Weighted Distillation with Unlabeled Examples

Anonymous Author(s)
Affiliation
Address
email

Abstract

Distillation with unlabeled examples is a popular and powerful method for training deep neural networks in settings where the amount of labeled data is limited: A large "teacher" neural network is trained on the labeled data available, and then it is used to generate labels on an unlabeled dataset (typically much larger in size). These labels are then utilized to train the smaller "student" model which will actually be deployed. Naturally, the success of the approach depends on the quality of the teacher's labels, since the student could be confused if trained on inaccurate data. This paper proposes a principled approach for addressing this issue based on an importance reweighting scheme tailored to the distillation training paradigm. Our method is hyper-parameter free, data-agnostic, and simple to implement. We demonstrate significant improvements on popular academic datasets when compared to conventional distillation with unlabeled examples. We also accompany our results with a theoretical analysis which rigorously justifies the performance of our method in certain settings.

1 Introduction

In many modern applications of deep neural networks, where the amount of labeled data for training is limited, *distillation with unlabeled data* [5, 14] has been enormously successful. In this two-stage training paradigm a larger and more sophisticated "teacher model" — typically non-deployable for the purposes of the application — is trained to learn from the limited amount of available training data in the first stage. In the second stage, the teacher model is used to generate labels on an unlabeled dataset, which is usually much larger in size than the original dataset the teacher itself was trained on. These labels are then utilized to train the "student model", namely the model which will actually be deployed.

While this has proven to be a very powerful approach in practice, its success depends on the quality of labels provided by the teacher model. Indeed, often times the teacher model generates inaccurate labels on a non-negligible portion of the unlabeled dataset, confusing the student. As overparameterized deep neural networks are susceptible to overfitting to corrupted labels [46], training the student on the teacher's noisy labels can lead to degradation in its generalization performance.

We address this shortcoming in an elegant and mathematically optimal way – given our model for the noise introduced by the teacher, there is a principled way of reweighting training examples to produce an unbiased estimate of the student's clean objective. From a practical standpoint, this produces a fully "plug-and-play" distillation technique which adds minimal implementation overhead.

The idea of compressing a teacher model into a smaller student model by matching the predictions of the teacher was initially introduced by Buciluǎ, Caruana and Niculescu-Mizil [4] and, since then, variations of this method [14, 19, 26, 30, 35, 38, 45] have been applied in a wide variety of contexts [32, 44, 49]. (Notably, some of these applications go beyond compression — see e.g. the method of Pseudo-Labels [5, 42, 44, 50] for reference.) In the simplest form of the method [19], and

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

using classification as a canonical example, the labels produced by the teacher are one-hot vectors that represent the class which has the maximum predicted probability — this method is often referred to as "hard"-distillation. More generally, Hinton et. al. [5, 14] have shown that it is often beneficial to train the student so that it minimizes the cross-entropy (or KL-divergence) with the probability distribution produced by the teacher while also potentially using a temperature higher than 1 in the softmax of both models ("soft"-distillation). (Temperatures higher than 1 are used in order to emphasize the difference between the probabilities of the classes with lower likelihood of corresponding to the correct label according to the teacher model.)

The main contribution of this work is a principled method for improving distillation with unlabeled examples by reweighting the loss function of the student. That is, we assign importance weights to the examples labeled by the teacher so that each weight reflects (i) how likely it is that the teacher has made an inaccurate prediction regarding the label of the example and (ii) how "distorted" the unweighted loss function we use to train the student is (measured with respect to using the ground-truth label for the example instead of the teacher's label). More concretely, our importance reweighting strategy is inspired by considering and analyzing a certain robust supervised learning setting, which is designed to model the behavior of the teacher in distillation. In this setting, we are able to come up with a closed-form solution for weights that "de-noise" the objective in the sense that (on expectation) they simulate having access to clean labels. Our idea now is to view the teacher in distillation as the source of noise in the latter setting. Crucially, we empirically observe that the key characteristics of the induced "noisy" distribution can be effectively estimated through a small validation dataset. In particular, we use the validation dataset to learn a map that takes as input the teacher's and student's "confidence" for the label of a certain example and outputs estimates for the quantities mentioned in items (i) and (ii) above. Finally, we plug in these estimates to our closed-form solution for the weights so that, overall, we obtain an automated way of computing the importance weights assignment in practice. A detailed description of our method can be found in Section 2.3.

Our main findings and contributions can be summarized as follows:

- We propose a *principled and hyperparameter-free* importance reweighting method for knowledge distillation with unlabeled examples. The method is efficient, data-agnostic and simple to implement.
- We present extensive experimental results which show that our method provides significant improvements when evaluated in standard benchmarks.
- Our importance reweighting technique comes with provable guarantees: (i) it is information-theoretically optimal; and (ii) under standard assumptions SGD optimization of the reweighted objective learns a solution with nearly optimal generalization.

