A Study of Robust Learning under Label Noise with Neural Networks

Deep Patel
Advisor: Prof P. S. Sastry

Learning Systems and Multimedia Lab
Department of Electrical Engineering
Indian Institute of Science, Bangalore - 560 012

23rd April 2021

Outline

Supervised Learning - Classification problem

- Classifier learning is a widely studied problem

Supervised Learning - Classification problem

- Classifier learning is a widely studied problem
- Typically, one has access to labelled training set that's used to train a model/classifier. Hence the name.

Supervised Learning - Classification problem

- Classifier learning is a widely studied problem
- Typically, one has access to labelled training set that's used to train a model/classifier. Hence the name.
- Given the success of deep learning in the last decade and need for large scale datasets, label errors are inevitable.

Supervised Learning - Classification problem

- Classifier learning is a widely studied problem
- Typically, one has access to labelled training set that's used to train a model/classifier. Hence the name.
- Given the success of deep learning in the last decade and need for large scale datasets, label errors are inevitable.
- These labelling errors can be due to automated labelling processes, crowdsourced annotations, human errors, etc.

Supervised Learning - Classification problem

- Classifier learning is a widely studied problem
- Typically, one has access to labelled training set that's used to train a model/classifier. Hence the name.
- Given the success of deep learning in the last decade and need for large scale datasets, label errors are inevitable.
- These labelling errors can be due to automated labelling processes, crowdsourced annotations, human errors, etc.
- That's why the problem of robust learning under label noise is relevant and needs to be studied in detail.

Supervised Learning - Formulation

- **Assumption:** i.i.d. samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$ $(\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathbb{R}^n, y_i \in \mathcal{Y} \subseteq \mathbb{R}^K)$ drawn from underlying distribution, $\mathcal{D}$

Supervised Learning - Formulation

- **Assumption:** i.i.d. samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$ $(\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathbb{R}^n, y_i \in \mathcal{Y} \subseteq \mathbb{R}^K)$ drawn from underlying distribution, $\mathcal{D}$
- $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$ constitute the training set, S

Supervised Learning - Formulation

- **Assumption:** i.i.d. samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$ $(\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathbb{R}^n, y_i \in \mathcal{Y} \subseteq \mathbb{R}^K)$ drawn from underlying distribution, $\mathcal{D}$
- $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$ constitute the training set, S
- The goal is to learn a score function, $f : \mathcal{X} \to \mathcal{Y}$.

Supervised Learning - Formulation

- **Assumption:** i.i.d. samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$ $(\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathbb{R}^n, y_i \in \mathcal{Y} \subseteq \mathbb{R}^K)$ drawn from underlying distribution, $\mathcal{D}$
- $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$ constitute the training set, S
- The goal is to learn a score function, $f : \mathcal{X} \to \mathcal{Y}$.
- For classification problem, $\mathcal{Y} = [K] = \{1, 2, \ldots, K\}$ where $K \in \mathbb{Z}_+$.

Supervised Learning - Formulation

- **Assumption:** i.i.d. samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ $(\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathbb{R}^n, y_i \in \mathcal{Y} \subseteq \mathbb{R}^K)$ drawn from underlying distribution, $\mathcal{D}$
- $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ constitute the training set, S
- The goal is to learn a score function, $f : \mathcal{X} \to \mathcal{Y}$.
- For classification problem, $\mathcal{Y} = [K] = \{1, 2, \ldots, K\}$ where $K \in \mathbb{Z}_+$.
- After learning the score function, $f : \mathcal{X} \to \mathcal{M} \subseteq \mathbb{R}^K$, the label prediction is done with the help of a 'pred' function

Supervised Learning - Formulation

- **Assumption:** i.i.d. samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$ $(\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathbb{R}^n, y_i \in \mathcal{Y} \subseteq \mathbb{R}^K)$ drawn from underlying distribution, $\mathcal{D}$
- $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$ constitute the training set, S
- The goal is to learn a score function, $f : \mathcal{X} \to \mathcal{Y}$.
- For classification problem, $\mathcal{Y} = [K] = \{1, 2, \ldots, K\}$ where $K \in \mathbb{Z}_+$.
- After learning the score function, $f : \mathcal{X} \to \mathcal{M} \subseteq \mathbb{R}^K$, the label prediction is done with the help of a 'pred' function
- So, the classifier function that we want to learn can be written as $h(\cdot) = pred \circ f(\cdot)$. For instance, pred($o$) could be the max function selecting the index corresponding to the maximum component of $o \in \mathcal{M}$

Risk Minimization framework

- Many supervised algorithms such as Support Vector Machines (SVMs), AdaBoost, logistic regression, neural networks, etc. can be formulated as a *risk minimization* problem.

Risk Minimization framework

- Many supervised algorithms such as Support Vector Machines (SVMs), AdaBoost, logistic regression, neural networks, etc. can be formulated as a *risk minimization* problem.
- *Risk* is defined w.r.t a particular distribution, $\mathcal{D}$, and loss function, L

Risk Minimization framework

- Many supervised algorithms such as Support Vector Machines (SVMs), AdaBoost, logistic regression, neural networks, etc. can be formulated as a *risk minimization* problem.
- *Risk* is defined w.r.t a particular distribution, $\mathcal{D}$, and loss function, L
- Loss function, $L : (\mathcal{M} \times \mathcal{Y}) \to \mathbb{R}_+$, is user-defined and chosen such that it captures the objectives of the learning process.

