Task Agnostic Robust Learning on Corrupt Outputs by Correlation-Guided Mixture Density Networks

Sungjoon Choi
Kakao Brain
sam.choi@kakaobrain.com

Sanghoon Hong
Kakao Brain
sanghoon.hong@kakaobrain.com

Kyungjae Lee
Seoul National University
kyungjae.lee@rllab.snu.ac.kr

Sungbin Lim*
UNIST
sungbin@unist.ac.kr

Abstract

In this paper, we focus on weakly supervised learning with noisy training data for both classification and regression problems. We assume that the training outputs are collected from a mixture of a target and correlated noise distributions. Our proposed method simultaneously estimates the target distribution and the quality of each data which is defined as the correlation between the target and data generating distributions. The cornerstone of the proposed method is a Cholesky Block that enables modeling dependencies among mixture distributions in a differentiable manner where we maintain the distribution over the network weights. We first provide illustrative examples in both regression and classification tasks to show the effectiveness of the proposed method. Then, the proposed method is extensively evaluated in a number of experiments where we show that it constantly shows comparable or superior performances compared to existing baseline methods in the handling of noisy data.

1. Introduction

Training a deep neural network requires immense amounts of training data which are often collected using crowd-sourcing methods, such as Amazon's Mechanical Turk (AMT). However, in practice, the crowd-sourced labels are often noisy [4]. Furthermore, naive training of deep neural networks is often vulnerable to over-fitting given the noisy training data in that they are capable of memorizing the entire dataset even with inconsistent labels, leading to a poor generalization performance [44]. We address this problem through the principled idea of *revealing the correlations* between the target distribution and the other (possibly noise) distributions by assuming that a training dataset is sampled from a mixture of a target distribution and other correlated distributions.

Throughout this paper, we aim to address the following two questions: 1) How can we define (or measure) the quality of training data in a principled manner? 2) In the presence of inconsistent outputs, how can we infer the target distribution in a scalable manner? Traditionally, noisy outputs are handled by modeling additive random distributions, often leading to robust loss functions [16] or estimating the structures of label corruptions in classifications tasks [22] (for more details, refer to Section 2).

To address the first question, we leverage the concept of a correlation. Precisely, we define and measure the quality of training data using the correlation between the target distribution and the data generating distribution. However, estimating the correct correlation requires access to the target distribution, whereas learning the correct target distribution requires knowing the correlation between the distributions to be known, making it a chicken-and-egg problem. To address the second question, we present a novel method that simultaneously estimates the target distribution as well as the correlation in a fully differentiable manner using stochastic gradient descent methods.

The cornerstone of the proposed method is a *Cholesky Block* in which we employ the Cholesky transform for sampling the weights of a neural network that enables us to model correlated outputs. Similar to Bayesian neural networks [6], we maintain the probability distributions over the weights, but we also leverage mixture distributions to handle the inconsistencies in a dataset. To the best of our knowledge, this is the first approach simultaneously to infer the target distribution and the output correlations using a neural network in an end-to-end manner. We will refer to this framework as *ChoiceNet*.

ChoiceNet is first applied to synthetic regression tasks and a real-world regression task where we demonstrate its robustness to extreme outliers and its ability to distinguish the target distribution and noise distributions. Subsequently, we move on to image classification tasks using a number of benchmark datasets where we show that it shows comparable or superior performances compared to existing baseline

*This work is done at Kakao Brain

methods in terms of robustness with regard to handling different types of noisy labels.

2. Related Work

Recently, robustness in deep learning has been actively studied [11] as deep neural networks are being applied to diverse tasks involving real-world applications such as autonomous driving [33] or medical diagnosis [15] where a simple malfunction can have catastrophic results [1].

Existing work for handing noisy training data can be categorized into four groups: small-loss tricks [17, 21, 30, 37], estimating label corruptions [2, 13, 19, 34, 38, 41], using robust loss functions [3, 32], and explicit and implicit regularization methods [14, 26, 27, 31, 36, 39, 40, 42, 45]. Our proposed method is mostly related to the robust loss function approach but cannot fully be categorized into this group in that we present a novel architecture, a mixture of correlated densities network block, for achieving robustness based on the correlation estimation.

First of all, the small-loss tricks selectively focus on training instances based on a certain criterion, such as having small cost values [17]. [30] proposed a meta-algorithm for tackling the noisy label problem by training two networks only when the predictions of the two networks disagree, where selecting a proper network from among the two networks can be done using an additional clean dataset. [37] reweighs the weight of each training instance using a small amount of clean validation data. MentorNet [21] concentrated on the training of an additional neural network, which assigns a weight to each instance of training data to supervise the training of a base network, termed StudentNet, to overcome the over-fitting of corrupt training data. Recent work [17] presented Co-teaching by maintaining two separate networks where each network is trained with small-loss instances selected from its peer network.

The second group of estimating label corruption information is mainly presented for classification tasks where training labels are assumed to be corrupt with a possibly unknown corruption matrix. An earlier study in [2] proposed an extra layer for the modeling of output noises. [22] extended the approach mentioned above by adding an additional noise adaptation layer with aggressive dropout regularization. A similar method was then proposed in [34] which initially estimated the label corruption matrix with a learned classifier and used the corruption matrix to fine-tune the classifier. Other researchers [13] presented a robust training method that mimics the EM algorithm to train a neural network, with the label noise modeled as an additional softmax layer, similar to earlier work [22]. A self-error-correcting network was also presented [29]. It switches the training labels based on the learned model at the beginning stages by assuming that the deep model is more accurate during the earlier stage of training.

Researchers have also focussed on using robust loss functions; [32] studied the problem of binary classification in the presence of random labels and presented a robust surrogate loss function for handling noisy labels. Existing loss functions for classification were studied [12], with the results showing that the mean absolute value of error is inherently robust to label noise. In other work [3], a robust loss function for deep regression tasks was proposed using Tukey's biweight function with the median absolute deviation of the residuals.

