Making Deep Neural Networks Robust to Label Noise: a Loss Correction Approach

Giorgio Patrini[1,2], Alessandro Rozza[3], Aditya Krishna Menon[2,1], Richard Nock[2,1,4], Lizhen Qu[2,1]
[1]Australian National University, [2]Data61, [3]Waynaut, [4]University of Sydney
{name.surname}@data61.csiro.au, alessandro.rozza@waynaut.com

Abstract

We present a theoretically grounded approach to train deep neural networks, including recurrent networks, subject to class-dependent label noise. We propose two procedures for loss correction that are agnostic to both application domain and network architecture. They simply amount to at most a matrix inversion and multiplication, provided that we know the probability of each class being corrupted into another. We further show how one can estimate *these probabilities, adapting a recent technique for noise estimation to the multi-class setting, and thus providing an end-to-end framework. Extensive experiments on MNIST, IMDB, CIFAR-10, CIFAR-100 and a large scale dataset of clothing images employing a diversity of architectures — stacking dense, convolutional, pooling, dropout, batch normalization, word embedding, LSTM and residual layers — demonstrate the noise robustness of our proposals. Incidentally, we also prove that, when ReLU is the only non-linearity, the loss curvature is immune to class-dependent label noise.*

1. Introduction

Large datasets used in training modern machine learning models, such as deep neural networks, are often affected by label noise. The problem is pervasive for a simple reason: manual expert-labelling of each instance at a large scale is not feasible, and so researchers often resort to cheap but imperfect surrogates. Two such popular surrogates are crowdsourcing using non-expert labellers and — especially for images — the use of search engines to query instances by a keyword, assuming the keyword as a valid label [5, 35, 3, 29, 17] Both approaches offer the possibility to scale the acquisition of training labels, but invariably result in the introduction of label noise, which may adversely affect model training.

Our goal is to effectively train deep neural networks with modern architectures under label noise. We do so by marrying two different lines of recent research. The first strand is work on *ad-hoc deep architectures* tailored to the problem, primarily developed in Computer Vision [27, 32, 39, 42]. While some such approaches have shown good experimental performance on specific domains, they lack a solid theoretical framework and often need a large amount of clean labels to obtain acceptable results — in particular, for pre-training or validating hyper-parameters [42, 17, 32].

The second strand is recent Machine Learning research on *theoretically grounded means of combating label noise*. In particular, we are interested in the design of *corrected losses* that are *robust* to label noise [38, 28, 30]. Despite their formal guarantees, these methods have not been fully appreciated in practice because, crucially, they require noise rates to be known *a priori*.

An estimate of the noise is often available to practitioners by polishing a subset of the training data [42] — which is useful and often necessary for model selection. Yet, interestingly, recent work has provided practical algorithms for estimating the noise rates [36, 34, 21, 26, 31]; remarkably, this is achievable with *absolutely no knowledge of ground truth labels*. To our knowledge, no prior work has combined those estimators with loss correction techniques, nor has either idea been applied to modern deep architectures. Our contributions aim to unify these research streams:

- We introduce two alternative procedures for loss correction, provided that we know a stochastic matrix T summarizing the probability of one class being flipped into another under noise. The first procedure, a multi-class extension of [28, 30] applied to neural networks, is called "*backward*" as it multiplies the loss by T^{-1}. The second, inspired by [39], is named "*forward*" as it multiplies the network predictions by T.

- We prove that both procedures enjoy formal robustness guarantees *w.r.t.* the clean data distribution. Since we only operate on the loss function, the approach is both architecture and application domain independent, as well as viable for any chosen loss function.

- We take a further step and extend the noise estimator of [26] to our multi-class setting, thus formulating an

end-to-end solution to the problem.

- We prove that for ReLU networks the Hessian of the loss is independent from label noise.

We apply our loss corrections to image recognition on MNIST, CIFAR-10, CIFAR-100 and sentiment analysis on IMDB; we simulate corruption by artificially injecting noise on the training labels. In order to show that no architectural choice is the secret ingredient of our robustification recipe, we experiment with a variety of network modules currently in fashion: convolutions and pooling [20], dropout [37], batch normalization [15], word embedding and residual units [11, 12]. Additional tests on LSTM [13] confirm that the procedures can be seamlessly applied to recurrent neural networks as well. Comparisons with non-corrected losses and several known methods confirm robustness of our two procedures, with the forward correction dominating the backward. Unsurprisingly, the noise estimator is the bottleneck in obtaining near-perfect robustness, yet in most experiments our approach is often the best compared to prior work. Finally, we experiment with Clothing1M, the $1M$ clothing images dataset of [42], and establish the new state of the art.

2. Related work

Our work leverages recent research in a number of different areas, summarized below.

Noise robustness[1]. Learning with noisy labels has been widely investigated in the literature [7]. From the theoretical standpoint label noise has been studied in two different regimes, with vastly different conclusions. In the case of low-capacity (typically linear) models, even mild symmetric, *i.e.* class-independent (versus asymmetric, *i.e.* class-dependent), label noise can produce solutions that are akin to random guessing [22]. On the other hand, the Bayes-optimal classifier remains unchanged under symmetric [28, 26] and even instance dependent label noise [25] implying that high-capacity models are robust to essentially any level of such noise, given sufficiently many samples.

Surrogate losses. Suppose one wishes to minimize a loss ℓ on clean data. When the level of noise is known *a priori*, [28] provided the general form of a *noise corrected* loss $\hat{\ell}$ such that minimization of $\hat{\ell}$ on noisy data is *equivalent* to minimization of ℓ on clean data. In the idealized case of symmetric label noise, for certain ℓ one in fact does not need to know the noise rate: [8] gives a sufficient condition for which ℓ is robust, and several examples of such robust non-convex losses, while [41] shows that the (convex) linear or *unhinged* loss is its own noise-corrected loss. Another robust non-convex loss is given in [24].

[1]We use the term robustness in its meaning of immunity to noise and not generically as "adaptivity to various scenarios", *e.g.* [6].

Noise rate estimation. Recent work has provided methods to estimate label flip probabilities directly from noisy samples. Typically, it is required that the generating distribution is such that for each class, there exists some "perfect" instance, *i.e.* one that is classified with probability equal to one. Proposed estimators involve either the use of kernel mean embedding [31], or post-processing the output of a standard class-probability estimator such as logistic regression using order statistics on the range of scores [21, 26] or the slope of the induced ROC curve [34].

Deep learning with noisy labels. Several works in Deep Learning have attempted to deal with noisy labels of late, especially in Computer Vision. This is often achieved by formulating noise-aware models. [27] builds a noise model for binary classification of aerial image patches, which can handle omission and wrong location of training labels. [42] constructs a more sophisticated mix of symmetric, asymmetric and instance-dependent noise; two networks are learned by EM as models for classifier and noise type. It is often the case that a small set of clean labels is needed in order either to pre-train or fine-tune the model [42, 17, 32].

The work of [39] deserves a particular mention. The method augments the architecture by adding a linear layer on top of the network. Once learned, this layer plays the role of our matrix T. However, learning this architecture appears problematic; heuristics such as trace regularization and a fixed updating schedule for the linear layer are necessary. We sidestep those issues by decoupling the two phases: we first estimate T and then learn with loss correction.

We are not aware of any other attempt at either applying the noise-corrected loss approach of [28] to neural networks, nor on combining those losses with the above noise rate estimators. Our work sits precisely in this intersection. Note that, even though in principle loss correction should not be necessary for high-capacity models like deep neural networks, owing to aforementioned theoretical results, in practice, such correction may offset the sub-optimality of these models arising from training on finite samples. Specifically, we expect that directly optimizing the (corrected) objective we care about will be beneficial in the finite-sample case.

