Rethinking Cost-Sensitive Classification in Deep Learning via Adversarial Data Augmentation

Qiyuan Chen; , Raed Al Kontar; , Maher Nouiehed; , X. Jessie Yang; , Corey Lester

https://pubsonline.informs.org/journal/ijds

INFORMS JOURNAL ON DATA SCIENCE
Vol. 4, No. 1, January–March 2025, pp. 1–19
ISSN 2694-4022 (print), ISSN 2694-4030 (online)

Rethinking Cost-Sensitive Classification in Deep Learning via Adversarial Data Augmentation

Qiyuan Chen,[a] **Raed Al Kontar,**[a,*] **Maher Nouiehed,**[b] **X. Jessie Yang,**[a] **Corey Lester**[c]

[a] Industrial & Operation Engineering, University of Michigan, Ann Arbor, Michigan 48109; [b] Industrial Engineering, American University of Beirut, Beirut 1107 2020, Lebanon; [c] College of Pharmacy, University of Michigan, Ann Arbor, Michigan 48109

*Corresponding author

Contact: cqiyuan@umich.edu, https://orcid.org/0009-0006-4112-014X (QC); alkontar@umich.edu, https://orcid.org/0000-0002-4546-324X (RAK); mn102@aub.edu.lb, https://orcid.org/0000-0001-8089-7011 (MN); xijyang@umich.edu, https://orcid.org/0000-0001-6071-0387 (XJY); lesterca@med.umich.edu, https://orcid.org/0000-0001-8774-793X (CL)

Received: December 5, 2022
Revised: June 20, 2023; November 5, 2023
Accepted: January 9, 2024
Published Online in Articles in Advance: February 28, 2024

https://doi.org/10.1287/ijds.2022.0033

Copyright: © 2024 INFORMS

Abstract. Cost-sensitive classification is critical in applications where misclassification errors widely vary in cost. However, overparameterization poses fundamental challenges to the cost-sensitive modeling of deep neural networks (DNNs). The ability of a DNN to fully interpolate a training data set can render a DNN, evaluated purely on the training set, ineffective in distinguishing a cost-sensitive solution from its overall accuracy maximization counterpart. This necessitates rethinking cost-sensitive classification in DNNs. To address this challenge, this paper proposes a cost-sensitive adversarial data augmentation (CSADA) framework to make overparameterized models cost sensitive. The overarching idea is to generate targeted adversarial examples that push the decision boundary in cost-aware directions. These targeted adversarial samples are generated by maximizing the probability of critical misclassifications and used to train a model with more conservative decisions on costly pairs. Experiments on well-known data sets and a pharmacy medication image (PMI) data set, made publicly available, show that our method can effectively minimize the overall cost and reduce critical errors while achieving comparable overall accuracy.

History: Nick Street served as the senior editor for this article.
Funding: Research reported in this publication was supported by the National Library of medicine of the National Institutes of Health in the United States under award number R01LM013624.
Data Ethics & Reproducibility Note: This paper abides by data ethics requirements. Data used are publicly available online at https://deepblue.lib.umich.edu/data/concern/data_sets/6d56zw997. Codes to replicate the results of this paper are available on Code Ocean at https://doi.org/10.24433/CO.2139841.v1 and in the e-Companion to this article (available at https://doi.org/10.1287/ijds.2022.0033).

Keywords: cost-sensitive learning • adversarial data augmentation • deep neural networks • multiclass classification • overparametrization

1. Introduction

Multiclass classification is one of the most fundamental tasks in modern statistics and machine learning. It has seen many success stories across a wide variety of real-life applications such as computer vision (Dosovitskiy et al. 2020, Liu et al. 2021), natural language processing (Devlin et al. 2018, Brown et al. 2020), anomaly detection (Sun et al. 2022), and fairness (Yue et al. 2021). Typical classification problems treat the cost of misclassifications equally and aim to maximize the overall accuracy in expectation across all classes and data points. However, in many practical settings, some misclassifications can be far more costly and critical compared with others - often accompanied by real-life safety implications. In the presence of such costs, models should be carefully designed to prevent critical errors. To contextualize this, consider the following real-life situation on medication errors that motivates this paper.

Medication errors occur when patients take medications that differ in ingredient, strength, or dosage (e.g., oral tablet, oral capsule) from the medication prescribed for dispensing. According to the Department of Health and Human Services, these errors result in around 3 million outpatient medical appointments, 1 million emergency department visits, and 125,000 hospital admissions each year (Health Services Department 2019). In general, such errors can expose the patient to dangerous side effects or result in a patient going untreated for their medical condition. However, not all medication errors have the same consequences. For example, a patient accidentally receiving glipizide, a medication for lowering blood sugar levels, instead of trazodone, a medication for sleep, could lower blood sugar too much, which, if ignored, can cause seizures and result in brain damage. In fact, it is more dangerous for a patient to receive glipizide instead of trazodone than for a patient to receive trazodone instead of glipizide as the consequences of taking glipizide are greater compared with trazodone (Institute for Safe Medication

Practices (ISMP) 2021). In this paper, we propose a cost-aware classification model that relies on generating targeted adversarial examples to train a cost-sensitive model that avoids critical errors. We demonstrate the effectiveness of our proposed model by applying it to a -pharmacy medication image (PMI) that we made publicly available at https://deepblue.lib.umich.edu/data/concern/data_sets/6d56zw997.

The appearance of critical errors in multiclass classification is a challenge not unique to medication dispensing. Indeed, several papers have tried to address this challenge under the notion of cost-sensitive learning. Perhaps one of the earliest approaches dates back to Breiman et al. (1984) that considered cost-sensitive classification via decision trees. They propose a model that uses decision trees to estimate the expected cost for each class and selects the label with the least cost. Since then, a wide variety of methods have been proposed for cost-sensitive learning across a wide variety of applications. A snapshot of the current literature is given in Section 2.

Yet, modern machine learning poses a fundamental challenge to traditional cost-sensitive learning, specifically when it comes to overparameterized models such as deep neural networks (DNNs). More specifically, overparameterized DNNs can fully interpolate (or overfit) a training data set in practice. Indeed, many experiments (Zhang et al. 2016, 2021; Bartlett et al. 2020) have shown that DNNs can readily achieve zero cross-entropy training loss even on data sets that are completely unstructured random noise. Fortunately, despite overfitting and in contrast to conventional statistical understanding, DNNs have exhibited outstanding generalization performance on unseen data (Devlin et al. 2018, Brown et al. 2020). This phenomenon, known as benign overfitting, has shown the potential benefits of overparameterized modeling in recent years.

Despite their wide success in various applications, overparameterized DNNs face a fundamental challenge in cost-sensitive classification tasks. If a model is clairvoyant, that is, it can always reveal the underlying truth, then critical error costs will not affect the training as no misclassifications exist. This phenomenon motivates rethinking cost-sensitive classification in DNNs and reveals the need to go beyond training examples in cost-sensitive learning.

This paper tackles this exact problem. Specifically, we focus on general multiclass classification problems where costs are pairwise and possibly asymmetric. Under this setting, we propose a cost-sensitive adversarial data augmentation (CSADA) framework to render overparameterized models cost sensitive. The overarching idea is to generate targeted adversarial examples that push the decision boundary in a cost-aware direction. Such focused adversarial samples are data perturbations that maximize the probability of getting critical misclassifications and are used to generate more conservative decisions on costly pairs. Experiments on multiple data sets, including the PMI data set, show that our method can effectively minimize the overall cost and reduce critical errors. An interesting observation is that, although our main objective is to reduce cost, our method is still able to maintain accuracy comparable to non–cost-sensitive settings. This lies in sharp contrast to general adversarial deep learning where robustness to worst-case perturbations significantly affects clean accuracy (i.e., classification accuracy of the known classes). Also, although our main focus in this paper is DNNs, our approach can be directly incorporated within any classification approach to induce cost sensitivity.

1.1. Organization

The rest of the paper is organized as follows. We first start with a simple motivational example to fully contextualize the previous discussion. Then in Section 2, we provide a brief overview of relevant related work. In Section 3, we present our model and the training algorithm, followed by an illustrative proof-of-concept that sheds light on our model's intuition in Section 4. Experiments on various classification tasks and the PMI data set are then presented in Section 5. Finally, Section 6 concludes the paper with a brief discussion.

1.2. Simple Motivational Example

We first start with a simple example highlighting the challenges faced by cost-sensitive modeling in DNNs. Consider a classification problem where given observable data $\boldsymbol{x} \in \mathcal{X}$; we aim to predict the class label $y \in \mathcal{Y}$. Suppose the training data are $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1...N}$, where N denotes the total number of training points. Now, consider the setting where the costs of misclassifications are not equal among all the pairs. Let matrix $\mathcal{C}$ be the cost matrix with nonnegative entries $c(y, z)$ where for a given class $y \in \mathcal{Y}$, a cost $c(y, z)$ is incurred when a model predicts $z \in \mathcal{Y}$ for a data point with true label y. Note that $c(y, z)$ need not be equal to $c(z, y)$ (recall the medical example). Moreover, assume that a correct classification incurs zero cost so that all elements on the diagonal of $\mathcal{C}$ are zero.

In a typical classification task, we aim to learn a parameterized model that predicts the label of an input $\boldsymbol{x}$. This is typically done by estimating the probability that a data sample $\boldsymbol{x}$ returns a label $z \in \mathcal{Y}$, which we denote by $p_z(\boldsymbol{\theta}; \boldsymbol{x})$, and selecting the label with the highest probability.

In a conventional cost-insensitive setting, the goal is to maximize the probability of returning the true label:

$$\max_{\boldsymbol{\theta}} \ \mathbb{E}_{x,y}[p_y(\boldsymbol{\theta}; \boldsymbol{x})] \approx \frac{1}{N} \cdot \sum_{i=1}^{N} [p_{y_i}(\boldsymbol{\theta}; \boldsymbol{x}_i)], \qquad (1)$$

where $p_{y_i}(\boldsymbol{\theta}; \boldsymbol{x}_i)$ is the correct prediction probability. If we simply add a natural logarithm to the predicted probabilities, then (1) becomes the well-known cross-entropy loss that is the most common loss function used in DNN classification tasks.

Now, for cost-sensitive settings, our aim is to minimize the overall cost. A natural extension of (1) is given by

$$\begin{aligned}&\min_{\boldsymbol{\theta}} \ \mathbb{E}_{x,y}\left[\sum_{z\in\mathcal{Y}} c(y,z)\cdot p_z(\boldsymbol{\theta};\boldsymbol{x})\right]\\ &\approx \min_{\boldsymbol{\theta}} \frac{1}{N}\cdot\sum_{i=1}^{N}\left[\sum_{z\in\mathcal{Y}} c(y_i,z)\cdot p_z(\boldsymbol{\theta};\boldsymbol{x}_i)\right],\end{aligned} \qquad (2)$$

where $c(\cdot,\cdot)$ are nonnegative costs of misclassifications. If $c(y,z)=1$ for all $y, z\in\mathcal{Y}$, then (2) and (1) retrieve the same optimal solution set. Moreover, one can easily show that both Objectives (1) and (2) achieve the optimal solution if and only if $p_{y_i}(\boldsymbol{\theta};\boldsymbol{x}_i)=1$ or equivalently $p_z(\boldsymbol{\theta};\boldsymbol{x}_i)=0$ for all $\{z\neq y_i; z\in\mathcal{Y}\}$.

Hence, if the functional space is unconstrained or the model capacity is large enough to fully interpolate the training examples, solving (2) becomes equivalent to solving (1).

Overparameterization exists when the model has more parameters than what is needed to explain the data. For overparameterized DNNs, it is often the case that there exists a $\boldsymbol{\theta}$ such that $p_{y_i}(\boldsymbol{\theta};\boldsymbol{x}_i)\approx 1$ for all the training data points in $\mathcal{D}$ (Zhang et al. 2016, Arpit et al. 2017, Belkin et al. 2018). Therefore, the cost matrix $\mathcal{C}$ will not affect the models as no misclassifications occur, and the cost is always zero on the training data set. More specifically, if the model is able to perfectly fit training data, then any loss function evaluated purely on the training set becomes ineffective in distinguishing a cost-sensitive solution in (2) from its overall-accuracy maximization counterpart in (1). Although existing weight-scheming methods automatically fail in such settings, our proposed method pushes decision boundaries to make them cost-sensitive by generating perturbed samples prone to misclassification, which, when used in training, help reduce the total cost.