The last group focusses on using implicit or explicit regularization methods while training. Adding small label noises while training is known to be beneficial to training, as it can be regarded as an effective regularization method [14, 27]. Similar methods have been proposed to tackle noisy outputs. A bootstrapping method [36] which trains a neural network with a convex combination of the output of the current network and the noisy target was proposed. [42] proposed DisturbLabel, a simple method that randomly replaces a percentage of the labels with incorrect values for each iteration. Mixing both input and output data was also proposed [40, 45]. One study [45] considered the image recognition problem under label noise and the other [40] focused on a sound recognition problem. The temporal ensemble was proposed in [26] where an unsupervised loss term of fitting the output of an augmented input to the augmented target updated with an exponential moving average. [39] extends the temporal ensemble in [26] by introducing a consistency cost function that minimizes the distance between the weights of the student model and the teacher model. [31] presented a new regularization method based on virtual adversarial loss which measures the smoothness of conditional label distribution given input. Minimizing the virtual adversarial loss has a regularizing effect in that it makes the model smooth at each data point.

The foundation of the proposed method is a *Cholesky Block* where the output distribution is modeled using a mixture of correlated distributions. Modeling correlations of output training data has been actively studied in light of Gaussian processes [35]. MTGPP [7] that models the correlations of multiple tasks via Gaussian process regression was proposed in the context of a multi-task setting. [9] proposed a robust learning from demonstration method using a sparse constrained leverage optimization method which estimates the correlation between the training outputs and showed its robustness compared to several baselines.

3. Proposed Method

In this section, we present the motivation and the model architecture of the proposed method. The main ingredient is a *Cholesky Block* which can be built on top of any arbitrary base network. First, we illustrate the motivations of designing the *Cholesky Block* in Section 3.1. Section 3.2 introduces a Cholesky transform which enables correlated sampling procedures. Subsequently, we present the overall mechanism of the proposed method and its loss functions for regression and classification tasks in Section 3.3 followed by illustrative synthetic examples in Section 3.4.

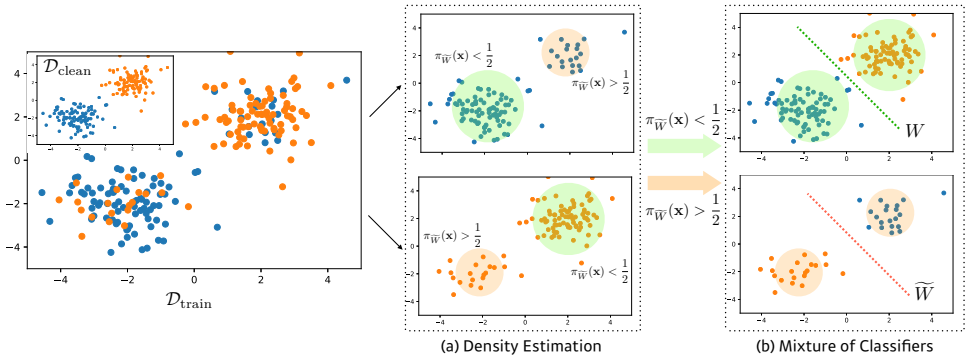


Figure 1: A process of binary classification on corrupt data using the mixture of (a) densities and (b) classifiers through (4). $\pi_{\widetilde{W}}(\mathbf{x})$ is the mixture weight which models the choice probability of $\widetilde{W}$ and estimates the corruption probability π_{corrupt} in (3).

3.1. Motivation

Denote training data with correct (clean) labels by $\mathcal{D}_{\text{clean}}$ whose elements $(\mathbf{x}, y) \in \mathcal{D}_{\text{clean}}$ are determined by a relation $y = f(\mathbf{x})$ for a regression task and $y \in L$ for a classification task where L is a discrete set. In this paper, we assume that the corrupt training data $(\mathbf{x}, \hat{y}) \in \mathcal{D}_{\text{train}}$ is generated by

Regression:

$$\hat{y} = \begin{cases} f(\mathbf{x}) + \epsilon & \text{with } 1-p \\ g(\mathbf{x}) + \xi & \text{with } p \end{cases} \tag{1}$$

Classification:

$$\hat{y} = \begin{cases} y & \text{with } 1-p \\ L \setminus \{y\} & \text{with } p \end{cases} \tag{2}$$

where g is an arbitrary function. Here ϵ and ξ are heteroscedastic additive noises and p indicates the corruption probability. Then, the above settings naturally employ the mixture of the conditional distributions:

$$P(\hat{y}|\mathbf{x}) = (1-\pi_{\text{corrupt}})P_{\text{target}}(\hat{y}|\mathbf{x}) + \pi_{\text{corrupt}}P_{\text{noise}}(\hat{y}|\mathbf{x}) \tag{3}$$

where π_{corrupt} models the corrupt ratio p. In particular, we model the target conditional density $P_{\text{target}}(\cdot|\cdot)$ using a parametrized distribution with a Gaussian distribution with a fixed variance $\hat{\sigma}^2$, i.e., $P(\cdot|\mathbf{x}) = \mathcal{N}(f_\theta(\mathbf{x}), \hat{\sigma}^2)$ where $f_\theta(\cdot)$ can be any functions, e.g., a feed-forward network, parametrized with θ.

While it may look similar to a Huber's ϵ-contamination model [20], one major difference is that, our method quantifies the irrelevance (or independence) of noise distributions by utilizing the input-dependent correlation $\rho(\mathbf{x})$ between $P_{\text{target}}(\cdot|\mathbf{x})$ and $P_{\text{noise}}(\cdot|\mathbf{x})$. To be specific, $P_{\text{noise}}(\cdot|\mathbf{x})$ will be a function of $P_{\text{target}}(\cdot|\mathbf{x})$ and $\rho(\mathbf{x})$.

In the training phase, we jointly optimize $P_{\text{target}}(\cdot|\mathbf{x})$ and $\rho(\mathbf{x})$. Intuitively speaking, irrelevant noisy data will be modeled to be collected from a class of P_{noise} with relatively small or negative ρ which can be shown in Figure 4. Since we assume that the correlation (quality) is not explicitly given, we model the ρ of each data to be a function of an input $\mathbf{x}$ i.e., $\rho_\phi(\mathbf{x})$, parametrized by ϕ and jointly optimize ϕ and θ.

