Rowan University

Rowan Digital Works

Rethinking Noisy Label Models: Labeler-Dependent Noise with Adversarial Awareness

Glenn Dawson

Robi Polikar
Rowan University, polikar@rowan.edu

Recommended Citation

Rethinking Noisy Label Models: Labeler-Dependent Noise with Adversarial Awareness

Glenn Dawson
Rowan University
Glassboro, New Jersey, USA
`dawsong@rowan.edu`

Robi Polikar
Rowan University
Glassboro, New Jersey, USA
`polikar@rowan.edu`

Abstract

Most studies on learning from noisy labels rely on unrealistic models of i.i.d. label noise, such as class-conditional transition matrices. More recent work on instance-dependent noise models are more realistic, but assume a single generative process for label noise across the entire dataset. We propose a more principled model of label noise that generalizes instance-dependent noise to multiple labelers, based on the observation that modern datasets are typically annotated using distributed crowdsourcing methods. Under our labeler-dependent model, label noise manifests itself under two modalities: *natural error* of good-faith labelers, and *adversarial labels* provided by malicious actors. We present two adversarial attack vectors that more accurately reflect the label noise that may be encountered in real-world settings, and demonstrate that under our multimodal noisy labels model, state-of-the-art approaches for learning from noisy labels are defeated by adversarial label attacks. Finally, we propose a multi-stage, labeler-aware, model-agnostic framework that reliably filters noisy labels by leveraging knowledge about which data partitions were labeled by which labeler, and show that our proposed framework remains robust even in the presence of extreme adversarial label noise.

1 Introduction

Learning from labeled data relies on the availability of accurate labels. However, obtaining accurate labels on large datasets is costly, sometimes prohibitively so [37, 22, 28, 46], which has led to crowdsourcing as an attractive and cost-effective solution for distributed label gathering [109, 68, 96, 95, 114, 79, 76]. Unfortunately, the labels obtained via crowdsourcing are of unreliable veracity, and models trained naively on such unreliable data are vulnerable to overfitting on noisy labels [106, 113]. Furthermore, some labelers on crowdsourcing platforms may be adversarial, providing intentionally incorrect labels and greatly increasing the incidence of label noise [34, 41, 53, 10].

Learning in the presence of noisy labels has emerged as an area of active research [23, 1, 84], and generally follows one of two modalities. The *learning from crowds* approach assumes that labels are generated by multiple annotators, each with an associated but unknown probability of generating a correct label [21, 69, 101, 111, 20], while the *learning from noisy labels* approach treats the label noise as i.i.d., with an unobserved label flipping process governing the incidence of label noise for all examples [59, 8, 103, 6]. While both approaches have been successful for addressing label uncertainty under their respective assumptions, they each depend on unrealistic restrictions. Methods for learning from crowds rely on redundant labels, with the "wisdom of the crowd" overruling unreliable or adversarial labels [75, 108, 91]. However, requiring multiple labels per data instance greatly increases the burden of label gathering, running counter to the original problem of label gathering. Meanwhile, both class-conditional and instance-dependent approaches for learning from noisy labels ignore important information about the source of each label that can be leveraged for more robust performance, as well as the possibility of non-i.i.d. label noise distributions [32, 23].

We address these drawbacks by combining the strengths of each approach. We propose a labeler-dependent model of label noise that inherits the multiple-labeler paradigm of crowdsourcing *without* requiring overlapping label redundancy, while simultaneously incorporating instance-dependent noise modeling and introducing the threat of adversarial labels. We believe that our framework more accurately models the types of label noise that a learner may encounter in a real-world scenario.

Furthermore, to address the challenges introduced by our multimodal label noise model, we propose a novel three-stage learning framework. We draw inspiration from social learning theory as well as theory of mind, which provide evidence for humans modeling the unobserved opinions of others during their own learning and decision-making processes [42, 24, 64, 35, 39, 15]. First, we create an ensemble of semi-supervised models of each labeler, and query each labeler model for an estimate of the label that its corresponding labeler *would* have provided for each data instance, thus achieving synthetic label redundancy without requiring overlapping labels directly from each labeler. Then, we integrate the queried labels into a single label per instance that is more reliable than the queried labels using methods for learning from crowds. Finally, we use our filtered (but still noisy) labels to train a classifier using algorithms for learning from noisy labels. Our framework is modular and agnostic to its component algorithms, allowing for refinement in each stage as respective advances continue in the fields of semi-supervised learning, learning from crowds, and learning from noisy labels.

2   Preliminaries

For a K-class problem, we define the label space $\mathbf{Y}$ as a $K \times K$ identity matrix for one-hot encoding of the classes, with y_k representing the kth class label. The true label for instance $\boldsymbol{x}_i$ is denoted as $y_i \in \mathbf{Y}$, and a noisy label for instance $\boldsymbol{x}_i$ is denoted as $\hat{y}_i \in \{\mathbf{Y} \setminus y_i\}$.

Class-conditional label noise   Label noise is commonly treated as a class-conditional phenomenon, where the noisy labels are treated strictly as a function of the true label [54, 57, 72, 98, 33, 102, 51, 67, 58, 5, 11, 4, 3, 94, 100, 90, 104, 93]. Under this treatment, label corruption is modeled using a transition matrix T describing the probability of a true label being randomly flipped [59, 66, 14, 88]. Two common forms of T for performing evaluations of noisy label learning include *symmetric* and *asymmetric* noise [87]. Under symmetric noise, T follows a uniform distribution, with each label y_i having an equal probability of corruption to any other label y_k. In contrast, under asymmetric label noise T is deliberately constructed to place higher probabilities on classes heuristically similar to y_i [66, 77, 46, 85, 73, 29, 92].

Instance-dependent label noise   Both symmetric and asymmetric label noise are attractive schemes for synthetically generating label noise, as they allow complete control over the amount of label noise injected into the dataset. However, these approaches ignore the data-driven dependencies of the labels, and produce unrealistic data-label distributions. More realistic models include early work on feature-dependent label noise [44, 56], and, more recently, instance-dependent noise [25, 12, 92, 14, 116, 112, 99]. One such approach involves projecting each instance onto a randomly-sampled set of K vectors and sampling from a combination of this projection with the clean label to generate a noisy label [99, 116, 14]. Other approaches generate instance-specific noise by training a deep neural network on the clean training dataset and using its noisy outputs; some authors use the entire set of noisy labels generated by the network [92], while others retain only a percentage of the noisy labels [12]. The polynomial margin diminishing noise introduced in [112] extends confidence-based noise to stochastically flip labels based on the outputs of a neural network trained on clean labels.

Noisy labels from crowdsourcing   Both class-conditional and instance-dependent noise models assume that the noisy labels are generated by a single process for the entire training dataset. This is an unrealistic assumption in the context of big data, where labels may be gathered via distributed methods such as crowdsourcing. Addressing unreliable labels is a fundamental objective of learning from crowds [81, 2, 115, 74, 107, 13, 50, 26, 16, 89], and can be traced to the seminal work of Dawid and Skene, who modeled the accuracy of each labeler as a hidden confusion matrix [17]. The GLAD model adds instance-specific dependence by modeling the difficulty of correctly labeling each instance alongside each labeler's expertise [96]. A parameterization assuming that labels are generated from a Gaussian mixture model encoding both the labeler- and instance-specific noise was proposed in [95]. The personal classifiers model assumes that labels are generated by labeler-specific logistic regression models, characterized by their deviation from an optimal base classifier [36].
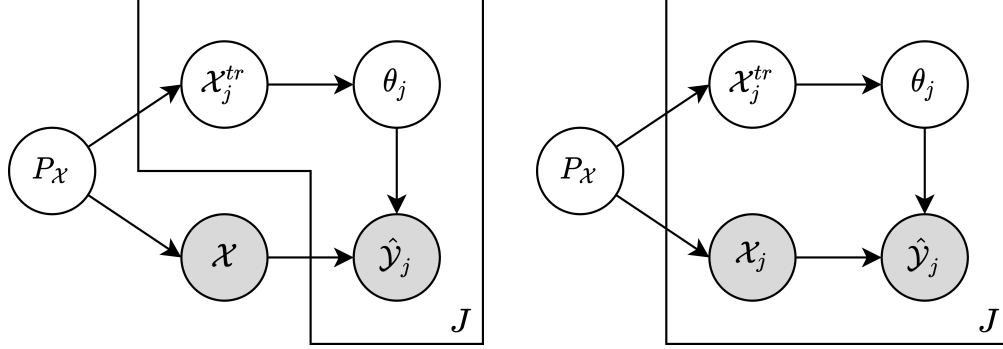
Figure 1: **Left:** In conventional crowdsourcing, the full training dataset $\mathcal{X}$ is drawn from the underlying distribution $P_\mathcal{X}$, and J labelers provide redundant labels $\hat{\mathcal{y}}_j$ on $\mathcal{X}$. **Right:** Our model assumes that each labeler j observes an independently-drawn subset $\mathcal{X}_j \sim P_\mathcal{X}$, and provides labels $\hat{\mathcal{y}}_j$ only on $\mathcal{X}_j$. In both models, the parameters θ_j are trained on independently-drawn data $\mathcal{X}_j^{tr} \sim P_\mathcal{X}$.

Adversarial label noise Despite a considerable body of work on adversarial evasion attacks against deep neural networks, there is comparatively little research on adversarial label attacks. The possibility of adversarial labelers in crowdsourcing was acknowledged in [96], but without explicitly addressing such threats. Other efforts have investigated the vulnerability of support vector machines to label flipping attacks [9], and have studied identifying adversarial workers in crowdsourcing [34]. Arbitrary adversaries that violate the Dawid-Skene model, which may even be in collusion, were considered in [41]; their work was extended to the task assignment matrix not being dense in [53].

3   Labeler-dependent noise: mixing natural error and adversarial labels

In this section, we propose a hybrid labeler-dependent model for noisy labels that integrates key insights from each of the previously-discussed settings. To our knowledge, ours is the first work that has attempted to extend instance-dependent label noise to the multiple-labelers paradigm.

Multiple labelers We introduce the multiple-labelers paradigm from crowdsourcing into the instance-dependent noise model by modeling the training dataset $\mathcal{X}$ as the concatenation of multiple partitioned subsets $\mathcal{X}_j$, each labeled by an independent labeler j. We note the differences between our proposed model and the conventional crowdsourcing model, which assumes that all labelers provide redundant labels on the same dataset $\mathcal{X}$; instead, we consider that each data instance is associated with only a single label. Assuming that each $\mathcal{X}_j$ is drawn from the same underlying distribution $P_\mathcal{X}$, the resultant dataset $\mathcal{X} = \{\bigcup_j \mathcal{X}_j\}$ is identical to a supervised dataset with noisy labels, but with additional information about which subsets of data were labeled by which labeler. The differences between the conventional crowdsourcing model and our proposed model are shown in Figure 1.