1.1 Related work

As we have already discussed, the main conceptual contribution of our work is viewing the teacher model in distillation with unlabeled examples as the source of noise in a certain robust supervised learning setting. Naturally, one may wonder what is the relationship between our method and works from the vast literature of learning with noisy data (e.g. [7, 9, 18, 28, 31]). The answer is that our method *does not attempt to solve the generic "learning with noisy data"* problem, which is a highly non-trivial task in the absence of assumptions (both in theory and in practice). Instead, it focuses on the setting of distillation with unlabeled examples and *exploits its structure*. As a concrete example, while researchers have proposed importance reweighting schemes [3, 16, 18, 21, 25, 34] for the generic "learning with noisy data" problem, these are recipes agnostic to the source of the noise (or they make very conservative assumptions about it) so that the resulting weighting scheme is widely applicable. On the other hand, our method is *distillation-specific* and makes use of the fact that the noise in the data comes from a teacher model in two major ways: (i) the adversarial distribution defined in Section 2.2 models naturally the behavior of the teacher model, but it is certainly too specific to capture the generic learning with noisy data problem; (ii) when estimating our weights we are able to exploit the access we have to the teacher's confidence for the label of each specific example. Additionally, works from the learning with noisy data literature typically assume that the training dataset consists of corrupted *hard* labels, and often times their proposed method is not compatible with *soft*-distillation (in the sense that they degrade its performance, making it much less effective) — see e.g. [27] for a study of this phenomenon in the case of label smoothening [23, 39].

Finally, there exist methods in the literature of semi-supervised learning that propose (often times non-trivial) modifications to the standard distillation training paradigm, such as training the teacher and student simultaneously [30], training additional networks [16, 20], using data augmentation such as mixup [2, 47], using robust loss functions [22, 24, 33, 43, 48] and adjusting the network architecture [10, 11, 40, 41]. We do not compare our method with these works as our goal is to propose a principled, efficient, "plug-and-play" method for improving standard distillation with unlabeled examples — which can be crucial for large-scale applications where deviating from the standard distillation paradigm is often not an option. In fact, our method is generic enough so that it can be combined with these approaches in a straightforward way.

1.2 Organization of the paper.

In Section 2 we present our method in detail, while in Section 3 we present our key experimental results. In Section 4 we discuss the theoretical aspects of our work. In Section 5 we summarize our results, we discuss the benefits and limitations of our method, and future work. Finally, we present extended experimental and theoretical results in the Supplementary Material.

2 Weighted distillation with unlabeled examples

In this section we present our method. In Section 2.1 we review multiclass classification, in Section 2.2 we introduce the robust supervised learning setting which inspired our approach and, finally, in Section 2.3 we describe our method in detail.

2.1 Multiclass classification

In multiclass classification, we are given a training sample $S = \{(x_1, y_1), (x_2, y_2), \dots (x_n, y_n)\}$ drawn from $\mathbb{P}^n$, where $\mathbb{P}$ is an unknown distribution over instances $\mathcal{X}$ and labels $\mathcal{Y} = [L] = \{1, 2, \dots, L\}$. Our goal is to learn a predictor $f : \mathcal{X} \to \mathbb{R}^L$, namely to minimize the *risk* of f. The latter is defined as the expected loss of f:

$$R(f) = \mathbb{E}[\ell(y, f(x))] \tag{1}$$

where (x, y) is drawn from $\mathbb{P}$, and $\ell : [L] \times \mathbb{R}^L \to \mathbb{R}_+$ is a loss function such that, for a label $y \in [L]$ and prediction vector $f(x) \in \mathbb{R}^L$, $\ell(y, f(x))$ is the loss incurred for predicting $f(x)$ when the true label is y. The most common way to approximate the risk of a predictor f is via the so-called *empirical risk*:

$$R_S(f) = \frac{1}{|S|} \sum_{(x,y) \in S} \ell(y, f(x)). \tag{2}$$

That is, given a hypothesis class of predictors $\mathcal{F}$, our goal is typically to find $\min_{f \in \mathcal{F}} R_S(f)$ as a way to estimate $\min_{f \in \mathcal{F}} R(f)$.

2.2 Learning under adversarial noise

As we have already discussed, the main drawback of distillation with unlabeled examples is essentially that the empirical risk minimizer corresponding to the dataset labeled by the teacher cannot be trusted, since the teacher may generate inaccurate labels. To quantify this phenomenon and guide our algorithmic design, we consider the following simple but natural robust supervised learning scenario.

Let $\mathbb{P}$ be an unknown distribution over instances $\mathcal{X}$. We assume the existence of a *ground-truth classifier* so that each $x \in \mathcal{X}$ is associated with a ground-truth label $f_{\text{true}}(x) \in [L] = \{1, 2, \dots, L\}$. In other words, a clean labeled example is of the form $(x, f_{\text{true}}(x)) \sim \mathbb{P}$. Additionally, we consider an adversary that given a sample $x \sim \mathbb{D}$, outputs a "corrupted" label $y_{\text{adv}}(x)$ with probability $p(x)$, and the ground-truth label $f_{\text{true}}(x)$ with probability $1 - p(x)$. Let $\mathbb{D}$ denote the induced adversarial distribution over instances and labels.

It is not hard to see that the empirical risk with respect to a predictor f and sample from $\mathbb{D}$ is *not* an unbiased estimator of the risk

$$R(f) = \mathbb{E}_{x \sim \mathbb{P}}[\ell(f_{\text{true}}(x), f(x))] \tag{3}$$

— see Proposition 2.1. On the other hand, the following *weighted* empirical risk (5) is indeed an unbiased estimator of $R(f)$. For each $x \in \mathcal{X}$ let

$$w_f(x) = \frac{1}{1 + p(x)\,(\mathrm{distortion}_f(x) - 1)}, \quad \text{where} \quad \mathrm{distortion}_f(x) = \frac{\ell(y_{\mathrm{adv}}(x), f(x))}{\ell(f_{\mathrm{true}}(x), f(x))}, \tag{4}$$

and define

$$R_S^w = \frac{1}{|S|} \sum_{(x,y)\in S} w_f(x)\,\ell(y, f(x)), \tag{5}$$

where $S = \{(x_i, y_i)\}_{i=1}^n \sim \mathbb{D}^n$. Observe that the weight for each instance x depends on (i) how likely it is the adversary corrupts its label; and on (ii) how this corrupted label "distorts" the loss we observe at instance x. In the following proposition we establish that the standard (unweighted) empirical risk with respect to distribution $\mathbb{D}$ and a predictor f is a biased estimator of the risk of f under the clean distribution $\mathbb{P}$, while the weighted empirical risk (5) is an unbiased one.