Why correlation matters in mixture modeling? Let us suppose a binary classification problem $L = \{-1, 1\}$ and a feature map $h : \mathcal{X} \to \mathcal{F}$ and a (stochastic) linear functional $W : \mathcal{F} \to \mathbb{R}$ are given. For $(\mathbf{x}, y) \in \mathcal{D}_{\text{clean}}$, we expect that W and h will be optimized as follows:

$$Wh(\mathbf{x}) \sim \begin{cases} \mathcal{N}(\mu_+, \sigma^2) & : y = 1 \\ \mathcal{N}(\mu_-, \sigma^2) & : y = -1 \end{cases}, \quad \mu_+ > 0, \mu_- < 0$$

so that $-Wh(\mathbf{x}) \cdot y < 0$ holds. However, if corrupt training data are given by (2), the linear functional W and the feature map $h(\cdot)$ may have $-Wh(\mathbf{x}) \cdot \hat{y} > 0$ and using an ordinary loss function, such as a cross-entropy loss, might lead to over-fitting of the contaminated pattern of $\mathcal{D}_{\text{train}}$. Motivated from (3), we employ the mixture density to discriminate the corrupt data by using another linear classifier $\widetilde{W}$ which is expected to reveal the reverse patterns by minimizing the following mixture classification loss:

$$-\left\{(1-\pi_{\widetilde{W}}(\mathbf{x}))Wh(\mathbf{x}) - \pi_{\widetilde{W}}(\mathbf{x})\widetilde{W}h(\mathbf{x})\right\} \cdot \hat{y} \tag{4}$$

Here $\pi_{\widetilde{W}}(\mathbf{x})$ is the mixture weight which models the choice probability of the reverse classifier $\widetilde{W}$ and eventually estimates the corruption probability π_{corrupt} in (3). See Figure 1 for the illustrative example in binary classification.

However, the above setting is not likely to work in practice as both W and $\widetilde{W}$ may learn the corrupt patterns independently hence $\pi_{\widetilde{W}}(\mathbf{x})$ adhere to $1/2$ under (4). In other words, the lack of dependencies between W and $\widetilde{W}$ makes it hard to distinguish clean and corrupt patterns. To aid W to learn the pattern of data with a clean label, we need a self-regularizing way to help $\pi_{\widetilde{W}}$ to infer the corruption probability of given data point $\mathbf{x}$ by guiding $\widetilde{W}$ to learn

the reverse mapping of given feature $h(\mathbf{x})$. To resolve this problem, let us consider different linear functional $\widehat{W}$ with negative correlation with W, i.e., $\rho(\widehat{W}, W) < 0$. Then this functional maps the feature $h(\mathbf{x})$ as follows:

$$\widehat{W}h(\mathbf{x}) \sim \begin{cases} \mathcal{N}(\rho\mu_+, \sigma^2) & : y = 1 \\ \mathcal{N}(\rho\mu_-, \sigma^2) & : y = -1 \end{cases}$$

since $\rho(\widehat{W}, W) = \rho(\widehat{W}h, Wh)$ so we have $-\widehat{W}h(\mathbf{x}) \cdot \hat{y} < 0$ if $\hat{y} = -y$. Eventually, (4) is minimized when $\pi_{\widehat{W}}(\mathbf{x}) \approx 1$ if output is corrupted, i.e., $\hat{y} = -y$, and $\pi_{\widehat{W}}(\mathbf{x}) \approx 0$ otherwise. In this way, we can make $\pi_{\widehat{W}}(\mathbf{x})$ to learn the corrupt probability for given data point.

We provide illustrative examples in both regression and classification that can support the aforementioned motivation in Section 3.4.

3.2. Cholesky Block for Correlated Sampling

Now we introduce a Cholesky transform that enables modeling dependencies among output mixtures in a differentiable manner. The Cholesky transform is a way of constructing a random variable which is correlated with other random variables and can be used as a sampling routine for sampling correlated matrices from two uncorrelated random matrices.

To be specific, suppose that $w \sim \mathcal{N}(\mu_w, \sigma_w^2)$ and $z \sim \mathcal{N}(0, \sigma_z^2)$ and our goal is to construct a random variable $\tilde{w}$ such that the correlation between w and $\tilde{w}$ becomes ρ. Then, the Cholesky transform which is defined as

$$\begin{aligned} &\texttt{Cholesky}(w, z, \rho, \mu_w, \sigma_w, \sigma_z) \\ &:= \rho\mu + \sqrt{1-\rho^2}\left(\rho\frac{\sigma_z}{\sigma_w}(w-\mu) + z\sqrt{1-\rho^2}\right) \end{aligned} \tag{5}$$

is a mapping from $(w, z) \in \mathbb{R}^2$ to $\mathbb{R}$ and can be used to construct $\tilde{w}$. In fact, $\tilde{w} = \texttt{Cholesky}(w, z, \rho, \mu_w, \sigma_w, \sigma_z)$ and we can easily use (5) to construct a feed-forward layer with a correlated weight matrix which will be referred to as a *Cholesky Block* as shown in Figure 2.

We also show that the correlation is preserved through the affine transformation making it applicable to a single fully-connected layer where all the derivations and proofs can be found in the supplementary material. In other words, we model correlated outputs by first sampling correlated weight matrices using Cholesky transfrom in an element-wise fashion and using the sampled weights for an affine transformation of a feature vector of a feed-forward layer. One can simply use a reparametrization trick [25] for implementations.

3.3. Overall Mechanism of ChoiceNet

In this section we describe the model architecture and the overall mechanism of ChoiceNet. In the following, $\tau^{-1} > 0$ is a small constant indicating the expected variance of the target distriubtion. $\mathbf{W}_{\mathbf{h}\to\boldsymbol{\rho}}$, $\mathbf{W}_{\mathbf{h}\to\boldsymbol{\pi}} \in \mathbb{R}^{K\times Q}$ and $\mathbf{W}_{\mathbf{h}\to\boldsymbol{\Sigma_0}} \in \mathbb{R}^{D\times Q}$ are weight matrices where Q and D denote the dimensions of a feature vector $\mathbf{h}$ and output $\mathbf{y}$, respectively, and K is the number of mixtures[1].

