Debiased Self-Training for Semi-Supervised Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Deep neural networks achieve remarkable performances on a wide range of tasks with the aid of large-scale labeled datasets. Yet these datasets are time-consuming and labor-exhaustive to obtain on realistic tasks. To mitigate the requirement for labeled data, self-training is widely used in semi-supervised learning by iteratively assigning pseudo labels to unlabeled samples. Despite its popularity, self-training is well-believed to be unreliable and often leads to training instability. Our experimental studies further reveal that the bias in semi-supervised learning arises from both the problem itself and the inappropriate training with potentially incorrect pseudo labels, which accumulates the error in the iterative self-training process. To reduce the above bias, we propose *Debiased Self-Training (DST)*. First, the generation and utilization of pseudo labels are decoupled by two parameter-independent classifier heads to avoid direct error accumulation. Second, we estimate the worst case of self-training bias, where the pseudo labeling function is accurate on labeled samples, yet makes as many mistakes as possible on unlabeled samples. We then adversarially optimize the representations to improve the quality of pseudo labels by avoiding the worst case. Extensive experiments justify that *DST* achieves an average improvement of 6.3% against state-of-the-art methods on standard semi-supervised learning benchmark datasets and 18.9% against FixMatch on 13 diverse tasks. Furthermore, DST can be seamlessly adapted to other self-training methods and help stabilize their training and balance performance across classes in both cases of training from scratch and finetuning from pre-trained models.

1 Introduction

Deep learning has achieved great success in many machine learning problems in the past decades, especially where large-scale labeled datasets are present. In real-world applications, however, manually labeling sufficient data is time-consuming and labor-exhaustive. To reduce the requirement for labeled data, semi-supervised learning (SSL) improves the data efficiency of deep models by learning from a few labeled samples and a large number of unlabeled samples [15, 25, 37, 6]. Among them, self-training is an effective approach to deal with the lack of labeled data. Typical self-training methods [25, 35] assign pseudo labels to unlabeled samples with the model's predictions and then iteratively train the model with these pseudo labeled samples as if they were labeled examples.

Although self-training has achieved great advances in benchmark datasets, they still exhibit large training instability and extreme performance imbalance across classes. For instance, the accuracy of FixMatch [35], one of the state-of-the-art self-training methods, fluctuates greatly when trained from scratch (see Figure 7). Though its performance will gradually recover after a sudden sharp drop, this is still not expected, since in most real-world applications, *pre-trained* models are more often adopted [12, 6, 19], and the performance of pre-trained models is difficult to recover after a drastic decline due to catastrophic forgetting [20]. Besides, although FixMatch improves the average accuracy, it also leads to the *Matthew effect*, *i.e.*, the accuracy of well-behaved categories is further increased

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

while that of poorly-behaved ones is decreased to nearly zero (see Figure 4). This is also not expected, since most machine learning models prefer performance balance across categories, even when the class imbalance exists in the training data [51]. The above findings are caused by the *bias* between the pseudo labeling function with the unknown target labeling function. Training with biased and unreliable pseudo labels has the chance to accumulate errors and ultimately lead to performance fluctuations. And for those poorly-behaved categories, the bias of the pseudo labels is worse and will be further enhanced as self-training progresses, ultimately leading to the Matthew effect.

To escape from the dilemma, we first delved into the bias issues arising from the self-training process and found that they can be briefly grouped into two kinds: (1) *Data bias* which is the bias inherent in the SSL tasks; (2) *Training bias* which refers to the bias increment brought by self-training with incorrect pseudo labels. Further, we present *Debiased Self-Training (DST)*, a novel approach to decrease the above bias in self-training. Specifically, to reduce the *training bias*, the classifier head is only trained with clean labeled samples and no longer trained with unreliable pseudo-labeled samples. In other words, the generation and utilization of pseudo labels are decoupled to mitigate bias accumulation and boost the model's tolerance to biased pseudo labels. Further, to decrease the *data bias* which cannot be calculated directly, we turn to estimate the worst case of training bias that implicitly reflects the data bias. Then we optimize the representations to decrease the worst-case bias and thereby improve the quality of pseudo labels.

The contributions of this work are summarized as follows: (1) We systematically analyze the problem and the causes of self-training bias in SSL. (2) We propose *DST*, a novel approach to mitigate the self-training bias and boost the stability and performance balance across classes, as well as a universal add-on for mainstream self-training methods. (3) We conduct extensive experiments and validate that *DST* achieves an average improvement of 6.3% against state-of-the-art methods on standard SSL benchmarks datasets and 18.9% against FixMatch on 13 diverse tasks.

2 Related Work

Self-training [46, 32, 15, 25] is a widely-used approach to utilize unlabeled data. Pseudo Label [25], one popular self-training method, iteratively generates pseudo labels and utilizes them with the same model. However, this paradigm suffers from the problem of confirmation bias [1], where the learner struggles to correct its own mistakes when learning from inaccurate pseudo labels. The bias issue is also mentioned in DebiasMatch [40] where they define the bias as the quantity imbalance for each category. Note that the bias in our paper refers to the deviation between the pseudo labeling function and the ground truth labeling function, which is a more essential problem existing in most self-training methods. Recent works mainly tackle this bias issue from the following two aspects.

