Targeted Data-driven Regularization for Out-of-Distribution Generalization

Mohammad Mahdi Kamani, Sadegh Farhang, Mehrdad Mahdavi and James Z. Wang
{mqk5591,smf5604,mzm616,jwang}@psu.edu
The Pennsylvania State University, University Park, Pennsylvania

ABSTRACT

Due to biases introduced by large real-world datasets, deviations of deep learning models from their expected behavior on out-of-distribution test data are worrisome. Especially when data come from imbalanced or heavy-tailed label distributions, or minority groups of a sensitive feature. Classical approaches to address these biases are mostly data- or application-dependent, hence are burdensome to tune. Some meta-learning approaches, on the other hand, aim to learn hyperparameters in the learning process using different objective functions on training and validation data. However, these methods suffer from high computational complexity and are not scalable to large datasets. In this paper, we propose a unified data-driven regularization approach to learn a generalizable model from biased data. The proposed framework, named as **targeted data-driven regularization** (TDR), is model- and dataset-agnostic, and employs a target dataset that resembles the desired nature of test data in order to guide the learning process in a coupled manner. We cast the problem as a bilevel optimization and propose an efficient stochastic gradient descent based method to solve it. The framework can be utilized to alleviate various types of biases in real-world applications. We empirically show, on both synthetic and real-world datasets, the superior performance of TDR for resolving issues stem from these biases.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; **Machine learning algorithms**; **Regularization**; **Cost-sensitive learning**; *Batch learning*; **Stochastic games**; *Neural networks*.

KEYWORDS

out-of-distribution generalization, data-driven regularization, bilevel programming

ACM Reference Format:
Mohammad Mahdi Kamani, Sadegh Farhang, Mehrdad Mahdavi and James Z. Wang. 2020. Targeted Data-driven Regularization for Out-of-Distribution Generalization. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3394486.3403131

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403131

1 INTRODUCTION

Drastically improving their performance, machine learning, and more distinctively, deep learning models, are becoming the main propulsion of technology in a variety of domains. Notwithstanding their success, they still suffer from different forms of biases in the training data distribution. Biases, regardless of their nature, cause a mismatch between training and testing data distributions, which leads to a poor out-of-distribution generalization performance of the model. Machine learning models inherit these biases due to the only objective of minimizing the empirical risk on the training data in their learning process. However, empirical risk by itself seems incapable of avoiding these biases in training data for better out-of-distribution generalization, and needs to be accompanied by other objectives [35].

These biases can appear in different forms in training a machine learning model. A palpable form of them happens when the size of different classes or groups are unbalanced. When class sizes are not balanced, the imbalanced dataset problem stems [9, 24, 44], where majority classes' distribution can dominate the training process, resulting in a model with low accuracy on minority classes. A severe form of imbalanced dataset problem, appears in most real-world big datasets with immense number of classes, is long-tailed data distribution [7, 14, 38], where the distribution of classes is skewed. In this case, most of the data belongs to a few prevailing classes, while a large number of classes are represented by a few number of samples. Another form of bias happens when the sample size of different groups in a categorical feature are unbalanced (*e.g.*, gender). As a result, fairness issues affect the training process, resulting in a biased performance toward minority groups in the dataset [22]. Other forms of these biases include label noise [26], source and target distribution mismatch [42], spurious inherent feature correlation [5], to name but a few.

While there are a variety of different ideas for addressing each of the aforementioned biases in the literature, there is a lack of a unifying approach that can address all different forms. For the class imbalance problem, a generic idea is to adapt the training distribution to the desired properties of an unbiased dataset, whether by resampling or assigning weights based on training loss to have a cost-sensitive weighting scheme [16, 33]. On the other side, problems raised from fairness issues are handled by pre-processing the data [32], post-processing the models' output [22, 31], or during the training using constrained optimization [15]. To address the label noise problem, approaches are mostly to try to detect the affected samples and reduce their effects on the training procedure either by late sampling [8, 21] or reweighting the probability of sampling [26] of those samples. However, relying merely on the training distribution has shown to be not practically efficient [41], especially for out-of-distribution generalization [5].

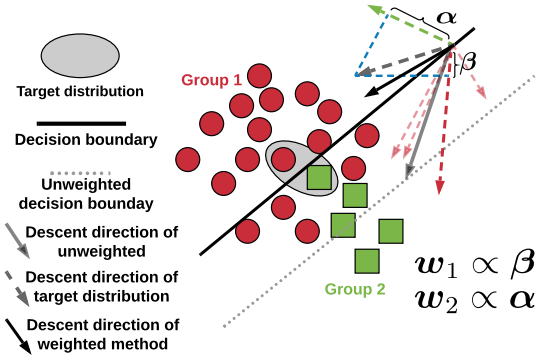


Figure 1: In a standard (unweighted) gradient-descent binary classification with unbalanced data between two classes (Group 1 versus Group 2), the decision boundary is mostly determined based on the majority class's descent direction. In targeted data-driven regularization, we use a small and *unbiased* target dataset, with which we can reweight each class's gradients based on their inner products with the gradient of the target dataset. This will decrease the weight of majority class and increase the weight of the minority class. See Section 3.3 for more details.

A new strand of research is to augment single-objective learning models with additional data-driven constraints in order to alleviate the effect of bias in training data [13, 20, 26, 37, 41]. The main idea behind constrained learning schema stems from the observation that the accuracy on training data is not a satisfactory criterion by itself, and should be accompanied by a data-driven regularization. An appealing data-driven regularization idea is to create a *target dataset* that resembles the desired properties of the unbiased distribution, and impose it to the training process as an additional constraint. A common practice of tuning hyperparameters at the end of training using a validation set is a simple example of this regularization, which is a cumbersome task. Some meta-learning approaches like [18, 41], on the other hand, introduce a coupled framework to interlace the hyperparameter tuning using a validation set with the training process in order to guide it.

While data regularized methods have shown to be successful in some applications, the computational burden of tuning hyperparameters limits their scalability to large datasets or training data with skewed classes. Moreover, these methods are usually application specific and lack a principled way to be generalized to different types of biases discussed before. To overcome these challenges, in this paper, we introduce the **targeted data-driven regularization** (TDR) framework, which employs a small target distribution, free of those biases in training data, and bilevel programming [12] to model the multi-objective structure on both training and target distributions. As a bilevel programming, TDR has two levels, one dealing with the main training process, while the other uses a well-tuned target dataset in order to optimize the weights of each desired class or group in the dataset. As we elaborate later, the weight for each group is proportioned to the inner product of the gradients of target dataset and each group's gradient vectors. Hence, as it is depicted in Figure 1, TDR, despite the standard training, learns the optimal weight for each group to prevent the majority groups to prevail in finding the descent direction in gradient descent approaches.

The main **contributions** of this work are:

- We propose a unified framework for out-of-distribution generalization that exploits a small well-crafted target distribution to guide the learning process and cast it to a bilevel programming problem to learn the optimal model. The framework is generic and can be utilized to tackle different types of biases and is free of hyperparameter tuning of other frameworks.
- Our framework addresses the computation and stability issues of previous frameworks, *e.g.* [41], while greatly improving the performance of the model.
- We propose an efficient stochastic algorithm for TDR and validate it via synthetic and real-world datasets on various problems.