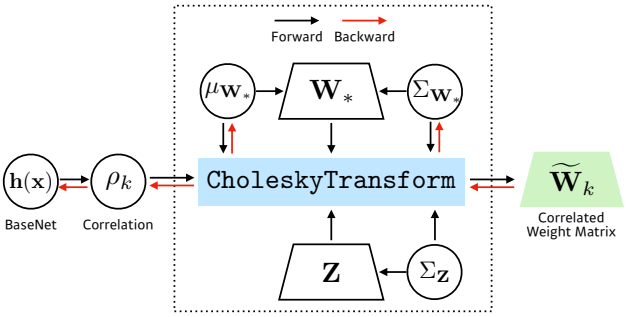


Figure 2: Illustration of a *Cholesky Block*. Every block shares target weight matrix $\mathbf{W}_*$ and auxiliary matrix $\mathbf{Z}$, and outputs correlated weight matrix $\widetilde{\mathbf{W}}_k$ through `CholeskyTransform` (see (5)) to distinguish the abnormal pattern from normal one which will be learned by $\mathbf{W}_*$.

ChoiceNet is a twofold architecture: (a) an arbitrary base network and (b) the *Cholesky Block* (see Figure 2). Once the base network extracts features, The *Cholesky Block* estimates the mixture of the target distribution and other correlated distributions using the Cholesky transform in (5). While it may seem to resemble a mixture density network (MDN) [5], this ability to model dependencies between mixtures lets us to effectively infer and distinguish the target distributions from other noisy distributions as will be shown in the experimental sections. When using the MDN, particularly, it is not straightforward to select which mixture to use other than using the one with the largest mixture probability which may lead to inferior performance given noisy datasets[2].

Let us elaborate on the overall mechanism of ChoiceNet. Given an input $\mathbf{x}$, a feature vector $\mathbf{h} \in \mathbb{R}^Q$ is computed from any arbitrary mapping such as ResNet [18] for images or word embedding [28] for natural languages. Then the network computes $K-1$ correlations, $\{\rho_1, \rho_2(\mathbf{h}), ..., \rho_K(\mathbf{h})\}$, for K mixtures where the first $\rho_1 = 1$ is reserved to model the target distribution, In other words, the first mixture becomes our target distribution and we use the predictions from the first mixture in the inference phase.

The mean and variance of the weight matrix, $\mu_{\mathbf{W}} \in \mathbb{R}^{Q\times D}$ and $\Sigma_{\mathbf{W}} \in \mathbb{R}^{Q\times D}$, are defined and updated for modeling correlated output distributions which is analogous to a Bayesian neural network [6], These matrices can be back-propagated using the reparametrization trick. The *Cholesky Block* also computes the base output variance $\Sigma_0(\mathbf{h})$ similar to an MDN.

Then we sample K weight matrices $\{\widetilde{\mathbf{W}}_i\}_{i=1}^K$ from $\{\mu_*, \Sigma_*\}$ and $\{\rho_1, \rho_2(\mathbf{h}), ..., \rho_K(\mathbf{h})\}$ using the Cholesky

[1] Ablation studies regarding changing K and τ^{-1} are shown in the supplementary material where the results show that these hyper-parameters are not sensitive to the overall learning results.

[2] We test the MDN in both regression and classification tasks and the MDN shows poor performances.

Algorithm 1: ChoiceNet Algorithm

Input : $\mathcal{D}_{\text{train}}$, K, τ, λ, $\mathbf{h} : \mathcal{X} \to \mathbb{R}^Q$
Initialize $\mu_*, \Sigma_* \in \mathbb{R}^{Q \times D}$
$\mathbf{W}_{\mathbf{h}\to\boldsymbol{\rho}}, \mathbf{W}_{\mathbf{h}\to\boldsymbol{\pi}}, \mathbf{W}_{\mathbf{h}\to\Sigma_0} \in \mathbb{R}^{K \times Q}$
while *True* **do**
  Sample $\mathbf{W}_* \sim \mathcal{N}(\mu_*, \Sigma_*)$, $\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, \Sigma_{\mathbf{Z}})$
  for $k \in \{1, \dots, K\}$ **do**
    $\rho_k = \tanh(\mathbf{W}_{\mathbf{h}\to\boldsymbol{\rho}}\mathbf{h})_k \qquad (\rho_1 = 1)$
    $\pi_k = \text{softmax}(\mathbf{W}_{\mathbf{h}\to\boldsymbol{\pi}}\mathbf{h})_k$
    $(\Sigma_0)_k = \exp(\mathbf{W}_{\mathbf{h}\to\Sigma_0}\mathbf{h})_k$
    $\Sigma_k = (1 - \rho_k^2)(\Sigma_0)_k + \tau^{-1}$
    $\widetilde{\mathbf{W}}_k = \texttt{Cholesky}(\mathbf{W}_*, \mathbf{Z}, \rho_k, \mu_*, \Sigma_*, \Sigma_{\mathbf{Z}})$
    $\mu_k = \widetilde{\mathbf{W}}_k\mathbf{h}$
  end
  Compute $\mathcal{L}(\mathcal{D}_{\text{train}}|(\pi_k, \mu_k, \Sigma_k)_{k=1}^K)$
  Update $\mathbf{h}, \mathbf{W}_{\mathbf{h}\to\boldsymbol{\rho}}, \mathbf{W}_{\mathbf{h}\to\boldsymbol{\pi}}, \mathbf{W}_{\mathbf{h}\to\Sigma_0}, \mu_*, \Sigma_*$
end
return $\mathbf{W}_*, \mathbf{h}$

transform (5) so that the correlations between $\widetilde{\mathbf{W}}_i$ and $\mathbf{W}_*$ becomes $\rho_i(\cdot)$. Note that the correlation is preserved through an affine transform. The K sampled feedforward weights, $\{\widetilde{\mathbf{W}}_i\}_{i=1}^K$, are used to compute K correlated output mean vectors, $\{\mu_i\}_{i=1}^K$. Note that the correlation between μ_1 and μ_i also becomes $\rho_i(\cdot)$. We would like to emphasize that, as we employ Gaussian distributions in the Cholesky transform, the influences of uninformative or independent data, whose correlations, ρ, are close to 0, is attenuated as their variances increase [23].