Proposition 2.1 (Debiasing Weights)**.** *Let $S = \{(x_i, y_i)\}_{i=1}^n \sim \mathbb{D}^n$ be a sample from the adversarial distribution. Defining* $\mathrm{Bias}(f) = \mathbb{E}_{x\sim\mathbb{P}}\left[p(x) \cdot (\mathrm{distortion}_f(x) - 1)\right]$ *we have:*

$$\text{“reweighted”} \quad \mathbb{E}[R_S^w(f)] = R(f) \qquad \text{vs} \qquad \text{“standard”} \quad \mathbb{E}[R_S(f)] = R(f) + \mathrm{Bias}(f)\,.$$

Notice that, as expected, the bias of the unweighted predictor is a function of the "power" of the adversary, i.e., a function of how often they can corrupt the label of an instance, and the "distortion" this corruption causes to the loss we observe. The proof of Proposition 2.1 follows from simple, direct calculations and it can be found in the Supplementary Material.

Intuitively, given a sufficiently large sample $S \sim \mathbb{D}^n$, optimizing an unbiased estimator for the risk should provide a better approximation for $\min_{f\in F} R(f)$ compared to optimizing an estimator with constant bias. We formalize this intuition in Section 4 and in the Supplementary Material.

2.3 Our method

We consider the standard setting for distillation with unlabeled examples where we are given a dataset $S_\ell = \{(x_i, y_i)\}_{i=1}^m$ of m labeled examples from an unknown distribution $\mathbb{P}$, and a dataset $S_u = \{x_i\}_{i=m+1}^{m+n}$ of n unlabeled examples — typically, $n \geq m$. We also assume the existence of a (small) clean validation dataset $S_v = \{(x_i, y_i)\}_{i=m+n+1}^{i=m+n+q}$ of size q. Finally, let $\ell : \mathbb{R}^L \times \mathbb{R}^L \to \mathbb{R}_+$ be a loss function that takes as input two vectors over the set of labels $[L]$. We describe our method below and formally in Algorithm 2.

Training the teacher. The teacher model is trained on dataset S_ℓ, and then it is used to generate labels for the instances in S_u. The labels can be one-hot vectors or probability distributions on $[L]$, depending on whether we apply "hard" or "soft" distillation, respectively.

Training the student. We start by pretraining the student model on dataset S_ℓ. Then, the idea is to think of the teacher model as the adversary in the setting of Section 2.2, compute a weight $w(x)$ for each example x based on (4), and finally train the student on the union of labeled and teacher-labeled examples by minimizing the weighted empirical risk (examples from S_ℓ are assigned unit weight).

We point out two remarks. First, in order to apply (4) to compute the weight of an example x, we need to have estimates of $p(x)$ and $\mathrm{distortion}(x)$. To obtain these estimates we use the validation dataset S_v as we describe in the next paragraph. Second, observe that, according to (4), the weight of an example is a function of the predictor, namely the parameters of the model in the case of neural networks. This means that ideally we should be updating our weights assignment every time the parameters of our model get updated during training, which could be very computationally expensive. Fortunately though, we empirically observe that estimating the weights assignment *only once* during the whole training process (right after training the student model on S_ℓ) suffices for our purposes, and so our method adds minimal overhead to the standard training process.

Estimating the weights. To estimate the weights assignment with respect to a certain student model we use the validation dataset as described in Algorithm 1. The idea is to use the nearest neighbors method on S_v to learn a map that takes as input the teacher's and student's "confidence" for the label of a certain example x, and outputs estimates for $p(x)$ and $\mathrm{distortion}(x)$ so we can apply (4). Note

Algorithm 1 Procedure for estimating the weights

1: **procedure** ESTIMATEWEIGHTS($\text{Teacher}, \text{Student}, V, D$)
2: ▷ V is the validation dataset and D is the teacher-labeled dataset
3: $U \leftarrow \emptyset$, $k \leftarrow \lceil \frac{1}{2}\sqrt{|V|} \rceil$
4: **for** every $(x, y) \in V$ **do**
5: $X \leftarrow (\text{MarginScore}(\text{Teacher}(x)), \text{MarginScore}(\text{Student}(x)))$
6: **if** $\arg\max(\text{Teacher}(x)) = \arg\max(y)$ **then**:
7: $(p, \text{distortion}) \leftarrow (0, 1)$
8: **else**:
9: $(p, \text{distortion}) \leftarrow \left(1, \frac{\ell(\text{Teacher}(x), \text{Student}(x))}{\ell(y, \text{Student}(x))}\right)$
10: $Y \leftarrow (p, \text{distortion})$
11: $U \leftarrow U \cup \{(X, Y)\}$
12: $\text{Weights} = \varnothing$ ▷ Initialize and empty list for the weights
13: **for** every $(x, y) \in D$ **do**
14: $\text{Query} \leftarrow (\text{MarginScore}(\text{Teacher}(x)), \text{MarginScore}(\text{Student}(x)))$
15: $(\hat{p}, \hat{d}) \leftarrow k\text{-NN}(U, \text{Query})$ ▷ Predict $p(x)$ and $\text{distortion}(x)$ from the k nearest neighbors of Query in U
16: $w(x) \leftarrow \min\left\{1, \frac{1}{1+\hat{p}(\hat{d}-1)}\right\}$
17: Append $w(x)$ to Weights
18: **return** Weights

that we measure the confidence of a model on an example via its *margin-score*, i.e., the difference between the largest two predicted classes, just as is done in the so-called "margin" uncertainty sampling variant [36].