Generate higher-quality pseudo labels. MixMatch [3] averages predictions from multiple augmentations as pseudo labels. ReMixMatch [2], UDA [43], and FixMatch [35] adopt confidence thresholds to generate pseudo labels on weakly augmented samples and utilize these pseudo-labels as annotations for strongly augmented samples. Dash [45] and FlexMatch [48] dynamically adjust the thresholds in a curriculum learning manner. Label Propagation methods [34, 18] assign pseudo labels with the density of the local neighborhood. Meta Pseudo Labels [30] proposes to generate pseudo labels with a meta learner. Different from the above methods that manually design specific criteria to improve the quality of pseudo labels, we estimate the *worst case* of self-training bias and adversarially optimize the representations to improve the quality of pseudo labels automatically.

Improve tolerance to inaccurate pseudo labels. To mitigate confirmation bias, existing methods maintain a mismatch between the generation and utilization of pseudo labels. Temporal Ensembling [24] and Mean Teacher [37] generate pseudo labels from the average of previous predictions or an exponential moving average of the model, respectively. Noisy Student [44] assigns pseudo labels by a fixed teacher from the previous round. Co-training [4], MMT [14] and Multi-head Tri-training [33] introduce multiple models or classifier heads and learn in an online mutual-teaching manner. In these methods, each classifier head is still trained with potentially incorrect pseudo labels generated by other heads. In contrast, in our method, *the classifier head that generates pseudo labels is never trained with pseudo labels*, leading to better tolerance to inaccurate pseudo labels (see Table 3).

Self-supervised methods [12, 16] are also used on unlabeled data to improve the model with few labeled samples, either in the pre-training stage [6] or in the downstream tasks [39]. However, the training of self-supervision usually relies on big data and heavy computation, which is not feasible in

most applications. Besides, although these methods avoid the use of unreliable pseudo labels, it is difficult for them to learn task-specific information from unlabeled data for better performance.

3 Analysis of Bias in Self-Training

In this section, we provide some analysis of where the bias in self-training comes from. Let $\mathcal{P}$ denote a distribution over input space $\mathcal{X}$. For classification with K classes, let P^k denote the class-conditional distribution of $\mathbf{x}$ conditioned on ground truth $f^*(\mathbf{x}) = k$. Assume that pseudolabeler f_{pl} is obtained via training a classifier on the n labeled data $\widehat{P}_n$. Let $\mathcal{M}(f_{\text{pl}}) \triangleq \{\mathbf{x} : f_{\text{pl}}(\mathbf{x}) \neq f^*(\mathbf{x})\}$ denote the mistaken pseudolabeled samples. The bias in the self-training refers to *the deviation between the learned decision hyperplanes and the true decision hyperplanes*, which can be measured by the fraction of incorrectly pseudolabeled samples in any classes $\mathcal{B}(f_{\text{pl}}) = \{P^k(\mathcal{M}(f_{\text{pl}}))\}_{k=1}^K$ [41]. By analyzing self-training bias under different training conditions, we have the following findings.

The sampling of labeled data will largely influence the self-training bias. As shown in Figure 1, when the data sampling is different, the accuracy of the same category may be very high or very low. The reason is that the distances between different data points and the true decision hyperplanes are not the same, with some supporting data points closer and others far away. When there are few labeled data, there may be a big difference in the distances between supporting data of each category and the true decision hyperplanes, thus the learned decision hyperplanes will be biased towards some categories.

The pre-trained representations also affect the self-training bias. Figure 2 shows that different pre-trained representations lead to different category bias, even if the pre-trained dataset and the downstream labeled dataset are both identical. One possible reason is that the representations learned by different pre-trained models focus on different aspects of the data [50]. Therefore, the same data could also have different distances to the decision hyperplanes in the representation level with different pre-trained models.

Training with pseudo labels aggressively in turn enlarges the self-training bias on some categories. Figure 3 shows that after training with pseudo labels (*e.g.*, using FixMatch), the performance gap for different categories greatly enlarges, with the accuracy of some categories increasing from 60%

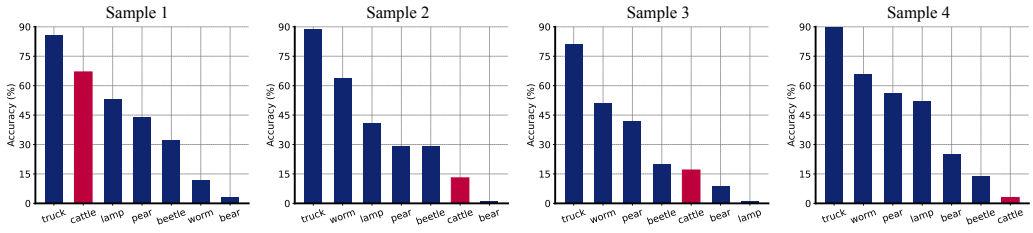


Figure 1: Effect of *data sampling*. Top-1 accuracy of 7 randomly selected categories when trained with different labeled data sampled from *CIFAR-100*. The same category (such as cattle) may have completely different accuracy in different samples. Following FixMatch [35], 4 labeled data are sampled for each category by default in our analysis.

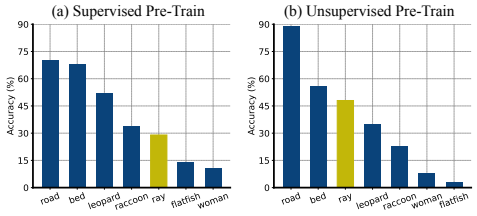


Figure 2: Effect of *pre-trained representations*. Accuracy of 7 randomly selected categories with different pre-trained models on *CIFAR-100*. Different pre-trained models show different category preferences.