Next, we provide a numerical example highlighting the failure of cost-sensitive learning in DNNs. We consider the popular MNIST data set comprising of handwritten images of digits 0–9 (Deng 2012) trained on ResNet-34. Using a cross-entropy loss, we train a vanilla Baseline model. Details on experimental settings can be found in Section 5, where we revisit this experiment. For the Baseline model, at the end of the last epoch, the model reaches 100% accuracy on the training set. Figure 1(a) shows the pairwise error rate across the 10 classes.

Now starting with the Baseline model and for every possible pair (y',z') between the 10 classes, we train a cost-sensitive DNN where only predicting y' as z' incurs a cost of one while all the other error costs are set to zero. We then define an extreme case of cross-entropy loss where the sole goal is to prevent the model from making critical errors. Starting from the Baseline model, the objective is given as

$$\min_{\boldsymbol{\theta}} \frac{1}{N}\cdot\sum_{i=1}^{N}[-\mathbb{1}_{y_i=y'}\cdot\log(1-p_{z'}(\boldsymbol{\theta};\boldsymbol{x}_i))]. \qquad (3)$$

The purpose of this test is to observe whether this strong cost-sensitive penalty can bias an overfitted model to reduce critical errors. The results for all 90 experiments are shown in Figure 1(b). In the figure, each entry of the matrix corresponds to the pairwise error rate of the experiment done, with this pair being the critical one. Indeed, the results confirm the previous discussion. Even with an extremely biased objective aiming only to minimize the critical error rate, we see almost no improvement over the baseline. This again confirms that in overparameterized models, cost-sensitive training can be very challenging if we completely depend on a training set that can be perfectly fit. Further illustrations along this line can be seen in Section 5. Motivated by this phenomenon, our method uses adversarial data augmentation to distinguish cost-sensitive solutions in overparameterized settings. Our proposed method in Section 3 goes beyond the training set by generating adversarial examples that are specifically targeted to induce cost sensitivity.

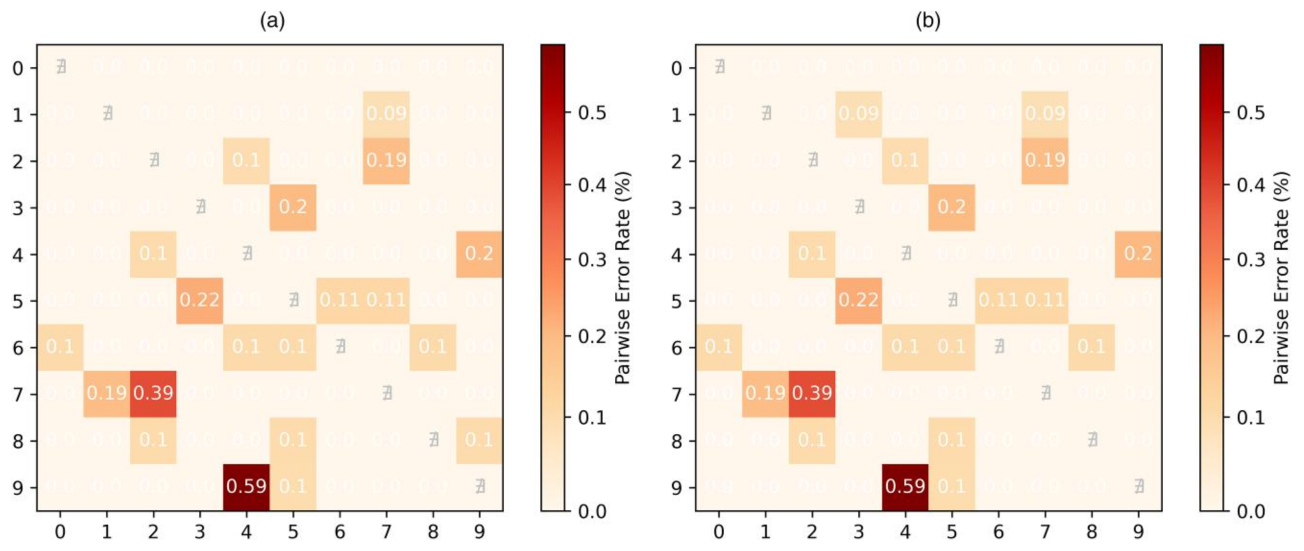
2. Literature Overview

2.1. Cost-Sensitive Learning

The setting where misclassification errors have different costs has been studied under the umbrella of cost-sensitive learning. Most existing research is on traditional models like logistic regression (Bahnsen et al. 2014), decision trees (Lomax and Vadera 2013), and support vector machines (Iranmehr et al. 2019). For neural networks (NNs), cost-sensitive NNs were first studied by Kukar and Kononenko (1998). In their work, a cost-weighted empirical risk function was introduced where misclassification probabilities are multiplied by their corresponding costs. Later on, Zhou and Liu (2005) extended this approach to NN with imbalanced data sets where methods such as oversampling and undersampling classes were introduced to induce cost sensitivity. The authors show that their methods were effective in shallow NN. Cost-sensitive deep neural network (CSDNN) (Chung et al. 2015) is one of the earliest works that develop effective cost-sensitive methods for DNNs by incorporating cost into the softmax layer. The performance of this approach was challenged by Chung et al. (2020) when the model is an overparameterized DNN.

In general, cost-sensitive literature, including the work mentioned previously, can be decomposed into three categories. The first category makes no change during the training phase but incorporates cost during inference/prediction. Mainly, after predictive probability vectors of all classes are attained, they are multiplied with a cost matrix to get the expected cost for each class.

Figure 1. (Color online) Similarity of Baseline and Traditional Cost-Sensitive Model



Notes. (a) Baseline model. (b) Cost-sensitive model by reweighting.

The class with the least cost is selected as the cost-sensitive prediction (Breiman et al. 1984, Domingos 1999, Zadrozny and Elkan 2001). These methods can be incorporated into any cost-sensitive training approach. However, they rely on the accuracy of the predicted probability vectors, which are usually overconfident in overparameterized NNs (Guo et al. 2017). The second category makes model-specific modifications (Tu and Lin 2010, Chung et al. 2015). For example, Tu and Lin (2010) modifies the SVM model to predict a cost vector instead of probabilities. Following a similar idea, Chung et al. (2015) proposes CSDNN and CNN$_{\text{SOSR}}$, which incorporate a regression layer to the NN to predict costs. One limitation of this category is that algorithms are often specialized and cannot be generalized to other models. For instance, CSDNN uses a special neural network that consists of fully connected layers and sigmoid activation. This limits its applications in other NN structures such as ResNet (He et al. 2016) or Transformers (Vaswani et al. 2017). The third category modifies the training data distribution. As aforementioned, the most widely adopted approach in this category is replacing the empirical risk with a cost-weighted empirical risk (Kukar and Kononenko 1998, Chan and Stolfo 1999, Zadrozny et al. 2003, Zhou and Liu 2005). However, as explained in Section 1.2, reweighting empirical risk is ineffective in overfitted models.

The method proposed in this paper is closest to the third category, but instead of reweighting data points, we incorporate targeted adversarial examples that can effectively move the decision boundaries of a trained DNN in cost-aware directions in the presence of overfitting. Being a data augmentation method, our framework can be applied to most DNN structures and models beyond NNs.

2.2. Adversarial Attacks and Adversarial Training

Although DNNs have outstanding performance on modern machine learning tasks, they are known for their fragility to adversarial attacks (Szegedy et al. 2013). One can adversarially add an imperceptible perturbation to the input data and significantly change outputs. Such perturbed data are called adversarial examples. Gradient-based attacks are commonly used in generating adversarial examples. Common methods include the fast gradient sign method (FGSM) (Goodfellow et al. 2014), DeepFool (Moosavi-Dezfooli et al. 2016), and projected gradient descent (PGD) (Kurakin et al. 2018). Interestingly, research has shown that overparameterization leads to higher sensitivity to adversarial attacks in linear models (Donhauser et al. 2021) and random feature regression (Hassani and Javanmard 2022). To address this issue, several methods use the generated adversarial examples in training to improve models' robustness (Volpi et al. 2018). Another interesting property observed by Cao and Gong (2017) is that adversarial examples are located near the decision boundaries of NNs. Based on this observation, Heo et al. (2019) uses adversarial examples to depict the decision boundaries of NNs and uses the adversarial examples as boundary-supporting samples for knowledge distillation.

Recent literature has seen work on cost-sensitive robustness against adversarial attacks (Zhang and Evans 2018, Shen et al. 2023). Cost-sensitive robustness in literature refers to surviving an adversarial attack (e.g., FGSM, PGD) on costly pairs. Unlike cost-sensitive robustness, our work does not aim at defending against adversarial attacks. However, we focus on using adversarial data augmentation to push the decision boundary of a pretrained model in cost-aware directions such that the model becomes cost-sensitive on *natural* examples.

2.3. Distinctions to Data Augmentation Schemes for Long Tail Problems

Adversarial data augmentation has also been used in the long tail problem in classification (or the imbalanced data set problem) (Kozerawski et al. 2020, Li et al. 2021, Park et al. 2022, Zhang et al. 2023). Long tail problems aim at correcting the bias introduced by imbalanced data sets, mainly by conducting data augmentation for the minority class. While adapting a data augmentation scheme, our method solves a different problem setting where costs are different between pairs regardless of whether the data set is balanced or not. In fact, many of our experiments were done on balanced data sets (CIFAR-10, MNIST, and the OCT data set in Appendix B), and our algorithm showed significant cost improvement compared with the Baseline. Furthermore, algorithms for long tail problems cannot be adapted to solve our problem as they augment for classes rather than pairs. When the cost matrix is asymmetric, it is unclear which class should be augmented.

3. Model Development

Our problem can be viewed through the lens of adversarial training of statistical and machine learning models. Unlike typical settings, our model's adversary aims to generate targeted adversarial examples that increase the chance of having critical errors. The defender, however, aims at finding a cost-aware robust model that avoids such errors. As such, our proposed model generates targeted adversarial examples to train a cost-aware robust model against critical errors.

Consider a classification model $f(\boldsymbol{\theta};\cdot)$ parameterized by $\boldsymbol{\theta}$, which predicts a class from $\mathcal{Y}$. Let $p_z(\boldsymbol{\theta};\boldsymbol{x}_i)$ be the probability of classifying point $\boldsymbol{x}_i$ with class $z\in\mathcal{Y}$. Usually, in an NN, p_z is an output of the soft-max layer, and $f(\boldsymbol{\theta};\cdot)$ simply outputs the class with the largest probability. For a given data point $(\boldsymbol{x}_i, y_i)$ and a specific class pair (y_i,z), we solve a maximization objective that finds a perturbation of the input data that maximizes the zth class predicted probability as follows:

$$\boldsymbol{\delta}_i^z = \underset{\|\boldsymbol{\delta}\|\le\epsilon}{\arg\max}\ p_z(\boldsymbol{\theta};\boldsymbol{x}_i+\boldsymbol{\delta}), \tag{4}$$

where $\|\cdot\|$ is a user-defined norm function, and ϵ is the threshold that limits the magnitude of the perturbation. This is repeated for each data point (x_i, y_i) and for each $z\in\mathcal{Y}\setminus y_i$. The purpose of this maximization problem is to find targeted attacks on a sample input data x_i aiming for directions that of critical errors. More specifically, for a given data sample, $\boldsymbol{x}_i$, we find a data perturbation $\boldsymbol{\delta}_i^z$ that maximizes the probability of classifying the data point to class z. Our overreaching goal is to find data samples that fool the model to make costly errors. These samples are then used in training to create cost-aware decision boundaries that reduce critical misclassification errors. Formally, we propose the following penalized cost-aware bilevel optimization formulation of the problem:

$$\begin{aligned}
&\min_{\boldsymbol{\theta}} \quad \ell_{\text{augmented}}(\boldsymbol{\theta};\boldsymbol{x},y,\boldsymbol{\delta})\\
&=\min_{\boldsymbol{\theta}} \quad \frac{1}{N}\cdot\sum_{i=1}^{N}\Bigg[\ell(p(\boldsymbol{\theta};\boldsymbol{x}_i),y_i)\\
&\qquad\qquad +\lambda\cdot\sum_{z\in\mathcal{Y}}\tilde{c}(y_i,z)\ell(p(\boldsymbol{\theta};\boldsymbol{x}_i+\boldsymbol{\delta}_i^z),y_i)\Bigg]\\
&\text{s.t.} \quad \boldsymbol{\delta}_i^z = \underset{\|\delta_i^z\|\le\epsilon}{\arg\max}\ p_z(\boldsymbol{\theta};\boldsymbol{x}_i+\boldsymbol{\delta}_i^z),
\end{aligned} \tag{5}$$

where $\ell(\cdot,\cdot)$ is the *cross-entropy loss*, $p(\boldsymbol{\theta};\boldsymbol{x}_i)$ is the vector that collects the predicted probabilities $p_y(\boldsymbol{\theta};\boldsymbol{x}_i)$ for all $y\in\mathcal{Y}$, and $\boldsymbol{\theta}$ is a vector of model parameters. Our formulation penalizes the objective with a cost-aware weighted sum of critical misclassification errors. The first term of our objective is a typical empirical risk objective that measures the sum of losses between the true labels $\{y_i\}_{i=1}^N$ and the corresponding model outputs $\{p(\boldsymbol{\theta};\boldsymbol{x}_i)\}_{i=1}^N$. The second term, however, is a penalty term that penalizes misclassifications of augmented examples according to their corresponding weights. In particular, the second term is the weighted sum of the losses incurred by the examples generated by the maximization problems. The weights $\tilde{c}(y_i,z)$ are defined by $c(y_i,z)^\tau/\sum_{y,z\in|\mathcal{Y}|}c(y,z)^\tau$, where τ is a temperature value that controls the emphasis on critical costs. A higher temperature will help filter out smaller costs when there are many classes. The higher the cost of a misclassification error, the higher the penalty coefficient of that term. Moreover, λ is a hyper-parameter that regulates the tradeoff between the first term that corresponds to the regular empirical loss and the penalty term, which penalizes critical misclassification errors for perturbed data. To summarize, our maximization problem generates $|\mathcal{Y}|$ adversarial perturbations for each data point targeting all classes $z\in\mathcal{Y}$. These examples are further used in the minimization problem for finding optimal model parameters for a cost-sensitive penalized empirical risk formulation that penalizes critical errors.