Risk Minimization framework

- Many supervised algorithms such as Support Vector Machines (SVMs), AdaBoost, logistic regression, neural networks, etc. can be formulated as a *risk minimization* problem.
- *Risk* is defined w.r.t a particular distribution, $\mathcal{D}$, and loss function, L
- Loss function, $L : (\mathcal{M} \times \mathcal{Y}) \to \mathbb{R}_+$, is user-defined and chosen such that it captures the objectives of the learning process.
- *Risk* for a given classifier function, f, and loss function, L, w.r.t the underlying distribution, $\mathcal{D}$, can be defined as:

$$R_L(f) = \mathbb{E}_{\mathcal{D}}[L(f(\boldsymbol{x}), y_{\boldsymbol{x}})] \tag{1}$$

Risk Minimization framework (contd.)

- We denote a global minimizer of this risk by f^*:

$$f^* = \arg\min_{f \in \mathcal{F}} R_L(f) = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{\mathcal{D}}[L(f(\boldsymbol{x}), y_{\boldsymbol{x}})]$$

  where $\mathcal{F}$ is a hypothesis space over which we learn a classifier function.

Risk Minimization framework (contd.)

- We denote a global minimizer of this risk by f^*:

$$f^* = \arg\min_{f \in \mathcal{F}} R_L(f) = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{\mathcal{D}}[L(f(\boldsymbol{x}), y_{\boldsymbol{x}})]$$

  where $\mathcal{F}$ is a hypothesis space over which we learn a classifier function.
- In practice, we minimize empirical risk as we only have a few samples from the unknown, underlying distribution.

Risk Minimization framework (contd.)

- We denote a global minimizer of this risk by f^*:

$$f^* = \arg\min_{f \in \mathcal{F}} R_L(f) = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{\mathcal{D}}[L(f(\boldsymbol{x}), y_{\boldsymbol{x}})]$$

  where $\mathcal{F}$ is a hypothesis space over which we learn a classifier function.
- In practice, we minimize empirical risk as we only have a few samples from the unknown, underlying distribution.
- Empirical risk is defined as:

$$\hat{R}_L(f) = \frac{1}{m} \sum_{i=1}^{m} L(f(\boldsymbol{x}_i), y_{\boldsymbol{x}_i}) \tag{2}$$

Risk Minimization framework (contd.)

- We denote a global minimizer of this risk by f^*:

$$f^* = \arg\min_{f \in \mathcal{F}} R_L(f) = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{\mathcal{D}}[L(f(\boldsymbol{x}), y_{\boldsymbol{x}})]$$

  where $\mathcal{F}$ is a hypothesis space over which we learn a classifier function.
- In practice, we minimize empirical risk as we only have a few samples from the unknown, underlying distribution.
- Empirical risk is defined as:

$$\hat{R}_L(f) = \frac{1}{m} \sum_{i=1}^{m} L(f(\boldsymbol{x}_i), y_{\boldsymbol{x}_i}) \tag{2}$$

- For this thesis, all training algorithms implement empirical risk minimization for a user-specified loss function.

Why study learning under label noise?

- In case of label noise, one has access only to samples from the noisy distribution, $\mathcal{D}_\eta$, and not the clean distribution, $\mathcal{D}$.

Why study learning under label noise?

- In case of label noise, one has access only to samples from the noisy distribution, $\mathcal{D}_\eta$, and not the clean distribution, $\mathcal{D}$.

Why study learning under label noise?

- In case of label noise, one has access only to samples from the noisy distribution, $\mathcal{D}_\eta$, and not the clean distribution, $\mathcal{D}$.
- Many supervised learning algorithms such as naive Bayes, k-Nearest Neighbours (kNNs), SVMs, Decision Trees (DTs), logistic regression, AdaBoost, etc. perform poorly in presence of label noise.

Why study learning under label noise?

- In case of label noise, one has access only to samples from the noisy distribution, $\mathcal{D}_\eta$, and not the clean distribution, $\mathcal{D}$.
- Many supervised learning algorithms such as naive Bayes, k-Nearest Neighbours (kNNs), SVMs, Decision Trees (DTs), logistic regression, AdaBoost, etc. perform poorly in presence of label noise.
- In the context of neural networks, [54, 3] demonstrate that neural networks are able to memorize the entire training dataset for any amount of label noise. This raises interesting questions for generalization in deep networks.

Problem Formulation

- We will now formally state the problem of robust learning under label noise

Problem Formulation

- We will now formally state the problem of robust learning under label noise
- Risk Minimization framework gives us a language to formally state it. So, throughout this thesis, we use that as a backbone for understanding the problem of learning under label noise.

Problem Formulation

- We will now formally state the problem of robust learning under label noise
- Risk Minimization framework gives us a language to formally state it. So, throughout this thesis, we use that as a backbone for understanding the problem of learning under label noise.
- Before we go ahead, we will look at the notation first

Problem Formulation (contd.)

Notation:

- $\mathcal{D}$ – clean, underlying distribution

Problem Formulation (contd.)

Notation:

- $\mathcal{D}$ – clean, underlying distribution
- $\mathcal{D}_\eta$ – noisy distribution

Problem Formulation (contd.)

Notation:

- $\mathcal{D}$ – clean, underlying distribution
- $\mathcal{D}_\eta$ – noisy distribution
- $S = \{(\boldsymbol{x}, y_{\boldsymbol{x}}^{cl})\}_{i=1}^{m}$ – training set with true labels

Problem Formulation (contd.)

Notation:

- $\mathcal{D}$ – clean, underlying distribution
- $\mathcal{D}_\eta$ – noisy distribution
- $S = \{(\boldsymbol{x}, y_{\boldsymbol{x}}^{cl})\}_{i=1}^{m}$ – training set with true labels
- $S_\eta = \{(\boldsymbol{x}, y_{\boldsymbol{x}})\}_{i=1}^{m}$ – training set with noisy labels

Problem Formulation (contd.)