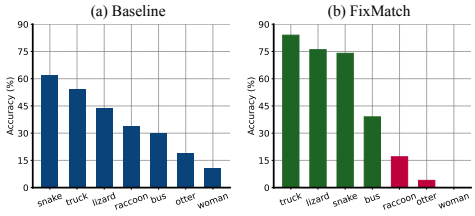


Figure 3: Effect of *self-training algorithm*. Accuracy of 7 randomly selected categories with different training methods on *CIFAR-100*. FixMatch largely increases the bias of poorly-behaved categories.

to 80% and that of some categories dropping from 15% to 0%. The reason is that for well-behaved categories, the pseudo labels are almost accurate, thus using them for training could further reduce the bias. Yet for many poorly-behaved categories, the pseudo labels are not reliable, and the common self-training mechanism that uses these incorrect pseudo labels to train the model will further increase the bias, and fail to correct it back in the follow-up training.

Based on the above observations, we divide the bias in self-training into two categories.

Data bias: the bias inherent in SSL tasks, such as the bias of sampling and pre-trained representations on unlabeled data. Formally, data bias is defined as $\mathcal{B}(f_{\text{pl}}(\widehat{P}_n, \psi_0)) - \mathcal{B}(f^*)$ (blue area in Fig. 4), where the pseudolabeler $f_{\text{pl}}(\widehat{P}_n, \psi_0)$ is obtained from a biased sampling $\widehat{P}_n$ with a biased parameter initialization ψ_0.

Training bias: the bias increment brought by some unreasonable training strategies. Formally, training bias is $\mathcal{B}(f_{\text{pl}}(\widehat{P}_n, \psi_0, \mathcal{S})) - \mathcal{B}(f_{\text{pl}}(\widehat{P}_n, \psi_0))$ (yellow area in Fig. 4) where $f_{\text{pl}}(\widehat{P}_n, \psi_0, \mathcal{S})$ is a pseudolabeler obtained with self-training strategy $\mathcal{S}$.

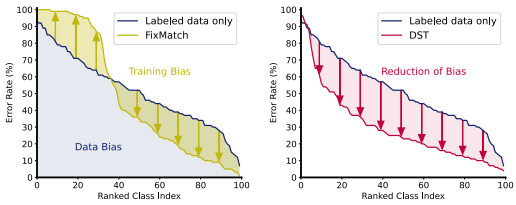


Figure 4: Error rate of pseudo labels in any classes on *CIFAR-100* (ResNet50, 4 labels per category). FixMatch decreases the bias on well-behaved categories while increasing that of poorly-behaved categories. In contrast, DST effectively balances the performance between different categories.

Next we will introduce how to reduce training bias and data bias in self-training (red area in Fig. 4).

4 Debiased Self-Training

In SSL, we have a labeled dataset $\mathcal{L} = \{(\mathbf{x}_i^l, y_i^l)\}_{i=1}^{n_l}$ of n_l labeled samples and an unlabeled dataset $\mathcal{U} = \{(\mathbf{x}_j^u)\}_{j=1}^{n_u}$ of n_u unlabeled samples, where the size of the labeled dataset is usually much smaller than that of the unlabeled dataset, *i.e.*, $n_l \ll n_u$. Denote ψ the feature generator, and h the task-specific head. The standard cross-entropy loss on weakly augmented labeled examples is

$$L_{\mathcal{L}}(\psi, h) = \frac{1}{n_l} \sum_{i=1}^{n_l} L_{\text{CE}}\big((h \circ \psi \circ \alpha)(\mathbf{x}_i^l), y_i^l\big), \tag{1}$$

where α is the weak augmentation function. Since there are few labeled samples, the feature generator and the task-specific head will easily over-fit, and typical SSL methods use these pseudo labels on plenty of unlabeled data to decrease the generalization error. Different SSL methods design different pseudo labeling function $\widehat{f}$ [25, 45, 31]. Take FixMatch [35] for an instance. FixMatch first generates predictions $\widehat{\mathbf{p}} = (h \circ \psi \circ \alpha)(\mathbf{x})$ on a weakly augmented version of given unlabeled images, and adopts a confidence threshold τ to filter out unreliable pseudo labels

$$\widehat{f}_{\psi,h}(\mathbf{x}) = \begin{cases} \arg\max \widehat{\mathbf{p}}, & \max \widehat{\mathbf{p}} \geq \tau, \\ -1, & \text{otherwise}, \end{cases} \tag{2}$$

where $\widehat{f}_{\psi,h}$ refers to the pseudo labeling by model $h \circ \psi$, hyperparameter τ specifies the threshold above which a pseudo label is retained and -1 indicates that this pseudo label is ignored in training. Then FixMatch utilizes selected pseudo labels to train on strongly augmented unlabeled images,

$$L_{\mathcal{U}}(\psi, h, \widehat{f}) = \frac{1}{n_u} \sum_{j=1}^{n_u} L_{\text{CE}}\big((h \circ \psi \circ \mathcal{A})(\mathbf{x}_j^u), \widehat{f}(\mathbf{x}_j^u)\big), \tag{3}$$

where $\widehat{f}$ is a notation of general pseudo labeling function and $\mathcal{A}$ is the strong augmentation function. As shown in Figure 5(a), the optimization objective for FixMatch is

$$\min_{\psi,h} L_{\mathcal{L}}(\psi, h) + \lambda L_{\mathcal{U}}(\psi, h, \widehat{f}_{\psi,h}), \tag{4}$$

where λ is the trade-off between the loss on labeled data and that on unlabeled data. FixMatch filters out low-confidence samples during the pseudo labeling process, yet two issues remain: (1) The

pseudo labels are generated and utilized by the same head, which leads to the training bias, *i.e.*, the errors of the model might be amplified as the self-training progresses. (2) When trained with extreme few labeled samples, the problem of unreliable pseudo labeling caused by data bias cannot be ignored anymore even with the confidence threshold mechanism. To tackle the above issues, we propose two important designs to decrease training bias and data bias in Section 4.1 and 4.2 respectively.