We point out two remarks. First, the number k of neighbors we use for our weights-estimation is always $k = \frac{\sqrt{|S_v|}}{2}$. This is because choosing $k = \Theta(q^{2/2+D})$, where q is the size of the validation dataset and D is the dimension of the underlying metric space ($D = 2$ in our case), is asymptotically optimal, and $1/2$ is a popular choice for the hidden constant used in practice, see e.g. [6, 12]. Second, notice that (4) implies that the weight of an example could be larger than 1 if (and only if) the corresponding distortion value (4) at that example is less than 1. This could happen for example if both the student and teacher have the same (or very similar) inaccurate prediction for a certain example. In such a case, the value of the weight in (4) informs us that the loss at this example should be larger than the (low) value the unweighted loss function suggests. However, since we do not have the ground-truth label for a point during training — but only the inaccurate prediction of the teacher — having a weight larger than 1 in this case would most likely guide our student model to fit an inaccurate label. To avoid this phenomenon, we always project our weights onto the $[0, 1]$ interval in Line 16 of Algorithm 1. In the Supplementary Material we discuss an additional reason why it is beneficial to project the weights of examples of low distortion onto $[0, 1]$ based on a MSE analysis.

3 Experimental results

In this section we present experimental results comparing our method against conventional hard [19] and soft [14] distillation with unlabeled examples on different popular benchmarks.

3.1 Experimental setup

Our experiments are of the following form. The academic dataset we use each time is first split into two parts A and B. Part A, which is typically smaller, is used as the labeled dataset S_ℓ where the teacher model is trained on (recall the setting we described in Section 2.3). Part B is randomly split again into two parts which represent the unlabeled dataset S_u and validation dataset S_v, respectively. Then, (i) the teacher and student models are trained once on the labeled dataset S_ℓ; (ii) the teacher model is used to generate labels for the unlabeled dataset S_u (either "hard" or "soft" labels, depending on the experiment); (iii) we train the student model on the union of S_ℓ and S_u using our method and conventional distillation with unlabeled examples. We repeat step (iii) a number of times: in each

Algorithm 2 Weighted distillation with unlabeled examples

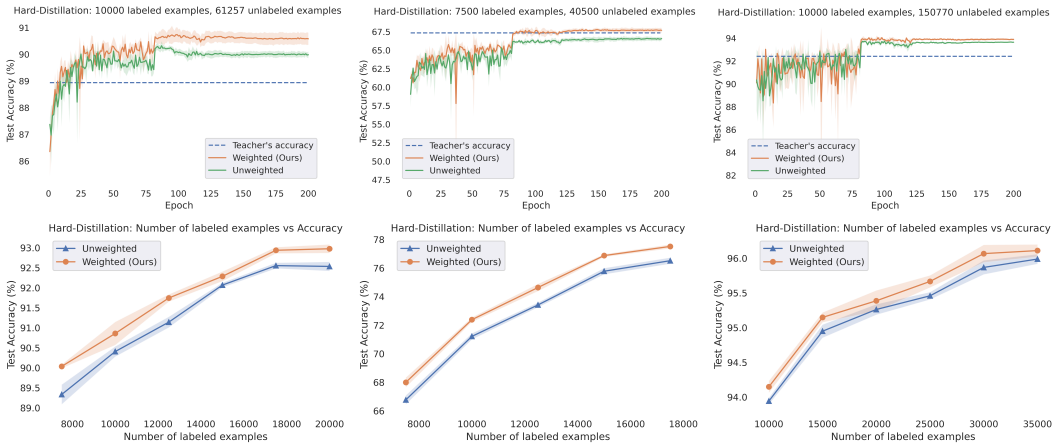
Require: model Teacher, model Student, labeled dataset $S_\ell = \{(x_i, y_i)\}_{i=1}^{m}$, unlabeled dataset $S_u = \{x_i\}_{i=m+1}^{m+n}$, validation dataset $S_v = \{(x_i, y_i)\}_{i=m+n+1}^{m+n+q}$, number of weights-estimating iterations r

Train Teacher and Student on S_ℓ
Use Teacher to generate labels for S_u to obtain set $S_u^\ell = \{(x, \text{Teacher}(x)) \mid x \in S_u\}$
for $i = m+1$ to $n+m$ **do**
    $y_i \leftarrow \text{Teacher}(x_i)$
$S \leftarrow \{(x_i, y_i)\}_{i=1}^{m+n}$
for $i = 1$ to m **do**
    $w(x_i) \leftarrow 1$
$\{w(x_{m+1}), \ldots, w(x_{m+n})\} \leftarrow$ ESTIMATEWEIGHTS(Teacher, Student, S_v, S_u^ℓ)
Train Student on S using the weighted empirical risk:

$$\frac{1}{m+n} \sum_{i=1}^{m+n} w(x_i) \ell(y_i, \text{Student}(x_i))$$

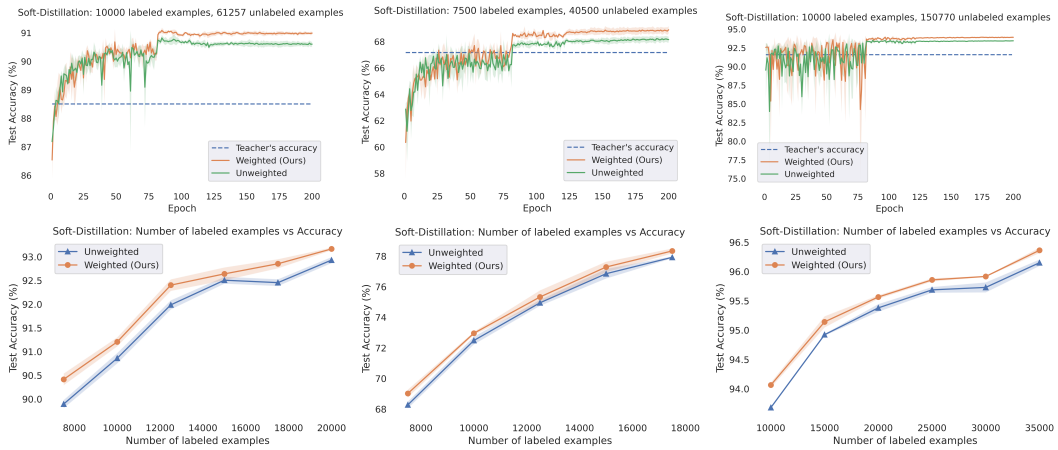
trial we partition part B randomly and independently, and then the student model is trained using the (student-)weights reached after completing the training on dataset S_ℓ in step (i) as initialization, both for our method and the conventional approaches [14, 19] .

We evaluate our method to benchmark vision datasets. We compare our method to conventional distillation with unlabeled examples both in terms of the best test accuracy achieved by the student, and in terms of convergence speed (see Figures 1, 2). We also evaluate the comparative advantage of our method as a function of the number of labeled examples available (size of dataset S_ℓ). We perform experiments both in the "hard" and "soft" distillation setting. In the latter case, we always choose the temperature in the softmax of the models to be 1 for simplicity and consistency. Implementation details for our experiments and additional results can be found in the Supplementary Material. We implemented all algorithms in Python making use of the TensorFlow deep learning library [1]. We use 64 Cloud TPU v4s each with two cores.



SVHN CIFAR-10 CelebA

Figure 1: The student's test accuracy over the training trajectory (first row) and student's best test-accuracy achieved over all (second row) when using **hard-distillation**. The teacher model is a MobileNet with depth multiplier 2. In the cases of SVHN and CIFAR-10 the student model is a MobileNet of depth multiplier 1, and in the case of CelebA is a ResNet-11. Our approach leads to consistently better models in terms of test-accuracy and convergence speed. Shaded regions correspond to values within one standard deviation of the mean.



SVHN CIFAR-10 CelebA

Figure 2: The student's test accuracy over the training trajectory (first row) and student's best test-accuracy achieved over all (second row) when using **soft-distillation**. Our approach leads to consistently better models in terms of test-accuracy and convergence speed. The model-architectures used are the same as in Figure 1.

3.2 SVHN experiments.

SVHN [29] is an image classification dataset where the task is to classify street view numbers (10 classes). The train set of SVHN contains 73257 labeled images and its test set contains 26032 images. We use a MobileNet [15] with depth multiplier 2 as the teacher, and a MobileNet with depth multiplier 1 as the student[1]. The tables in Figure 3 contain the results of our experiments (averages over 3 trials). In each experiment we use the first $N \in \{7500, 10000, 12500, 15000, 17500, 20000\}$ examples as the labeled dataset S_ℓ, and then the rest $73257 - N$ images are randomly split to a labeled validation dataset S_v of size 2000, and an unlabeled dataset S_u of size $71257 - N$.

Labeled Examples	7500	10000	12500	15000	17500
Teacher (soft)	87.31%	88.5%	88.45%	91.38%	91.32%
Weighted (Ours)	$\mathbf{90.41 \pm 0.12\%}$	$\mathbf{91.21 \pm 0.09\%}$	$\mathbf{92.4 \pm 0.12\%}$	$\mathbf{92.64 \pm 0.14\%}$	$\mathbf{92.85 \pm 0.1\%}$
Unweighted [14]	$89.89 \pm 0.09\%$	$90.86 \pm 0.11\%$	$91.99 \pm 0.1\%$	$92.51 \pm 0.07\%$	$92.45 \pm 0.07\%$
Teacher (hard)	87.53%	88.94%	90.42%	91.17%	91.85%
Weighted (Ours)	$\mathbf{90.03 \pm 0.05\%}$	$\mathbf{90.86 \pm 0.29\%}$	$\mathbf{91.74 \pm 0.09\%}$	$\mathbf{92.28 \pm 0.12\%}$	$\mathbf{92.94 \pm 0.08\%}$
Unweighted [19]	$89.33 \pm 0.25\%$	$90.41 \pm 0.13\%$	$91.14 \pm 0.13\%$	$92.07 \pm 0.06\%$	$92.55 \pm 0.07\%$

Figure 3: Experiments on SVHN. See Section 3.2 for details.

3.3 CelebA experiments.

CelebA [8] is a large-scale face attributes dataset with more than 200000 celebrity images, each with forty attribute annotations. Here we consider the binary male/female classification task. The train set of CelebA contains 162770 images and its test set contains 19962 images. We use a MobileNet with depth multiplier 2 as the teacher, and a ResNet-11 [13] as the student. The tables in Figure 4 contain the results of our experiments (averages over 3 trials). In each experiment we use the first $N \in \{10000, 15000, 20000, 25000, 30000, 35000\}$ examples as the labeled dataset S_ℓ, and then the rest $162770 - N$ images are randomly split to a labeled validation dataset S_v of size 2000, and an unlabeled dataset S_u of size $160770 - N$.