The output variances of k-th mixture is computed from $\rho_k(\mathbf{h})$ and the based output variance $\Sigma_0(\mathbf{h})$ as follows:

$$\Sigma_k = (1 - \rho_k^2(\mathbf{h}))\Sigma_0(\mathbf{h}) + \tau^{-1} \in \mathbb{R}^D \tag{6}$$

This has an effect of increasing the variance of the mixture which is less related (in terms of the absolute value of a correlation) with the target distribution. The *Cholesky Block* also computes the mixture probabilities of k-th mixture, $\pi_k(\mathbf{h})$, akin to an MDN. The overall process of the *Cholesky Block* is summarized in Algorithm 1 and Figure 3. Now, let us introduce the loss functions for regression and classification tasks.

Regression For the regression task, we employ both L_2-loss and the standard MDN loss ([5, 8, 10])

$$\mathcal{L}(\mathcal{D}) = \frac{1}{N}\sum_{i=1}^N \lambda_1 \|\mathbf{y}_i - \mu_1(\mathbf{x}_i)\|_2^2 + \frac{1}{N}\sum_{i=1}^N \lambda_2 \log\left(\sum_{k=1}^K \pi_k(\mathbf{x}_i)\mathcal{N}(\mathbf{y}_i; \mu_k(\mathbf{x}_i), \Sigma_k(\mathbf{x}_i))\right) \tag{7}$$

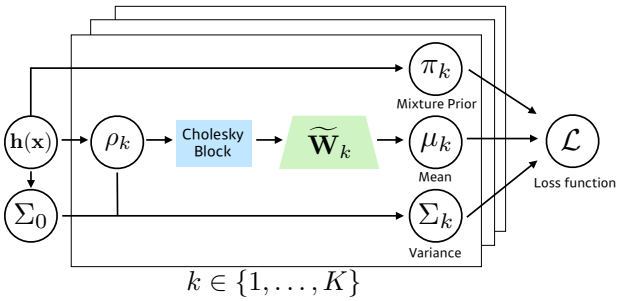


Figure 3: Overall mechanism of ChoiceNet. It consists of K mixtures and each mixture outputs triplet (π_k, μ_k, Σ_k) via Algorithm 1. $\rho_1 = 1$ is reserved to model the target distribution.

where λ_1 and λ_2 are hyper-parameters and $\mathcal{N}(\cdot|\mu, \Sigma)$ is the density of multivariate Gaussian:

$$\mathcal{N}(\mathbf{y}_i; \mu_k, \Sigma_k) = \prod_{d=1}^D \frac{1}{\sqrt{2\pi\Sigma_k^{(d)}}} \exp\left(-\frac{|y_i^{(d)} - \mu_k^{(d)}|^2}{2\Sigma_k^{(d)}}\right)$$

We also propose the following Kullback-Leibler regularizer:

$$\mathbb{KL}(\bar{\boldsymbol{\rho}}\|\boldsymbol{\pi}) = \sum_{k=1}^K \bar{\boldsymbol{\rho}}_k \log\frac{\bar{\boldsymbol{\rho}}_k}{\pi_k}, \quad \bar{\boldsymbol{\rho}} = \text{softmax}(\boldsymbol{\rho}) \tag{8}$$

The above KL regularizer encourages the mixture components with the strong correlations to have high mixture probabilities. Note that we use the notion of correlation to evaluate the goodness of each training data where the first mixture whose correlation is always 1 is reserved for modeling the target distribution.

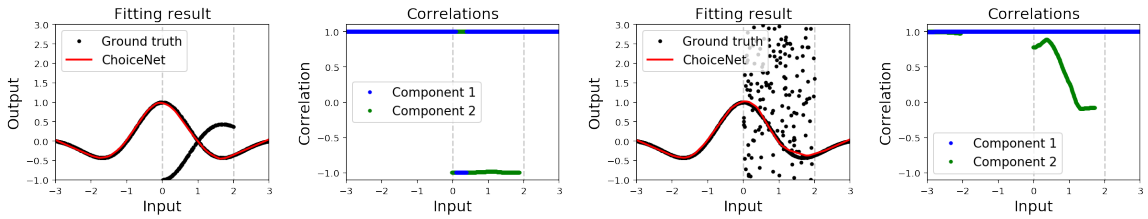
Classification In the classification task, we suppose each $\mathbf{y}_i$ is a D-dimensional one-hot vector. Unlike the regression task, (7) is not appropriate for the classification task. We employ the following loss function:

$$\mathcal{L}(\mathcal{D}) = -\frac{1}{N}\sum_{i=1}^N \sum_{k=1}^K \pi_k(\mathbf{x}_i) l(\mathbf{x}_i, \mathbf{y}_i) \tag{9}$$

where

$$l(\mathbf{x}_i, \mathbf{y}_i) = \langle \text{softmax}(\hat{\mathbf{y}}_k(\mathbf{x}_i)), \mathbf{y}_i \rangle - \lambda_{\text{reg}} \log\left(\sum_{d=1}^D \exp(\hat{y}_k^{(d)}(\mathbf{x}_i))\right).$$

Here $\langle \cdot, \cdot \rangle$ denotes inner product, $\hat{\mathbf{y}}_k = (\hat{y}_k^{(1)}, \dots, \hat{y}_k^{(D)})$, and $\hat{y}_k^{(d)}(\mathbf{x}_i) = \mu_k^{(d)} + \sqrt{\Sigma_k^{(d)}}\varepsilon$ where $\varepsilon \sim \mathcal{N}(0, 1)$. Similar loss function was used in [24] which also utilizes a Gaussian mixture model.



(a) (b) (c) (d)

Figure 4: Fitting results on datasets with (a) flipped function and (c) uniform corruptions. Resulting correlations of two components with (b) flipped function and (d) uniform corruptions.

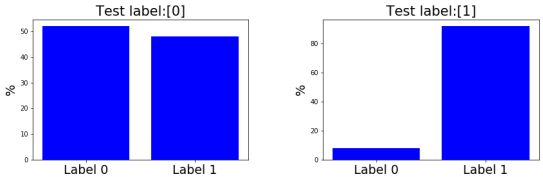
3.4. Illustrative Synthetic Examples

Here, we provide synthetic examples in both regression and classification to illustrate how the proposed method can be used to robustly learn the underlying target distribution given a noisy training dataset.