4.1 Generate and utilize pseudo labels independently

The training bias of FixMatch stems from the way of training on the pseudo labels generated by itself. To alleviate this bias, some methods turn to generate pseudo labels from a better teacher model, such as the moving average of the original model [37] in Figure 5(b) or the model obtained from the previous round of training [44] in Figure 5(c), and then utilize these pseudo labels to train both the feature generator ψ and the task-specific head h. However, there is still a tight relationship between the teacher model that generates pseudo labels and the student model that utilizes pseudo labels in the above methods, and the decision hyperplanes of the student model $h \circ \psi$ strongly depend on the biased pseudo labeling $\widehat{f}$. As a result, training bias is still large in the self-training process.

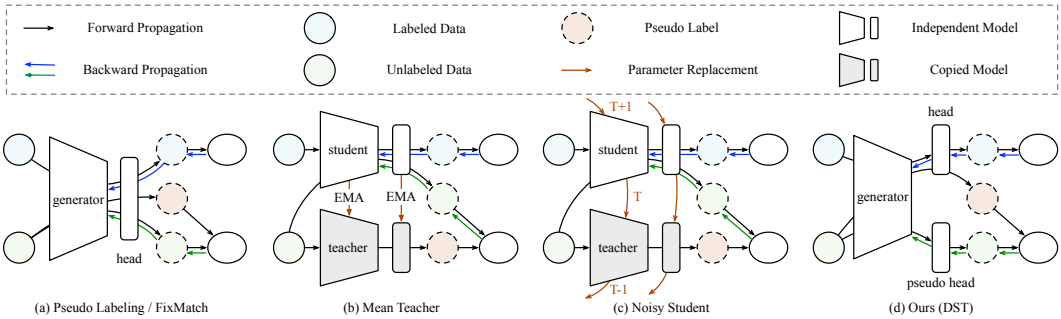


Figure 5: Comparisons on how different self-training methods generate and utilize pseudo labels. **(a)** Pseudo Labeling and FixMatch generate and utilize pseudo labels on the same model. **(b)** Mean Teacher generates pseudo labels from the Exponential Moving Average (EMA) of the current model. **(c)** Noisy Student generates pseudo labels from the teacher model which is obtained from the previous round of training. **(d)** DST generates pseudo labels from head h and utilizes pseudo labels on a parameter independent pseudo head h_{pseudo}.

To further decrease the training bias when utilizing the pseudo labels, we optimize the task-specific head h, only with the clean labels on $\mathcal{L}$ and without any unreliable pseudo labels from $\mathcal{U}$. To prevent the deep models from over-fitting to the few labeled samples, we still use pseudo labels, but only for learning a better representation. As shown in Figure 5(d), we introduce a pseudo head h_{pseudo}, which is connected to the feature generator ψ and only optimized with pseudo labels from $\mathcal{U}$. Then the training objective is

$$\min_{\psi,h,h_{\text{pseudo}}} L_{\mathcal{L}}(\psi, h) + \lambda L_{\mathcal{U}}(\psi, h_{\text{pseudo}}, \widehat{f}_{\psi,h}), \tag{5}$$

where the pseudo labels are generated by head h and utilized by a completely parameter *independent* pseudo head h_{pseudo}. Although h and h_{pseudo} are fed with features from the same backbone network, their parameters are independent, thus training the pseudo head h_{pseudo} with some wrong pseudo labels will not accumulate the bias of head h directly in the iterative self-training process. Note that the pseudo head h_{pseudo} is only responsible for gradient backpropagation to the feature generator ψ during training and will be discarded during inference, and thus will introduce no inference cost.

4.2 Reduce generation of erroneous pseudo labels

Section 4.1 presents a solution to reduce the training bias, yet the data bias still exists in the pseudo labeling $\widehat{f}$. As shown in Figure 6(a), due to the data bias, labeled samples of each class have different distances to the decision hyperplanes in the representation space, which leads to a deviation between the learned hyperplanes and the real decision hyperplanes, especially when the size of labeled samples

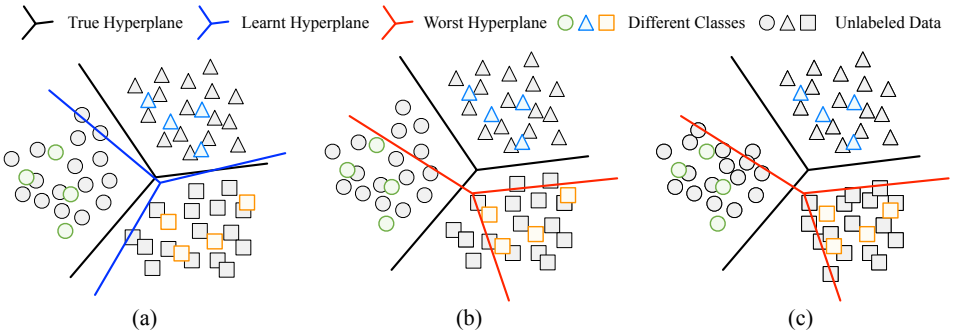


Figure 6: Concept explanations. **(a)** Shift between the hyperplanes learned on limited labeled data and the true hyperplanes. **(b)** The worst hyperplanes are hyperplanes that correctly distinguish labeled samples while making as many mistakes as possible on unlabeled samples. **(c)** Feature representations are optimized to improve the performance of the worst hyperplanes.

is very small. As a result, pseudo labeling $\widehat{f}$ is very likely to generate incorrect pseudo labels on unlabeled data points that are close to these biased decision hyperplanes. And our objective now is to optimize the feature representations to reduce the data bias, and finally improve the quality of pseudo labels.

