Delving into Noisy Label Detection with Clean Data

Chenglin Yu [1] Xinsong Ma [1] Weiwei Liu [1]

Abstract

A critical element of learning with noisy labels is noisy label detection. Notably, numerous previous works assume that no source of labels can be clean in a noisy label detection context. In this work, we relax this assumption and assume that a small subset of the training data is clean, which enables substantial noisy label detection performance gains. Specifically, we propose a novel framework that leverages clean data by framing the problem of noisy label detection with clean data as a multiple hypothesis testing problem. Moreover, we propose BHN, a simple yet effective approach for noisy label detection that integrates the Benjamini-Hochberg (BH) procedure into deep neural networks. BHN achieves *state-of-the-art* performance and outperforms baselines by **28.48**% in terms of false discovery rate (FDR) and by **18.99**% in terms of F1 on CIFAR-10. Extensive ablation studies further demonstrate the superiority of BHN. Our code is available at https://github.com/ChenglinYu/BHN.

1. Introduction

Although deep neural networks (DNNs) have already achieved tremendous success in a variety of applications, ranging from computer vision (Krizhevsky et al., 2012; Liu et al., 2019; 2017) to natural language processing (Ragesh et al., 2021) and medical image analysis (Qian et al., 2022), such success requires large-scale datasets with correct annotations to be available for training (Liu & Tsang, 2015; 2017). Creating such large-scale datasets is arduous, and label annotation is often impacted by human error (Cheng et al., 2021). For instance, the recent utilization of crowdsourcing (Welinder et al., 2010) or search engines (Xiao et al., 2015) in dataset construction potentially results in the problem of *noisy labels* (Yi & Wu, 2019; Liu et al., 2020; Wang et al., 2021). Training a model with noisy labels could be detrimental, as networks have been shown to fit the mislabeled training examples (a.k.a. *memorization*) (Arpit et al., 2017; Zhang et al., 2017).

To mitigate the issue of *memorization* from noisy labels, extensive research has recently been conducted into noisy label learning algorithms. Among them, the *noise-cleansing* approach (Malach & Shalev-Shwartz, 2017; Tanaka et al., 2018; Chen et al., 2019; Zheng et al., 2020; Wang et al., 2021; Kim et al., 2021) is one of the most popular methods due to its promising experimental results. Noisy label detection focuses on segregating the clean data from the corrupted dataset, based on the outputs of a classifier trained on the noisy dataset. For example, Chen et al. (2019) randomly divide noisy training data, then employ cross-validation to classify true-labeled examples while removing large-loss examples in each training round.

Despite this promise, existing noisy label detection methods largely overlook clean data. We find that there is usually a small set of clean data in real-world noisy datasets. For example, Clothing1M (Xiao et al., 2015) comprises 1M images with real noisy labels and an additional ∼48K points of verified clean data. Moreover, the clean data have been used to boost the performance of the classifiers (Li et al., 2017; Veit et al., 2017; Hendrycks et al., 2018). Motivated by the above, this paper aims to explore the benefits of clean data for noisy label detection.

Accordingly, in this paper, we propose BHN, a simple yet effective approach that enables the effective exploitation of clean data for noisy label detection. We first frame the problem of noisy label detection with clean data as a multiple hypothesis testing problem. Subsequently, we define the p-values based on the neural network with the clean data. The p-values are then applied to the Benjamini-Hochberg (BH) procedure (Yoav & Daniel, 2001), which is a classical multiple hypothesis testing algorithm, to detect corrupted examples. Our BHN establishes the *state-of-the-art* performance on the real-world Clothing1M benchmark (Xiao et al., 2015) and the CIFAR-10 and CIFAR-100 datasets (Krizhevsky et al., 2009) with synthetic noise across different network depths. In particular, our method outperforms the previ-

[1]School of Computer Science, National Engineering Research Center for Multimedia Software, Institute of Artificial Intelligence and Hubei Key Laboratory of Multimedia and Network Communication Engineering, Wuhan University, Wuhan, China. Correspondence to: Weiwei Liu <liuweiwei863@gmail.com>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

ous best method by **28.48%** in terms of FDR and **18.99%** in terms of F1 on CIFAR-10, and by **2.26%** in the test accuracy of the final classifier on Clothing1M. Furthermore, extensive ablation studies are performed to demonstrate the superiority of our method.

Notation. For an integer N, we denote by $[N]$ the set of integers $\{1, \ldots, N\}$. For a vector a, we denote by $||a||_2$ the L_2 norm of a, and by a_y the y-th element of a. We use $\boldsymbol{e}_y$ to denote a standard basis vector with the y-th element 1. For a set A, $|A|$ represents the cardinality of this set. Random variables are always denoted with uppercase letters, while the realized values of the variable are denoted by the corresponding lowercase letters.

2. Preliminaries: Noisy Label Detection

We consider a classification problem from an instance space $\mathcal{X}$ to a label space $\mathcal{Y} = \{1, \ldots, K\}$. There is an (unknown) joint probability distribution $\mathcal{D}$ on $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ from which labeled examples $Z = (X, Y)$ are drawn. In the standard (non-noisy) supervised learning setting, the learner is given a training sample $D = \{(x_i, y_i)\}_{i \in [N]}$, sampled independently and identically distributed (*i.i.d.*) from $\mathcal{D}$; here the goal is to learn a classifier $f : \mathcal{X} \to \mathbb{R}^K$. An ideal classifier can be obtained by minimizing the following expected risk:

$$\mathcal{R}_{\mathcal{L}}(f) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\mathcal{L}\left(f(x), y\right)\right],$$

where $\mathcal{L}$ is the commonly used cross-entropy loss with the softmax activation function:

$$\mathcal{L}_{\mathrm{CE}}(f(x), y) = -\log \mathbb{P}(y|x) = -\log \frac{e^{f_y(x)}}{\sum_{i=1}^{K} e^{f_i(x)}}. \tag{1}$$

Here, $\mathbb{P}(y|x)$ is the softmax probability. We use $f'(x;\theta) = [\mathbb{P}(y=1|x), \mathbb{P}(y=2|x), \ldots, \mathbb{P}(y=K|x)]^T$ to denote the softmax logits of the neural network.

When learning from noisy labels, however, the learner encounters noisy examples $\tilde{Z} = (X, \tilde{Y})$, where $\tilde{Y}$ denotes a noisy version of Y. In more detail, the learner receives a noisy training sample $\widetilde{D}_1 = \{(x_i, \tilde{y}_i)\}_{i \in [N]}$, sampled *i.i.d.* from the noisy distribution $\widetilde{\mathcal{D}}$. The *noisy label* $\tilde{y}_i$ is referred to as *corrupted* if $\tilde{y}_i \neq y_i$, and *clean* otherwise. The corresponding examples are referred to as corrupted examples and clean examples respectively. Following Zhu et al. (2022), we focus on the case of closed-set label noise in which Y and $\widetilde{Y}$ are assumed to be in the same label space $[K]$. We are also given a set of clean data $D_0 = \{z_i^0\}_{i=1}^{2m}$ drawn from $\mathcal{D}$. The noisy label detection task can be formulated as a hypothesis testing problem: specifically, one of determining whether an example $(x, y) \in (\mathcal{X} \times \mathcal{Y})$ is from $\mathcal{D}$ or not (corrupted).

3. Noisy Label Detection via Multiple Hypothesis Testing

In this section, we formulate the noisy label detection task as a multiple (hypothesis) testing problem (Section 3.2). Subsequently, we demonstrate how this multiple testing problem can be integrated into modern neural networks (Section 3.3).

3.1. Multiple Hypothesis Testing

In this subsection, we provide some background on hypothesis testing that will be essential to an understanding of our approach. We begin with single hypothesis testing, and then further expand on multiple hypothesis testing.

Single Hypothesis Testing. In a single hypothesis testing problem, one considers two hypotheses (the *null* H_0 and the *alternative* H_1), and decides whether the data at hand sufficiently support the null. Specifically, given a set of observations (e.g., coin tosses), our goal is to increase the probability of making a true discovery (e.g., declaring that a coin is biased given that it is truly biased), while maintaining a prescribed level of type I error, α (e.g., declaring that a coin is biased given that it is not). This is typically accomplished by examining the p-value, which serves as a universal language for hypothesis testing (Bradley, 2012).

Definition 3.1 (p-value (George & Roger L, 2021))**.** A p-value $p(\widetilde{X})$ is a test statistic satisfying $0 \le p(\widetilde{X}) \le 1$ for every sample[1] $\widetilde{X}$. Small values of $p(\widetilde{X})$ give evidence that H_1 is true. A p-value is *valid* if, for every $0 \le \alpha \le 1$,

$$\mathbb{P}_{H_0}[p(\widetilde{X}) \le \alpha] \le \alpha \tag{2}$$

An important benefit of using p-values is that they enable a meaningful error rate to be maintained. There are many methods employed to maintain such error rates, which are based on p-values (Sture, 1979; Hochberg, 1988; Yoav & Yosef, 1995).