Regression Task We first focus on how the proposed method can distinguish between the target distribution and noise distributions in a regression task and show empirical evidence that our method can achieve this by estimating the target distribution and the correlation of noise distribution simultaneously. We train on two datasets with the same target function but with different types of corruptions by replacing 50% of the output values whose input values are within 0 to 2: one uniformly sampled from -1 to 3 and the other from a flipped target function as shown in Figure 4(a) and 4(c). Throughout this experiment, we set $K = 2$ for better visualization.

As shown in Figure 4, ChoiceNet successfully estimates the target function with partial corruption. To further understand how ChoiceNet works, we also plot the correlation of each mixture at each input with different colors. When using the first dataset where we flip the outputs whose inputs are between 0 and 2, the correlations of the second mixture at the corrupt region becomes -1 (see Figure 4(b)). This is exactly what we wanted ChoiceNet to behave in that having -1 correlation will simply flip the output. In other words, the second mixture *takes care of* the noisy (flipped) data by assigning -1 correlation while the first mixture component reserved for the target distribution is less affected by the noisy training data.

When using the second dataset, the correlations of the second mixture at the corrupt region are not -1 but decreases as the input increase from 0 to 2 (see Figure 4(d)). Intuitively speaking, this is because the average deviations between the noisy output and the target output increases as the input increases. Since decreasing the correlation from 1 to 0 will increase the output variance as shown in (6), the correlation of the second mixture tends to decrease as the input increase between 0 to 2. This clearly shows the capability of ChoiceNet to distinguish the target distribution from noisy distributions.



(a) (b)

Figure 5: The predictions results of the second mixture of test inputs whose labels are (a) 0 and (b) 1, respectively.

Binary Classification Task We also provide an illustrative binary classification task using label 0 and 1 from the MNIST dataset. In particular, we replaced 40% of the training data with label 0 to 1. To implement ChoiceNet, we use two-layer convolutional neural networks with 64 channels and two mixtures. We trained ChoiceNet for 10 epochs where the final train and test accuracies are 81.7% and 98.1%, respectively, which indicates ChoiceNet successfully infers clean data distribution. As the first mixture is deserved for inferring the clean target distribution, we expect the second mixture to take away corrupted labels. Figure 5 shows two prediction results of the second mixture when given test inputs with label 0 and 1. As 40% of training labels whose labels are originally 0 are replaced to 1, almost half of the second mixture predictions of test images whose labels are 0 are 1 indicating that it takes away corrupted labels while training. On the contrary, the second mixture predictions of test inputs whose original labels are 1 are mostly focusing on label 1.

4. Experiments

In this section, we validate the performance of ChoiceNet on both regression problems in Section 4.1 and classification problems in Section 4.2 where we mainly focus on evaluating the robustness of the proposed method.

4.1. Regression Tasks

Here, we show the regression performances using synthetic regression tasks and a Boston housing dataset with

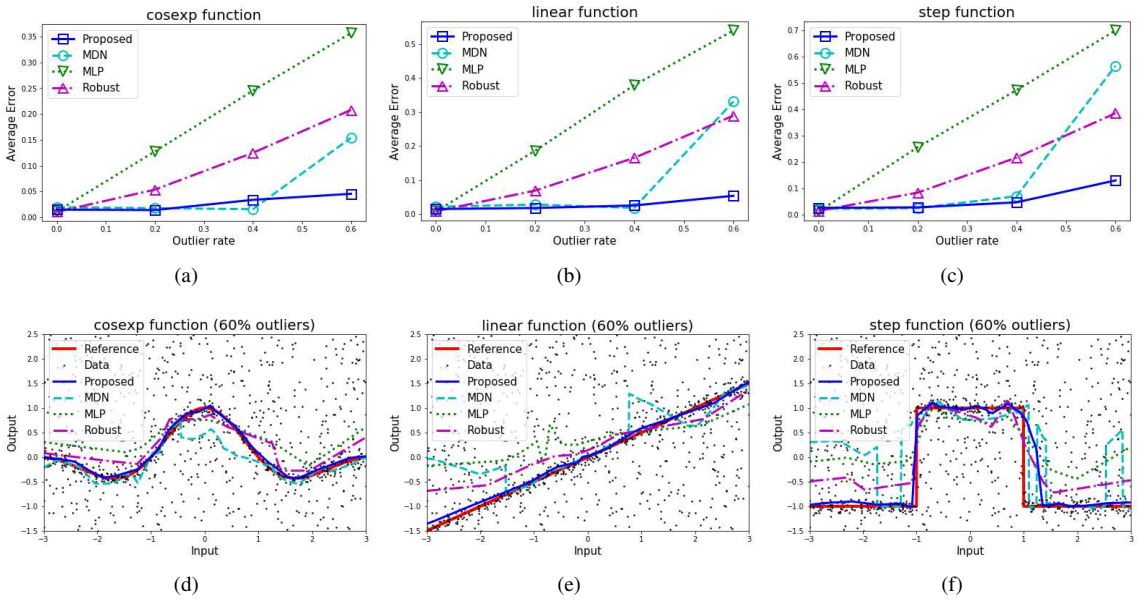


Figure 6: (a-c) Average fitting errors while varying the outlier rates and (e-f) fitting results of the compared methods with 60% outliers using *cosexp*, *linear*, and *step* functions.

outliers. More experimental results using synthetic data and behavior cloning scenarios in MuJoCo environments where the demonstrations are collected and mixed from both expert and adversarial policies as well as detailed experimental settings can be found in the supplementary material.

Synthetic Data In order to evaluate the robustness of the proposed method, we use three different 1-D target functions. Particularly, we use the following target functions: *cosexp*, *linear*, and *step* functions as shown in Figure 6(d)-6(f), respectively, and collected $1,000$ points per each function while replacing a certain portion of outputs to random values uniformly sampled between -1.5 and 2.5. We compared our proposed method with a mixture density network (MDN) [5] and fully connected layers with an L_2-loss (MLP) and a robust loss (Robust) proposed in [3]. We use three-layers with 64 units and ReLU activations, and for both the proposed method and an MDN, five mixtures are used.

