Robust Learning from Untrusted Sources

Nikola Konstantinov [1] **Christoph H. Lampert** [1]

Abstract

Modern machine learning methods often require more data for training than a single expert can provide. Therefore, it has become a standard procedure to collect data from multiple external sources, e.g. via crowdsourcing. Unfortunately, the quality of these sources is not always guaranteed. As further complications, the data might be stored in a distributed way, or might even have to remain private. In this work, we address the question of how to learn robustly in such scenarios. Studying the problem through the lens of statistical learning theory, we derive a procedure that allows for learning from all available sources, yet automatically suppresses irrelevant or corrupted data. We show by extensive experiments that our method provides significant improvements over alternative approaches from robust statistics and distributed optimization.

1. Introduction

Due to the outstanding performance of modern machine learning algorithms on various real-world tasks, there is an increasing amount of interest by practitioners in producing predictive models, specific to their purposes. In many application domains, however, it may be prohibitively expensive for a single expert to produce a high-quality labeled dataset, that is large enough for training a good model. Therefore, it has become a common practice to obtain data from various external data sources. Examples range from the use of crowdsourcing platforms, through collecting data from different websites and social networks profiles, to collaborating with other parties working in similar domains.

Naturally, datasets obtained from such sources vary greatly in quality, reliability and relevance for the learning task. For instance, genetic data from multiple laboratories may have been obtained via different measurement devices or data preprocessing techniques (Wahlsten et al., 2003). In the case of crowdsourcing, a typical problem is label bias and label noise, due to incompetent or malicious workers (Wais et al., 2010). More generally, statistical and machine learning models are known to suffer in performance due to gross errors, contaminations and adversarial modifications of the data (Tukey, 1960; Biggio et al., 2012). The variety of possible deviations from the target data distribution, as well as the large volume and dimensionality of the data in real-world applications, make the assessment of the quality of the provided data a difficult task. An additional complication is that the data might have to remain decentralized, because of high communication costs, or it might not be directly available for inspection, due to privacy constraints.

In this paper we study the problem of *how to learn from multiple untrusted sources*, while being *robust to any corruptions* of the data provided from each of them. As an alternative to the naive approaches of simply training on all data or only on a trusted subset, we propose a method that automatically assigns weights to the sources. To this end, we build up on techniques from the domain adaptation literature and prove an upper bound on the expected loss of a predictor, learned by minimizing any weighted version of the empirical loss. Based on these theoretical insights, our algorithm selects the weights for the sources by approximately minimizing this upper bound.

Intuitively, *the weights are assigned to the sources according to the quality and reliability of the data they provide*, quantified by an appropriate measure of trust we introduce. This is achieved by comparing the data from each source to a *small reference dataset*, obtained or trusted by the learner. The measure can also be computed locally at every source or by a gradient-based optimization procedure, which allows for the implementation of the algorithm under *privacy constraints*, as well as its integration into any *standard distributed learning framework*.

We perform an extensive experimental evaluation [1] of our algorithm and demonstrate its ability to learn from all available data, while successfully suppressing the effect of corrupted or irrelevant sources. It consistently outperforms both naive approaches of learning on all available data directly or

[1]Institute of Science and Technology, Klosterneuburg, Austria. Correspondence to: Nikola Konstantinov <nkonstan@ist.ac.at>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

[1]Code is available at https://github.com/NikolaKon1994/Robust-Learning-from-Untrusted-Sources

learning on the reference dataset only, *for any amount and any type of data contamination* considered. We also observe its performance to be superior to multiple baseline methods from robust statistics and robust distributed learning.

2. Related work

Learning from multiple sources is a topic relevant for many applications of machine learning and data corruption is a problem acknowledged in some of these areas. In particular, (Bi et al., 2014; Kajino et al., 2012; Awasthi et al., 2017) and references therein consider the problem of label noise in *crowdsourced data*. The non-i.i.d. split of the data on local devices is one of the main characteristics of *federated learning* and the pioneering work of (McMahan et al., 2017) addresses this by occasionally averaging local models to ensure global consistency. Fault tolerance and prevention of sybil attacks in federated learning have been considered by (Smith et al., 2017) and (Fung et al., 2018) respectively. Robustness has also been explored in the context of *multi-view* learning, where data arrives from various feature extractors (Zhao et al., 2017; Xie, 2017; Zhang et al., 2017).

The work closest in spirit to ours is the one of (Qiao & Valiant, 2018), who provide efficient algorithms for learning from batches of data, an ϵ-fraction of which can be malicious. Their focus is different though, as they only study algorithms for learning discrete distributions and explore the regime where each data source provides a small amount of samples. In contrast, we are interested in general supervised learning problems and work in a setting where more data is available per source. (Charikar et al., 2017; Hendrycks et al., 2018) also study learning with a reference dataset as a protection against data corruption, but focus on a single untrusted dataset only and on convex objectives and label noise respectively. There is a vast body of literature focusing on robustness of learning algorithms to corruptions *within* a dataset, e.g. (Tukey, 1960; Huber, 2011; Diakonikolas et al., 2016; Prasad et al., 2018), and on identifying data corruptions at *prediction time*, e.g. (Hendrycks & Gimpel, 2017; Sun & Lampert, 2018). These lines of work are orthogonal to ours, since we consider *multiple training datasets*, some of which are corrupted, and hence a literature review in this direction is beyond the scope of this paper.

Another related area is the one of *robust distributed learning and optimization*. The work of (Feng et al., 2014) develops a method for implementing any robust supervised learning algorithm in a distributed manner. (Feng, 2017) provide lower bounds for the communication complexity of PAC learning in a distributed environment, in the presence of malicious outliers. Fault tolerance and resistance to adversarial behavior of individual nodes in a distributed system have been studied from the point of view of Byzantine-robust distributed optimization, e.g. (Blanchard et al., 2017; Yin et al., 2018; Alistarh et al., 2018a). These works consider arbitrary (even adversarial) behavior of the nodes, however they study the convergence of gradient-based optimization procedures and typically have to assume that at least half of the nodes behave normally. In contrast, we are interested in the generalization performance of empirical risk minimizers and make no assumptions about the number of corrupted sources. The worst-case performance of distributed SGD has also been studied in the context of asynchronous training (De Sa et al., 2015; Alistarh et al., 2018b).

On the methodological level, we borrow techniques from the field of domain adaptation. To measure the difference between data distributions, we use the same integral probability metric as (Mohri & Medina, 2012; Zimin & Lampert, 2017). The problem we study is related to *multi-source domain adaptation*, e.g. (Crammer et al., 2008; Ben-David et al., 2010), and to *multi-task learning* with unlabeled data (Pentina & Lampert, 2017). In particular, our Theorem 1 is similar to a result in (Zhang et al., 2013). We refer to the paragraph after Theorem 1 for a more detailed comparison. However, all these works focus on sharing information between similar domains, in order to obtain better predictors for a target task, while we are interested in applying such techniques for detecting untrustworthy sources of data and improving the robustness of the learning procedure.

A relation between robustness and domain adaptation has been explored in the work of (Mansour & Schain, 2014), who use a property called *algorithmic robustness* to derive generalization bounds for domain adaptation. Another related line of work is the one of (Mansour et al., 2009; Hoffman et al., 2018), who provide guarantees for a classifier learned on data from N domains on any target distribution that is a mixture of the distributions of the sources. Domain adaptation techniques were also used by (Song et al., 2019), for improving the test-time robustness of predictive models to adversarial examples.

3. Robust learning from untrusted sources

Given a *small reference dataset*, we want to leverage additional training data from *multiple untrusted sources* in an optimal way, so that the obtained predictor performs well on a target distribution. A naive approach will be to trust all data, merge it into one dataset and train end-to-end to obtain a predictive model. Such an approach will intuitively be vulnerable to irrelevant or low-quality data provided by some sources. In this section, we design a more *robust* algorithm that instead minimizes a weighted empirical loss.

3.1. Theory

Setup. Let $\mathcal{X}$ be an input space and $\mathcal{Y}$ be an output space. Our theoretical setup covers both the case of classification

($\mathcal{Y} = \{1, 2, \dots, K\}$) and regression ($\mathcal{Y} = \mathbb{R}$). We assume that the learner has access to a *small reference dataset* $S_T := \{(x_{T,1}, y_{T,1}), \dots, (x_{T,m_T}, y_{T,m_T})\}$ of m_T samples drawn i.i.d. from a target distribution $\mathcal{D}_T$ over $\mathcal{X} \times \mathcal{Y}$. In addition, training data is available from N *untrusted data sources*, each of them characterized by its own distribution, $\mathcal{D}_i$, over $\mathcal{X} \times \mathcal{Y}$, possibly different from $\mathcal{D}_T$. We denote the number of samples from source i by m_i. Let the corresponding i.i.d. datasets be $S_i := \{(x_{i,1}, y_{i,1}), \dots, (x_{i,m_i}, y_{i,m_i})\} \overset{i.i.d.}{\sim} \mathcal{D}_i$ for each $i = 1, \dots, N$.

Let $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ be a loss function, bounded by some $M > 0$. For any distribution $\mathbb{P}$ on $\mathcal{X} \times \mathcal{Y}$ and any function $h : \mathcal{X} \to \mathcal{Y}$, denote by

$$\epsilon_{\mathbb{P}}(h) = \mathbb{E}_{(x,y)\sim\mathbb{P}}(L(h(x), y))$$

the expected loss of the predictor h with respect to the distribution $\mathbb{P}$. Let $\epsilon_i(h) = \epsilon_{\mathcal{D}_i}(h)$ be the expected loss of a predictor h on the distribution of the i-th source. Denote by $\hat{\epsilon}_i$ the corresponding empirical counterparts.

Given a hypothesis class $\mathcal{H} \subset \{h : \mathcal{X} \to \mathcal{Y}\}$, our goal is to use all samples from the *sources* to construct a hypothesis with low expected loss on the target distribution $\mathcal{D}_T$. Note that if we also want to use the reference data at training time, we can simply include it as one of the data sources.

Source-specific weights. For a vector of weights $\alpha = (\alpha_1, \dots, \alpha_N)$, such that $\sum_{i=1}^N \alpha_i = 1$ and $\alpha_i \geq 0$ for all i, we define the α-weighted expected risk of a predictor h as:

$$\epsilon_\alpha(h) = \sum_{i=1}^N \alpha_i \epsilon_i(h) = \sum_{i=1}^N \alpha_i \mathbb{E}_{(x,y)\sim\mathcal{D}_i}(L(h(x), y)) \quad (1)$$

and its empirical counterpart as:

$$\hat{\epsilon}_\alpha(h) = \sum_{i=1}^N \alpha_i \hat{\epsilon}_i(h) = \sum_{i=1}^N \frac{\alpha_i}{m_i} \sum_{j=1}^{m_i} L(h(x_{i,j}), y_{i,j}). \quad (2)$$

With $\mathcal{H}$ as our hypothesis class, let $\hat{h}_\alpha = \text{argmin}_{h\in\mathcal{H}} \hat{\epsilon}_\alpha(h)$.

We aim to find weights α, such that the predictor $\hat{h}_\alpha$ performs well on the target task, i.e. such that $\epsilon_T(\hat{h}_\alpha)$ is small.