Natural error of good-faith labelers We consider each labeler j to generate labels following a categorical distribution parameterized by θ_j such that $\hat{y}_{ij} \sim \mathrm{Cat}(\mathbf{Y}|\boldsymbol{\pi}_{ij})$, where $\boldsymbol{\pi}_{ij} = f_{\boldsymbol{\theta}_j}(\boldsymbol{x}_i)$ can be thought of as the softmax output of the classifier $f_{\boldsymbol{\theta}_j}$ with θ_j learned on an unobserved training dataset $\mathcal{D}_j^{tr} = \{\mathcal{X}_j^{tr} \nsubseteq \mathcal{X}, \mathcal{Y}_j^{tr}\}$. The natural classification error of each labeler assumes the role of instance-dependent label noise; our model is similar to the instance-dependent noise technique of training a deep neural network on the cleanly-labeled training data and using its outputs as noisy labels. However, we distinguish our model in two ways: (i) the noisy labels are generated not by a single neural network but by an ensemble of independent networks, causing more complex instance- and labeler-dependent label noise dynamics; and (ii), the parameters θ_j are not learned on the same data for which the noisy labels are provided, avoiding data leakage and maintaining a more realistic model of the relations between the labelers and the data.

Adversarial labels from bad-faith labelers The good-faith labeler model can be extended to describe *adversarial* labelers. We posit that an intelligent adversary will present the learner with a false label based on not only its own best guess as to the correct label, but also its *second* best

3

guess. We define the function $\arg_2\max$ as identical to the argmax function, except that $\arg_2\max(A)$ returns the index of the element of A with the second highest value instead of the maximum (with ties broken arbitrarily). An adversarial labeler therefore provides noisy labels following

$$\hat{y}_{ij} = \arg_2\max_k f_{\theta_j}(\boldsymbol{x}_i). \tag{1}$$

The labels that the adversary presents are representative of the adversary's best guess about the false labels that are most likely to be confused with the correct labels. This approach is similar to that in [112], where the label of the second most-confident category is integrated into the noise model. Note that the adversary may itself have an incorrect belief about the true label, and in its attempt to provide a false label may accidentally provide a correct label.

Following the crowdsourcing paradigm, we assume that the label gathering process draws from a pool of labelers containing a mixture of both good-faith (but imperfect) labelers and malicious, adversarial labelers. By combining natural and adversarial noise from multiple labelers, we define a multimodal, labeler-dependent model that is both more realistic than class-conditional approaches and more general than single-process instance-dependent approaches. We note that the characteristics of both natural and adversarial noise will vary from labeler to labeler, and that even when adversarial noise is absent there may be variations in the levels and characteristics of the natural error associated with each labeler.

Multiple-labeler adversarial attack vectors   We define two realistic vectors of adversarial attacks that should form the basis of future work on adversary-aware learning from noisy labels. A *data flooding attack* occurs when a single adversary provides an overwhelming quantity of adversarial labels relative to the quantity of labels provided by the good-faith labelers. A *multiple adversaries attack* occurs when multiple bad-faith labelers invade the labeling process; unlike the data flooding attack, each adversarial labeler need not provide large quantities of labels. Both attacks exploit vulnerabilities in distributed label collection, especially in crowdsourcing and online learning scenarios, and have the capability to introduce large amounts of adversarial noise into the training data.

Under our model, an algorithm for learning from noisy labels must be resilient to both natural and adversarial label noise, and algorithmic robustness can be measured as its vulnerability to increasing levels of adversarial labels. While the most favorable learning environment will include no adversarial noise, a robust algorithm will exhibit little to no reduction in performance as more noise is introduced; a weak algorithm, however, will be heavily compromised by increasingly adversarial labels, and its performance will be negatively impacted.

4   A three-stage framework for labeler- and adversary-aware learning

---

Algorithm 1 Stage one: semi-supervised labeler modeling.

Inputs: $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_J\}$, the set of data-label pairs provided by each label source $j \in J$; SSL, a semi-supervised classification algorithm
1: $\boldsymbol{\Phi} \leftarrow$ Initialize an empty set of trained models
2: **for** each labeler $j = 1$ to J **do**
3:     $\mathcal{X}_L, \hat{\mathcal{Y}}_L \leftarrow \mathcal{D}_j$ (Gather labeled data from labeler j)
4:     $\mathcal{X}_U \leftarrow \{\bigcup_{\ell \neq j} \mathcal{X}_\ell\} \, \forall \ell \in J$ (Gather unlabeled data from all other labelers $\ell \neq j$)
5:     $\phi_j \leftarrow \mathrm{SSL}(\mathcal{X}_L, \hat{\mathcal{Y}}_L, \mathcal{X}_U)$ (Estimate θ_j)
6:     $\boldsymbol{\Phi} \leftarrow \{\boldsymbol{\Phi} \cup \phi_j\}$
7: **end for**
Output: $\boldsymbol{\Phi}$, the set of estimates of labeler parameters

---

We consider the set of partitioned subsets of noisily-labeled data provided by each labeler, $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_J\}$, where each $\mathcal{D}_j = \{\mathcal{X}_j, \hat{\mathcal{Y}}_j\}$. Because each labeler j provides labels only on $\mathcal{X}_j$, and each $\mathcal{X}_j$ is disjoint from every other $\mathcal{X}_{\ell \neq j}$, we cannot directly use methods for learning from crowds to exploit our labeler-aware knowledge. To accommodate this scenario, Stage 1 of our framework trains a set of models $\boldsymbol{\Phi} = \{\phi_1, \ldots, \phi_J\}$, respectively estimating each θ_j, which can then be queried to obtain *estimates* of the labels that each labeler j *might* have provided. To do so, we observe that while the labels $\hat{\mathcal{Y}}_{\ell \neq j}$ from all labelers $\ell \neq j$ are uninformative for estimating θ_j, the data $\mathcal{X}_{\ell \neq j}$ may

still be exploited assuming that each $\mathcal{X}_j$ are drawn from the same underlying distribution $P_{\mathcal{X}}$. We leverage semi-supervised learning (SSL) to train each ϕ_j by treating the data-label pairs $\{\mathcal{X}_j, \hat{\mathcal{Y}}_j\}$ as labeled data, and the union $\{\bigcup_\ell \mathcal{X}_{\ell \neq j}\}$ as unlabeled data. For modularity, we place no restrictions on which SSL methods may be used. It is also not essential for any ϕ_j to achieve high accuracy on test data; in this stage, we are interested only in estimating, as closely as possible, the parameters θ_j that describe the label generation dynamics of each labeler j, regardless of generalization performance. The procedure for semi-supervised labeler modeling is shown in Algorithm 1.

Algorithm 2 Stage two: combining real and synthetic labels with learning from crowds.

Inputs: $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_J\}$, the set of data-label pairs provided by each label source $j \in J$; $\Phi = \{\phi_1, \ldots, \phi_J\}$, the set of estimates of each θ_j for each label source j; LFC, an algorithm for classification by learning from crowds
1: $\mathcal{W} \leftarrow$ Initialize an empty $J \times N$ matrix of labels from J labelers on N instances
2: **for** each labeler $j = 1$ to J **do**
3:     **for** each dataset $\ell = 1$ to J **do**
4:         $\mathcal{W}_{j\ell} \leftarrow \begin{cases} \hat{\mathcal{Y}}_j, & \ell = j \\ g_{\phi_j}(\mathcal{X}_\ell), & \ell \neq j \end{cases}$
5:     **end for**
6: **end for**
7: $\widetilde{\mathcal{Y}} \leftarrow \text{LFC}(\mathcal{W})$ (Estimate true labels from labels matrix using learning from crowds)
Output: $\widetilde{\mathcal{Y}}$, the set of estimates for the true labels of $\mathcal{X}$

Once all ϕ_j are obtained, Stage 2 of our framework queries each model to obtain synthetic labels on the entire set of training data. Combining these synthetic labels (queried from ϕ_j) with the labeler-provided labels $\hat{\mathcal{Y}}_j$ (generated by θ_j), we now have a full set of J redundant labels for each $x_i \in \mathcal{X}$. Hence, we can use learning from crowds to integrate the redundant noisy labels into a single filtered label per instance. As in Stage 1, we place no restrictions on which methods for learning from crowds may be utilized. Combining real and synthetic labels is shown in Algorithm 2.

Algorithm 3 Three-stage framework for labeler-aware learning under adversarial label noise.

Inputs: $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_J\}$, the set of data-label pairs provided by each label source $j \in J$; SSL, a semi-supervised classification algorithm; LFC, an algorithm for classification by learning from crowds; LNL, an algorithm for learning from noisy labels
1: Learn $\Phi(\mathcal{D}, \text{SSL})$ following Algorithm 1
2: Learn $\widetilde{\mathcal{Y}}(\mathcal{D}, \Phi, \text{LFC})$ following Algorithm 2 to filter adversarial noise
3: Learn $\Theta \leftarrow \text{LNL}(\widetilde{\mathcal{Y}})$ to filter natural noise
Output: Θ, a classifier for learning from multiple noisy labelers

---

While methods for learning from crowds are effective at filtering adversarial noise, the labels that they produce are not free of natural noise. To reduce the impact of natural noise, we use the labels produced in Stage 2 to train Θ, a model for learning from noisy labels. The first two stages therefore constitute a bootstrapping procedure (similar to [71]), with Stage 3 being stacked on top of the bootstrapped labels. As before, we place no restrictions on which methods for learning from noisy labels can be used. The complete procedure for our three-stage framework is shown in Algorithm 3.

5 Comparisons to state-of-the-art under adversarial label noise

We demonstrate our label-gathering process, as well as our proposed algorithm for addressing multi-modal label noise, via experiments that simulate the process under both data flooding and multiple adversaries attack scenarios. We used the MNIST [45] and SVHN [60] datasets, two commonly-used benchmark datasets for learning from noisy labels. We evaluate against the DivideMix [47], self-evolution average label (SEAL) [12], and progressive label correction (PLC) [112] algorithms, representing the current state-of-the-art in learning from noisy labels under both class-conditional and instance-dependent noise models. We also compare the effectiveness of our framework using only the first two stages against the complete three-stage framework.

Experimental setup Our labeler-dependent noise model generates noisy labels as a function of θ_j, preventing precise control of the amount of label noise. Instead, we parameterize each experiment by the amount of data provided by adversarial labelers. For data flooding attack, we fix the amount of data provided by good-faith labelers and vary the amount of data provided by a single adversarial labeler; for the multiple adversaries attack, we fix the number of labelers and the amount of data provided by each labeler, and vary the number of adversarial labelers.

For each experiment, we modeled J label sources by training J neural networks θ_j on small subsets $\mathcal{D}_j^{tr}$, drawn without replacement from the training dataset $\mathcal{D}$. Each labeler j then provided labels on randomly-partitioned subsets $\mathcal{X}_j$ of the remainder of the training data, with $\bigcup_j \mathcal{X}_j = \{\mathcal{X} \setminus \{\bigcup_j \mathcal{X}_j^{tr}\}\}$. Good-faith labelers provided labels following $\hat{y}_{ij} = \operatorname{argmax}_k f_{\theta_j}(\boldsymbol{x}_i)$, while adversarial labelers followed $\hat{y}_{ij} = \arg_2\max_k f_{\theta_j}(\boldsymbol{x}_i)$ (Eqn. 1). For the MNIST dataset, we set $J = 10$, $N_{tr} = 200$, and each θ_j was a randomly-initialized ResNet-18 [30], producing ~7.6% natural error. For the SVHN dataset, we set $J = 5$, $N_{tr} = 20,000$, and each θ_j was a randomly-initialized Wide ResNet-50 [105], producing ~9% natural error. The complete experimental procedure is detailed in Appendix C.