Multiple Hypothesis Testing. Multiple testing is performed when a statistical analysis involves multiple simultaneous tests, each of which has the potential to produce a "discovery". A stated confidence level generally applies only to each test considered individually; however, it is often desirable to determine a confidence level for a whole family of simultaneous tests. Notably, failure to compensate for multiple comparisons may have severe consequences. For example, consider a set of m coins, where our goal is to detect biased coins. Assume we test the hypothesis that a coin is biased with a confidence level of α. Furthermore, assume that all the coins are in reality unbiased. We can accordingly expect to make αm false discoveries. The probability that we make at least one false discovery (referred

[1]A sample means a sequence of examples.

to as the *Family-Wise Error Rate* (FWER)) increases as m grows. As a result, we are very likely to report that there is at least one biased coin in the set, even if the coins are all fair. This issue can be addressed by applying the Bonferroni correction (Jelle J & Aldo, 2014): it can be shown that by performing each test at a confidence level of α/m, we are guaranteed to achieve a FWER no greater than α. However, the Bonferroni correction is also very conservative, with the result that few discoveries are likely to be made.

To overcome these issues, Yoav and Yosef (1995) propose to control the *False Discovery Rate* (FDR), and introduce a simple approach for controlling the FDR, commonly known as the Benjamini-Hochberg (BH) procedure.

Definition 3.2 (FDR)**.** Let V denote the number of true null hypotheses rejected; moreover, let R be the number of rejected hypotheses, and Q be the unobservable random quotient,

$$Q = \begin{cases} V/R, & \text{if } R > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

where Q is the proportion of rejections that are false rejections (unless there are no rejections at all, in which case Q is defined to be zero). The expectation of Q is called the FDR: FDR = $\mathbb{E}[Q]$.

Definition 3.3 (The Benjamini-Hochberg Procedure (Yoav & Daniel, 2001))**.** Let $p_{(1)} \leq p_{(2)} \leq \cdots \leq p_{(N)}$ be the ordered observed p-values. Define $k = \max\{i : p_{(i)} \leq \frac{i}{N}\alpha\}$. The BH procedure rejects $H_{0,(1)}, \ldots, H_{0,(k)}$, where $H_{0,(i)}$ is the hypothesis corresponding to $p_{(i)}$. If no such i exists, the BH procedure rejects no hypothesis.

As is evident from above, FDR is evidently less conservative than the FWER. Thus, FDR-controlling procedures can result in more discoveries than FWER-controlling procedures, at the cost of an increased number of type I errors.

3.2. Noisy Label Detection as a Multiple Hypothesis Testing Problem

The noisy distribution ($\widetilde{\mathcal{D}}$) can be regarded as a mixture of clean distribution ($\mathcal{D}$) and corrupted distribution ($\mathcal{D}_{\text{cor}}$). We formalize the problem of determining which examples in $\widetilde{D}_1$ are corrupted as the following N testing problems:

$$\begin{aligned} H_{0,1} &: (X_1, \tilde{Y}_1) \sim \mathcal{D}, \quad H_{1,1} : (X_1, \tilde{Y}_1) \sim \mathcal{D}_{\text{cor}} \\ H_{0,2} &: (X_2, \tilde{Y}_2) \sim \mathcal{D}, \quad H_{1,2} : (X_2, \tilde{Y}_2) \sim \mathcal{D}_{\text{cor}} \\ &\vdots \\ H_{0,N} &: (X_N, \tilde{Y}_N) \sim \mathcal{D}, \quad H_{1,N} : (X_N, \tilde{Y}_N) \sim \mathcal{D}_{\text{cor}}. \end{aligned} \tag{4}$$

By rejecting $H_{0,i}$, we conclude that the true label of X_i is not Y_i.

3.3. P-Value with Neural Networks

In this section, we define the p-value with neural networks.

Motivation. According to the definition of p-value given in Equation (2), we can decisively reject $H_{0,i}$ when the p-value is small. On the other hand, the score function has been used to detect corrupted examples (Northcutt et al., 2021; Cheng et al., 2021) based on neural networks. A score function $\hat{s}$ assigns a scalar value to any noisy example, such that small values of $\hat{s}$ indicate that the example may be corrupted. We therefore define the p-value based on the score function to incorporate the neural network. There are two critical components: the score function and the computation of the p-value based on the score.

Noisy Label Detection Score Function. Various score functions have been designed to detect corrupted examples, hard-to-learn instances (Liu et al., 2021), and out-of-distribution examples (Hendrycks & Gimpel, 2017; Wei et al., 2022; Mu & Yixuan, 2023). The majority of these score functions are based on the softmax probability of the neural network. Therefore, for the sake of simplicity, we directly adopt the negative cross-entropy loss of the neural network on the noisy example as our score function $\hat{s} : \mathcal{Z} \rightarrow \mathbb{R}$ for the sake of simplicity (i.e., the negative logarithm of the softmax probability corresponding to the noisy label $\hat{s}((x, \tilde{y})) = -\mathcal{L}_{\text{CE}}(f(x), \tilde{y})$). The key insight is that the true label should trigger a relatively higher softmax probability than that of a corrupted label for a well-trained neural network (Hendrycks & Gimpel, 2017).

Having established the score function, we next show how to transform the score into a valid p-value.

P-Value based on the Score Function. Suppose that $\hat{s}(\tilde{Z})$ follows a continuous distribution if $\tilde{Z} \sim \mathcal{D}$ is independent of the data used to train $\hat{s}$. We define F as the cumulative distribution function (CDF) of $\hat{s}(\tilde{Z})$. If F was known, then we could use $F(\hat{s}(\tilde{Z}_i))$ as an exact p-value for the null hypothesis $H_{0,i} : Z_i \sim \mathcal{D}$, in the sense that $F(\hat{s}(\tilde{Z}_i))$ would be uniformly distributed if $H_{0,i}$ is true. In practice, however, we do not know F; instead we have access to a small set of clean data D_0. Based on this clean data, we determine form the empirical CDF of $\hat{s}(\tilde{Z})$. We split D_0 into two sets. The first set, $D_0^{\text{train}} = \{z_1^0, \ldots, z_m^0\}$, is used to train the score function; the second set, $D_0^{\text{cal}} = \{z_{m+1}^0, \ldots, z_{2m}^0\}$, is used as a calibration set for estimating the CDF of $\hat{s}(\tilde{Z})$. The empirical CDF is given by

$$\hat{F}_m(t) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{I}_{(\infty, t]}[\hat{s}(Z_{m+i}^0)], \tag{5}$$

where $\mathbb{I}_{(\infty,t]}[s]$ is a $\{0, 1\}$-valued indicator function for the event $\{s \leq t\}$. Unfortunately, the composition of $\hat{F}_m$ onto $\hat{s}$ (i.e., $(\hat{F}_m \circ \hat{s})(\tilde{Z}) = \frac{|\{1 \leq i \leq m : \hat{s}(Z_{m+i}^0) \leq \hat{s}(\tilde{Z})\}|}{m}$) is not a valid

p-value, which is uniformly distributed on $\{0, \frac{1}{m}, \frac{2}{m}, \ldots, 1\}$ if $\tilde{Z} \sim \mathcal{D}$ is independent of the data in D_0^{train}. For example, when $m = 6, \alpha = 0.25$, $\mathbb{P}_{H_0}[p(Z) \leq 0.25] = \frac{2}{7} \approx 0.28 > 0.25$, which contradicts the validity of the p-value according to Equation (2). We therefore consider a modified composition in which we add 1 to both the numerator and denominator of the original composition, as follows:

$$p(\tilde{Z}) = \frac{|\{1 \leq i \leq m : \hat{s}(Z_{m+i}^0) \leq \hat{s}(\tilde{Z})\}| + 1}{m+1}. \tag{6}$$

Since $p(\tilde{Z})$ as defined in Equation (6) is uniformly distributed on $\{\frac{1}{m+1}, \frac{2}{m+1}, \ldots, 1\}$ when $\hat{s}(\tilde{Z})$ follows a continuous distribution and $\tilde{Z} \sim \mathcal{D}$ is independent of the data in D_0^{train}, the adjusted composition is a valid p-value. We refer to the p-value defined in Equation (6) as the *empirical p-value*.

BH Procedure. In Section 3.1, we have shown that the BH procedure can control the FDR. The following theorem establishes that the BH procedure maintains the FDR at a specified significance level when using the empirical p-values defined above.

Theorem 3.4 (BH procedure with empirical p-values can control the FDR)**.** *Assume that $\hat{s}(\tilde{Z})$ is continuously distributed. Consider N noisy examples $\tilde{z}_1, \ldots, \tilde{z}_N$, where the clean examples are jointly independent of each other and of the data in D_0. Then, the BH procedure applied at level $\alpha \in (0, 1)$ to $(p(\tilde{Z}_1), \ldots, p(\tilde{Z}_N))$ controls the FDR at level $\pi_0\alpha$, where π_0 is the proportion of true nulls; that is,*

$$\mathbb{E}\left[\frac{|\mathcal{R} \cap \mathcal{H}_0|}{\max\{1, |\mathcal{R}|\}}\right] \leq \pi_0\alpha \leq \alpha, \tag{7}$$

where $\mathcal{H}_0 = \{i : H_{0,i}$ holds$\} \subseteq \{1, \ldots, N\}$ is the subset of clean examples in the noisy dataset, and $\mathcal{R} \subseteq \{1, \ldots, N\}$ is the subset of examples reported as corrupted examples.

Proof of Theorem 3.4. It is known that the BH procedure can control the FDR when a particular type of mutual p-value dependence, called Positive Regression Dependent on a Subset (PRDS) (Vladimir, 2013), is employed. Therefore, our main task is to prove that the empirical p-values are PRDS. Since the property PRDS is based on the concept of the *increasing set*, we first provide the definition of an increasing set.