3.1. Relation to Adversarial Training

Typical adversarial training problems are formulated as min-max optimization problems that search for optimal model parameters when using worst-case data perturbations. In particular, the adversary in typical adversarial training aims at finding adversarial examples that maximize the loss objective, that is, data perturbations that fool the model the most. The defender, however, aims at finding a robust model that minimizes the loss when using the perturbed data.

Unlike the typical setting, our model generates and exploits targeted adversarial examples that increase the chance of having critical errors. This major difference results in distinct objectives for the max and min optimization problems. We next show that, in binary classification settings, the bilevel optimization problem in (5) can be formulated as a min-max non–zero-sum game.

Theorem 1. *Consider the objective defined in (5) with binary labels $\{y, \overline{y} = 1 - y\}$ and $\ell(\cdot)$ being the cross-entropy loss. Then solving (5) is equivalent to solving the following min-max problem:*

$$\begin{aligned}&\min_{\boldsymbol{\theta}} \max_{\|\boldsymbol{\delta}\| \le \epsilon} \ell_{augmented}(\boldsymbol{\theta}; \boldsymbol{x}, y, \boldsymbol{\delta})\\ &= \min_{\boldsymbol{\theta}} \max_{\|\boldsymbol{\delta}\| \le \epsilon} \frac{1}{N} \sum_{i=1}^{N} [\ell(p(\boldsymbol{\theta}; \boldsymbol{x}_i), y_i)\\ &\quad + \lambda \tilde{c}(y_i, \overline{y}_i) \ell(p(\boldsymbol{\theta}; \boldsymbol{x}_i + \boldsymbol{\delta}_i), y_i)], \end{aligned} \tag{6}$$

where $\boldsymbol{\delta} = \boldsymbol{\delta}_i$ for all $i \in \{1, \ldots, N\}$.

Proof. Because the functions are disjoint, we first obtain

$$\begin{aligned}&\max_{\|\boldsymbol{\delta}\| \le \epsilon} \sum_{i=1}^{N} \tilde{c}(y_i, \overline{y}_i) \ell(p(\boldsymbol{\theta}; \boldsymbol{x}_i + \boldsymbol{\delta}_i), y_i)\\ &= \sum_{i=1}^{N} \tilde{c}(y_i, \overline{y}_i) \max_{\|\boldsymbol{\delta}\| \le \epsilon} \ell(p(\boldsymbol{\theta}; \boldsymbol{x}_i + \boldsymbol{\delta}_i), y_i).\end{aligned}$$

To complete the proof, it suffices to show that

$$\arg\max_{\|\boldsymbol{\delta}_i\| \le \epsilon} \ell(p(\boldsymbol{\theta}; \boldsymbol{x}_i + \boldsymbol{\delta}_i), y_i) = \arg\max_{\|\boldsymbol{\delta}_i\| \le \epsilon} p_{\overline{y}_i}(\boldsymbol{\theta}; \boldsymbol{x}_i + \boldsymbol{\delta}_i).$$

In binary classification settings, the softmax layer is simply a sigmoid function, which we denote by $\sigma(\cdot)$. Let $r_{y_i}(\boldsymbol{\theta}; \boldsymbol{x}_i + \boldsymbol{\delta}_i)$ be the latent representation corresponding to class y_i of the data point $\boldsymbol{x}_i + \boldsymbol{\delta}_i$ before the sigmoid function. With $\ell(\cdot, \cdot)$ being the cross-entropy loss, we obtain

$$\begin{aligned}\ell(p(\boldsymbol{\theta}; \boldsymbol{x}_i + \boldsymbol{\delta}_i), y_i) &= -\log \sigma(r_{y_i}(\boldsymbol{\theta}; \boldsymbol{x}_i + \boldsymbol{\delta}_i))\\ &= -\log(1 - \sigma(-r_{y_i}(\boldsymbol{\theta}; \boldsymbol{x}_i + \boldsymbol{\delta}_i)))\\ &= -\log(1 - \sigma(r_{\overline{y}_i}(\boldsymbol{\theta}; \boldsymbol{x}_i + \boldsymbol{\delta}_i))),\end{aligned}$$

where the last equality holds due to the binary setting assumption. Knowing that $-\log(\cdot)$ is a monotonically decreasing function, we get

$$\begin{aligned}&\arg\max_{\|\boldsymbol{\delta}_i\| \le \epsilon} \ell(p(\boldsymbol{\theta}; \boldsymbol{x}_i + \boldsymbol{\delta}_i), y_i) = \arg\min_{\|\boldsymbol{\delta}_i\| \le \epsilon} [1 - \sigma(r_{\overline{y}_i}(\boldsymbol{\theta}; \boldsymbol{x}_i + \boldsymbol{\delta}_i))]\\ &= \arg\max_{\|\boldsymbol{\delta}_i\| \le \epsilon} \sigma(r_{\overline{y}_i}(\boldsymbol{\theta}; \boldsymbol{x}_i + \boldsymbol{\delta}_i)) = \arg\max_{\|\boldsymbol{\delta}_i\| \le \epsilon} p_{\overline{y}_i}(\boldsymbol{\theta}; \boldsymbol{x}_i + \boldsymbol{\delta}_i),\end{aligned}$$

which completes the proof. □

Remark 1. In the binary case, maximizing the probability of the untrue label is equivalent to minimizing the probability of the true label, which translates to maximizing the cross-entropy loss. The last implication results from the monotone nature of the log function. This relation allows us to formulate our problem as a min-max non–zero-sum game. The formulation presented in Theorem 1 can be seen as a cost-aware adversarial objective that further penalizes errors that cost more. This model can be applied in settings where false positive and false negative errors have different costs (e.g., medication dispensing example in Section 1).

The result presented in Theorem 1 fails to hold in nonbinary classification settings. A simple counterexample is presented for a three-label classification problem. Consider a data point (x, y) for which

$$r_y(\boldsymbol{\theta}; (x + \delta)) = 3\delta, \quad r_z(\boldsymbol{\theta}; (x + \delta)) = 2\delta, \quad \text{and} \quad r_3(\boldsymbol{\theta}; (x + \delta)) = -10\delta,$$

where $r_3(\boldsymbol{\theta}; (x + \delta))$ is the latent representation for the third class. Then,

$$\arg\max_{\|\delta\| \le \epsilon} \ell(p(\boldsymbol{\theta}; x + \delta), y) = \arg\max_{\|\delta\| \le \epsilon} -\log \sigma(r_y(\boldsymbol{\theta}; (x + \delta)))$$

is maximized by setting $\delta = -\epsilon$. However,

$$\arg\max_{\|\delta\| \le \epsilon} p_z(\boldsymbol{\theta}; x + \delta) > 0.$$

This counterexample shows that generating targeted examples according to the maximization problem in (5) is not equivalent to maximizing the loss function as used in typical adversarial training. This is mainly due to the nature of our problem, which aims at generating targeted adversarial examples instead of worst-case perturbations. Despite having distinct objectives, our formulation can be seen through the lens of non–zero-sum games. In the next section, we discuss the algorithm proposed to solve the objective presented in (5).

3.2. Multistep Gradient Descent Ascent with Rejection

Recent years have witnessed extensive research for solving min-max problems arising from adversarial settings. One of the first algorithms developed for adversarial training is the FGSM proposed by Goodfellow et al. (2014). This algorithm can be seen as a single-step method that constructs an adversarial example by solving a linearized approximation of the inner maximization problem. A multistep variant of FGSM, denoted by K-PGD, that applies K gradient ascent steps for solving the maximization problem was proposed by Madry et al. (2018). Several variants of this multistep ascent descent algorithm have been widely deployed in saddle point formulations. Despite having no convergence guarantees in nonconvex settings, such approaches have shown wide empirical success. In fact, no algorithm is known to converge in general nonconvex nonconcave optimization problems. In this paper, we use a variant of the multistep ascent descent method for solving the problem defined in (5). More specifically, our algorithm performs multiple ascent steps to generate targeted perturbed adversarial examples and then performs a gradient descent step on the cost-aware penalized objective.

Our overreaching goal is to induce critical errors in training by generating data samples that are close to the decision boundary between the corresponding labels. As illustrated in Section 4, having data points near the boundary is particularly important in overfitted models. Motivated by this idea, we propose an algorithm that performs multiple gradient ascent steps to generate perturbations of data samples that maximize the probability score of a given target label. To avoid undesirable perturbations, we impose a rejection mechanism that rejects generated samples with a label different from the true/targeted labels and stops whenever the targeted label is predicted. The latter is mainly justified since the gradient ascent steps of such a perturbation must have pushed the point close to the decision boundary between y and z.

Algorithm 1 (Clipped Gradient Ascent with Rejection)

```
1:  Input: Data point $\{x, y\}$, direction $z$, and weights $\boldsymbol{\theta}$
2:  Initialize $\boldsymbol{\delta}^{(0)} \leftarrow 0$
3:  if preattack prediction $f(x, \boldsymbol{\theta})$ is not $y$ then
4:      Output: $\boldsymbol{\delta}^{(0)}$
5:  end if
6:  for $k = 1, \ldots, K$ do
7:      $\Delta^{(k)} = \eta_2 \cdot \nabla_{\boldsymbol{\delta}} \log[p_z(\boldsymbol{\theta}; x + \boldsymbol{\delta}^{(k-1)})]$
8:      $\boldsymbol{\delta}^{(k)} \leftarrow \text{Clip}_{\|\cdot\| \le \epsilon}[\boldsymbol{\delta}^{(k-1)} + \boldsymbol{\Delta}^{(k)}]$
9:      if postattack prediction $f(x + \boldsymbol{\delta}^{(k)}, \boldsymbol{\theta}) \notin \{y, z\}$ then
10:         Output: $\boldsymbol{\delta}^{(k-1)}$
11:     else if postattack prediction $f(x + \boldsymbol{\delta}^{(k)}, \boldsymbol{\theta})$ is $z$ then
12:         Output: $\boldsymbol{\delta}^{(k)}$
13:     end if
14: end for
15: Output: $\boldsymbol{\delta}^{(K)}$
```

Our algorithm for generating directed adversarial examples is detailed in Algorithm 1. Our proposed algorithm takes as input a training data point $\{x, y\}$, model parameters $\boldsymbol{\theta}$, and a targeted output label z. The algorithm first checks whether the true label y is returned by the model (in Algorithm 1, line 3). If the true label is not returned, the point is considered unreliable, and no further updates are performed. Otherwise, our algorithm iteratively performs clipped gradient ascent steps to perturb in directions that increase the probability of labeling the point as z. This guarantees pushing the point closer to the decision boundary between the true label y and the label z. Specifically, at each iteration k, we first find an ascent direction by computing the gradient of the log probability score (i.e., $\nabla_{\boldsymbol{\delta}} \log[p_z(\boldsymbol{\theta}; x + \boldsymbol{\delta}^{(k-1)})]$) for class z (see Algorithm 1, line 7). Then, to bound the norm of the perturbation, we clip the ascent direction such that the norm of the perturbation $\boldsymbol{\delta}^{(k)}$ is less than ϵ (see Algorithm 1, line 8). In our experiments, we adopted the ℓ_∞ norm and used the clipping scheme described by Kurakin et al. (2018) such that

$$\boldsymbol{\delta}^{(k)} = \text{Clip}_{\|\cdot\| \le \epsilon}[\boldsymbol{\delta}^{(k-1)} + \boldsymbol{\Delta}^{(k)}]$$
$$:= \min\{\mathbf{1}\epsilon, \max\{-\mathbf{1}\epsilon, \boldsymbol{\delta}^{(k-1)} + \boldsymbol{\Delta}^{(k)}\}\},$$

where the minimum and maximum are both elementwise. In general, one can use any gradient clipping scheme that works for the specified norm function. Finally, we check the model's prediction on perturbed data $x + \boldsymbol{\delta}^{(k)}$. If the model predicts the target class z, then it stops and outputs the perturbation. If the model still predicts y, it proceeds to the next iteration. Otherwise, if the prediction is neither z nor y, the previous step $\boldsymbol{\delta}^{(k-1)}$ is returned.