3.4 CIFAR-10 and CIFAR-100 experiments.

CIFAR-10 and CIFAR-100 [17] are image classification datasets with 10 and 100 classes respectively. Each of them consists of 60000 labeled images, which we split to a training set of 50000 images, and

[1]Note that we see the student outperform the teacher here and in other experiments, as can often happen with distillation with unlabeled examples, particularly when the teacher is trained on limited data.

Labeled Examples	10000	15000	20000	25000	30000
Teacher (soft)	91.59%	93.76%	94.41%	94.86%	94.92%
Weighted (Ours)	**94.06 ± 0.04%**	**95.14 ± 0.1%**	**95.56 ± 0.03%**	**95.86 ± 0.03%**	**95.92 ± 0.01%**
Unweighted [14]	93.68 ± 0.01%	94.92 ± 0.02%	95.38 ± 0.07%	95.69 ± 0.05%	95.73 ± 0.09%
Teacher (hard)	92.93%	93.3%	55.7%	94.33%	94.69%
Weighted (Ours)	**94.15 ± 0.08%**	**95.15 ± 0.07%**	**95.39 ± 0.15%**	**95.67 ± 0.09%**	**96.07 ± 0.13%**
Unweighted [19]	93.94 ± 0.05%	94.95 ± 0.09%	95.26 ± 0.08%	95.46 ± 0.05%	95.87 ± 0.1%

Figure 4: Experiments on CelebA. See Section 3.3 for details.

a test set of 10000 images. For CIFAR-10, we use a Mobilenet with depth multiplier 2 as the teacher, and a Mobilenet with depth multiplier 1 as the student. For CIFAR-100, we use a ResNet-110 as a teacher, and a ResNet-56 as the student. We use a validation set of 2000 examples. The results of our experiments (averages over 3 trials) can be found in the tables of Figures 5, 6.

Labeled Examples	7500	10000	12500	15000	17500
Teacher (soft)	67.17%	71.39%	74.69%	77%	78.46%
Weighted (Ours)	**69.01 ± 0.25%**	**72.96 ± 0.1%**	**75.33 ± 0.43%**	**77.69 ± 0.35%**	**78.34 ± 0.16%**
Unweighted [14]	68.27 ± 0.22%	72.48 ± 0.26%	74.95 ± 0.2%	76.85 ± 0.32%	77.92 ± 0.06%
Teacher (hard)	67.34%	72.36%	74.9%	77.18%	78.5%
Weighted (Ours)	**67.99 ± 0.34%**	**72.38 ± 0.16%**	**74.63 ± 0.25%**	**76.87 ± 0.1%**	**77.51 ± 0.14%**
Unweighted [19]	66.76 ± 0.31%	71.21 ± 0.21%	73.41 ± 0.16%	75.77 ± 0.22%	76.51 ± 0.21%

Figure 5: Experiments on CIFAR-10. See Section 3.4 for details.

Labeled Examples	7500	10000	12500	15000	17500
Teacher (soft)	44.1%	51.31%	55.54%	59.05%	62.17%
Weighted (Ours)	**46.81 ± 0.34%**	**53.31 ± 0.66%**	**57.5 ± 0.3%**	**60.94 ± 0.32%**	**62.86 ± 0.41%**
Unweighted [14]	46.29 ± 0.04%	52.83 ± 0.53%	56.89 ± 0.65%	60.73 ± 0.03%	62.79 ± 0.09%
Teacher (hard)	45.1%	50.56%	55.7%	59.69%	61.26%
Weighted (Ours)	**44.88 ± 0.38%**	**50.97 ± 0.13%**	**55.43 ± 0.26%**	**59.06 ± 0.15%**	**61.24 ± 0.34%**
Unweighted [19]	44.66 ± 0.16%	50.36 ± 0.17%	54.97 ± 0.28%	58.57 ± 0.17%	61.06 ± 0.11%

Figure 6: Experiments on CIFAR-100. See Section 3.4 for details.

3.5 ImageNet experiments.

ImageNet [37] is a large-scale image classification dataset with 1000 classes consisting of approximately $I \approx 1.3$M images. For our experiments, we use a ResNet-50 as the teacher, and a ResNet-18 as the student. In each experiment we use the first $N \in \{64058, 128116\}$ labeled examples (5% and 10% of I, respectively) as the labeled dataset S_ℓ, and the rest $I - N$ examples are randomly split to a labeled validation dataset S_v of size 10000, and an unlabeled dataset S_u of size $I - N - 10000$. The results of our experiments (averages over 10 trials) can be found in Figure 7.

Labeled Examples	5% of ImageNet	10% of ImageNet
Teacher (soft)	36.74%	51.88%
Weighted (Ours)	**38.60 ± 0.07%**	**53.59 ± 0.09%**
Unweighted [14]	38.44 ± 0.06	53.43 ± 0.08%

Labeled Examples	5% of ImageNet	10% of ImageNet
Teacher (hard)	36.74%	51.88%
Weighted (Ours)	**38.56 ± 0.07%**	**53.34 ± 0.11%**
Unweighted [19]	38.42 ± 0.06	53.18 ± 0.07%

Figure 7: Experiments on ImageNet. See Section 3.5 for details.

