Improving Task-Specific Generalization in Few-Shot Learning via Adaptive Vicinal Risk Minimization

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recent years have witnessed the rapid development of meta-learning in improving the meta generalization over tasks in few-shot learning. However, the task-specific level generalization is overlooked in most algorithms. For a novel few-shot learning task where the empirical distribution likely deviates from the true distribution, the model obtained via minimizing the empirical loss can hardly generalize to unseen data. A viable solution to improving the generalization comes as a more accurate approximation of the true distribution; that is, admitting a Gaussian-like vicinal distribution for each of the limited training samples. Thereupon we derive the resulting vicinal loss function over vicinities of all training samples and minimize it instead of the conventional empirical loss over training samples only, favorably free from the exhaustive sampling of all vicinal samples. It remains challenging to obtain the statistical parameters of the vicinal distribution for each sample. To tackle this challenge, we further propose to estimate the statistical parameters as the weighted mean and variance of a set of unlabeled data it passed by a random walk starting from training samples. To verify the performance of the proposed method, we conduct experiments on four standard few-shot learning benchmarks and consolidate the superiority of the proposed method over state-of-the-art few-shot learning baselines.

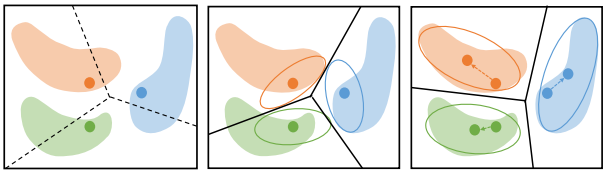
1 Introduction

In recent years, deep learning algorithms have made tremendous progress and have been widely applied in vision [14], language [31] and speech [8] areas. Despite the prevalence, deep learning algorithms rely heavily on a sufficient amount of labeled data and often fail when only a limited number of labeled samples are available. Considering the high cost of extensive data collection and annotation, it has attracted increasing attention to develop algorithms learning from a limited number of training samples, which has been widely known as few-shot learning [33, 4].

The recent focus of few-shot learning is built on the meta-learning framework [6, 33, 27]. The goal of meta-learning is to train a model that can be quickly adapted to a new task with a small number of labeled samples by leveraging the knowledge from previous related tasks. Typically, a meta-learning model consists of the meta-training stage where a well-generalized feature extractor is learned from meta-training tasks and the meta-testing stage during which a classifier is built on top of the feature extractor for novel tasks. While the majority of meta-learning works revolved around developing better meta-training strategies to reduce the meta generalization error over all tasks, how to fully leverage the meta-trained feature extractor to build a robust classifier and improve a task-specific generalization in the meta-testing stage remains less explored.

The challenge to building such a classifier lies in the significant deviation of the empirical distribution (i.e. the distribution formed by the training data) from the population distribution (i.e. the true

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



(a) (b) (c)

Figure 1: (a) The classifier overfits to few-shot examples that inadequately represent the true distribution. (b) The vicinal distribution without adjusting the means slightly improves the classifier. (c) The vicinal distribution with updated means contributes a well-generalizable classifier.

distribution). As illustrated in Figure 1, a model well fitting the training data (support set) may generalize poorly on the testing data (query set) from the true distribution due to the bias in the empirical distribution. One viable solution is to approximate the true distribution more accurately – by viewing the empirical distribution as the assembling of delta functions located on each training sample, Chapelle et al. [3] proposed the Vicinal Risk Minimization loss function where the delta functions are replaced by some density estimate in the vicinity of each training data. In the era of deep learning, the VRM learning principle is implemented as Mixup [41, 32], which is simple yet effective given a sufficiently large number of training data. However, as we have verified in Table 3, the performance improvement by applying Mixup on the support set is limited in few-shot learning where the vicinal distributions assembled by Mixup is still significantly biased. To improve the generalization of a few-shot classifier during meta-testing, we are motivated to revisit the VRM principle to 1) generate adaptive vicinal distribution for each sample in support set and 2) minimize the expected vicinal loss over the vicinal distributions. However, both are challenging.

Without loss of generality, we assume that the vicinity of each training point admits a Gaussian distribution. In [3], the means of the vicinal distributions are located on the original training points. As illustrated in Figure 1(b), however, setting the mean in this way is not sufficient to approximate the true distribution, as an original training point likely deviates from the mean of the ground-truth distribution of its corresponding class. In fact, it is difficult to estimate the mean and variance based only on the labeled training samples. In this paper, we assume that unlabeled samples are also available, and leverage these unlabeled samples to estimate the mean and variance of the vicinal distribution for each labeled sample. Concretely, inspired by [42], we first construct a transition probability matrix based on pairwise distances between features of all samples, and secondly perform the lazy random walk among unlabeled samples starting from each labeled sample. After walking a certain number of steps, we obtain the visit probabilities of all the samples and infer the vicinal distribution whose mean and variance are the weighted mean and variance of all samples visited.

All that remains is the loss function over the vicinal distributions of all training samples. An intuitive way to do this is to exploit the Monte Carlo method, which repeats random sampling to explicitly generate data points from each vicinal distribution and minimizes the loss on all the sampled data points. Unfortunately, it requires to draw a large amount of samples to minimize the Monte Carlo approximation error, which significantly increases the computation cost. Instead, we follow [3] to derive a closed form of the expected vicinal loss over the vicinal distributions. In this paper, we focus on building an optimization-based classifier in optimization-based meta-learning algorithms [6, 16], which has been proved with much more adaptability and generalization capacity in novel meta-testing tasks than a nearest-neighbour classifier in metric-based meta-learning algorithms [33, 27, 28, 21]. Particularly, we consider both the cross-entropy loss and the SVM loss functions which are used as classifier in the state-of-the-art meta-learning algorithms [37, 16]. For the vicinal version of cross-entropy loss, it cannot be computed in closed form. We instead derive its quadratic approximation. For SVM, we derive its closed-form vicinal loss.