Evaluating the quality of a source. Intuitively, a good learning algorithm will assign more weight to sources, whose distribution is similar to the target one, and less weight to those that provide different or low-quality data. Although any standard distance measure on the space of distributions could in theory be used to measure such differences, most of them would not provide any guarantees on the performance of the learned classifier. Furthermore, most similarity measures between distributions, e.g. the Kullback-Leibler divergence, are hard to estimate from finite data and overly strict, as they are independent of the learning setup.

We therefore adopt a specific notion of distance that depends on the hypothesis class and allows us to reason about the change in performance of a predictor from $\mathcal{H}$ learned on one distribution, but applied to the other. Following (Mohri & Medina, 2012), we define the *discrepancy* between the distributions $\mathcal{D}_i$ and $\mathcal{D}_T$ with respect to the hypothesis class $\mathcal{H}$ as:

$$d_{\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_T) = \sup_{h\in\mathcal{H}} (|\epsilon_i(h) - \epsilon_T(h)|). \quad (3)$$

Intuitively, the discrepancy between the two distributions is large, if there exists a predictor that performs well on one of them and badly on the other. On the other hand, if all functions in the hypothesis class perform similarly on both, then $\mathcal{D}_i$ and $\mathcal{D}_T$ have low discrepancy.

The following theorem provides a bound on the expected loss on the target distribution of the predictor $\hat{h}_\alpha$, i.e. the minimizer of the α-weighted sum of the empirical losses over the source data.

Theorem 1. *Given the setup above, let $\hat{h}_\alpha = argmin_{h\in\mathcal{H}} \hat{\epsilon}_\alpha(h)$ and $h_T^* = argmin_{h\in\mathcal{H}} \epsilon_T(h)$. For any $\delta > 0$, with probability at least $1 - \delta$ over the data:*

$$\epsilon_T(\hat{h}_\alpha) \leq \epsilon_T(h_T^*) + 4\sum_{i=1}^N \alpha_i \mathcal{R}_i(\mathcal{H}) + 2\sum_{i=1}^N \alpha_i d_{\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_T)$$
$$+ 6\sqrt{\frac{\log\left(\frac{4}{\delta}\right) M^2}{2}} \sqrt{\sum_{i=1}^N \frac{\alpha_i^2}{m_i}}, \quad (4)$$

where, for each source $i = 1, \dots, N$,

$$\mathcal{R}_i(\mathcal{H}) = \mathbb{E}_\sigma \left(\sup_{f\in\mathcal{H}} \left(\frac{1}{m_i} \sum_{j=1}^{m_i} \sigma_{i,j} L(f(x_{i,j}), y_{i,j}) \right) \right)$$

and $\sigma_{i,j}$ are independent Rademacher random variables.

A proof is provided in the supplementary material.

We note that a similar result appears as Theorem 5.2 in the arXiv version (Zhang et al., 2013) of the NIPS paper (Zhang et al., 2012). The authors bound the gap between the weighted empirical loss on the source data of any classifier and its expected loss on the target task, with the additional assumption of a deterministic labeling function for each source. Based on this, they study the asymptotic convergence of domain adaptation algorithms as the sample sizes at all sources go to infinity. In contrast, our theorem compares the performance of the minimizer $\hat{h}_\alpha$ of the α-weighted empirical loss on the target task to the performance of the optimal (but unknown) h_T^* and does not require deterministic labeling functions. Our target application is also different, since we use the bound to design learning algorithms that are robust to corrupted or irrelevant data, given finite amount of samples from each source.

3.2. From bound to algorithm

Algorithm description. To obtain a good predictor for the target task, we would like to choose α, such that $\epsilon_T(\hat{h}_\alpha)$ is

Algorithm 1 Robust learning from untrusted sources

```
Inputs: 1. Loss $L$, hypothesis set $\mathcal{H}$, parameter $\lambda$
        2. Reference dataset $S_T$
        3. Datasets $S_1, \ldots, S_N$ from the $N$ sources
for $i = 1$ to $N$ do {Potentially in parallel}
    Compute $d_{\mathcal{H}}(S_i, S_T)$
end for
Select $\alpha$ by solving (6).
Minimize $\alpha$-weighted loss: $\hat{h}_\alpha = \text{argmin}_{h\in\mathcal{H}} \hat{\epsilon}_\alpha(h)$
Return: $\hat{h}_\alpha$
```

as close as possible to $\epsilon_T(h_T^*)$ (the expected loss of the best hypothesis in $\mathcal{H}$). This suggests selecting the weights by minimizing the right-hand side of (4).

While the Rademacher complexities are functions of both the underlying distribution and the hypothesis class, in practice one usually works with a computable upper bound that is distribution-independent (e.g. using VC dimension). For some common examples of such bounds we refer to the supplementary material, as well as to (Bousquet et al., 2004; Shalev-Shwartz & Ben-David, 2014). In our setting the hypothesis space $\mathcal{H}$ is fixed and therefore these bounds would be identical for all i. Therefore, we expect the $\mathcal{R}_i(\mathcal{H})$ to be of similar order to each other and the impact of α on the second term in the bound to be negligible. We thus concentrate on optimizing the remaining terms.

Because the true discrepancies are unknown, we estimate them from the data by their empirical counterparts:

$$\begin{aligned} d_{\mathcal{H}}(S_i, S_T) &= \sup_{h\in\mathcal{H}} \left(|\hat{\epsilon}_i(h) - \hat{\epsilon}_T(h)|\right) \\ &= \sup_{h\in\mathcal{H}} (|\frac{1}{m_i}\sum_{j=1}^{m_i} L\left(h\left(x_{i,j}\right), y_{i,j}\right) \\ &\quad - \frac{1}{m_T}\sum_{j=1}^{m_T} L\left(h\left(x_{T,j}\right), y_{T,j}\right)|). \end{aligned} \tag{5}$$

In summary, the bound suggests to choose a weighting for the sources by minimizing:

$$\begin{aligned} &\min_{\alpha} \sum_{i=1}^{N} \alpha_i d_{\mathcal{H}}(S_i, S_T) + \lambda \sqrt{\sum_{i=1}^{N} \frac{\alpha_i^2}{m_i}}, \\ &\text{subject to: } \sum_{i=1}^{N} \alpha_i = 1 \text{ and } \alpha_i \geq 0 \text{ for all } i, \end{aligned} \tag{6}$$

where $\lambda > 0$ is a hyperparameter that can be selected by cross-validation on the reference dataset. The algorithm then proceeds to minimize the α-weighted empirical risk over the sources (2), possibly with a regularization term. Pseudocode of the algorithm is given in Algorithm 1.

Discussion. While derived from our theoretical results, the minimization procedure for selecting the weights also has an intuitive interpretation. Note that the first term in (6) is small whenever large weights are paired with small discrepancies and hence encourages trusting sources that provide data similar to the reference target sample. The second term is small whenever the weights are distributed proportionally to the number of samples per source. Thus, it acts as a form of regularization, by encouraging the usage of information from as many sources as possible.

The hyperparamater λ controls a trade-off between exploiting similar tasks and leveraging information from all sources. As $\lambda \to \infty$, all tasks are assigned weights proportional to the amount of training samples they provide and the model minimizes the empirical risk over all the data, regardless of the quality of the samples. In contrast, as $\lambda \to 0$, the model becomes more sensitive to differences between the source data and the clean reference set, until all weight is assigned to the source closest to the target domain. Assuming that the reference set is included as one of the data sources, these extremes correspond to the naive approach of trusting all sources and training on a merged dataset and not trusting any of them and training on the initial clean data only. In our experiments in Section 4 we will see that there is a better operating point between those two extremes. It naturally depends on the actual quality of the available data and our algorithm identifies it successfully.

3.3. Learning from private or decentralized data

The described algorithm is straightforward to implement on top of any standard learning procedure, when the data from all N sources is directly available to the learner. We now discuss how we can learn robustly in cases where the sources cannot fully reveal their data. There are many applications where such a situation can arise. For example, this can be due to privacy reasons in the case of medical and biological data or to communication costs and storage limitations in the case of distributed learning (McMahan et al., 2017).

Here we focus on ways to compute the discrepancies under such constraints. Once this is done, the vector α can be computed easily and then any standard distributed training procedure, e.g. (Dean et al., 2012; McMahan et al., 2017), can be used to obtain the α-weighted empirical loss minimizer. Standard approaches in distributed learning only require the exchange of gradients of minibatches with respect to the current state of the model between the data sources and the central server, so in particular the actual local datasets are never observed by the learner. In cases when the gradients may reveal sensitive information about the data, secure aggregation (Bonawitz et al., 2017) or other privacy-preserving distributed learning methods (Shokri & Shmatikov, 2015) can be used on top to ensure privacy.

We distinguish two cases, depending on whether the reference dataset can be shared with the sources.

Case 1: the reference dataset is available to all nodes. If the reference dataset can be shared with the sources without privacy and communication complications, the discrepancies can be estimated *locally on every source, in parallel*. If necessary, the computational protocol can be executed via a trusted computation method (Chen et al., 2009), for example by using Software Guard Extensions (SGX extensions) (McKeen et al., 2016), to ensure the correctness of the procedure. The discrepancies alone can then be sent to the learner and the algorithm proceeds as described above. This approach ensures the privacy of the local datasets and allows for all discrepancies to be computed in parallel.

Case 2: the reference dataset can not be shared. In this case the learner can still compute the empirical discrepancies without observing the data from the sources directly, by using a gradient-based optimization procedure. This is because the function inside the supremum in (5) decomposes into a term depending only on the reference dataset and a term depending only on the data of the source. Therefore, each discrepancy can be estimated by using a sequence of queries to the source about the gradient of a minibatch from its data with respect to a current candidate for the predictor achieving the supremum.

4. Experiments

4.1. Method and baselines

We perform two large sets of experiments, following the setup considered in our paper. We train our algorithm on the data from all sources, including the reference dataset. The hyperparameter λ is selected by 5-fold cross-validation *on the trusted data*. The prediction tasks we consider here are binary classification problems with the $0/1$-loss, so we compute the empirical discrepancies by approximately solving the optimization problem (5) as follows. Given the two datasets S_i and S_T, the binary labels of one of them are flipped. The optimization can then be reduced to an empirical risk minimization problem that we solve using a standard convex relaxation approach. We refer to the supplementary material for a more formal description.

We compare the performance of our algorithm to the two naive approaches: training on the reference dataset only (corresponding to $\lambda = 0$ in our algorithm; denoted as *"Reference only"* in the plots and tables) and merging the sources and training on all the data (corresponding to $\lambda \to \infty$; referred to as *"All data"* in the plots and tables). All three methods use linear predictors and are trained by regularized logistic regression. The regularization parameter is always selected by 5-fold cross-validation on the reference data. The learned models are then evaluated on held out test data.

Our aim is to test whether the proposed algorithm successfully leverages information from the sources, while being robust to various perturbations in the distributions of the local datasets, and whether exploiting the multi-source structure of the data gives any improvement over the two standard learning procedures. We also compare the performance of our algorithm to the following robust learning baselines.