Since we have no labels for $\mathcal{U}$, we cannot directly measure and thereby reduce data bias. Yet training bias has some correlations with data bias. Recall in Section 4.1, the task-specific head h is only optimized with clean labeled data, since optimization with incorrect pseudo labels will push the learned hyperplanes in a more biased direction and lead to the training bias. Therefore, training bias can be considered as the accumulation of data bias with inappropriate utilization of pseudo labels, which is training algorithm dependent. And the worst training bias that can be achieved by some self-training methods is a good measure of data bias. Specifically, the worst training bias corresponds to the worst possible head h' learned by pseudo labeling, such that h' predicts correctly on all the labeled samples $\mathcal{L}$ while making as many mistakes as possible on unlabeled data $\mathcal{U}$,

$$h_{\text{worst}}(\psi) = \arg\max_{h'} L_{\mathcal{U}}(\psi, h', \widehat{f}_{\psi,h}) - L_{\mathcal{L}}(\psi, h'), \tag{6}$$

where the mistakes of h' on unlabeled data are estimated by its discrepancy with the current pseudo labeling function $\widehat{f}$. Equation 6 aims to find the worst-case of task-specific head h that might be learned in the future when trained with pseudo labeling on the current feature generator ψ and the current data sampling. It is also the *worst hyperplanes* as shown in Figure 6(b), which deviates as much as possible from the currently learned hyperplanes while ensuring that all labeled samples are correctly distinguished. Note that Equation 6 measures the degree of data bias, which depends on the feature representations generated by ψ, thus we can adversarially optimize feature generator ψ to indirectly decrease the data bias,

$$\min_{\psi} L_{\mathcal{U}}(\psi, h_{\text{worst}}(\psi), \widehat{f}_{\psi,h}) - L_{\mathcal{L}}(\psi, h_{\text{worst}}(\psi)). \tag{7}$$

As shown in Figure 6(c), Equation 7 encourages the feature of unlabeled samples to be distinguished correctly even by the worst hyperplanes, i.e., be generated far away from the current hyperplanes, thereby reducing the data bias in feature representations.

5 Experiments

Following [35, 45], we evaluate the proposed DST with random initialization on common SSL datasets, including *CIFAR-10* [23], *CIFAR-100* [23], *SVHN* [27] and *STL-10* [9]. Following [39], we also evaluate DST with both supervised pre-trained models and unsupervised pre-trained models on 11 downstream tasks, including **(1)** superordinate-level object classification: *CIFAR-10* [23], *CIFAR-100* [23], *Caltech-101* [13]; **(2)** fine-grained object classification: *Food-101* [5], *CUB-200-2011* [38], *Stanford Cars* [22], *FGVC Aircraft* [26], *OxfordIIIT Pets* [29], *Oxford Flowers* [28]; **(3)** texture classification: *DTD* [8]; **(4)** scene classification: *SUN397* [42]. The complete training dataset size ranges from $2,040$ to $75,750$ and the number of classes ranges from 10 to 397. Following [21], we

report mean accuracy per-class on *Caltech-101*, *FGVC Aircraft*, *OxfordIIIT Pets*, *Oxford Flowers*, and top-1 accuracy for other datasets. Following [35], we construct a labeled subset with 4 labels per category to verify the effectiveness of DST in extremely label-scarce settings. To make a fair comparison, we keep the labeled subset for each dataset the same throughout our experiments.

For experiments with random initialization, we follow [35] and adopt Wide ResNet variants [47]. For experiments with pre-trained models, we adopt ResNet50 [17] with an input size of 224×224 and pre-trained on ImageNet [11]. We adopt MoCo v2 [7] as unsupervised pre-trained models. We compare our method with many state-of-the-art SSL methods, including Pseudo Label [25], Π-Model [24], Mean Teacher [37], UDA [43], MixMatch [3], ReMixMatch [2], FixMatch [35], Dash [45], Self-Tuning [39], FlexMatch [48] and DebiasMatch [40].

When training from scratch, we adopt the same hyperparameters as FixMatch [35], with learning rate of 0.03, mini-batch size of 512. For other experiments, we use SGD with momentum 0.9 and weight-decay in $\{0.0005, 0.001\}$, learning rates in $\{0.001, 0.003, 0.01, 0.03\}$. The mini-batch size is set to 64 following [36]. For each image, we first apply random-resize-crop and then use RandAugment [10] for strong augmentation $\mathcal{A}$ and random-horizontal-flip for weak augmentation α. The trade-off hyperparameter λ is set to 1 for all datasets. More details on hyperparameter selection can be found in Appendix A.2. Each experiment is repeated three times with different random seeds. *We submit our code in the supplemental material and will release the codebase for all the methods.*

