$\pm$0.23 | 3.50$\pm$0.28 |
| UDA | 29.05$\pm$5.93 | 8.82$\pm$1.08 | 4.88$\pm$0.18 | 59.28$\pm$0.88 | 33.13$\pm$0.22 | 24.50$\pm$0.25 | 52.63$\pm$20.51 | 5.69$\pm$2.76 | 2.46$\pm$0.24 |
| ReMixMatch | 19.10$\pm$9.64 | 5.44$\pm$0.05 | 4.72$\pm$0.13 | 44.28$\pm$2.06 | **27.43**$\pm$0.31 | 23.03$\pm$0.56 | 3.34$\pm$0.20 | 2.92$\pm$0.48 | 2.65$\pm$0.08 |
| FixMatch (RA) | 13.81$\pm$3.37 | 5.07$\pm$0.65 | 4.26$\pm$0.05 | 48.85$\pm$1.75 | 28.29$\pm$0.11 | 22.60$\pm$0.12 | 3.96$\pm$2.17 | 2.48$\pm$0.38 | 2.28$\pm$0.11 |
| FixMatch (CTA) | 11.39$\pm$3.35 | 5.07$\pm$0.33 | 4.31$\pm$0.15 | 49.95$\pm$3.01 | 28.64$\pm$0.24 | 23.18$\pm$0.11 | 7.65$\pm$7.65 | 2.64$\pm$0.64 | 2.36$\pm$0.19 |
| USADTM | 9.54$\pm$1.04 | 4.80$\pm$0.32 | 4.40$\pm$0.15 | 43.36$\pm$1.89 | 28.11$\pm$0.21 | **21.35**$\pm$0.17 | 3.01$\pm$1.97 | **2.11**$\pm$0.65 | **1.96**$\pm$0.05 |
| DP-SSL (ours) | **6.54**$\pm$0.98 | **4.78**$\pm$0.26 | **4.23**$\pm$0.20 | **43.17**$\pm$1.29 | 28.00$\pm$0.79 | 22.24$\pm$0.31 | **2.98**$\pm$0.86 | 2.16$\pm$0.36 | 1.99$\pm$0.18 |
| Fully Supervised | 2.74 | | | 16.84 | | | 1.48 | | |

Table 2: Results of error rate on STL-10.

| STL-10 | | | | | | | |
|---|---|---|---|---|---|---|---|
| Method | 1000 labels | Method | 1000 labels | Method | 40 labels | 250 labels | 1000 labels |
| Π -Model | 26.23$\pm$0.82 | UDA | 7.66$\pm$0.56 | USADTM | 9.63$\pm$1.35 | 6.85$\pm$1.09 | **4.01**$\pm$0.59 |
| Pseudo-Labeling | 27.99$\pm$0.80 | ReMixMatch | 5.23$\pm$0.45 | DP-SSL (ours) | **9.32**$\pm$0.91 | **6.83**$\pm$0.71 | 4.97$\pm$0.42 |
| Mean Teacher | 21.43$\pm$2.39 | FixMatch (RA) | 7.98$\pm$1.50 | Fully Supervised | | 1.48 | |
| MixMatch | 10.41$\pm$0.61 | FixMatch (CTA) | 5.17$\pm$0.63 | | | | |

### 4.4 Analysis

**Annotation performance**. Intuitively, the holistic performance of the end model in our method highly depends on the quality of annotation results. Thus, we present the macro precision/recall/F1 score and coverage of the annotated labels of our method on CIFAR-10, CIFAR-100, and SVHN in Tab. 3. We can see that our method achieves over 99% coverage, which means that it produces probabilistic labels for almost all unlabeled data. Comparing to the results in [35], the label model with 40 labeled samples outperforms the proxy label generator, FixMatch and USADTM get 88.51% and 89.48% accuracy, respectively. Furthermore, our method achieves 97.36% accuracy for unlabeled data with the top-500 highest probabilities in each category. Meanwhile, we also present results of Majority Voting and FlyingSquid [20] in Tab. 3 based on the noisy labels from Step 1 of our method for comparison. Majority Voting gets bad performance because the number of LFs triggered for different categories is not equal. For FlyingSquid, we implement it with $C$ one-versus-all models to support multi-class tasks, and the large $C$ in CIFAR-100 results in the worst performance.