Definition 3.5 (Increasing set)**.** For vectors a and b of equal dimension, we say that $a \succeq b$ if every coordinate of a is no smaller than the corresponding coordinate of b. Moreover, $A \subset \mathbb{R}^N$ is *increasing* if $a \in A$ and $b \succeq a$ implies $b \in A$.

The definition of PRDS is as presented below:

Definition 3.6 (PRDS)**.** A random vector $V = (V_1, \ldots, V_N)$ is PRDS on a set $I_0 \subseteq \{1, \ldots, N\}$ if, for any $i \in I_0$ and any increasing set A, the probability $\mathbb{P}[V \in A | V_i = v]$ is increasing in v.

In the multiple testing literature, $(p(\tilde{Z}_1), p(\tilde{Z}_2), \ldots, p(\tilde{Z}_N))$ is often said to be PRDS if it is PRDS on the set of nulls.

Theorem 3.7 (Empirical p-values based on the score function are PRDS)**.** *Assume that $\hat{s}(\tilde{Z})$ is continuously distributed. Consider N noisy examples $\tilde{z}_1, \ldots, \tilde{z}_N$, where the clean examples are jointly independent of each other and of the data in D_0. The p-values $(p(\tilde{Z}_1), \ldots, p(\tilde{Z}_N))$ are then PRDS on the set of clean examples.*

The proof of Theorem 3.7 is provided in Appendix G. Next, we present the main corollary based on Theorem 3.7, which completes the proof of Theorem 3.4.

Corollary 3.8 (Benjamini and Yekutieli (Yoav & Daniel, 2001))**.** *In the setting of Theorem 3.7, the BH procedure applied at level $\alpha \in (0, 1)$ to $(p(\tilde{Z}_1), \ldots, p(\tilde{Z}_N))$ controls the FDR at level $\pi_0\alpha$, where π_0 is the proportion of true nulls. That is,*

$$\mathbb{E}\left[\frac{|\mathcal{R} \cap \mathcal{H}_0|}{\max\{1, |\mathcal{R}|\}}\right] \leq \pi_0\alpha \leq \alpha, \tag{8}$$

where $\mathcal{H}_0 = \{i : H_{0,i}$ holds$\} \subseteq \{1, \ldots, N\}$ is the subset of clean examples in the noisy dataset, and $\mathcal{R} \subseteq \{1, \ldots, N\}$ is the subset of examples reported as likely corrupted examples.

The proof is completed. $\square$

We summarize our approach in Algorithm 1. Our algorithm offers two compelling advantages:

1. **Distributional assumption free**: Our method does not impose distributional assumptions about the underlying *noise transition matrix*, which provides a probabilistic formulation on label transition from true label y to noisy label $\tilde{y}$. Therefore, our method provides stronger flexibility and generality, and is applicable no matter what the noise type is.
2. **Statistical guarantee**: Our method returns a valid p-value for each noisy example. This operation guarantees maintaining a FDR on the noisy training set, thereby avoiding an unnecessary waste of clean examples.

4. Experiments

In this section, we first introduce implementation details and baselines for experiments in Section 4.1. Then, we provide quantitative performance results in Sections 4.2 and 4.3.

Table 1. Comparisons of FDR, Recall, and F1 (%) on CIFAR-10 and CIFAR-100. We add three types of synthetic noise to each dataset. We use ResNet34 as the backbone for each method. ↑ indicates that larger values are better, and vice versa. Top 2 results are marked in **bold**.

Method	**CIFAR-10**									**CIFAR-100**								
	Symm. 0.6			**Asym. 0.3**			**Inst. 0.4**			**Symm. 0.6**			**Asym. 0.3**			**Inst. 0.4**		
	FDR↓	Recall↑	F1↑	FDR↓	Recall↑	F1↑	FDR↓	Recall↑	F1↑	FDR↓	Recall↑	F1↑	FDR↓	Recall↑	F1↑	FDR↓	Recall↑	F1↑
CORES	**11.92**	86.05	87.05	59.06	5.75	10.09	**19.59**	32.45	46.24	**6.19**	48.02	63.52	81.82	0.04	0.09	**14.40**	2.47	4.81
CL	40.00	84.43	70.15	70.23	87.43	44.41	60.33	85.42	54.18	39.50	**89.10**	72.06	70.16	**93.52**	45.25	60.55	**91.72**	55.16
SimiFeat-V	12.23	90.64	**89.18**	**35.65**	84.31	**72.99**	30.20	74.77	72.20	11.77	54.41	67.31	**49.29**	71.99	**59.50**	35.89	69.59	**66.74**
SimiFeat-R	31.59	**99.34**	81.02	44.00	**91.70**	69.54	34.57	**91.29**	**76.23**	29.25	**99.33**	**82.64**	60.09	**91.45**	55.57	48.42	**93.53**	66.50
BHN	**3.80**	**95.90**	**96.05**	**7.17**	**91.13**	**91.98**	**5.98**	**91.18**	**92.58**	**4.10**	81.31	**88.01**	**7.79**	67.14	**77.71**	**6.48**	72.99	**81.99**

Algorithm 1 Noisy Label Detection via Multiple Testing

```
1: Input: Clean data $D_0$. Noisy data: $\widetilde{D}_1 = \{(x_i, \tilde{y}_i)\}_{i\in[N]}$.
2: Split $D_0$ into two sets of even size $D_0^{\text{train}}$ and $D_0^{\text{cal}}$.
3: Train the neural network $f$ using $D_0^{\text{train}}$
4: Calculate the $p$-values of $\widetilde{D}_1$.
5: Using BH procedure to determine whether to reject or accept for each null hypothesis. The examples corresponding to the rejected hypotheses are regarded as corrupted examples.
```

4.1. Experimental Setup

Datasets and Baselines. We evaluate BHN on three benchmark datasets: CIFAR-10, CIFAR100 (Krizhevsky et al., 2009), and Clothing1M (Xiao et al., 2015). CIFAR10 and CIFAR-100 are the most popular datasets used in the literature of learning with noisy labels (Reed et al., 2014; Patrini et al., 2017; Jiang et al., 2018). As these two datasets are clean and lack noisy labels, we divide the training set into two different sets, D_0 and D_1, and follow the settings commonly used in the literature (Guo et al., 2018; Jiang et al., 2018) to add synthetic noise into D_1. D_0 is used as the clean set. No noisy labels are added in the test sets. Three different types of synthetic label noise are generated, as outlined below.

- **Symmetric Noise** flips labels uniformly to all other classes (Cheng et al., 2021).
- **Asymmetric Noise** is generated by pair-wise flipping, i.e., randomly flipping the true label i to the next class $(i \mod K) + 1$ (Zhu et al., 2022).
- **Instance-dependent Noise** flips labels according to the probability that an example is mislabeled. This probability is computed based on the corresponding feature of the example (Zhu et al., 2021).

We denote the ratio of corrupted examples in the noisy dataset by η. In line with (Zhu et al., 2022), we use symmetric noise with $\eta = 0.6$ (Symm. 0.6), asymmetric noise with $\eta = 0.3$ (Asym. 0.3), and instance-dependent noise with $\eta = 0.4$ (Inst. 0.4) in our experiments. More details on each type of noise are provided in Appendix A.

Moreover, we conduct experiments on a large real-world dataset, Clothing1M (Xiao et al., 2015), which is composed of clothing data crawled from online shopping websites.

We compare our proposed BHN with four state-of-the-art baselines: **CORES** (Cheng et al., 2021), **Confident Learning (CL)** (Northcutt et al., 2021), and **SimiFeat-V and SimiFeat-R** (Zhu et al., 2022). To facilitate a fair comparison, we implement the same backbone model for all methods. Following the convention established by Cheng et al. (2021), we use ResNet34 (He et al., 2016) for CIFAR-10 and CIFAR-100 and ResNet50 (He et al., 2016) for Clothing1M. More details are provided in Appendix C.

Evaluation Metrics. We compare BHN with the baselines on two aspects.

- **Noisy Label Detection:** (1) the false discovery rate (FDR) of detected corrupted examples; (2) the recall of detected corrupted examples (Recall); (3) the F1 of the detected corrupted examples, which is the harmonic mean of precision and recall.
- **Image Classification**: We compare the accuracy of the final image classifier. BHN and all other baselines are evaluated on the same clean test set.

4.2. Noisy Label Detection Performance on Noisy Datasets

We present the results of comparing the noisy label detection performance of BHN and the other baselines on two synthetic datasets with the aforementioned three types of noise in Table 1. As the table shows, BHN significantly improves the FDR and F1 of noisy label detection in all cases. For example, we observe that BHN reduces the FDR from 35.65% to 7.17% under the Asym. 0.3 noise on CIFAR-10 compared with the best baseline, a direct improvement of **28.48**%; moreover, BHN outperforms the best baseline by **18.99**% in terms of F1. On the CIFAR-100 dataset, BHN improves the F1 from 66.74% to 81.99% under the Inst. 0.4 noise compared with the best baseline, a direct improvement of **15.25**%. Furthermore, when counting the frequency of reaching top-2 recall, we find that BHN outperforms all other baselines.

BHN Guarantees Maintenance of the FDR. Table 1

Table 2. Comparisons of test accuracy of the final classifier (%) on CIFAR-10 and CIFAR-100. All methods use ResNet34 as the backbone. Top 1 results are marked in **bold**.