Once the noisy labels were generated by each labeler, the noisily-labeled data were passed to the learning algorithm under test. For our labeler-aware approach, we provided each labeler's data-label pairs $\{\mathcal{X}_j, \hat{\mathcal{Y}}_j\}$ as separate datasets. For the labeler-agnostic approaches, we combined the data-label pairs from each source into a single dataset (i.e. as they would be observed under labeler-agnostic assumptions). We repeated each experiment ten times for MNIST and five times for SVHN, and we report the results as the mean classification accuracies in response to increasing adversarial noise, bounded by their 95% confidence intervals based on the two-sided Student's t-test.

Model selection and hyperparameter tuning Due to modular nature of our framework, the selections for the semi-supervised learning, learning from crowds, and learning from noisy labels algorithms to use for each stage constitute its main high-level hyperparameters. We note that our framework's modularity allows for the seamless replacement of any or all of these algorithmic choices; we demonstrate this modularity by selecting different SSL algorithms for each experiment. For the MNIST dataset, we use auxiliary deep generative models as our SSL algorithm due to its small parameter footprint [55]; for the SVHN dataset, we chose the FixMatch algorithm as representative of the current state-of-the-art for semi-supervised learning [82]. For both datasets, we use OpinionRank as our learning from crowds algorithm due to its nonparametric nature and fast performance [18], and selected DivideMix for learning from noisy labels due to its state-of-the-art performance on a wide variety of noisy labels tasks [47].

Since we cannot assume the presence of a clean validation set, we do not perform any ground truth-based hyperparameter tuning or model selection, as in [6]. Instead, the hyperparameters for each component of our modular framework, as well as those of the algorithms against which we are comparing, were selected based on the suggestions of the original authors of each algorithm, available in their publicly-available online implementations. We made only minor adjustments in order to accommodate differences in intended datasets, and in all cases we tested our changes in the non-adversarial setting to ensure fair comparison. All implementation details and experimental parameters can be found in Appendix C.

Due to lack of any fine-tuning, we are likely reporting conservative results for all algorithms, including our own. However, we are emphatically *not* presenting our results as benchmark scores; rather, we are interested in the overall *trends of behavior* that characterize the vulnerabilities of each algorithm to increasing intensities of adversarial attacks. While it may be possible to obtain minor improvements through extensive hyperparameter tuning on a clean validation dataset, we argue that because we cannot assume the existence of such a set, such tuning would constitute unfair data leakage. More importantly, we believe that such minor improvements would not change the overall structure of our results with respect to the characteristics of each algorithm's vulnerability to adversarial label noise.

Data flooding For the MNIST dataset, nine good-faith labelers each provided $N_j = 1,000$ labels featuring only natural error, and a single adversarial labeler provided noisy labels varying between 0 and 49,000. For the SVHN dataset, four good-faith labelers each provided $N_j = 20,000$ labels featuring only natural error, and a single adversarial labeler provided noisy labels varying between 0 and 424,000. For data flooding experiments, the total amount of data visible to the learner varies with the size of the adversary, with the total amount of good-faith labeled data remaining fixed.
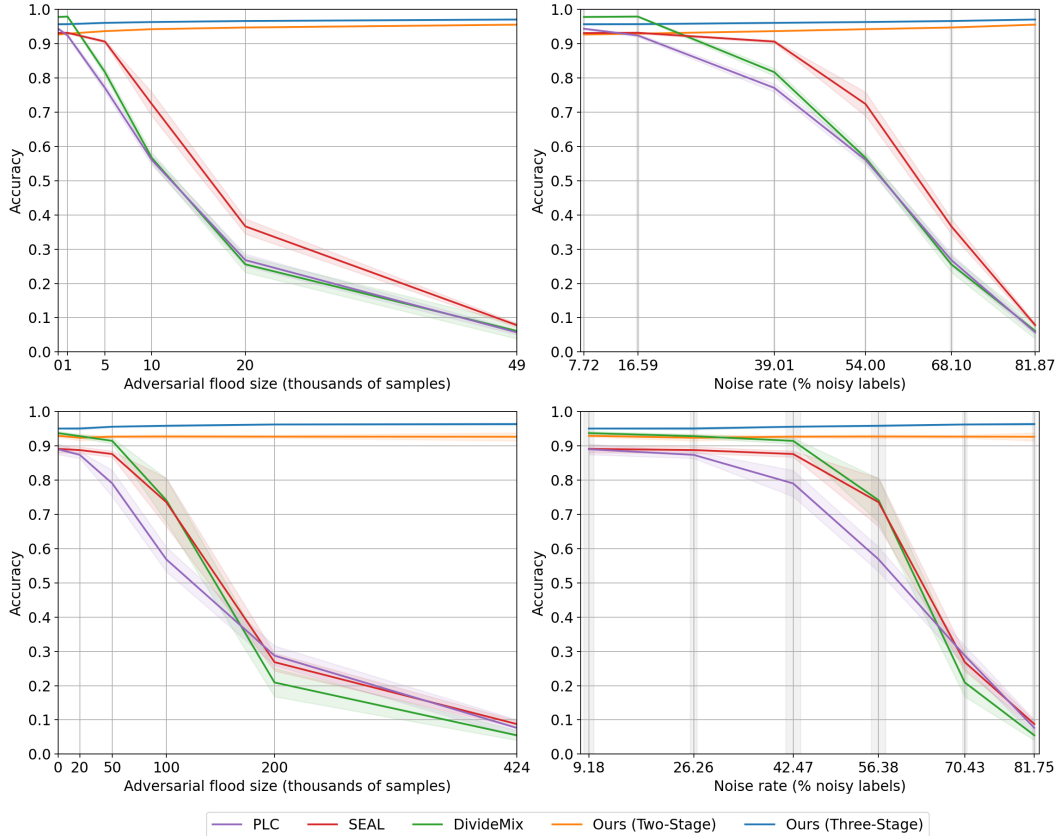
Figure 2: Classification accuracy under data flooding attacks evaluated on the MNIST (top row) and SVHN (bottom row) datasets. Shaded regions show the 95% confidence intervals based on the two-sided Student's t-test. **Left column:** Horizontal axis scaled to the adversarial attack size. **Right column:** Horizontal axis scaled to the approximate label noise rate. Vertical grey bars indicate standard deviation about the mean of experimental noise rates.

Figure 2 shows the classification accuracies of each algorithm in response to increasing amounts of label noise from a single adversary. We observe that all three state-of-the-art algorithms for learning from noisy labels fail under increasing levels of adversarial noise. In contrast, our labeler-aware approach remains robust even under extreme adversarial label noise. These results indicate that recognizing the multiple-labeler paradigm of label gathering is critical in designing robust algorithms for learning from noisy labels. We also note the ablative comparisons between the naïve application of the DivideMix algorithm, our two-stage framework without stacking, and our three-stage framework that stacks DivideMix on top of our bootstrapped filtered labels. While labeler-agnostic DivideMix is vulnerable to adversarial noise, and our two-stage labeler-aware framework is robust to adversarial labels but suffers from natural noise, our complete three-stage framework successfully achieves robust performance against both types of noise.

Multiple adversaries For the MNIST dataset, each of the ten labelers provided $N_j = 5,800$ noisy labels, and the number of adversaries A was varied between 0 and 9. For the SVHN dataset, each of the five labelers provided $N_j = 100,000$ noisy labels, and A was varied between 0 and 4. For multiple adversaries experiments, the total amount of data visible to the learner is fixed.

Figure 3 shows the classification accuracies of each algorithm in response to increasing amounts of adversarial noise caused by multiple adversaries. We observe that our labeler-aware framework remains robust against larger fractions of adversarial labelers compared to other methods. Naturally, all algorithms, including ours, fail under extreme numbers of adversarial labelers, a well-known phenomenon from ensemble learning that can be traced back to the Condorcet jury theorem [19] and its modern extensions [38, 27, 63, 49].
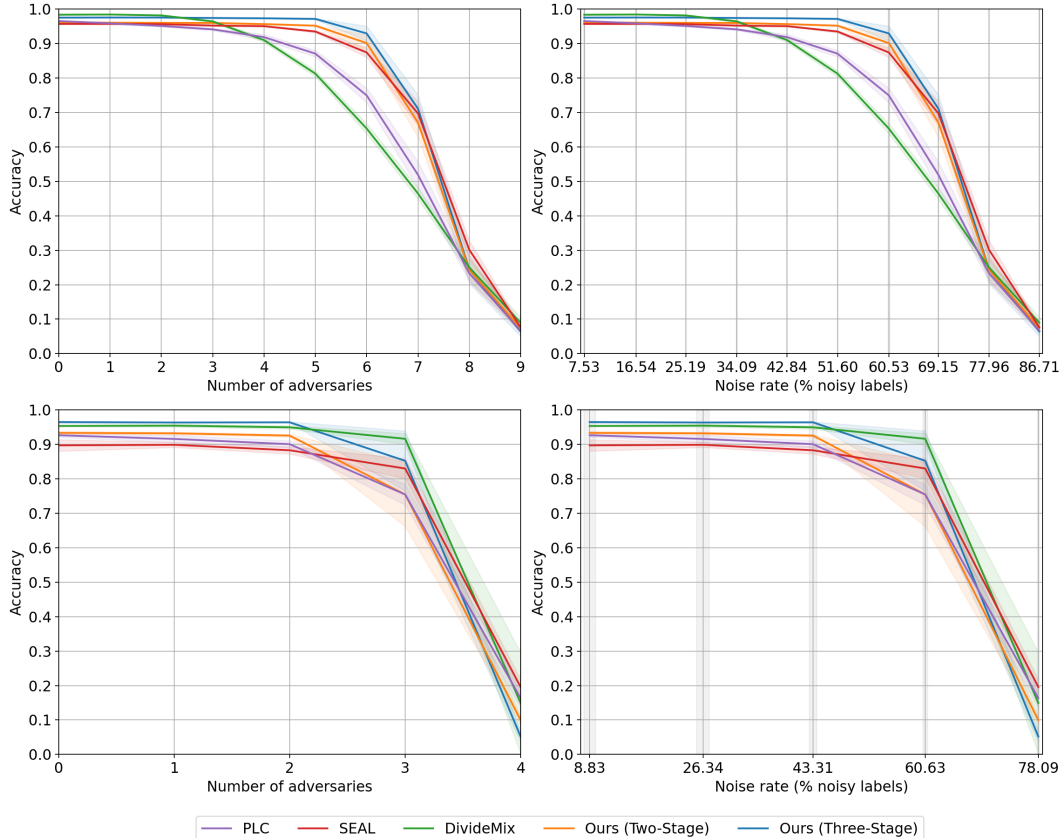
7

Figure 3: Classification accuracy under multiple adversaries attacks evaluated on the MNIST (top row) and SVHN (bottom row) datasets. Shaded regions show the 95% confidence intervals based on the two-sided Student's t-test. **Left column:** Horizontal axis scaled to the adversarial attack size. **Right column:** Horizontal axis scaled to the approximate label noise rate. Vertical grey bars indicate standard deviation about the mean of experimental noise rates.

6 Limitations

As discussed above, an important limitation of our labeler-aware framework is its inability to handle extreme fractions of adversarial labelers. This difficulty is well-known in the context of learning from crowds [114, 96, 36], and remains an area of open research.