**Barely supervised learning**. We conduct experiments to test the performance (accuracy and STD) of our method on CIFAR-10 for some extreme cases (10, 20 and 30 labeled samples) to verify the effectiveness of our method. Here, we select the labeled data through the scripts of FixMatch with 5 different random seeds. As claimed in FixMatch, it reaches between 48.58% and 85.32% test accuracy with a median of 64.28% for 10 labeled samples, while our method obtains accuracy from 61.32% to 83.7%. As for 20 and 30 labeled samples, our method gets $(85.29 \pm 3.14)$% and $(89.81 \pm 1.59)$% accuracy respectively, which have much smaller STDs than that reported in [36].

Table 3: The macro Precision/Recall/F1 Score/Coverage of the annotated labels on CIFAR-10, CIFAR-100, and SVHN for our method and two typical existing label models.

| | | CIFAR-10 | | | CIFAR-100 | | | SVHN | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Method | Metrics | 40 labels | 250 labels | 4000 labels | 400 labels | 2500 labels | 10000 labels | 40 labels | 250 labels | 1000 labels |
| Majority Vote | F1 Score | 85.96 | 94.23 | 95.77 | 49.97 | 69.81 | 76.03 | 90.86 | 95.38 | 96.14 |
| FlyingSquid[20] | F1 Score | 90.25 | 94.99 | 95.85 | 48.90 | 69.73 | 74.12 | 93.92 | 97.24 | 97.70 |
| DP-SSL (ours) | Precision | 93.47 | 95.30 | 95.89 | 55.62 | 71.91 | 75.12 | 95.20 | 97.65 | 97.79 |
| | Recall | 93.82 | 95.33 | 95.91 | 56.86 | 72.01 | 78.35 | 96.78 | 97.64 | 97.94 |
| | F1 Score | **93.61** | **95.19** | **95.90** | **54.42** | **71.89** | **76.36** | **95.95** | **97.59** | **97.81** |
| | Coverage | 99.35 | 99.79 | 99.91 | 99.33 | 99.87 | 99.94 | 99.15 | 99.67 | 99.93 |

### 4.5 Ablation Study

In DP-SSL, LFs and the label model are the core components to assign probabilistic labels for training the end model. Here, we check the effects of the following factors in the process of producing probabilistic labels by taking CIFAR-10 as the example. For ease of exposition, only the accuracy of predicted labels is presented in Tab. 4.

Table 4: Annotation performance for different configurations on CIFAR-10 with 40 and 250 labels. $K$ and $\rho$ are set to 50 and 0.2 by default.

| Experiments | 40 labels | 250 labels |
|---|---|---|
| Exp1: *w.o.* MCL | 92.46 | 95.02 |
| Exp2: MCL *w.o.* FT | 91.61 | 94.98 |
| Exp3: MCL *w.* FT | **93.82** | **95.33** |
| Exp4: K=20, $\rho$=0.2 | 91.27 | 94.75 |
| Exp5: K=40, $\rho$=0.2 | 93.46 | 95.03 |
| Exp6: K=50, $\rho$=0.2 | **93.82** | 95.33 |
| Exp7: K=60, $\rho$=0.2 | 93.54 | 95.28 |
| Exp8: K=100, $\rho$=0.2 | 93.68 | **95.35** |
| Exp9: K=50, $\rho$=0.1 | 92.95 | 94.90 |
| Exp10: K=50, $\rho$=0.2 | **93.82** | **95.33** |
| Exp11: K=50, $\rho$=0.3 | 93.55 | 95.23 |
| Exp12: K=50, $\rho$=0.5 | 93.07 | 94.86 |
| Exp13: K=50, $\rho$=1.0 | 92.46 | 95.02 |
| Exp14: *w.o.* Regularizer | 93.19 | 94.94 |
| Exp15: Regularizer | **93.82** | **95.33** |

**MCL**. Feature transformation (FT) described in Eq. (1) can be regarded as a weighted spatial pooling for extracted features. It is proposed to boost the diversity of generated LFs. We conduct comparative experiments for three configurations: 1) *Exp1*: *w.o.* MCL, 2) *Exp2*: MCL *w.o.* FT, 3) *Exp3*: MCL *w.* FT. The results are presented in Tab. 4. It is interesting to see that *Exp1* is better than *Exp2* but worse than *Exp3*. In fact, *Exp1* is a simple ensemble model with a shared backbone, where each LF is trained independently and predicts the labels within $C$ categories. In *Exp2*, we observe that some classifiers have never been optimized in the training phase and thus have an empty specialized set when only a few labeled samples per class are available. Moreover, the specialized sets of many LFs are duplicate, which incurs a negative impact on the performance. However, MCL with FT addresses the drawbacks and helpes our method obtain versatile LFs. Some detailed examples are presented in **Appendix**