The average absolute fitting errors of three different functions are shown in Figure 6(a)-6(c), respectively where we can see that the proposed method outperforms or show comparable results with low outlier rates and shows superior performances with a high outlier rate ($> 50\%$). Figure 6(d)-6(f) show the fitting results along with training data. While our proposed method is built on top of an MDN, it is worthwhile noting the severe performance degradation of an MDN with an extreme noise level (60%). It is mainly because an MDN fails to allocate high mixture probability on the mixture corresponds to the target distribution as there are no dependencies among different mixtures.

Table 1: The RMSEs of compared methods on the Boston Housing Dataset

Outliers	ChoiceNet	$L2$	$L1$	RL	LeakyRL	MDN
0%	3.29	**3.22**	3.26	4.28	3.36	3.46
10%	**3.99**	5.97	5.72	6.36	5.71	6.5
20%	**4.77**	7.51	7.16	8.08	7.08	8.62
30%	**5.94**	9.04	8.65	10.54	8.67	8.97
40%	**6.80**	9.88	9.69	10.94	9.68	10.44

Table 2: Test accuracies on the CIFAR-10 dataset with by symmetric and asymmetric noises.

	Pair-45%	Sym-50%	Sym-20%
Standard	49.5	48.87	76.25
+ ChoiceNet	70.3	**85.2**	**91.0**
MentorNet	58.14	71.10	80.76
Co-teaching	**72.62**	74.02	82.32
F-correction	6.61	59.83	59.83
MDN	51.4	58.6	81.4

Boston Housing Dataset A Boston housing price dataset is used to check the robustness of the proposed method. We compare our method with standard feedforward networks using four different loss functions: standard $L2$-loss, $L1$-loss which is known to be robust to outliers, a robust loss (RL) function proposed in [3], and a leaky robust loss (LeakyRL) function where the results are shown in Table 1. Implementation details can be found in the supplementary material.

Data	Model	MNIST 20%	MNIST 50%	FMNIST 20%	FMNIST 50%	SVHN 20%	SVHN 50%	CIFAR10 20%	CIFAR10 50%	CIFAR100 20%	CIFAR100 50%
D_{test}	WideResNet	82.19	56.63	82.32	56.55	82.21	55.94	81.93	54.97	80.19	50.48
	+ ChoiceNet	99.66	99.03	97.46	95.36	94.40	78.32	96.58	90.28	85.81	68.89
D_{train}	WideResNet	99.29	92.49	98.62	92.42	99.34	95.91	99.97	99.82	99.96	99.91
	+ ChoiceNet	81.99	55.38	80.63	54.45	78.99	48.69	81.88	56.83	26.03	18.14
Expected True Ratio		82%	55%	82%	55%	82%	55%	82 %	55%	80.2%	50.5 %

Table 3: The comparison between naive WideResNet and ChoiceNet on multile benchmark datasets.

4.2. Classification Tasks

Here, we conduct comprehensive classification experiments to investigate how ChoiceNet performs on image classification tasks with noisy labels. More experimental results on MNIST, CIFAR-10, and a natural language processing task and ablation studies of hyper-parameters can also be found in the supplementary material.

CIFAR-10 Comparison We first evaluate the performance of our method and compare it with MentorNet [21], Co-teaching [17] and F-correction [34] on noisy CIFAR-10 datasets. We follow three different noise settings from [17]: PairFlip-45%, Symmetry-50%, and Symmetry-20%. On 'Symmetry' settings, noisy labels can be assigned to all classes, while, on 'PairFlip' settings, all noisy labels from the same true label are assigned to a single noisy label. A model is trained on a noisy dataset and evaluated on the clean test set. For fair comparisons, we keep other configurations such as the network topology to be the same as [17].

Table 2 shows the test accuracy of compared methods under different noise settings. Our proposed method outperforms all compared methods on the symmetric noise settings with a large margin over 10%p. On asymmetric noise settings (Pair-45%), our method shows the second best performance, and this reveals the weakness of the proposed method. As Pair-45% assigns 45% of each label to its next label, The *Cholesky Block* fails to infer the dominant label distributions correctly. However, we would like to note that Co-teaching [17] is complementary to our method where one can combine these two methods by using two ChoiceNets and update each network using Co-teaching.

Generalization to Other Datasets We conduct additional experiments to investigate how our method works on other image datasets. We adopt the structure of WideResNet [43] and design a baseline network with a depth of 22 and a widen factor of 4. We also construct ChoiceNet by replacing the last two layers ('average pooling' and 'linear') with the *Cholesky Block*. We train the networks on CIFAR-10, CIFAR-100, FMNIST, MNIST and SVHN datasets with noisy labels. We generate noisy datasets with symmetric noise setting and vary corruption ratios from 20% to 50%. We would like to emphasize that we use the same hyper-parameters, which are tuned for the CIFAR-10 experiments, for all the datasets except for CIFAR-100[3].

Table 3 shows test accuracies of our method and the baseline on various image datasets with noisy labels. ChoiceNet consistently outperforms the baseline in most of the configurations except for 80%-corrupted SVHN dataset. Moreover, we expect that performance gains can increase when dataset-specific hyperparameter tuning is applied. The results suggest that the proposed ChoiceNet can easily be applied to other noisy datasets and show a performance improvement without large efforts.

We would like to emphasize that the training accuracy of the proposed method in Table 3 is close to the expected true ratio[4]. This implies that our proposed *Cholesky Block* can separate true labels and false labels from a noisy dataset. We also note that training ChoiceNet on CIFAR-100 datasets requires a modification in a loss weight. We hypothesize that the hyperparameters of our proposed method are sensitive to a number of target classes in a dataset.

5. Conclusion

In this paper, we have presented ChoiceNet that can robustly learn a target distribution given noisy training data. The keystone of ChoiceNet is the mixture of correlated densities network block which can simultaneously estimate the underlying target distribution and the quality of each data where the quality is defined by the correlation between the target and generating distributions. We have demonstrated that the proposed method can robustly infer the target distribution on corrupt training data in both regression and classification tasks. However, we have seen that in the case of extreme asymmetric noises, the proposed method showed suboptimal performances. We believe that it could resolved by combining our method with other robust learning methods where we have demonstrated that ChoiceNet can effectively be combined with mix-up [45]. Furthermore, one can use ChoiceNet for active learning by evaluating the quality of each training data using through the lens of correlations.