Robust aggregation of local models. We consider two recently proposed approaches for robust distributed learning. Following (Feng et al., 2014), one baseline learns a separate linear model based on each of the source datasets. The final linear predictor is then constructed as the geometric median of these locally learned weight vectors. Another baseline, inspired by (Yin et al., 2018), takes the component-wise median instead. Thirdly, based on the locally learned models all N estimates for the probability that a test point belongs to a certain class are computed and the final prediction for the label of that point is obtained by taking the median of these probabilities and thresholding it (referred to as "Median of probs" in the plots and tables). All these baselines aim at learning a robust ensemble of local models.

Robust logistic regression. We use the method of (Pregibon, 1982), based on the minimization of a Huber-type modification of the logistic loss. Specifically, the method minimizes the following robust loss function, instead of the classic logistic loss:

$$L(\boldsymbol{w}, \boldsymbol{x}, y) = \begin{cases} \log(1 + e^{-y\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}}), \text{ if } \log(1 + e^{-y\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}}) \leq c \\ 2\sqrt{c \log(1 + e^{-y\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}})} - c, \text{ otherwise} \end{cases}$$

In our experiments, we use the recommended threshold value of $c = 1.345^2$, under which the estimate of the linear predictor has been shown to achieve a 95% asymptotic relative efficiency (Pregibon, 1982). We also include a regularization term here and learn the regularization parameter by 5-fold cross-validation on the reference data. This baseline is an example of learning robustly on the whole dataset.

Batch normalization. Inspired by the success of *batch normalization* in deep learning (Ioffe & Szegedy, 2015), we compute the mean and standard deviation of the data at each source separately. We then subtract from each data point the mean and divide by the standard deviation of its corresponding dataset. We do the same for the reference data. We then merge all data together and train a logistic regression model with a regularization term. Finally, at test time every input is preprocessed by subtracting the mean and dividing by the standard deviation of the reference dataset, before applying the classifier. This approach aims at increasing robustness to source-specific biases.

4.2. Amazon Products data

Our first set of experiments is on the "Multitask dataset of product reviews"[2] (Pentina & Lampert, 2017), containing

[2]http://cvml.ist.ac.at/productreviews/

customer reviews for 957 Amazon products from the "Amazon product data" (McAuley et al., 2015a;b), together with a binary label indicating whether each review is positive or negative. All reviews in the data set are represented via 25-dimensional feature vectors, obtained by computing a GloVe word embedding (Pennington et al., 2014) and applying the sentence embedding procedure of (Arora et al., 2017). We treat the classification of a review as positive or negative as a separate prediction task for each of the products, resulting in a total of 957 input-output distributions.

As a first, illustrative, experiment, we chose 20 books and 20 other, purposely different, products (e.g. USB drives, mobile apps, meal replacement products). For simplicity, we refer to these additional products as "non-books". Intuitively, when learning to classify book reviews and given access to reviews from both some books and some non-books, a good learning algorithm will be able to leverage all this information, while being robust to the potentially misleading data coming from the less relevant products.

We randomly sample one of the books and 300 positive and 300 negative reviews for it. Out of those, 100 randomly selected reviews are made available to the learner as a reference dataset. The 500 remaining reviews from the product are used for testing. For a given value of $n \in \{0, 1, \dots, 10\}$ the learner also has access to 100 labeled reviews from each of $10 - n$ other randomly selected books and from each of n randomly selected non-books. Our algorithm, as well as all baselines, are trained on this available data and the learned predictors are evaluated on the test set for the target product. For each n, we repeat this experiment 1000 times.

The results are plotted in Figure 1. The x-axis corresponds to the number n of non-books and the y-axis gives the average classification error. The error bars correspond to the standard errors of the mean estimates. We see that our method (green) performs uniformly better than the naive approaches of training on the reference dataset from the target product only (red) and training by merging all data together (blue). When reviews from many books are available, our algorithm is able to use this additional information even better than the model learned on all data. As the proportion of non-books increases, the performance of the second approach degrades, confirming the intuition that the reviews for the non-books provide less useful information for the target task. On the other hand, our algorithm successfully incorporates the information from the useful sources only, converging to the performance of the model learned on the reference data as all additional sources become non-books.

Our algorithm also outperforms all baselines. The batch normalization approach appears to reduce the effect of irrelevant sources, but its performance degrades as $n \to 10$. The median-based approaches perform reasonably when at most half of the sources are non-books, but eventually become worse than the other methods. The component-wise median and the robust loss baselines were excluded from the plot for clarity, as they performed uniformly worse than the other baselines, ranging in average classification error from 0.338 to 0.375 and from 0.348 to 0.372 respectively. Note that the robust loss function of (Pregibon, 1982) is non-convex, so the poor performance of this baseline is presumably due to failure of the gradient descent optimization procedure to converge to a good local minimum.

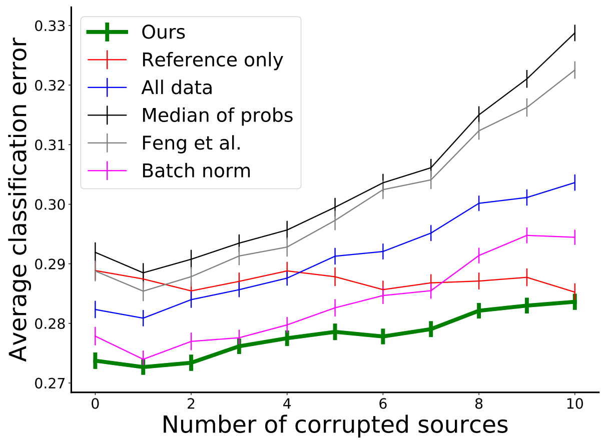
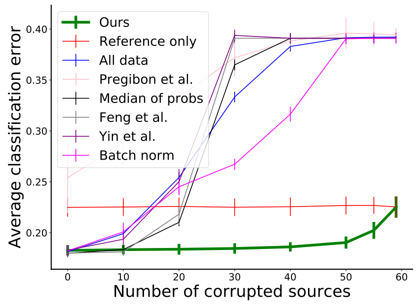


Figure 1: Results from the experiments on 20 books and 20 other products from the "Multitask dataset of product reviews". The x-axis gives the number n of non-books in an experiment and the y-axis - the mean classification error. Error bars give the standard error of the estimates.

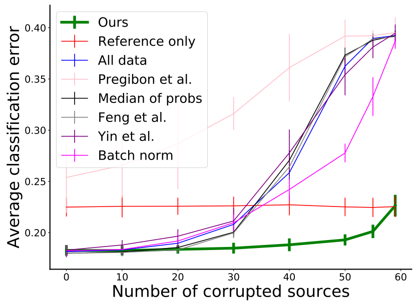
Table 1: Results from the experiment on all 957 products.

Algorithm	Mean classification error
Ours	$\mathbf{0.289 \pm 0.0016}$
Reference only	0.301 ± 0.0019
All data	0.312 ± 0.0017
Median of probs.	0.325 ± 0.0021
Geom.median (Feng et al., 2014)	0.329 ± 0.0021
Comp.median (Yin et al., 2018)	0.329 ± 0.0021
Robust loss (Pregibon, 1982)	0.353 ± 0.0021
Batch norm	0.298 ± 0.0016

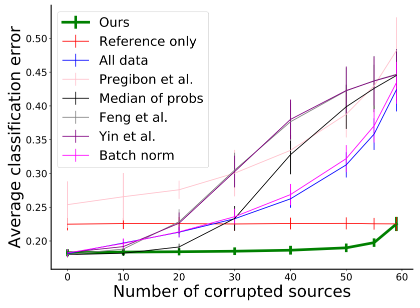
Additionally, we performed an experiment on the set of all 957 products. With every product as a prediction task, we randomly selected 100 reviews from it as a reference dataset, leaving 500 for testing. An additional set of 100 labeled reviews were available from every other product. The algorithms were trained on all available data and evaluated on the test set. The average classification errors achieved by the algorithms are presented in Table 1, together with the standard errors of those estimates. We see in particular that our algorithm successfully uses the information from multiple sources to achieve the best overall performance.



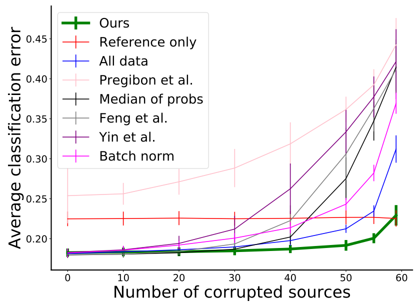
(a) Label bias



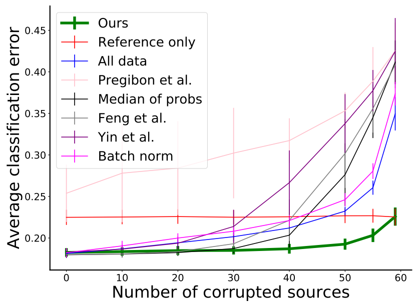
(b) Shuffled labels



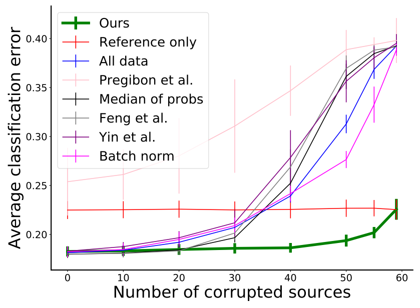
(c) Shuffled features



(d) Blurred images



(e) Dead pixels



(f) RGB channels swapped

Figure 2: Results for the attribute "black" from the Animals with Attributes 2 dataset. Each plot corresponds to a different contamination type. The x-axis gives the number n of corrupted sources and the y-axis gives the average classification error of the algorithms, achieved over 100 different runs. Error bars correspond to the standard deviation around those means.

4.3. Animals with Attributes 2

The Animals with Attributes 2 dataset (Xian et al., 2018) contains 37322 images of 50 animal classes. The classes are aligned to 85 binary attributes, e.g. color, habitat and others, via a class-attribute binary matrix, indicating whether an animal possesses each feature. This results in a total of 85 different binary prediction tasks of identifying whether an animal on a given image possesses a certain attribute or not.

Feature representations of the images are obtained via the following procedure. We use a ResNet50 network (He et al., 2016), pretrained [3] on ImageNet (Russakovsky et al., 2015), to obtain feature representations of the ImageNet data and reduce their dimension to 100 by PCA. Finally, for each image in the Animals with Attributes 2 dataset, we compute the ResNet50 feature representation and apply the PCA projection pre-learned on ImageNet.

We perform an independent set of experiments for each attribute and for various types and levels of corruption of the data sources. In each run, we randomly split the data into 60 groups of 500 images, with the remaining 7322 images left out for testing. One of the groups is selected at random as the clean reference dataset available to the learner. The remaining 59 groups correspond to the data sources, some of which provide low-quality or corrupted data. We consider six different types of corruptions. Three act on the labels or the feature representations directly and the next three are synthetic modifications of the images themselves. In the second case, the corresponding images are manipulated before the feature representations are extracted.