We also note that while our multimodal noisy labels model is more principled than transition matrix-based models, the label noise generation procedure used in our experiments restricts the datasets on which comparisons can be made. For example, while the CIFAR-10 dataset is a very popular dataset, it is too small in relation to its difficulty to apply our experimental procedure. This is because there is no amount of labeler training data $\mathcal{D}_j^{tr}$ (drawn without replacement from the full training dataset) that would allow us to train the labeler models θ_j to a reasonable level of accuracy while still retaining enough data to provide observable labels to the algorithms under test. Any dataset on which experiments are performed under our labeler-dependent noise model must be either simple enough to learn to a reasonable degree of proficiency on a small subset of the training data (such as the MNIST dataset), or large enough that a greater absolute number of examples could be sampled without replacement to train the labeler models (such as the SVHN dataset). We note, however, that the severe impact of adversarial noise is still highly damaging to state-of-the-art approaches even on simple datasets, so these datasets remain valuable as testing environments.

Finally, we acknowledge that our multi-stage framework for learning from multiple labelers requires sufficient computational resources. Stage 1 models may however be trained in parallel, saving on

8

time expenditure; alternatively, an online label gathering environment allows for pipelined training of semi-supervised labeler models as labelers submit their data.

7 Conclusion

In this work, we discussed the assumptions and limitations of conventional models of label noise, including transition matrix-based class-conditional noise, instance-dependent noise, and the presence of unreliable or adversarial labelers in the crowdsourcing setting. We propose a more realistic, labeler-dependent characterization of label noise that integrates both natural error and adversarial labels, acknowledging the reality of multiple-labeler practices for label gathering. We define two vectors of adversarial label attacks, and propose to evaluate the robustness of noisy label algorithms as their vulnerability to adversarial label noise. Finally, we present a labeler-aware, multi-stage, modular framework for learning under our multimodal label noise model, and demonstrate our framework's superior robustness against more challenging adversarial noise when compared against state-of-the-art methods for learning from noisy labels.

8 Broader Impacts

Future research directions   The ubiquity of label noise in real-world data cannot be overstated, a fact that is reflected in the increasingly large body of work focused on mitigating its effects. While conventional models of class-conditional label noise have made great strides, recent work on instance-dependent noise has rightfully pointed out that real-world noise is not necessarily class-conditional. Our extension of this observation from feature-awareness to labeler-awareness marries the parallel fields of learning from noisy labels and learning from crowds. This marriage, along with the natural integration of adversarial threats, should form the basis for future research in this critical area.

Adversarial threat modeling   As with any work proposing consideration of novel attack vectors, the types of attacks described in this work pose serious and potentially destabilizing threats to systems that ignore the threat of adversarial labelers. There is real-world precedent for the types of attacks we describe: Microsoft's infamous Twitter chatbot Tay [97], its successor Zo [78], and even commonplace home assistant tools such as Google Home and Amazon Echo [110] have all been subject to malicious data flooding attacks from multiple adversaries. By codifying these threats and proposing methods for addressing them, our work serves as a preliminary effort in proactively establishing robust defensive measures against adversarial label attacks.

Labeler-dependent biases   Labeler-dependent biases may exist as natural label noise based on the characteristics of the data on which a labeler learned their beliefs. In the case of machine labelers, these characteristics will reflect the biases that may be present in the model's training data. In the case of human labelers, this "training data" may take the form of the social, cultural, economic, or educational backgrounds and experiences that inform their beliefs. Our proposed methods do not leverage these biases, and in fact attempt to minimize the effect of such biases on the downstream classifier.

Practical applications   We foresee that our work may be of great benefit for industrial applications, where large volumes of noisily-labeled data are commonplace. We also envision our framework seamlessly integrating into future work on continual or online learning as autonomous agents require the ability to synthesize information from multiple unreliable data sources.

Labeler-aware datasets   Finally, we hope to initiate an awareness and a shift in how labeled data are gathered, with datasets retaining the information about which data were labeled by which, and by how many, labelers. With recent work discussing labeling errors in commonly-used benchmark datasets [7, 70, 62, 61], we are convinced that labeler-aware approaches such as ours are better equipped to more reliably produce accurate labels, identify mislabeled instances, and promote further study and refinement of label quality in standard benchmarks.

Acknowledgments and Disclosure of Funding

References

[1] G. Algan and I. Ulusoy. Image classification with deep learning in the presence of noisy labels: A survey. *Knowledge-Based Systems*, 215:106771, 2021.

[2] A. Alush and J. Goldberger. Ensemble segmentation using efficient integer linear programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(10):1966–1977, 2012. doi: 10.1109/TPAMI.2011.280.

[3] E. Amid, M. K. Warmuth, and S. Srinivasan. Two-temperature logistic regression based on the tsallis divergence. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 2388–2396. PMLR, 16–18 Apr 2019.

[4] E. Amid, M. K. K. Warmuth, R. Anil, and T. Koren. Robust bi-tempered logistic loss based on bregman divergences. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[5] D. Angluin and P. Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

[6] E. Arazo, D. Ortego, P. Albert, N. O'Connor, and K. McGuinness. Unsupervised label noise modeling and loss correction. In *International Conference on Machine Learning*, pages 312–321. PMLR, 2019.

[7] B. Barz and J. Denzler. Do we train on test data? purging cifar of near-duplicates. *Journal of Imaging*, 6 (6):41, 2020.

[8] A. J. Bekker and J. Goldberger. Training deep neural-networks based on unreliable labels. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2682–2686. IEEE, 2016.

[9] B. Biggio, B. Nelson, and P. Laskov. Support vector machines under adversarial label noise. In *Asian Conference on Machine Learning*, pages 97–112. PMLR, 2011.

[10] A. Checco, J. Bates, and G. Demartini. Adversarial attacks on crowdsourcing quality control. *Journal of Artificial Intelligence Research*, 67:375–408, 2020.

[11] P. Chen, B. B. Liao, G. Chen, and S. Zhang. Understanding and utilizing deep neural networks trained with noisy labels. In *International Conference on Machine Learning*, pages 1062–1070. PMLR, 2019.

[12] P. Chen, J. Ye, G. Chen, J. Zhao, and P.-A. Heng. Beyond class-conditional assumption: A primary attempt to combat instance-dependent label noise. *arXiv preprint arXiv:2012.05458*, 2020.

[13] X. Chen, Q. Lin, and D. Zhou. Optimistic knowledge gradient policy for optimal budget allocation in crowdsourcing. In *International Conference on Machine Learning*, pages 64–72. PMLR, 2013.

[14] J. Cheng, T. Liu, K. Ramamohanarao, and D. Tao. Learning with bounded instance and label-dependent label noise. In *International Conference on Machine Learning*, pages 1789–1799. PMLR, 2020.

[15] C. F. da Silva and T. A. Hare. Humans primarily use model-based inference in the two-stage task. *Nature Human Behaviour*, 4(10):1053–1066, 2020.

[16] N. Dalvi, A. Dasgupta, R. Kumar, and V. Rastogi. Aggregating crowdsourced binary ratings. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 285–294, 2013.

[17] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28(1):20–28, 1979.

[18] G. Dawson and R. Polikar. Opinionrank: Extracting ground truth labels from unreliable expert opinions with graph-based spectral ranking. *arXiv preprint arXiv:2102.05884*, 2021.

[19] M. J. A. N. de Caritat Marquis de Condorcet. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. L'imprimerie royale, 1785.

[20] P. Donmez, J. G. Carbonell, and J. Schneider. Efficiently learning the accuracy of labeling sources for selective sampling. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 259–268, 2009.

[21] P. Donmez, J. Carbonell, and J. Schneider. A probabilistic framework to learn from multiple annotators with time-varying accuracy. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 826–837. SIAM, 2010.

[22] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, 2017.

[23] B. Frénay and M. Verleysen. Classification in the presence of label noise: a survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869, 2013.

[24] C. D. Frith. The role of metacognition in human social interactions. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 367(1599):2213–2223, 2012.

[25] L. P. Garcia, J. Lehmann, A. C. de Carvalho, and A. C. Lorena. New label noise injection methods for the evaluation of noise filters. *Knowledge-Based Systems*, 163:693–704, 2019.

[26] A. Ghosh, S. Kale, and P. McAfee. Who moderates the moderators? crowdsourcing abuse detection in user-generated content. In *Proceedings of the 12th ACM Conference on Electronic Commerce*, pages 167–176, 2011.

[27] B. Grofman. A comment on 'democratic theory: A preliminary mathematical model.'. *Public Choice*, 21 (1):99–103, 1975.

[28] V. Gulshan, L. Peng, M. Coram, M. C. Stumpe, D. Wu, A. Narayanaswamy, S. Venugopalan, K. Widner, T. Madams, J. Cuadros, et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *JAMA*, 316(22):2402–2410, 2016.

[29] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, and M. Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[30] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[31] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[32] R. J. Hickey. Noise modelling and evaluating learning from examples. *Artificial Intelligence*, 82(1-2): 157–179, 1996.

[33] L. Huang, C. Zhang, and H. Zhang. Self-adaptive training: beyond empirical risk minimization. In *Advances in Neural Information Processing Systems*, volume 33, pages 19365–19376. Curran Associates, Inc., 2020.

[34] S. Jagabathula, L. Subramanian, and A. Venkataraman. Identifying unreliable and adversarial workers in crowdsourced labeling tasks. *The Journal of Machine Learning Research*, 18(1):3233–3299, 2017.

[35] J. Joiner, M. Piva, C. Turrin, and S. W. Chang. Social learning through prediction error in the brain. *NPJ science of learning*, 2(1):1–9, 2017.

[36] H. Kajino, Y. Tsuboi, and H. Kashima. A convex formulation for learning from crowds. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, 2012.

[37] D. Karimi, H. Dou, S. K. Warfield, and A. Gholipour. Deep learning with noisy labels: Exploring techniques and remedies in medical image analysis. *Medical Image Analysis*, 65:101759, 2020. ISSN 1361-8415.

[38] R. G. Kazmann. Democratic organization: A preliminary mathematical model. *Public Choice*, 16(1): 17–26, 1973.

[39] K. Khalvati, S. A. Park, S. Mirbagheri, R. Philippe, M. Sestito, J.-C. Dreher, and R. P. Rao. Modeling other minds: Bayesian inference explains human choices in group decision-making. *Science Advances*, 5 (11):eaax8783, 2019.

[40] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[41] M. Kleindessner and P. Awasthi. Crowdsourcing with arbitrary adversaries. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2708–2717. PMLR, 10–15 Jul 2018.

[42] Á. M. Kovács, E. Téglás, and A. D. Endress. The social sense: Susceptibility to others' beliefs in human infants and adults. *Science*, 330(6012):1830–1834, 2010.

[43] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[44] P. A. Lachenbruch. Discriminant analysis when the initial samples are misclassified ii: non-random misclassification models. *Technometrics*, 16(3):419–424, 1974.

[45] Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010.

[46] J. Li, Y. Wong, Q. Zhao, and M. S. Kankanhalli. Learning to learn from noisy labeled data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5051–5059, 2019.

[47] J. Li, R. Socher, and S. C. Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. In *International Conference on Learning Representations*, 2020.

[48] W. Li, L. Wang, W. Li, E. Agustsson, and L. Van Gool. Webvision database: Visual learning and understanding from web data. *arXiv preprint arXiv:1708.02862*, 2017.

[49] C. List and R. E. Goodin. Epistemic democracy: Generalizing the condorcet jury theorem. *J. Political Philos.*, 9(3):277–306, 2001. doi: https://doi.org/10.1111/1467-9760.00128.