Notation:

- $\mathcal{D}$ – clean, underlying distribution
- $\mathcal{D}_\eta$ – noisy distribution
- $S = \{(\boldsymbol{x}, y_{\boldsymbol{x}}^{cl})\}_{i=1}^{m}$ – training set with true labels
- $S_\eta = \{(\boldsymbol{x}, y_{\boldsymbol{x}})\}_{i=1}^{m}$ – training set with noisy labels
- f – classifier function

Problem Formulation (contd.)

Notation:

- $\mathcal{D}$ – clean, underlying distribution
- $\mathcal{D}_\eta$ – noisy distribution
- $S = \{(\boldsymbol{x}, y_{\boldsymbol{x}}^{cl})\}_{i=1}^{m}$ – training set with true labels
- $S_\eta = \{(\boldsymbol{x}, y_{\boldsymbol{x}})\}_{i=1}^{m}$ – training set with noisy labels
- f – classifier function
- $R_L(f)$ – risk for classifier, f, and loss function, L, w.r.t. $\mathcal{D}$

Problem Formulation (contd.)

Notation:

- $\mathcal{D}$ – clean, underlying distribution
- $\mathcal{D}_\eta$ – noisy distribution
- $S = \{(\boldsymbol{x}, y_{\boldsymbol{x}}^{cl})\}_{i=1}^{m}$ – training set with true labels
- $S_\eta = \{(\boldsymbol{x}, y_{\boldsymbol{x}})\}_{i=1}^{m}$ – training set with noisy labels
- f – classifier function
- $R_L(f)$ – risk for classifier, f, and loss function, L, w.r.t. $\mathcal{D}$
- f^* – a global minimizer of $R_L(f)$

Problem Formulation (contd.)

Notation:

- $\mathcal{D}$ – clean, underlying distribution
- $\mathcal{D}_\eta$ – noisy distribution
- $S = \{(\boldsymbol{x}, y_{\boldsymbol{x}}^{cl})\}_{i=1}^m$ – training set with true labels
- $S_\eta = \{(\boldsymbol{x}, y_{\boldsymbol{x}})\}_{i=1}^m$ – training set with noisy labels
- f – classifier function
- $R_L(f)$ – risk for classifier, f, and loss function, L, w.r.t. $\mathcal{D}$
- f^* – a global minimizer of $R_L(f)$
- $R_L^\eta(f)$ – risk for classifier, f, and loss function, L, w.r.t. $\mathcal{D}_\eta$

Problem Formulation (contd.)

Notation:

- $\mathcal{D}$ – clean, underlying distribution
- $\mathcal{D}_\eta$ – noisy distribution
- $S = \{(\boldsymbol{x}, y_{\boldsymbol{x}}^{cl})\}_{i=1}^{m}$ – training set with true labels
- $S_\eta = \{(\boldsymbol{x}, y_{\boldsymbol{x}})\}_{i=1}^{m}$ – training set with noisy labels
- f – classifier function
- $R_L(f)$ – risk for classifier, f, and loss function, L, w.r.t. $\mathcal{D}$
- f^* – a global minimizer of $R_L(f)$
- $R_L^\eta(f)$ – risk for classifier, f, and loss function, L, w.r.t. $\mathcal{D}_\eta$
- f_η^* – a global minimizer of $R_L^\eta(f)$

Problem Formulation (contd.)

- The subscript η denotes noisy data or distribution.

Problem Formulation (contd.)

- The subscript η denotes noisy data or distribution.
- But what's the relation between the corrupted or noisy ($y_{\boldsymbol{x}}$) and clean labels ($y_{\boldsymbol{x}}^{cl}$)?

Problem Formulation (contd.)

The $y_{\boldsymbol{x}}$ here are the corrupted or noisy labels and they are random variables dependent on the clean labels, $y_{\boldsymbol{x}}^{cl}$, through the following conditional probability relations:

$$\eta_{\boldsymbol{x},ij} = P(y_{\boldsymbol{x}} = j | \boldsymbol{x}, y_{\boldsymbol{x}}^{cl} = i) \ \forall \ j \in [K] \tag{3}$$

- These conditional probabilities are called noise rates.

Problem Formulation (contd.)

The $y_{\boldsymbol{x}}$ here are the corrupted or noisy labels and they are random variables dependent on the clean labels, $y_{\boldsymbol{x}}^{cl}$, through the following conditional probability relations:

$$\eta_{\boldsymbol{x},ij} = P(y_{\boldsymbol{x}} = j|\boldsymbol{x}, y_{\boldsymbol{x}}^{cl} = i) \; \forall \, j \in [K] \tag{3}$$

- These conditional probabilities are called noise rates.
- Based on these probabilities, we can define the noise rate matrix, N, where $N_{ij} = P(y_{\boldsymbol{x}} = j|\boldsymbol{x}, y_{\boldsymbol{x}}^{cl} = i) = \eta_{\boldsymbol{x},ij}$, which completely specifies the noise model.

Problem Formulation (contd.)

The $y_{\boldsymbol{x}}$ here are the corrupted or noisy labels and they are random variables dependent on the clean labels, $y_{\boldsymbol{x}}^{cl}$, through the following conditional probability relations:

$$\eta_{\boldsymbol{x},ij} = P(y_{\boldsymbol{x}} = j | \boldsymbol{x}, y_{\boldsymbol{x}}^{cl} = i) \; \forall \; j \in [K] \tag{3}$$

- These conditional probabilities are called noise rates.
- Based on these probabilities, we can define the noise rate matrix, N, where $N_{ij} = P(y_{\boldsymbol{x}} = j | \boldsymbol{x}, y_{\boldsymbol{x}}^{cl} = i) = \eta_{\boldsymbol{x},ij}$, which completely specifies the noise model.
- Note that:

$$\sum_{j \in [K]} \eta_{\boldsymbol{x},ij} = 1 \; \forall \; i \in [K], \; \forall \; x \tag{4}$$

Problem Formulation (contd.)