2 RELATED WORK

Two main challenges raised from biases in datasets are *class imbalance* and *fairness* problems. Contrary to fairness, studying imbalanced datasets is not a new problem in machine learning, hence there are a number of approaches to rectify this issue.

Class Imbalance: Most of the approaches revolve around adopting the biased training distribution to the desired properties of an unbiased one, by either re-sampling or cost-sensitive mechanisms. In over-sampling, we add samples from minority classes to the dataset via either repetition or novel approaches like domain adaptation or synthetic data generation [4] and interpolation of neighboring samples [11]. Cost-sensitive learning, on the other hand, is the method of assigning weights to each sample's loss based on the data distribution [27, 33]. It is common to choose weights as the inverse of class frequency for imbalanced datasets [24]. Recent efforts use more intuitive approaches than inverse of class frequency for reweighting samples. For instance, Cui et al. [14], based on the sample size of each class calculated the effective number of samples for that class, and balanced the loss based on that number. Our *targeted* framework for out-of-distribution generalization belongs to the latter group, where we use an unbiased and small target dataset to learn the weights of each group interleaved with the main training procedure. Having these weights trained during the main training task is the advantage of our framework over others.

Fairness: Algorithmic fairness in machine learning has attracted much interest in recent years. The efforts on algorithmic fairness are divided into three categories. The first approach is to alter the pretrained classifier for fairness improvement [22, 40]. Approaches taking into account fairness during the training phase belong to the second category [2, 15, 45]. In the third category, fairness can be achieved via modifying the data while using standard machine learning algorithms [10]. In the context of fairness, our approach belongs to the second approach, where we update the weight for each group to compensate for the bias in the training data.

Meta-learning: Apart from all these classical approaches, which are mostly burdensome to tune for each specific dataset or application, we can benefit from the new trend arising in machine

learning domain, called meta-learning [3, 17]. While meta-learning approaches are mostly used in few-shot and multi-task learning settings, they have also been adopted in hyperparameter optimization tasks [18] or addressing noise in data [25]. These frameworks aim at learning the parameters of the optimization on top of the main learning process using gradient descent. Following this trend, Ren et al. [41] proposed a new framework to address biases in the dataset, using a validation set, similar to test set and by using the perturbation idea of [34], they perturb each example's weight in the training. Although its effectiveness is comparable to classical approaches, it encounters several issues, such as high computational complexity and instability of the perturbed weights, making it impractical in large-scale problems. Litany and Freedman [37] proposed a very similar framework to example reweighting for transfer learning. In TDR, we use bilevel programming to develop a framework for addressing biases in the dataset, while avoiding the aforementioned issues. For a thorough discussion of existing methods please refer to Appendix A.

Out-of-distribution Generalization: Attracting much attention recently, various frameworks have been proposed for the problem of out-of-distribution generalization. Shankar et al. [42] suggest a combination of gradients on different domains to learn a model that generalizes well to new domains, and does not need domain signals like prior methods. Sun et al. [43] introduce a training approach during the inference time to adapt the model to new domains. Arjovsky et al. [5] try to make the model invariant to the spurious correlation in the feature data. The main advantage of TDR over these approaches is that we use a small target dataset that can be adopted based on different applications, instead of using the whole dataset from other domains.

3 TARGETED DATA-DRIVEN REGULARIZATION (TDR)

In this section, we introduce the TDR framework and cast it as a bilevel optimization problem with two objectives. Then, we propose an efficient two-level stochastic optimizer to learn the model.

3.1 The Proposed Framework

The way we define two objective functions and their parameters is problem-independent. However, for the sake of exposition, we tackle problems such as imbalanced dataset and fairness in classification models. These two types of problems have a common ingredient, *i.e.*, unbalanced distribution of different categories in the dataset. If this unbalanced nature of the data happens in the labels, the problem is imbalanced dataset. But if one of the sensitive categorical features in the dataset (*e.g.*, gender) happens to be unbalanced, the issue of fairness would raise. In both scenarios, we are solving a prediction problem on dataset $\mathcal{T}$ with n training samples, from input space $\mathcal{X}$ to label domain $\mathcal{Y}$, where each sample point is defined as $(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. We use $g_i = \ell\,(\boldsymbol{\theta}; (\boldsymbol{x}_i, y_i))$, where $\ell(\cdot;\cdot)$ is the training loss function and $\boldsymbol{\theta}$ is the parameters of the model, to denote the training loss on ith sample $(\boldsymbol{x}_i, y_i) \in \mathcal{T}$.

In order to address the aforementioned biases, we need to weight loss of samples from different classes or groups separately; hence, we define a weight vector, $\boldsymbol{w} \in \mathbb{R}_+^c$, where c is the number groups of the discriminatory feature (*i.e.*, label in imbalanced data or sensitive feature in fair learning). Let $\mathbf{D} \in \{0,1\}^{n\times c}$ denote the assignments of n training samples to c groups of the discriminatory feature. For a model parameter $\boldsymbol{\theta}$ and a fixed weight vector $\boldsymbol{w}$, we define the loss over training examples as

$$G(\boldsymbol{w}, \boldsymbol{\theta}; \mathcal{T}) = (\mathbf{D}\boldsymbol{w})^\top \boldsymbol{g}\,, \tag{1}$$

where $\boldsymbol{g} = [g_1, \ldots, g_n]^\top$ is the vector of losses per sample, and $G(.,.;.)$ is the weighted loss over training samples.

Equipped with Eq. (1) as the training goal, for a known weight vector $\boldsymbol{w}$, we can find the optimal parameters $\boldsymbol{\theta}$ by minimizing the objective. However, we use the samples in the target dataset to adaptively learn the optimal weight vector and guide the training process. To this end, we define the loss over a *small unbiased target dataset* $\mathcal{V}$ as:

$$F(\boldsymbol{w}, \boldsymbol{\theta}^*(\boldsymbol{w}); \mathcal{V}) = \frac{1}{|\mathcal{V}|} \sum_{(\boldsymbol{x}_i, y_i)\in\mathcal{V}} f(\boldsymbol{\theta}^*(\boldsymbol{w}); (\boldsymbol{x}_i, y_i))\,, \tag{2}$$

where $|\mathcal{V}|$ is the number of samples in $\mathcal{V}$, $\boldsymbol{\theta}^*(\boldsymbol{w})$ is the minimizer of the loss function in Eq. (1), and $f(\cdot;\cdot)$ is the target loss function which may or may not be same as training loss $\ell(\cdot;\cdot)$. We emphasize that the target dataset could be a part of training dataset or it could be separated, similar to a validation dataset.

Having two interlaced optimization problems, we can cast the TDR as a game between two players, called leader and follower [6]. Both players want to minimize their specific objective functions which results in the following bilevel programming:

$$\begin{aligned} \boldsymbol{w}^* \in \arg\min_{\boldsymbol{w}} \;\; & F(\boldsymbol{w}, \boldsymbol{\theta}^*(\boldsymbol{w}); \mathcal{V}) \\ \text{s.t.} \;\; & \boldsymbol{\theta}^*(\boldsymbol{w}) \in \arg\min_{\boldsymbol{\theta}} G(\boldsymbol{w}, \boldsymbol{\theta}; \mathcal{T})\,, \end{aligned} \tag{3}$$

where the objective of the leader and the follower, $F(\cdot;\cdot)$ and $G(\cdot;\cdot)$, are defined in Eqs. (2) and (1), respectively.