**Hyperparameters**. $K$ and $\rho$ are the number of LFs and the ratio of specialists in Eq. (2). *Exp4-13* in Tab. 4 present the variance of performance for different $K$ and $\rho$. In *Exp4-8*, *Exp6* with $K$=50 *performs* the best when 40 labeled samples are available, while *Exp8* with $K$=100 wins the others when 250 labeled samples are provided. To trade off the cost of computation and performance, we set $K$=50 as the default setting in our paper. On the other hand, performance reaches the best from *Exp9* to *Exp13* when $\rho$=0.2. Actually, when $\rho$ is 1.0, MCL becomes a naive ensemble model.

**Regularizer**. The regularizer is proposed to impose a global guidance and improve the robustness of the label model. As shown in Tab. 4, the regularizer does boost the accuracy, especially when facing less labeled samples. Besides, as mentioned in Sec. 4.3, the high-quality guidance of the regularizer also reduces the label model's performance variance, thus improves its robustness.

## 5 Conclusion

In this paper, we explore the data programming idea to boost SSL when only a small number of labeled samples available by providing more accurate labels for unlabeled data. To this end, we propose a new SSL method DP-SSL that employs an innovative DP mechanism to automatically generate labeling functions. To make the labeling functions diverse and specialized, a multiple choice learning based approach is developed. Furthermore, we design an effective label model by incorporating a novel potential and a regularizer with estimated accuracy. With this model, probabilistic labels are inferred by resolving the conflict and overlap among noisy labels from the labeling functions. Finally, an end model is trained under the supervision of the probabilistic labels. Extensive experiments show that DP-SSL can produce high-quality probabilistic labels, and outperforms the existing methods to achieve a new SOTA, especially when only a small number of labeled samples are available.

## 6 Limitations of This Work

In this work, we use coarse accuracy estimation as the statistic information to guide the label model for simplicity. As described in Sec. 3.4, we estimate the accuracy $P_\theta(\lambda_i^k = Y_i | \lambda_i^k \neq 0)$, rather than class-wise accuracy $P_\theta(\lambda_i^k = Y_i | \lambda_i^k = 1)$. Besides, we do not consider the dependency between $C$ one-vs-all tasks.

# References


[1] M. Gao, Z. Zhang, G. Yu, S. Ö. Arık, L. S. Davis, and T. Pfister, “Consistency-based semi-supervised active learning: Towards minimizing labeling cost,” in *ECCV*. Springer, 2020, pp. 510–526.

[2] C. Vondrick, D. Patterson, and D. Ramanan, “Efficiently scaling up crowdsourced video annotation,” *IJCV*, vol. 101, no. 1, pp. 184–204, 2013.

[3] Y. Yao, A. Zhang, X. Han, M. Li, C. Weber, Z. Liu, S. Wermter, and M. Sun, “Visual distant supervision for scene graph generation,” *arXiv preprint arXiv:2103.15365*, 2021.

[4] K. Sohn, D. Berthelot, N. Carlini, Z. Zhang, H. Zhang, C. A. Raffel, E. D. Cubuk, A. Kurakin, and C.-L. Li, “Fixmatch: Simplifying semi-supervised learning with consistency and confidence,” *NeurIPS*, vol. 33, 2020.

[5] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, “Is object localization for free?-weakly-supervised learning with convolutional neural networks,” in *CVPR*, 2015, pp. 685–694.

[6] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in *ICML*. PMLR, 2020, pp. 1597–1607.

[7] D.-H. Lee *et al.*, “Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks,” in *Workshop on ICML*, vol. 3, no. 2, 2013.

[8] M. Sajjadi, M. Javanmardi, and T. Tasdizen, “Regularization with stochastic transformations and perturbations for deep semi-supervised learning,” in *NeurIPS*, 2016.

[9] B. Zoph, G. Ghiasi, T.-Y. Lin, Y. Cui, H. Liu, E. D. Cubuk, and Q. Le, “Rethinking pre-training and self-training,” *NeurIPS*, vol. 33, 2020.