- Label bias: The labels of all (corrupted) samples are switched to class 1.
- Shuffled labels: The labels of all samples are shuffled randomly, separately in each corrupted source.
- Shuffled features: Given a permutation of the indexes between 1 and 100, the features of all samples are shuffled according to it.
- Blurred images: Each image is blurred by filtering with a Gaussian kernel with standard deviation $\sigma = 6$.
- Dead pixels: In each image a random 30% of the pixels are set to pure black or white.
- RGB channels swapped: The values in the red and the blue color channels of each image are swapped.

Given an attribute, a type of corruption and a value of $n \in \{0, 10, 20, 30, 40, 50, 55, 59\}$, the data is split randomly, as described above, and the samples of n randomly

[3] We use a pretrained model from the TensorNets package, https://github.com/taehoonlee/tensornets.

Baseline \ n	$n=0$	$n=10$	$n=20$	$n=30$	$n=40$	$n=50$	$n=55$	$n=59$
Reference only	84/1/0	505/5/0	497/13/0	487/23/0	475/35/0	442/68/0	325/185/0	0/510/0
All data	0/85/0	115/395/0	267/243/0	370/140/0	438/72/0	468/42/0	479/31/0	484/26/0
Median of probs.	9/76/0	47/463/0	172/338/0	336/174/0	469/41/0	504/6/0	502/8/0	499/11/0
Geom.median (Feng et al., 2014)	8/77/0	32/478/0	110/400/0	338/172/0	457/53/0	504/6/0	502/8/0	497/13/0
Comp.median (Yin et al., 2018)	14/71/0	179/331/0	390/120/0	432/78/0	472/38/0	502/8/0	503/7/0	497/13/0
Robust loss (Pregibon, 1982)	55/30/0	308/202/0	361/149/0	416/94/0	437/73/0	455/55/0	470/40/0	485/25/0
Batch norm	0/85/0	107/403/0	317/193/0	416/94/0	446/63/1	478/32/0	487/23/0	482/28/0

Table 2: Summary of the results from the Animals with Attributes 2 experiments, over all 85 prediction tasks and all 6 types of corruption. Given a number of corrupted sources n (columns) and a baseline (rows), we report values in the form A/B/C, where A is the number of times that our method performed significantly better than the corresponding baseline, B is the number of times it performed equally well and C is the number of times it performed significantly worse, summed over the various types of corruptions and all attributes. More details are provided in the main body of the text.

chosen sources are corrupted. Our algorithm, as well as all baselines, then learn a model based on the resulting data and the performance of the obtained predictors is evaluated on the test data. For any combination of target attribute, corruption strategy and value of n, the experiment is repeated 100 times with a different random seed to obtain error estimates.

The results for the first attribute from the Animals with Attributes 2 data ("black") are given in Figure 2. Each plot corresponds to a different type of contamination. The x-axis gives the number of sources providing corrupted data and the y-axis corresponds to the average error that an algorithm achieved on the test set, over the 100 runs for each experimental setup. The error bars give the standard deviation around this average.

Our algorithm (green) performs at least as well as or strictly better than all baselines, for *any* type of corruption and *any* proportion of corrupted sources. When all sources provide clean data, the performance of our method matches the one of the classic regularized logistic regression approach on i.i.d. data (blue). As the number of corrupted sources increases, the performance of all baselines gradually degrades, while our algorithm is able to leverage the remaining clean data and suppress the effect of the corruptions. The median-based baselines perform reasonably when less than half of the sources are corrupted, but fail for larger proportions. The robust logistic regression baseline performs poorly, again likely due to the non-convexity of the loss function. As all sources become unreliable, our method performs as well as the approach of learning from the reference dataset only, which is indeed optimal since all other data is corrupted.

We summarize the results from all attributes in Table 2. For any number of corrupted sources n (columns), we compare our method to the performance of each baseline (rows). We report values in the form A/B/C, where A is the number of times that our method performed significantly better than the corresponding baseline, B is the number of times it performed equally well and C is the number of times it performed significantly worse, summed over the various types of corruptions and all attributes. For a fixed type of corruption and attribute, we say that one method performs significantly better than another over the set of 100 runs with this setup, if the difference in the average performance of the two models is larger than the sum of the standard deviations around those means (that is, if the error bars, as in Figure 2, do not intersect).

The results in Table 2 show that our method *performs significantly better than all baselines for many types of corruption and many values of* n, especially for high levels of contamination, while *essentially never performing significantly worse than any baseline*. Tables with a more detailed breakdown, depending on the type of corruption, as well as results for lower levels of contamination per source, are deferred to the supplementary material.

5. Conclusion

We introduce an algorithm for learning from data provided by multiple untrusted sources. It incorporates information from all of them, while being robust to arbitrary corruptions and manipulations of the data. By making use of the grouped structure of the task and a reference dataset, the method is able to successfully learn even if more than half of the available data is corrupted or uninformative. Our method is theoretically justified and easy to implement, even in cases when the data is decentralized and/or private. We demonstrated its effectiveness through two sets of extensive experiments, showing its superior performance to all baselines, for various levels and types of corruption.

In our experiments we observed that a relatively small clean dataset was enough to protect the learning from the effects of corrupted data. Quantifying the trade-off between the size of the reference dataset and the gains of our algorithm in terms of achieved test-time performance is thus an interesting and promising direction for future work.

Acknowledgements

We thank Dan Alistarh for helpful suggestions regarding the discussion in Section 3.3. This work was in parts funded by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement no 308036. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 665385.

References

Alistarh, D., Allen-Zhu, Z., and Li, J. Byzantine stochastic gradient descent. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2018a.

Alistarh, D., De Sa, C., and Konstantinov, N. The convergence of stochastic gradient descent in asynchronous shared memory. In *ACM Symposium on Principles of Distributed Computing*, PODC, 2018b.

Arora, S., Liang, Y., and Ma, T. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations (ICLR)*, 2017.

Awasthi, P., Blum, A., Haghtalab, N., and Mansour, Y. Efficient PAC learning from the crowd. *Conference on Computational Learning Theory (COLT)*, 2017.

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175, 2010.

Bi, W., Wang, L., Kwok, J. T., and Tu, Z. Learning to predict from crowdsourced data. In *UAI*, pp. 82–91, 2014.

Biggio, B., Nelson, B., and Laskov, P. Poisoning attacks against support vector machines. In *International Conference on Machine Learing (ICML)*, 2012.

Blanchard, P., Guerraoui, R., Stainer, J., et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Conference on Neural Information Processing Systems (NIPS)*, 2017.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017.

Bousquet, O., Boucheron, S., and Lugosi, G. Introduction to statistical learning theory. In *Advanced lectures on machine learning*, pp. 169–207. Springer, 2004.

Charikar, M., Steinhardt, J., and Valiant, G. Learning from untrusted data. In *ACM SIGACT Symposium on Theory of Computing*, 2017.

Chen, L., Mitchell, C. J., and Martin, A. P. (eds.). *Trusted Computing, Second International Conference, Trust 2009, Oxford, UK, April 6-8, 2009, Proceedings*, volume 5471 of *Lecture Notes in Computer Science*, 2009.

Crammer, K., Kearns, M., and Wortman, J. Learning from multiple sources. *Journal of Machine Learning Research (JMLR)*, 9(Aug):1757–1774, 2008.

De Sa, C. M., Zhang, C., Olukotun, K., Ré, C., and Ré, C. Taming the wild: A unified analysis of hogwild-style algorithms. In *Conference on Neural Information Processing Systems (NIPS)*. 2015.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Senior, A., Tucker, P., Yang, K., Le, Q. V., et al. Large scale distributed deep networks. In *Conference on Neural Information Processing Systems (NIPS)*, 2012.

Diakonikolas, I., Kamath, G., Kane, D. M., Li, J., Moitra, A., and Stewart, A. Robust estimators in high dimensions without the computational intractability. In *Foundations of Computer Science (FOCS)*, pp. 655–664. IEEE, 2016.

Feng, J. On fundamental limits of robust learning. *arXiv preprint arXiv:1703.10444*, 2017.

Feng, J., Xu, H., and Mannor, S. Distributed robust learning. *arXiv preprint arXiv:1409.5937*, 2014.

Fung, C., Yoon, C. J., and Beschastnikh, I. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866*, 2018.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Hendrycks, D. and Gimpel, K. Improving the generalization of adversarial training with domain adaptation. In *International Conference on Learning Representations (ICLR)*, 2017.

Hendrycks, D., Mazeika, M., Wilson, D., and Gimpel, K. Using trusted data to train deep networks on labels corrupted by severe noise. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

Hoffman, J., Mohri, M., and Zhang, N. Algorithms and theory for multiple-source adaptation. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

Huber, P. J. *Robust statistics*. Springer, 2011.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learing (ICML)*, 2015.

Kajino, H., Tsuboi, Y., and Kashima, H. A convex formulation for learning from crowds. *Transactions of the Japanese Society for Artificial Intelligence*, 27(3):133–142, 2012.

Mansour, Y. and Schain, M. Robust domain adaptation. *Annals of Mathematics and Artificial Intelligence*, 71(4): 365–380, 2014.

Mansour, Y., Mohri, M., and Rostamizadeh, A. Domain adaptation with multiple sources. In *Conference on Neural Information Processing Systems (NIPS)*, pp. 1041–1048, 2009.

McAuley, J., Pandey, R., and Leskovec, J. Inferring networks of substitutable and complementary products. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015a.

McAuley, J., Targett, C., Shi, Q., and Van Den Hengel, A. Image-based recommendations on styles and substitutes. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, 2015b.

McKeen, F., Alexandrovich, I., Anati, I., Caspi, D., Johnson, S., Leslie-Hurd, R., and Rozas, C. Intel® software guard extensions (intel® sgx) support for dynamic memory management inside an enclave. In *Proceedings of the Hardware and Architectural Support for Security and Privacy 2016*, pp. 10, 2016.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Conference on Uncertainty in Artificial Intelligence (AISTATS)*, 2017.

Mohri, M. and Medina, A. M. New analysis and algorithm for learning with drifting distributions. In *International Conference on Algorithmic Learning Theory (ALT)*, 2012.

Pennington, J., Socher, R., and Manning, C. Glove: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

Pentina, A. and Lampert, C. H. Multi-task learning with labeled and unlabeled tasks. In *International Conference on Machine Learing (ICML)*, 2017.

Prasad, A., Suggala, A. S., Balakrishnan, S., and Ravikumar, P. Robust estimation via robust gradient estimation. *arXiv preprint arXiv:1802.06485*, 2018.

Pregibon, D. Resistant fits for some commonly used logistic models with medical applications. *Biometrics*, pp. 485–498, 1982.

Qiao, M. and Valiant, G. Learning discrete distributions from untrusted batches. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 94. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

Shalev-Shwartz, S. and Ben-David, S. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

Shokri, R. and Shmatikov, V. Privacy-preserving deep learning. In *ACM SIGSAC conference on computer and communications security*, 2015.

Smith, V., Chiang, C.-K., Sanjabi, M., and Talwalkar, A. S. Federated multi-task learning. In *Conference on Neural Information Processing Systems (NIPS)*, 2017.

Song, C., He, K., Wang, L., and Hopcroft, J. E. Improving the generalization of adversarial training with domain adaptation. In *International Conference on Learning Representations (ICLR)*, 2019.