In summary, our contributions are threefold:

- We revisit vicinal risk minimization (VRM) in deep few-shot learning and derive the resulting expected vicinal loss functions for both the cross-entropy loss and SVM.
- We propose to harness unlabeled data and a lazy random walk algorithm to generate adaptive statistical parameters for the vicinal distribution of each training sample.

- We have validated the superiority of the proposed algorithm with extensive experiments on three benchmarks, i.e., miniImageNet, CUB, CIFAR-FS.

2 Related Works

2.1 Few-Shot Learning and Meta-Learning

Few-shot learning aims to recognize novel classes with insufficient labeled instances. The recent focus of few-shot learning algorithms has been on the meta-learning framework, which trains the model with multiple meta-episodes to enable quick learning and applies the model to improve the learning in novel tasks. The meta-learning methods fall into two categories: metric-based methods and optimization-based methods. Metric-based methods [13, 33, 27, 28, 20, 39] learn a metric to compare two samples' feature embeddings encoded by the feature extractor and classify a sample in the query set based on its distances to the samples in the support set. Optimization-based approaches [6, 23, 25, 29, 35] learn a good initialization such that it can be fast adapted to novel tasks based on a few instances.

To tackle the data deficiency issue in few-shot learning, researchers have explored the idea of exploiting unlabeled data and developed transductive learning or semi-supervised learning techniques for few-shot learning, either by leveraging nearest neighbor graph for propagating labels [36, 12, 18] or by leveraging predicted soft or hard labels on unlabeled samples to update the class prototypes or metric [24, 10, 15, 22, 39]. Our method is not built on metric-based method. We make use of the unlabeled data to derive the vicinal distribution for each support sample and optimize the consequential vicinal loss to learn a classifier. Our approach shares a similar spirit of [37]. It [37] proposes to calibrate the distribution of the support samples by transferring statistics from the base classes and train the classifier on a set of samples sampled from the calibrated distributions. Unlike [37], our method obtains more accurate vicinal distributions by making use of the unlabeled data from the same task. Besides, instead of sampling extensive data from the distributions, our model directly optimizes the expected vicinal loss over the distributions, resulting in more efficient training and higher performance.

2.2 Vicinal Risk Minimization

The vicinal Risk Minimization principle is introduced in [3, 30]. It proposes to assume each training sample admits a vicinal distribution and to optimize the expected loss over the vicinal distributions. In [3], only binary VRM classifier are presented. Our method considers multi-class classifier and cross-entropy loss, which is more challenging to derive the expected vicinal loss. Besides, Chapelle et al. [3] consider Gaussian vicinities whose means lie on the original training samples. Our approach also considers Gaussian vicinities but estimates the mean by the unlabeled data, which is generally not on the training samples.

In the era of deep learning, the most popular VRM technique in deep learning algorithms is Mixup [41, 32]. Mixup constructs the vicinal distributions as the convex combination of the training data, which can be understood as a data augmentation technique that encourages the model to behave linearly inter training examples. It has been demonstrated that Mixup is simple yet efficient and improves the generalization of deep model given sufficiently many training samples. Mixup is also introduced on the meta-learning algorithm in [38, 7], but is applied on the query set for meta model optimization. As shown by the experiment (Table 6) in [38], applying Mixup on the support set could even hurt the performance. Our approach instead leverages the unlabeled data for a more accurate vicinal distribution estimate for an extremely small support set.

3 Adaptive Vicinal Few-Shot Learning

In this section, we will introduce the proposed algorithm, **AD**aptive **V**icinal Few-Shot Learning (ADV). We will first introduce the VRM; then derive the expected vicinal loss for a linear classifier with cross-entropy loss and for SVM; and finally propose the algorithm based on lazy random walks to obtain the adaptive vicinal distribution for each labeled support sample.

3.1 Vicinal Risk Minimization

Given data x, y following the distribution $P(x, y)$, the loss function $\mathcal{L}$, we would like the optimize a model $f_\theta(x)$ by minimizing the expected risk

$$\mathcal{L}(\theta) = \int \ell(f_\theta(x), y)\, \mathrm{d}P(x, y).$$

Here, x is the input data, y is its label and θ is the model parameters. In practice, $P(x, y)$ is unknown and a set of training data $\{x_i, y_i\}_{i=1}^N$ is available, where N is the size of training data. To learn a model, one can minimize the empirical risk, which is equivalent to minimizing the expected loss with respect to an empirical density estimate $\mathrm{d}P_\delta(x, y) = \frac{1}{N}\sum_i^N \delta_{x_i}(x)\delta_{y_i}(y)$, where $\delta_a(\cdot)$ is a Dirac delta function centered at a.