Now that we have Algorithm 1 that generates directed adversarial examples, we present Algorithm 2 to induce cost sensitivity. Because our proposed approach is designed to push the decision boundary of an existing pretrained baseline model in cost-aware directions, we first train the baseline model by solving the optimization problem with $\lambda = 0$. Thereafter, we generate adversarial examples for training data points and every possible attack direction (i.e., every other class except the true class) using Algorithm 2. The generated adversarial examples are fed along with the original data to perform a gradient descent step on the penalized cost-aware objective. The complete algorithm is detailed in Algorithm 2. In the next section, we describe our stochastic version of the proposed algorithm.

Algorithm 2 (Multistep Gradient Decent-Ascent)

```
1:  Input: Natural data set $\mathcal{D}$, cost matrix $\mathcal{C}$
2:  Output: Trained weights $\boldsymbol{\theta}^{(T)}$
3:  Initialize $\boldsymbol{\theta}^{(0)}$ as pretrained weights on $\ell(p(\boldsymbol{\theta}; x), y)$
4:  for $t = 1, \ldots, T$ do
5:      for $i = 1, \ldots, N$ do
6:          for $z \in \mathcal{Y} \setminus \{y_i\}$ do
7:              $\boldsymbol{\delta}_i^z \leftarrow$ Algorithm 1$(x_i, y_i, z, \boldsymbol{\theta}^{(t-1)})$
8:          end for
9:      end for
10:     $\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)} - \eta_1 \cdot \nabla_{\boldsymbol{\theta}} \ell_{\text{augmented}}(\boldsymbol{\theta}^{(t-1)}; x, y, \boldsymbol{\delta})$
11: end for
```

3.3. Stochastic Multiascent Descent with Rejection

One important challenge for our proposed algorithm is the computational complexity incurred by generating $(|\mathcal{Y}| - 1)$ adversarial examples for each data sample. This becomes computationally intractable when the number of data samples or the number of classes is large. In this section, we propose a stochastic version of the algorithm that samples at each iteration a batch of data and *one* critical pair, which are then used to compute an unbiased estimate of the gradient of the main objective. More specifically, at each iteration t, our proposed method randomly samples a batch from the training data, denoted by $\mathcal{B}_t$, and samples one critical pair from a categorical distribution. We define our critical pair sampling distribution according to the cost of the pairs such that the probability of selecting one specific pair $(y_{\mathcal{B}_t}, z_{\mathcal{B}_t}) \in (\mathcal{Y}, \mathcal{Y})$ is equal to the normalized cost $\tilde{c}(y_{\mathcal{B}_t}, z_{\mathcal{B}_t})$. As such, stochasticity is induced by sampling a batch from the data and a single critical pair from all pairs. This reduces the call of Algorithm 1 in each iteration t from $(|\mathcal{Y}|-1)\cdot N$ to $|\mathcal{B}_t|$. For a given batch $\mathcal{B}_t$ and its corresponding critical pair selection $(y_{\mathcal{B}_t}, z_{\mathcal{B}_t})$, our stochastic objective can be defined as

$$\ell_{\text{stochastic}}(\boldsymbol{\theta}; \boldsymbol{x}, y, \boldsymbol{\delta}_{\mathcal{B}_t}) = \frac{1}{|\mathcal{B}_t|} \cdot \sum_{i \in \mathcal{B}_t} \left[\ell(p(\boldsymbol{\theta}; \boldsymbol{x}_i), y_i) + \lambda \cdot \ell(p(\boldsymbol{\theta}; \boldsymbol{x}_i + \boldsymbol{\delta}_i^{z_{\mathcal{B}_t}}), y_i)) \right],$$

where

$$\boldsymbol{\delta}_{\mathcal{B}_t}^{z_{\mathcal{B}_t}} = \begin{cases} \arg\max\limits_{\|\boldsymbol{\delta}\| \le \epsilon} p_{z_{\mathcal{B}_t}}(\boldsymbol{\theta}; \boldsymbol{x}_i + \boldsymbol{\delta}) & \text{if } y_i = y_{\mathcal{B}_t} \\ 0 & \text{otherwise.} \end{cases}$$

One can easily notice that the gradient of $\ell_{\text{stochastic}}$ is an unbiased estimator of the gradient of $\ell_{\text{augmented}}$ which is defined in (5). The detailed algorithm proposed for solving the stochastic version of our model is detailed in Algorithm 3.

Algorithm 3 (Multistep Stochastic Gradient Decent-Ascent)

```
1:  Input: Natural data set $\mathcal{D}$, cost matrix $\mathcal{C}$
2:  Output: Trained weights $\boldsymbol{\theta}^{(T)}$
3:  Initialize $\boldsymbol{\theta}^{(0)}$ as pretrained weights on $\ell(p(\boldsymbol{\theta}; \boldsymbol{x}), y)$
4:  for $t = 1, \ldots, T$ do
5:      Sample a mini-batch $\mathcal{B}_t \subseteq [N]$
6:      Sample one pair $(y_{\mathcal{B}_t}, z_{\mathcal{B}_t})$ with probability $\tilde{c}(y_{\mathcal{B}_t}, z_{\mathcal{B}_t})$
7:      for $i$ in $\mathcal{B}_t$ do
8:          if $y_i = y_{\mathcal{B}_t}$ then
9:              $\boldsymbol{\delta}_{\mathcal{B}_t}^{z_{\mathcal{B}_t}} \leftarrow$ Algorithm 1$(\boldsymbol{x}_i, y_i, z_{\mathcal{B}_t}, \boldsymbol{\theta}^{(t-1)})$
10:         end if
11:     end for
12:     $\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)} - \eta_1 \cdot \nabla_{\boldsymbol{\theta}} \ell_{\text{stochastic}}(\boldsymbol{\theta}^{(t-1)}; \boldsymbol{x}, y, \boldsymbol{\delta}_{\mathcal{B}_t})$
13: end for
```

4. Proof of Concept

A toy classification problem is presented in this section to demonstrate our model's ability to prevent critical errors. Three classes are generated from independent two-dimensional Gaussian distributions, namely $y_r := y_{\text{red}}$ (the top region), $y_b := y_{\text{blue}}$ (the bottom left region), and $y_g := y_{\text{green}}$ (the bottom right region) (Table 1; Figure 2(a)). We define the cost from y_g to y_b to be one and all others to be zero (Table 2) and set $\tau = 1$. As such, only misclassifying y_g with y_b incurs a cost. For each class, 50 points (shown in hollow circles) are sampled for training, and 500 points are sampled for testing (samples shown in solid translucent circles). For this classification task, we train a multilayer perception (MLP) with 50 nodes. We first optimize

Table 1. Toy Example Data Setting

Class	Mean	Covariance	Train samples	Test samples
y_r	$[0, 8]^\top$	$\begin{bmatrix} 2 & 0.5 \\ 0.5 & 2 \end{bmatrix}$	50	500
y_g	$[7, -6]^\top$		50	500
y_b	$[-7, -6]^\top$		50	500

the vanilla MLP without augmentation using gradient descent. At convergence, the decision boundaries are shown in Figure 2(a). One can directly realize that the model successfully reaches 100% accuracy on the training samples. However, the decision boundary between y_g and y_b leads to multiple critical errors on the test set (labeled in crosses).

Using the approach detailed in Algorithm 2, we show in Figure 2(b) the trajectories of the adversarial attacks on five different points. The large box in black is enlarged in Figure 2(c) for better visualization. These trajectories highlight the idea behind our augmentation scheme. The generated adversarial examples pushed y_g data points toward the boundary between y_g and y_b regions. The grey points are the intermediate gradient ascent steps, whereas the black points are the outputted adversarial examples. The adversarial attacks either stop when the points go across the classification boundary or when the designated number of maximization steps is reached. Boxes surrounding each point denotes the norm constraint $\|\boldsymbol{\delta}\| \le \epsilon$. Here we use infinity norm, $\epsilon = 1.5$, $K = 5$, and $\eta_2 = 0.05$.

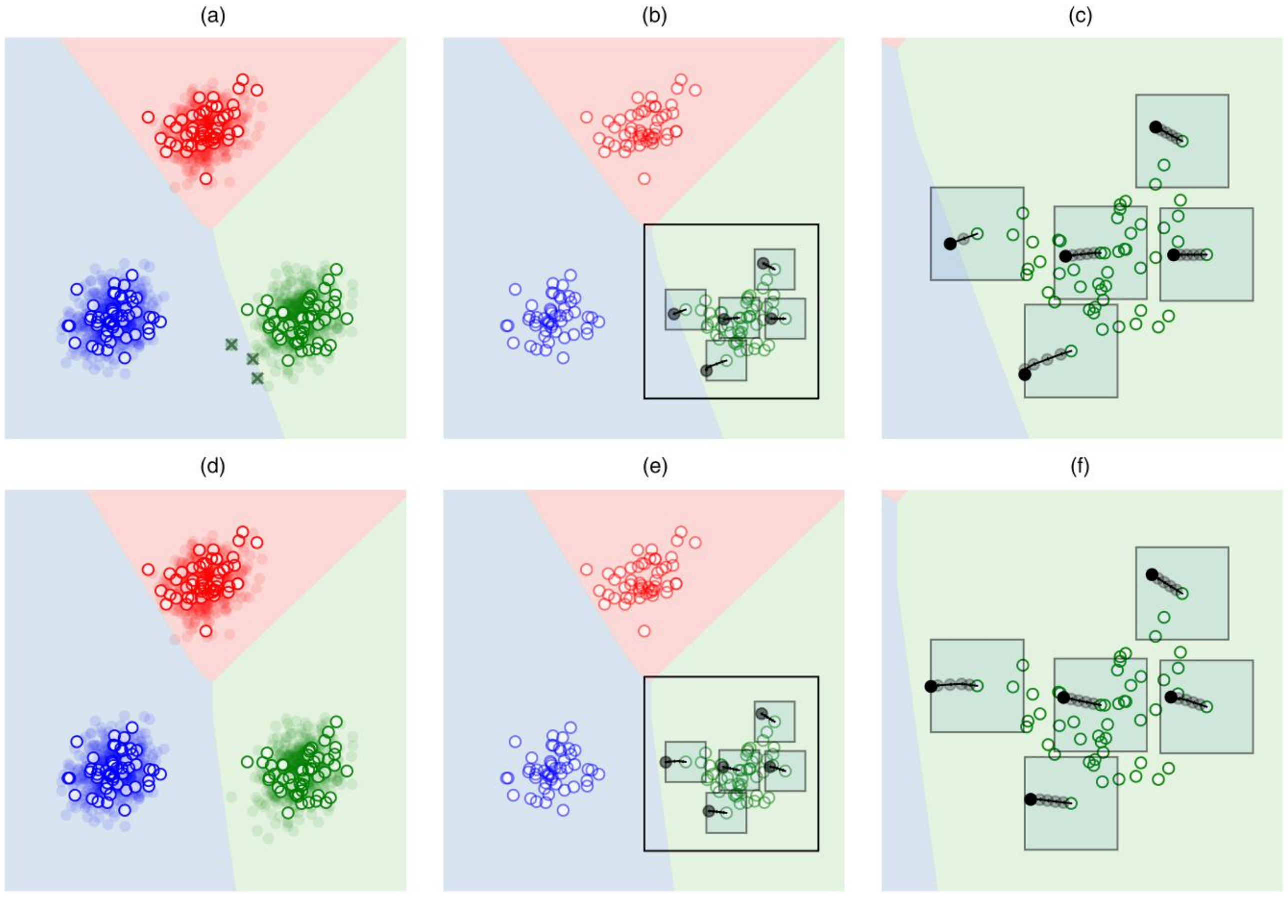
The results of our trained model are shown in Figure 2(d). After incorporating adversarial samples, the decision boundary between y_g and y_b was shifted toward class y_b, while the other boundaries were mostly unaffected. Interestingly, our updated model has no critical errors on the test set. As shown in Figure 2, (e) and (f), at convergence, the boundary between y_g and y_b is far enough from class y_g so that no more successful attacks (attacks that go across the boundary) can be achieved within the given number of maximization steps (i.e., budget).