3. Preliminaries

We begin by fixing notation. We let $[c] \doteq \{1, \dots, c\}$ for any c positive integer. Column vectors are written in bold (*e.g.* $\boldsymbol{v}$) and matrices in capitals (*e.g.* V). Coordinates of a vector are denoted by a subscript (*e.g.* $\boldsymbol{v}_j$), while rows and columns of a matrix are denoted *e.g.* $V_{j\cdot}$ and $V_{\cdot j}$ respectively. We denote the all-ones vector by $\mathbf{1}$, with size clear from context, and $\Delta^{c-1} \subset [0,1]^c$ the c-dimensional simplex.

In supervised c-class classification, one has feature space $\mathcal{X} \subseteq \mathbb{R}^d$ and label space $\mathcal{Y} = \{\boldsymbol{e}^i : i \in [c]\}$, where $\boldsymbol{e}^i$ denotes the ith standard canonical vector in $\mathbb{R}^c$ by , *i.e.* $\boldsymbol{e}^i \in \{0,1\}^c, \mathbf{1}^\top \boldsymbol{e}^i = 1$. One observes examples $(\boldsymbol{x}, \boldsymbol{y})$ drawn

from an unknown distribution $p(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})$ over $\mathcal{X} \times \mathcal{Y}$. We denote expectations over $p(\boldsymbol{x}, \boldsymbol{y})$ by $\mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}$. Note that each $\boldsymbol{y}$ only has one non-zero value at the coordinate corresponding to the underlying label.

An n-layer neural network[2] comprises a transformation $\boldsymbol{h}\colon \mathcal{X} \to \mathbb{R}^c$, where $\boldsymbol{h} = (\boldsymbol{h}^{(n)} \circ \boldsymbol{h}^{(n-1)} \circ \cdots \circ \boldsymbol{h}^{(1)})$ is the composition of a number of intermediate transformations — the layers — defined by:

$$(\forall i \in [n-1])\, \boldsymbol{h}^{(i)}(\boldsymbol{z}) = \boldsymbol{\sigma}(W^{(i)}\boldsymbol{z} + \boldsymbol{b}^{(i)}) \ ,$$
$$\boldsymbol{h}^{(n)}(\boldsymbol{z}) = W^{(i)}\boldsymbol{z} + \boldsymbol{b}^{(i)} \ .$$

where $W^{(i)} \in \mathbb{R}^{d^{(i)} \times d^{(i-1)}}$ and $\boldsymbol{b}^{(i)} \in \mathbb{R}^{d^{(i)}}$ are parameters to be estimated[3], and $\boldsymbol{\sigma}$ is any activation function that acts *coordinate-wise*, such as the ReLU $\boldsymbol{\sigma}(\boldsymbol{x})_i = \max(0, \boldsymbol{x}_i)$. Observe that the final layer applies a *linear* projection, unlike all preceding layers. To simplify notation, we write:

$$(\forall i \in [n])\, \boldsymbol{x}^{(i)} \doteq \boldsymbol{h}^{(i)}(\boldsymbol{x}^{(i-1)}),$$

with the base case $\boldsymbol{x}^{(0)} \doteq \boldsymbol{x}$, so that *e.g.* $\boldsymbol{x}^{(1)}$ is exactly the representation in the first layer. The coordinates of $\boldsymbol{h}(\boldsymbol{x})$ represent the relative weights that the model assigns to each class $i = 1, \ldots, c$ to be predicted. The predicted label is thus given by $\arg\max_{i \in [c]} \boldsymbol{h}_i(\boldsymbol{x})$. In the training phase, the output of the final layer is contrasted with the true label $\boldsymbol{y}$ via two steps. First, $\boldsymbol{h}(\cdot)$ passes through the *softmax* function $e^{\boldsymbol{h}_i(\boldsymbol{x})} / \sum_{k=1}^{c} e^{\boldsymbol{h}_k(\boldsymbol{x})}$. The softmax output can be interpreted as a vector approximating the class-conditional probabilities $p(\boldsymbol{y}|\boldsymbol{x})$; we denote it by $\hat{p}(\boldsymbol{y}|\boldsymbol{x}) \in \Delta^{c-1}$. Next, we measure the discrepancy between label $\boldsymbol{y} = \boldsymbol{e}^i$ and network output by a loss function $\ell\colon \mathcal{Y} \times \Delta^{c-1} \to \mathbb{R}$, for example by means of *cross-entropy*:

$$\ell(\boldsymbol{e}^i, \hat{p}(\boldsymbol{y}|\boldsymbol{x})) = -(\boldsymbol{e}^i)^\top \log \hat{p}(\boldsymbol{y}|\boldsymbol{x}) = -\log \hat{p}(\boldsymbol{y} = \boldsymbol{e}^i|\boldsymbol{x}) \ . \tag{1}$$

With some abuse of notation, we also define a loss in vector form $\boldsymbol{\ell} : \Delta^{c-1} \to \mathbb{R}^c$, computed on every possible label:

$$\boldsymbol{\ell}(\hat{p}(\boldsymbol{y}|\boldsymbol{x})) = \big(\ell(\boldsymbol{e}^1, \hat{p}(\boldsymbol{y}|\boldsymbol{x})), \ldots, \ell(\boldsymbol{e}^c, \hat{p}(\boldsymbol{y}|\boldsymbol{x}))\big)^\top \in \mathbb{R}^c \ . \tag{2}$$

In the following, formal results hold under very mild conditions on a generic loss function ℓ; at times we provide examples for the cross-entropy. For simplicity, one could think of cross-entropy every time ℓ is mentioned.

[2] *W.l.o.g.*, we assume all layers to be *fully connected*, or dense; for example, convolutions can be represented by dense layers with shared sparse weights.

[3] Here, $d^{(0)} = d$, the original feature dimensionality, and $d^{(n)} = c$, the label dimensionality.

4. Label noise and loss robustness

We now consider label noise. We assume the *asymmetric*, *i.e.* class-conditional noise setting [28], where each label $\boldsymbol{y}$ in the training set is flipped to $\tilde{\boldsymbol{y}} \in \mathcal{Y}$ with probability $p(\tilde{\boldsymbol{y}}|\boldsymbol{y})$; feature vectors are untouched. Thus, we observe samples from a distribution $p(\boldsymbol{x}, \tilde{\boldsymbol{y}}) = \sum_{\boldsymbol{y}} p(\tilde{\boldsymbol{y}}|\boldsymbol{y})p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})$. Denote by $T \in [0,1]^{c \times c}$ the noise transition matrix specifying the probability of one label being flipped to another, so that $\forall i, j \ T_{ij} = p(\tilde{\boldsymbol{y}} = \boldsymbol{e}^j | \boldsymbol{y} = \boldsymbol{e}^i)$. The matrix is row-stochastic and not necessarily symmetric across the classes.

This is an approximation of real-world corruption which can still be useful in certain scenarios. One such case is that of classes representing a fine-grained hierarchy of concepts, for example dog breeds and bird species [17] or narrow categories of clothing [42]. Classes may be too similar between each other for non-expert human labellers to distinguish, regardless of the specific instances. Little is known about learning under the more generic *feature dependent* noise, with few exceptions [42, 8, 25].

We aim to modify a loss ℓ so as to make it robust to asymmetric label noise; in fact, this is possible if T is known. Under this assumption — that we relax later on — we introduce two alternative corrections inspired by [28] and [39].

4.1. The backward correction procedure

We can build an *unbiased estimator* of the loss function, such that *under expected label noise* the corrected loss equals the original one computed on clean data. This property is stated in the next Theorem, a multi-class generalization of [28, Theorem 1]. The Theorem is also a particular instance of the more abstract [40, Theorem 3.2].