It is worth noting that despite the similarity of this algorithm with universal bilevel programming, the two levels are being optimized on different data distributions. This is the key difference that makes the TDR framework capable of solving some challenging problems with a data-driven regularization using a target dataset.

We note that unlike constrained optimization problems, where first-order algorithms such as gradient descent (GD) or stochastic gradient descent can be easily utilized, in the above formulation, to find the optimal $\boldsymbol{w}$, we need to first solve the inner objective $G(\cdot;\cdot)$ with respect to $\boldsymbol{\theta}$ which depends on $\boldsymbol{w}$. As a result, any iteration of GD algorithm requires fully optimizing the inner optimization which makes the optimization intrinsically hard. In next subsection, we propose an efficient stochastic optimization algorithm to approximately solve the stated bilevel optimization problem.

3.2 Stochastic Bilevel Optimizer

In general, if we want to solve a bilevel program, we have to solve each level to reach a local minimum. However, in order to control the computational complexity of the algorithm, we propose an stochastic method where we iteratively update the parameters of the outer optimization problem. In this setting, instead of solving the inner problem completely per outer iteration, we only take few gradient steps over its parameters. Then, we update the weights of each group using a gradient step over the target dataset. We continue these loops until we reach a convergence or early stop

Algorithm 1: Targeted Data-driven Regularization

input $\hat{\theta}_0 \in \mathbb{R}^d$, $\boldsymbol{w}_0 \in \mathbb{R}^c$, η_{in}, η_{out}, t_{in}
for $k = 0, 1, \ldots$ **do**
  set: $\boldsymbol{\theta}_0 = \hat{\theta}_k$
  for $m = 0, 1, \ldots, t_{in} - 1$ **do**
    $\boldsymbol{\theta}_{m+1} = \boldsymbol{\theta}_m - \eta_{\text{in}} \cdot \nabla_{\boldsymbol{\theta}} \tilde{g}(\boldsymbol{w}_k, \boldsymbol{\theta})$
  end
  set: $\hat{\theta}_k = \boldsymbol{\theta}_{t_{\text{in}}}$
  $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \eta_{\text{out}} \cdot \nabla_{\boldsymbol{w}} \tilde{f}(\boldsymbol{w}, \hat{\theta}_k)$
end
return $\hat{\theta}_k$

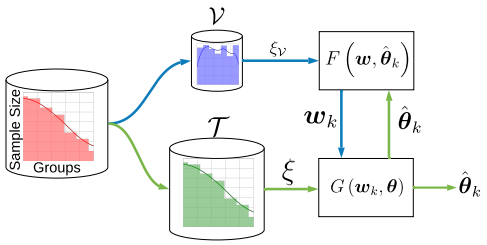


Figure 2: The proposed targeted data-driven regularization for out-of-distribution generalization.

at the best solution. This requires to define a number for inner iterations t_{in}, which indicates how many stochastic steps we have to take each time we are in the inner problem. We note that, similar to [19], t_{in} could be tuned adaptively to a monotonically increasing sequence, because in the beginning we might be far away from a local minimum of the inner problem. As we move along in the training process, we are getting closer to the local minimum of the inner problem, and hence we might need more stochastic steps toward it before updating weights.

The inner problem in TDR is minimizing the objective function of the main problem, defined in Eq. (1), with respect to its parameter set, $\boldsymbol{\theta}$, on the training data distribution $\mathcal{T}$. Using mini-batch gradient descent, we want to minimize the inner problem as follows:

$$\boldsymbol{\theta}^*(\boldsymbol{w}) \in \arg\min_{\boldsymbol{\theta} \in \Theta} \{ \tilde{g}(\boldsymbol{w}, \boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\xi \sim \mathcal{T}} [\, G(\boldsymbol{w}, \boldsymbol{\theta}; \xi) \,] \} \,, \tag{4}$$

where Θ is the domain of all feasible solutions for $\boldsymbol{\theta}$, ξ is a mini-batch drawn from the training data distribution $\mathcal{T}$ with the mini-batch size of $|\xi| = b$ and $\tilde{g}(\cdot)$ denotes the empirical loss over ξ. Then the outer problem gets the solution of the inner problem and minimizes its own objective function, defined in Eq. (2), over its parameter set, $\boldsymbol{w}$, using data samples of the target dataset $\mathcal{V}$:

$$\boldsymbol{w}^* \in \arg\min_{\boldsymbol{w} \in \Omega} \{ \tilde{f}(\boldsymbol{w}, \boldsymbol{\theta}^*(\boldsymbol{w})) = \mathop{\mathbb{E}}_{\xi_v \sim \mathcal{V}} \left[F(\boldsymbol{w}, \boldsymbol{\theta}^*(\boldsymbol{w}); \xi_v) \right] \} \,, \tag{5}$$

where Ω is the feasible domain for $\boldsymbol{w}$, and ξ_v is a mini-batch drawn from the target distribution with the size $|\xi_v| = b_v$ and $\tilde{f}(\cdot)$ is the loss over ξ_v. The key point that connects two levels together is the solution of the inner level, which is a function of $\boldsymbol{w}$. The proposed generic TDR framework is defined in Algorithm 1, in which the procedure is illustrated in Figure 2.

For both aforementioned use cases of TDR, namely, imbalanced dataset and fairness issues, we need a discriminatory feature vector. In imbalanced dataset scenario, this vector is the label of samples. In fairness problems, generally, there is an unbalanced categorical feature, which turns to an unfair classifier in favor of the majority category of that feature in the dataset. As an example, when in a dataset the numbers of samples for different genders are unbalanced, the resulting class is inclined to have a better performance in terms of accuracy for the majority category rather than the minority group. In this case, the discriminatory vector is a feature vector from the data itself, where we want to optimally weight each category in the classification. The weight for each category can be seen as a sampling probability, where in a simple stochastic GD it is considered to be a uniform distribution, while in TDR we find the optimal probability distribution for categories. Thus, to make these weights resemble a probability distribution, we have to impose two constraints on the weights. The weights for each sample in the mini-batch in step k is $\tilde{\boldsymbol{w}}_k \in \mathbb{R}^{b \times 1}$, and it is defined as:

$$\boldsymbol{w}_k^+ = \max(0, \boldsymbol{w}_k) \,, \tag{6}$$

$$\tilde{\boldsymbol{w}}_k \triangleq \frac{\tilde{\mathbf{D}} \boldsymbol{w}_k^+}{\| \tilde{\mathbf{D}} \boldsymbol{w}_k^+ \|_1} \,, \tag{7}$$

where $\boldsymbol{w}_k$ comes from step k in Algorithm 1, and $\tilde{\mathbf{D}}$ is the stochastic counterpart of the discrimination matrix for mini-batch ξ_k. We want each sample's weight in the mini-batch to be positive and also sum over all samples' weights in the mini-batch to be one. Hence, we normalize weights of samples in the mini-batch with their l_1 norm to be in range of $[0, 1]$ and sums up to 1.