5. Experiments

5.1. Failure of Simple Reweighting

Here we revisit the example in Section 1.2 using our approach. We present results on both CIFAR-10 and MNIST. We train a ResNet-34 (He et al. 2016) DNN for 300 epochs under cross-entropy loss using stochastic gradient descent (SGD) with momentum 0.9 and weight decay 10^{-4}. The learning rate for SGD starts with 0.1 and decays by 0.33 every 50 epochs. We also incorporate standard data augmentation practices when learning these data sets as in (He et al. 2016). We denote this model as the Baseline model as we only aim to minimize the empirical risk $(\frac{1}{N} \cdot \sum_{i=1}^{N} [\ell(p(\boldsymbol{\theta}; \boldsymbol{x}_i), y_i)])$. For both data sets, the Baseline model reaches a training loss of less than 0.001.

Figure 2. (Color online) Before ((a), (b), (c)) and After ((d), (e), (f)) CSADA



Notes. (a) Critical errors (in crosses). (b) CSADA examples (before). (c) Enlarged (b). (d) Avoided critical errors. (e) CSADA examples (after). (f) Enlarged (e).

Now starting with the pretrained model and for every possible pair (y', z') between the 10 classes, we train the penalty method in (3) and our adversarial augmentation model. We assume that predicting y' as z' incurs a cost of one and all other costs are zero. All models are trained for 10 epochs with a learning rate of $\eta_1 = 10^{-4}$. For CSADA, in CIFAR-10, the hyperparameters are $\eta_2 = 0.001$, $K = 5$, $\epsilon = 1$, $\tau = 1$, $\lambda = 10$. In MNIST, the hyperparameters are $\eta_2 = 0.05$, $K = 5$, $\epsilon = 10$, $\tau = 1$, and $\lambda = 10$. All experiments in this section are done using the stochastic version of CSADA in Algorithm 3.

The results on both MNIST and CIFAR-10 are shown in Figures 3 and 4. In the figures, each matrix entry corresponds to the pairwise error rate of the experiment done, with this pair being the critical one. From the results, we can obtain two key insights. (i) Even with an objective that only aims to reduce critical errors, the results did not improve compared with the Baseline. This again asserts the motivation of the paper and the need to rethink cost-sensitive learning in overparametrized models. (ii) We see that our alternative approach via CSADA results in a significant reduction in the pairwise error rate, showing the superior performance of our model.

Table 2. Toy Example Cost Matrix

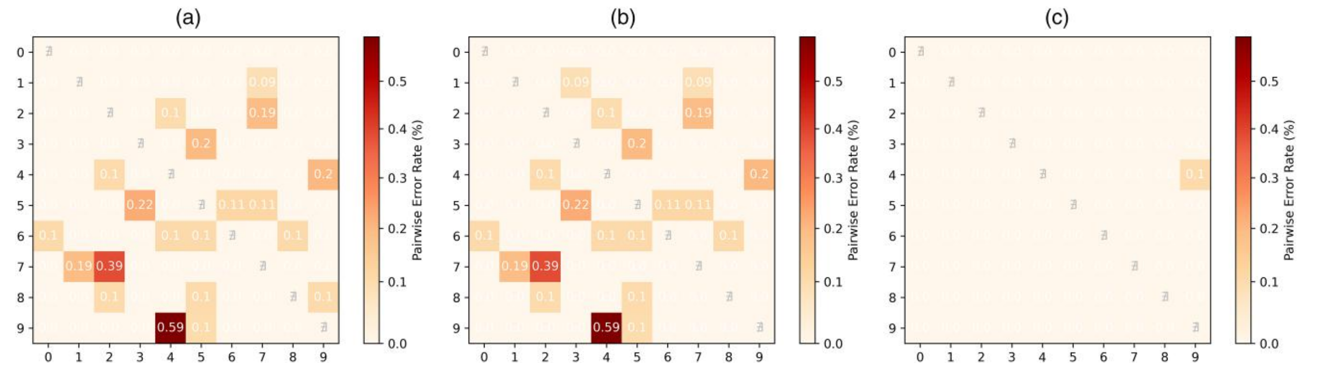
	y_r	y_g	y_b
y_r	0	0	0
y_g	0	0	1
y_b	0	0	0

5.2. Cost-Sensitive Training on CIFAR-10 and MNIST

Using the same Baseline model trained previously, we perform sensitivity analysis on our method and compare the results with other benchmarks. We generate a cost matrix $\mathcal{C}$, where $c(y,z) \sim \text{Pareto}(1, 1.5)$. We choose a Pareto distribution to emulate real-life situations such as pharmacy medical dispensing, where some mistakes are life-threatening and far more costly than others.

We start with sensitivity analysis on λ using CIFAR-10. Starting with the Baseline model, for each λ, we trained CSADA for 10 epochs at a fixed learning rate η_1 of 5×10^{-7}. The temperature τ is set to be three in this experiment. A small learning rate was chosen because we are refining the decision boundaries rather than

Figure 3. (Color online) Comparison of Baseline, Penalty, and CSADA Models on MNIST



Notes. (a) Baseline model. (b) Penalty method. (c) CSADA.

finding a completely different solution from the baseline. We use the infinity norm on $\boldsymbol{\delta}$, and set $\epsilon = 1$, $K = 10$, $\eta_2 = 5 \times 10^{-4}$, and $\tau = 3$. The experiment on each λ is replicated for three times to offset randomness.

Testing results are shown in Figure 5 where we report the weighted error rate (WER). The weighted error rate finds the average cost per prediction and is given as

$$\frac{1}{N_{test}} \cdot \sum_i c\left(y_i, \arg\max_{z \in \mathcal{Y}} \; p_z(\boldsymbol{\theta}; \boldsymbol{x}_i)\right).$$

When costs are all equal to one, WER recovers the overall accuracy.

Figure 5 provides interesting insights on the sensitivity of our model to λ. When $\lambda = 0$, we pertain a regular training with no cost-awareness. Conversely, when λ approaches infinity, the focus mainly shifts to adversarial samples which can prevent the model from learning important representations on the natural data set. For instance, our results in Figure 5 show that the total cost significantly reduces as we increase λ from 0.1 to 1. Afterward, the model is almost stable for values of λ between 1 and 10. Further increasing λ will shift the focus of the model to adversarial samples, which results in an increase in the total cost function. In practice, to prevent a misspecified large λ, one should monitor the training accuracy (i.e., accuracy on the training data set) of the cost-sensitive model such that it does not deviate too much from the pretrained model. This aligns with our goal to find a cost-sensitive solution within the feasible region of Problem (1).

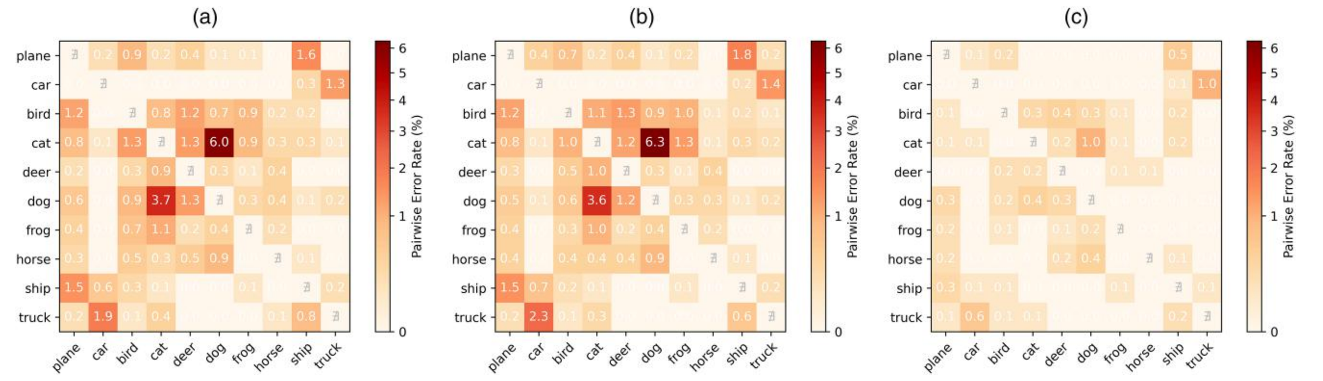
We also test model convergence on CIFAR-10 in Figure 6. Based on Figure 5, we set $\lambda = 2$. Similarly, the model is run for 10 epochs, and we replicate the experiment 10 times. In Figure 6, we report the mean and 90% prediction intervals for our loss function ($\ell_{\text{stochastic}}$) and the WER. The results again show a significant decrease in both loss and cost at the end of training.

Next, we compare our approach with two benchmarks: (i) CNN$_{\text{SOSR}}$ proposed by Chung et al. (2015) and (ii) the adjusted penalty method (AP), which uses misclassification cost regularization term to penalize critical errors. More specifically, the loss function is given as

$$\min_{\boldsymbol{\theta}} -\frac{1}{N} \cdot \sum_{i=1}^{N} (\log(p_{y_i}(\boldsymbol{\theta}; \boldsymbol{x}_i)) + \alpha \cdot \sum_{z \in \mathcal{Y}} c(y_i, z) \cdot \log(1 - p_z(\boldsymbol{\theta}; \boldsymbol{x}_i))).$$

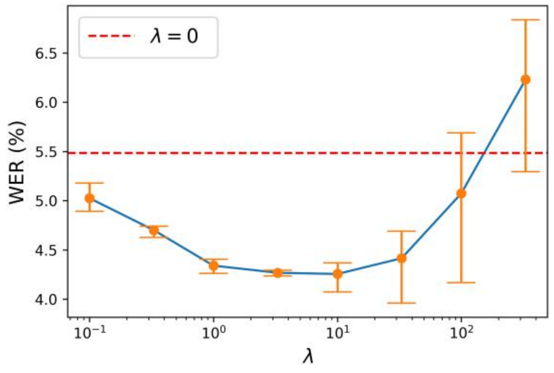
For all methods, we use ResNet-34 and start from the pretrained Baseline model. For CNN$_{\text{SOSR}}$, we add a

Figure 4. (Color online) Comparison of Baseline, Penalty, and CSADA Models on CIFAR-10



Notes. (a) Baseline model. (b) Penalty method. (c) CSADA.

Figure 5. (Color online) Cost Changes in Response to Hyperparameter λ



required regression layer. Settings for our model are kept the same. For the benchmarks, we found more epochs are needed for better performance. As such, on the two benchmarks, we use a starting learning rate of 10^{-4} that decays by 0.33 every 10 epochs; α is set to five. We followed the method of Chung et al. (2015) by pretraining only on the cross-entropy loss, so there are no tuning hyperparameters in $\mathrm{CNN}_{\mathrm{SOSR}}$. The mean and standard deviation of WER are shown in Table 3. On CIFAR-10, it can be seen that our algorithm has the lowest cost among the three. Overall, our algorithm requires fewer training epochs, has a lower cost, and does not require additional layers in NNs.

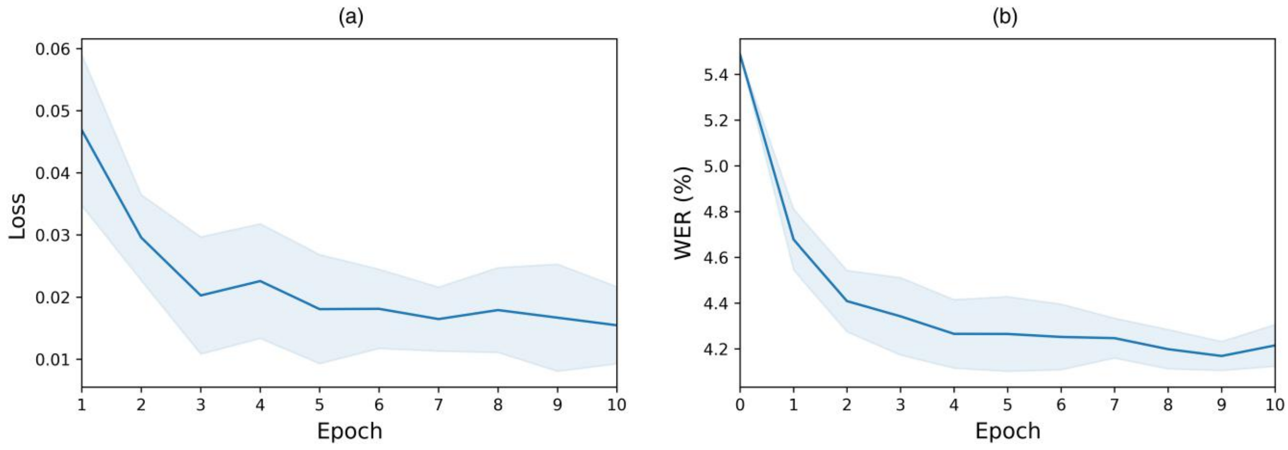
A similar comparison is further made for the MNIST data set. Each method was replicated five times. Here, the ϵ is relaxed to three, $K = 5$, and $\eta_2 = 0.05$. Other settings remain the same as in the CIFAR-10. As shown in Table 4, competing methods can only slightly reduce costs. In contrast, our approach was able to reduce cost while maintaining similar accuracy to the Baseline model.