	CIFAR-10			CIFAR-100		
Method	**Symm. 0.6**	**Asym. 0.3**	**Inst. 0.4**	**Symm. 0.6**	**Asym. 0.3**	**Inst. 0.4**
CORES	73.47	65.28	64.72	30.43	49.96	43.10
CL	25.00	48.41	35.80	7.12	10.34	9.13
SimiFeat-V	77.08	83.74	74.60	37.40	54.88	50.72
SimiFeat-R	63.89	84.13	79.48	33.92	46.50	42.68
BHN (ours)	**86.30**	**89.79**	**88.63**	**46.16**	**61.46**	**58.06**

shows that all the FDRs of BHN are smaller than the prescribed level of FDR α (i.e., 10%). which demonstrates that BHN guarantees maintenance of the FDR.

BHN is Free from Distributional Assumptions. As can be seen from Table 1, BHN consistently performs well under various types of noisy labels without any access to noise information (i.e., noise ratio and noise type). For example, our BHN achieves F1 scores of 96.05%, 91.98%, and 92.58% under the conditions of Symm. 0.6 noise, Asym. 0.3 noise, and Inst. 0.4 noise respectively on CIFAR-10. By contrast, CORES achieves 87.05% F1 under Symm. 0.6 noise, but 10.09% and 46.24% F1 under Asym. 0.3 noise and Inst. 0.4 noise respectively on CIFAR-10. This observation suggests that BHN does not depend on the distributional assumptions on the noisy distribution, and further informs us that customized training processes might not be universally applicable.

In addition to ResNet34, we also compare our method with other baselines using ResNet18 and ResNet50 on CIFAR-10 and CIFAR-100. The results, provided in Table 5 of Appendix D, show that BHN consistently outperforms the baselines under all networks in terms of F1. The superior performances at various network depths demonstrate the effectiveness and general applicability of BHN.

4.3. Classification Performance on Clean Test Sets

We further verify whether BHN improves the classification performance of the final classifier. For all methods, we remove the detected corrupted examples, and use the remaining examples to train the classifier. Our results in Table 2 show that BHN consistently outperforms all other baselines in all cases. For example, we can observe that BHN surpasses the best baseline (SimiFeat-V) by **12.24%** in terms of test accuracy on CIFAR-100 with Symm. 0.6 noise. In addition to ResNet50, we also compare BHN with other methods using ResNet18 and ResNet34 on Clothing1M. The results are provided in Table 6 of Appendix E. Again, BHN can be seen to outperform all the other approaches.

Results on Clothing1M. While the previous results originate from synthetic datasets, Table 3 presents the experimental results on Clothing1M, which is a dataset with real-world noisy labels. Our BHN achieves the *state-of-the-art* performance, outperforming the runner-up method by **2.26%** in terms of test accuracy of the final classifier.

Table 3. The best epoch (clean) test accuracy for each method on Clothing1M. None: Standard training with 1M noisy data. Top 1 results are marked in **bold**.

Method	None	CORES	SimiFeat-R	BHN
Accuracy	70.32	73.24	72.37	**75.50**

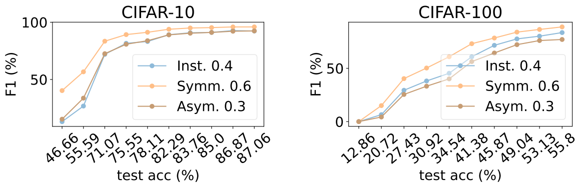

Figure 1. Ablation on the test accuracy of the neural network on CIFAR-10 (left) and CIFAR-100 (right).

5. A Closer Look at BHN

In this section, we provide further analysis and ablations to facilitate a deeper understanding of the behavior of BHN. Unless explicitly specified, the ablations are based on the ResNet18 model.

5.1. Effect of Test Accuracy of the Neural Network Adopted in the Score Function

Figure 1 illustrates the effect of the test accuracy of the neural network adopted in the score function. We observe that the F1 score increases with the test accuracy of the neural network. Moreover, BHN achieves relatively high F1 based on neural networks with relatively low test accuracy. For example, we obtain an F1 of approximately 83% for Symm. 0.6 noise on CIFAR-100, while the test accuracy of the neural network is about 49%.

Analysis. Figure 2 presents a histogram of the scores of clean and corrupted examples generated by neural networks with different test accuracies on CIFAR-100. As we can observe from Figure 2, there is substantial overlap between the score densities of clean and corrupted examples when the test accuracy of the neural network is 12.86%. As the test accuracy of the neural network increases, the densities of clean examples become more concentrated on the high score side, while the densities of corrupted examples do not change much. When the test accuracy of the neural network is 45.87%, the overlap between the score densities of clean and corrupted examples becomes small. We can therefore conclude that the neural network with relatively low test accuracy can produce highly distinguishable scores between clean and corrupted examples.

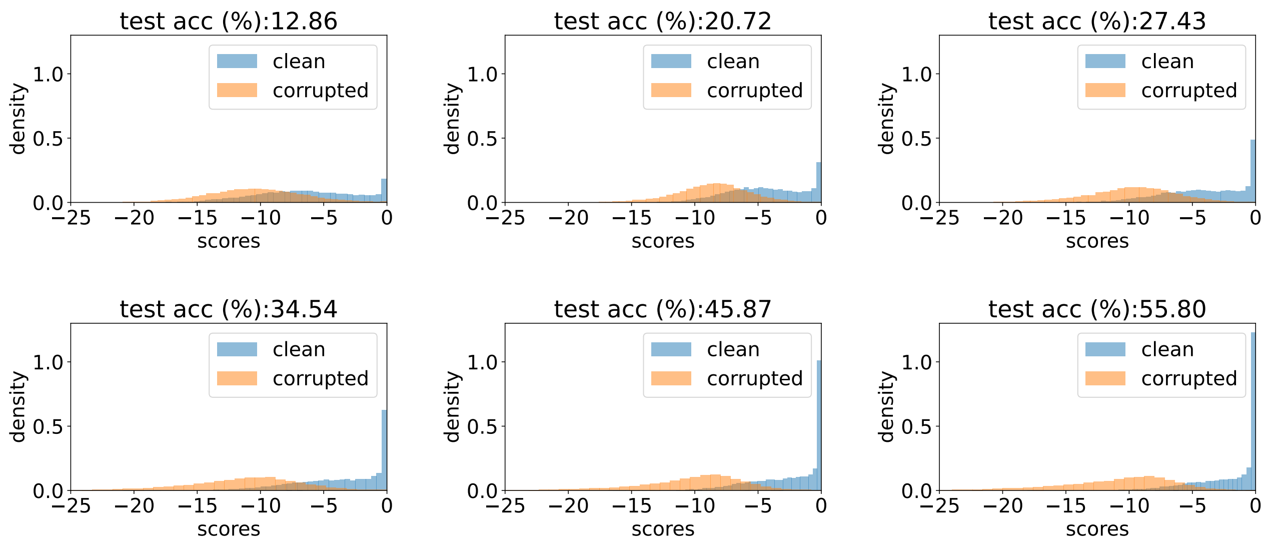

Figure 2. Scores of clean and corrupted examples on CIFAR-100 with Symm. 0.6 noise using the ResNet18 model with different test accuracies. The test accuracies of the neural network increase steadily from the upper left to the lower right.

5.2. Alternative Score Function

We further note that our framework could also be compatible with alternative forms of score function $\hat{s}$. For example, we explore another variant of our method, BHN-cosin, which uses the cosine similarity (Techapanurak & Okatani, 2019; Hsu et al., 2020; Zhu et al., 2022) between the softmax logits of the neural network and the one-hot vector of the label as the score function, i.e.,

$$\hat{s}((x, \tilde{y})) = \frac{f'(x;\theta)\boldsymbol{e}_{\tilde{y}}}{\|f'(x;\theta)\|_2\|\boldsymbol{e}_{\tilde{y}}\|_2} \tag{9}$$

where $\boldsymbol{e}_{\tilde{y}}$ is the one-hot encoding of label $\tilde{y}$.

Table 4 contrasts the performances of BHN and BHN-cosin on CIFAR-10 and CIFAR-100 using ResNet18. Notably, the same results are obtained on ResNet34 and ResNet50 (Tables 7 and 8 in Appendix F). Our results reveal that BHN-cosin achieves almost the same performance as BHN, which suggest that the norm of softmax logits has little effect on the noisy label detection performance. To explore this phenomenon further, we contrast the norms of softmax logits on clean and corrupted examples using neural networks with different test accuracies in Figure 4 of Appendix F. The results show that the norms on clean and corrupted examples have almost the same densities, and moreover that the norms of softmax logits cannot enhance the discrimination beween clean and corrupted examples.

5.3. Effect of the Noise Rate η

In Figure 3, we investigate the effect of different symmetric noise rates varying from 0.1 to 0.6. Note that a smaller symmetric noise rate indicates more clean data and less corrupted data. Here, we consistently use 60% of the overall training set as D_1. We highlight several observations: (1) The F1 for all methods generally decreases with decreasing noise ratio. In particular, a smaller noise ratio translates into a more difficult detection problem, because the overlap between the noisy and clean distributions becomes significant. For example, on CIFAR-10, the F1 of SimiFeat-V increases from 51.37% (η=0.1) to 88.34% (η=0.6). (2) Our methods, BHN and BHN-cosin, are overall more robust than the baselines at small values of η. In a challenging case with η=0.1, BHN and BHN-cosin outperform SimiFeat-V by **33.16**% and **32.95**% in terms of F1 on CIFAR-10, respectively. These experimental results further suggest that BHN does not rely on specific assumptions regarding the noisy distribution. (3) Our methods always maintain the FDR under different levels of noise.