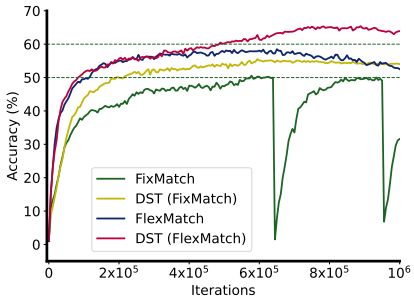
5.1 Main results

Table 1 shows that DST yields consistent improvement on all tasks. On the challenging *CIFAR-100* and *STL-10* tasks, DST boosts the accuracy of FixMatch and FlexMatch by **8.3%** and **10.7%**, respectively. Figure 7 depicts the top-1 accuracy during the training procedure on *CIFAR-100*. We observe that the performance of FixMatch suffers from significant fluctuations during training. In contrast, the accuracy of DST (FixMatch) increases steadily and surpasses the best accuracy of FixMatch by **10.9%**, relatively. Note that the accuracy of FlexMatch also drops by over 6% in the final stages of training while DST (FlexMatch) suffers from a much smaller drop by reducing erroneous pseudo labels during the self-training process. Besides, DST also improves the performance balance across categories (see Appendix B.2).

Table 1: Top-1 accuracy on *CIFAR-10/100*, *SVHN*, and *STL-10* datasets (Wide ResNet, train from scratch, 4 labels per category).

Method	CF-10	CF-100	SVHN	STL-10	Avg
Psuedo Label [25]	25.4	12.6	25.3	25.3	22.2
MixMatch [3]	52.6	32.4	57.5	45.1	46.9
UDA [43]	71.0	40.7	47.4	62.6	55.4
ReMixMatch [2]	80.9	55.7	96.6	64.0	74.3
Dash [45]	86.8	55.2	**97.0**	64.5	75.9
FixMatch [35]	87.2	50.6	96.5	67.1	75.4
DST (FixMatch)	**89.3**	**56.1**	96.7	**71.0**	**78.3**
FlexMatch [48]	94.7	59.5	89.6	71.3	78.8
DST (FlexMatch)	**95.0**	**65.4**	**94.2**	**79.6**	**83.6**

Figure 7: Top-1 accuracy on *CIFAR-100* (train from scratch, 4 labels per category). DST accelerates convergence and improves stability.



5.2 Transfer from a pre-trained model

Supervised pre-training. Table 2 reveals that typical self-training methods, e.g. FixMatch, lead to relatively mild improvements with supervised pre-trained models, which is consistent with previous findings [36, 39]. In contrast, incorporating DST into FixMatch significantly boosts the performance and surpasses FixMatch by **19.9%** on all datasets on average, relatively. Compared with recent advances, DST outperforms FlexMatch in 10 out of 11 tasks and achieves comparable accuracy on *SUN397*. DST also outperforms DebiasMatch in 10 out of 11 tasks and yields a **6.3%** improvement on average, relatively. With a pre-trained model, self-training has better training stability. Yet once the performance degradation occurs, the process is also irreversible (Appendix B.1), partly due to the catastrophic forgetting of pre-trained representation. Also, self-training suffers from a more

Table 2: Comparison between DST and various baselines (ResNet50, supervised and unsupervised pre-trained, 4 labels per category). ↓ indicates a performance degradation compared with the baseline.

		Caltech101	CIFAR-10	CIFAR-100	SUN397	DTD	Aircraft	CUB	Flowers	Pets	Cars	Food101	Average
Supervised	Baseline	81.4	65.2	48.2	39.9	47.7	25.4	46.5	85.2	78.1	33.3	33.8	53.2
	Pseudo Label [25]	86.3	83.3	54.7	41.0	50.2	27.2	54.3	92.3	87.8	41.4	38.0	59.7
	Π-Model [24]	83.5	73.1	49.2	39.7↓	50.3	24.3↓	47.1	90.7	82.2	30.9	33.9	55.0
	Mean Teacher [37]	83.7	82.1	56.0	37.9↓	51.6	30.7	49.6	91.0	82.8	39.1	40.3	58.6
	UDA [43]	85.8	83.6	54.7	41.3	49.0	27.1	52.1	92.0	83.1	45.6	41.7	59.6
	FixMatch [35]	86.3	84.6	53.1	41.3	48.6	25.2↓	52.3	93.2	83.7	46.4	37.1	59.3
	Self-Tuning [39]	87.2	76.0	57.1	41.8	50.7	35.2	58.9	92.6	86.6	58.3	41.9	62.4
	FlexMatch [48]	87.1	89.0	63.4	**48.3**	52.5	34.0	54.9	94.5	88.3	57.5	49.5	65.4
	DebiasMatch [40]	88.6	91.0	65.7	46.6	52.4	37.5	58.6	**95.6**	86.4	60.5	53.5	66.9
	DST (FixMatch)	**89.6**	**94.9**	**70.4**	48.1	**53.5**	**43.2**	**68.7**	94.8	**89.8**	**71.0**	**58.5**	**71.1**
Unsupervised	Baseline	79.5	66.6	46.5	38.1	47.9	28.7	37.5	87.7	60.0	38.1	32.9	51.2
	Pseudo Label [25]	86.2	70.8	49.8	38.6	50.0	26.6↓	41.8	93.0	68.4	37.3↓	32.8↓	54.1
	Π-Model [24]	80.1	76.2	44.8	37.8↓	50.0	23.5↓	31.6↓	93.1	62.8	25.6↓	30.4↓	50.5
	Mean Teacher [37]	80.4	80.8	51.3	34.2↓	48.8	33.8	41.6	92.9	67.0	50.5	39.1	56.4
	UDA [43]	85.0	87.4	53.6	42.3	46.2↓	35.7	41.4	94.1	69.3	51.5	39.3	58.7
	FixMatch [35]	83.1	82.2	51.4	39.2	43.9↓	30.1	36.8↓	94.3	65.7	48.6	36.8	55.6
	Self-Tuning [39]	81.6	63.6↓	47.8	38.8	45.5↓	31.4	41.6	91.0	66.9	52.0	34.0	54.0
	FlexMatch [48]	86.4	**96.7**	60.2	45.3	53.9	42.0	49.2	**95.8**	72.9	69.0	37.5	64.4
	DebiasMatch [40]	86.4	96.3	66.3	44.5	53.9	44.8	51.2	95.4	70.9	**72.5**	53.6	66.9
	DST (FixMatch)	**90.1**	95.0	**68.2**	**46.8**	**54.2**	**47.7**	**53.6**	95.6	**75.4**	72.0	**57.1**	**68.7**