5.3. PMI Data Set

5.3.1. Medication Dispensing Errors Overview. When medications are dispensed, they must match the prescription issued by the physician. Failing to dispense correct medications can lead to serious medical consequences. If a computer model is able to recognize the pill inside the medication bottle and confirm it matches the product written on the prescription, dispensing errors can be minimized. For this case study, the classification model takes pill images as input and predicts the medication product. However, in using a computer model for supporting medication dispensing, the costs can be different across different pairs of pills. For example, classifying the prescribed amiodarone hydrochloride 200 mg oral tablet (i.e., a medication used to control a person's heart rate) when the true label is allopurinol 100 mg oral tablet (i.e., a medication used to prevent gout attacks) can result in a patient going untreated for gout or can cause toxicity of the lungs (Wyeth Pharmaceuticals Inc. 2018). Other prediction errors, such as confusing one manufacturer for a second manufacturer for an acid reflux medication (ranitidine 150 mg oral tablet), are not likely to result in any harm to the patient. In our study, we assigned higher costs to critical mistakes. The imbalanced cost nature of medication errors makes this a good application for cost-sensitive training.

5.3.2. PMI Data Set Description. The data set consists of a collection of pill images and a meta-table reporting their national drug codes (NDCs). NDCs are unique product identifiers that are used to distinguish different medication products based on ingredient, strength, dose form, and manufacturer. The physical features of the pill for each NDC are a distinct combination of shape, size, color, scoring, and imprint. These pill images show the inside of filled medication bottles from a top-down view (Figure 7). The data set includes 13,955 images from 20

Figure 6. (Color online) Convergence of Algorithm 3 on CIFAR-10



Notes. (a) Loss (training). (b) WER (testing).

Table 3. Comparison of Methods on CIFAR-10 Data Set

	Baseline	CSADA	AP	SOSR
WER (%)	5.49	**4.20 (0.03)**	5.07 (2.06)	5.12 (2.08)
Top 1 cost pair error (%)	1.50	0.83 (0.08)	1.20 (0.07)	1.14 (0.16)
Top 2 cost pair error (%)	1.20	0.81 (0.08)	1.06 (0.08)	1.10 (0.29)
Top 3 cost pair error (%)	6.00	3.21 (0.16)	5.08 (0.18)	5.04 (0.91)
Overall accuracy (%)	95.70	95.54 (0.05)	95.58 (0.15)	95.53 (0.10)

Note. The best performance is highlighted in bold.

distinct NDCs. A list of the NDCs, along with their key physical features, is presented in Appendix A. Sample sizes in different NDCs are imbalanced. For each NDC, images are randomly split into training/validation/testing sets at a ratio of approximately 6:2:2. The size of the input images is $1{,}024 \times 960$ (pixel) and was center-cropped to 960×960 (pixel). The data set was made publicly available at https://deepblue.lib.umich.edu/data/concern/data_sets/6d56zw997 for reproducibility purposes and to encourage further research along this line.

5.3.3. Baseline Model Training on PMI Data Set. To train our Baseline model, we take the following steps for data augmentation. Whenever an image is sampled, it is first randomly cropped to a ratio that is generated randomly in the range of $(0.8, 1)$. Then, the image is resized to 224×224. The resized images are rotated by a degree that is generated randomly from the tuple $(0, 90, 180, 270)$. Finally, images are standardized with calculated mean and variance. We start with a pretrained ResNet-34 model and train our vanilla Baseline model on the PMI data set. We first replace the last layer of the pretrained model with randomly initialized weights. Freezing all but the last layer, the model is first trained for 10 epochs with a fixed learning rate of 10^{-4}. Then, all the layers were defrosted and trained for 20 epochs with a learning rate of 10^{-5} for the first 10 epochs and 10^{-6} for the rest. The ADAM optimizer (Kingma and Ba 2015) is used in this task. At the end of the training, the vanilla Baseline model achieved 99.61% accuracy on the test set. However, despite the good overall accuracy, we observed examples of several critical errors in the validation set.

5.3.4. Cost-Sensitive Training on the PMI Data Set. Expert costs are assigned to pairs according to their critical levels. A pharmacist (CL) reviewed the errors from the vanilla Baseline model to identify examples of critical pairs, which were weighted based on their potential danger. Table 5 lists the assigned expert-costs for the four critical pairs, and the remaining noncritical pairs are assigned costs of one. Starting from the vanilla Baseline model, we trained the model with three different methods and compared their performance with the vanilla Baseline model. To be consistent with the Baseline model, the ADAM optimizer is used for all three cost-sensitive methods. Our method (CSADA) was trained for 10 epochs at a fixed learning rate η_1 of 1×10^{-7}. Other hyperparameters are $\eta_2 = 1 \times 10^{-4}$, $K = 50$, $\lambda = 2$, and $\epsilon = 1$. The penalty method and CNN_{SOSR} were trained for 50 epochs, and the learning rate starts at 10^{-5} and decays by 0.33 every 10 epochs. We choose $\alpha = 5$ for the AP method. The hyperparameters for the two benchmarks were selected to achieve the best performance on the validation data set. The testing performance is shown in Table 6. Five replications were conducted for each method, where the mean of the five replications is presented along with the standard deviation in the bracket. Here we directly reported the total costs since the mistakes are few. The total cost is simply the summation of expert costs on all misclassifications. We also reported the number of errors on different pairs in Table 5. The results show that only CSADA was able to cut the cost (by half) and prevent some critical errors. In turn, AP and SOSR exhibited no statistically significant improvement over the Baseline model, especially given the high variance. Indeed, this is not surprising as the training loss was about 0.003. This again confirms that in overparametrized models where model capacity is not a concern, cost-sensitive training can be very challenging if we completely depend on a training set that can be perfectly fit.

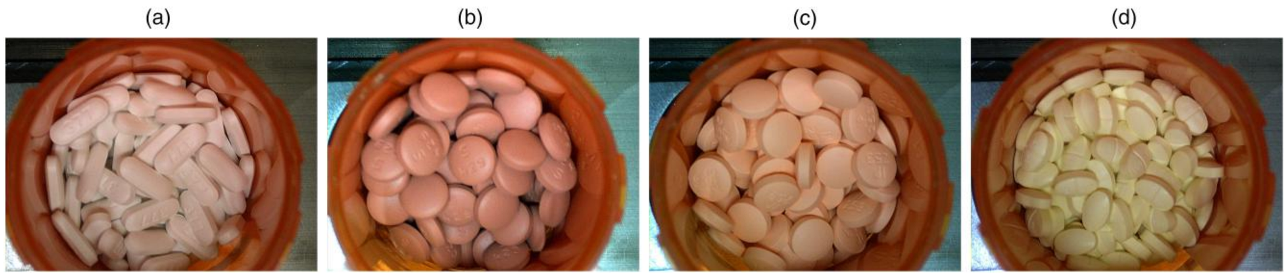
To further confirm the challenges in overparameterization, we compared the three models on a shallower network (ResNet-9). We trained the Baseline model

Table 4. Comparison of Methods on MNIST Data set

	Baseline	CSADA	AP	SOSR
WER (%)	0.46	**0.25 (0.01)**	0.44 (0.01)	0.43 (0.06)
Top 1 cost pair error (%)	0.10	0.00 (0.00)	0.10 (0.00)	0.10 (0.00)
Top 2 cost pair error (%)	0.59	0.10 (0.00)	0.54 (0.05)	0.50 (0.15)
Top 3 cost pair error (%)	0.00	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Overall accuracy (%)	99.67	99.62 (0.01)	99.66 (0.01)	99.64 (0.02)

Note. The best performance is highlighted in bold.

Figure 7. (Color online) PMI Data Set Examples



Notes. (a) 57664-0377. (b) 64380-0803. (c) 65162-0253. (d) 67253-0901.

Table 5. Average Number of Errors Across Five Replications

Error type	Expert cost	Baseline	CSADA	AP	SOSR
50111-0434 to 00591-0461	10	1	0.0 (0.0)	0.2 (0.4)	0.6 (0.5)
53489-0156 to 68382-0227	10	0	0.0 (0.0)	0.0 (0.0)	0.2 (0.4)
53746-0544 to 00378-0208	10	1	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)
68382-0227 to 53489-0156	8	1	0.0 (0.0)	1.0 (0.7)	0.0 (0.0)
Noncritical pairs	1	8	8.0 (1.0)	11 (2.3)	11.2 (0.8)

Table 6. Comparison Across Benchamrks (ResNet-34)

	Baseline	CSADA	AP	SOSR
Total cost	36.0	**18.0 (1.0)**	31.0 (7.8)	29.2 (8.8)
50111-0434 to 00591-0461 error rate (%)	0.50	0.00 (0.00)	0.10 (0.22)	0.30 (0.27)
53489-0156 to 68382-0227 error rate (%)	0.00	0.00 (0.00)	0.00 (0.00)	0.10 (0.22)
53746-0544 to 00378-0208 error rate (%)	3.12	3.12 (0.00)	3.12 (0.00)	3.12 (0.00)
68382-0227 to 53489-0156 error rate (%)	2.56	0.00 (0.00)	2.56 (1.81)	0.00 (0.00)
Overall accuracy (%)	99.61	99.68 (0.04)	99.53 (0.09)	99.53 (0.05)

Note. The best performance is highlighted in bold.

Table 7. Comparison of Methods (ResNet-9)

	Baseline	CSADA	AP	SOSR
Total cost	434.0	**345.4 (14.2)**	381.2 (16.6)	413.6 (23.8)
50111-0434 to 00591-0461 error rate (%)	4.50	1.70 (0.84)	2.00 (0.79)	1.40 (0.42)
53489-0156 to 68382-0227 error rate (%)	1.50	1.70 (0.45)	1.70 (0.27)	1.20 (0.91)
53746-0544 to 00378-0208 error rate (%)	18.75	15.12 (2.83)	15.62 (2.21)	13.12 (1.40)
68382-0227 to 53489-0156 error rate (%)	25.64	21.54 (2.92)	21.54 (3.44)	30.77 (11.47)
Overall accuracy (%)	92.75	92.84 (0.17)	92.43 (0.17)	91.21 (0.35)

Note. The best performance is highlighted in bold.

under cross-entropy loss using ADAM optimizer for 300 epochs. The learning rate started with 0.1 and decayed every 50 epochs by 0.33. Using the Baseline model, CSADA was trained for additional 10 epochs with a fixed learning rate η_1 of 5×10^{-5}. Other hyperparameters follow $\eta_2 = 0.01$, $K = 10$, $\lambda = 10$, $\epsilon = 1$, and $\tau = 3$. Both AP and SOSR were trained for 50 epochs starting from the Baseline model. The learning rate started at 0.001 and decayed every 10 epochs by 0.33. Hyperparameter α in AP was set to five.

As shown in Table 7 the testing accuracy is now much smaller because we are using a model with less capacity. Also, the training error of the Baseline model now increased to around 0.09. Although AP and SOSR reduced the cost, our model still shows superior performance. Interestingly, both methods AP and SOSR performed better on the smaller, less-complex network compared with ResNet-34. This further emphasizes our claim that existing cost-aware approaches encounter degraded performance when overparameterized networks are used.

6. Conclusion

In this paper, we study the problem of cost-sensitive classification that arises in applications where different misclassification errors have different costs. Despite the rich literature on cost-sensitive learning, we demonstrated the failure of conventional methods when applied to overparameterized models such as DNNs. For this problem, we propose a cost-sensitive method that generates targeted adversarial examples that are used in training to push decision boundaries in directions that minimize critical errors. Mathematically, we propose a penalized cost-aware bilevel optimization framework that penalizes the loss incurred by the generated adversarial examples. We further propose a multiascent-descent gradient-based algorithm for solving the optimization problem. The empirical performance of our model is demonstrated through various experiments on MNIST, CIFAR-10, and a pharmacy medication image (PMI) data set, which we made publicly available. In all experiments, our method effectively minimized the overall cost and reduced critical errors while achieving comparable performance in terms of overall accuracy.