[10] A. Ratner, C. De Sa, S. Wu, D. Selsam, and C. Ré, “Data programming: Creating large training sets, quickly,” *NeurIPS*, vol. 29, p. 3567, 2016.

[11] A. Awasthi, S. Ghosh, R. Goyal, and S. Sarawagi, “Learning from rules generalizing labeled exemplars,” *ICLR*, 2020.

[12] A. J. Ratner, S. H. Bach, H. R. Ehrenberg, and C. Ré, “Snorkel: Fast training set generation for information extraction,” in *SIGMOD*, 2017, pp. 1683–1686.

[13] A. Ratner, B. Hancock, J. Dunnmon, F. Sala, S. Pandey, and C. Ré, “Training complex models with multi-task weak supervision,” in *AAAI*, vol. 33, no. 01, 2019, pp. 4763–4771.

[14] O. Chatterjee, G. Ramakrishnan, and S. Sarawagi, “Data programming using continuous and quality-guided labeling functions,” *AAAI*, 2020.

[15] V. S. Chen, P. Varma, R. Krishna, M. Bernstein, C. Re, and L. Fei-Fei, “Scene graph prediction with limited labels,” in *ICCV*, 2019, pp. 2580–2590.

[16] S. Hooper, M. Wornow, H. S. Ying, H. Kellman, Peter andXue, F. Sala, C. Langlotz, and C. Ré, “Cut out the annotator, keep the cutout: better segmentation with weak supervision,” *ICLR*, 2021.

[17] N. C. Garcia, S. A. Bargal, V. Ablavsky, P. Morerio, V. Murino, and S. Sclaroff, “Distillation multiple choice learning for multimodal action recognition,” in *WACV*, 2021, pp. 2755–2764.

[18] J. E. Van Engelen and H. H. Hoos, “A survey on semi-supervised learning,” *Machine Learning*, vol. 109, no. 2, pp. 373–440, 2020.

[19] M. F. Chen, D. Y. Fu, F. Sala, S. Wu, R. T. Mullapudi, F. Poms, K. Fatahalian, and C. Ré, “Train and you’ll miss it: Interactive model iteration with weak supervision and pre-trained embeddings,” *arXiv preprint arXiv:2006.15168*, 2020.

[20] D. Fu, M. Chen, F. Sala, S. Hooper, K. Fatahalian, and C. Ré, “Fast and three-rious: Speeding up weak supervision with triplet methods,” in *ICML*. PMLR, 2020, pp. 3280–3291.

[21] A. Guzman-Rivera, P. Kohli, D. Batra, and R. Rutenbar, “Efficiently enforcing diversity in multi-output structured prediction,” in *Artificial Intelligence and Statistics*. PMLR, 2014, pp. 284–292.

[22] S. Lee, S. Purushwalkam, M. Cogswell, V. Ranjan, D. J. Crandall, and D. Batra, “Stochastic multiple choice learning for training diverse deep ensembles,” in *NeurIPS*, 2016.

[23] K. Lee, C. Hwang, K. Park, and J. Shin, "Confident multiple choice learning," in *ICML*. PMLR, 2017, pp. 2014–2023.

[24] K. Tian, Y. Xu, S. Zhou, and J. Guan, "Versatile multiple choice learning and its application to vision computing," in *CVPR*, 2019, pp. 6349–6357.

[25] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*. IEEE, 2009, pp. 248–255.

[26] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*. Springer, 2014, pp. 740–755.

[27] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *CVPR*, 2016, pp. 3213–3223.

[28] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," in *ICLR*, 2017.

[29] A. Rasmus, H. Valpola, M. Honkala, M. Berglund, and T. Raiko, "Semi-supervised learning with ladder networks," in *NeurIPS*, 2015.

[30] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," in *NeurIPS*, 2019.

[31] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *NeurIPS*, 2017.

[32] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: a regularization method for supervised and semi-supervised learning," *TPAMI*, vol. 41, no. 8, pp. 1979–1993, 2018.

[33] Q. Xie, Z. Dai, E. Hovy, T. Luong, and Q. Le, "Unsupervised data augmentation for consistency training," in *NeurIPS*, vol. 33, 2020.

[34] D. Berthelot, N. Carlini, E. D. Cubuk, A. Kurakin, K. Sohn, H. Zhang, and C. Raffel, "Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring," in *ICLR*, 2020.

[35] T. Han, J. Gao, Y. Yuan, and Q. Wang, "Unsupervised semantic aggregation and deformable template matching for semi-supervised learning," in *NeurIPS*, 2020.