The training framework defined in Algorithm 1 is straightforward and can be implemented using any machine learning or deep learning libraries such as Tensorflow [1] and PyTorch [39][1]. The main challenge is the connection between inner and outer models. Because the outer objective function is an implicit function of the weights used in the inner level, we should create this relationship in the model construction time so the derivative of the outer objective function with respect to the weights could be algorithmically computable. For instance, in Tensorflow using Estimator API, after construction of the inner model's graph, we can create the outer model's graph using $\boldsymbol{\theta}_{\text{out}} = \boldsymbol{\theta}_{\text{in}} - \eta_{\text{in}} \nabla_{\boldsymbol{\theta}} \tilde{g}(\boldsymbol{w}, \boldsymbol{\theta})$, which is simply the update rule of the inner problem. Thus, the parameters of the outer model would be an implicit function of weights.

3.3 Weights Interpretation

To better understand the learned weights for each class or group during the training, we further elaborate on the update rules of Algorithm 1. First, in the inner loop, we update the parameters of the model by propagating weighted gradients of t_{in} mini-batches. Thus, the update rule of the inner level at the step k of the outer level can be rewritten as:

$$\begin{aligned} \boldsymbol{\theta}_{m+1} &= \boldsymbol{\theta}_m - \eta_{\text{in}} \cdot \nabla_{\boldsymbol{\theta}} \tilde{g}(\boldsymbol{w}_k, \boldsymbol{\theta}) \\ &= \boldsymbol{\theta}_m - \frac{\eta_{\text{in}}}{|\xi|} \cdot \sum_{i=1}^{|\xi|} w_k^{c_i} \cdot \frac{\partial g_i(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \bigg|_{\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}_{\boldsymbol{m}}} \,, \end{aligned} \tag{8}$$

[1]The code repository https://github.com/mmkamani7/Targeted-Meta-Learning

where $w_k^{c_i}$ is the weight of the ith sample in the mini-batch, which belongs to class or group c_i. Now, we can write the update rule of the outer level for weights of each class or group separately as follows:

$$
\begin{aligned}
w_{k+1}^c &= w_k^c - \eta_{\text{out}} \cdot \left[\nabla_{\boldsymbol{w}} \tilde{f}(\boldsymbol{w}, \hat{\boldsymbol{\theta}}_k)\right]_c \\
&= w_k^c - \frac{\eta_{\text{out}}}{|\xi_v|} \cdot \sum_{j=1}^{|\xi_v|} \frac{\partial f_j\left(\boldsymbol{\theta}(\boldsymbol{w})\right)}{\partial \boldsymbol{w}^c}\Bigg|_{\boldsymbol{w}=\boldsymbol{w}_k} \\
&= w_k^c - \frac{\eta_{\text{out}}}{|\xi_v|} \cdot \sum_{j=1}^{|\xi_v|} \left(\frac{\partial f_j\left(\boldsymbol{\theta}(\boldsymbol{w})\right)}{\partial \boldsymbol{\theta}}\right)^\top \left(\frac{\partial \boldsymbol{\theta}\left(\boldsymbol{w}\right)}{\partial \boldsymbol{w}^c}\right) \\
&= w_k^c + \frac{\eta_{\text{out}}\eta_{\text{in}}}{|\xi_v||\xi|} \cdot \left(\sum_{j=1}^{|\xi_v|} \frac{\partial f_j\left(\boldsymbol{\theta}(\boldsymbol{w})\right)}{\partial \boldsymbol{\theta}}\right)^\top \left(\sum_{\substack{i=1 \\ c_i=c}}^{|\xi|} \frac{\partial g_i(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right),
\end{aligned}
\tag{9}
$$

where $f_j(\cdot)$ is the empirical risk of the jth sample of the mini-batch drawn from the target dataset and $[\cdot]_c$ is the cth element of the vector.

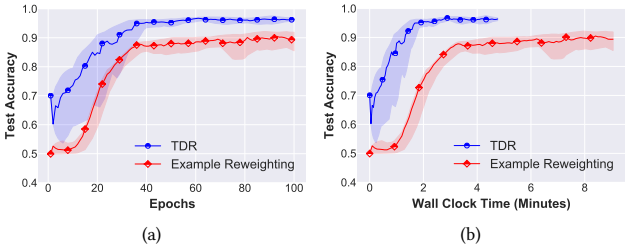
The final equality reveals an elegant property about the learned weights in the TDR framework. Indeed, it demonstrates that the weight of each class or group changes with the conformity of the average gradients of samples from that class or group with the average gradients of the target dataset. As long as the average gradients of one group is aligned with the average gradient of the target dataset, its weight will increase; otherwise, it will decrease. In this case, when the average gradients of a majority group is not highly correlated with the average gradients of the target dataset, they fail to dominate the descent direction for updating the parameters of the network.

4 EXPERIMENTAL RESULTS

We provide experimental results for the TDR algorithm to show its efficacy in dealing with imbalanced datasets, as well as fairness concerns in a classification problem. To show the performance of the TDR framework on imbalanced dataset problem, we use the MNIST[2] and CIFAR10[3] datasets. In addition, we apply TDR to a real-world dataset for severe weather prediction consisting of radar images of the whole continental United States using data gathered from NEXRAD level III radars (or WSR-88D)[4] of the National Weather Services. For demonstrating how TDR could preserve fairness in classification tasks, we use the *Adult*[5] dataset, where the goal is to predict the range of people's salary based on some demographic information of each person. Because the number of samples for male and female is unbalanced in this dataset, normal classification tasks would have poor accuracy on minority groups in this dataset. In all these experiments we use $t_{\text{in}} = 10$.

4.1 Imbalanced and Long-tailed Datasets

As stated in Section 3, in an imbalanced dataset scenario, the discriminatory vector is the label vector in the dataset. Thus, in this case for each class we learn a weight and update it based on the gradient descent direction of the target objective function. In order to better measure the imbalance level in a dataset, we introduce an *imbalance ratio* (ρ) which is the ratio between the number of samples in the smallest class (n_{m}) and the largest class (n_{M}), *i.e.*, $\rho \triangleq n_{\text{m}}/n_{\text{M}}$. Using TDR, we train networks on imbalanced and long-tailed data distributions using a small balanced target distribution. In order to demonstrate the robustness of our algorithm, we compare it to example reweighting [41]. In this experiment, we choose $b = 50$, $b_v = 10$, $\eta_{\text{in}} = 0.001$, and $\eta_{\text{out}} = 0.2$.



(a) (b)

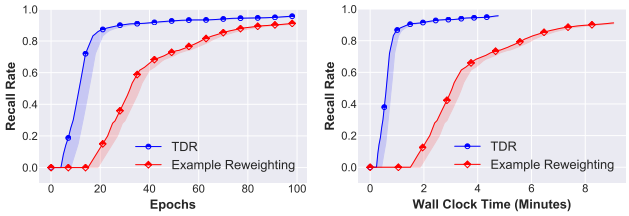
Figure 3: Test accuracy on imbalanced dataset containing two classes ('4' and '9') of MNIST. The test dataset is a balanced dataset of the two classes. Total number of training samples is $5,000$ and class 4 is in the smaller class with the imbalance ratio of $\rho = 0.005$. Target distribution has 50 balanced samples. TDR outperforms example reweighting, both in terms of accuracy and speed.