Theorem 1 *Suppose that the noise matrix T is non-singular. Given a loss $\boldsymbol{\ell}$,* backward *corrected loss is defined as:*

$$\boldsymbol{\ell}^{\leftarrow}(\hat{p}(\boldsymbol{y}|\boldsymbol{x})) = T^{-1}\boldsymbol{\ell}(\hat{p}(\boldsymbol{y}|\boldsymbol{x})) \ .$$

Then, the loss correction is unbiased, i.e. *:*

$$\forall \boldsymbol{x}, \quad \mathbb{E}_{\tilde{\boldsymbol{y}}|\boldsymbol{x}}\, \boldsymbol{\ell}^{\leftarrow}(\boldsymbol{y}, \hat{p}(\boldsymbol{y}|\boldsymbol{x})) = \mathbb{E}_{\boldsymbol{y}|\boldsymbol{x}}\, \boldsymbol{\ell}(\boldsymbol{y}, \hat{p}(\boldsymbol{y}|\boldsymbol{x})) \ ,$$

and therefore the minimizers are the same:

$$\operatorname*{argmin}_{\hat{p}(\boldsymbol{y}|\boldsymbol{x})} \mathbb{E}_{\boldsymbol{x},\tilde{\boldsymbol{y}}}\, \ell^{\leftarrow}(\boldsymbol{y}, \hat{p}(\boldsymbol{y}|\boldsymbol{x})) = \operatorname*{argmin}_{\hat{p}(\boldsymbol{y}|\boldsymbol{x})} \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}\, \ell(\boldsymbol{y}, \hat{p}(\boldsymbol{y}|\boldsymbol{x})) \, .$$

Proof. $\mathbb{E}_{\tilde{\boldsymbol{y}}|\boldsymbol{x}}\, \boldsymbol{\ell}^{\leftarrow}(\hat{p}(\boldsymbol{y}|\boldsymbol{x})) = \mathbb{E}_{\boldsymbol{y}|\boldsymbol{x}}\, T\boldsymbol{\ell}^{\leftarrow}(\hat{p}(\boldsymbol{y}|\boldsymbol{x})) = \mathbb{E}_{\boldsymbol{y}|\boldsymbol{x}}\, T\, T^{-1}\boldsymbol{\ell}(\hat{p}(\boldsymbol{y}|\boldsymbol{x})) = \mathbb{E}_{\boldsymbol{y}|\boldsymbol{x}}\, \boldsymbol{\ell}(\hat{p}(\boldsymbol{y}|\boldsymbol{x}))$. ■

The corrected loss is effectively a linear combination of the loss values for each observable label, whose coefficients are due to the probability that T^{-1} attributes to each possible true label $\boldsymbol{y}$, given the observed one $\tilde{\boldsymbol{y}}$. Intuitively, we are "going one step back" in the noise process described by

the Markov chain T. The corrected loss is differentiable — although not always non-negative — and can be minimized with any off-the-shelf algorithm for back-propagation. Although in practice T would be invertible almost surely, its condition number may be problematic. A simple solution is to mix T with the identity matrix before inversion; this may be seen as taking a more conservative noise-free prior.

4.2. The forward correction procedure

Alternatively, we can correct the model predictions. Following [39], we start by observing that a neural network learned with no loss correction would result in a predictor for noisy labels $\hat{p}(\tilde{\boldsymbol{y}}|\boldsymbol{x})$. We can make explicit the dependency on T. For instance, with cross-entropy we have:

$$\ell(\boldsymbol{e}^i, \hat{p}(\boldsymbol{y}|\boldsymbol{x})) = -\log \hat{p}(\tilde{\boldsymbol{y}} = \boldsymbol{e}^i|\boldsymbol{x}) \tag{3}$$

$$= -\log \sum\nolimits_{j=1}^{c} p(\tilde{\boldsymbol{y}} = \boldsymbol{e}^i|\boldsymbol{y} = \boldsymbol{e}^j)\, \hat{p}(\boldsymbol{y} = \boldsymbol{e}^j|\boldsymbol{x}) \tag{4}$$

$$= -\log \sum\nolimits_{j=1}^{c} T_{ji}\, \hat{p}(\boldsymbol{y} = \boldsymbol{e}^j|\boldsymbol{x}) \ , \tag{5}$$

or in matrix form $\boldsymbol{\ell}(\hat{p}(\boldsymbol{y}|\boldsymbol{x})) = -\log T^\top \hat{p}(\boldsymbol{y}|\boldsymbol{x})$. This loss compares the noisy label $\tilde{\boldsymbol{y}}$ to averaged noisy prediction corrupted by T. We call this procedure "forward" correction. In order to analyze its behavior, we first need to recall definition and properties of a broad family of losses named *proper composite* [33, Section 4]. Consider a *link function* $\boldsymbol{\psi} : \Delta^{c-1} \to \mathbb{R}^c$, invertible. Many losses are said to be *composite*, and denoted by $\ell_{\boldsymbol{\psi}} : \mathcal{Y} \times \mathbb{R}^c \to \mathbb{R}$, in the sense that they can be expressed by the aid of a link function as

$$\ell_{\boldsymbol{\psi}}(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \boldsymbol{\ell}(\boldsymbol{y}, \boldsymbol{\psi}^{-1}(\boldsymbol{h}(\boldsymbol{x}))) \ . \tag{6}$$

In the case of cross-entropy, the softmax is the *inverse* link function. When composite losses are also *proper* [33], their minimizer assumes the particular shape of the link function applied to the class-conditional probabilities $p(\boldsymbol{y}|\boldsymbol{x})$:

$$\operatorname*{argmin}_{\boldsymbol{h}} \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}\, \ell_{\boldsymbol{\psi}}(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \boldsymbol{\psi}(p(\boldsymbol{y}|\boldsymbol{x})) \ . \tag{7}$$

Cross-entropy and square are examples of *proper composite* losses. An intriguing robustness property holds for forward correction of proper composite losses.

Theorem 2 *Suppose that the noise matrix T is non-singular. Given a proper composite loss $\boldsymbol{\ell}_{\boldsymbol{\psi}}$, define the* forward *loss correction as:*

$$\boldsymbol{\ell}^{\rightarrow}_{\boldsymbol{\psi}}(\boldsymbol{h}(\boldsymbol{x})) = \boldsymbol{\ell}(T^\top \boldsymbol{\psi}^{-1}(\boldsymbol{h}(\boldsymbol{x}))) \ .$$

Then, the minimizer of the corrected loss under the noisy distribution is the same as the minimizer of the original loss under the clean distribution:

$$\operatorname*{argmin}_{\boldsymbol{h}} \mathbb{E}_{\boldsymbol{x},\tilde{\boldsymbol{y}}}\, \ell^{\rightarrow}_{\boldsymbol{\psi}}(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \operatorname*{argmin}_{\boldsymbol{h}} \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}\, \ell_{\boldsymbol{\psi}}(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) \ .$$