[50] Q. Liu, J. Peng, and A. T. Ihler. Variational inference for crowdsourcing. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[51] S. Liu, J. Niles-Weed, N. Razavian, and C. Fernandez-Granda. Early-learning regularization prevents memorization of noisy labels. *Advances in Neural Information Processing Systems*, 33, 2020.

[52] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

[53] Q. Ma and A. Olshevsky. Adversarial crowdsourcing through robust rank-one matrix completion. *Advances in Neural Information Processing Systems*, 33, 2020.

[54] X. Ma, Y. Wang, M. E. Houle, S. Zhou, S. Erfani, S. Xia, S. Wijewickrema, and J. Bailey. Dimensionality-driven learning with noisy labels. In *International Conference on Machine Learning*, pages 3355–3364. PMLR, 2018.

[55] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. Auxiliary deep generative models. In *International Conference on Machine Learning*, pages 1445–1453. PMLR, 2016.

[56] N. Manwani and P. Sastry. Noise tolerance under risk minimization. *IEEE Transactions on Cybernetics*, 43(3):1146–1151, 2013.

[57] A. K. Menon, A. S. Rawat, S. Kumar, and S. Reddi. Can gradient clipping mitigate label noise? In *International Conference on Learning Representations (ICLR)*, 2020.

[58] B. Mirzasoleiman, K. Cao, and J. Leskovec. Coresets for robust training of deep neural networks against noisy labels. In *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc., 2020.

[59] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari. Learning with noisy labels. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[60] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[61] C. Northcutt, L. Jiang, and I. Chuang. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, 70:1373–1411, 2021.

[62] C. G. Northcutt, A. Athalye, and J. Mueller. Pervasive label errors in test sets destabilize machine learning benchmarks. *arXiv preprint arXiv:2103.14749*, 2021.

[63] G. Owen, B. Grofman, and S. L. Feld. Proving a distribution-free generalization of the condorcet jury theorem. *Math. Soc. Sci.*, 17(1):1–16, 1989.

[64] S. A. Park, S. Goïame, D. A. O'Connor, and J.-C. Dreher. Integration of individual and social information for decision-making in groups of different sizes. *PLoS biology*, 15(6):e2001958, 2017.

[65] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[66] G. Patrini, A. Rozza, A. Krishna Menon, R. Nock, and L. Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1944–1952, 2017.

[67] G. Pleiss, T. Zhang, E. Elenberg, and K. Q. Weinberger. Identifying mislabeled data using the area under the margin ranking. In *Advances in Neural Information Processing Systems*, volume 33, pages 17044–17056. Curran Associates, Inc., 2020.

[68] V. C. Raykar, S. Yu, L. H. Zhao, A. K. Jerebko, C. Florin, G. H. Valadez, L. Bogoni, and L. Moy. Supervised learning from multiple experts: whom to trust when everyone lies a bit. In *ICML*, 2009.

[69] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *Journal of Machine Learning Research*, 11(4), 2010.

[70] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pages 5389–5400. PMLR, 2019.

[71] S. E. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich. Training deep neural networks on noisy labels with bootstrapping. In *ICLR 2015*, 2015.

[72] H. Reeve et al. Classification with unknown class-conditional label noise on non-compact feature spaces. In *Conference on Learning Theory*, pages 2624–2651. PMLR, 2019.

[73] M. Ren, W. Zeng, B. Yang, and R. Urtasun. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning*, pages 4334–4343. PMLR, 2018.

[74] F. Rodrigues, F. Pereira, and B. Ribeiro. Learning from multiple annotators: distinguishing good from random labelers. *Pattern Recognition Letters*, 34(12):1428–1436, 2013.

[75] D. Roman. Crowdsourcing and the question of expertise. *Communications of the ACM*, 52(12):12–12, 2009.

[76] F. Schroff, A. Criminisi, and A. Zisserman. Harvesting image databases from the web. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4):754–766, 2010.

[77] C. Scott, G. Blanchard, and G. Handy. Classification with asymmetric label noise: Consistency and maximal denoising. In *Conference on Learning Theory*, pages 489–511. PMLR, 2013.

[78] J. Seering, M. Luria, G. Kaufman, and J. Hammer. Beyond dyadic interactions: Considering chatbots as community members. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2019.

[79] V. S. Sheng, J. Zhang, B. Gu, and X. Wu. Majority voting and pairing with multiple noisy labeling. *IEEE Transactions on Knowledge and Data Engineering*, 31(7):1355–1368, 2017.

[80] L. N. Smith and N. Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, page 1100612. International Society for Optics and Photonics, 2019.

[81] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. Cheap and fast—but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, page 254–263, 2008.

[82] K. Sohn, D. Berthelot, N. Carlini, Z. Zhang, H. Zhang, C. A. Raffel, E. D. Cubuk, A. Kurakin, and C.-L. Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *Advances in Neural Information Processing Systems*, volume 33, pages 596–608. Curran Associates, Inc., 2020.

[83] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. Ladder variational autoencoders. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[84] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee. Learning from noisy labels with deep neural networks: A survey. *arXiv preprint arXiv:2007.08199*, 2020.

[85] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, and R. Fergus. Training convolutional networks with noisy labels. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[86] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013.

[87] D. Tanaka, D. Ikami, T. Yamasaki, and K. Aizawa. Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5552–5560, 2018.

[88] R. Tanno, A. Saeedi, S. Sankaranarayanan, D. C. Alexander, and N. Silberman. Learning from noisy labels by regularized estimation of annotator confusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11244–11253, 2019.

[89] F. Tao, L. Jiang, and C. Li. Label similarity-based weighted soft majority voting and pairing for crowdsourcing. *Knowledge and Information Systems*, 62:2521–2538, 2020.

[90] S. Thulasidasan, T. Bhattacharya, J. Bilmes, G. Chennupati, and J. Mohd-Yusof. Combating label noise in deep learning using abstention. In *International Conference on Machine Learning*, pages 6234–6243. PMLR, 2019.

[91] J. W. Vaughan. Making better use of the crowd: How crowdsourcing can advance machine learning research. *Journal of Machine Learning Research*, 18(1):7026–7071, 2017.

[92] Q. Wang, B. Han, T. Liu, G. Niu, J. Yang, and C. Gong. Tackling instance-dependent label noise via a universal probabilistic model. *arXiv preprint arXiv:2101.05467*, 2021.

[93] Z. Wang, G. Hu, and Q. Hu. Training noise-robust deep neural networks via meta-learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4524–4533, 2020.

[94] J. Wei and Y. Liu. When optimizing f-divergence is robust with label noise. *ICLR*, 2021.

[95] P. Welinder, S. Branson, P. Perona, and S. Belongie. The multidimensional wisdom of crowds. *Advances in Neural Information Processing Systems*, 23:2424–2432, 2010.

[96] J. Whitehill, T.-f. Wu, J. Bergsma, J. Movellan, and P. Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. *Advances in Neural Information Processing Systems*, 22:2035–2043, 2009.

[97] M. J. Wolf, K. W. Miller, and F. S. Grodzinsky. Why we should have seen that coming: comments on microsoft's tay "experiment," and wider implications. *The ORBIT Journal*, 1(2):1–12, 2017.

[98] P. Wu, S. Zheng, M. Goswami, D. Metaxas, and C. Chen. A topological filter for learning with label noise. In *Advances in Neural Information Processing Systems*, volume 33, pages 21382–21393. Curran Associates, Inc., 2020.

[99] X. Xia, T. Liu, B. Han, N. Wang, M. Gong, H. Liu, G. Niu, D. Tao, and M. Sugiyama. Part-dependent label noise: Towards instance-dependent label noise. In *Advances in Neural Information Processing Systems*, volume 33, pages 7597–7610. Curran Associates, Inc., 2020.

[100] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2691–2699, 2015.

[101] Y. Yan, R. Rosales, G. Fung, R. Subramanian, and J. Dy. Learning from multiple annotators with varying expertise. *Machine Learning*, 95(3):291–327, 2014.

[102] Y. Yao, T. Liu, B. Han, M. Gong, J. Deng, G. Niu, and M. Sugiyama. Dual t: Reducing estimation error for transition matrix in label-noise learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 7260–7271. Curran Associates, Inc., 2020.

[103] K. Yi and J. Wu. Probabilistic end-to-end noise correction for learning with noisy labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7017–7025, 2019.

[104] X. Yu, B. Han, J. Yao, G. Niu, I. Tsang, and M. Sugiyama. How does disagreement help generalization against label corruption? In *International Conference on Machine Learning*, pages 7164–7173. PMLR, 2019.

[105] S. Zagoruyko and N. Komodakis. Wide residual networks. In *BMVC*, 2016.

[106] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.

[107] J. Zhang and X. Wu. Multi-label inference for crowdsourcing. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2738–2747, 2018.

[108] J. Zhang, X. Wu, and V. S. Sheng. Learning from crowdsourced labeled data: a survey. *Artificial Intelligence Review*, 46(4):543–576, 2016.

[109] J. Zhang, V. S. Sheng, T. Li, and X. Wu. Improving crowdsourced label quality using noise correction. *IEEE Transactions on Neural Networks and Learning Systems*, 29(5):1675–1688, 2017.

[110] J. Zhang, B. Zhang, and B. Zhang. Defending adversarial attacks on cloud-aided automatic speech recognition systems. In *Proceedings of the Seventh International Workshop on Security in Cloud Computing*, pages 23–31, 2019.

[111] Y. Zhang, X. Chen, D. Zhou, and M. I. Jordan. Spectral methods meet em: A provably optimal algorithm for crowdsourcing. *Advances in Neural Information Processing Systems*, 27:1260–1268, 2014.

[112] Y. Zhang, S. Zheng, P. Wu, M. Goswami, and C. Chen. Learning with feature dependent label noise: a progressive approach. *ICLR*, 2021.

[113] Z. Zhang and M. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[114] J. Zhong, P. Yang, and K. Tang. A quality-sensitive method for learning from crowds. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2643–2654, 2017.

[115] D. Zhou, Q. Liu, J. C. Platt, and C. Meek. Aggregating ordinal labels from crowds by minimax conditional entropy. In *International Conference on Machine Learning*, 2014.

[116] Z. Zhu, T. Liu, and Y. Liu. A second-order approach to learning with instance-dependent label noise. *arXiv preprint arXiv:2012.11854*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 8.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Supplemental Materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5 and Supplemental Materials.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 5, we report 95% confidence intervals for Student's t-test over multiple repeated experiments.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Supplemental Materials.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] See Supplemental Materials.

(b) Did you mention the license of the assets? [Yes] See Supplemental Materials.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See Supplemental Materials.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A   Additional remarks on labeler-dependent noise

A.1   Labeler-dependent noise is a general form of instance-dependent noise

Consider instance-dependent noise models that assume a single label noise generation process for the entire training dataset (such as those described in Section 2 of the main paper). For these models, a noisy label $\hat{y}_i$ for instance $\boldsymbol{x}_i$ can be said to be generated following

$$\hat{y}_i \sim \text{Cat}(\mathbf{Y}|\boldsymbol{\pi}_i) \tag{2}$$

where $\boldsymbol{\pi}_i = h(\boldsymbol{x}_i)$ is an instance-dependent probability distribution characterized by the generating dynamics h of the instance-dependent noise model. By adding a labeler dependency following

$$\hat{y}_{ij} \sim \text{Cat}(\mathbf{Y}|\boldsymbol{\pi}_{ij}) \tag{3}$$

with $\boldsymbol{\pi}_{ij} = h_j(\boldsymbol{x}_i)$, we obtain the formulation of labeler-dependent noise defined in Section 3 of the main paper. Instance-dependent noise of any form can therefore be considered a special case of labeler-dependent noise where $J = 1$. We note that while for this work we choose $h_j = f_{\theta_j}$, our labeler-dependent noise model holds without loss of generality even if h is chosen to be a different process.