6. Related Works

Existing approaches to learning with noisy labels can be classified into two types: (1) detecting corrupted labels and then cleansing potential corrupted labels, or reducing their impacts in subsequent training; (2) directly training noise-robust models with noisy labels.

(1) **Noise-cleansing-based Approaches** attempt to detect corrupted labels (a.k.a. sample selection) and then cleanse potential corrupted labels or reduce their impacts on subsequent training (Krueger et al., 2017; Malach & Shalev-Shwartz, 2017; Jiang et al., 2018; Chen et al., 2019; Song et al., 2019a; Lyu & Tsang, 2020; Zhang et al., 2021; Song

Table 4. Comparisons of FDR, Recall, F1, and test accuracy (%) of BHN and BHN-cosin on CIFAR-10 and CIFAR-100. All methods are based on the ResNet18 network.

Method	**Symm. 0.6**				**Asym. 0.3**				**Inst. 0.4**			
	FDR↓	Recall↑	F1↑	Accuracy ↑	FDR↓	Recall↑	F1↑	Accuracy ↑	FDR↓	Recall↑	F1↑	Accuracy ↑
						CIFAR-10						
BHN	4.00	96.40	96.20	86.65	7.40	92.40	92.50	90.14	6.30	91.40	92.50	88.09
BHN-cosin	3.71	95.50	95.89	86.17	6.82	91.35	92.25	90.61	5.77	90.29	92.22	87.84
						CIFAR-100						
BHN	3.68	80.82	87.89	49.56	6.90	65.39	76.82	62.46	6.15	72.73	81.95	59.46
BHN-cosin	4.23	79.35	86.79	50.24	7.30	63.22	75.17	63.01	6.38	71.55	81.11	58.93

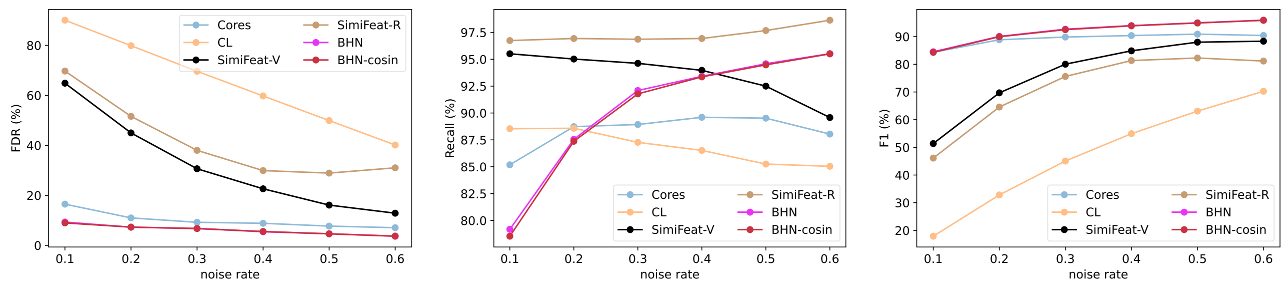

Figure 3. FDR, Recall, and F1 of different methods on CIFAR-10 using ResNet18 with different noise rates

et al., 2021; Zhu et al., 2022). The tendency of DNNs to fall into memorization has been explored theoretically and empirically to facilitate the identification of clean examples from among noisy training data (Krueger et al., 2017; Zhang et al., 2020a). Since DNNs tend to first learn simple and generalized patterns, then gradually overfit to noisy patterns (Arpit et al., 2017; Song et al., 2019b), one common approach to robust training method design involves treating small-loss training examples as clean (Jiang et al., 2018; Shen & Sanghavi, 2019; Chen et al., 2019; Li et al., 2020). Recent methods often leverage multiple DNNs to cooperate with one another, or run multiple training rounds (Wang et al., 2018). Notably, all these methods require training DNNs with noisy supervision and are impacted by the memorization of corrupted examples. Zhu et al. (2022) propose a training-free solution to detect noisy labels by using the neighborhood information of features. Almost all previous methods are largely dependent only on the noisy dataset. By contrast, our BHN enables us to leverage clean data to detect noisy labels.

(2) **Noise-robust Approaches** aim to design a robust classifier with noisy labels, including by utilizing explicit regularizations (Liu et al., 2020; Wei et al., 2021), designing robust architectures (Xiao et al., 2015; Chen & Gupta, 2015; Bekker & Goldberger, 2016; Jindal et al., 2016; Goldberger & Ben-Reuven, 2017; Yao et al., 2019; Cheng et al., 2020), or improving the loss functions (Zhang & Sabuncu, 2018; Wang et al., 2019; Ma et al., 2020) to design a robust classifier. Some of these methods assume the availability of a small set of clean data, then leverage this clean data to train a classifier robust to label noise (Li et al., 2017; Hendrycks et al., 2018; Ren et al., 2018; Zhang et al., 2020b). Li et al. (2017) propose distilling the predictions of a model pre-trained on clean labels into a second network trained on both these predictions and the noisy labels. Hendrycks et al. (2018) utilize clean data by proposing a loss correction technique that utilizes clean examples in a data-efficient manner to mitigate the effects of label noise on deep neural network classifiers. Ren et al. (2018) propose a loss correction approach that uses a set of clean data and meta-learning. Zhang et al. (2020b) leverage a small set of clean data to estimate the exemplar weights and labels, then train models in a supervised manner that is highly invulnerable to label noise. By contrast, the clean data in our work is leveraged for noisy label detection.

7. Conclusion

In this paper, we present BHN, a simple yet effective approach for noisy label detection that leverages a set of clean data. Throughout our experiments, we demonstrate that BHN outperforms previous noisy label detection methods across various datasets and networks, which we show by considering several types of noise and noise strengths. These results demonstrate that the BHN is a powerful data-efficient method for corrupted label detection.

Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grant 61976161, the Fundamental Research Funds for the Central Universities under Grant 2042022rc0016.

References

Arpit, D., Jastrzebski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A. C., Bengio, Y., and Lacoste-Julien, S. A closer look at memorization in deep networks. In *ICML*, 2017.

Bekker, A. J. and Goldberger, J. Training deep neural-networks based on unreliable labels. In *ICASSP*, 2016.

Bradley, E. *Large-scale inference: empirical Bayes methods for estimation, testing, and prediction*, volume 1. 2012.

Chen, P., Liao, B., Chen, G., and Zhang, S. Understanding and utilizing deep neural networks trained with noisy labels. In *ICML*, 2019.

Chen, X. and Gupta, A. Webly supervised learning of convolutional networks. In *ICCV*, 2015.

Cheng, H., Zhu, Z., Li, X., Gong, Y., Sun, X., and Liu, Y. Learning with instance-dependent label noise: A sample sieve approach. In *ICLR*, 2021.

Cheng, L., Zhou, X., Zhao, L., Li, D., Shang, H., Zheng, Y., Pan, P., and Xu, Y. Weakly supervised learning with side information for noisy labeled images. In *ECCV*, 2020.

George, C. and Roger L, B. *Statistical inference*. 2021.

Goldberger, J. and Ben-Reuven, E. Training deep neural-networks using a noise adaptation layer. In *ICLR*, 2017.

Guo, S., Huang, W., Zhang, H., Zhuang, C., Dong, D., Scott, M. R., and Huang, D. Curriculumnet: Weakly supervised learning from large-scale web images. In *ECCV*, 2018.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Hendrycks, D. and Gimpel, K. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *ICLR*, 2017.

Hendrycks, D., Mazeika, M., Wilson, D., and Gimpel, K. Using trusted data to train deep networks on labels corrupted by severe noise. In *NeurIPS*, 2018.

Hochberg, Y. A sharper bonferroni procedure for multiple tests of significance. *Biometrika*, 75(4):800–802, 1988.

Hsu, Y., Shen, Y., Jin, H., and Kira, Z. Generalized ODIN: detecting out-of-distribution image without learning from out-of-distribution data. In *CVPR*, 2020.

Jelle J, G. and Aldo, S. Multiple hypothesis testing in genomics. *Statistics in medicine*, 33(11), 2014.

Jiang, L., Zhou, Z., Leung, T., Li, L.-J., and Fei-Fei, L. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*, 2018.

Jindal, I., Nokleby, M. S., and Chen, X. Learning deep networks from noisy labels with dropout regularization. In *ICDM*, 2016.

Kim, T., Ko, J., Cho, S., Choi, J., and Yun, S. FINE samples for learning with noisy labels. In *NeurIPS*, 2021.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. Technical report, 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

Krueger, D., Ballas, N., Jastrzebski, S., Arpit, D., Kanwal, M. S., Maharaj, T., Bengio, E., Fischer, A., and Courville, A. C. Deep nets don't learn via memorization. In *ICLR Workshop*, 2017.

Li, J., Socher, R., and Hoi, S. C. Dividemix: Learning with noisy labels as semi-supervised learning. In *ICLR*, 2020.

Li, Y., Yang, J., Song, Y., Cao, L., Luo, J., and Li, L. Learning from noisy labels with distillation. In *ICCV*, 2017.