4 Theoretical motivation

In this section we show that our debiasing reweighting method described in Section 2.2 comes with provable **statistical** and **optimization** guarantees. We first show that the method is statistically consistent in the sense that, with a sufficiently large dataset, the reweighted risk converges to the true or "clean" risk. We show (see Supplementary Material) the following convergence guarantee.

Theorem 4.1 (Uniform Convergence of Reweighted Risk)**.** *Assume that S is a dataset of i.i.d. "noisy" samples from $\mathbb{D}$. Under standard capacity assumptions for the class of models $\mathcal{F}$ and regularity assumptions for the loss $\ell(\cdot)$, for every $f \in \mathcal{F}$ it holds that*

$$\lim_{|S|\to\infty} R_S^w(f) = R(f) \quad \textit{and} \quad \lim_{|S|\to\infty} R_S(f) = R(f) + \mathrm{Bias}(f)$$

To prove our optimization guarantees we analyze the reweighted objective in the fundamental case where the model $f(x;\Theta)$ is linear, i.e., $f(x;\Theta) = \Theta x \in \mathbb{R}^L$, and the loss $\ell(y,z)$ is convex in z for every y. In this case, the composition of the loss and the model $f(x;\Theta)$ is convex as a function of the parameter $\Theta \in \mathbb{R}^{L\times d}$. Recall that we denote by $f_{\mathrm{true}}(x) : \mathbb{R}^d \mapsto \mathbb{R}^L$ the ground truth classifier and by $\mathbb{P}$ the "clean" distribution, i.e., a sample from $\mathbb{P}$ has the form $(x, f_{\mathrm{true}}(x))$ where x is drawn from a distribution $\mathbb{X}$ supported on (a subset of) $\mathbb{R}^d$. Finally, we denote by $\mathbb{D}$ the "noisy" labeled distribution on $\mathbb{R}^d \times \mathbb{R}^L$ and assume that the x-marginal of $\mathbb{D}$ is also $\mathbb{X}$.

We next give a general definition of debiasing weight functions, i.e., weighting mechanisms that make the corresponding objective function an unbiased estimator of the clean objective $R(\Theta)$ for every parameter vector $\Theta \in \mathbb{R}^d$. Recall that the weight function defined in Section 2.2 is debiasing.

Definition 4.2 (Debiasing Weights)**.** *We say that a weight function $w(x, y_{\mathrm{adv}}; \Theta)$ is a debiasing weight function if it holds that*

$$R^w(\Theta) \triangleq \mathbb{E}_{(x,y_{\mathrm{adv}})\sim\mathbb{D}}[w(x, y_{\mathrm{adv}}; \Theta)\ell(y_{\mathrm{adv}}, f(x;\Theta))] = R(\Theta)\,.$$

Since the loss is convex in Θ, one could try to optimize the naive objective that does not reweight and simply minimizes $\ell(\cdot)$ over the noisy examples, $R^{\mathrm{naive}}(\Theta) \triangleq \mathbb{E}_{(x,y)\sim\mathbb{D}}[\ell(y_{\mathrm{adv}}, \Theta x)]$. We show (unsurprisingly) that doing this is a bad idea: there are instances where optimizing the naive objective produces classifiers with bad generalization error over clean examples. For the formal statement and proof we refer the reader to the Supplementary Material.

SGD on the naive objective $R^{\mathrm{naive}}(\cdot)$ learns parameters with arbitrarily bad generalization error over the "clean" data.

Our main theoretical insight is that optimizing linear models with the reweighted loss leads to parameters with almost optimal generalization.

Given a debiasing weight function $w(\cdot)$, SGD on the reweighted objective $R^w(\cdot)$ learns a parameter with almost optimal generalization error over the "clean" data.

The main issue with optimizing the reweighted objective is that, in general, we have no guarantees that the weight function preserves its convexity (recall that it depends on the parameter Θ). However, we know that its population version corresponds to the clean objective $R(\cdot)$ which is a convex objective. We show that we can use the convexity of the underlying clean objective to show results for stochastic gradient descent, by proving the following key structural property.

Proposition 4.3 (Stationary Points of the Reweighted Objective Suffice (Informal))**.** *Let S be a dataset of $n = \mathrm{poly}(dL/\epsilon)$ i.i.d. samples from the noisy distribution $\mathbb{D}$. Let $\widehat{\Theta}$ be any stationary point of the weighted objective $R^w_S(\Theta)$ constrained on the unit ball (with respect to the Frobenious norm $\|\cdot\|_F$). Then, with probability at least 99%, it holds that*

$$R(\widehat{\Theta}) \leq \min_{\|\Theta\|_F \leq 1} R(\Theta) + \epsilon\,.$$

5 Conclusion

We propose a principled importance reweighting method for distillation with unlabeled examples. Our method applies both to "soft" and "hard" distillation, it is hyper-parameter free, and adds minimal implementation overhead. Our approach comes with theoretical guarantees. We evaluated our method on standard benchmarks and we showed that it provides significant improvements, especially (but not only) in the important case where the number of labeled examples is small compared to the number of unlabeled examples. We note that investigating improved data-driven ways of estimating the weights (4) could be of interest, as the simplicity of the nearest neighbors method we use for estimation is effectively the main limitation of our approach. We have not tried more sophisticated methods for estimating the weights, and so it is plausible that there is room for improvement. We leave this question open for future work. To conclude, we envisage that our approach can be used in high-impact, large-scale applications, due to its simplicity and effectiveness.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] Eric Arazo, Diego Ortego, Paul Albert, Noel O'Connor, and Kevin McGuinness. Unsupervised label noise modeling and loss correction. In *International conference on machine learning*, pages 312–321. PMLR, 2019.