Given a limited number of training sample, the empirical density estimate $\mathrm{d}P_\delta(x, y)$ could bias from the true density $\mathrm{d}P(x, y)$. For a more accurate density estimate, Chapelle et al. [3] proposed to replace the delta functions $\delta_{x_i}(x)$ by some estimate of density in the vicinity of x_i. Given a vicinal distribution of x_i as $P_{x_i}(x)$, the improved density estimate is $P_\nu(x, y) = \frac{1}{N}\sum_{i=1}^N P_{x_i}(x)\delta_{y_i}(y)$. And the corresponding vicinal risk is defined as

$$\mathcal{L}_\nu(\theta) = \int \ell(f_\theta(x), y)\, \mathrm{d}P_\mu(x, y) = \frac{1}{N}\sum_{i=1}^N \int \ell(f_\theta(x), y_i)\, \mathrm{d}P_{x_i}(x).$$

3.2 Vicinal Loss

In this paper, we assume the feature extractor is trained and fixed for novel tasks. Each tasks is C-way K-shot, consisting of $N = CK$ data support samples $\{(x_i, y_i)\}_{i=1}^N$. To perform VRM, we assume the feature z_i of each sample x_i in the support set admits a Gaussian vicinal distribution $\mathcal{N}(\mu_i, \Sigma_i)$ such that a sample $\tilde{z}$ is drawn from the vicinal distribution with probability $P_{z_i}(\tilde{z}) = \det(2\pi\Sigma_i)^{-\frac{1}{2}} \exp(-\frac{1}{2}(\tilde{z} - \mu_i)^\top \Sigma_i^{-1}(\tilde{z} - \mu_i))$, where μ_i and Σ_i is the mean and variance of the vicinal distribution of z_i.

3.2.1 Vicinal Cross-Entropy Loss

We first focus on linear classifier with Cross-Entropy loss. A linear classifier $f_\theta(z) = (W^\top z + b)$ maps a feature $z \in \mathbb{R}^d$ to its logits, where $W \in \mathbb{R}^{d\times C}$ and $b \in \mathbb{R}^C$ is the parameter of the classifier.

The cross-entropy loss is defined as $\ell(h, y) = -e_y^\top \log(\mathrm{softmax}(f_\theta(z)))$, where $y \in 1, 2, ..., C$ is the label; e_y is a one-hot vector with the y-th element being 1. The expected loss of cross entropy over Gaussian vicinal distribution of x is defined as

$$\ell_\nu(f_\theta(z), y) = \int -e_y^\top \log(\mathrm{softmax}(f_\theta(\tilde{z})))\, \mathrm{d}P_z(\tilde{z}) = \int \ell(f_\theta(\tilde{z}), y)\, \mathrm{d}P_z(\tilde{z}) = \mathbb{E}_{P_z(\tilde{z})}[\ell(f_\theta(\tilde{z}), y)]. \quad (1)$$

Due to the softmax function, the expected loss (1) cannot be computed in closed form. Instead, we utilize Taylor expansion to derive an approximation of (1). By taking second-order Taylor expansion of $\ell(f_\theta(\tilde{z}), y)$ around the mean of its vicinal distribution μ, we obtain

$$\ell(f_\theta(\tilde{z}), y) \approx \ell(f_\theta(\mu), y) + G(\tilde{z} - \mu) + \frac{1}{2}(\tilde{z} - \mu)^\top H(\tilde{z} - \mu), \quad (2)$$

where G is the gradient and H is the Hessian w.r.t. input μ. Note that $H = W(\mathrm{diag}(h) - hh^\top)W^\top$ where $h = \mathrm{softmax}(f_\theta(\mu))$ and $\mathrm{diag}(h)$ is a diagonal matrix whose diagonal elements are equal to h [26]. By taking expectation over $\tilde{z}$, we have

$$\mathbb{E}_{P_z(\tilde{z})}[G(\tilde{z} - \mu)] = 0 \quad (3)$$

$$\mathbb{E}_{P_z(\tilde{z})}[(\tilde{z} - \mu)^\top H(\tilde{z} - \mu)] = \sum_{k=1}^C (h_k W_k^\top \Sigma W_k) - \sum_{j=1}^C \sum_{k=1}^C h_j h_k W_j^\top \Sigma W_k := R_\theta(\Sigma), \quad (4)$$

where Σ is the variance of $\tilde{z}$, h_k is the k-th elements of h, and W_k is the k-th column of W. Plugging (2), (3) and (4) into (1), we obtain the approximated expected vicinal cross-entropy loss for the linear classifier as

$$\mathcal{L}_\mu(\theta) = \frac{1}{N}\sum_{i=1}^N \left[\ell(f_\theta(\mu_i), y_i) + \frac{1}{2}R_\theta(\Sigma_i)\right]. \quad (5)$$

Algorithm 1 Adaptive Vicinal Few-Shot Learning (ADV)

Input: Support data $\{z_i, y_i\}_{i=1}^N$, unlabeled data $\{u_j\}_{j=1}^{N_u}$, lazy stay parameter β, number of step in lazy random walk T
Compute the distance matrix D^{SU} and D^{UU} by (10) and (11)
for all z_i **do**
    Compute the transition probabilities matrix P_i by (12) and (13)
    Compute the T-step visit probability v_i by (14)
    Compute μ_i and Σ_i as the weighted mean of variance of z_i and $\{u_j\}_{j=1}^{N_u}$ with the weights being v_i
end for
Compute the objective as the vicinal cross-entropy loss in (5) or as the Vicinal SVM in (9)
Optimize the objective until convergence