Proof. First notice that:

$$\ell^{\rightarrow}_{\boldsymbol{\psi}}(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \ell(\boldsymbol{y}, T^\top \boldsymbol{\psi}^{-1}(\boldsymbol{h}(\boldsymbol{x}))) = \ell_{\boldsymbol{\phi}}(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) \tag{8}$$

where we denote $\boldsymbol{\phi}^{-1} = \boldsymbol{\psi}^{-1} \circ T^\top$. Equivalently, $\boldsymbol{\phi} = (T^{-1})^\top \circ \boldsymbol{\psi}$ is invertible by composition of invertible functions, its domain is Δ^{c-1} as of $\boldsymbol{\psi}$ and its codomain is $\mathbb{R}^c$. The last loss in Equation 8 is therefore proper composite with link $\boldsymbol{\phi}$. Finally, from Equation 7, the loss minimizer over the noisy distribution is

$$\operatorname*{argmin}_{\boldsymbol{h}} \mathbb{E}_{\boldsymbol{x},\tilde{\boldsymbol{y}}}\, \ell_{\boldsymbol{\phi}}(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \boldsymbol{\phi}(p(\tilde{\boldsymbol{y}}|\boldsymbol{x})) \tag{9}$$

$$= \boldsymbol{\psi}((T^{-1})^\top p(\tilde{\boldsymbol{y}}|\boldsymbol{x})) = \boldsymbol{\psi}(p(\boldsymbol{y}|\boldsymbol{x})) \ , \tag{10}$$

that proves the Theorem by Equation 7 once again. ■

Recall that $\hat{p}(\boldsymbol{y}|\boldsymbol{x})$ approximates $p(\boldsymbol{y}|\boldsymbol{x})$ and thus we can relate to the result by taking any neural network that enough expressive. Although, the property is weaker than unbiasedness of Theorem 1. Robustness applies to the minimizer only, that is, the model learned by forward correction is the minimizer over the *clean* distribution. Yet, Theorem 2 guarantees noise robustness with no explicit matrix inversion; the "de-noising" link function $\boldsymbol{\phi}$ does it behind the scene. This turns out to be an important factor in practice; see below.

4.3. The overall algorithm

A limitation of the above procedures is that they require knowing T. In most applications, the matrix T would be unknown and to be estimated. We present here an extension of the recent noise estimator of [21, 26] to the multi-class settings. It is derived under two assumptions.

Theorem 3 *Assume $p(\boldsymbol{x}, \boldsymbol{y})$ is such that:*

(1) There exist "perfect examples" of each of class $j \in [c]$, in the sense that

$$(\exists \bar{\boldsymbol{x}}^j \in \mathcal{X}) : \ \ p(\bar{\boldsymbol{x}}^j) > 0 \wedge p(\boldsymbol{y} = \boldsymbol{e}^j|\bar{\boldsymbol{x}}^j) = 1.$$

(2) given sufficiently many corrupted samples, $\boldsymbol{h}$ is rich enough to model $p(\tilde{\boldsymbol{y}}|\boldsymbol{x})$ accurately.

It follows that $\forall i, j \in [c], \ \ T_{ij} = p(\tilde{\boldsymbol{y}} = \boldsymbol{e}^j|\bar{\boldsymbol{x}}^i)$.

Proof. By (2), we can consider $p(\tilde{\boldsymbol{y}}|\boldsymbol{x})$ instead of $\hat{p}(\tilde{\boldsymbol{y}}|\boldsymbol{x})$. For any $j \in [c]$ and any $\boldsymbol{x} \in \mathcal{X}$, we have that:

$$p(\tilde{\boldsymbol{y}} = \boldsymbol{e}^j|\boldsymbol{x}) = \sum\nolimits_{k=1}^{c} p(\tilde{\boldsymbol{y}} = \boldsymbol{e}^j|\boldsymbol{y} = \boldsymbol{e}^k)\, p(\boldsymbol{y} = \boldsymbol{e}^k|\boldsymbol{x})$$

$$= \sum\nolimits_{k=1}^{c} T_{kj}\, p(\boldsymbol{y} = \boldsymbol{e}^k|\boldsymbol{x}) \ . \tag{11}$$

By (1), when $\boldsymbol{x} = \bar{\boldsymbol{x}}^i$, $p(\boldsymbol{y} = \boldsymbol{e}^k|\bar{\boldsymbol{x}}^i) = 0$ for $k \neq i$. ■

Rather surprisingly, Theorem 3 tells us that we can estimate each component of matrix T *just based on noisy class probability estimates*, that is, the output of the softmax of a

Algorithm 1 Robust two-stage training

Input: the noisy training set $\mathcal{S}$, any loss ℓ
If T is unknown:
  Train a network $\boldsymbol{h}(\boldsymbol{x})$ on $\mathcal{S}$ with loss ℓ
  Obtain an unlabeled sample X'
  Estimate $\hat{T}$ by Equations (12)-(13) on X'
Train the network $\boldsymbol{h}(\boldsymbol{x})$ on $\mathcal{S}$ with loss $\ell^{\leftarrow}$ or $\ell^{\rightarrow}$
Output: $\boldsymbol{h}(\cdot)$

network trained with noisy labels. In particular, let X' be any set of features vectors. This can be the training set itself, but not necessarily: we do not require this sample to have *any* label at all and therefore any unlabeled sample from the same distributions can be used as well. We can approximate T with two steps:

$$\bar{\boldsymbol{x}}^i = \text{argmax}_{\boldsymbol{x}\in X'}\ \hat{p}(\tilde{\boldsymbol{y}} = \boldsymbol{e}^i|\boldsymbol{x}) \tag{12}$$

$$\hat{T}_{ij} = \hat{p}(\tilde{\boldsymbol{y}} = \boldsymbol{e}^j|\bar{\boldsymbol{x}}^i)\ . \tag{13}$$

In practice, assumption (1) of Theorem 3 might hold true when X' is large enough. Assumption (2) of Theorem 3 is more difficult to justify; we require that the network can perfectly model the probability of *the noisy labels*. Although, in the experiments we can often recover T close to the ground truth and find that small estimation errors have a mild, not catastrophic effect on the quality of the correction.

Algorithm 1 summarizes the end-to-end approach. If we know T, for example by cleaning manually a subset of training data, we can train with $\ell^{\leftarrow}$ or $\ell^{\rightarrow}$. Otherwise, we first have to train the network with ℓ on noisy data, and obtain from it estimates of $p(\tilde{\boldsymbol{y}}|\boldsymbol{x})$ for each class via the output of the softmax. After training $\hat{T}$ is computable in $O(c^2 \cdot |X'|)$. Finally, we re-train with the corrected loss, while potentially utilizing the first network to help initializing the second one.

4.4. Digression: noise free Hessians via ReLU

We now present a result of independent interest in the context of label noise. The ReLU activation function appears to be a good fit for an architecture in our noise model, since it brings the particular convenience that the Hessian of the loss *does not depend on noise*, and hence the local curvature is left unchanged. At the same time, we are assured that backward correction by T — or any arbitrarily bad estimator of the matrix — has no impact on those second order properties of the loss — something that does not hold for the forward correction though. We stress the fact that other activation functions like the sigmoid do not share this guarantee. The proof makes use of the factorization trick due to [30].

Theorem 4 *Assume that all activation functions are ReLUs*[4]*. Then, the Hessian of ℓ does not change under noise. Moreover, the Hessians of $\ell^{\leftarrow}$ and ℓ are the same for any T.*

[4]A caveat: ℓ must be a linear-odd loss studied in [30]; cross-entropy and square loss are such. At the same time, we could generalize Theorem 4 to any neural network that expresses a piece-wise linear function, including for example max-pooling.

loss	correction	$\mathbb{E}_{\boldsymbol{x},\tilde{\boldsymbol{y}}}$	Hessian of $\mathbb{E}_{\boldsymbol{x},\tilde{\boldsymbol{y}}}$
ℓ	-	no guarantee	unchanged
$\ell^{\leftarrow}$	$T^{-1}\cdot$	unbiased estimator of ℓ	unchanged
$\ell^{\rightarrow}$	$T\cdot$	same minimizer of ℓ	no guarantee

Table 1: Qualitative comparison of loss corrections.