Label noise can be categorized into 3 types:

- **Symmetric/Uniform Label Noise (SLN):** The labels are uniformly flipped to any one of the remaining classes. That is, $\eta_{\boldsymbol{x},ij}$ do not depend on $\boldsymbol{x}$.

$$\eta_{\boldsymbol{x},ij} = \frac{\eta}{K-1} \, (\forall \, j \in [K], j \neq i) \tag{5}$$

$$\eta_{\boldsymbol{x},ii} = 1 - \eta \tag{6}$$

Problem Formulation (contd.)

Label noise can be categorized into 3 types:

- **Class-conditional Label Noise (CCLN):** This is, relative to symmetric label noise, a more realistic model of label noise. As per this model, the noise rates are dependent on the true labels ($y_{\boldsymbol{x}}^{cl}$'s) only .i.e.

$$\eta_{\boldsymbol{x},ij} = P(y_{\boldsymbol{x}} = j|\boldsymbol{x}, y_{\boldsymbol{x}}^{cl} = i) = P(y_{\boldsymbol{x}} = j|y_{\boldsymbol{x}}^{cl} = i) \tag{7}$$

Problem Formulation (contd.)

- **Non-Uniform label noise (NULN):** This is the most general case of label noise. As per this model, the noise rates are dependent on both the feature vectors ($\boldsymbol{x}$'s) and the true labels ($y_{\boldsymbol{x}}^{cl}$'s).

Problem Formulation (contd.)

- In case of label noise, we have access only to S_η and not S.

Problem Formulation (contd.)

- In case of label noise, we have access only to S_η and not S.
- So, for robust learning under label noise, the objective is to learn a classifier from noisy training data, S_η, such that the classifier performs "well" on data drawn from $\mathcal{D}$.

Problem Formulation (contd.)

- In case of label noise, we have access only to S_η and not S.
- So, for robust learning under label noise, the objective is to learn a classifier from noisy training data, S_η, such that the classifier performs "well" on data drawn from $\mathcal{D}$.
- We capture this idea of **robustness** through the *risk minimization* framework and call it **Robust Risk Minimization**.

Problem Formulation (contd.)

- Risk minimization for loss function, L, is said to be **robust** under label noise if [33]:

$$\text{Prob}_{\mathcal{D}}(\{pred \circ f^*(\boldsymbol{x}) = y_{\boldsymbol{x}}\}) = \text{Prob}_{\mathcal{D}}(\{pred \circ f^*_{\eta}(\boldsymbol{x}) = y_{\boldsymbol{x}}\})$$

Recall that

- f^* – a global minimizer of $R_L(f)$
- f^*_{η} – a global minimizer of $R^{\eta}_L(f)$

Problem Formulation (contd.)

- Risk minimization for loss function, L, is said to be **robust** under label noise if [33]:

$$\text{Prob}_{\mathcal{D}}(\{pred \circ f^*(\boldsymbol{x}) = y_{\boldsymbol{x}}\}) = \text{Prob}_{\mathcal{D}}(\{pred \circ f^*_{\eta}(\boldsymbol{x}) = y_{\boldsymbol{x}}\})$$

Recall that

- f^* – a global minimizer of $R_L(f)$
- f^*_{η} – a global minimizer of $R^{\eta}_L(f)$

- Note that the probability in the above definition is w.r.t. $\mathcal{D}$, the underlying (unknown) distribution, that chracterizes the clean or true labels.

Problem Formulation (contd.)

- Risk minimization for loss function, L, is said to be **robust** under label noise if [33]:

$$\text{Prob}_{\mathcal{D}}(\{pred \circ f^*(\boldsymbol{x}) = y_{\boldsymbol{x}}\}) = \text{Prob}_{\mathcal{D}}(\{pred \circ f^*_{\eta}(\boldsymbol{x}) = y_{\boldsymbol{x}}\}) \quad (8)$$

*more on this later

Problem Formulation (contd.)

- Risk minimization for loss function, L, is said to be **robust** under label noise if [33]:

$$\text{Prob}_{\mathcal{D}}(\{pred \circ f^{*}(\boldsymbol{x}) = y_{\boldsymbol{x}}\}) = \text{Prob}_{\mathcal{D}}(\{pred \circ f_{\eta}^{*}(\boldsymbol{x}) = y_{\boldsymbol{x}}\}) \quad (8)$$

- [33, 11] show that if loss functions satisfy the property of **symmetry***,then risk minimization for that loss function is **robust** (as defined above) for certain kinds of label noise.

*more on this later

Problem Formulation (contd.)

- Risk minimization for loss function, L, is said to be **robust** under label noise if [33]:

$$\text{Prob}_{\mathcal{D}}(\{pred \circ f^*(\boldsymbol{x}) = y_{\boldsymbol{x}}\}) = \text{Prob}_{\mathcal{D}}(\{pred \circ f^*_\eta(\boldsymbol{x}) = y_{\boldsymbol{x}}\}) \quad (8)$$

- [33, 11] show that if loss functions satisfy the property of **symmetry***,then risk minimization for that loss function is **robust** (as defined above) for certain kinds of label noise.
- A stronger notion of robustness could also be defined wherein a minimizer of $R^\eta_L(f)$ is required to also be a minimizer of $R_L(f)$.

*more on this later

Problem Formulation (contd.)