Sun, R. and Lampert, C. H. KS(conf): A light-weight test if a convnet operates outside of its specifications. In *German Conference on Pattern Recognition (GCPR)*, 2018.

Tukey, J. W. A survey of sampling from contaminated distributions. *Contributions to probability and statistics*, pp. 448–485, 1960.

Wahlsten, D., Metten, P., Phillips, T. J., Boehm, S. L., Burkhart-Kasch, S., Dorow, J., Doerksen, S., Downing, C., Fogarty, J., Rodd-Henricks, K., et al. Different data from different labs: lessons from studies of gene–environment interaction. *Journal of neurobiology*, 54(1): 283–311, 2003.

Wais, P., Lingamneni, S., Cook, D., Fennell, J., Goldenberg, B., Lubarov, D., Marin, D., and Simons, H. Towards building a high-quality workforce with mechanical turk. In *NIPS Workshop on Computational Social Science and the Wisdom of Crowds*, 2010.

Xian, Y., Lampert, C. H., Schiele, B., and Akata, Z. Zero-shot learning-a comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 2018.

Xie, T. *Robust Learning from Multiple Information Sources*. PhD thesis, University of Michigan, 2017.

Yin, D., Chen, Y., Kannan, R., and Bartlett, P. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learing (ICML)*, 2018.

Zhang, C., Zhang, L., and Ye, J. Generalization bounds for domain adaptation. In *Conference on Neural Information Processing Systems (NIPS)*, 2012.

Zhang, C., Zhang, L., and Ye, J. Generalization bounds for domain adaptation. *arXiv preprint arXiv:1304.1574*, 2013.

Zhang, G., Iwata, T., and Kashima, H. Robust multi-view topic modeling by incorporating detecting anomalies. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2017.

Zhao, J., Xie, X., Xu, X., and Sun, S. Multi-view learning overview: Recent progress and new challenges. *Information Fusion*, 38:43–54, 2017.

Zimin, A. and Lampert, C. H. Learning theory for conditional risk minimization. In *Conference on Uncertainty in Artificial Intelligence (AISTATS)*, 2017.

A. Proof of Theorem 1

First we bound $|\hat{\epsilon}_\alpha(h) - \epsilon_\alpha(h)|$ with high probability and uniformly over $\mathcal{H}$. We adapt the classical proofs of generalization bounds in terms of the Rademacher complexity of a hypothesis class, e.g. (Bousquet et al., 2004).

Proposition 1. *Given the setup and assumptions described above, for any $\delta > 0$ with probability at least $1-\delta$ over the data, for any function $h \in \mathcal{H}$:*

$$\begin{aligned}|\epsilon_\alpha(h) - \hat{\epsilon}_\alpha(h)| &\leq 2\sum_{i=1}^{N} \alpha_i \mathcal{R}_i(\mathcal{H}) \\ &\quad + 3\sqrt{\frac{\log\left(\frac{4}{\delta}\right)M^2}{2}}\sqrt{\sum_{i=1}^{N}\frac{\alpha_i^2}{m_i}},\end{aligned} \tag{7}$$

where for each $i = 1, 2, \dots, N$:

$$\mathcal{R}_i(\mathcal{H}) = \mathbb{E}_\sigma\left(\sup_{f\in\mathcal{H}}\left(\frac{1}{m_i}\sum_{j=1}^{m_i}\sigma_{i,j}L(f(x_{i,j}), y_{i,j})\right)\right), \tag{8}$$

and where $\sigma_{i,j}$ are independent Rademacher random variables.

Proof. Write:

$$\epsilon_\alpha(h) \leq \hat{\epsilon}_\alpha(h) + \sup_{f\in\mathcal{H}}\left(\epsilon_\alpha(f) - \hat{\epsilon}_\alpha(f)\right) \tag{9}$$

To link the second term to its expectation, we prove the following:

Lemma 1. *Define the function $\phi : (\mathcal{X}\times\mathcal{Y})^m \to \mathbb{R}$ by:*

$$\phi(\{x_{1,1}, y_{1,1}\}, \dots, \{x_{N,m_N}, y_{N,m_N}\}) = \sup_{f\in\mathcal{H}}\left(\epsilon_\alpha(f) - \hat{\epsilon}_\alpha(f)\right).$$

Denote for brevity $z_{i,j} = \{x_{i,j}, y_{i,j}\}$. Then, for any $i \in \{1, 2, \dots, N\}, j \in \{1, 2, \dots, m_i\}$:

$$\begin{aligned}\sup_{z_{1,1},\dots,z_{N,m_N},z'_{i,j}} &|\phi(z_{1,1},\dots,z_{i,j},\dots,z_{N,m_N}) \\ &- \phi\left(z_{1,1},\dots,z'_{i,j},\dots,z_{N,m_N}\right)| \leq \frac{\alpha_i}{m_i}M\end{aligned} \tag{10}$$

Proof. Fix any i,j and any $z_{1,1},\dots,z_{N,m_N}, z'_{i,j}$. Denote the α-weighted empirical average of the loss with respect to the sample $z_{1,1},\dots,z'_{i,j},\dots,z_{N,m_N}$ by ϵ'_α. Then we have that:

$$\begin{aligned}&|\phi(\dots,z_{i,j},\dots) - \phi(\dots,z'_{i,j},\dots)| \\ &= |\sup_{f\in\mathcal{H}}\left(\epsilon_\alpha(f) - \hat{\epsilon}_\alpha(f)\right) \\ &\quad - \sup_{f\in\mathcal{H}}(\epsilon_\alpha(f) - \hat{\epsilon}'_\alpha(f))| \\ &\leq |\sup_{f\in\mathcal{H}}(\hat{\epsilon}'_\alpha(f) - \hat{\epsilon}_\alpha(f))| \\ &= \frac{\alpha_i}{m_i}|\sup_{f\in\mathcal{H}}\left(L\left(f(x'_{i,j}), y'_{i,j}\right) - L\left(f(x_{i,j}), y_{i,j}\right)\right)| \\ &\leq \frac{\alpha_i}{m_i}M\end{aligned}$$

Note: the inequality we used above holds for bounded functions inside the supremum. $\square$

Let S denote a random sample of size m drawn from a distribution as the one generating out data (i.e. m_i samples from $\mathcal{D}_i$ for each i). Now, using Lemma 1, McDiarmid's inequality gives:

$$\begin{aligned}\mathbb{P}\left(\phi(S) - \mathbb{E}(\phi(S)) \geq t\right) &\leq \exp\left(-\frac{2t^2}{\sum_{i=1}^{N}\sum_{j=1}^{m_i}\frac{\alpha_i^2}{m_i^2}M^2}\right) \\ &= \exp\left(-\frac{2t^2}{M^2\sum_{i=1}^{N}\frac{\alpha_i^2}{m_i}}\right)\end{aligned}$$

For any $\delta > 0$, setting the right-hand side above to be $\delta/4$ and using (9), we obtain that with probability at least $1-\delta/4$:

$$\begin{aligned}\epsilon_\alpha(h) &\leq \hat{\epsilon}_\alpha(h) + \mathbb{E}_S\left(\sup_{f\in\mathcal{H}}\left(\epsilon_\alpha(f) - \hat{\epsilon}_\alpha(f)\right)\right) \\ &\quad + \sqrt{\frac{\log\left(\frac{4}{\delta}\right)M^2}{2}}\sqrt{\sum_{i=1}^{N}\frac{\alpha_i^2}{m_i}}\end{aligned} \tag{11}$$

To deal with the expected loss inside the second term, introduce a ghost sample (denoted by S'), drawn from the same distributions as our original sample (denoted by S). Denoting the weighted empirical loss with respect to the ghost sample by ϵ'_α, $\beta_i = m_i/m$ for all i, and using the convexity of the supremum, we obtain:

$$\begin{aligned}&\mathbb{E}_S\left(\sup_{f\in\mathcal{H}}\left(\epsilon_\alpha(f) - \hat{\epsilon}_\alpha(f)\right)\right) \\ &= \mathbb{E}_S\left(\sup_{f\in\mathcal{H}}\left(\mathbb{E}_{S'}\left(\hat{\epsilon}'_\alpha(f)\right) - \hat{\epsilon}_\alpha(f)\right)\right) \\ &\leq \mathbb{E}_{S,S'}\left(\sup_{f\in\mathcal{H}}\left(\hat{\epsilon}'_\alpha(f) - \hat{\epsilon}_\alpha(f)\right)\right) \\ &= \mathbb{E}_{S,S'}\left(\sup_{f\in\mathcal{H}}\left(\frac{1}{m}\sum_{i=1}^{N}\sum_{j=1}^{m_i}\frac{\alpha_i}{\beta_i}\left(L(f(x'_{i,j}), y'_{i,j})\right.\right.\right. \\ &\qquad \left.\left.\left. - L(f(x_{i,j}), y_{i,j})\right)\right)\right)\end{aligned}$$

Introducing m independent Rademacher random variables and noting that $L(f(x'), y') - L(f(x), y)$ and

$\sigma\left(L(f(x^{'}),y^{'})-L(f(x),y)\right)$ have the same distribution, as long as (x,y) and $(x^{'},y^{'})$ have the same distribution:

$$\begin{aligned}
&\mathbb{E}_S\left(\sup_{f\in\mathcal{H}}\left(\epsilon_\alpha(f)-\hat{\epsilon}_\alpha(f)\right)\right)\\
&\quad\le \mathbb{E}_{S,S',\sigma}\left(\sup_{f\in\mathcal{H}}\left(\frac{1}{m}\sum_{i=1}^{N}\sum_{j=1}^{m_i}\frac{\alpha_i}{\beta_i}\sigma_{i,j}\left(L(f(x^{'}_{i,j}),y^{'}_{i,j})\right.\right.\right.\\
&\qquad\left.\left.\left.-L(f(x_{i,j}),y_{i,j})\right)\right)\right)\\
&\quad\le \mathbb{E}_{S',\sigma}\left(\sup_{f\in\mathcal{H}}\left(\frac{1}{m}\sum_{i=1}^{N}\sum_{j=1}^{m_i}\frac{\alpha_i}{\beta_i}\sigma_{i,j}L(f(x^{'}_{i,j}),y^{'}_{i,j})\right)\right)\\
&\quad+\mathbb{E}_{S,\sigma}\left(\sup_{f\in\mathcal{H}}\left(\frac{1}{m}\sum_{i=1}^{N}\sum_{j=1}^{m_i}\frac{\alpha_i}{\beta_i}\left(-\sigma_{i,j}\right)L(f(x_{i,j}),y_{i,j})\right)\right)\\
&\quad=2\mathbb{E}_{S,\sigma}\left(\sup_{f\in\mathcal{H}}\left(\frac{1}{m}\sum_{i=1}^{N}\sum_{j=1}^{m_i}\frac{\alpha_i}{\beta_i}\sigma_{i,j}L(f(x_{i,j}),y_{i,j})\right)\right)
\end{aligned}$$