4.1.1 Synthetic Data. To show the effectiveness of TDR, we first apply it to the MNIST dataset. We generate two different datasets, one with only two classes and the other using all ten classes in the dataset. For the first experiment, we want to create an imbalanced dataset of two digits. To better examine the framework, we choose the two most confused digits based on the MNIST classification confusion matrix, which are '4' and '9'. For this experiment the class 4 is the minority class and 9 is the majority class. If the total number of samples in training is N, the size of minority class and that of majority class are $\frac{\rho N}{\rho+1}$ and $\frac{N}{\rho+1}$, respectively. We use the LeNet [36] model as the base model with cross entropy over output logits as the loss function. This is a simple convolutional neural network with two convolution layers each with a max pooling at the output following with 3 fully connected layers. For the first experiment, we choose ratios from $\rho \in \{0.1, 0.05, 0.02, 0.005\}$ and generate different imbalanced datasets for $N = 5,000$. For instance, $\rho = 0.005$ means that for every 200 samples from the larger class, only one sample is from the smaller class. The target distribution is a dataset with 50 data points balanced over different classes. The test dataset is a balanced set with $2,000$ samples. Figure 3 shows the performance of TDR and sample reweight on the aforementioned imbalanced dataset with ratio of $\rho = 0.005$, based on epochs (Figure 3(a)) and wall clock time (Figure 3(b)) of training. It can be inferred that the TDR outperforms example reweighting on accuracy with a substantially higher convergence speed.

In addition to accuracy on balanced test set, in classification of an imbalanced dataset, the crucial metric is the recall rate, which shows the performance of the classifier on the positive (smaller)

[2] http://yann.lecun.com/exdb/mnist/
[3] https://www.cs.toronto.edu/~kriz/cifar.html
[4] https://mesonet.agron.iastate.edu/archive/
[5] https://archive.ics.uci.edu/ml/datasets/Adult

class data points. Figure 4 shows the recall rate of the class 4 in training procedure, in which the superiority of the TDR is noticeable.



(a) (b)

Figure 4: Recall rate of the smaller class 4 during the training of the experiment in Figure 3 with imbalance ratio $\rho = 0.005$, based on epoch (a), and wall-clock time (b). Similar to test accuracy, TDR outperforms example reweight in recall rate.

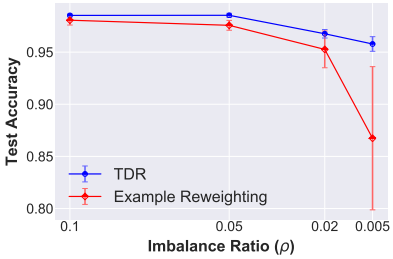
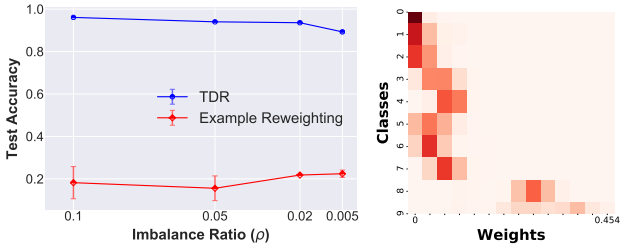


Figure 5: Comparing the test accuracy of TDR with example reweighting for different imbalance ratios from $\rho \in \{0.1, 0.05, 0.02, 0.005\}$. The dataset is MNIST with two classes (4 and 9), with the same setting as in Figure 3. We repeat each experiment 5 times to examine the robustness of algorithms.

Figure 5 presents a comparison of these two frameworks on different imbalance ratios. To compare reliably, we repeat the training procedure for 5 times and report the average and standard deviation of the test accuracy. TDR not only boosts the performance, but also is more stable compared to example reweighting. This is the result of incremental class weight updates in TDR, compared to perturbing each sample's weight on each iteration in the example reweighting.

As for the second experiment, we use the same model but create a long-tailed dataset. To generate a long-tailed dataset from MNIST, using the same scheme as [14], we use $n_i = n_0 \mu^i$, in order to decrease the class size exponentially, where n_i is the number of samples in class i. Hence, based on the definition, the imbalance ratio would be $\rho = \frac{n_9}{n_0} = \mu^9$. We use the same imbalance ratio as before from $\rho \in \{0.1, 0.05, 0.02, 0.005\}$, and test both TDR and sample reweight on these datasets. Figure 6(a) shows the result of this experiment on two frameworks. As it can be implied, example reweighting has an extremely poor performance on long-tailed dataset. This is due to perturbing weights for each sample at each iteration, which can introduce a large noise to the training. Figure 6(b) exhibits the distribution of learned weights for each class in the training process. This observation shows that, the optimal weights



(a) (b)

Figure 6: (a) The test accuracy of targeted data-driven regularization vs. example reweighting for different imbalance ratios $\rho \in \{0.1, 0.05, 0.02, 0.005\}$ with long-tailed MNIST dataset. Each experiment is repeated 5 times. (b) Distribution of learned weights for each class for $\rho = 0.02$. Each row shows the distribution of weights for each class during training. The size of each class is decreased exponentially, however, the learned weights are not exactly proportional to the inverse of their size.

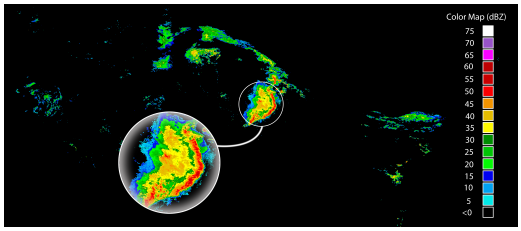


Figure 7: A NEXRAD level III radar image of the United States with a bow echo on May 24, 2008, 10:20 GMT. The bow echo part is being magnified.

for each class would not necessarily correlate with its inverse of frequency.

4.1.2 Severe Weather Prediction Dataset. In real-world applications, the primary challenge is often to detect critical incidents in datasets. Notwithstanding their importance in the classification tasks, normally those critical incidents are scarce in the dataset. Therefore, based on earlier discussions, a typical classifier would fail miserably in detecting these incidents.

One of the conspicuous examples is severe weather prediction using radar, satellite, and other sensor data. Severe weather conditions, such as tornadoes, thunderstorms, and straight-line winds, are occasional phenomena, but can be spotted in radar or satellite images with some specific patterns to provide early warnings. One of these patterns, associated with some severe weather conditions such as thunderstorms and straight-line winds, is called *Bow Echo*, because it has archer's bow shape pattern in radar images as it is depicted in Figure 7. Detecting and predicting the formation of bow echoes could help meteorologists predict related severe weather conditions and prevent their detrimental consequences [28–30].