3.2.2 Vicinal SVM

SVM is used as classifier for meta learning algorithms in [16, 37]. We follow [16] to use the Crammer and Singer formulation of multi-class SVM [5]. In [16, 5], the dual formulation of the objective for a linear multi-class SVM is obtained as:

$$\max_{\alpha^k} \left[-\frac{1}{2} \sum_{k=1}^{C} \sum_{i,j=1}^{N} (\alpha_i^k \alpha_j^k) z_i^\top z_j + \sum_{i=1}^{n} \alpha_i^{y_i} \right] \tag{6}$$
$$s.t.\ \ \alpha_i^{y_i} \leq \lambda;\ \ \alpha_i^k \leq 0\ \ \forall k \neq y_i;\ \ \sum_k \alpha_i^k = 0\ \ \forall i,$$

where α_i^k is the dual variable and λ is a regularization parameter. The linear classifier for the k-th class is $f_\theta^k(z) = w_k^\top z$, where $w_k = \sum_i \alpha_i^k z_i$. Note that here z is an unseen data and does not assume to admit a vicinal distribution. If each z_i admits a vicinal distribution $\mathcal{N}(\mu_i, \Sigma_i)$, the vicinal expectation for the term $z_i^\top z_j$ in (6) becomes

$$\mathbb{E}_{\tilde{z}_i \sim \mathcal{N}(\mu_i, \Sigma_i)} \mathbb{E}_{\tilde{z}_j \sim \mathcal{N}(\mu_j, \Sigma_j)} [\tilde{z}_i^\top \tilde{z}_j] = \mu_i^\top \mu_j,$$

and the classifier becomes $\mathbb{E}_{\tilde{z}_i \sim \mathcal{N}(\mu_i, \Sigma_i)} [f_\theta^k(\tilde{z}_i)] = (\sum_i \alpha_i^k \mu_i)^\top z$. If $\mu_i = z_i$, the objective of the vicinal SVM coincides with the original one. Even when $\mu_i \neq z_i$, it still does not take the variance of the vicinal distribution into consideration and may result in suboptimal performance. To exploit the variance information to improve the SVM classifier, we consider nonlinear classifier, i.e. the kernel-based SVM.

Multi-class kernel-based SVM uses a kernel $\mathcal{K}(\cdot, \cdot)$ and admits a similar dual formulation to (6). The difference lays in that it replaces the inner product of z_i and z_j in (6) by $\mathcal{K}(z_i, z_j)$ and replace the inner product of z_i and z in the classifier by $\mathcal{K}(z_i, z)$. For the vicinal multi-class kernel-based SVM, these two terms are further replaced by a two-vicinal kernel $\mathcal{M}(z_i, z_j)$ and a one-vicinal kernel $\mathcal{Q}(z, z_i)$. In this paper, we consider the widely-used RBF kernel $K(z, z') = \exp(-\frac{||z-z'||^2}{2\sigma^2})$ and construct the vicinal kernels for it as:

$$\mathcal{Q}(z_i, z) = \frac{|A(\sigma^2)|^{\frac{1}{2}}}{|A(\sigma^2) + \Sigma_i|^{-\frac{1}{2}}} \exp\left(-\frac{1}{2} \|z - \mu_i\|_{M_i}^2 \right), \tag{7}$$

$$\mathcal{M}(z_i, z_j) = \frac{|A(\sigma^2)|^{\frac{1}{2}}}{|A(\sigma^2) + \Sigma_i + \Sigma_j|^{-\frac{1}{2}}} \exp\left(-\frac{1}{2} \|\mu_i - \mu_j\|_{M_{ij}}^2 \right). \tag{8}$$

Here, $A(\sigma^2) \in \mathbb{R}^{d \times d}$ is a diagonal matrix with diagonal elements being σ^2; $M_i = (A(\sigma^2) + \Sigma_i)^{-1}$; $M_{ij} = (A(\sigma^2) + \Sigma_i + \Sigma_j)^{-1}$; the operator $\|z\|_M^2 = z^\top M z$. Note that for computation simplification, in vicinal SVM, we assume the variance Σ_i is diagonal. The derivation is available in Appendix.

Now we achieve the vicinal objective for multi-class kernel-based SVM as:

$$\max_{\alpha^k} \left[-\frac{1}{2} \sum_{k=1}^{C} \sum_{i,j=1}^{N} (\alpha_i^k \alpha_j^k) \mathcal{M}(z_i, z_j) + \sum_{i=1}^{n} \alpha_i^{y_i} \right] \tag{9}$$
$$s.t.\ \ \alpha_i^{y_i} \leq \lambda;\ \ \alpha_i^k \leq 0\ \ \forall k \neq y_i;\ \ \sum_k \alpha_i^k = 0\ \ \forall i,$$

with the classifier being $f_\theta^k(z) = \sum_i \alpha_i^k \mathcal{Q}(z_i, z)$. The objective in (9) is a quadratic program (QP) over dual variables $\{\alpha_i^k\}$ and can be solved by using a differentiable GPU-based QP solver [1].

3.3 Lazy Random Walk for Adaptive Vicinal Distribution

As illustrated in Figure 1, the performance of the model learned by VRM has a heavy reliance on the quality of the vicinal distribution to approximate the true distribution. However, it is almost impossible to estimate accurate proximate distributions from a limited number of support samples, as they may deviate from the true distribution. Therefore, we propose to exploit extra data, i.e. the unlabeled data from the same task, to reduce the estimation error of the statistical parameters for the vicinal distribution and, thus, improve the performance of the model.