We can now link the last term to the empirical analog of the Rademacher complexity, by using the McDiarmid Inequality (with an observation similar to Lemma 1). Putting this together, we obtain that for any $\delta>0$ with probability at least $1-\delta/2$:

$$\begin{aligned}
&\epsilon_\alpha(h)\le\hat{\epsilon}_\alpha(h)\\
&\quad+2\mathbb{E}_\sigma\left(\sup_{f\in\mathcal{H}}\left(\frac{1}{m}\sum_{i=1}^{N}\sum_{j=1}^{m_i}\frac{\alpha_i}{\beta_i}\sigma_{i,j}L(f(x_{i,j}),y_{i,j})\right)\right)\\
&\quad+3\sqrt{\frac{\log\left(\frac{4}{\delta}\right)M^2}{2}}\sqrt{\sum_{i=1}^{N}\frac{\alpha_i^2}{m_i}}
\end{aligned}\tag{12}$$

Finally, note that:

$$\begin{aligned}
&\mathbb{E}_\sigma\left(\sup_{f\in\mathcal{H}}\left(\frac{1}{m}\sum_{i=1}^{N}\sum_{j=1}^{m_i}\frac{\alpha_i}{\beta_i}\sigma_{i,j}L(f(x_{i,j}),y_{i,j})\right)\right)\\
&\quad\le\mathbb{E}_\sigma\left(\sum_{i=1}^{N}\alpha_i\sup_{f\in\mathcal{H}}\left(\frac{1}{m_i}\sum_{j=1}^{m_i}\sigma_{i,j}L(f(x_{i,j}),y_{i,j})\right)\right)\\
&\quad=\sum_{i=1}^{N}\alpha_i\mathbb{E}_\sigma\left(\sup_{f\in\mathcal{H}}\left(\frac{1}{m_i}\sum_{j=1}^{m_i}\sigma_{i,j}L(f(x_{i,j}),y_{i,j})\right)\right)\\
&\quad=\sum_{i=1}^{N}\alpha_i\mathcal{R}_i\left(\mathcal{H}\right)
\end{aligned}$$

Bounding $\hat{\epsilon}_\alpha(h)-\epsilon_\alpha(h)$ with the same quantity and with probability at least $1-\delta/2$ follows by a similar argument. The result then follows by applying the union bound. □

Now we show:

Theorem 1. *Given the setup above, let* $\hat{h}_\alpha=\text{argmin}_{h\in\mathcal{H}}\hat{\epsilon}_\alpha(h)$ *and* $h_T^*=\text{argmin}_{h\in\mathcal{H}}\epsilon_T(h)$*. For any* $\delta>0$*, with probability at least* $1-\delta$ *over the data:*

$$\begin{aligned}
\epsilon_T(\hat{h}_\alpha)&\le\epsilon_T(h_T^*)+4\sum_{i=1}^{N}\alpha_i\mathcal{R}_i\left(\mathcal{H}\right)+2\sum_{i=1}^{N}\alpha_i d_{\mathcal{H}}\left(\mathcal{D}_i,\mathcal{D}_T\right)\\
&\quad+6\sqrt{\frac{\log\left(\frac{4}{\delta}\right)M^2}{2}}\sqrt{\sum_{i=1}^{N}\frac{\alpha_i^2}{m_i}},
\end{aligned}\tag{4}$$

where, for each source $i=1,\dots,N$,

$$\mathcal{R}_i\left(\mathcal{H}\right)=\mathbb{E}_\sigma\left(\sup_{f\in\mathcal{H}}\left(\frac{1}{m_i}\sum_{j=1}^{m_i}\sigma_{i,j}L(f(x_{i,j}),y_{i,j})\right)\right)$$

and $\sigma_{i,j}$ *are independent Rademacher random variables.*

Proof. For any $h\in\mathcal{H}$:

$$\begin{aligned}
|\epsilon_\alpha(h)-\epsilon_T(h)|&=|\sum_{i=1}^{N}\alpha_i\epsilon_i(h)-\epsilon_T(h)|\\
&\le\sum_{i=1}^{N}\alpha_i|\epsilon_i(h)-\epsilon_T(h)|\\
&\le\sum_{i=1}^{N}\alpha_i d_{\mathcal{H}}\left(\mathcal{D}_i,\mathcal{D}_T\right).
\end{aligned}$$

Now applying this bound twice and using Proposition 1, we get that with probability at least $1-\delta$:

$$\begin{aligned}
\epsilon_T(\hat{h}_\alpha)&\le\epsilon_\alpha(\hat{h}_\alpha)+\sum_{i=1}^{N}\alpha_i d_{\mathcal{H}}\left(\mathcal{D}_i,\mathcal{D}_T\right)\\
&\le\hat{\epsilon}_\alpha(\hat{h}_\alpha)+2\sum_{i=1}^{N}\alpha_i\mathcal{R}_i\left(\mathcal{H}\right)\\
&\quad+3\sqrt{\frac{\log\left(\frac{4}{\delta}\right)M^2}{2}}\sqrt{\sum_{i=1}^{N}\frac{\alpha_i^2}{m_i}}+\sum_{i=1}^{N}\alpha_i d_{\mathcal{H}}\left(\mathcal{D}_i,\mathcal{D}_T\right)\\
&\le\hat{\epsilon}_\alpha(h_T^*)+2\sum_{i=1}^{N}\alpha_i\mathcal{R}_i\left(\mathcal{H}\right)\\
&\quad+3\sqrt{\frac{\log\left(\frac{4}{\delta}\right)M^2}{2}}\sqrt{\sum_{i=1}^{N}\frac{\alpha_i^2}{m_i}}+\sum_{i=1}^{N}\alpha_i d_{\mathcal{H}}\left(\mathcal{D}_i,\mathcal{D}_T\right)\\
&\le\epsilon_\alpha(h_T^*)+4\sum_{i=1}^{N}\alpha_i\mathcal{R}_i\left(\mathcal{H}\right)\\
&\quad+6\sqrt{\frac{\log\left(\frac{4}{\delta}\right)M^2}{2}}\sqrt{\sum_{i=1}^{N}\frac{\alpha_i^2}{m_i}}+\sum_{i=1}^{N}\alpha_i d_{\mathcal{H}}\left(\mathcal{D}_i,\mathcal{D}_T\right)
\end{aligned}$$

$$\leq \epsilon_T(h_T^*) + 4\sum_{i=1}^{N} \alpha_i \mathcal{R}_i\left(\mathcal{H}\right) + 6\sqrt{\frac{\log\left(\frac{4}{\delta}\right) M^2}{2}}\sqrt{\sum_{i=1}^{N}\frac{\alpha_i^2}{m_i}} + 2\sum_{i=1}^{N}\alpha_i d_{\mathcal{H}}\left(\mathcal{D}_i, \mathcal{D}_T\right)$$

□

B. Details about Algorithm 1

B.1. Distribution-independent upper bounds on the Rademacher complexity

Here we give examples of some well-known upper bounds on the Rademacher complexity of certain function classes, which are distribution-independent. Applying such a bound on the Rademacher terms in Theorem 1 will make the dependence of the second term in the bound on the weights disappear. Therefore, we focus on the remaining terms in our algorithm.

Throughout this section, we discuss the Rademacher complexity of a function class $\mathcal{H}$ with respect to a set of samples $\{x_1, \dots, x_n\} \sim \mathcal{D}$, defined as:

$$\mathcal{R}\left(\mathcal{H}\right) = \mathbb{E}_{\sigma}\left(\sup_{f\in\mathcal{H}}\left(\frac{1}{n}\sum_{i=1}^{n}\sigma_i f(x_i)\right)\right) \tag{13}$$

In the case of bounded binary linear classifiers $\mathcal{H} = \{x \to \langle \mathbf{w}, \mathbf{x}\rangle : \|\mathbf{w}\|_2 \leq B\}$, acting on a bounded domain $\mathcal{X}$ (i.e. for all $x \in \mathcal{X}$, $\|x\|_2 \leq D$), Lemma 26.10 in (Shalev-Shwartz & Ben-David, 2014) shows that:

$$\mathcal{R}\left(\mathcal{H}\right) \leq \frac{BD}{\sqrt{n}}.$$

More generally, the Rademacher complexity of a set of binary classifiers with a finite VC dimention h can be bounded by (Bousquet et al., 2004):

$$\mathcal{R}\left(\mathcal{H}\right) \leq C\sqrt{\frac{h}{n}},$$

for some constant C. The Rademacher complexity is also related to another popular complexity measure, the covering number, via Dudley's entropy bound (Bousquet et al., 2004):

$$\mathcal{R}\left(\mathcal{H}\right) \leq \frac{C}{\sqrt{n}}\int_0^{\infty}\sqrt{\log N\left(\mathcal{H}, t, n\right)}\,dt,$$

where $N\left(\mathcal{H}, t, n\right)$ is the size of the smallest t-cover of the space $\mathcal{H}$, under the metric:

$$d_n\left(h, h'\right) = \frac{1}{n}|\{h(x_i) \neq h'(x_i) : i = 1, \dots, n\}|.$$

Note that both the VC-dimension and the covering number are distribution-independent measures of complexity, so the corresponding upper bounds do not depend on $\mathcal{D}$ as well.

B.2. Computing the empirical discrepancies

As explained in Section 3.2, the discrepancies:

$$d_{\mathcal{H}}\left(\mathcal{D}_i, \mathcal{D}_T\right) = \sup_{h\in\mathcal{H}}\left(|\epsilon_i(h) - \epsilon_T(h)|\right). \tag{14}$$

are unknown in practice and therefore need to be estimated from their empirical counterparts:

$$\begin{aligned} d_{\mathcal{H}}\left(S_i, S_T\right) &= \sup_{h\in\mathcal{H}}\left(|\hat{\epsilon}_i(h) - \hat{\epsilon}_T(h)|\right) \\ &= \sup_{h\in\mathcal{H}}(|\frac{1}{m_i}\sum_{j=1}^{m_i} L\left(h\left(x_{i,j}\right), y_{i,j}\right) \\ &\quad - \frac{1}{m_T}\sum_{j=1}^{m_T} L\left(h\left(x_{T,j}\right), y_{T,j}\right)|). \end{aligned} \tag{15}$$

Here, we explain how these are computed in our experiments. Notice that for the $0/1$-loss, a symmetric hypothesis class $\mathcal{H}$ and whenever $\mathcal{Y} = \{-1, +1\}$, we have:

$$\begin{aligned} &d_{\mathcal{H}}\left(S_i, S_T\right) \\ &= \sup_{h\in\mathcal{H}}|\frac{1}{m_i}\sum_{j=1}^{m_i}\mathbb{1}_{\{h(x_{i,j})y_{i,j}<0\}} - \frac{1}{m_T}\sum_{j=1}^{m_T}\mathbb{1}_{\{h(x_{T,j})y_{T,j}<0\}}| \\ &= \sup_{h\in\mathcal{H}}\left(\frac{1}{m_i}\sum_{j=1}^{m_i}\mathbb{1}_{\{h(x_{i,j})y_{i,j}<0\}} - \frac{1}{m_T}\sum_{j=1}^{m_T}\mathbb{1}_{\{h(x_{T,j})y_{T,j}<0\}}\right) \\ &= \sup_{h\in\mathcal{H}}\left(1 - (\frac{1}{m_i}\sum_{j=1}^{m_i}\mathbb{1}_{\{h(x_{i,j})\bar{y}_{i,j}<0\}} + \frac{1}{m_T}\sum_{j=1}^{m_T}\mathbb{1}_{\{h(x_{T,j})y_{T,j}<0\}})\right) \\ &= 1 - \inf_{h\in\mathcal{H}}\left(\frac{1}{m_i}\sum_{j=1}^{m_i}\mathbb{1}_{\{h(x_{i,j})\bar{y}_{i,j}<0\}} + \frac{1}{m_T}\sum_{j=1}^{m_T}\mathbb{1}_{\{h(x_{T,j})y_{T,j}<0\}}\right), \end{aligned} \tag{16}$$

where $\bar{y}_{i,j} = 1 - y_{i,j}$ is the flipped label of the j-th data point from the i-th source. Now notice that computing the infimum in equation (16) is equivalent to solving a (weighted) empirical risk minimization problem with the input data from the source and the target merged and the labels being the flipped labels from the source and the actual labels from the target.