Liu, E. Z., Haghgoo, B., Chen, A. S., Raghunathan, A., Koh, P. W., Sagawa, S., Liang, P., and Finn, C. Just train twice: Improving group robustness without training group information. In *ICML*, 2021.

Liu, S., Niles-Weed, J., Razavian, N., and Fernandez-Granda, C. Early-learning regularization prevents memorization of noisy labels. In *NeurIPS*, 2020.

Liu, W. and Tsang, I. W. Large margin metric learning for multi-label prediction. In *AAAI*, 2015.

Liu, W. and Tsang, I. W. Making decision trees feasible in ultrahigh feature and label dimensions. *Journal of Machine Learning Research*, 18:81:1–81:36, 2017.

Liu, W., Tsang, I. W., and Müller, K. An easy-to-hard learning paradigm for multiple classes and multiple labels. *Journal of Machine Learning Research*, 18:94:1–94:38, 2017.

Liu, W., Xu, D., Tsang, I. W., and Zhang, W. Metric learning for multi-output tasks. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 41(2):408–422, 2019.

Lyu, Y. and Tsang, I. W. Curriculum loss: Robust learning and generalization against label corruption. In *ICLR*, 2020.

Ma, X., Huang, H., Wang, Y., Romano, S., Erfani, S. M., and Bailey, J. Normalized loss functions for deep learning with noisy labels. In *ICML*, 2020.

Malach, E. and Shalev-Shwartz, S. Decoupling ”when to update” from ”how to update”. In *NeurIPS*, 2017.

Mu, C. and Yixuan, L. Out-of-distribution detection via frequency-regularized generative models. In *WACV*, 2023.

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

Patrini, G., Rozza, A., Krishna Menon, A., Nock, R., and Qu, L. Making deep neural networks robust to label noise: A loss correction approach. In *CVPR*, 2017.

Qian, R., Lai, X., and Li, X. 3d object detection for autonomous driving: A survey. *Pattern Recognition*, 130: 108796, 2022.

Ragesh, R., Sellamanickam, S., Iyer, A., Bairi, R., and Lingam, V. Hetegcn: Heterogeneous graph convolutional networks for text classification. In *WSDM*, 2021.

Reed, S., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., and Rabinovich, A. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014.

Ren, M., Zeng, W., Yang, B., and Urtasun, R. Learning to reweight examples for robust deep learning. In *ICML*, 2018.

Shen, Y. and Sanghavi, S. Learning with bad training data via iterative trimmed loss minimization. In *ICML*, 2019.

Song, H., Kim, M., and Lee, J. SELFIE: refurbishing unclean samples for robust deep learning. In *ICML*, 2019a.

Song, H., Kim, M., Park, D., and Lee, J. Prestopping: How does early stopping help generalization against label noise? *CoRR*, abs/1911.08059, 2019b.

Song, H., Kim, M., Park, D., Shin, Y., and Lee, J. Robust learning by self-transition for handling noisy labels. In *KDD*, 2021.

Sture, H. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pp. 65–70, 1979.

Tanaka, D., Ikami, D., Yamasaki, T., and Aizawa, K. Joint optimization framework for learning with noisy labels. In *CVPR*, 2018.

Techapanurak, E. and Okatani, T. Hyperparameter-free out-of-distribution detection using softmax of scaled cosine similarity. *CoRR*, abs/1905.10628, 2019.

Veit, A., Alldrin, N., Chechik, G., Krasin, I., Gupta, A., and Belongie, S. J. Learning from noisy large-scale datasets with minimal supervision. In *CVPR*, 2017.

Vladimir, V. Conditional validity of inductive conformal predictors. *Machine Learning*, 92(2-3):349–376, 2013.

Wang, X., Hua, Y., Kodirov, E., Clifton, D. A., and Robertson, N. M. Proselflc: Progressive self label correction for training robust deep neural networks. In *CVPR*, 2021.

Wang, Y., Liu, W., Ma, X., Bailey, J., Zha, H., Song, L., and Xia, S. Iterative learning with open-set noisy labels. In *CVPR*, 2018.

Wang, Y., Ma, X., Chen, Z., Luo, Y., Yi, J., and Bailey, J. Symmetric cross entropy for robust learning with noisy labels. In *ICCV*, 2019.

Wei, H., Tao, L., Xie, R., and An, B. Open-set label noise can improve robustness against inherent label noise. In *NeurIPS*, 2021.

Wei, H., Xie, R., Cheng, H., Feng, L., An, B., and Li, Y. Mitigating neural network overconfidence with logit normalization. In *ICML*, 2022.

Welinder, P., Branson, S., Belongie, S. J., and Perona, P. The multidimensional wisdom of crowds. In *NeurIPS*, 2010.

Xiao, T., Xia, T., Yang, Y., Huang, C., and Wang, X. Learning from massive noisy labeled data for image classification. In *CVPR*, 2015.

Yao, J., Wang, J., Tsang, I. W., Zhang, Y., Sun, J., Zhang, C., and Zhang, R. Deep learning from noisy image labels with quality embedding. *IEEE Transactions on Image Processing*, 28(4), 2019.

Yi, K. and Wu, J. Probabilistic end-to-end noise correction for learning with noisy labels. In *CVPR*, 2019.

Yoav, B. and Daniel, Y. The control of the false discovery rate in multiple testing under dependency. *Annals of Statistics*, pp. 1165–1188, 2001.

Yoav, B. and Yosef, H. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300, 1995.

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

Zhang, C., Bengio, S., Hardt, M., Mozer, M. C., and Singer, Y. Identity crisis: Memorization and generalization under extreme overparameterization. In *ICLR*, 2020a.

Zhang, M., Lee, J., and Agarwal, S. Learning from noisy labels with no change to the training process. In *ICML*, 2021.

Zhang, Z. and Sabuncu, M. Generalized cross entropy loss for training deep neural networks with noisy labels. In *NeurIPS*, 2018.

Zhang, Z., Zhang, H., Arik, S. Ö., Lee, H., and Pfister, T. Distilling effective supervision from severe label noise. In *CVPR*, 2020b.

Zheng, S., Wu, P., Goswami, A., Goswami, M., Metaxas, D. N., and Chen, C. Error-bounded correction of noisy labels. In *ICML*, 2020.

Zhu, Z., Song, Y., and Liu, Y. Clusterability as an alternative to anchor points when learning with noisy labels. In *ICML*, 2021.

Zhu, Z., Dong, Z., and Liu, Y. Detecting corrupted labels without training a model to predict. In *ICML*, 2022.

A. Synthetic Noisy Label Generation Process

Since CIFAR-10 and CIFAR-100 are assumed to have no noisy labels, we manage three types of noisy label for noisy label injection: the symmetric label noise, the asymmetric label noise, and the instance-dependent label noise. We explain details of these noisy label generation processes.

Both the symmetric and asymmetric noise models follow the class-dependent assumption (i.e., the label noise depends only on the clean class: $\mathbb{P}[\tilde{Y}|X, Y] = \mathbb{P}[\tilde{Y}|Y]$). Specially, the symmetric noise is generated by uniform flipping (i.e., randomly flipping a true label to the other possible classes with probability η) (Cheng et al., 2021). The asymmetric noise is generated by pair-wise flipping (i.e., randomly flipping true label i to the next class $(i \mod K) + 1$). Denote by d the dimension of features. The instance-dependent label noise is synthesized by randomly generating a $d \times K$ projection matrix w_i for each class i and project each incoming feature with true class y_i onto each column of w_{y_i}. Instance n is more likely to be flipped to class j if the projection value of x_n on the j-th column of w_{y_n} is high. We follow the instance-dependent noise generation process utilized at Zhu et al. (2021) illustrated as Algorithm 2.

Algorithm 2 Instance Dependent Noise Generation Process

```
Require: Clean examples $(x_n, y_n)_{n=1}^N$;, Noise rate $\eta$
1: Sample instance flip rates $q \in \mathbb{R}^N$ from the truncated normal distribution $N(\eta, 0.1^2, [0, 1])$ with mean $\eta$, variance $0.1^2$
   and truncation interval $[0, 1]$;
2: Independently sample $w_1, w_2, ..., w_K$ from the standard normal distribution $N(0, 1^2)$ with mean zero and variance 1;
3: for $n = 1, 2, ..., N$ do
4:     $p = x_n \times w_{y_n}$;
5:     $p_{y_n} = -\inf$;
6:     $p = q_n \times softmax(p)$;
7:     $p_{y_n} = 1 - q_n$;
8:     Randomly choose a label from the label space according to the possibilities $p$ as noisy label $\tilde{y}_n$;
9: end for
Output: Noisy examples $(x_n, \tilde{y}_n)_{n=1}^N$
```

B. Details of Datasets and Implementation

To facilitate a fair comparison, we implement same backbone model for all methods. Following Cheng et al. (2021), we use ResNet34 for CIFAR-10 and CIFAR-100 and ResNet50 for Clothing1M.

We use the same set of hyper-parameters for CIFAR-10 and CIFAR-100. During training the model, we set a batch size of 128. We use Stochastic Gradient Descent (SGD) with a weight decay 5×10^{-4} and a momentum of 0.9. We train the model for 200 epochs. We set the initial learning rate of 0.1 and decrease it by the factor of 10 after 160 epochs.