[36] J. Li, C. Xiong, and S. Hoi, "Comatch: Semi-supervised learning with contrastive graph regularization," *arXiv preprint arXiv:2011.11183*, 2020.

[37] Z. Hu, Z. Yang, X. Hu, and R. Nevatia, "Simple: Similar pseudo label exploitation for semi-supervised classification," *arXiv preprint arXiv:2103.16725*, 2021.

[38] P. Varma and C. Ré, "Snuba: automating weak supervision to label training data," in *VLDB*, vol. 12, no. 3. NIH Public Access, 2018, p. 223.

[39] G. Heo, Y. Roh, S. Hwang, D. Lee, and S. E. Whang, "Inspector gadget: A data programming-based labeling system for industrial images," *VLDB*, 2020.

[40] A. Pal and V. N. Balasubramanian, "Adversarial data programming: Using gans to relax the bottleneck of curated labeled data," in *CVPR*, 2018, pp. 1556–1565.

[41] N. Das, S. Chaba, R. Wu, S. Gandhi, D. H. Chau, and X. Chu, "Goggles: Automatic image labeling with affinity coding," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 1717–1732.

[42] B. Boecking, W. Neiswanger, E. Xing, and A. Dubrawski, "Interactive weak supervision: Learning useful heuristics for data labeling," *ICLR*, 2021.

[43] P. Varma, B. He, P. Bajaj, I. Banerjee, N. Khandwala, D. L. Rubin, and C. Ré, "Inferring generative model structure with static analysis," *NeurIPS*, vol. 30, p. 239, 2017.

[44] S. H. Bach, B. He, A. Ratner, and C. Ré, "Learning the structure of generative models without labeled data," in *ICML*. PMLR, 2017, pp. 273–282.

[45] P. Varma, F. Sala, A. He, A. Ratner, and C. Ré, "Learning dependency structures for weak supervision models," in *ICML*. PMLR, 2019, pp. 6418–6427.

[46] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, “Randaugment: Practical automated data augmentation with a reduced search space,” in *Workshop on CVPR*, 2020, pp. 702–703.

[47] T. DeVries and G. W. Taylor, “Improved regularization of convolutional neural networks with cutout,” *arXiv preprint arXiv:1708.04552*, 2017.

[48] S. Zagoruyko and N. Komodakis, “Wide residual networks,” in *BMVC*. BMVC, 2016.

[49] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell, “Actionvlad: Learning spatio-temporal aggregation for action classification,” in *CVPR*, 2017, pp. 971–980.

[50] A. Krizhevsky, “Learning multiple layers of features from tiny images,” *Master’s thesis, University of Tront*, 2009.

[51] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, “Reading digits in natural images with unsupervised feature learning,” in *Workshop on NIPS*, 2011.

[52] A. Coates, A. Ng, and H. Lee, “An analysis of single-layer networks in unsupervised feature learning,” in *Proceedings of the 14th international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 215–223.

[53] A. Jaffe, B. Nadler, and Y. Kluger, “Estimating the accuracies of multiple classifiers without labeled data,” in *Artificial Intelligence and Statistics*. PMLR, 2015, pp. 407–415.

[54] E. Platanios, H. Poon, T. M. Mitchell, and E. J. Horvitz, “Estimating accuracy from unlabeled data: A probabilistic logic approach,” *NeurIPS*, vol. 30, pp. 4361–4370, 2017.

[55] P. A. Traganitis, A. Pages-Zamora, and G. B. Giannakis, “Blind multiclass ensemble classification,” *IEEE Transactions on Signal Processing*, vol. 66, no. 18, pp. 4737–4752, 2018.

# Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes] See in Line.5-7 in abstract and the last paragraph in Sec. 1.
   (b) Did you describe the limitations of your work? [Yes] See in Sec. 6.
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] Our paper conforms to the ethics review guidelines.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See major experimental settings and data in Sec. 4.1 and Sec. 4.2. Code would be included in the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We follow the data splits of FixMatch [4] in Sec. 4.3. For common hyperparameters, we use the default value in FixMatch [4]. While for characteristic hyperparameters, see in Sec. 4.5.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] The error bars are attached in Tab. 1, and Tab. 2.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See the last sentence of 4.1.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We have cited all of them.

(b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We will attach our codes and some of the pretrained model in supplemental matrerial.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] Datasets in our paper are publicly available

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]