Proof. We give the proof for cross-entropy for simplicity; see [30] for a generalization. When $\boldsymbol{y} = \boldsymbol{e}^i$ the loss is:

$$-\log \hat{p}(\boldsymbol{y} = \boldsymbol{e}^i|\boldsymbol{x})_i = -\log \frac{e^{W^{(n)}_{i\cdot}\boldsymbol{x}^{(n-1)} + \boldsymbol{b}^{(n)}_i}}{\sum_{k=1}^c e^{W^{(n)}_{k\cdot}\boldsymbol{x}^{(n-1)} + \boldsymbol{b}^{(n)}_k}}$$
$$= -W^{(n)}_{i\cdot}\boldsymbol{x}^{(n-1)} + \boldsymbol{b}^{(n)}_i + \log \sum_{k=1}^c e^{W^{(n)}_{k\cdot}\boldsymbol{x}^{(n-1)} + \boldsymbol{b}^{(n)}_k}\ .$$

The only dependence on the true class $\boldsymbol{e}^i$ above are the first two terms. The log-partition is *independent* of the precise class i. Evidently, the noise affects the loss only through $W^{(n)}_{\cdot i}$ and $\boldsymbol{b}^{(n)}_i$: those are the *only* terms in which $\ell(\boldsymbol{y}, \hat{p}(\boldsymbol{y}|\boldsymbol{x}))$ and $\ell(\tilde{\boldsymbol{y}}, \hat{p}(\boldsymbol{y}|\boldsymbol{x}))$ may differ. Therefore we can rewrite the backward corrected loss as:

$$\ell^{\leftarrow}(\boldsymbol{e}^j, \hat{p}(\boldsymbol{y}|\boldsymbol{x})) = \left(T^{-1}\boldsymbol{\ell}(\hat{p}(\boldsymbol{y}|\boldsymbol{x}))\right)_j \tag{14}$$

$$= -\left(T^{-1}W^{(n)}\right)_{j\cdot}\boldsymbol{x}^{(n-1)} - \left(T^{-1}\boldsymbol{b}^{(n)}\right)_j \tag{15}$$

$$+ \log \sum_{k=1}^c e^{W^{(n)}_{k\cdot}\boldsymbol{x}^{(n-1)} + \boldsymbol{b}^{(n)}_k}\ . \tag{16}$$

In fact, note that T^{-1} does not affect the log-partition function. To see this, let $A(\boldsymbol{x}) = \log\left(\sum_{k=1}^c e^{W^{(n)}_{k\cdot}\boldsymbol{x}^{(n-1)} + \boldsymbol{b}^{(n)}_k}\right)$, with the (vector) log-partition being $A(\boldsymbol{x})\mathbf{1}$. It follows that its correction is $T^{-1}A(\boldsymbol{x})\mathbf{1} = A(\boldsymbol{x})\mathbf{1}$, by left-multiplication of T and because $T\mathbf{1} = \mathbf{1}$ since T is row-stochastic. Thus $\ell^{\leftarrow}(\boldsymbol{e}^j, \boldsymbol{h}_s(\boldsymbol{x})) = B(\boldsymbol{x}) + A(\boldsymbol{x})$, where $B(\boldsymbol{x}) = -(T^{-1}W^{(n)})_{j\cdot}\boldsymbol{x}^{(n-1)} - (T^{-1}\boldsymbol{b}^{(n)})_j$ is a *piece-wise linear* function of the *model parameters*, and the log-partition $A(\boldsymbol{x})$ is non-linear because of the loss and the architecture but *does not depend on noise*. Since the composition of piece-wise linear function is piece-wise linear, the Hessian of $B(\boldsymbol{x})$ vanishes, and therefore the Hessian of $\ell^{\leftarrow}$ is noise independent for any T. The same holds for ℓ (no correction) by taking $T = I$ and hence the Hessians are the same. ■

Theorem 4 does not provide any assurance on minima: indeed, stationary points may change location due to label noise. What it *does* guarantee is that the convergence rate of first-order methods is the same: the loss curvature cannot blow up or flat out and instead it is the same point by point in the model space. The Theorem advocates for use of ReLU networks, in line with the recent theoretical breakthrough allowing for deep learning with no local minima [16]. Table 1 summaries the properties of loss correction.

5. Experiments

We now test the theory on various deep neural networks trained on MNIST [20], IMDB [23], CIFAR-10, CIFAR-100 [18] and Clothing1M [42] so as to stress that our approach is independent on both architecture and data domain.

5.1. Loss corrections with T known or estimated

We artificially corrupt labels by a parametric matrix T. The rationale is to mimic some of the structure of real mistakes for similar classes, *e.g.* CAT $\rightarrow$ DOG. Transitions are parameterized by $N \in [0, 1]$ such that ground truth and wrong class have probability respectively of $1 - N, N$. An example of T used for MNIST with $N = 0.7$ is on the left:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & .3 & 0 & 0 & 0 & 0 & .7 & 0 & 0 \\ 0 & 0 & 0 & .3 & 0 & 0 & 0 & 0 & .7 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & .3 & .7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & .7 & .3 & 0 & 0 & 0 \\ 0 & .7 & 0 & 0 & 0 & 0 & 0 & .3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & 1 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & .33 & \epsilon & \epsilon & \epsilon & \epsilon & .67 & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & .35 & \epsilon & \epsilon & \epsilon & \epsilon & .65 & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & 1 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & .29 & .71 & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & .73 & .26 & \epsilon & \epsilon & \epsilon \\ \epsilon & .75 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & .25 & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & 1 & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & 1 \end{bmatrix} \quad (17)$$

Common to all experiments is what follows. The loss ℓ chosen for comparison is cross-entropy. 10% of training data is held out for validation. *The loss* is evaluated on it during training. With the corrected losses we can validate *on noisy data*, which is advantageous over other approaches that measure noisy validation accuracy instead. The available standard test sets are used for testing. We use ReLU for all networks and initialize weights prior to ReLUs as in [10], otherwise by uniform sampling in $[-0.05, 0.05]$. The mini-batch size is 128. The estimator of T from noisy labels is applied to X' being training and validation sets together. In fact, preliminary experiments highlighted that the large size X' improve sensibly the approximation of T; after estimation, we row-normalize the matrix. Following [26], we take a α-percentile in place of the argmax of Equation 12, and we found $\alpha = 97\%$ to work well for most experiments; the estimator performs very poorly with CIFAR-100, possibly due the small number of images per class, and we found it is better off computing the argmax instead.

Fully connected network on MNIST. In the first set of experiments we consider MNIST. Pixels are normalized in $[0, 1]$. Noise flips some of the similar digits: $2 \rightarrow 7, 3 \rightarrow 8, 5 \leftrightarrow 6, 7 \rightarrow 1$; see Equation (17, left). We train an architecture with two dense hidden layers of size 128, with probability 0.5 of dropout. AdaGrad [4] is run for 40 epochs with initial learning rate 0.01 and $\delta = 10^{-6}$. We repeat each experiment 5 times to account for noise and weight initialization. It is clear from Figure 1a that, although the model is somewhat robust to mild noise, high level of corruption has a disrupting effect on ℓ. Instead, our losses do not witness a drastic drop. With $\hat{T}$ estimated performance lays in between, yet it is significantly better than with no correction. An example of $\hat{T}$ is in Equation (17, right), with $\epsilon < 10^{-6}$.