Appendix A. PMI Data Set Metadata

The PMI data set is a subset of a large pharmaceutical data set with more than 300 medications (Lester et al. 2021). The 20 classes selected are the classes where the model makes the most mistakes. This appendix provides a reference table characterizing the key physical features of 20 selected medication (Table A.1). The *Imprint* column lists all the imprinted labels on medication pills. The size column reports the width of pills in millimeters. Within an NDC, images are randomly split into training/validation/testing sets at a ratio of approximately 6:2:2. There is a one-to-one correspondence between imprints and NDC, making it an important feature for image classification. Classes are imbalanced due to their different frequencies during data collection.

Table A.1. Metadata for PMI Data Set

NDC	Drug name	Shape	Color	Imprint	Size	Training size	Validation size	Testing size
00378-0208	Furosemide 20 mg oral tablet	Round	White	M2	6	602	199	200
00378-3855	Escitalopram 5 mg oral tablet	Round	White	M;EC5	6	129	42	42
00591-0461	Glipizide 10 mg oral tablet	Round	White	Watson;461	10	148	48	48
16729-0020	Pioglitazone 15 mg oral tablet	Round	White	P;15	5	258	84	85
16729-0168	Escitalopram 5 mg oral tablet	Round	White	5	5	129	42	42
50111-0434	Trazodone hydrochloride 100 mg oral tablet	Round	White	PLIVA;434	11	601	199	200
53489-0156	Allopurinol 100 mg oral tablet	Round	White	MP;71	10	600	200	200
53746-0544	Primidone 50 mg oral tablet	Round	White	AN;44	6	97	31	32
57664-0377	Tramadol hydrochloride 50 mg oral tablet	Capsule	White	377	13	601	199	200
62037-0831	24-hour metoprolol succinate 50 mg extended	Round	White	831	9	600	200	200
62037-0832	24-hour metoprolol succinate 100 mg extended	Round	White	832	10	601	200	200
64380-0803	Ranitidine 150 mg oral tablet (Strides Pharma)	Round	Brown	S;429	10	548	181	182
65162-0253	Ranitidine 150 mg oral tablet (Amneal Pharmaceuticals)	Round	Orange	IP;253	9	557	184	185
67253-0901	Alprazolam 0.5 mg oral tablet	Oval	Yellow	S901	9	497	164	165
68382-0008	Lamotrigine 100 mg oral tablet	Round	White	ZC;80	10	423	139	140
68382-0227	Amiodarone hydrochloride 200 mg oral tablet	Round	White	ZE;65	10	120	39	39
69097-0127	Amlodipine 5 mg oral tablet	Round	White	127;C	8	600	200	200
69097-0128	Amlodipine 10 mg oral tablet	Round	White	128;C	8	600	200	200
69315-0904	Lorazepam 0.5 mg oral tablet	Round	White	EP;904	5	357	118	118
69315-0905	Lorazepam 1 mg oral tablet	Round	White	EP;905;1	7	325	107	108

Appendix B. Additional Experiments

B.1. Experiments for Transformers

In demonstrate the effectiveness of our model on various inference tasks, we test our model on transformers (Vaswani et al. 2017), which have become very popular in large language models (Brown et al. 2020), and find application in computer vision (Dosovitskiy et al. 2020) and multimodal machine learning (Li et al. 2022).

We trained a vision transformer (ViT) as our Baseline model. Standard data augmentation for CIFAR-10 is applied to the Baseline model. We used AdamW optimizer (Loshchilov and Hutter 2017) and the one-cycle learning rate scheduler as described in Smith and Topin (2019), which is the state-of-the-art practice in training transformer models. Weight decay is set to 0.1, the maximum learning rate is set to 10^{-3}, and the batch size is set to 32. The model is trained for 200 epochs. The baseline model has a 93.90% test accuracy. The test accuracy is lower than ResNet-34. This is expected because transformers do not have a strong inductive bias as a CNN.

The three methods are then tested on the Baseline model. For our model CSADA, the parameters are set to be $\tau = 3, \eta_1 = 10^{-7}, \eta_2 = 10^{-4}, \epsilon = 1, K = 20, \lambda = 1, T = 10$. For the adjusted penalty method, hyperparameter $\alpha = 5$. Both benchmark methods (AP and SOSR) were trained for 50 epochs with a starting learning rate of 10^{-5}, which decays at a factor of 0.33 after every 10 epochs. We report the total cost on the test data set along with the error rates on the three most costly pairs. The mean and the standard deviations (in brackets) across different replications are shown in Table B.1. It can be seen from the table that our CSADA consistently returns the lowest cost and is more effective in reducing error rates on high-cost pairs.

B.2. Experiments for an OCT-Scan Data Set

Here, we present an OCT data set of the retina (Kermany et al. 2018) with real-world costs. The OCT data set has four classes (Figure B.1) and has an expert-specified cost matrix. Because our Baseline model made no critical errors on the original test set, we randomly move images from the training set to the test set such that each training class has 8,000 images, and each test class has 750 images. We use ResNet-34 for this experiment and SGD as the optimizer. Weight decay is set to be 10^{-4}, and the batch size is set to be 32. The Baseline model ($\lambda = 0$) is trained for 200. The starting learning rate is set to be 10^{-1}, which decays by a factor of 0.33 every 20 epoch.

The three methods are then tested on the Baseline model. For our CSADA, the parameters are set to be $\tau = 2, \eta_1 = 5 \times 10^{-7}, \eta_2 = 10^{-4}, \epsilon = 1, K = 20, \lambda = 1, T = 3$. For the adjusted penalty method, we use hyperparameter $\alpha = 5$. Both benchmark methods (AP and SOSR) were trained for 50 epochs with a starting learning rate of 10^{-5}, which decays at a factor of 0.33 after every 10 epochs. Five replications are run for each method. We report the total cost on the test data set along with the error rates on the four most costly pairs. The mean and the standard deviations (in brackets) across different replications are shown in Table B.2. Again, our CSADA achieved the lowest overall cost among all the methods, which confirms the superiority of our method.

B.3. Additional Experiments on Small Capacity Models

Here, we provide four additional experiments: (i) MobileNet V2 on the OCT data set, (ii) MobileNet V2 on our PMI data set, (iii) VGG16 on the OCT data set, and (iv) VGG16 on the PMI data set. We report the hyperparameters used for each method in Tables B.3–B.5. Both VGG16 and MobileNet have worse performance than ResNet34 we presented in this paper.

Across all four experiments in Tables B.6–B.9, we still constantly observe improvements in cost with the help of our method compared with the Baseline model. Along with the previous experiment using transformers, this again showcases the generalizability of our method to different backbones.

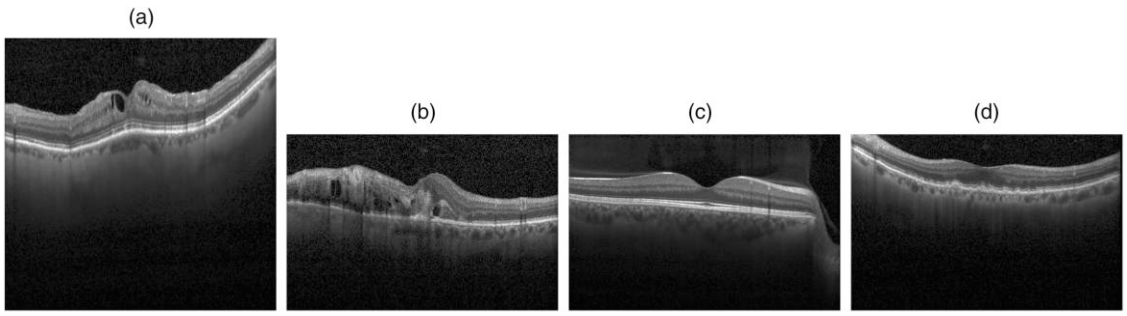
Besides that, it is interesting to notice that similar observations can be made when applying MobileNet on the PMI data set as in our previous experiment with ResNet 9. Although our method can still reduce costs, we no longer observe improvement over the conventional reweighting methods. Indeed, MobileNet is designed for edge deployment and is one of the smallest NNs. We used cross-entropy loss for training, the training loss of MobileNet can only reach around 0.04 in convergence, and the training accuracy is around 98.5%. Hence, it is expected that reweighting methods will be effective in this setting. This again confirms our motivation that our method is designed for overparameterized settings where traditional cost-sensitive models fail.

Table B.1. Comparison of Methods of ViT on CIFAR-10 Data Set

	Baseline	CSADA	AP	SOSR
WER (%)	7.09	**5.55 (0.05)**	6.93 (0.26)	7.62 (0.35)
Top 1 cost pair error (%)	1.40	0.60 (0.00)	1.22 (0.29)	1.86 (0.67)
Top 2 cost pair error (%)	2.20	1.56 (0.09)	2.28 (0.08)	2.06 (0.25)
Top 3 cost pair error (%)	6.80	3.30 (0.00)	5.84 (0.34)	6.82 (1.55)
Overall accuracy (%)	93.90	93.88 (0.03)	93.64 (0.10)	93.20 (0.12)

Note. The best performance is highlighted in bold.

Figure B.1. OCT Data Set Examples



Notes. (a) DME. (b) CNV. (c) NORMAL. (d) DRUSEN.

Table B.2. Comparison of Methods of ResNet-34 on OCT Data Set

	Baseline	CSADA	AP	SOSR
Cost	157	**151 (1.22)**	158.2 (2.17)	168.8 (11.30)
Top 1 cost pair error (%)	1.47	1.07 (0.00)	1.47 (0.00)	1.63 (0.30)
Top 2 cost pair error (%)	0.13	0.13 (0.00)	0.13 (0.00)	0.11 (0.11)
Top 3 cost pair error (%)	0.13	0.13 (0.00)	0.13 (0.00)	0.24 (0.11)
Top 4 cost pair error (%)	2.67	2.13 (0.10)	2.35 (0.28)	2.61 (0.22)
Overall accuracy (%)	96.67	96.44 (0.05)	96.54 (0.13)	96.39 (0.21)

Note. The best performance is highlighted in bold.

Table B.3. Hyperparameters Used in Training the Baseline Model

Setting	Optimizer	Learning rate	Epochs	Weight decay
Mobile+OCT	SGD	Starts 10^{-2}, decays 0.33 per 20 epochs	100	10^{-4}
VGG+OCT	SGD	Starts 10^{-2}, decays 0.33 per 20 epochs	100	10^{-4}
Mobile+PMI	SGD	Starts 10^{-2}, decays 0.33 per 25 epochs	200	10^{-4}
VGG+PMI	ADAM	Starts 10^{-4}, decays 0.33 per 50 epochs	200	10^{-4}

Table B.4. Hyperparameters Used for Cost-Sensitive Models

Setting	η_1	T	λ	ϵ	η_2	K	τ	α
Mobile+OCT	1×10^{-7}	10	0.5	1	10^{-4}	3	2	5
VGG+OCT	1×10^{-7}	5	0.5	1	10^{-3}	3	2	5
Mobile+PMI	5×10^{-7}	5	2	1	10^{-3}	5	3	5
VGG+PMI	5×10^{-7}	3	2	1	0.1	5	3	5

Table B.5. Hyperparameters Used for Training AP and SOSR

Setting	Optimizer	Learning rate	Epochs	Weight decay
Mobile+OCT	SGD	Starts 10^{-5}, decays 0.33 per 10 epochs	30	10^{-4}
VGG+OCT	SGD	Starts 10^{-5}, decays 0.33 per 10 epochs	30	10^{-4}
Mobile+PMI	SGD	Starts 10^{-5}, decays 0.33 per 10 epochs	30	10^{-4}
VGG+PMI	ADAM	Starts 10^{-5}, decays 0.33 per 10 epochs	30	10^{-4}

Table B.6. Comparison of Methods of Mobile Net on OCT Data Set

	Baseline	CSADA	AP	SOSR
Cost	118	**111.8 (1.69)**	121.6 (0.89)	131.4 (4.1)
Top 1 cost pair error (%)	1.2	0.80 (0.00)	1.20 (0.00)	1.25 (0.07)
Top 2 cost pair error (%)	0.13	0.13 (0.00)	0.13 (0.00)	0.13 (0.00)
Top 3 cost pair error (%)	0.27	0.27 (0.00)	0.27 (0.00)	0.27 (0.10)
Top 4 cost pair error (%)	1.60	1.20 (0.00)	1.73 (0.00)	1.68 (0.20)
Overall accuracy (%)	97.53	97.34 (0.06)	97.45 (0.03)	97.15 (0.15)

Note. The best performance is highlighted in bold.