For Clothing1M, we resize the image to 256×256, crop the middle 224×224 as input and perform normalization. We first perform noisy label detection on 1 million noisy training examples, then train only with the selected clean data to check the effectiveness. Particularly, we first train the neural network on the clean training set to obtain the score function. The neural network is trained for 80 epochs on 47,570 clean training images. Batch-size is set to 32. The initial learning rate is set as 2×10^{-3} and reduced by a factor of 10 at 40 epochs. We set the weight decay as 1×10^{-3}. We use 14,313 clean validation images as the calibration set. We perform BH on 1 million noisy training examples and train on the selected clean data to check the test accuracy of the final classifier. In the training process on selected data, for each epoch, we sample 1000 mini-batches from the training data while ensuring the (noisy) labels are balanced. Other training details are the same as those when training on the clean training set.

C. Baseline Description

We compare BH with four state-of-the-art baselines for learning with noisy labels: **CORES** (Cheng et al., 2021), **Confident Learning (CL)** (Northcutt et al., 2021), **SimiFeat-V and SimiFeat-R** (Zhu et al., 2022).

- **CORES** (Cheng et al., 2021): This work trains ResNet34 on the noisy dataset and uses its proposed sample sieve to filter out the corrupted examples. We adopt its default setting during training and calculate the F1 of the sieved out corrupted examples.

- **Confident Learning (CL)** (Northcutt et al., 2021): This work detects corrupted labels by firstly estimating probabilistic thresholds to characterize the label noise, ranking examples based on model predictions, then filtering out corrupted examples based on ranking and thresholds. We adopt its default hyper-parameter setting to train neural networks.
- **SimiFeat-V and SimiFeat-R** (Zhu et al., 2022): These two methods are based on the assumption that "closer" instances are more likely to share the same clean label. SimiFeat-V uses "local voting" via checking the noisy label consensuses of nearby features to determine if the example is corrupted. SimiFeat-R scores each instance based on the neighborhood information and filters out a guaranteed number of instances that are likely to be corrupted.

Since these methods all overlook the possible clean data, we evaluate these methods on the same noisy data set $\tilde{D}_1$.

D. Noisy Label Detection Performance using ResNet18 and ResNet50

To further demonstrate the superiority of BHN in noisy label detection, we evaluate the noisy label detection performance of BHN on CIFAR-10 and CIFAR-100 with different types of noise using ResNet18 and ResNet50, respectively. As illustrated in Table 5, our BHN also consistently outperforms all baselines in terms of FDR and F1 in all cases. For example, using ResNet50, BHN improves the F1 from 75.30% to 91.81% on CIFAR-10 under Inst. 0.4 noise, a direct improvement of **16.51**%; moreover, BHN reduces the FDR by **15.29**% compared with the best baseline. Using ResNet18, BHN outperforms the best baseline by **17.68**% in F1 on CIFAR-100 with Asym. 0.3 noise. Furthermore, when counting the frequency of reaching top-2 recall, we find that BHN consistently outperforms all other baselines under ResNet18 and ResNet50. The superior performances at various network depths demonstrate the effectiveness and general applicability of BHN.

Table 5. Comparisons of FDR, Recall, and F1 on CIFAR-10 and CIFAR-100. All methods use ResNet18 and ResNet50 as the backbone, respectively. ↑ indicates that larger values are better, and vice versa. Top 2 results are marked in **bold.**

	CIFAR-10									**CIFAR-100**								
Method	**Symm. 0.6**			**Asym. 0.3**			**Inst. 0.4**			**Symm. 0.6**			**Asym. 0.3**			**Inst. 0.4**		
	FDR↓	Recall↑	F1↑	FDR↓	Recall↑	F1↑	FDR↓	Recall↑	F1↑	FDR↓	Recall↑	F1↑	FDR↓	Recall↑	F1↑	FDR↓	Recall↑	F1↑
ResNet18																		
CORES	**7.07**	88.04	**90.42**	48.65	3.82	7.11	**13.86**	28.52	42.85	**4.96**	34.29	50.40	62.50	0.03	0.07	**7.64**	1.11	2.19
CL	40.14	85.04	70.26	70.21	87.47	44.44	60.46	85.07	54.10	39.67	**88.53**	71.76	70.25	**93.40**	45.13	60.52	**91.45**	55.15
SimiFeat-V	12.87	89.58	88.34	**36.71**	83.59	**72.04**	30.25	74.68	72.13	11.31	52.79	66.19	**49.64**	71.62	**59.14**	35.54	68.25	**66.30**
SimiFeat-R	31.02	**98.63**	81.18	44.86	**91.20**	68.73	34.81	**89.87**	**75.56**	20.88	**99.39**	**82.23**	60.40	**91.43**	55.27	48.90	**93.01**	65.96
BHN (ours)	**4.00**	**96.40**	**96.20**	**7.40**	**92.40**	**92.50**	**6.30**	**91.40**	**92.50**	**3.68**	80.82	**87.89**	**6.90**	65.39	**76.82**	**6.15**	72.73	**81.95**
ResNet50																		
CORES	24.10	85.53	80.43	60.18	8.00	13.32	**21.22**	37.78	51.07	**6.34**	64.50	76.39	61.32	0.46	0.90	**14.73**	7.39	13.60
CL	39.92	84.92	70.37	70.28	87.26	44.33	60.44	85.28	54.04	39.66	**89.15**	71.97	70.23	**94.47**	45.28	60.68	**91.98**	55.09
SimiFeat-V	**12.73**	88.70	**87.98**	**36.55**	82.53	**71.74**	30.56	74.30	71.78	12.04	57.78	69.75	**50.26**	71.78	**58.76**	36.56	71.54	**67.25**
SimiFeat-R	26.83	**98.25**	83.88	43.87	**90.25**	69.21	34.70	**88.93**	**75.30**	28.21	**99.02**	**83.23**	59.57	**88.81**	55.57	47.42	**92.21**	66.97
BHN (ours)	**3.84**	**94.83**	**95.49**	**7.01**	**90.17**	**91.56**	**5.93**	**89.66**	**91.81**	**4.10**	78.21	**86.16**	**7.63**	64.48	**75.95**	**6.64**	71.90	**81.24**

E. Classification Performance on Clean Test Sets using ResNet18 and ResNet50

We further verify whether BHN improves the classification performance of the final classifier using ResNet18 and ResNet50 on CIFAR-10 and CIFAR-100, respectively. For all methods, we remove the detected corrupted examples, and use the remaining examples to train the classifier. Our results in Table 6 show that BHN consistently outperforms all other baselines in all cases with synthetic noise. For example, we can observe that BHN outperforms the best baseline (SimiFeat-V) by **7.57%** in the test accuracy on CIFAR-100 with Inst. 0.4 noise under ResNet18, while surpasses SimiFeat-V by **5.21%** in terms of test accuracy on CIFAR-10 with Asym. 0.3 noise under ResNet50.

Table 6. Comparisons of test accuracy of the final classifier (%) on CIFAR-10 and CIFAR-100. All the methods use ResNet18 and ResNet50 as the backbone, respectively. Top 1 results are marked in **bold**.

	CIFAR-10			**CIFAR-100**		
Method	**Symm. 0.6**	**Asym. 0.3**	**Inst. 0.4**	**Symm. 0.6**	**Asym. 0.3**	**Inst. 0.4**
			ResNet18			
CORES	76.20	65.97	63.51	30.62	50.46	44.18
CL	25.55	49.37	35.77	7.98	12.39	11.22
SimiFeat-V	75.31	81.03	74.47	39.56	54.56	51.89
SimiFeat-R	58.56	82.34	77.93	35.55	48.36	45.55
BHN (ours)	**86.65**	**90.14**	**88.09**	**49.56**	**62.46**	**59.46**
			ResNet50			
CORES	68.34	70.15	66.31	37.75	47.71	38.98
CL	26.88	48.87	32.35	6.84	9.74	8.13
SimiFeat-V	75.42	82.68	76.36	39.27	55.26	51.07
SimiFeat-R	66.42	82.36	77.69	36.78	49.79	43.18
BHN (ours)	**85.05**	**87.89**	**86.58**	**48.57**	**62.71**	**58.70**

F. Alternative Score Function

We further evaluate the noisy label detection performance of BHN-cosin on the CIFAR datasets under the ResNet34 and ResNet50 backbone, respectively. The results are provided in Tables 7 and 8. Again, BHN-cosin achieves almost the same performance as BHN. which suggest that the norm of softmax logits has little effect on the performance of noisy label detection. To explore this phenomenon further we contrast the norms of softmax logits on clean and corrupted examples using neural networks with different test accuracy in Figure 4. The results show that the norms on clean and corrupted examples consistently have almost the same densities under neural networks with different test accuracy, and moreover that the norms of softmax logits cannot enhance the discrimination between clean and corrupted examples. Therefore, BHN and BHN-cosin achieve similar performance.

Table 7. Comparisons of FDR, Recall, and F1, and test accuracy (%) of BHN and BHN-cosin on CIFAR-10. All methods are based on the ResNet34 and ResNet50 network, respectively.