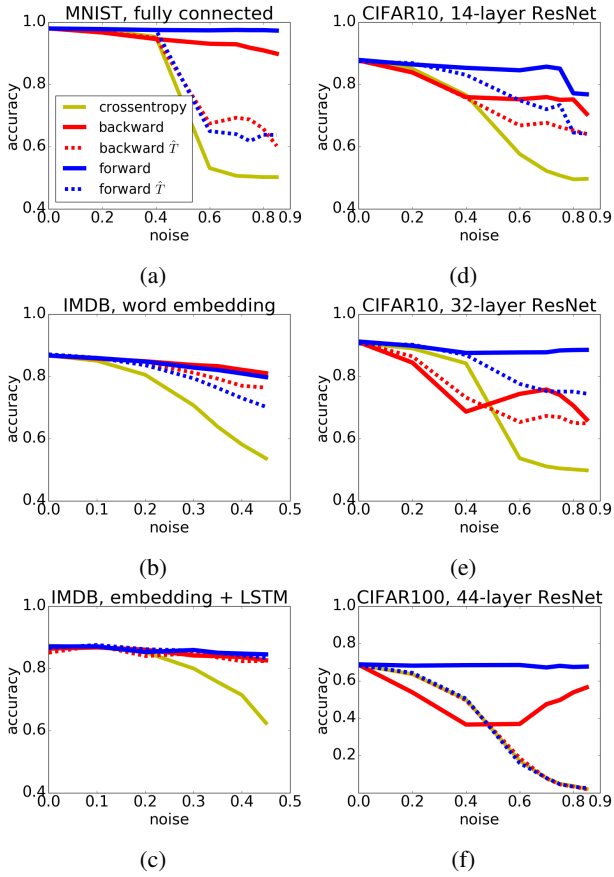

Figure 1: Comparison of cross-entropy with its corrections, with known or estimated T.

Word embedding and LSTM on IMDB. We keep only the top 5000 most frequent words in the corpus. Each review is either truncated or padded to be 400-word long. To simulate asymmetric noise in this binary problem, we keep constant noise for the transition $0 \rightarrow 1$ at 5%, while $1 \rightarrow 0$ is parameterized as above; $0/1$ are the two review's sentiments. We trained two models inspired by the baselines of [2]. The first maps words into 50-dimensional embeddings, before passing through ReLUs; dropout with probability 0.8 is applied to the embedding output. In the second model the embedding has dimension 256 and it is followed by an LSTM with 512 units and by a last 512-dimensional hidden layer with 0.5 dropout. AdaGrad is run for 50 epochs with the same setup as above; results are averages over 5 runs. Figures 1b-1c display an outcome similar to what previously observed on MNIST, in spite of difference in dataset, number of classes, architecture and structure of T. Noticeably, our approach is effective on recurrent networks as well. Correcting with $\hat{T}$ is in line with the true T here; we believe this is because estimation is easier on this binary problem.

Residual networks on CIFAR-10 and CIFAR-100. For both datasets we perform per-pixel mean subtraction and

data augmentation as in [11], by horizontal random flips and 32×32 random crops after padding with 4 pixels on each side. T for CIFAR-10 is described by: TRUCK $\rightarrow$ AUTOMOBILE, BIRD $\rightarrow$ AIRPLANE, DEER $\rightarrow$ HORSE, CAT $\leftrightarrow$ DOG. In CIFAR-100, the 100 classes are grouped into 20 5-size super-classes, *e.g.* AQUATIC mammals contain BEAVER, DOLPHIN, OTTER, SEAL and WHALE. Within super-classes, the noise flips each class into the next, circularly.

For the last experiments we use deep residual networks (ResNet), the CIFAR-10/100 architectures from [11]. In short, residual blocks implements a non-linear operation $F(\boldsymbol{x})$ in parallel with an identity shortcut: $\boldsymbol{x} \rightarrow \boldsymbol{x}+F(\boldsymbol{x})$. F is as cascade of twice batch normalization $\rightarrow$ ReLU $\rightarrow 3\times3$ convolution, following the "pre-activation" recommendation of [12]. Here we experiment with ResNets of depth 14 and 32 (CIFAR-10) and 44 (CIFAR-100). By common practice [14], we run SGD with 0.9 momentum and learning rate 0.01, and divide it by 10 after 40 and 80 epoch (120 in total) for CIFAR-10 and after 80 and 120 (150) for CIFAR-100; weight decay is 10^{-4}. Training deep ResNets is more time consuming and thus experiments are run only once. Since we use shallower networks than the ones in [11], performance is not comparable with the original work. In figures 1d-1f, forward correction does not suffer any significant loss. Except with the shallowest ResNet, backward correction does not seem to work well in the low noise regime. Finally, noise estimation is particularly difficult on CIFAR-100.

5.2. Comparing with other loss functions

We now compare with other methods. Data, architectures and artificial noise are the same as above. Additionally, we test the case of symmetric noise where N is the probability of label flip that is spread uniformly among all the other classes. We select methods prescribing changes in the loss function, similarly to ours: unhinged [41], sigmoid [8], Savage [24] and soft and hard bootstrapping [32]; hyper-parameters of the last two methods are set in accordance with their paper.

Unhinged loss is unbounded and cannot be used alone. In the original work L_2 regularization is applied to address the problem, when training non-parametric kernel models. We tried to regularize every layer with little success; learning either does not converge (too little regularization) or converge to very poor solutions (too much). On preliminary experiments sigmoid loss ran into the opposite issue, namely premature saturation; the loss reaches a plateau too quickly, a well-known problem with sigmoidal activation functions [9]. To make those losses usable for comparison, we stack a layer of batch normalization right before the loss function. Essentially, the network outputs are whitened and likely to operate in a bounded, non-saturated area of the loss; note that this is never required for linear or kernel models.

Table 2 presents the empirical analysis. We list the key findings: (a) In the absence of artificial noise (first column for each dataset), all losses reach similar accuracies with a spread of 2 points; exceptions are some instances of unhinged, sigmoid and Savage. Additionally, with IMDB there are cases († in Table 2) of loss correction with noise estimation that perform slightly better than assuming no noise. Clearly, the estimator is able to recover the *natural* noise in the sentiment reviews. (b) With low asymmetric noise (second column) results differ between simple architecture/tasks (datasets on the left) and deep networks/more difficult problems (right); in the former case, the two corrections behave similarly and are not statistically far from the competitors; in the latter case, forward correction with known T is unbeaten, with no clear winner among the remaining ones. (c) With asymmetric noise (last two columns) the two loss corrections with known T are overall the best performing, confirming the practical implications of their formal guarantees; forward is usually the best. (d) If we exclude CIFAR-100, the noise estimation accounts for average accuracy drops between 0 (IMBD with LSTM model) and 27 points (MNIST); nevertheless, our performance is better than every other method in many occasions. (e) In the experiment on CIFAR-100 we obtain essentially perfect noise robustness with the ideal forward correction. The noise estimation works well except in the very last column, yet it guarantees again better accuracy over competing methods. We discuss this issue in Section 6.

5.3. Experiments on Clothing1M

Finally, we test on Clothing1M [42], consisting of 1M images with noisy labels, with additional $50k, 14k, 10k$ of clean data respectively for training, validation and testing; we refer to those sets by their size. We aim to classify images within 14 classes, *e.g.* t-shirt, suit, vest. In the original work two AlexNets [19] are trained together via EM; the networks are pre-trained with ImageNet. Two practical tricks are fundamental: a first learning phase with the clean $50k$ to help EM ($\#1$ in Table 3) and a second phase with the mix of $50k$ bootstrapped to $500k$ and $1M$ ($\#3$). Data augmentation is also applied, same as in Section 5.1 for CIFAR-10.