We use NEXRAD level III radar data in order to create our dataset of radar images for the whole year of 2008 gathered from 160 high-resolution radars across the United States. We will test the model

on a balanced set of bow echo and non-bow echo samples from the year 2009. The year 2008 is chosen for training because of a high number of severe weather activities in that year. We use the reflectivity images of radar data, similar to Figure 7, which are 4-bit color map with spatial size of $2,600 \times 6,000$ pixels. The colors in these images, as indicated in Figure 7, show different amplitude of the reflected signal in dBZ from 0 to 75 dBZ. Bow echo patterns mostly revealed in areas with more than 50 dBZ, *i.e.*, in areas with orange or red colors in the images. These images are created from radar data every five minutes; hence, we have 288 images per day, which can lead to more than $105K$ images every year. Despite the enormous number of images each year, number of images with a bow echo sample on it is very low. For instance, in the year 2008, there are only $1,821$ images from 81 different instances that are labeled as bow echo samples. Therefore, this dataset, similar to other severe weather detection and prediction datasets, is greatly imbalanced with $\rho = 0.017$. The data distribution is immensely skewed toward normal data points, as it is the case for most related critical incident detection applications. Thus, we apply TDR framework on this dataset to overcome the imbalance problem in this case.

For this dataset, we apply TDR on ResNet20 [23] model, with image size of 52×180. The target distribution is a balanced dataset of both bow echo and non-bow echo samples from year 2008 with the size of $|\xi_v| = 273$ that has 137 bow echo samples. The balanced test set contains $3,524$ images from the year 2009, which has $1,762$ bow echo samples. In this experiment, we set $b = 50$, $b_v = 10$, $\eta_{\text{in}} = 0.001$, and $\eta_{\text{out}} = 0.2$. For comparison, we add the results for hard weighting classes with inverse of their frequency and also standard training without weights. The result of this training after 11 epochs in Figure 8 reveals that TDR has an exceptional capacity on addressing biases problem in this dataset, by achieving more than 86% in accuracy and 85% in bow echo recall rate.

4.2 CIFAR10 Dataset

Similar to the MNIST experiment in Figures 3 and 5, we run experiments on the CIFAR10 Dataset with having the same structure for creating imbalanced dataset of classes "4" and "9" with class "4" as the minority class in the dataset. We run the experiment for 4 different imbalance ratios from $\rho \in \{0.3, 0.2, 0.1, 0.05\}$. We also choose two different network architectures, ResNet20 [23] and a simple 4-layers ConvNet with 2 fully connected layers at the top for classification. In order to show the efficacy of the TDR framework, we compare with example reweighting [41], hard weighting with inverse of class frequency, random weighting, and without any weights. The results for the ConvNet on datasets with different imbalance ratio are depicted in Figure 9, where the superiority of TDR is noticeable. Example reweighting is better than random weights, but cannot even beat the learning without any weights, nor the hard weighting with inverse of frequency. The other major issue of example reweighting, is the high computational complexity, which prevents us from running it on larger networks like ResNet. The detailed results of this experiment on both models are in Table 1, in which TDR has the highest test accuracy on all the experiments.

In addition to two classes, we run a similar experiment to long-tailed MNIST on CIFAR10 data. In order to make the data long-tailed,

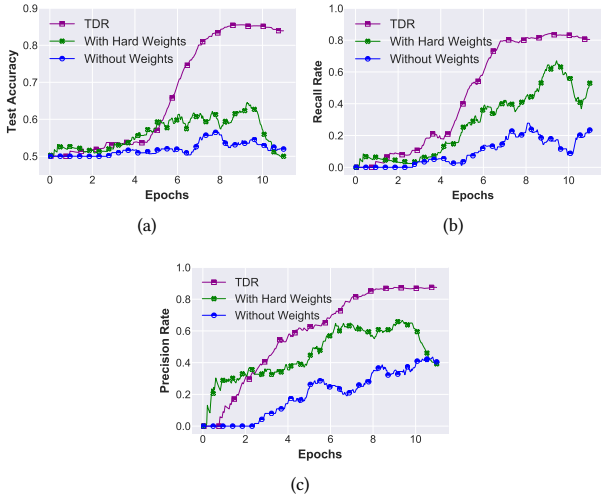


Figure 8: Accuracy, recall, and precision rates on balanced test dataset after 11 epochs of training for targeted-meta learning, hard reweighting with inverse of frequency, and without weights. The training dataset contains the complete radar images from year 2008 with $\rho = 0.017$, while the test set is a balanced dataset from the year 2009. Test accuracy and recall rate on bow echo samples reach to 0.8605 and 0.855, respectively. With hard weights the recall rate increases but at the cost of decreasing precision.

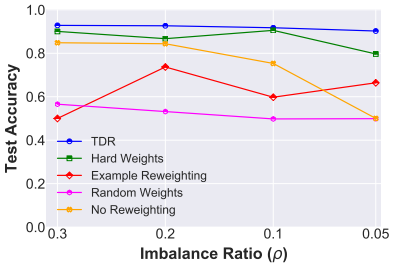


Figure 9: Comparing the test accuracy of targeted data-driven regularization with example reweighting, hard weighting with inverse of frequency, random weights and no reweighting. These experiments are run for different imbalance ratios from $\rho \in \{0.3, 0.2, 0.1, 0.05\}$. The dataset is CIFAR10 with two classes (4 and 9) and the model is a 4-layer ConvNet. We repeat each experiment 5 times and report the average results. The detailed results are in Table 1.

we decrease the size of classes with $n_i = n_0 \mu^i$ and generate two datasets with imbalance ratio of $\rho \in \{0.1, 0.2\}$. Then, we run our framework as well as example reweighting, hard weighting with inverse of class frequency, random weighting, and standard training, on two models of ResNet20 and the similar 4-layer ConvNet. Because of the aforementioned reasons, we are not able to run example reweighting on ResNet20 for comparison. The results of these experiments in Table 2 indicates the superiority of TDR for

	ResNet20				ConvNet			
Imbalance Ratio	0.3	0.2	0.1	0.05	0.3	0.2	0.1	0.05
TDR (Ours)	**0.9011 ±0.018**	**0.9222 ±0.005**	**0.8878 ±0.049**	**0.8756 ±0.014**	**0.9284 ±0.004**	**0.9266 ±0.002**	**0.9175 ±0.004**	**0.9025 ±0.003**
Example Reweighting [41]	N/A	N/A	N/A	N/A	0.4999 ±0.0	0.737 ±0.133	0.5977 ±0.135	0.6641 ±0.15
Hard Weighting	0.8545 ±0.091	0.8966 ±0.034	0.8498 ±0.067	0.7969 ±0.155	0.9007 ±0.019	0.8672 ±0.021	0.9055 ±0.01	0.8899 ±0.006
Random Weights	0.4998 ±0.002	0.5694 ±0.146	0.5005 ±0.002	0.5006 ±0.002	0.5656 ±0.147	0.5317 ±0.074	0.4978 ±0.002	0.499 ±0.001
No Reweighting	0.6564 ±0.159	0.5643 ±0.114	0.5438 ±0.108	0.4993 ±0.108	0.8485 ±0.144	0.8439 ±0.145	0.7532 ±0.166	0.5002 ±0.002

Table 1: Experiment results on imbalanced CIFAR10 dataset with two classes of "4" and "9", where "4" is the minority class. We use two different models, namely, ResNet20 and a simple 4-layer ConvNet with 2 fully connected layers as the classifier. We compare targeted data-driven regularization with example reweighting, hard weights with inverse of class frequency and no weights or standard training.