- Risk minimization for loss function, L, is said to be **robust** under label noise if [33]:

$$\text{Prob}_{\mathcal{D}}(\{pred \circ f^*(\boldsymbol{x}) = y_{\boldsymbol{x}}\}) = \text{Prob}_{\mathcal{D}}(\{pred \circ f^*_{\eta}(\boldsymbol{x}) = y_{\boldsymbol{x}}\}) \quad (8)$$

- [33, 11] show that if loss functions satisfy the property of **symmetry***,then risk minimization for that loss function is **robust** (as defined above) for certain kinds of label noise.
- A stronger notion of robustness could also be defined wherein a minimizer of $R_L^{\eta}(f)$ is required to also be a minimizer of $R_L(f)$.
- This idea of **robust risk minimization** will be explored further (in the context of neural networks) in the coming slides.

*more on this later

Literature Review

We briefly allude to the existing approaches for robustness under label noise via the following categorization:

- *Label filtering* algorithms[4, 19, 20, 50, 7, 14] attempt at identifying the samples that are likely to have incorrect labels. This is one of the oldest approaches and mainly incorporates removal of outliers based on *robust statistics* theory.

Literature Review

We briefly allude to the existing approaches for robustness under label noise via the following categorization:

- *Label filtering* algorithms[4, 19, 20, 50, 7, 14] attempt at identifying the samples that are likely to have incorrect labels. This is one of the oldest approaches and mainly incorporates removal of outliers based on *robust statistics* theory.
- *Label cleaning* algorithms [44] attempt at identifying and correcting the potentially incorrect labels through joint optimization of labels and network weights [45, 52] or treating true labels as latent variables and using EM-style algorithms to infer them [49]. Various heuristics such as using ensemble of classifiers [58] or entropy of the softmax output of a neural network [44] are also used to identify the most-likely correct label.

- *Loss correction* methods [37, 47, 16, 35] suitably modify loss function (or posterior probabilities) to correct for the effects of label noise on risk minimization such that minimizers of risk under the noisy distribution ($\mathcal{D}_\eta$) are same as those of risk under the clean distribution ($\mathcal{D}$); however, they need to know (or estimate) the noise rates.

- *Loss correction* methods [37, 47, 16, 35] suitably modify loss function (or posterior probabilities) to correct for the effects of label noise on risk minimization such that minimizers of risk under the noisy distribution ($\mathcal{D}_\eta$) are same as those of risk under the clean distribution ($\mathcal{D}$); however, they need to know (or estimate) the noise rates.
- *Robust Loss Function* based methods devise novel loss functions which are inherently robust (Equation 8) thereby enabling robust risk minimization in presence of label noise [11, 56, 33, 29, 23, 46, 5].

- *Regularization* methods devise regularizers that penalize the network parameters if the network starts overfitting to noisy data and direct the network to learn from clean data [1, 26, 30, 55, 40, 34, 27] instead.

† the notion of 'clean' or 'noisy' is user-defined

- *Regularization* methods devise regularizers that penalize the network parameters if the network starts overfitting to noisy data and direct the network to learn from clean data [1, 26, 30, 55, 40, 34, 27] instead.
- *Sample Reweighting* methods [18, 32, 28, 13, 53, 41, 43, 51] are one of the popular strategies for regularization to reduce overfitting to the noisy data. The idea is to optimize over a weighted loss wherein each sample's weight is adjusted such that the overfitting to noisy data is reduced and more weightage is given to the data that's believed to be 'clean' †.

† the notion of 'clean' or 'noisy' is user-defined

Sample Reweighting – A popular approach

- In the last few years, several algorithms proposed for robust learning are based on sample reweighting.

Sample Reweighting – A popular approach

- In the last few years, several algorithms proposed for robust learning are based on sample reweighting.
- Idea: Assign binary or real-valued weights to each sample in the training data and then minimize the weighted loss. When the weights are binary, we call it a sample selection algorithm.

Sample Reweighting – A popular approach

- In the last few years, several algorithms proposed for robust learning are based on sample reweighting.
- Idea: Assign binary or real-valued weights to each sample in the training data and then minimize the weighted loss. When the weights are binary, we call it a sample selection algorithm.
- This is done to reduce the influence of samples that are likely to have noisy labels thereby reducing overfitting to it.

Sample Reweighting – A popular approach

- In the last few years, several algorithms proposed for robust learning are based on sample reweighting.
- Idea: Assign binary or real-valued weights to each sample in the training data and then minimize the weighted loss. When the weights are binary, we call it a sample selection algorithm.
- This is done to reduce the influence of samples that are likely to have noisy labels thereby reducing overfitting to it.
- This idea is very similar to that of 'curriculum learning' wherein the objective is to find a sequencing of samples for training such that the network learns from 'easy' samples before learning from 'hard' ones.

Sample Reweighting – A popular approach

- This notion of 'easy'/'hard' is user-defined.

Sample Reweighting – A popular approach

- This notion of 'easy'/'hard' is user-defined.
- In the context of label noise, one can think of clean samples as the 'easy' ones and noisy samples as the 'hard' ones.

Sample Reweighting – A popular approach

- This notion of ‘easy’/‘hard’ is user-defined.
- In the context of label noise, one can think of clean samples as the ‘easy’ ones and noisy samples as the ‘hard’ ones.
- This is a plausible analogy as studies such as [3, 12, 31] show that neural networks, when trained on randomly-labelled data, seem to learn from clean data before overfitting to the noisy data.

Related Work

- Motivated by this, several strategies of 'curriculum learning' have been devised that aim to select (or give more weightage to) 'clean' samples for robustness against label noise [18, 13, 53, 51, 28].

Related Work

- Motivated by this, several strategies of ‘curriculum learning’ have been devised that aim to select (or give more weightage to) ‘clean’ samples for robustness against label noise [18, 13, 53, 51, 28].
- Many of these methods employ the heuristic of ‘small loss’ for sample selection: a fraction of small-loss valued samples are used for learning the network parameters.