Therefore, computing the empirical discrepancies is equivalent to solving an empirical risk minimization problem and standard convex upper bounds can be applied to make the problem tractable. In our experiments, we solve the ERM problem by using square loss.

C. Additional results from experiments

Over the next pages we present more detailed results from the experiments on the Animals with Attributes 2 dataset. The table from the main body of the paper (Table 1) is split according to the type of data corruption. In addition, we performed experiments in which a proportion p of the samples in the n corrupted sources are modified (instead of all of them). Apart from $p = 1$, we experimented with $p = 0.5$ and $p = 0.2$. We present the same type of results for these cases, together with a more detailed breakdown, depending on the type of corruption.

Table 3: Summary of the results for $p = 1$, over all 85 prediction tasks and all corruptions (same as Table 1).

Baseline \ n	$n = 0$	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 55$	$n = 59$
Reference only	84/1/0	505/5/0	497/13/0	487/23/0	475/35/0	442/68/0	325/185/0	0/510/0
All data	0/85/0	115/395/0	267/243/0	370/140/0	438/72/0	468/42/0	479/31/0	484/26/0
Median of probs.	9/76/0	47/463/0	172/338/0	336/174/0	469/41/0	504/6/0	502/8/0	499/11/0
(Feng et al., 2014)	8/77/0	32/478/0	110/400/0	338/172/0	457/53/0	504/6/0	502/8/0	497/13/0
(Yin et al., 2018)	14/71/0	179/331/0	390/120/0	432/78/0	472/38/0	502/8/0	503/7/0	497/13/0
(Pregibon, 1982)	55/30/0	308/202/0	361/149/0	416/94/0	437/73/0	455/55/0	470/40/0	485/25/0
Batch norm	0/85/0	107/403/0	317/193/0	416/94/0	446/63/1	478/32/0	487/23/0	482/28/0

Table 4: Summary of results for $p = 1$, split by the type of data corruption

(a) Summary of the results for $p = 1$ and label bias, over all 85 prediction tasks

Baseline \ n	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 55$	$n = 59$
Reference only	85/0/0	84/1/0	82/3/0	80/5/0	76/9/0	55/30/0	0/85/0
All data	61/24/0	82/3/0	85/0/0	85/0/0	85/0/0	85/0/0	84/1/0
Median of probs.	23/62/0	81/4/0	85/0/0	85/0/0	85/0/0	85/0/0	84/1/0
(Feng et al., 2014)	4/81/0	19/66/0	85/0/0	85/0/0	85/0/0	85/0/0	84/1/0
(Yin et al., 2018)	50/35/0	85/0/0	84/1/0	85/0/0	85/0/0	85/0/0	84/1/0
(Pregibon, 1982)	51/34/0	64/21/0	84/1/0	84/1/0	84/1/0	83/2/0	83/2/0
Batch norm	53/32/0	81/4/0	85/0/0	85/0/0	85/0/0	85/0/0	84/1/0

(b) Summary of the results for $p = 1$ and shuffled labels, over all 85 prediction tasks

Baseline \ n	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 55$	$n = 59$
Reference only	83/2/0	77/8/0	71/14/0	65/20/0	56/29/0	38/47/0	0/85/0
All data	4/81/0	51/34/0	69/16/0	74/11/0	82/3/0	79/6/0	79/6/0
Median of probs.	2/83/0	24/61/0	63/22/0	83/2/0	80/5/0	79/6/0	80/5/0
(Feng et al., 2014)	3/82/0	0/85/0	51/34/0	77/8/0	80/5/0	79/6/0	80/5/0
(Yin et al., 2018)	32/53/0	63/22/0	62/23/0	76/9/0	80/5/0	79/6/0	80/5/0
(Pregibon, 1982)	57/28/0	63/22/0	70/15/0	71/14/0	75/10/0	73/12/0	71/14/0
Batch norm	3/82/0	41/44/0	60/25/0	63/21/1	79/6/0	79/6/0	79/6/0

(c) Summary of the results for $p = 1$ and shuffled features, over all 85 prediction tasks

Baseline \ n	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 55$	$n = 59$
Reference only	84/1/0	85/0/0	83/2/0	81/4/0	78/7/0	62/23/0	0/85/0
All data	39/46/0	60/25/0	68/17/0	73/12/0	77/8/0	80/5/0	85/0/0
Median of probs.	6/79/0	30/55/0	84/1/0	85/0/0	85/0/0	85/0/0	85/0/0
(Feng et al., 2014)	10/75/0	72/13/0	84/1/0	85/0/0	85/0/0	85/0/0	85/0/0
(Yin et al., 2018)	18/67/0	76/9/0	84/1/0	85/0/0	85/0/0	85/0/0	85/0/0
(Pregibon, 1982)	56/29/0	58/27/0	67/18/0	74/11/0	78/7/0	85/0/0	85/0/0
Batch norm	40/45/0	66/19/0	72/13/0	77/8/0	78/7/0	79/6/0	80/5/0

(d) Summary of the results for $p = 1$ and blurred images, over all 85 prediction tasks

Baseline \ n	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 55$	$n = 59$
Reference only	84/1/0	84/1/0	83/2/0	82/3/0	77/8/0	59/26/0	0/85/0
All data	0/85/0	2/83/0	26/59/0	53/32/0	66/19/0	75/10/0	78/7/0
Median of probs.	5/80/0	6/79/0	19/66/0	72/13/0	85/0/0	85/0/0	85/0/0
(Feng et al., 2014)	5/80/0	6/79/0	30/55/0	70/15/0	85/0/0	85/0/0	84/1/0
(Yin et al., 2018)	26/59/0	47/38/0	61/24/0	73/12/0	84/1/0	85/0/0	84/1/0
(Pregibon, 1982)	61/24/0	71/14/0	75/10/0	81/4/0	83/2/0	85/0/0	85/0/0
Batch norm	8/77/0	44/41/0	65/20/0	72/13/0	78/7/0	82/3/0	81/4/0

(e) Summary of the results for $p = 1$ and dead pixels, over all 85 prediction tasks

Baseline \ n	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 55$	$n = 59$
Reference only	85/0/0	83/2/0	84/1/0	84/1/0	77/8/0	58/27/0	0/85/0
All data	0/85/0	12/73/0	44/41/0	70/15/0	74/11/0	77/8/0	78/7/0
Median of probs.	6/79/0	6/79/0	14/71/0	61/24/0	85/0/0	85/0/0	85/0/0
(Feng et al., 2014)	6/79/0	6/79/0	25/60/0	58/27/0	85/0/0	85/0/0	84/1/0
(Yin et al., 2018)	23/62/0	51/34/0	68/17/0	70/15/0	84/1/0	85/0/0	84/1/0
(Pregibon, 1982)	28/57/0	38/47/0	51/34/0	52/33/0	56/29/0	69/16/0	85/0/0
Batch norm	1/84/0	28/57/0	59/26/0	69/16/0	74/11/0	79/6/0	79/6/0

(f) Summary of the results for $p = 1$ and RGB channels swapped, over all 85 prediction tasks

Baseline \ n	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 55$	$n = 59$
Reference only	84/1/0	84/1/0	84/1/0	83/2/0	78/7/0	53/32/0	0/85/0
All data	11/74/0	60/25/0	78/7/0	83/2/0	84/1/0	83/2/0	80/5/0
Median of probs.	5/80/0	25/60/0	71/14/0	83/2/0	84/1/0	83/2/0	80/5/0
(Feng et al., 2014)	4/81/0	7/78/0	63/22/0	82/3/0	84/1/0	83/2/0	80/5/0
(Yin et al., 2018)	30/55/0	68/17/0	73/12/0	83/2/0	84/1/0	84/1/0	80/5/0
(Pregibon, 1982)	55/30/0	67/18/0	69/16/0	75/10/0	79/6/0	75/10/0	76/9/0
Batch norm	2/83/0	57/28/0	75/10/0	80/5/0	84/1/0	83/2/0	79/6/0

Table 5: Summary of the results for $p = 0.5$, over all 85 prediction tasks and all corruptions.

Baseline \ n	$n = 0$	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 55$	$n = 59$
Reference only	84/1/0	508/2/0	501/9/0	488/22/0	471/39/0	424/86/0	303/207/0	156/354/0
All data	0/85/0	0/510/0	82/428/0	158/352/0	215/295/0	241/269/0	223/287/0	168/342/0
Median of probs.	9/76/0	30/480/0	53/457/0	93/417/0	189/321/0	272/238/0	252/258/0	216/294/0
(Feng et al., 2014)	8/77/0	28/482/0	19/491/0	84/426/0	172/338/0	254/256/0	253/257/0	217/293/0
(Yin et al., 2018)	14/71/0	123/387/0	227/283/0	155/355/0	247/259/4	295/215/0	282/228/0	224/286/0
(Pregibon, 1982)	55/30/0	287/223/0	282/228/0	329/181/0	350/160/0	358/152/0	374/136/0	367/143/0
Batch norm	0/85/0	2/508/0	78/432/0	139/370/1	183/326/1	186/323/1	155/354/1	97/412/1

Table 6: Summary of results for $p = 0.5$, split by the type of data corruption

(a) Summary of the results for $p = 0.5$ and label bias, over all 85 prediction tasks

Baseline \ n	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 55$	$n = 59$
Reference only	85/0/0	84/1/0	82/3/0	79/6/0	66/19/0	39/46/0	0/85/0
All data	0/85/0	52/33/0	73/12/0	82/3/0	84/1/0	84/1/0	84/1/0
Median of probs.	9/76/0	41/44/0	72/13/0	80/5/0	84/1/0	84/1/0	84/1/0
(Feng et al., 2014)	5/80/0	7/78/0	62/23/0	76/9/0	83/2/0	84/1/0	84/1/0
(Yin et al., 2018)	27/58/0	57/28/0	48/37/0	79/6/0	84/1/0	83/2/0	83/2/0
(Pregibon, 1982)	48/37/0	49/36/0	58/27/0	65/20/0	70/15/0	79/6/0	84/1/0
Batch norm	1/84/0	46/39/0	69/16/0	76/9/0	79/6/0	76/9/0	76/9/0