	ResNet20		ConvNet	
Imbalance Ratio	0.1	0.2	0.1	0.2
TDR (Ours)	**0.5602 ±0.017**	**0.5758 ±0.046**	0.2596 ±0.018	0.2897 ±0.018
Example Reweighting [41]	N/A	N/A	0.0998 ±0.0	0.1001 ±0.0
Hard Weighting	0.367 ±0.058	0.3336 ±0.077	0.0921 ±0.02	0.3072 ±0.02
Random Weights	0.0908 ±0.15	0.1062 ±0.076	0.0911 ±0.004	0.0961 ±0.004
No Reweighting	0.3628 ±0.08	0.3308 ±0.066	**0.4482 ±0.154**	**0.4739 ±0.154**

Table 2: Experimental results on long-tailed CIFAR10, with imbalance ratio from $\rho \in \{0.1, 0.2\}$ with two models, ResNet20 and ConvNet. We use $n_i = n_0 \mu^i$ to decrease the class size and make the dataset long-tail. Targeted data-driven regularization achieves the best accuracies in ResNet20, however the normal training is doing better in ConvNet, which is not a suitable model for training of this dataset.

long-tailed CIFAR10 on ResNet20. Although ConvNet is not a suitable model choice for this training, we include it in the results for better comparison, where standard training outperforms all other weighting schemes.

4.3 Fairness in Classification

Considering the massive improvement in accuracy of machine learning and computer vision models, concerns about fairness of these models are arising. The issue of fairness comes from the fact that data collection is not fair among different groups or categories. Hence, analogous to imbalanced datasets, the dataset is skewed more toward the majority group. For instance, in the Adult dataset, out of 30,162 training samples, 20,380 samples are for male participants, while only 9,782 samples belong to female participants. For showing the performance of our framework on Adult dataset, we simply use a 3-layer multilayer perceptron (MLP), with 120, 84, and 2 units, respectively, with ReLU activation on the first two layers. We test the learned model on a balanced test set of both groups. Table 3 shows that using TDR we can achieve better performance on balanced set of both groups.

	Balanced Test Accuracy
Normal Training	0.7511
TDR	**0.7906**

Table 3: The accuracy of normal training versus targeted data-driven regularization on the Adult dataset. The test set contains balanced number of samples for male and female instances. It shows that TDR can achieve higher and equal true positive rates among groups on the test dataset.

5 CONCLUSIONS & FUTURE WORK

Biases in datasets can pose a variety of problems in training a machine learning model for out-of-distribution generalization. In this work, we advocate the use of a small unbiased target dataset in the form of a bilevel programming as a data-driven regularization for the main training with biased datasets. Our proposed targeted data-driven regularization utilizes this target dataset to learn the weight of each designated class or category in the training process using the bilevel program. We empirically show the efficacy of this framework in dealing with various forms of biases in datasets.

This work leaves interesting directions as future work. First, we believe our proposed data-driven regularization has the potential to be used on other learning scenarios such as adversarial training, and that is worthy of investigation. Also, time series prediction tasks are mostly engaging with imbalanced datasets, hence it is valuable to extend this model to temporal data classification and prediction tasks. Finally, a non-asymptotic convergence analysis of the proposed bilevel stochastic optimization algorithm for non-convex objective functions on both levels can help to better analyze the framework from a theoretical perspective.

ACKNOWLEDGMENTS

We gratefully acknowledge the generous support of the Microsoft AI for Earth program for providing Azure services, and Nvidia equipment grants. This work also used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by the National Science Foundation under grant number ACI-1548562.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *Proc. of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 265–283.

[2] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. 2018. A reductions approach to fair classification. *arXiv preprint arXiv:1803.02453* (2018).

[3] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. 2016. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*. 3981–3989.

[4] Antreas Antoniou, Amos Storkey, and Harrison Edwards. 2017. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340* (2017).

[5] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. 2019. Invariant risk minimization. *arXiv preprint arXiv:1907.02893* (2019).

[6] Tamer Basar and Geert Jan Olsder. 1999. *Dynamic Noncooperative Game Theory*. Vol. 23. SIAM.

[7] Samy Bengio. 2015. Sharing representations for long tail computer vision problems. In *Proc. of the ACM International Conference on Multimodal Interaction*. ACM, 1–1.

[8] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proc. of the International Conference on Machine Learning*. ACM, 41–48.

[9] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. 2018. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks* 106 (2018), 249–259.

[10] Flavio Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R Varshney. 2017. Optimized pre-processing for discrimination prevention. In *Advances in Neural Information Processing Systems*. 3992–4001.

[11] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research* 16 (2002), 321–357.

[12] Benoît Colson, Patrice Marcotte, and Gilles Savard. 2007. An overview of bilevel optimization. *Annals of Operations Research* 153, 1 (2007), 235–256.

[13] Andrew Cotter, Heinrich Jiang, and Karthik Sridharan. 2019. Two-Player Games for Efficient Non-Convex Constrained Optimization. In *Algorithmic Learning Theory*. 300–332.

[14] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. 2019. Class-Balanced Loss Based on Effective Number of Samples. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9260–9269.

[15] Michele Donini, Luca Oneto, Shai Ben-David, John S Shawe-Taylor, and Massimiliano Pontil. 2018. Empirical risk minimization under fairness constraints. In *Advances in Neural Information Processing Systems*. 2796–2806.

[16] Charles Elkan. 2001. The foundations of cost-sensitive learning. In *Proc. of the International Joint Conference on Artificial Intelligence*, Vol. 17. Lawrence Erlbaum Associates Ltd, 973–978.

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

[18] Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimilano Pontil. 2018. Bilevel programming for hyperparameter optimization and meta-learning. *arXiv preprint arXiv:1806.04910* (2018).

[19] Saeed Ghadimi and Mengdi Wang. 2018. Approximation Methods for Bilevel Programming. *arXiv preprint arXiv:1802.02246* (2018).

[20] Gabriel Goh, Andrew Cotter, Maya Gupta, and Michael P Friedlander. 2016. Satisfying real-world goals with dataset constraints. In *Advances in Neural Information Processing Systems*. 2415–2423.

[21] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in Neural Information Processing Systems*. 8535–8545.

[22] Moritz Hardt, Eric Price, Nati Srebro, et al. 2016. Equality of opportunity in supervised learning. In *Advances in Neural Information Processing Systems*. 3315–3323.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[24] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. 2016. Learning deep representation for imbalanced classification. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*. 5375–5384.

[25] Simon Jenni and Paolo Favaro. 2018. Deep bilevel learning. In *Proc. of the European Conference on Computer Vision*. 618–633.

[26] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *Proc. of the International Conference on Machine Learning*. 2309–2318.

[27] Herman Kahn and Andy W Marshall. 1953. Methods of reducing sample size in Monte Carlo computations. *Journal of the Operations Research Society of America* 1, 5 (1953), 263–278.

[28] Mohammad Mahdi Kamani, Sadegh Farhang, Mehrdad Mahdavi, and James Z Wang. 2019. Targeted meta-learning for critical incident detection in weather data. In *Proc. of the International Conference on Machine Learning, Workshop on Climate Change: How Can AI Help*.

[29] Mohammad Mahdi Kamani, Farshid Farhat, Stephen Wistar, and James Z Wang. 2016. Shape matching using skeleton context for automated bow echo detection. In *Proc. of the IEEE International Conference on Big Data*. IEEE, 901–908.