Method	**Symm. 0.6**				**Asym. 0.3**				**Inst. 0.4**			
	FDR$\downarrow$	Recall$\uparrow$	F1$\uparrow$	Accuracy $\uparrow$	FDR$\downarrow$	Recall$\uparrow$	F1$\uparrow$	Accuracy $\uparrow$	FDR$\downarrow$	Recall$\uparrow$	F1$\uparrow$	Accuracy $\uparrow$
						ResNet34						
BHN	3.80	95.90	96.05	86.30	7.17	91.13	91.98	89.79	5.98	91.18	92.58	88.63
BHN-cosin	3.78	95.82	96.02	86.15	7.25	91.00	91.87	90.07	5.98	90.85	92.41	87.89
						ResNet50						
BHN	3.84	94.83	95.49	85.05	7.01	90.17	91.56	87.89	5.93	89.66	91.81	86.58
BHN-cosin	3.84	94.72	95.44	85.61	7.04	90.19	91.56	88.10	5.96	89.64	91.79	87.18

G. Proof

Theorem 3.7 (Empirical p-values based on the score function are PRDS). Assume that $\hat{s}(\tilde{Z})$ is continuously distributed. Consider N noisy examples $\tilde{z}_1, \ldots, \tilde{z}_N$ where the clean examples are jointly independent of each other and of the data in

Table 8. Comparisons of FDR, Recall, F1 and test accuracy (%) of BHN and BHN-cosin on CIFAR-100. All methods are based on the ResNet34 and ResNet50 network, respectively.

Method	**Symm. 0.6**				**Asym. 0.3**				**Inst. 0.4**			
	FDR↓	Recall↑	F1↑	Accuracy ↑	FDR↓	Recall↑	F1↑	Accuracy ↑	FDR↓	Recall↑	F1↑	Accuracy ↑
							ResNet34					
BHN	4.10	81.31	88.01	46.16	7.79	67.14	77.71	61.46	6.48	72.99	81.99	58.06
BHN-cosin	4.18	78.28	86.17	44.68	7.95	64.10	75.57	62.02	6.39	69.11	79.51	56.82
							ResNet50					
BHN	4.10	78.21	86.16	48.57	7.63	64.48	75.95	62.71	6.64	71.90	81.24	58.70
BHN-cosin	4.15	76.42	85.04	45.13	7.73	62.28	74.36	60.98	6.27	68.59	79.21	56.72

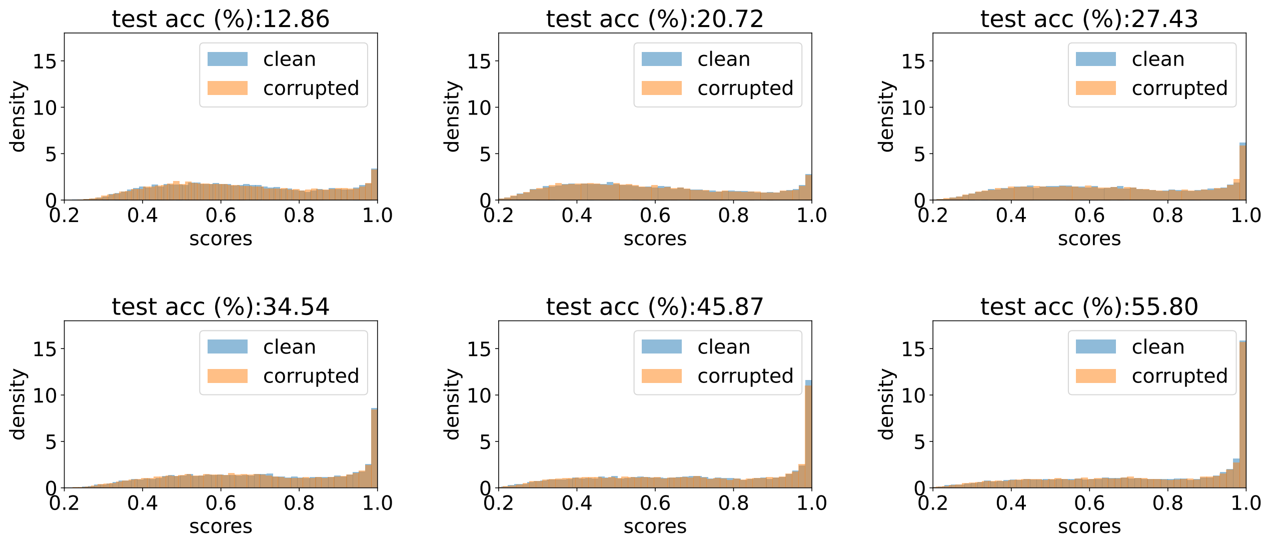

Figure 4. The norms of softmax logits on clean and corrupted examples on CIFAR-100 with the Symm. 0.6 noise using ResNet18 with different test accuracy.

D_0. Then the p-values $(p(\tilde{Z}_1), \dots, p(\tilde{Z}_N))$ are PRDS on the set of clean examples.

Definition 3.5 (Increasing set). For vectors a and b of equal dimension, we say $a \succeq b$ if every coordinate of a is no smaller than the corresponding coordinate of b, and a set $A \subset \mathbb{R}^N$ is *increasing* if $a \in A$ and $b \succeq a$ implies $b \in A$.

Definition 3.6 (PRDS). A random vector $V = (V_1, \dots, V_N)$ is PRDS on a set $I_0 \subseteq \{1, \dots, N\}$ if for any $i \in I_0$ and any increasing set A, the probability $\mathbb{P}[V \in A | V_i = v]$ is increasing in v.

In the multiple testing literature, $(p(\tilde{Z}_1), p(\tilde{Z}_2), \dots, p(\tilde{Z}_N))$ is often said to be PRDS if it is PRDS on the set of nulls.

Proof of Theorem 3.7. Let $S = (S_{(1)}, \dots, S_{(m)})$ be the order statistics of $(\hat{s}(Z_i^0))_{i \in \{m+1, \dots, 2m\}}$, the noisy label detection scores evaluated on the calibration set. Let $Y = (p_1, \dots, p_N)$ be the empirical p-values evaluated on the noisy set (i.e., $p_j = p(\tilde{Z}_j)$). Then, for any increasing set A, we have

$$\begin{aligned}\mathbb{P}[Y \in A \mid Y_i = y] &= \int \mathbb{P}[Y \in A \mid S = s] \, \mathbb{P}[S = s \mid Y_i = y] \, ds \\ &= \mathbb{E}_{S|Y_i = y}\left[\mathbb{P}[Y \in A \mid S]\right].\end{aligned}$$

With this representation, the conclusion will be implied by the following two lemmas.

Lemma G.1. *For an increasing set A and vectors s, s' such that $s \preceq s'$, then*

$$\mathbb{P}[Y \in A \mid S = s] \geq \mathbb{P}[Y \in A \mid S = s'].$$

Lemma G.2. *For $y \geq y'$, if i belongs to the set of nulls, there exists $S_1 \sim S \mid Y_i = y$ and $S_2 \sim S \mid Y_i = y'$ such that* $\mathbb{P}[S_1 \preceq S_2] = 1$.

In other words, Lemma G.1 states that the p-values increase as the scores on the calibration set decrease, while Lemma G.2 states that a larger p-value indicates the scores on the calibration set are smaller. The proof of Theorem 3.7 follows easily from Lemmas G.1 and G.2. Take any $y \geq y'$ and let S_1 and S_2 be as in the statement of Lemma G.2. Then, for any i belonging to the set of nulls,

$$\begin{aligned}\mathbb{P}[Y \in A \mid Y_i = y] &= \mathbb{E}_{S_1}\left[\mathbb{P}[Y \in A \mid S = S_1]\right] \\ &\geq \mathbb{E}_{S_2}\left[\mathbb{P}[Y \in A \mid S = S_2]\right] \\ &= \mathbb{P}[Y \in A \mid Y_i = y'].\end{aligned}$$

The inequality follows from Lemma G.1 and the fact that $\mathbb{P}[S_1 \preceq S_2] = 1$, which comes from Lemma G.2. □

Lemma G.1 follows immediately from the definition of empirical p-value in Equation (6). Lemma G.2 is proved below.

Proof of Lemma G.2. Since $\hat{s}(\tilde{Z})$ is continuously distributed, we can assume without loss of generality that the scores S_i follow the uniform distribution on $[0, 1]$. Let $S'_{(1)} \leq S'_{(2)} \leq \dots \leq S'_{(m+1)}$ be the order statistics of $(\hat{s}(Z^0_{m+1}), \dots, \hat{s}(Z^0_{2m}), \hat{s}(\tilde{Z}_1))$ and R_{2m+1} be the rank of $\hat{s}(\tilde{Z}_1)$ among these. By definition,

$$\left\{(S_{(1)}, \dots, S_{(m)}) \mid R_{2m+1} = k, S'_{(1)}, \dots, S'_{(m+1)}\right\} = (S'_{(1)}, \dots, S'_{(k-1)}, S'_{(k+1)}, \dots, S'_{(m+1)}).$$

Since $\hat{s}(\tilde{Z})$ is continuously distributed, R_{2m+1} is independent of $(S'_{(1)}, S'_{(2)}, \dots, S'_{(m+1)})$. As a result, for any positive integer $k \leq m + 1$,

$$\left\{(S_{(1)}, \dots, S_{(m)}) \mid R_{2m+1} = k\right\} \overset{d}{=} (S'_{(1)}, \dots, S'_{(k-1)}, S'_{(k+1)}, \dots, S'_{(m+1)}).$$

The right-hand-side is clearly entry-wise non-increasing in k. Since $p_1 = R_{2m+1}/(m+1)$, Lemma G.2 is proved for $i = 1$. The same proof carries over to other indices i belonging to the set of clean examples.

□