Related Work

- Motivated by this, several strategies of ‘curriculum learning’ have been devised that aim to select (or give more weightage to) ‘clean’ samples for robustness against label noise [18, 13, 53, 51, 28].
- Many of these methods employ the heuristic of ‘small loss’ for sample selection: a fraction of small-loss valued samples are used for learning the network parameters.
- Co-Teaching [13] and Co-Teaching+ [53] cross-train two similar neural network with this ‘small loss’ trick’. However, we need to know the noise rates.

Related Work (contd.)

- MentorNet [18] uses an auxiliary network trained with a small set of clean validation data as a sample selection function.

Related Work (contd.)

- MentorNet [18] uses an auxiliary network trained with a small set of clean validation data as a sample selection function.
- [41, 43] propose meta-learning based adapative sample reweighting methods which involve an additional learning step to infer a mapping from loss values to weights of samples.

Related Work (contd.)

- MentorNet [18] uses an auxiliary network trained with a small set of clean validation data as a sample selection function.
- [41, 43] propose meta-learning based adapative sample reweighting methods which involve an additional learning step to infer a mapping from loss values to weights of samples.
- [41, 43, 18] need additional computing resources as well as extra data with clean labels. In addition, they have extra hyperparameters for learning the sample weighting function which also needs tuning.

Related Work (contd.)

- MentorNet [18] uses an auxiliary network trained with a small set of clean validation data as a sample selection function.
- [41, 43] propose meta-learning based adapative sample reweighting methods which involve an additional learning step to infer a mapping from loss values to weights of samples.
- [41, 43, 18] need additional computing resources as well as extra data with clean labels. In addition, they have extra hyperparameters for learning the sample weighting function which also needs tuning.
- [28] propose a loss function that can be constructed using a surrogate loss of 0–1 loss function. This resultant loss is a (binary) weighted sum of the chosen surrogate loss values such that it's minimized with respect to the fixed threshold that depends on the noise rate.

Related Work (contd.)

- All these methods, in effect, assume that one can assess whether or not an example has clean label based on some function of the loss value of that example.

Related Work (contd.)

- All these methods, in effect, assume that one can assess whether or not an example has clean label based on some function of the loss value of that example.
- However, loss value of any specific example is itself a function of the current state of learning and it evolves with epochs.

Related Work (contd.)

- All these methods, in effect, assume that one can assess whether or not an example has clean label based on some function of the loss value of that example.
- However, loss value of any specific example is itself a function of the current state of learning and it evolves with epochs.
- Loss values of even clean samples may change over a significant range during the course of learning.

Related Work (contd.)

- All these methods, in effect, assume that one can assess whether or not an example has clean label based on some function of the loss value of that example.
- However, loss value of any specific example is itself a function of the current state of learning and it evolves with epochs.
- Loss values of even clean samples may change over a significant range during the course of learning.
- Further, the loss values achievable by a network even on clean samples may be different for examples of different classes.

An adaptive curriculum

- In essence, existing methods require knowledge of noise rates, access to a extra data with clean labels or availability of high computation resources. Quite often, these are not available.

An adaptive curriculum

- In essence, existing methods require knowledge of noise rates, access to a extra data with clean labels or availability of high computation resources. Quite often, these are not available.
- Motivated by all these considerations, we propose a simple, adaptive curriculum based sample selection strategy called **BA**tch **RE**weighting (**BARE**).

An adaptive curriculum

- In essence, existing methods require knowledge of noise rates, access to a extra data with clean labels or availability of high computation resources. Quite often, these are not available.
- Motivated by all these considerations, we propose a simple, adaptive curriculum based sample selection strategy called **BA**tch **RE**weighting (**BARE**).
- The idea is to focus on the current state of learning, in a given mini-batch, for identifying the noisy labels in it.

An adaptive curriculum

- In essence, existing methods require knowledge of noise rates, access to a extra data with clean labels or availability of high computation resources. Quite often, these are not available.
- Motivated by all these considerations, we propose a simple, adaptive curriculum based sample selection strategy called **BA**tch **RE**weighting (**BARE**).
- The idea is to focus on the current state of learning, in a given mini-batch, for identifying the noisy labels in it.
- This is done by using the batch statistics of loss values for a mini-batch to compute the threshold for sample selection. We do not need any knowledge of noise rates at all.

An adaptive curriculum (contd.)

General curriculum can be viewed as minimization of a weighted loss [24, 18]:

$$\min_{\theta, \mathbf{w} \in [0,1]^m} \mathcal{L}_{\text{wtd}}(\theta, \mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} w_i \mathcal{L}(f(x_i; \theta), y_i) + G(\mathbf{w}) + \beta ||\theta||^2$$

where $G(\mathbf{w})$ represents the curriculum, $f(\cdot; \theta) \in \Delta^{K-1}$ ($\Delta^{K-1} \subset [0,1]^K$ is the probability simplex) is a classifier function parameterized by θ, and $\mathcal{L}(f(\cdot; \theta), \cdot)$ is a loss function. We use CCE loss here.

An adaptive curriculum (contd.)

- One simple choice for the curriculum is [24] $G(\mathbf{w}) = -\lambda||\mathbf{w}||_1, \ \lambda > 0$. Putting this in the above, omitting the regularization term and taking $l_i = \mathcal{L}(f(x_i;\theta), y_i)$, the optimization problem becomes

$$\begin{aligned}
\min_{\theta,\mathbf{w}\in[0,1]^m} \mathcal{L}_{\text{wtd}}(\theta, \mathbf{w}) &= \frac{1}{m}\sum_{i=1}^{m}(w_i l_i - \lambda w_i) \\
&= \sum_{i=1}^{m}(w_i l_i + (1 - w_i)\lambda) - m\lambda
\end{aligned}$$

An adaptive curriculum (contd.)