[30] Mohammad Mahdi Kamani, Farshid Farhat, Stephen Wistar, and James Z Wang. 2018. Skeleton matching with applications in severe weather detection. *Applied Soft Computing* 70 (2018), 1154–1166.

[31] Mohammad Mahdi Kamani, Farzin Haddadpour, Rana Forsati, and Mehrdad Mahdavi. 2019. Efficient Fair Principal Component Analysis. *arXiv preprint arXiv:1911.04931* (2019).

[32] Faisal Kamiran and Toon Calders. 2012. Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems* 33, 1 (2012), 1–33.

[33] Salman H Khan, Munawar Hayat, Mohammed Bennamoun, Ferdous A Sohel, and Roberto Togneri. 2018. Cost-sensitive learning of deep feature representations from imbalanced data. *IEEE Transactions on Neural Networks and Learning Systems* 29, 8 (2018), 3573–3587.

[34] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *Proc. of the International Conference on Machine Learning-Volume 70*. JMLR. org, 1885–1894.

[35] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Remi Le Priol, and Aaron Courville. 2020. Out-of-distribution generalization via risk extrapolation (rex). *arXiv preprint arXiv:2003.00688* (2020).

[36] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86, 11 (1998), 2278–2324.

[37] Or Litany and Daniel Freedman. 2018. SOSELETO: A Unified Approach to Transfer Learning and Training with Noisy Labels. *arXiv preprint arXiv:1805.09622* (2018).

[38] Wanli Ouyang, Xiaogang Wang, Cong Zhang, and Xiaokang Yang. 2016. Factors in finetuning deep model for object detection with long-tail distribution. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*. 864–873.

[39] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *Proc. of the NIPS Autodiff Workshop: The Future of Gradient-based Machine Learning Software and Techniques*.

[40] Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. 2017. On fairness and calibration. In *Advances in Neural Information Processing Systems*. 5680–5689.

[41] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2018. Learning to Reweight Examples for Robust Deep Learning. In *Proc. of the International Conference on Machine Learning*. 4334–4343.

[42] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. 2018. Generalizing across domains via cross-gradient training. *arXiv preprint arXiv:1804.10745* (2018).

[43] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A Efros, and Moritz Hardt. 2019. Test-time training for out-of-distribution generalization. *arXiv preprint arXiv:1909.13231* (2019).

[44] Kai Ming Ting. 2000. A comparative study of cost-sensitive boosting algorithms. In *Proc. of the International Conference on Machine Learning*. Citeseer.

[45] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. 2017. Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *Proc. of the International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1171–1180.

A TIME COMPLEXITY COMPARISON

In this Appendix, we compare the computational complexity of other approaches that use a data-driven regularization scheme, closely related to our framework. Also, we show that they can be considered as a special case of our proposed TDR framework. These frameworks are example reweighting [41], SOSELETO [37], Mentornet [26], and Deep Bilevel learning [25]. In all of these frameworks, except for deep bilevel learning, the main idea is to have different weights in the cost function for different samples, reflecting samples' importance in each of the aforementioned tasks. Hence, the computational complexity of these approaches is at least t_{in} times the computational complexity of the TDR. Deep bilevel leaning is similar to TDR, except for their approximation for inner level, which makes it faster at the price of degradation in accuracy.

Example reweighting [41] uses the perturbation idea [34], where they investigate the effect of input perturbation on the output of a network. They try to address the problem of an imbalanced dataset using a balanced validation set, by perturbing the weight of each sample from zero. The main difference between example reweighting and TDR is that they perturb each samples' weight from zero using the gradient direction, rather than learning the weights with gradient descent approach. Hence, each time the network encounters a specific example, without considering its weight in last epoch, they assign a new weight to it using perturbation. In addition, due to sample-based nature of the algorithm, as indicated in the paper, the computational complexity of perturbing weights is linear in the number of samples, while in TDR it is linear in number of classes, which is a significant improvement. In practice, we showed that TDR outperforms example reweighting in both speed and accuracy. Their approach can be rewritten as a bilevel programming similar to Algorithm 1, where t_{in} is 1, which means every level updates only once before going to another level. This is equal to solving the bilevel programming as a Lagrangian form and finding the solution for primal and dual variables in an iterative fashion. Because of this high computational cost, applying this method to rather large networks such as ResNet20 is not feasible, compared to TDR.

SOSELETO introduces a framework for transfer learning using a target dataset [37]. In this framework, they consider two models for source and target datasets, and for each model, parameters considered as a union of two parts, namely, feature part and classification part. In both target and source models, the feature part of the parameters is the same. They optimize the classification part's parameters for the target model, using a weighted loss on the source data. Hence, learning weights for source data points and feature's part parameters for target model at the same time can be written as a bilevel program. If we combine the two parameter sets together and have a single parameter set, the framework is exactly similar to example reweighting framework with its aforementioned issues.

Mentornet classifies a dataset with noisy labels using the curriculum learning approach [8]. They train another network, with features extracted from the main network as its input, to learn the weight of each sample in the training operation. Similar to example reweighting, the objective of this framework can be written as a Lagrangian problem. It is worth noting that in this problem, the parameters of the main network or student net are not involved directly, but they are affected by changing the weights for each sample calculated from the simple network of $g(\cdot)$. The main difference between this framework and the previous ones is that the weights of each samples are output of a network, which should be trained (based on the optimal weights, 1 if the main label is correct and 0 otherwise). However, in previous ones we consider the samples' weights as a hyperparameter. Based on the optimization problem in Eq. (3), this problem can be rewritten as a bilevel problem with equal inner and outer functions.

While deep bilevel by [25] is closely related to our idea in order to cast the problem as a bilevel optimization, the idea of bilevel programming for hyperparameter tuning is not new (*e.g.* [18]). Further, their idea of having different weights for different batches and trying to learn those weights based on gradients on the validation set is more related to [41]. Despite the relatedness among these papers, our idea is different in three key aspects as follows:

- The main idea of TDR is to employ a target dataset that is free of those mentioned biases as a surrogate to not well-defined objectives in the training (when the training and test distributions are different). In deep bilevel, they randomly choose both training and validation datasets, hence, their framework is not designed to address problems such as imbalanced datasets. Further, their validation dataset is changing in each epoch and is relatively big in size, which is more like a cross-validation. However, in TDR the target dataset is very small ($\frac{|\mathcal{V}|}{|\mathcal{T}|} \leq 0.0002$), it could be part of the training itself, and is not changing during the training, which indicates the capability of TDR.
- The optimization approaches are completely different. They approximate the inner level with Taylor expansion, which makes it a quadratic function to reduce the computational complexity of the bilevel problem by avoiding the second order derivatives. We are approximating the bilevel programming with a stochastic version of it, in which the computational complexity depends on t_{in}.
- Their experiments are mainly focused on noise in data and labels, like other frameworks mentioned in our paper, such as [26, 41]. In most of these methods, the key property is that noisy data usually have higher loss, hence they should be ignored or weighted down to avoid their dominance in the training. Our framework is more general because we are not bounded to a specific problem and can adapt to different problems by adapting the target dataset, like using a small clean dataset as the target dataset for noisy training data.