We assume that a well-trained feature extractor on the meta-training tasks can preserve the similarity between the data in the feature space. Data from the same class in the novel task would have similar features with high probability. Then, we apply lazy random walks algorithm [42] to estimate the probability that the unlabeled data would be visited starting from a support sample, and compute the statistical parameters as the weighted mean and variance of the data visited by the lazy random walks.

To perform lazy random walk, we first compute the distance D^{SU} between the support sample z_i and the unlabeled data $\{u_j\}_{j=1}^{N_u}$ and compute the distance D_{UU} between the unlabeled data as:

$$D_{ij}^{SU} = \|z_i - u_j\|^2, \quad \text{and} \tag{10}$$

$$D_{ij}^{UU} = \begin{cases} \|u_i - u_j\|^2 & i \neq j \\ \infty & i = j \end{cases} \tag{11}$$

Secondly, we transform these distances into transition probabilities by softmaxing the distance matrix over columns with a temperature τ:

$$P_{ij}^{SU} = exp\left(-\frac{1}{\tau} D_{ij}^{SU}\right) / \sum_{j'} exp\left(-\frac{1}{\tau} D_{ij'}^{SU}\right), \quad P_{ij}^{UU} = exp\left(-\frac{1}{\tau} D_{ij}^{UU}\right) / \sum_{j'} exp\left(-\frac{1}{\tau} D_{ij'}^{UU}\right). \tag{12}$$

Note that we set $D_{jj}^{UU} = \infty$ such that $P_{jj}^{UU} = 0$.

Thirdly, we construct the transition probabilities matrix for z_i in lazy random walk by combining $P_{i:}^{SU}$ and P^{UU} and setting the lazy stay probability $\beta \in (0, 1)$ as:

$$\mathcal{P}_i = \begin{bmatrix} \beta, & (1-\beta) P_{i:}^{SU} \\ 0_{N_u \times 1}, & \beta I_{N_u} + (1-\beta) P^{UU} \end{bmatrix}, \tag{13}$$

where $0_{n_c \times 1}$ is a all zero vector, and I_{N_u} is an identity matrix.

We then compute the T step visit probability v_i for z_i:

$$v_i = e_1^\top \mathcal{P}_i^T, \tag{14}$$

where e_1 is a one-hot vector with the first element being 1. and finally we obtain μ_i and Σ_i as the weighted mean and variance of z_i and $\{u_j\}_{j=1}^{N_u}$, with the weight being v_i. That is: $\mu_i = \frac{1}{2} z_i + \frac{1}{2} \sum_{j=1}^{N_u} v_j u_j$, and $\Sigma_i = \frac{1}{2}(z_i - \mu_i)(z_i - \mu_i)^\top + \frac{1}{2} \sum_{j=1}^{N_u} v_j (u_j - \mu_i)(u_j - \mu_i)^\top$.

We summarize the proposed ADaptive Vicinal Few-Shot Learning (ADV) in Algorithm 1.

4 Experiment

In this section, we verify the effectiveness of the proposed approach in improving the generalization for few-shot learning. We first introduce the standard few-shot learning benchmark for performance evaluation and the implementation details. Then we present the results on the benchmarks, followed by a detailed study on the proposed approach.

We use three benchmarks for performance evaluation: *mini*ImageNet [33], CUB [34] and CIFAR-FS [2]. For all datasets, we construct 5-way 1-shot and 5-way 5-shot tasks by randomly sampling 5 classes. For performance evaluation, we construct $10,000$ meta-testing tasks and report the average classification accuracy with a 95% confidence interval.

Table 1: 5-way 1-shot and 5-way 5-shot classification accuracy (%) on miniImageNet, CUB and CIFAR-FS with 95% confidence intervals. For the backbone, CONV denotes a network consisting of 4 convolution layer; ResNet denotes ResNet-12; WRN denotes WideResNet-28. The best two accuracies are highlighted in bold.

Methods	Backbone	miniImageNet 5-way		CUB 5-way		CIFAR-FS 5-way	
		1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
MAML [6]	CONV	48.70 ± 1.84	63.11 ± 0.92	50.45 ± 0.97	59.60 ± 0.84	58.9 ± 1.9	71.5 ± 1.0
MetaSGD [17]	CONV	50.47 ± 1.87	64.03 ± 0.94	53.34 ± 0.97	67.59 ± 0.82	-	-
MatchningNet [33]	CONV	43.56 ± 0.84	55.31 ± 0.73	56.53 ± 0.99	63.54 ± 0.85	-	-
ProtoNet [27]	CONV	49.42 ± 0.78	68.20 ± 0.66	58.43	75.22	55.5 ± 0.7	72.0 ± 0.6
Relation Net [28]	CONV	50.44 ± 0.82	65.32 ± 0.70	62.45	76.11	55.0 ± 1.0	69.3 ± 0.8
R2D2 [2]	CONV	51.2 ± 0.6	68.8 ± 0.1	-	-	65.3 ± 0.2	79.4 ± 0.1
SNAIL [20]	ResNet	55.71 ± 0.99	68.88 ± 0.92	-	-	-	-
TADAM [21]	ResNet	58.50 ± 0.30	76.70 ± 0.30	-	-	-	-
MetaOptNet-SVM [16]	ResNet	62.64 ± 0.61	78.63 ± 0.46	-	-	72.0 ± 0.7	84.2 ± 0.5
Meta-Fun [35]	ResNet	62.12 ± 0.30	78.20 ± 0.16	-	-	-	-
LEO [25]	WRN	61.76 ± 0.08	77.59 ± 0.12	-	-	-	-
S2M2 [19]	WRN	64.93 ± 0.48	83.18 ± 0.72	80.68 ± 0.81	90.85 ±0.44	74.81 ± 0.19	**87.47 ± 0.13**
FreeLunch [37]	WRN	68.57 ± 0.55	82.88 ± 0.42	79.56 ± 0.87	90.67 ± 0.35	-	-
Transductive Methods							
TPN [18]	CONV	55.51 ± 0.86	69.86 ± 0.65	-	-	-	-
FEAT [39]	CONV	57.04 ± 0.20	72.89 ± 0.16	-	-	-	-
TEAM [22]	CONV	56.57	72.04	75.71	86.04	-	-
TPN [18]	ResNet	59.46	75.65	-	-	-	-
TEAM [22]	ResNet	60.07	75.90	80.16	87.17	-	-
SIB [11]	WRN	70.0 ± 0.6	79.2 ± 0.4	-	-	80.0 ± 0.6	85.3 ± 0.4
ADV-SVM	WRN	**72.18 ± 0.48**	**84.38 ± 0.30**	**83.25 ± 0.44**	**91.23 ± 0.23**	**81.12 ± 0.46**	87.41 ± 0.33
ADV-CE	WRN	**74.63 ± 0.48**	**86.12 ± 0.29**	**84.20 ± 0.43**	**91.54 ± 0.22**	**82.02 ± 0.48**	**88.25 ± 0.31**