severe performance imbalance across classes (Appendix B.2). DST effectively tackles these issues, indicating the importance of reducing bias.

Unsupervised pre-training. Table 2 shows that with unsupervised pre-trained models, more methods suffer from performance degradation after self-training on the unlabeled data. The difficulty comes from that the unsupervised pre-training task has a larger task discrepancy with the downstream classification tasks than the supervised pre-training task. Thus, the representations learned by unsupervised pre-trained models usually exhibit stronger data bias, and inappropriate usage of pseudo labels will lead to rapid accumulation errors and increase the training bias. By eliminating training bias and reducing data bias, DST brings improvement on all datasets and relatively outperforms FixMatch by **23.5%** on average, superior to FlexMatch and DebiasMatch in 9 and 10 tasks, respectively.

5.3 Ablation studies

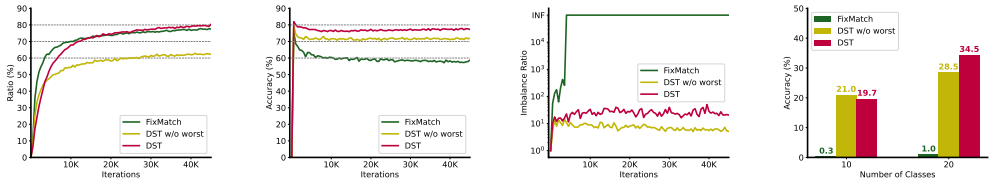
We examine the design of our method on *CIFAR-100* in Table 3 and have the following findings. **(1)** Compared with *Mutual Learning* [49, 14], where two heads provide pseudo labels to each other, the independent mechanism in our method where one head is only responsible for generating pseudo labels and the other head only uses them can better reduce the training bias. **(2)** A nonlinear pseudo head is always better than a linear pseudo head. We conjecture that nonlinear projection can reduce the degeneration of representation with biased pseudo labels. **(3)** The worst-case estimation of pseudo labeling improves the performance by large margins.

Table 3: Ablation study on *CIFAR-100* with different pre-trained models (4 labels per category).

Method	Multiple Heads	Linear Pseudo Head	Nonlinear Pseudo Head	Worst Case Estimation	Supervised Pre-training	Unsupervised Pre-training
FixMatch					53.1	51.4
Mutual Learning	✓				53.4	52.5
DST w/o worst	✓	✓			58.2	59.0
DST w/o worst	✓		✓		60.6	60.9
DST	✓		✓	✓	**70.4**	**68.2**

5.4 Analysis

To further investigate how DST improves pseudo labeling and self-training performance, we conduct some analysis on *CIFAR-100*. For simplicity, we only give the results with supervised pre-trained models. More comparisons can be found in Appendix B.4.



(a) Quantity (b) Quality (c) Quantity of bad classes (d) Quality of bad classes

Figure 8: The quantity and quality of pseudo labels on *CIFAR-100* (ResNet50, supervised pre-trained).

DST improves both the quantity and quality of pseudo labels. As shown in Figure 8(a) and 8(b), FixMatch exploits unlabeled data *aggressively*, on average producing more than 70% pseudo labels during training. But the cost is that the accuracy of pseudo labels continues to drop, eventually falling below 60%, which is consistent with our motivation in Section 3 that inappropriate utilization of pseudo labels will in turn enlarges the training bias. On the contrary, the accuracy of pseudo labels in DST suffers from a smaller drop. Rather, it keeps rising afterward and exceeds 70% throughout the training. Besides, DST generates more pseudo labels in the later stages of training.

DST generates better pseudo labels for poorly-behaved classes. To measure the quantity of pseudo labels on poorly-behaved classes, we calculate the class imbalance ratio I on a class-balanced validation set, $I = \max_c N(c)/\min_{c'} N(c')$, where $N(c)$ denotes the number of predictions that fall into category c. As shown in Figure 8(c), the class imbalance ratio of FixMatch rises rapidly and reaches infinity after 5000 iterations, indicating that the model completely ignores those poorly-learned classes. To measure the quality of pseudo labels on poorly-behaved classes, we calculate the average accuracy of 10 or 20 worst-behaved classes in Figure 8(d). The average accuracy on the worst 20 classes of FixMatch is only **1.0%**. By reducing training bias with the pseudo head, data bias with the worst-case estimation, the average accuracy balloons to **28.5%** and **34.5%**, respectively.