We learn a 50-layer ResNet pre-trained on ImageNet — the bottleneck architecture of [11] — with SGD with learning rate 10^{-3} and 10^{-4} for 5 epochs each, 0.9 momentum, and batch size 32. When we train with $50k$ we use weight decay of $5\cdot10^{-2}$ and data augmentation, while with $1M$ we use only weight decay of 10^{-3}. The ResNet gives an uplift of about 2.5% by training with $50k$ only ($\#7$ *vs.* $\#1$). However, the large amount of noisy images is essential to compete with $\#3$. Instead of estimating the matrix T by (12)-(13), we exploit the curated labels of $50k$ and their noisy versions in $1M$. Forward and backward corrections are confirmed to work better than cross-entropy ($\#6, \#5$ *vs.* $\#4$), yet cannot reach the state of the art without the additional clean data. Thus, we fine tune the networks with $50k$, with the same learning parameters as in $\#7$; due to

	MNIST, fully connected				CIFAR-10, 14-layer ResNet			
	NO NOISE	SYMM. $N = 0.2$	ASYMM. $N = 0.2$	ASYMM. $N = 0.6$	NO NOISE	SYMM. $N = 0.2$	ASYMM. $N = 0.2$	ASYMM. $N = 0.6$
cross-entropy	97.9 ± 0.0	$\mathbf{96.9 \pm 0.1}$	97.5 ± 0.0	53.0 ± 0.6	87.8	83.7	85.0	57.6
unhinged (BN)	97.6 ± 0.0	$\mathbf{96.9 \pm 0.1}$	97.0 ± 0.1	$\mathbf{71.2 \pm 1.0}$	86.9	**84.1**	83.8	52.1
sigmoid (BN)	97.2 ± 0.1	93.1 ± 0.2	96.7 ± 0.1	$\mathbf{71.4 \pm 1.3}$	76.0	66.6	71.8	57.0
Savage	97.3 ± 0.0	$\mathbf{96.9 \pm 0.0}$	97.0 ± 0.1	51.3 ± 0.4	80.1	77.4	76.0	50.5
bootstrap soft	97.9 ± 0.0	$\mathbf{96.9 \pm 0.0}$	97.5 ± 0.0	53.0 ± 0.4	87.7	**84.3**	84.6	57.8
bootstrap hard	97.9 ± 0.0	96.8 ± 0.0	97.4 ± 0.0	55.0 ± 1.3	87.3	83.6	84.7	58.3
backward	97.9 ± 0.0	96.3 ± 0.1	96.6 ± 1.1	$\mathbf{93.0 \pm 0.9}^\star$	87.6	81.5	83.8	$\mathbf{75.2}^\star$
backward $\hat{T}$	97.9 ± 0.0	$\mathbf{96.9 \pm 0.0}$	96.7 ± 0.1	67.4 ± 1.5	87.7	80.4	83.8	66.7
forward	97.9 ± 0.0	$\mathbf{97.3 \pm 0.0}^\star$	$\mathbf{97.7 \pm 0.0}$	$\mathbf{97.3 \pm 0.0}^\star$	87.8	$\mathbf{85.6}^\star$	86.3	$\mathbf{84.5}^\star$
forward $\hat{T}$	97.9 ± 0.0	$\mathbf{96.9 \pm 0.0}$	$\mathbf{97.7 \pm 0.0}$	64.9 ± 4.4	87.4	83.4	**87.0**	**74.8**
	IMBD, word embedding				CIFAR-10, 32-layer ResNet			
	NO NOISE	SYMM. $N = 0.1$	ASYMM. $N = 0.1$	ASYMM. $N = 0.4$	NO NOISE	SYMM. $N = 0.2$	ASYMM. $N = 0.2$	ASYMM. $N = 0.6$
cross-entropy	86.7 ± 0.0	84.6 ± 0.1	85.0 ± 0.2	58.1 ± 0.5	90.1	86.6	89.0	53.6
unhinged (BN)	83.3 ± 0.0	76.9 ± 0.5	80.6 ± 0.3	72.9 ± 0.4	90.2	86.5	87.1	60.0
sigmoid (BN)	84.3 ± 0.0	80.2 ± 0.3	81.7 ± 0.5	72.8 ± 0.6	81.6	69.6	79.1	61.8
Savage	86.5 ± 0.0	84.3 ± 0.4	85.2 ± 0.3	58.3 ± 1.0	88.3	86.2	86.3	53.5
bootstrap soft	86.7 ± 0.0	84.5 ± 0.1	85.1 ± 0.1	57.8 ± 0.7	90.9	86.9	88.6	53.1
bootstrap hard	86.7 ± 0.0	84.6 ± 0.3	85.1 ± 0.3	59.0 ± 0.6	90.4	86.4	88.6	54.7
backward	86.7 ± 0.0	$\mathbf{85.3 \pm 0.3}^\star$	$\mathbf{85.7 \pm 0.1}$	$\mathbf{82.1 \pm 0.1}^\star$	90.1	83.0	84.4	74.3
backward $\hat{T}$	$87.0 \pm 0.0^\dagger$	$\mathbf{85.1 \pm 0.4}$	$\mathbf{85.8 \pm 0.2}$	$\mathbf{77.0 \pm 1.4}$	90.8	86.9	86.4	66.7
forward	86.7 ± 0.0	$\mathbf{85.3 \pm 0.2}^\star$	$\mathbf{85.9 \pm 0.1}$	$\mathbf{80.9 \pm 1.3}^\star$	91.2	**87.7**	**89.9**	$\mathbf{87.6}^\star$
forward $\hat{T}$	$87.0 \pm 0.0^\dagger$	$\mathbf{85.2 \pm 0.3}$	$\mathbf{85.9 \pm 0.2}$	73.0 ± 1.2	90.5	**87.9**	**90.1**	**77.6**
	IMBD, word embedding + LSTM				CIFAR-100, 44-layer ResNet			
	NO NOISE	SYMM. $N = 0.1$	ASYMM. $N = 0.1$	ASYMM. $N = 0.4$	NO NOISE	SYMM. $N = 0.2$	ASYMM. $N = 0.2$	ASYMM. $N = 0.6$
cross-entropy	87.8 ± 0.4	$\mathbf{85.2 \pm 0.5}$	86.8 ± 0.4	71.4 ± 1.3	68.5	57.9	63.5	17.1
unhinged (BN)	84.3 ± 4.4	69.7 ± 15.9	85.2 ± 1.2	59.4 ± 12.9	50.9	47.5	48.0	14.5
sigmoid (BN)	87.7 ± 0.5	77.6 ± 13.6	$\mathbf{86.3 \pm 3.1}$	$\mathbf{70.0 \pm 14.6}$	58.2	47.6	55.6	16.4
Savage	87.4 ± 0.3	$\mathbf{85.1 \pm 0.6}$	$\mathbf{87.2 \pm 0.3}$	70.4 ± 3.8	1.4	2.0	1.8	1.6
bootstrap soft	87.1 ± 0.6	$\mathbf{83.5 \pm 2.5}$	$\mathbf{86.1 \pm 1.2}$	69.0 ± 5.3	67.9	57.8	**63.8**	16.3
bootstrap hard	86.5 ± 0.5	$\mathbf{84.3 \pm 1.0}$	86.7 ± 0.4	71.8 ± 3.3	68.5	57.3	**63.9**	17.0
backward	87.6 ± 0.2	$\mathbf{84.3 \pm 0.9}$	86.7 ± 0.5	$\mathbf{83.6 \pm 1.4}$	68.5	55.1	53.8	$\mathbf{36.8}^\star$
backward $\hat{T}$	87.2 ± 0.7	$\mathbf{82.8 \pm 2.7}$	$\mathbf{87.3 \pm 0.1}$	$\mathbf{82.3 \pm 1.7}$	68.6	51.7	63.8	**18.5**
forward	87.5 ± 0.2	$\mathbf{85.0 \pm 0.2}$	87.0 ± 0.4	$\mathbf{84.7 \pm 0.6}^\star$	68.8	$\mathbf{64.0}^\star$	$\mathbf{68.1}^\star$	$\mathbf{68.4}^\star$
forward $\hat{T}$	$87.8 \pm 1.5^\dagger$	$\mathbf{84.1 \pm 1.0}$	$\mathbf{87.5 \pm 0.2}$	$\mathbf{84.2 \pm 0.9}$	68.1	**58.6**	**64.2**	15.9

Table 2: Average accuracy with standard deviation (5 runs, left part) is bold faced when statistically far from the others, by means of passing a *Welch's t-test* with p-value $< 5\%$; in case the highest accuracy is due to $\ell^{\leftarrow}$ or $\ell^{\rightarrow}$ with the ground truth T, we denote those by * and highlight the next highest accuracy as well. For experiments with no standard deviation (right part), the same rule is applied, but bold face is given to the all accuracies in a range of 0.5 points from the highest. The meaning of N depends on symmetric *vs.* asymmetric noise and on number of classes (see Section 5.1). On the first columns with no injected noise, † indicates when the noise estimation recovers some *natural* noise and beats "loss correction" with $T = I$.