Implementation details

We focus on meta-testing phase and directly use the WideResNet-28-10(WRN) [40] feature extractors pre-trained following the previous work [19].

We denote the application of ADV to the linear classifier with vicinal cross-entropy loss by ADV-CE and denote the application of ADV to SVM by ADV-SVM. For ADV-CE, we optimize the vicinal loss in (5) by gradient descent for 100 steps. The learning rate is determined by performing grid search from 0.001 to 1 on the tasks constructed by meta-validation set. For ADV-SVM, we solve the QP in (9) by using a differentiable GPU-based QP solver [1]. The regularization parameter λ is set to 0.1 and the parameter σ for RBF kernel is obtained via grid search from 0.1 to 10. The number of step T, lazy stay probability β and temperature τ in the lazy random walks are obtained via grid search in $\{1, 2, 3, 4, 5\}$, $\{0.1, 0.2, 0.5\}$, $\{0.01, 0.1, 1, 10\}$, respectively. Notably, ADV accesses the query set as unlabeled data to perform lazy random walks. Thus, ADV is a transductive method. The number of query samples per class is 15.

4.1 Comparison with Baselines

We present the performance of ADV-CE, ADV-SVM with a WRN feature extractor and compare the results with existing works in Table 1. The results of baselines are from [16], [22] and [11]. From Table 1 ADV achieves the best performance in all the datasets for both 1-shot and 5-shot tasks. Notable, ADV outperforms the best competitor by at least 2% for 1-shot tasks. In 1-shot tasks, the empirical distribution estimated by the support set can deviate from the true distribution by a large margin. The extraordinary performance of ADV in 1-shot tasks indicates its effectiveness to exploit the information of the unlabeled data from the query set for a more accurate distribution estimate to the true distribution of each class.

4.2 Visualization of Vicinal Distributions

The vicinal risk minimization principle in the proposed method is built on the vicinal distributions obtained by lazy random walks. Thus, the performance of ADV depends on the exactness of vicinal distribution estimate. Here, we visualize the vicinal distributions to evaluate their exactness. We run the proposed lazy random walks in Section 3.3 on two 5-way 1-shot tasks randomly constructed on the meta-testing set of miniImageNet, generate 500 samples from each vicinal distribution for

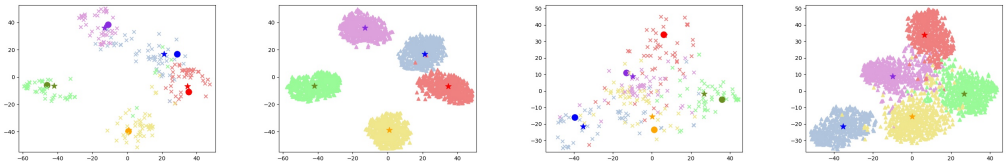


Figure 2: t-SNE visualization of the vicinal distribution and the true distribution. The first and second subfigures show the original data and the vicinal distributions for a task. The third and fourth subfigures show the results for another task. Different colors represent different classes. The '•' represents the support samples; '×' represents the query samples; '★' represents the mean of the vicinal distributions; and '▲' represents the samples generated from the vicinal distribution.

Table 2: Performance with different backbones on 5-way 1-shot and 5-way 5-shot classification accuracy (%) on miniImageNet with 95% confidence intervals.

Backbone	CONV		ResNet		WRN	
Methods	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
Kernel SVM	55.14 ± 0.62	68.37 ± 0.51	57.85 ± 0.63	73.47 ± 0.51	64.77 ± 0.44	80.79 ± 0.32
ADV-SVM	59.02 ± 0.70	**72.13 ± 0.53**	64.27 ± 0.74	76.37 ± 0.54	72.18 ± 0.48	84.38 ± 0.30
CE	54.41 ± 0.61	70.74 ± 0.52	56.87 ± 0.65	76.73 ± 0.49	65.38 ± 0.52	83.33 ± 0.30
ADV-CE	**59.22 ± 0.68**	72.07 ± 0.51	**65.03 ± 0.74**	**77.12 ± 0.52**	**74.63 ± 0.48**	**86.12 ± 0.29**

visualization, and perform t-SNE algorithm on all data. The visualization results are presented in Figure 2.