- One simple choice for the curriculum is [24] $G(\mathbf{w}) = -\lambda||\mathbf{w}||_1, \ \lambda > 0$. Putting this in the above, omitting the regularization term and taking $l_i = \mathcal{L}(f(x_i;\theta), y_i)$, the optimization problem becomes

$$\begin{aligned}
\min_{\theta, \mathbf{w} \in [0,1]^m} \mathcal{L}_{\text{wtd}}(\theta, \mathbf{w}) &= \frac{1}{m} \sum_{i=1}^{m} (w_i l_i - \lambda w_i) \\
&= \sum_{i=1}^{m} (w_i l_i + (1 - w_i)\lambda) - m\lambda
\end{aligned}$$

- Under the usual assumption that loss function is non-negative, for the above problem, the optimal $\mathbf{w}$ for any fixed θ is: $w_i = 1$ if $l_i < \lambda$ and $w_i = 0$ otherwise.

An adaptive curriculum (contd.)

- If we want an adaptive curriculum, we want λ to be dynamically adjusted based on the current state of learning. First, let us consider the case where we make λ depend on the class label. The optimization problem becomes

$$\begin{aligned}\min_{\theta,\mathbf{w}\in[0,1]^m} \mathcal{L}_{\text{wtd}}(\theta,\mathbf{w}) &= \frac{1}{m}\sum_{i=1}^{m}(w_i l_i - \lambda(y_i)w_i) \\ &= \frac{1}{m}\sum_{j=1}^{K}\sum_{i:y_i=e_j}(w_i l_i - \lambda_j w_i)\end{aligned}$$

$$= \sum_{j=1}^{K}\sum_{i:y_i=e_j}(w_i l_i + (1-w_i)\lambda_j) - \sum_{j=1}^{K}\sum_{i:y_i=e_j}\lambda_j$$

where $\lambda_j = \lambda(e_j)$.

An adaptive curriculum (contd.

- As is easy to see, the optimal w_i (for any fixed θ) are still given by the same relation: for an i with $y_i = e_j$, $w_i = 1$ when $l_i < \lambda_j$.

An adaptive curriculum (contd.

- As is easy to see, the optimal w_i (for any fixed θ) are still given by the same relation: for an i with $y_i = e_j$, $w_i = 1$ when $l_i < \lambda_j$.
- This relation for optimal w_i is true even if we make λ_j a function of θ and of all x_i with $y_i = e_j$.

An adaptive curriculum (contd.

- As is easy to see, the optimal w_i (for any fixed θ) are still given by the same relation: for an i with $y_i = e_j$, $w_i = 1$ when $l_i < \lambda_j$.
- This relation for optimal w_i is true even if we make λ_j a function of θ and of all x_i with $y_i = e_j$.
- Thus we can have a truly dynamically adaptive curriculum by making these λ_j depend on all x_i of that class in the mini-batch and the current θ.

An adaptive curriculum (contd.

- As is easy to see, the optimal w_i (for any fixed θ) are still given by the same relation: for an i with $y_i = e_j$, $w_i = 1$ when $l_i < \lambda_j$.
- This relation for optimal w_i is true even if we make λ_j a function of θ and of all x_i with $y_i = e_j$.
- Thus we can have a truly dynamically adaptive curriculum by making these λ_j depend on all x_i of that class in the mini-batch and the current θ.
- The next question is how we should decide or evolve these λ_j.

An adaptive curriculum (contd)

- As mentioned earlier, we want these λ_j's to be determined by the statistics of loss values in the mini-batch; equivalently statistics of posterior probabilities.

An adaptive curriculum (contd)

- As mentioned earlier, we want these λ_j's to be determined by the statistics of loss values in the mini-batch; equivalently statistics of posterior probabilities.
- So, we set the sample weights in the following manner:

$$w_i = \begin{cases} 1 & \text{if } f_{y_i}(x_i;\theta) \geq \frac{1}{|\mathcal{S}_{y_i}|} \sum_{s \in \mathcal{S}_{y_i}} f_{y_s}(x_s;\theta) + \sigma_{y_i} \\ 0 & \text{else} \end{cases} \tag{9}$$

where $\mathcal{S}_{y_i} = \{k \in [m] \mid y_k = y_i\}$ and σ_{e_j} indicates the sample variance of the class posterior probabilities for class-j in the given mini-batch.

BARE - The Algorithm

Keeping in mind that the neural networks are trained in a mini-batch manner, the BARE algorithm consists of three parts:

- Computing sample selection threshold, $T_{K \times 1}$, for a given mini-batch of data (Equation 9)

BARE - The Algorithm

Keeping in mind that the neural networks are trained in a mini-batch manner, the BARE algorithm consists of three parts:

- Computing sample selection threshold, $T_{K\times 1}$, for a given mini-batch of data (Equation 9)
- Sample selection based on this threshold as per Equation 9

BARE - The Algorithm

Keeping in mind that the neural networks are trained in a mini-batch manner, the BARE algorithm consists of three parts:

- Computing sample selection threshold, $T_{K\times 1}$, for a given mini-batch of data (Equation 9)
- Sample selection based on this threshold as per Equation 9
- Parameter updation using these selected samples

Experimental Setup

Datasets:

- MNIST [25] (– No data augmentation)
- CIFAR-10 [22] (– random cropping with size 4 padding and random horizontal flips)
- Clothing-1M [49] (– random cropping while ensuring fixed image size)

Table 1: Dataset details

	TRAIN SIZE	TEST SIZE	# CLASS	SIZE
MNIST	60,000	10,000	10	28×28
CIFAR-10	50,000	10,000	10	32×32
CLOTHING-1M	10,00,000	10,000	14	224×224

Experimental Setup (contd.)