Clothing1M					
#	model	loss	init	training	accuracy
1	AlexNet	cross-.	ImageNet	$50k$	72.63
2	AlexNet [39]	cross-.	#1	$1M, 50k$	76.22
3	AlexNet [42]	cross-.	#1	$1M, 50k$	78.24
4	50-ResNet	cross-	ImageNet	$1M$	68.94
5	50-ResNet	backward	ImageNet	$1M$	69.13
6	50-ResNet	forward	ImageNet	$1M$	69.84
7	50-ResNet	cross-.	ImageNet	$50k$	75.19
8	50-ResNet	cross-.	#6	$50k$	**80.38**

Table 3: Results on the top section are from [42]. In $\#2, \#3$ the clean $50k$ are bootstrapped to $500k$. Best result $\#8$ is obtained by fine tuning a net trained with forward correction.

reasons of time we only tune $\#6$. The new state of the art is $\#8$ that outperforms [42] of more than 2 percent, which is achieved without time consuming bootstrapping of the $50k$.

6. Discussion and Conclusion

We have proposed a framework for training deep neural networks with noisy labels that boils down to two loss corrections. Accuracy is consistently only few percent points away from training cross-entropy *on clean data*, while corruption can worsen performance of cross-entropy by 40 percent or more. Forward correction often performs better. We believe the reason is not statistical — Theorems 1 and 2 guarantee optimality, in the limit of infinite data. The cause may be either numerical (via matrix inversion) or a drastic change of the loss (in particular its Hessian), which may have a detrimental effect on optimization. Indeed, backward correction is a linear combination of losses for every possible label, with coefficients that can be far by orders of magnitude and thus makes the learning harder. Instead, forward correction projects predictions into a probability distribution in $[0, 1]$.

The quality of noise estimation is a key factor for obtaining robustness. In practice, it works well in most experiments with a median drop of only 10 points of accuracy with respect to using the true T. The exception is the last column for CIFAR-100, where estimation destroys most of the gain from loss correction. We believe that the mix of high noise and limited number of images per class (500) is detrimental to the estimator. This is confirmed by the sensitivity of α.

Future work shall improve the estimation phase by incorporating priors of the noise structure, for example assuming low rank T. Improvements on this direction may also widen the applicability to massively multi-class scenarios. It remains an open question whether *instance*-dependent noise may be included into our approach [42, 25]. Finally, we anticipate the use of our approach as a tool for pre-training models with noisy data from the Web, in the spirit of [17].

Acknowledgments. *We gratefully acknowledge NVIDIA Corporation for the donation of a Tesla K40 GPU. We also thank the developers of Keras [1].*

References

[1] F. Chollet. Keras. `github.com/fchollet/keras`.

[2] A. M. Dai and Q. V. Le. Semi-supervised sequence learning. In *NIPS*29*, 2015.

[3] S. Divvala, A. Farhadi, and C. Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *27th IEEE CVPR*, 2014.

[4] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, 2011.

[5] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from internet image searches. *Proceedings of the IEEE*, 98(8):1453–1466, 2010.

[6] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. In *NIPS*29*, 2015.

[7] 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, May 2014.

[8] A. Ghosh, N. Manwani, and P. S. Sastry. Making risk minimization tolerant to label noise. *Neurocomputing*, 2015.

[9] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

[11] K. He, X. Zhang., S. Ren, and J. Sun. Deep residual learning for image recognition. In *29th IEEE CVPR*, 2016.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *14 th ECCV*, 2016.

[13] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[14] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Weinberger. Deep networks with stochastic depth. In *14 th ECCV*, 2016.

[15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *32 th ICML*, 2015.

[16] K. Kawaguchi. Deep learning without poor local minima. In *NIPS*30*, 2016.

[17] J. Krause, B. Sapp, A. Howard, H. Zhou, A. Toshev, T. Duerig, J. Philbin, and L. Fei-Fei. The unreasonable effectiveness of noisy data for fine-grained recognition. In *14 th ECCV*, 2016.

[18] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*26*, 2012.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[21] T. Liu and D. Tao. Classification with noisy labels by importance reweighting. *IEEE Transactions on PAMI*, 38(3):447–461, 2016.

[22] P. M. Long and R. A. Servedio. Random classification noise defeats all convex potential boosters. *Machine learning*, 78(3):287–304, 2010.

[23] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *49 th ACL*, 2011.

[24] H. Masnadi-Shirazi and N. Vasconcelos. On the design of loss functions for classification: theory, robustness to outliers, and savageboost. In *NIPS*23*, 2009.

[25] A. Menon, B. van Rooyen, and N. Natarajan. Learning from binary labels with instance-dependent corruption. *arXiv preprint arXiv:1605.00751*, 2016.

[26] A. Menon, B. van Rooyen, C. S. Ong, and B. Williamson. Learning from corrupted binary labels via class-probability estimation. In *32 th ICML*, 2015.

[27] V. Mnih and G. E. Hinton. Learning to label aerial images from noisy data. In *29 th ICML*, 2012.

[28] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari. Learning with noisy labels. In *NIPS*27*, 2013.

[29] L. Niu, W. Li, and D. Xu. Visual recognition by learning from web data: A weakly supervised domain generalization approach. In *28th IEEE CVPR*, 2015.

[30] G. Patrini, F. Nielsen, R. Nock, and M. Carioni. Loss factorization, weakly supervised learning and label noise robustness. In *33 th ICML*, 2016.

[31] H. G. Ramaswamy, C. Scott, and A. Tewari. Mixture proportion estimation via kernel embedding of distributions. In *33 th ICML*, 2016.

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

[33] M. D. Reid and R. C. Williamson. Composite binary losses. *JMLR*, 11:2387–2422, 2010.

[34] T. Sanderson and C. C. Scott. Class proportion estimation with application to multiclass anomaly rejection. In *AISTATS*, 2014.

[35] F. Schroff, A. Criminisi, and A. Zisserman. Harvesting image databases from the web. *IEEE Transactions on PAMI*, 33(4):754–766, 2011.

[36] C. Scott, G. Blanchard, and G. Handy. Classification with asymmetric label noise : Consistency and maximal denoising. In *26 rd COLT*, 2013.

[37] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014.

[38] G. Stempfel and L. Ralaivola. Learning SVMs from sloppily labeled data. In *Artificial Neural Networks (ICANN)*, pages 884–893. Springer, 2009.

[39] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, and R. Fergus. Training convolutional networks with noisy labels. In *ICLR Workshops*, 2015.

[40] B. van Rooyen. *Machine Learning via Transitions*. PhD thesis, The Australian National University, 2015.

[41] B. van Rooyen, A. K. Menon, and R. C. Williamson. Learning with symmetric label noise: The importance of being unhinged. In *NIPS*29*, 2015.

[42] T. Xiao, T. Xia, T. Yang, C. Huang, and X. Wang. Learning from massive noisy labeled data for image classification. In *28th IEEE CVPR*, 2015.