5.5 DST as a general add-on

To explore incorporating DST into different state-of-the-art self-training methods, we consider three mainstream paradigms of self-training shown in Figure 5, including FixMatch [35], Mean Teacher [37] and Noisy Student [44], as well as an incremental work FlexMatch [48]. Implementation details of DST versions of these methods can be found in Appendix A.3. Table 4 compares the original and DST versions of these methods on *CIFAR-100* with both supervised pre-trained and unsupervised pre-trained models. Results show that the proposed DST yields large improvement on all these self-training methods, indicating that self-training bias widely exists in existing self-training methods and DST can serve as a universal add-on to reduce bias.

Table 4: DST as a general add-on to 4 self-training methods on *CIFAR-100*.

Pre-training		Supervised		Unsupervised	
Label Amount		400	1000	400	1000
Mean Teacher	Base	56.0	67.0	51.3	63.5
	DST	**62.7**	**70.7**	**60.7**	**69.3**
Noisy Student	Base	52.8	64.3	55.6	65.8
	DST	**68.9**	**74.8**	**66.6**	**75.2**
FixMatch	Base	53.1	67.8	51.4	64.2
	DST	**70.4**	**75.6**	**68.2**	**76.8**
FlexMatch	Base	63.4	71.2	60.2	71.1
	DST	**70.8**	**77.3**	**68.9**	**77.5**

6 Conclusion

To mitigate the requirement for labeled data, pseudo labels are widely used on the unlabeled data, yet they suffer from severe confirmation bias. In this paper, we systematically delved into the bias issues and present *DST*, a novel approach to decrease bias in self-training. Experimentally, *DST* achieves state-of-the-art performance on 13 tasks and can serve as a universal add-on.

References

[1] Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *IJCNN*, 2020.

[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. In *ICLR*, 2020.

[3] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *NeurIPS*, 2019.

[4] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, 1998.

[5] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *ECCV*, 2014.

[6] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. In *NeurIPS*, 2020.

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

[8] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *CVPR*, 2014.

[9] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, 2011.

[10] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPR*, 2020.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[13] Li Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVPR*, 2004.

[14] Yixiao Ge, Dapeng Chen, and Hongsheng Li. Mutual mean-teaching: Pseudo label refinery for unsupervised domain adaptation on person re-identification. In *ICLR*, 2020.

[15] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *NeurIPS*, 2005.

[16] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[18] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label propagation for deep semi-supervised learning. In *CVPR*, 2019.

[19] Junguang Jiang, Yang Shu, Jianmin Wang, and Mingsheng Long. Transferability in deep learning: A survey, 2022.

[20] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 2017.

[21] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *CVPR*, 2019.

[22] Jonathan Krause, Jia Deng, Michael Stark, and Li Fei-Fei. Collecting a large-scale dataset of fine-grained cars. In *FGVC*, 2013.

[23] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Technical report, University of Toronto*, 2009.

[24] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *ICLR*, 2017.

[25] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML*, 2013.

[26] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

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

[28] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *ICVGIP*, 2008.

[29] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *CVPR*, 2012.

[30] Hieu Pham, Zihang Dai, Qizhe Xie, and Quoc V Le. Meta pseudo labels. In *CVPR*, 2021.

[31] Mamshad Nayeem Rizve, Kevin Duarte, Yogesh S Rawat, and Mubarak Shah. In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning. In *ICLR*, 2021.

[32] Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. Semi-supervised self-training of object detection models. In *WACV*, 2005.

[33] Sebastian Ruder and Barbara Plank. Strong baselines for neural semi-supervised learning under domain shift. In *ACL*, 2018.

[34] Weiwei Shi, Yihong Gong, Chris Ding, Zhiheng MaXiaoyu Tao, and Nanning Zheng. Transductive semi-supervised deep learning using min-max features. In *ECCV*, 2018.

[35] 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. In *NeurIPS*, 2020.

[36] Jong-Chyi Su, Zezhou Cheng, and Subhransu Maji. A realistic evaluation of semi-supervised learning for fine-grained classification. In *CVPR*, 2021.

[37] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*, 2017.

[38] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. *Technical Report CNS-TR-2011-001, California Institute of Technology*, 2011.

[39] Ximei Wang, Jinghan Gao, Mingsheng Long, and Jianmin Wang. Self-tuning for data-efficient deep learning. In *ICML*, 2021.

[40] Xudong Wang, Zhirong Wu, Long Lian, and Stella X Yu. Debiased learning from naturally imbalanced pseudo-labels for zero-shot and semi-supervised learning. In *CVPR*, 2022.

[41] Colin Wei, Kendrick Shen, Yining Chen, and Tengyu Ma. Theoretical analysis of self-training with deep networks on unlabeled data. In *ICLR*, 2021.

[42] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010.

[43] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Unsupervised data augmentation for consistency training. In *NeurIPS*, 2020.

[44] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *CVPR*, 2020.

[45] 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 *ICML*, 2021.

[46] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL*, 1995.

[47] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

[48] 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 *NeurIPS*, 2021.

[49] Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep mutual learning. In *CVPR*, 2018.

[50] Nanxuan Zhao, Zhirong Wu, Rynson W. H. Lau, and Stephen Lin. What makes instance discrimination good for transfer learning? In *ICLR*, 2021.

[51] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. In *PAMI*, 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]
   (b) Did you describe the limitations of your work? [No]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
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] We include the code in the supplemental material and the data is publicly available.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5 and Appendix A.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] To be consistent with previous paper. we report the average performance with 3 seeds.
   (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] We include the type of resources in Appendix A.
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 cite the creators in Section 5.
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No] We do not use new assets.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] All datasets are public.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] All datasets are legal.
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]