From Figure 2, we can observe that the vicinal distribution generated by ADV covers most of the query samples from the same class. This validates that the lazy random walks module in ADV can provide a good estimate of the vicinal distribution, which overlaps with the ground truth distribution. By optimizing the expected loss over the vicinal distributions, the model learned by ADV has a smaller probability of overfitting the support samples and has a better generalization to the query set.

4.3 Effect of the Network Architecture of Feature Extractor

The proposed algorithm ADV can be applied on any pre-trained feature extractor. Here, we validate the effectiveness of ADV on three feature extractors with different backbones that are trained by different meta-learning methods. We consider WideResNet (WRN) [40], ResNet-12 [9] and a shallow network of four convolutional blocks (CONV) [2]. We meta-train the WRN, ResNet, CONV following [19], [16] and [2], respectively.

We report the performance of a kernel SVM classifier and a linear classifier trained by cross-entropy (denoted by CE) with and without ADV. The results in Table 2 show the performance improvement of ADV is consistent over all three different feature extractors. This verifies that the effectiveness of ADV is feature extractor-independent. It is worth noting the larger performance gains are observed on the better feature extractor. For example, the overall performance of WRN is better than CONV, and the performance boost of ADV-CE over CE of 1-shot tasks of WRN is 10.5%, which is larger than the boost on CONV (4.8%). This observation indicates: 1) even with a superior feature extractor that well preserves the similarity between data in feature space, the classifier trained from a limited number of high-quality support data still overfits the data; 2) for a better feature extractor, ADV can exploit more information from the similarity between data to estimate the vicinal distribution of support data and alleviate the overfitting.

4.4 Ablation Study

We perform an ablation study on ADV about the gain of the VRM principle. To do this, we train a vanilla linear classifier by cross-entropy loss (denoted by CE), a linear classifier with Mixup vicinal distribution [41] (denoted by CE-Mixup), a linear classifier with Delta vicinal distribution whose

Table 3: Ablation Study of the proposed ADV-CE algorithm on 5-way 1-shot and 5-way 5-shot classification accuracy (%) on miniImageNet, CUB and CIFAR-FS with 95% confidence intervals.

Methods	miniImageNet 5-way		CUB 5-way		CIFAR-FS 5-way	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
CE	65.38 ± 0.52	82.03 ± 0.32	79.40 ± 0.46	90.36 ± 0.24	74.06 ± 0.47	87.74 ± 0.32
CE-Mixup	65.53 ± 0.43	82.77 ± 0.30	79.02 ± 0.45	90.14 ± 0.25	74.99 ± 0.47	87.45 ± 0.32
ADV-CE-DD	72.37 ± 0.49	84.79 ± 0.31	83.04 ± 0.46	90.61 ± 0.25	80.56 ± 0.47	88.11 ± 0.31
ADV-CE	**74.63 ± 0.48**	**86.12 ± 0.29**	**84.20 ± 0.43**	**91.54 ± 0.22**	**82.02 ± 0.48**	**88.25 ± 0.31**
SVM	64.33 ± 0.44	83.57 ± 0.30	78.46 ± 0.45	90.93 ± 0.23	74.29 ± 0.46	87.31 ± 0.32
Kernel SVM	64.77 ± 0.44	80.79 ± 0.32	77.69 ± 0.45	88.96 ± 0.27	74.03 ± 0.46	86.28 ± 0.34
ADV-SVM-DD	65.33 ± 0.43	81.36 ± 0.32	80.97 ± 0.48	90.94 ± 0.23	80.21 ± 0.48	87.37 ± 0.33
ADV-SVM	**72.18 ± 0.48**	**84.38 ± 0.30**	**83.25 ± 0.44**	**91.23 ± 0.23**	**81.12 ± 0.46**	**87.41 ± 0.33**

mean locates on the same mean estimate by our lazy random walks module and whose variance is zero (denoted by ADV-CE-DD), and compare them with ADV-CE. Note that by using a Delta vicinal distribution, only the mean will contributes to the loss. We also train a multi-class SVM classifier, a multi-class kernel SVM classifier, and a kernel SVM classifier with Delta vicinal distribution (denoted by ADV-SVM-DD). The training of multi-class SVM classifier all follows [5]. We report the performance on three benchmarks in Table 3.

In Table 3, we can observe that the performance of CE-Mixup is just competitive with CE in miniImageNet and CIFAR-FS, and even inferior to CE on CUB dataset. This suggests the Mixup vicinal distribution fails to improve the generalization given a limited number of data. In contrast, the vicinal distribution obtained by the lazy random walk algorithm is a more accurate estimate for the true distribution, as evidenced by the fact that the performance of ADV-CE-DD and ADV-CE is better than CE and CE-Mixup. To be concrete, the performance boost of ADV-CE-DD over CE evidences the mean of vicinal distribution, which does not located on the original sample, but on the weighted mean of the data passing through by the lazy random walks, is closer to the ground-truth distribution. The performance improvement of ADV-CE over ADV-CE-DD demonstrates that the variance of the vicinal distribution helps to alleviate the overfitting problem.