A.2   On the importance of labeler awareness

Assume the existence of a theoretical mapping $\mathcal{F}_{\theta^*} : \mathcal{X}^* \mapsto \mathbf{Y}$, where $\mathcal{X}^*$ is the set of all possible data and $\mathbf{Y}$ is the label space, parameterized by an optimal set of parameters θ^*. Consider that each labeler j generates labels following the parameters θ_j, where θ_j was learned on an unobserved training dataset $\mathcal{D}_j^{tr} = \{\mathcal{X}_j^{tr} \subsetneq \mathcal{X}^*, \mathcal{Y}_j^{tr}\}$. θ_j represents a mapping $\mathcal{F}_{\theta_j} : \mathcal{X}^* \mapsto \mathbf{Y}$ that approximates, but does not equal, $\mathcal{F}_{\theta^*}$. We observe that for any pair of labelers $\{j, \ell \mid j \neq \ell\}$, under the assumption that $\theta_j \neq \theta_\ell$, the mappings of each labeler will differ, $\mathcal{F}_{\theta_j} \neq \mathcal{F}_{\theta_\ell}$. We note that differences between θ_j and θ_ℓ may arise due to different parameterizations, differing $\mathcal{D}_j^{tr} \neq \mathcal{D}_\ell^{tr}$, or both.

Suppose that the data partitions on which each labeler provides data are sampled randomly from $\mathcal{X}^*$ such that $\mathcal{X}_j \neq \mathcal{X}_\ell$ and $\{\mathcal{X}_j, \mathcal{X}_\ell\} \subsetneq \mathcal{X}^*$. The labels of the data concatenation $\mathcal{X} = \{\mathcal{X}_j, \mathcal{X}_\ell\}$ are therefore generated as a mixture model $\mathcal{F}$, which includes information from both $\mathcal{F}_{\theta_j}$ and $\mathcal{F}_{\theta_\ell}$. Without knowledge of which data were labeled by which labeler, the agent being trained on $\mathcal{X}$ is attempting to learn the mixture model $\mathcal{F}$. Under the assumption that either or both of j or ℓ may be adversarial, this mixture model may deviate strongly from $\mathcal{F}_{\theta^*}$, forcing the agent to learn extreme bias.

By retaining information about which labeler provided labels on which data, the agent has the ability to estimate the individual labeler mappings (as in Stage 1 of our proposed framework). Because each labeler mapping is learned using only the labels provided by that labeler, the damage caused by an adversarial labeler is restricted to only the learner's estimate of the adversary's mapping, thus minimizing the adversary's potential impact.

A.3   Examples of labeler-dependent noise

Visual examples of instance-dependent noise removing the class-conditional assumption for label noise by generating feature-dependent labels can be found in previous works, such as [12]. Here, we illustrate labeler-dependent noise as a generalization of instance-dependent noise. We trained ten labelers identically as described in Appendix C.2, with seven labelers being good-faith and three being adversarial. In Table 1, we see how the labels generated by each labeler vary widely as a function of both the random initializations as well as the data $\mathcal{X}_j^{tr}$ on which was labeler was trained. Furthermore, even within just the pool of good-faith labelers, there is a wide variability between labelers with respect to the incorrect label each labeler provides; if label noise were simply instance-dependent, then we would expect to see the same incorrect label provided by all labelers. We note that the labels provided by adversarial labelers are not random, but rather represent the adversary's best guess about which label will cause maximum confusion to the learning agent; this "maximum confusion" approach is illustrated via the coincidence of adversarial labels with erroneous labels provided by good-faith labelers. Sometimes, however, the adversaries are wrong in their beliefs; in this run, we identified 1168 instances where an adversary would have accidentally provided the correct label.

Table 1: Examples of labeler-dependent noisy labels produced by seven good-faith labelers and three adversarial labelers on the MNIST dataset. The labels in each row were produced by the same labeler. Adversarial labels were generated by selecting the second-highest-confidence label based on the adversary's beliefs about the true labels. Correct labels in boldface; incorrect labels marked in red.

True label	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Good-faith labelers	1	**1**	9	**3**	9	**5**	2	3	**8**	**9**
	0	**1**	6	2	4	2	0	8	**8**	4
	6	4	5	8	4	6	**6**	4	4	**9**
	4	8	**2**	**3**	**4**	**5**	5	**7**	**8**	3
	0	3	**0**	**3**	2	0	**6**	**7**	9	7
	0	**1**	**2**	7	**4**	**5**	8	9	5	**9**
	7	8	**2**	**3**	1	**5**	**6**	**7**	**8**	7
Adversarial labelers	5	8	9	2	2	7	0	5	9	5
	6	6	9	7	9	6	2	8	9	7
	7	3	6	1	1	6	2	9	1	7

Table 2: Examples of labeler-dependent noisy labels produced by seven good-faith labelers and three adversarial labelers on the SVHN dataset. The labels in each row were produced by the same labeler. Adversarial labels were generated by selecting the second-highest-confidence label based on the adversary's beliefs about the true labels. Correct labels in boldface; incorrect labels marked in red.

True label	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Good-faith labelers	**0**	**1**	9	9	**4**	**5**	**6**	**7**	3	0
	0	3	2	3	**4**	**5**	**6**	**7**	0	**9**
	3	**1**	5	6	8	7	0	**7**	**8**	**9**
	6	7	1	5	2	**5**	**6**	3	**8**	4
	0	5	**2**	3	7	9	8	8	**8**	2
	5	**1**	**2**	3	0	0	9	**7**	6	**9**
	9	6	3	8	**4**	3	5	9	5	0
Adversarial labelers	6	8	7	2	2	3	4	2	1	4
	5	4	7	9	0	3	8	1	6	0
	3	4	3	5	1	6	8	2	5	8

Table 2 shows an example of labels produced on the SVHN dataset following the same procedure as for Table 1.

B   Visualization and remarks on our three-stage learning framework

Our multi-stage learning framework is illustrated in Figure 4. We depict an example with $J = 3$ labelers; the framework extends easily to any number of labelers. After each labeler provides noisy labels $\hat{\mathcal{Y}}_j$ on their data partitions $\mathcal{X}_j$, in Stage 1 of our framework we construct artificial semi-supervised datasets by treating the labeled data from labelers $\ell \neq j$ as unlabeled. Each semi-supervised dataset is then used to train models $\phi_j \approx \theta_j$ by exploiting the labels only from labeler j. Each ϕ_j is then queried to obtain estimates of what labels $g_{\phi_j}(\mathcal{X})$ labeler j would provide for all available data $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_J\}$. Note that Stage 1 is completely parallel, and so can be completed efficiently given appropriate resources. In Stage 2, the full set of J labels for all available data are combined into filtered labels $\widetilde{\mathcal{Y}}$ using methods for learning from crowds. Finally, in Stage 3 we train a classifier Θ using the filtered (but still noisy) labels from Stage 2 using an algorithm for learning from noisy labels.
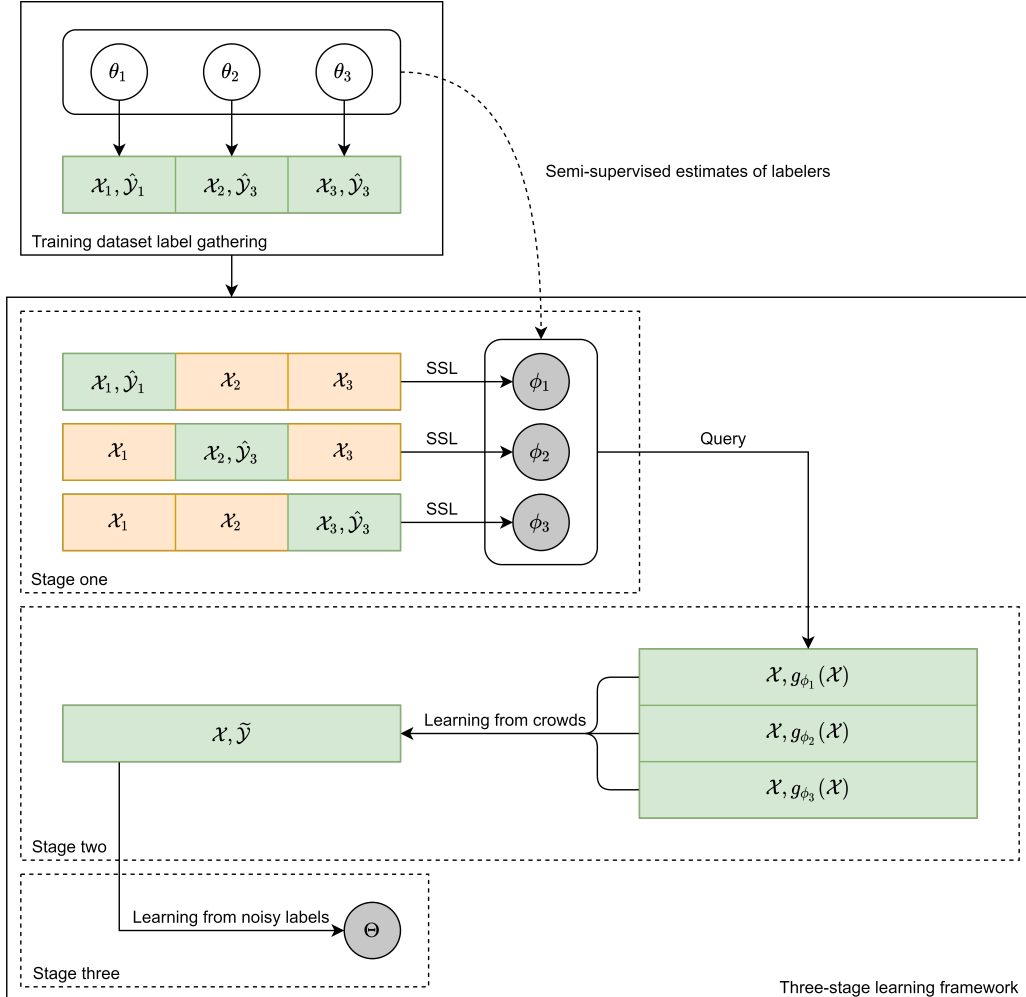
Figure 4: Three-stage learning framework for learning from multiple noisy labelers.

Note that for all three stages, there is no prescription on which specific algorithms must be used. In this sense, our framework is highly modular and adaptive to improvements and advances in each area. Furthermore, because each stage is self-contained, our framework is able to leverage the theoretical guarantees of each component algorithm achieve the objectives of each stage.