[3] Noga Bar, Tomer Koren, and Raja Giryes. Multiplicative reweighting for robust neural network optimization. *arXiv preprint arXiv:2102.12192*, 2021.

[4] Cristian Buciluă, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006.

[5] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255, 2020.

[6] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[7] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2013.

[8] Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *International Conference on Machine Learning*, pages 1607–1616. PMLR, 2018.

[9] Dragan Gamberger, Nada Lavrac, and Ciril Groselj. Experiments with noise filtering in a medical domain. In *ICML*, volume 99, pages 143–151, 1999.

[10] Jacob Goldberger and Ehud Ben-Reuven. Training deep neural-networks using a noise adaptation layer. 2016.

[11] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *Advances in neural information processing systems*, 31, 2018.

[12] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

[15] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[16] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*, pages 2304–2313. PMLR, 2018.

[17] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[18] Abhishek Kumar and Ehsan Amid. Constrained instance and class reweighting for robust learning under label noise. *arXiv preprint arXiv:2111.05428*, 2021.

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

[20] Lu Liu and Robby T Tan. Certainty driven consistency loss on multi-teacher networks for semi-supervised learning. *Pattern Recognition*, 120:108140, 2021.

[21] Tongliang Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. *IEEE Transactions on pattern analysis and machine intelligence*, 38(3):447–461, 2015.

[22] Yang Liu and Hongyi Guo. Peer loss functions: Learning from noisy labels without knowing noise rates. In *International Conference on Machine Learning*, pages 6226–6236. PMLR, 2020.

[23] Michal Lukasik, Srinadh Bhojanapalli, Aditya Menon, and Sanjiv Kumar. Does label smoothing mitigate label noise? In *International Conference on Machine Learning*, pages 6448–6458. PMLR, 2020.

[24] Xingjun Ma, Yisen Wang, Michael E Houle, Shuo Zhou, Sarah Erfani, Shutao Xia, Sudanthi Wijewickrema, and James Bailey. Dimensionality-driven learning with noisy labels. In *International Conference on Machine Learning*, pages 3355–3364. PMLR, 2018.

[25] Negin Majidi, Ehsan Amid, Hossein Talebi, and Manfred K Warmuth. Exponentiated gradient reweighting for robust training under label noise and beyond. *arXiv preprint arXiv:2104.01493*, 2021.

[26] Rafael Müller, Simon Kornblith, and Geoffrey Hinton. Subclass distillation. *arXiv preprint arXiv:2002.03936*, 2020.

[27] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? *Advances in neural information processing systems*, 32, 2019.

[28] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. *Advances in neural information processing systems*, 26, 2013.

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

[30] Hieu Pham, Zihang Dai, Qizhe Xie, and Quoc V Le. Meta pseudo labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11557–11568, 2021.

[31] Geoff Pleiss, Tianyi Zhang, Ethan Elenberg, and Kilian Q Weinberger. Identifying mislabeled data using the area under the margin ranking. *Advances in Neural Information Processing Systems*, 33:17044–17056, 2020.

[32] Ilija Radosavovic, Piotr Dollár, Ross Girshick, Georgia Gkioxari, and Kaiming He. Data distillation: Towards omni-supervised learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4119–4128, 2018.

[33] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014.

[34] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International conference on machine learning*, pages 4334–4343. PMLR, 2018.

[35] Ellen Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings of the national conference on artificial intelligence*, pages 1044–1049, 1996.

[36] Dan Roth and Kevin Small. Margin-based active learning for structured output spaces. In *European Conference on Machine Learning*, pages 413–424. Springer, 2006.

[37] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[38] Henry Scudder. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371, 1965.

[39] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[40] Hongxin Wei, Lei Feng, Xiangyu Chen, and Bo An. Combating noisy labels by agreement: A joint training method with co-regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13726–13735, 2020.

[41] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2691–2699, 2015.

[42] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10687–10698, 2020.

[43] Yilun Xu, Peng Cao, Yuqing Kong, and Yizhou Wang. L_dmi: An information-theoretic noise-robust loss function. *arXiv preprint arXiv:1909.03388*, 2019.

[44] I Zeki Yalniz, Hervé Jégou, Kan Chen, Manohar Paluri, and Dhruv Mahajan. Billion-scale semi-supervised learning for image classification. *arXiv preprint arXiv:1905.00546*, 2019.

[45] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pages 189–196, 1995.

[46] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

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

[48] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, 31, 2018.

[49] Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin Dogus Cubuk, and Quoc Le. Rethinking pre-training and self-training. *Advances in neural information processing systems*, 33:3833–3845, 2020.

[50] Yang Zou, Zhiding Yu, Xiaofeng Liu, BVK Kumar, and Jinsong Wang. Confidence regularized self-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5982–5991, 2019.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

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 Section 3 for an empirical evaluation of our method. See Section 4 for a theoretical justification for the performance of our approach.
   (b) Did you describe the limitations of your work? [Yes] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (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? [Yes] See theoretical results in Section 4 and the Supplementary Material.
   (b) Did you include complete proofs of all theoretical results? [Yes] We provide complete proofs of all theoretical results in the Supplementary Material.
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)? [No]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 3 and the Supplementary Material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] All plots in Section 3 contain shaded regions that correspond to the values within one standard deviation of the mean.
   (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 Sec. 3
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] See Section 3 and the Supplementary Material.
   (b) Did you mention the license of the assets? [Yes] See the Supplementary Material.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (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]