As shown in Table 3, the performance of Kernel SVM is worse than SVM in most cases, which may be due to the linear separability of the features extracted by WRN, while kernel does not help to improve the separability. Though the performance of Kernel SVM is inferior to SVM, we observe performance boost of ADV-SVM, which is built on Kernel SVM. This indicates that learning from the vicinal distributions that are close to the true distribution would sufficiently improve the performance. We also observe the performance gain of ADV-SVM over ADV-SVM-DD, again evidencing the importance of the variance of vicinal distributions in alleviating the overfitting.

5 Conclusion and Limitations

In this paper, we discuss the overfitting issue in few-shot learning, which is caused by the deviation of empirical distribution from the ground-truth distribution. To alleviate the overfitting, we revisit the Vicinal Risk Minimization principle in deep few-shot learning and propose to harness unlabeled data and a lazy random walk algorithm to generate adaptive statistical parameters for the vicinal distribution of each training sample. Following the VRM principle, we learn the model by optimizing the resulting expected vicinal loss. We derive the vicinal loss for linear classifiers trained with cross-entropy loss and for SVM classifier and develop the resulting Adaptive Vicinal classifiers as ADV-CE and ADV-SVM. We evaluate the performance of ADV-CE and ADV-SVM on three benchmarks and validate the effectiveness of ADV in mitigating overfitting and improving generalization of classifiers learned from a limited number of samples. Currently, our proposed ADC is only compatible with CE and SVM. We would extend it to other loss functions. For the vicinal Cross-Entropy loss, we optimize an approximation obtained by second-order Taylor expansion. However, the approximation error can be large. Thus, the performance of ADV-CE may be suboptimal and we can further improve it by deriving a better approximation for the vicinal Cross-Entropy loss.

References

[1] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.

[2] Luca Bertinetto, Joao F Henriques, Philip HS Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. *ICLR*, 2019.

[3] Olivier Chapelle, Jason Weston, Léon Bottou, and Vladimir Vapnik. Vicinal risk minimization. *Advances in neural information processing systems*, pages 416–422, 2000.

[4] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

[5] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of machine learning research*, 2(Dec):265–292, 2001.

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[7] Jason Freeman. Metamix: Between unity and collaboration. 2002.

[8] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

[9] 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.

[10] Ruibing Hou, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. Cross attention network for few-shot classification. *arXiv preprint arXiv:1910.07677*, 2019.

[11] Shell Xu Hu, Pablo G Moreno, Yang Xiao, Xi Shen, Guillaume Obozinski, Neil D Lawrence, and Andreas Damianou. Empirical bayes transductive meta-learning with synthetic gradients. *arXiv preprint arXiv:2004.12696*, 2020.

[12] Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D Yoo. Edge-labeling graph neural network for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11–20, 2019.

[13] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[15] Seong Min Kye, Hae Beom Lee, Hoirin Kim, and Sung Ju Hwang. Meta-learned confidence for transductive few-shot learning. 2020.

[16] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10657–10665, 2019.

[17] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

[18] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sung Ju Hwang, and Yi Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. *arXiv preprint arXiv:1805.10002*, 2018.

[19] Puneet Mangla, Nupur Kumari, Abhishek Sinha, Mayank Singh, Balaji Krishnamurthy, and Vineeth N Balasubramanian. Charting the right manifold: Manifold mixup for few-shot learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2218–2227, 2020.

[20] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

[21] Boris N Oreshkin, Pau Rodriguez, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. *arXiv preprint arXiv:1805.10123*, 2018.

[22] Limeng Qiao, Yemin Shi, Jia Li, Yaowei Wang, Tiejun Huang, and Yonghong Tian. Transductive episodic-wise adaptive metric for few-shot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3603–3612, 2019.

[23] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. *arXiv preprint arXiv:1909.09157*, 2019.

[24] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*, 2018.

[25] Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018.

[26] Sahil Singla, Eric Wallace, Shi Feng, and Soheil Feizi. Understanding impacts of high-order loss approximations and features in deep learning interpretation. In *International Conference on Machine Learning*, pages 5848–5856. PMLR, 2019.

[27] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087, 2017.

[28] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, 2018.

[29] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. *arXiv preprint arXiv:1903.03096*, 2019.

[30] Vladimir N Vapnik. The vicinal risk minimization principle and the svms. In *The nature of statistical learning theory*, pages 267–290. Springer, 2000.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[32] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, pages 6438–6447. PMLR, 2019.

[33] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[34] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[35] Jin Xu, Jean-Francois Ton, Hyunjik Kim, Adam R Kosiorek, and Yee Whye Teh. Metafun: Meta-learning with iterative functional updates. *arXiv preprint arXiv:1912.02738*, 2019.

[36] Ling Yang, Liangliang Li, Zilun Zhang, Xinyu Zhou, Erjin Zhou, and Yu Liu. Dpgn: Distribution propagation graph network for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13390–13399, 2020.

[37] Shuo Yang, Lu Liu, and Min Xu. Free lunch for few-shot learning: Distribution calibration. *ICLR*, 2021.

[38] Huaxiu Yao, Long-Kai Huang, Linjun Zhang, Ying Wei, Li Tian, James Zou, Junzhou Huang, et al. Improving generalization in meta-learning via task augmentation. In *International Conference on Machine Learning*, pages 11887–11897. PMLR, 2021.

[39] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

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

[41] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *ICLR*, 2018.

[42] Dengyong Zhou and Bernhard Schölkopf. Learning from labeled and unlabeled data using random walks. In *Joint Pattern Recognition Symposium*, pages 237–244. Springer, 2004.

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.

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? [Yes] . See Section 5
   (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? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]

(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)? [No]

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]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [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]