B.1 Remarks on semi-supervised learning of labeler models

In Stage 1 of our framework, we estimate the label generation dynamics θ_j of each labeler j by leveraging semi-supervised learning to train a representative model ϕ_j. The labeled data comprises the data-label pairs provided by labeler j directly, and the unlabeled data comprises the data (but not the labels) provided by every other labeler $\ell \neq j$. A critical consideration is that any label set $\hat{\mathcal{Y}}_j$ may be adversarial, and exhibit extreme label noise; therefore, in this case the θ_j that is being estimated by ϕ_j will be a poor model for learning a classifier. Our framework explicitly does not attempt to correct this potential label noise in Stage 1, because there is no way to know *a priori* whether a label set $\hat{\mathcal{Y}}_j$ is adversarial. Therefore, in the presence of adversarial label sources, it is almost surely guaranteed that the corresponding models ϕ_j will be poor general classifiers, as the label sets that they produce upon being queried in Stage 2 will exhibit labels of similar quality to those produced by θ_j.

B.2 Remarks on learning from crowds from queried labels

Methods for learning from crowds assume that the observed labels are provided directly by each labeler; we are not aware of any literature that attempts to impute missing labels as we do in Stage 1. We note that our framework subtly violates the parametric assumptions of many methods for learning from crowds by adding an additional layer of joint estimation (ϕ_j) between the original labeler and the queried labels. While we have remained intentionally agnostic regarding the parameters θ_j and ϕ_j, a considerable drawback of commonly-used approaches such as expectation-maximization [96, 74] or Bayesian inference [95] is their dependence on the correct modeling of the precise parameters being estimated. For this reason, we suggest using nonparametric approaches, such as weighted majority voting [79, 89] or OpinionRank [18].

B.3 Remarks on learning from bootstrapped noisy labels

In Stage 2, we filter adversarial label noise; in Stage 3, we bootstrap off of the filtered labels $\widetilde{\mathcal{Y}}$ generated in Stage 2 to further filter natural-error label noise. This treatment is supported by our empirical results, particularly the ablative comparisons between our framework without the third stage and our complete framework including bootstrapping (Section 5 of the main paper).

C Experimental details

In this section, we report the detailed parameters of each experiment, including the exact parameters of each algorithm.

C.1 Datasets

Our experiments investigating algorithmic robustness to data flooding and multiple adversaries attacks are performed on the MNIST [45] and SVHN [60] datasets. We would like to acknowledge some other datasets that are commonly used in literature for learning from noisy labels.

The CIFAR-10 dataset [43] comprises 50,000 training images and 10,000 test images in ten classes. While we attempted to use the CIFAR-10 dataset for our experiments (following the procedure in Appendix C.2), we found that we could not find an amount of data $\mathcal{D}_j^{tr}$ on which to train each labeler model θ_j to a reasonable level of accuracy (~90%) such that the remaining observable data $\mathcal{D} \setminus \{\bigcup_j \mathcal{D}_j^{tr}\}$ was sufficient to train a classifier. We consider this inability to be a limitation of our experimental procedure, as the CIFAR-10 dataset is a difficult problem to learn in the low-data regime. However, we emphasize that this experimental limitation does not extend to the broader impacts of our labeler-aware model, as the severe impact of adversarial label attacks is clearly evident on both the MNIST and SVHN datasets.

The Clothing1M dataset [100] comprises one million images of clothing in 14 classes. The images are scraped from websites of fashion retailers, and the labels contain a large, but unknown, amount of label noise. Because labels for this dataset were gathered using frequency analysis of the text surrounding each image, the dataset does not have any labeler information that can be used to fairly evaluate our proposed learning framework. Furthermore, because the amount of label noise is unknown, we cannot apply the experimental procedure described in Appendix C.2 to perform fair comparisons as the procedure assumes experimental availability of clean ground truth labels.. However, we note that although the attacks described in this work are not directly applicable to this type of label gathering, the methods used to obtain the labels for this dataset are vulnerable to adversarial attacks in the form of false descriptions in the text surrounding each image. We leave a further investigation of this attack vector for future work.

The WebVision dataset [48] comprises approximately 2.4 million images in 1,000 classes. The images were obtained by crawling from the Flickr website, as well as Google Images search. The labels for this dataset were assigned as the crawl queries that generated each image. Similarly to the Clothing1M dataset, because there is a large, unknown amount of label noise present in the dataset we cannot apply the experimental procedure described in Appendix C.2 to perform fair comparisons. Furthermore, this dataset's label-gathering procedure does not retain information about individual labelers, relying instead on the accuracy of the tagging systems of Flickr and Google Images. While the authors did use crowdsourcing methods to obtain "clean" labels for the validation and test sets,

the information about which labelers assigned which labels to which data has been lost. As with the Clothing1M dataset, we note the vulnerability of this procedure to adversarial false tagging attacks; we leave the study of this method of attack to future work.

C.2 Labeler modeling

Algorithm 4 Multiple-labeler multimodal label noise generation.

Inputs: J, the number of label sources; A, the number of adversarial labelers; N_{tr}, the number of data on which to train each source; N_j, the number of data on which labels were provided by each source $j \in J$; $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$, the full training dataset with ground truth labels; θ, the labeler model architecture

1: $\mathcal{D}_L \leftarrow$ Initialize the set of noisy label datasets
2: **for** each labeler $j = 1$ to J **do**
3:     Sample $\mathcal{D}_j^{tr} \subset \mathcal{D}$ with $|\mathcal{D}_j^{tr}| = N_{tr}$
4:     $\mathcal{D} \leftarrow \{\mathcal{D} \setminus \mathcal{D}_j^{tr}\}$ (remove labeler training data from pool)
5:     Train θ_j on $\mathcal{D}_j^{tr} = \{\mathcal{X}_j^{tr}, \mathcal{Y}_j^{tr}\}$
6:     Sample $\mathcal{D}_j \subset \mathcal{D}$ with $|\mathcal{D}_j| = N_j$
7:     $\mathcal{D} \leftarrow \{\mathcal{D} \setminus \mathcal{D}_j\}$ (remove labeler's provided data from pool)
8:     **if** $j \notin A$ **then**
9:         $\hat{\mathcal{Y}}_j \leftarrow \mathrm{argmax}_k \, f_{\theta_j}(\mathcal{X}_j)$ (select most confident class as good-faith label)
10:    **else**
11:        $\hat{\mathcal{Y}}_j \leftarrow \mathrm{arg_2max}_k \, f_{\theta_j}(\mathcal{X}_j)$ (select the second most confident class as adversarial label)
12:    **end if**
13:    $\mathcal{D}_L \leftarrow \{\mathcal{D}_L \cup \{\mathcal{X}_j, \hat{\mathcal{Y}}_j\}\}$
14: **end for**
Output: $\mathcal{D}_L$

For all experiments, labeler-dependent noisy labels were generated following Algorithm 4. For the MNIST dataset, each labeler model θ_j was trained from a randomly-initialized ResNet-18 [30]. For the SVHN dataset, each labeler model θ_j was trained from a randomly-initialized Wide ResNet-50 [105]. Both types of θ_j architectures were initialized using the implementation provided by PyTorch [65]. All θ_j models were trained for 30 epochs. All θ_j models were trained using the AdamW optimizer with default parameters [52]. The learning rate for all θ_j models followed the 1cycle policy [80] with a maximum learning rate of 0.003. For the MNIST dataset, the training dataset size for each θ_j was $N_{tr} = 200$, with a minibatch size of 10. For the SVHN dataset, the training dataset size for each θ_j was $N_{tr} = 20,000$, with a minibatch size of 512. All labeler hyperparameters for both datasets were tuned such that the accuracy of each θ_j would be approximately 90-95%. We note that because we discard both the θ_j models as well as the $\mathcal{D}_j^{tr}$ on which they were trained after obtaining their labels $\hat{\mathcal{Y}}_j$, tuning the hyperparameters for the labeler models does not constitute data leakage.

For the MNIST dataset, the total number of labelers was $J = 10$, which allowed each labeler to provide a reasonable amount of labels while still retaining a large pool of data for which the adversarial labeler could provide labels in the data flooding experiments. For the SVHN dataset, the total number of labelers was $J = 5$; due to the much larger amount of data required for each $\mathcal{D}_j^{tr}$, a smaller total number of labelers was required such that the adversarial labeler could still flood the dataset in the data flooding experiments.

C.2.1 Labeler accuracy distributions

To estimate the consistency of our simulated labelers, we trained 10,000 θ_j models on random samples from the MNIST training dataset. Each model was trained on 200 randomly-sampled examples, and we recorded the accuracy of each model as well as its adversarial counterpart on the MNIST test set. The distributions of the good-faith and adversarial labeler models are shown in Figure 5.

We observe that even without adversarial labels, the distribution of good-faith labelers trained on very small subsets of the training data do not generalize to 100% accuracy on the testing data. Furthermore, there is a relatively wide variance in the accuracy distribution, supporting our hypothesis that label
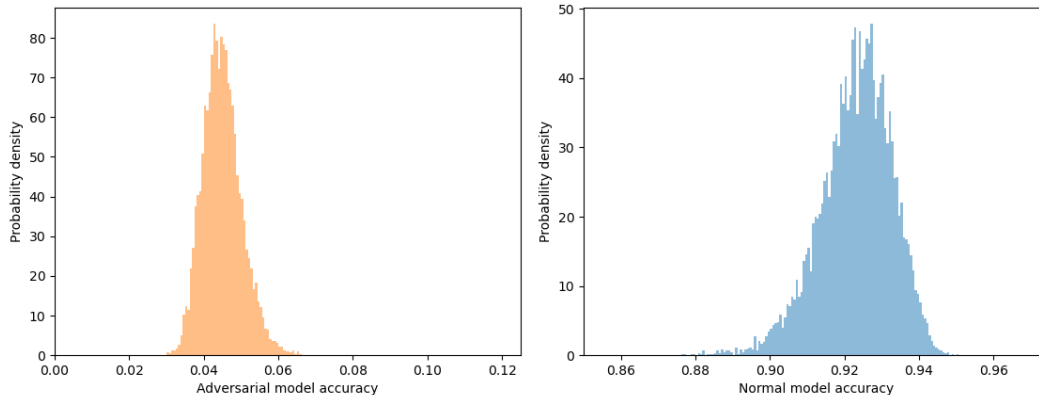
Figure 5: Experimental labeler model accuracy distributions for the MNIST dataset. **Left:** Accuracy distribution for adversarial labelers. **Right:** Accuracy distribution for good-faith labelers.

noise from multiple labelers will have more complex behavior than assuming a single labels generation process for the entire training dataset.

C.3 Implementation details and hyperparameters

Similarly to [6], we do not assume the existence of a cleanly-labeled validation set on which to perform hyperparameter tuning or model selection. Instead, as described below most hyperparameters were kept at the default suggested values found in the publicly-available implementations of each authors' algorithms. Generally, we only adjusted hyperparameters to speed up experimental procedures; in each case where this was done, we performed initial training runs to verify that the adjusted hyperparameters were fair. As in the main paper, we emphasize that due to our lack of hyperparameter fine-tuning, we are likely reporting slightly conservative scores for all algorithms, including our own. However, we are not interested in squeezing out optimal performance, but rather in obtaining representative evaluations of how each algorithm behaves in response to adversarial label noise.

C.3.1 Hyperparameters for MNIST experiments

All algorithms under test used the same parameters on both the data flooding and multiple adversaries experiments on the MNIST dataset. All experiments were repeated ten times using uncontrolled random seeds, and our results are reported in the main paper as the mean testing accuracy bounded by the 95% confidence interval computed using the two-sided Student's t-test. Sources of variability include the randomly-sampled data on which each θ_j are trained, the random initializations of each network's weights and biases, and any random data augmentation that may be performed as part of any algorithm under test.