Table B.7. Comparison of Methods of VGG on OCT Data Set

	Baseline	CSADA	AP	SOSR
Cost	119	**110.6 (1.1)**	120.6 (0.89)	133.4 (20.2)
Top 1 cost pair error (%)	0.93	0.67 (0.00)	0.93 (0.00)	1.04 (0.61)
Top 2 cost pair error (%)	0.13	0.00 (0.00)	0.13 (0.00)	0.05 (0.07)
Top 3 cost pair error (%)	0.00	0.00 (0.00)	0.00 (0.00)	0.05 (0.12)
Top 4 cost pair error (%)	2.00	1.44 (0.06)	1.92 (0.07)	2.05 (0.86)
Overall accuracy (%)	97.33	97.17 (0.03)	97.26 (0.02)	96.90 (0.19)

Note. The best performance is highlighted in bold.

Table B.8. Comparison of Methods of Mobile Net on PMI Data Set

	Baseline	CSADA	AP	SOSR
Cost	297.0	264.6 (35.9)	251.0 (26.4)	660.8 (97.4)
Top 1 cost pair error (%)	2.00	0.40 (0.55)	0.00 (0.00)	0.00 (0.00)
Top 2 cost pair error (%)	0.50	0.50 (0.00)	0.50 (0.00)	0.00 (0.00)
Top 3 cost pair error (%)	15.62	6.25 (0.00)	5.62 (2.62)	48.75 (27.74)
Top 4 cost pair error (%)	17.95	25.13 (9.14)	23.08 (7.48)	52.82 (18.01)
Overall accuracy (%)	94.33	94.19 (0.29)	94.16 (0.29)	86.49 (0.66)

Table B.9. Comparison of Methods of VGG on PMI Data Set

	Baseline	CSADA	AP	SOSR
Cost	430.0	369.2 (6.1)	373.0 (19.6)	378.8 (21.9)
Top 1 cost pair error (%)	3.50	0.20 (0.27)	1.90 (1.29)	1.30 (0.91)
Top 2 cost pair error (%)	3.50	1.40 (0.65)	2.10 (1.29)	3.30 (1.82)
Top 3 cost pair error (%)	21.88	15.12 (1.40)	17.50 (3.57)	18.12 (4.64)
Top 4 cost pair error (%)	12.82	22.56 (5.56)	16.41 (1.41)	14.36 (6.93)
Overall accuracy (%)	92.61	91.54 (0.17)	92.61 (0.31)	92.66 (0.21)

References

Arpit D, Jastrzębski S, Ballas N, Krueger D, Bengio E, Kanwal MS, Maharaj T, et al. (2017) A closer look at memorization in deep networks. *Proc. Internat. Conf. on Machine Learn.* (PMLR, New York), 233–242.

Bahnsen AC, Aouada D, Ottersten B (2014) Example-dependent cost-sensitive logistic regression for credit scoring. *Proc. 13th Internat. Conf. on Machine Learn. and Applications* (IEEE, New York), 263–269.

Bartlett PL, Long PM, Lugosi G, Tsigler A (2020) Benign overfitting in linear regression. *Proc. Natl. Acad. Sci. USA* 117(48):30063–30070.

Belkin M, Hsu DJ, Mitra P (2018) Overfitting or perfect fitting? Risk bounds for classification and regression rules that interpolate. *Adv. Neural Inform. Processing Systems 31* (MIT Press, Cambridge, MA), 2300–2311.

Breiman L, Friedman J, Olshen R, Stone C (1984) *Classification and Regression Trees* (Wadsworth, Belmont, CA).

Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, Neelakantan A, et al. (2020) Language models are few-shot learners. *Adv. Neural Inform. Processing Systems 33* (MIT Press, Cambridge, MA), 1877–1901.

Cao X, Gong NZ (2017) Mitigating evasion attacks to deep neural networks via region-based classification. *Proc. 33rd Annual Computer Security Applications Conf.* (ACM, New York), 278–287.

Chan P, Stolfo SJ (1999) Toward scalable learning with non-uniform distributions: Effects and a multi-classifier approach. *Proc. 4th Internat. Conf. on Knowledge Discovery and Data Mining* (ACM, New York).

Chung YA, Lin HT, Yang SW (2015) Cost-aware pre-training for multiclass cost-sensitive deep learning. Preprint, submitted November 30, https://arxiv.org/abs/1511.09337.

Chung YA, Yang SW, Lin HT (2020) Cost-sensitive deep learning with layer-wise cost estimation. *Proc. Internat. Conf. on Tech. and Applications of Artificial Intelligence* (IEEE, New York), 108–113.

Deng L (2012) The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* 29(6):141–142.

Devlin J, Chang MW, Lee K, Toutanova K (2018) Bert: Pre-training of deep bidirectional transformers for language understanding.

Preprint, submitted October 11, https://arxiv.org/abs/1810.04805.

Domingos P (1999) Metacost: A general method for making classifiers cost-sensitive. *Proc. 5th ACM SIGKDD Internat. Conf. on Knowledge Discovery and Data Mining* (ACM, New York), 155–164.

Donhauser K, Tifrea A, Aerni M, Heckel R, Yang F (2021) Interpolation can hurt robust generalization even when there is no noise. *Adv. Neural Inform. Processing Systems 34* (MIT Press, Cambridge, MA), 23465–23477.

Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, et al. (2020) An image is worth 16x16 words: Transformers for image recognition at scale. Preprint, submitted October 22, https://arxiv.org/abs/2010.11929.

Goodfellow IJ, Shlens J, Szegedy C (2014) Explaining and harnessing adversarial examples. Preprint, submitted December 20, https://arxiv.org/abs/1412.6572.

Guo C, Pleiss G, Sun Y, Weinberger KQ (2017) On calibration of modern neural networks. *Proc. Internat. Conf. on Machine Learn.* (PMLR, New York), 1321–1330.

Hassani H, Javanmard A (2022) The curse of overparametrization in adversarial training: Precise analysis of robust generalization for random features regression. Preprint, submitted January 13, https://arxiv.org/abs/2201.05149.

He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (IEEE, Piscataway, NJ), 770–778.

Health Services Department (2019) National action plan for adverse drug event prevention, office of disease prevention and health promotion, department of health and human services. Accessed November 1, 2021, https://health.gov/sites/default/files/2019-09/ADE-Action-Plan-508c.pdf.

Heo B, Lee M, Yun S, Choi JY (2019) Knowledge distillation with adversarial samples supporting decision boundary. *Proc. Conf. AAAI Artificial Intelligence* 33(01):3771–3778.

Institute for Safe Medication Practices (ISMP) (2021) ISMP list of high-alert medications in community/ambulatory care settings. Accessed July 10, 2022, https://www.ismp.org/recommendations/high-alert-medications-community-ambulatory-list.

Iranmehr A, Masnadi-Shirazi H, Vasconcelos N (2019) Cost-sensitive support vector machines. *Neurocomputing* 343:50–64.

Kermany DS, Goldbaum M, Cai W, Valentim CC, Liang H, Baxter SL, McKeown A, et al. (2018) Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell* 172(5):1122–1131.

Kingma DP, Ba J (2015) Adam: A method for stochastic optimization. *3rd Internat. Conf. Learning Representations (San Diego)*.

Kozerawski J, Fragoso V, Karianakis N, Mittal G, Turk M, Chen M (2020) Blt: Balancing long-tailed data sets with adversarially-perturbed images. *Proc. Asian Conf. on Computer Vision* (ACM, New York).

Kukar M, Kononenko I (1998) Cost-sensitive learning with neural networks. *ECAI* 15(27):88–94.

Kurakin A, Goodfellow IJ, Bengio S (2018) Adversarial examples in the physical world. *Artificial Intelligence Safety and Security* (Chapman and Hall/CRC, Boca Raton, FL), 99–112.

Lester CA, Li J, Ding Y, Rowell B, Yang J, Kontar RA (2021) Performance evaluation of a prescription medication image classification model: An observational cohort. *NPJ Digital Medicine* 4(1):1–8.

Li J, Li D, Xiong C, Hoi S (2022) Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. *Proc. Internat. Conf. on Machine Learn.* (PMLR, New York), 12888–12900.

Li X, Ma H, Meng L, Meng X (2021) Comparative study of adversarial training methods for long-tailed classification. *Proc. 1st Internat. Workshop on Adversarial Learn. for Multimedia* (ACM, New York), 1–7.

Liu Z, Lin Y, Cao Y, Hu H, Wei Y, Zhang Z, Lin S, et al. (2021) Swin transformer: Hierarchical vision transformer using shifted windows. *Proc. IEEE/CVF Internat. Conf. on Computer Vision* (IEEE, Piscataway, NJ), 10012–10022.

Lomax S, Vadera S (2013) A survey of cost-sensitive decision tree induction algorithms. *ACM Comput. Survey* 45(2):1–35.

Loshchilov I, Hutter F (2017) Decoupled weight decay regularization. Preprint, submitted November 14, https://arxiv.org/abs/1711.05101.

Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A (2018) Toward deep learning models resistant to adversarial attacks. *Proc. Internat. Conf. on Learn. Representations* (ICLR, Appleton, WI).

Moosavi-Dezfooli SM, Fawzi A, Frossard P (2016) Deepfool: A simple and accurate method to fool deep neural networks. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (IEEE, Piscataway, NJ), 2574–2582.

Park S, Hong Y, Heo B, Yun S, Choi JY (2022) The majority can help the minority: Context-rich minority oversampling for long-tailed classification. *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition* (IEEE, Piscataway, NJ), 6887–6896.

Shen H, Chen S, Wang R, Wang X (2023) Adversarial learning with cost-sensitive classes. *IEEE Trans. Cybernetics* 53(8): 4855–4866.

Smith LN, Topin N (2019) Super-convergence: Very fast training of neural networks using large learning rates. *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, vol. 11006 (SPIE, Bellingham, WA), 369–386.

Sun W, Kontar RA, Jin J, Chang TS (2022) A continual learning framework for adaptive defect classification and inspection. Preprint, submitted March 16, https://arxiv.org/abs/2203.08796.

Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, Fergus R (2013) Intriguing properties of neural networks. Preprint, submitted December 21, https://arxiv.org/abs/1312.6199.

Tu HH, Lin HT (2010) One-sided support vector regression for multiclass cost-sensitive classification. *Proc. Internat. Conf. on Machine Learn.* (PMLR, New York).

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, et al. (2017) Attention is all you need. *Adv. Neural Inform. Processing Systems 30* (MIT Press, Cambridge, MA), 5998–6008.

Volpi R, Namkoong H, Sener O, Duchi JC, Murino V, Savarese S (2018) Generalizing to unseen domains via adversarial data augmentation. Preprint, submitted May 30, https://arxiv.org/abs/1805.12018.

Wyeth Pharmaceuticals Inc (2018) Cordarone [package insert]. Accessed July 8, 2022, https://www.accessdata.fda.gov/drugsatfda_docs/label/2018/018972s054lbl.pdf.

Yue X, Nouiehed M, Kontar RA (2021) Gifair-fl: An approach for group and individual fairness in federated learning. Preprint, submitted August 5, https://arxiv.org/abs/2108.02741.

Zadrozny B, Elkan C (2001) Learning and making decisions when costs and probabilities are both unknown. *Proc. 7th ACM SIGKDD Internat. Conf. on Knowledge Discovery and Data Mining* (ACM, New York), 204–213.

Zadrozny B, Langford J, Abe N (2003) Cost-sensitive learning by cost-proportionate example weighting. *Proc. 3rd IEEE Internat. Conf. on Data Mining* (IEEE, New York), 435–442.

Zhang X, Evans D (2018) Cost-sensitive robustness against adversarial examples. Preprint, submitted October 22, https://arxiv.org/abs/1810.09225.

Zhang C, Bengio S, Hardt M, Recht B, Vinyals O (2016) Understanding deep learning requires rethinking generalization. Preprint, submitted November 10, https://arxiv.org/abs/1611.03530.

Zhang C, Bengio S, Hardt M, Recht B, Vinyals O (2021) Understanding deep learning (still) requires rethinking generalization. *Comm. ACM* 64(3):107–115.

Zhang Y, Kang B, Hooi B, Yan S, Feng J (2023) Deep long-tailed learning: A survey. *IEEE Trans. Pattern Anal. Machine Intelligence* (IEEE, Piscataway, NJ).

Zhou ZH, Liu XY (2005) Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Trans. Knowledge Data Engrg.* 18(1):63–77.