[3] On CIFAR-100 dataset, we found the network underfits with our default hyper-parameters. Therefore, we enlarged λ_{reg} to $1e-3$ for CIFAR-100 experiments.

[4] Since a noisy label can be assigned to any labels, we can expect 82%, 55% true labels on noisy datasets with a corruption probability of 20%, 50%, respectively.

References

[1] AP and REUTERS. Tesla working on 'improvements' to its autopilot radar changes after model s owner became the first self-driving fatality., June 2016.

[2] Alan Joseph Bekker and Jacob Goldberger. Training deep neural-networks based on unreliable labels. In *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2682–2686. IEEE, 2016.

[3] Vasileios Belagiannis, Christian Rupprecht, Gustavo Carneiro, and Nassir Navab. Robust optimization for deep regression. In *Proc. of the IEEE International Conference on Computer Vision*, pages 2830–2838, 2015.

[4] Wei Bi, Liwei Wang, James T Kwok, and Zhuowen Tu. Learning to predict from crowdsourced data. In *UAI*, pages 82–91, 2014.

[5] Christopher M Bishop. Mixture density networks. 1994.

[6] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *International Conference on Machine Learning (ICML)*, 2015.

[7] Edwin V Bonilla, Kian M Chai, and Christopher Williams. Multi-task gaussian process prediction. In *Proc. of the Advances in Neural Information Processing Systems*, pages 153–160, 2008.

[8] Sungjoon Choi, Kyungjae Lee, Sungbin Lim, and Songhwai Oh. Uncertainty-aware learning from demonstration using mixture density networks with sampling-free variance modeling. *arXiv preprint arXiv:1709.02249*, 2017.

[9] Sungjoon Choi, Kyungjae Lee, and Songhwai Oh. Robust learning from demonstration using leveraged Gaussian processes and sparse constrained opimization. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2016.

[10] M Bishop Christopher. *PATTERN RECOGNITION AND MACHINE LEARNING*. Springer-Verlag New York, 2016.

[11] Alhussein Fawzi, Seyed Mohsen Moosavi Dezfooli, and Pascal Frossard. A geometric perspective on the robustness of deep networks. *IEEE Signal Processing Magazine*, 2017.

[12] Aritra Ghosh, Himanshu Kumar, and PS Sastry. Robust loss functions under label noise for deep neural networks. In *Proc. of the AAAI Conference on Artificial Intelligence*, pages 1919–1925, 2017.

[13] Jacob Goldberger and Ehud Ben-Reuven. Training deep neural-networks using a noise adaptation layer. In *Proc. of International Conference on Learning Representations*, 2017.

[14] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[15] Varun Gulshan, Lily Peng, Marc Coram, Martin C Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Journal of the American Medical Association*, 316(22):2402–2410, 2016.

[16] Frank R Hampel, Elvezio M Ronchetti, Peter J Rousseeuw, and Werner A Stahel. *Robust statistics: the approach based on influence functions*, volume 196. John Wiley & Sons, 2011.

[17] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: robust training deep neural networks with extremely noisy labels. In *Proc. of the Advances in Neural Information Processing Systems*, 2018.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of the IEEE conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[19] Dan Hendrycks, Mantas Mazeika, Duncan Wilson, and Kevin Gimpel. Using trusted data to train deep networks on labels corrupted by severe noise. *arXiv preprint arXiv:1802.05300*, 2018.

[20] Peter J Huber. *Robust statistics*. Springer, 2011.

[21] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Regularizing very deep neural networks on corrupted labels. *arXiv preprint arXiv:1712.05055*, 2017.

[22] Ishan Jindal, Matthew Nokleby, and Xuewen Chen. Learning deep networks from noisy labels with dropout regularization. In *Proc. of IEEE International Conference onData Mining*, pages 967–972. IEEE, 2016.

[23] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems*, pages 5580–5590, 2017.

[24] Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? In *Proc. of the Advances in Neural Information Processing Systems*, 2017.

[25] Diederik P Kingma. Variational inference & deep learning: A new synthesis. *University of Amsterdam*, 2017.

[26] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *Proc. of International Conference on Learning Representations*, 2017.

[27] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, volume 3, page 2, 2013.

[28] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185, 2014.

[29] Xin Liu, Shaoxin Li, Meina Kan, Shiguang Shan, and Xilin Chen. Self-error-correcting convolutional neural network for learning with noisy labels. In *Proc. of IEEE International Conference on Automatic Face & Gesture Recognition*, pages 111–117. IEEE, 2017.

[30] Eran Malach and Shai Shalev-Shwartz. Decoupling" when to update" from" how to update". In *Advances in Neural Information Processing Systems*, pages 961–971, 2017.

[31] Takeru Miyato, Shin-ichi Maeda, Shin Ishii, and Masanori Koyama. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[32] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Proc. of the Advances in Neural Information Processing Systems*, pages 1196–1204, 2013.

[33] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, 2016.

[34] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: a loss correction approach. In *Proc. of the Conference on Computer Vision and Pattern Recognition*, volume 1050, page 22, 2017.

[35] Carl Edward Rasmussen. Gaussian processes for machine

learning. 2006.

[36] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014.

[37] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *Proc. of International Conference on Machine Learning*, 2018.

[38] Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*, 2014.

[39] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204, 2017.

[40] Yuji Tokozume, Yoshitaka Ushiku, and Tatsuya Harada. Proc. of international conference on learning representations. 2018.

[41] Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge Belongie. Learning from noisy large-scale datasets with minimal supervision. In *Conference on Computer Vision and Pattern Recognition*, 2017.

[42] Lingxi Xie, Jingdong Wang, Zhen Wei, Meng Wang, and Qi Tian. Disturblabel: Regularizing cnn on the loss layer. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4753–4762, 2016.

[43] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

[44] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *Proc. of International Conference on Learning Representations*, 2016.

[45] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *Proc. of International Conference on Learning Representations*, 2017.