(b) Summary of the results for $p = 0.5$ and shuffled labels, over all 85 prediction tasks

Baseline \ n	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 55$	$n = 59$
Reference only	84/1/0	82/3/0	72/13/0	59/26/0	42/43/0	28/57/0	17/68/0
All data	0/85/0	0/85/0	9/76/0	31/54/0	47/38/0	50/35/0	49/36/0
Median of probs.	3/82/0	0/85/0	0/85/0	13/72/0	40/45/0	47/38/0	49/36/0
(Feng et al., 2014)	4/81/0	0/85/0	0/85/0	4/81/0	29/56/0	44/41/0	48/37/0
(Yin et al., 2018)	25/60/0	44/41/0	17/68/0	14/67/4	19/66/0	28/57/0	39/46/0
(Pregibon, 1982)	56/29/0	44/41/0	59/26/0	59/26/0	60/25/0	67/18/0	64/21/0
Batch norm	0/85/0	0/85/0	1/83/1	10/74/1	4/80/1	2/82/1	2/82/1

(c) Summary of the results for $p = 0.5$ and shuffled features, over all 85 prediction tasks

Baseline \ n	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 55$	$n = 59$
Reference only	84/1/0	85/0/0	83/2/0	82/3/0	76/9/0	44/41/0	0/85/0
All data	0/85/0	26/59/0	51/34/0	57/28/0	57/28/0	41/44/0	0/85/0
Median of probs.	6/79/0	4/81/0	10/75/0	56/29/0	69/16/0	50/35/0	6/79/0
(Feng et al., 2014)	6/79/0	4/81/0	14/71/0	61/24/0	72/13/0	56/29/0	6/79/0
(Yin et al., 2018)	9/76/0	18/67/0	49/36/0	73/12/0	77/8/0	67/18/0	6/79/0
(Pregibon, 1982)	49/36/0	48/37/0	54/31/0	60/25/0	60/25/0	56/29/0	50/35/0
Batch norm	1/84/0	32/53/0	53/32/0	60/25/0	61/24/0	55/30/0	14/71/0

(d) Summary of the results for $p = 0.5$ and blurred images, over all 85 prediction tasks

Baseline \ n	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 55$	$n = 59$
Reference only	85/0/0	83/2/0	83/2/0	83/2/0	82/3/0	75/10/0	67/18/0
All data	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0
Median of probs.	4/81/0	3/82/0	4/81/0	6/79/0	17/68/0	14/71/0	24/61/0
(Feng et al., 2014)	4/81/0	3/82/0	3/82/0	5/80/0	16/69/0	16/69/0	26/59/0
(Yin et al., 2018)	19/66/0	31/54/0	10/75/0	29/56/0	34/51/0	32/53/0	38/47/0
(Pregibon, 1982)	58/27/0	58/27/0	63/22/0	67/18/0	70/15/0	72/13/0	73/12/0
Batch norm	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0

(e) Summary of the results for $p = 0.5$ and dead pixels, over all 85 prediction tasks

Baseline \ n	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 55$	$n = 59$
Reference only	85/0/0	83/2/0	84/1/0	84/1/0	80/5/0	70/15/0	62/23/0
All data	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0
Median of probs.	4/81/0	3/82/0	4/81/0	4/81/0	10/75/0	9/76/0	18/67/0
(Feng et al., 2014)	4/81/0	3/82/0	3/82/0	4/81/0	9/76/0	10/75/0	16/69/0
(Yin et al., 2018)	20/65/0	32/53/0	11/74/0	21/64/0	29/56/0	20/65/0	21/64/0
(Pregibon, 1982)	25/60/0	24/61/0	37/48/0	37/48/0	38/47/0	40/45/0	37/48/0
Batch norm	0/85/0	0/85/0	0/85/0	2/83/0	4/81/0	4/81/0	1/84/0

(f) Summary of the results for $p = 0.5$ and RGB channels swapped, over all 85 prediction tasks

Baseline \ n	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 55$	$n = 59$
Reference only	85/0/0	84/1/0	84/1/0	84/1/0	78/7/0	47/38/0	10/75/0
All data	0/85/0	4/81/0	25/60/0	45/40/0	53/32/0	48/37/0	35/50/0
Median of probs.	4/81/0	2/83/0	3/82/0	30/55/0	52/33/0	48/37/0	35/50/0
(Feng et al., 2014)	5/80/0	2/83/0	2/83/0	22/63/0	45/40/0	43/42/0	37/48/0
(Yin et al., 2018)	23/62/0	45/40/0	20/65/0	31/54/0	52/33/0	52/33/0	37/48/0
(Pregibon, 1982)	51/34/0	59/26/0	58/27/0	62/23/0	60/25/0	60/25/0	59/26/0
Batch norm	0/85/0	0/85/0	16/69/0	35/50/0	38/47/0	18/67/0	4/81/0

Table 7: Summary of the results for $p = 0.2$, over all 85 prediction tasks and all corruptions.

Baseline \ n	$n = 0$	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 55$	$n = 59$
Reference only	84/1/0	507/3/0	505/5/0	504/6/0	492/18/0	459/51/0	429/81/0	404/106/0
All data	0/85/0	0/510/0	0/510/0	0/510/0	0/510/0	1/509/0	2/508/0	1/509/0
Median of probs.	9/76/0	28/482/0	21/489/0	16/494/0	17/493/0	28/482/0	30/478/2	31/479/0
(Feng et al., 2014)	8/77/0	30/480/0	24/486/0	16/494/0	16/494/0	20/489/1	23/485/2	26/484/0
(Yin et al., 2018)	14/71/0	95/415/0	146/364/0	34/476/0	42/468/0	39/471/0	36/474/0	40/470/0
(Pregibon, 1982)	55/30/0	282/228/0	282/228/0	275/235/0	287/223/0	264/246/0	281/229/0	267/243/0
Batch norm	0/85/0	0/510/0	0/510/0	0/510/0	0/510/0	0/509/1	0/509/1	1/508/1

Table 8: Summary of results for $p = 0.2$, split by the type of data corruption

(a) Summary of the results for $p = 0.2$ and label bias, over all 85 prediction tasks

Baseline \ n	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 55$	$n = 59$
Reference only	85/0/0	84/1/0	84/1/0	76/9/0	60/25/0	46/39/0	28/57/0
All data	0/85/0	0/85/0	0/85/0	0/85/0	1/84/0	2/83/0	1/84/0
Median of probs.	5/80/0	1/84/0	0/85/0	0/85/0	11/74/0	13/70/2	13/72/0
(Feng et al., 2014)	5/80/0	4/81/0	0/85/0	0/85/0	3/81/1	5/78/2	8/77/0
(Yin et al., 2018)	25/60/0	49/36/0	7/78/0	19/66/0	12/73/0	4/81/0	2/83/0
(Pregibon, 1982)	47/38/0	40/45/0	48/37/0	44/41/0	40/45/0	44/41/0	41/44/0
Batch norm	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0

(b) Summary of the results for $p = 0.2$ and shuffled labels, over all 85 prediction tasks

Baseline \ n	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 55$	$n = 59$
Reference only	84/1/0	84/1/0	84/1/0	84/1/0	77/8/0	74/11/0	73/12/0
All data	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0
Median of probs.	4/81/0	4/81/0	1/84/0	1/84/0	1/84/0	0/85/0	0/85/0
(Feng et al., 2014)	5/80/0	4/81/0	1/84/0	1/84/0	1/84/0	0/85/0	0/85/0
(Yin et al., 2018)	17/68/0	31/54/0	4/81/0	1/84/0	1/84/0	0/85/0	2/83/0
(Pregibon, 1982)	47/38/0	54/31/0	49/36/0	53/32/0	52/33/0	54/31/0	51/34/0
Batch norm	0/85/0	0/85/0	0/85/0	0/85/0	0/84/1	0/84/1	0/84/1

(c) Summary of the results for $p = 0.2$ and shuffled features, over all 85 prediction tasks

Baseline \ n	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 55$	$n = 59$
Reference only	84/1/0	85/0/0	83/2/0	82/3/0	73/12/0	66/19/0	61/24/0
All data	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0
Median of probs.	5/80/0	4/81/0	4/81/0	4/81/0	3/82/0	3/82/0	2/83/0
(Feng et al., 2014)	5/80/0	4/81/0	4/81/0	4/81/0	3/82/0	3/82/0	2/83/0
(Yin et al., 2018)	8/77/0	5/80/0	4/81/0	4/81/0	3/82/0	3/82/0	2/83/0
(Pregibon, 1982)	50/35/0	44/41/0	44/41/0	52/33/0	49/36/0	51/34/0	39/46/0
Batch norm	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0

(d) Summary of the results for $p = 0.2$ and blurred images, over all 85 prediction tasks

Baseline \ n	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 55$	$n = 59$
Reference only	85/0/0	84/1/0	84/1/0	83/2/0	83/2/0	84/1/0	84/1/0
All data	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0
Median of probs.	4/81/0	3/82/0	4/81/0	6/79/0	7/78/0	10/75/0	10/75/0
(Feng et al., 2014)	4/81/0	3/82/0	4/81/0	6/79/0	7/78/0	10/75/0	11/74/0
(Yin et al., 2018)	15/70/0	15/70/0	6/79/0	9/76/0	14/71/0	18/67/0	22/63/0
(Pregibon, 1982)	57/28/0	56/29/0	56/29/0	55/30/0	55/30/0	55/30/0	58/27/0
Batch norm	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0

(e) Summary of the results for $p = 0.2$ and dead pixels, over all 85 prediction tasks

Baseline \ n	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 55$	$n = 59$
Reference only	85/0/0	84/1/0	84/1/0	83/2/0	84/1/0	82/3/0	83/2/0
All data	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0
Median of probs.	6/79/0	5/80/0	4/81/0	4/81/0	4/81/0	3/82/0	5/80/0
(Feng et al., 2014)	6/79/0	5/80/0	4/81/0	3/82/0	4/81/0	4/81/0	4/81/0
(Yin et al., 2018)	12/73/0	14/71/0	7/78/0	6/79/0	6/79/0	9/76/0	10/75/0
(Pregibon, 1982)	30/55/0	36/49/0	26/59/0	27/58/0	23/62/0	25/60/0	26/59/0
Batch norm	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	1/84/0

(f) Summary of the results for $p = 0.2$ and RGB channels swapped, over all 85 prediction tasks

Baseline \ n	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 55$	$n = 59$
Reference only	84/1/0	84/1/0	85/0/0	84/1/0	82/3/0	77/8/0	75/10/0
All data	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0
Median of probs.	4/81/0	4/81/0	3/82/0	2/83/0	2/83/0	1/84/0	1/84/0
(Feng et al., 2014)	5/80/0	4/81/0	3/82/0	2/83/0	2/83/0	1/84/0	1/84/0
(Yin et al., 2018)	18/67/0	32/53/0	6/79/0	3/82/0	3/82/0	2/83/0	2/83/0
(Pregibon, 1982)	51/34/0	52/33/0	52/33/0	56/29/0	45/40/0	52/33/0	52/33/0
Batch norm	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0	0/85/0