A summary of algorithmic hyperparameters is shown in Table 3.

Progressive label correction (PLC)   For the PLC algorithm [112], we used the open-source implementation provided by the authors, available at `https://github.com/pxiangwu/PLC`. Following the authors' suggestions, the minibatch size was 128, and the base network was a randomly-initialized PreAct ResNet-34 [31]. The optimizer was SGD with Nesterov momentum [86], with an initial learning rate of 0.01, momentum of 0.9, and weight decay of 0.0005, and the warm-up period was 8 epochs. Our implementation differs from that of the original authors by training for 30 epochs, and performing learning rate annealing on the 20th epoch. These changes are due to the difference in dataset, and our preliminary experiments showed that continuing to train past 30 epochs did not lead to improved performance. Therefore, we truncated the training process to conserve computational resources and speed up experiments.

Self-evolution average label (SEAL)   For the SEAL algorithm [12], we used the open-source implementation provided by the authors, available at `https://github.com/chenpf1025/IDN`. Because the authors provide their code for training on the MNIST dataset, we used their suggested

Table 3: Hyperparameters for MNIST experiments. Parameters marked with a * follow the recommended settings of the original authors.

Algorithm	Parameter	Value
PLC [112]	Base model*	PreAct ResNet-34
	Minibatch size*	128
	Optimizer*	SGD with Nesterov momentum
	Initial LR*	0.01
	LR annealing	0.1 at epoch 20
	Momentum*	0.9
	Weight decay*	0.0005
	Warm-up*	8 epochs
	Epochs	30
SEAL [12]	Base model*	Custom CNN* (Table 4)
	Minibatch size*	64
	Optimizer*	SGD with momentum
	Initial LR*	0.01
	Momentum*	0.5
	Epochs per iteration*	50
	Iterations*	10
DivideMix [47]	Base model*	PreAct ResNet-18
	Minibatch size	128
	Optimizer*	SGD with momentum
	Initial LR*	0.02
	Momentum*	0.9
	Weight decay*	0.0005
	Epochs	30
Auxiliary DGM [55]	$\dim(a, z)^*$	100
	$\dim(h)^*$	600
	Minibatch size*	200
	Optimizer*	Adam
	Initial LR*	0.0003
	β_1, β_2 *	0.9, 0.999
	Classification loss weight*	$\frac{N_L + N_U}{N_L}$
	Warm up*	200 steps
	Epochs*	200

Table 4: Custom convolutional neural network used by SEAL for the MNIST dataset.

Layer	Parameters	Activation
2D convolution	20 channels, 5×5 filters	ReLU
2D max pool	2×2	–
2D convolution	50 channels, 5×5 filters	ReLU
2D max pool	2×2	–
Flatten	–	–
Fully-connected	500 output features	ReLU
Fully-connected	10 output features	Log softmax

hyperparameters. The minibatch size was 64, and the base model was a randomly-initialized custom convolutional neural network (Table 4). The optimizer was SGD with momentum, with an initial learning rate of 0.01 and momentum of 0.5. Each iteration was trained for 50 epochs, and the network was trained for 10 iterations.

DivideMix   For the DivideMix algorithm [47], we used the open-source implementation provided by the authors, available at `https://github.com/LiJunnan1992/DivideMix`. Following the authors' suggestions, the base networks were both a randomly-initialized PreAct ResNet-18 [31], and the optimizers were SGD with momentum, with an initial learning rate of 0.02, momentum of 0.9, and weight decay of 0.0005. Our implementation differs from that of the original authors by training for 30 epochs, with a minibatch size of 128. These changes are due to the difference in dataset, and our preliminary experiments showed that continuing to train past 30 epochs did not lead to improved performance. Therefore, we truncated the training process to conserve computational resources and speed up experiments.

Our framework   For our framework, we require the selection of three component algorithms to use for each of the three stages:

- We used auxiliary deep generative models [55] as our semi-supervised learning algorithm for estimating labeler models. We used the open-source implementation available at `https://github.com/wohlert/semi-supervised-pytorch`.

- We used OpinionRank [18] as our learning from crowds algorithm for integrating redundant labels. Our implementation is available in our supplemental code.

- We used DivideMix [47] as our algorithm for learning from noisy labels. We used the open-source implementation provided by the authors, available at `https://github.com/LiJunnan1992/DivideMix`.

Auxiliary deep generative models were randomly initialized with $\dim(z) = 100$, $\dim(h) = 600$, and $\dim(a) = 100$, and were trained for 200 epochs with a minibatch size of 200. The optimizer was Adam [40], with an initial learning rate of 0.0003 and default β parameters of $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The classification loss weight was $\alpha = \frac{N_L + N_U}{N_L}$, following [55]. A deterministic warm-up period of 200 steps was used, following [83].

OpinionRank is a nonparametric, deterministic algorithm, and so did not require any hyperparameter selection.

Our usage of DivideMix as the third stage of our framework was implemented identically as described above.

C.3.2   Hyperparameters for SVHN experiments

All algorithms under test used the same parameters on both the data flooding and multiple adversaries experiments on the SVHN dataset. All experiments were repeated five times using uncontrolled random seeds, and our results are reported in the main paper as the mean of these results bounded by the 95% confidence interval computed using the two-sided Student's t-test. Sources of variability include the randomly-sampled data on which each θ_j are trained, the random initializations of each network's weights and biases, and any random data augmentation that may be performed as part of any algorithm under test. We combined the standard set of 73,257 training examples with the extra set of 531,131 additional training examples to use as the training dataset $\mathcal{D}$.

A summary of algorithmic hyperparameters is shown in Table 5.

Progressive label correction (PLC)   For the PLC algorithm [112], we used the open-source implementation provided by the authors, available at `https://github.com/pxiangwu/PLC`. Following the authors' suggestions, the minibatch size was 128, and the base network was a randomly-initialized PreAct ResNet-34 [31]. The optimizer was SGD with Nesterov momentum [86], with an initial learning rate of 0.01, momentum of 0.9, and weight decay of 0.0005, and the warm-up period was 8 epochs. Our implementation differs from that of the original authors by training for 30 epochs, and performing learning rate annealing on the 20th epoch. These changes are due to the difference in dataset, and our preliminary experiments showed that continuing to train past 30 epochs did not lead

Table 5: Hyperparameters for SVHN experiments. Parameters marked with a * follow the recommended settings of the original authors.

Algorithm	Parameter	Value
PLC [112]	Base model*	PreAct ResNet-34
	Minibatch size*	128
	Optimizer*	SGD with Nesterov momentum
	Initial LR*	0.01
	LR annealing	0.1 at epoch 20
	Momentum*	0.9
	Weight decay*	0.0005
	Warm-up*	8 epochs
	Epochs	30
SEAL [12]	Base model	ResNet-50
	Minibatch size	256
	Optimizer*	SGD with momentum
	Initial LR*	0.01
	Momentum*	0.5
	Epochs per iteration*	50
	Iterations*	3
DivideMix [47]	Base model*	PreAct ResNet-18
	Minibatch size	128
	Optimizer*	SGD with momentum
	Initial LR*	0.02
	Momentum*	0.9
	Weight decay*	0.0005
	Epochs	30
FixMatch [82]	Base model*	Wide ResNet-28
	Minibatch size*	64
	Optimizer*	SGD with Nesterov momentum
	Initial LR*	0.03
	Momentum*	0.9
	Weight decay*	0.0005
	EMA decay*	0.999
	Unlabeled minibatch size ratio*	7
	Unlabeled loss coefficient*	1.0
	Pseudo label temperature*	0.95
	Training steps	9,000

to improved performance. Therefore, we truncated the training process to conserve computational resources and speed up experiments.

Self-evolution average label (SEAL) For the SEAL algorithm [12], we used the open-source implementation provided by the authors, available at `https://github.com/chenpf1025/IDN`. Following the authors' suggestions, the optimizer was SGD with momentum, with an initial learning rate of 0.01 and momentum of 0.5. Our implementation differs from that of the original authors in that the minibatch size was 256, and the base model was a randomly-initialized ResNet-50 [30]. Each iteration was trained for 50 epochs, and the network was trained for 3 iterations. These changes are due to the difference in dataset, and are based on the authors' recommendations for other datasets; our preliminary experiments showed that these hyperparameter settings yielded strong performance in the scenario with no adversarial label noise, so we considered them to be fair for the purposes of our experiments.

DivideMix For the DivideMix algorithm [47], we used the open-source implementation provided by the authors, available at `https://github.com/LiJunnan1992/DivideMix`. Following the authors' suggestions, the base networks were both a randomly-initialized PreAct ResNet-18 [31], and

the optimizers were SGD with momentum, with an initial learning rate of 0.02, momentum of 0.9, and weight decay of 0.0005. Our implementation differs from that of the original authors by training for 30 epochs, with a minibatch size of 128. These changes are due to the difference in dataset, and our preliminary experiments showed that continuing to train past 30 epochs did not lead to improved performance. Therefore, we truncated the training process to conserve computational resources and speed up experiments.

Our framework For our framework, we require the selection of three component algorithms to use for each of the three stages:

- We used FixMatch [82] as our semi-supervised learning algorithm for estimating labeler models. We used the open-source implementation available at `https://github.com/kekmodel/FixMatch-pytorch`.

- We used OpinionRank [18] as our learning from crowds algorithm for integrating redundant labels. Our implementation is available in our supplemental code.

- We used DivideMix [47] as our algorithm for learning from noisy labels. We used the open-source implementation provided by the authors, available at `https://github.com/LiJunnan1992/DivideMix`.

The base network for FixMatch was a randomly-initialized Wide Resnet-28 [105]. Following the original authors' suggestions, the minibatch size was 64, the optimizer was SGD with Nesterov momentum [86] with an initial learning rate of 0.03 and momentum of 0.9, the weight decay was 0.0005, the EMA decay rate was 0.999, the unlabeled minibatch size ratio was 7, the unlabeled loss coefficient was 1.0, and the pseudo label temperature was 0.95. Our implementation differs from that of the original authors by training for 9,000 steps. This change is due to the difference in dataset, and our preliminary experiments showed that continuing to train past 9,000 steps did not lead to improved performance. Therefore, we truncated the training process to conserve computational resources and speed up experiments.

OpinionRank is a nonparametric, deterministic algorithm, and so did not require any hyperparameter selection.

Our usage of DivideMix as the third stage of our framework was implemented identically as described above.

C.4 Hardware

Each experiment was performed on a single Nvidia Titan RTX on an internal cluster. Experiments were performed in parallel, as our cluster supports multiple such GPUs. Our dual-socket cluster is served by two Intel Xeon Gold 5218 2.30 GHz 16-core CPUs.

C.5 Licenses

The original authors' DivideMix code is licensed under the terms of the MIT license, available at `https://github.com/LiJunnan1992/DivideMix`. Jesper Wohlert's implementation of auxiliary deep generative models is licensed under the terms of the MIT license, available at `https://github.com/wohlert/semi-supervised-pytorch`. Jungdae Kim's implementation of FixMatch is licensed under the terms of the MIT license, available at `https://github.com/kekmodel/FixMatch-pytorch`.

We pledge to release our non-anonymized code (available in anonymized form in these supplemental materials) under the MIT license upon acceptance of this work.