Baselines: We compare the proposed algorithm with the following algorithms from literature:

- Co-Teaching (CoT) [13] which involves cross-training of two similar networks by selecting samples using a loss threshold based on (estimated) noise rates;

Experimental Setup (contd.)

Baselines: We compare the proposed algorithm with the following algorithms from literature:

- Co-Teaching (CoT) [13] which involves cross-training of two similar networks by selecting samples using a loss threshold based on (estimated) noise rates;
- Co-Teaching+ (CoT+) [53] which improves upon CoT with the difference that samples are selected from those upon which the two networks' predictions disagree;

Experimental Setup (contd.)

Baselines: We compare the proposed algorithm with the following algorithms from literature:

- Co-Teaching (CoT) [13] which involves cross-training of two similar networks by selecting samples using a loss threshold based on (estimated) noise rates;
- Co-Teaching+ (CoT+) [53] which improves upon CoT with the difference that samples are selected from those upon which the two networks' predictions disagree;
- Meta-Ren (MR) [41], which involves meta-learning of sample weights on-the-fly;

Experimental Setup (contd.)

Baselines: We compare the proposed algorithm with the following algorithms from literature:

- Co-Teaching (CoT) [13] which involves cross-training of two similar networks by selecting samples using a loss threshold based on (estimated) noise rates;
- Co-Teaching+ (CoT+) [53] which improves upon CoT with the difference that samples are selected from those upon which the two networks' predictions disagree;
- Meta-Ren (MR) [41], which involves meta-learning of sample weights on-the-fly;
- Meta-Net (MN) [43], which improves upon MR by explicitly learning sample weights via a separate neural network;

Experimental Setup (contd.)

Baselines: We compare the proposed algorithm with the following algorithms from literature:

- Co-Teaching (CoT) [13] which involves cross-training of two similar networks by selecting samples using a loss threshold based on (estimated) noise rates;
- Co-Teaching+ (CoT+) [53] which improves upon CoT with the difference that samples are selected from those upon which the two networks' predictions disagree;
- Meta-Ren (MR) [41], which involves meta-learning of sample weights on-the-fly;
- Meta-Net (MN) [43], which improves upon MR by explicitly learning sample weights via a separate neural network;
- Curriculum Loss (CL) [28], which involves a curriculum for sample selection based on (estimated) noise rates;

Experimental Setup (contd.)

Baselines: We compare the proposed algorithm with the following algorithms from literature:

- Co-Teaching (CoT) [13] which involves cross-training of two similar networks by selecting samples using a loss threshold based on (estimated) noise rates;
- Co-Teaching+ (CoT+) [53] which improves upon CoT with the difference that samples are selected from those upon which the two networks' predictions disagree;
- Meta-Ren (MR) [41], which involves meta-learning of sample weights on-the-fly;
- Meta-Net (MN) [43], which improves upon MR by explicitly learning sample weights via a separate neural network;
- Curriculum Loss (CL) [28], which involves a curriculum for sample selection based on (estimated) noise rates;
- Standard (CCE), which is the usual training through empirical risk minimization with cross-entropy loss (using the data with noisy labels).

Experimental Setup (contd.)

- CoT, CoT+, and CL are sample selection algorithms that require knowledge of noise rates. CoT+ and CL require a warm-up period. We use 5 epochs and 10 epochs as warm up period during training for MNIST and CIFAR-10 respectively.
- For all the datasets, 80% of the training set is used for training and, from the remaining 20% data, we sample 1000 images that constitute the validation set.
- MR and MN assume access to a small set of clean validation data. Because of this, and for a fair comparison among all the baselines, a clean validation set of 1000 samples is used in case of MR and MN, and the same set of samples but with the noisy labels is used for the rest of the algorithms including the proposed one.

Experimental Setup (contd.)

Types of Label Noise simulated

- Symmetric Label Noise

Experimental Setup (contd.)

Types of Label Noise simulated

- Symmetric Label Noise
- Class-conditional Label Noise
  - For MNIST, the following flipping is done: $1 \leftarrow 7$, $2 \rightarrow 7$, $3 \rightarrow 8$, and $5 \leftrightarrow 6$
  - For CIFAR10, the following flipping is done: TRUCK $\rightarrow$ AUTOMOBILE, BIRD $\rightarrow$ AIRPLANE, DEER $\rightarrow$ HORSE, CAT $\leftrightarrow$ DOG

Experimental Setup (contd.)

Network architectures and optimizers

- For MNIST: 1-hidden layer fully-connected network with Adam (learning rate $= 2 \times 10^{-4}$ and a learning rate scheduler: ReduceLROnPlateau)

Experimental Setup (contd.)

Network architectures and optimizers

- For MNIST: 1-hidden layer fully-connected network with Adam (learning rate $= 2 \times 10^{-4}$ and a learning rate scheduler: ReduceLROnPlateau)
- For CIFAR10: 4-layer CNN with Adam [21] (learning rate $= 2 \times 10^{-3}$ and a learning rate scheduler: ReduceLROnPlateau)

Experimental Setup (contd.)

Network architectures and optimizers

- For MNIST: 1-hidden layer fully-connected network with Adam (learning rate $= 2 \times 10^{-4}$ and a learning rate scheduler: ReduceLROnPlateau)
- For CIFAR10: 4-layer CNN with Adam [21] (learning rate $= 2 \times 10^{-3}$ and a learning rate scheduler: ReduceLROnPlateau)
- All networks are trained for 200 epochs.

Experimental Setup (contd.)

Network architectures and optimizers

- For MNIST: 1-hidden layer fully-connected network with Adam (learning rate $= 2 \times 10^{-4}$ and a learning rate scheduler: ReduceLROnPlateau)
- For CIFAR10: 4-layer CNN with Adam [21] (learning rate $= 2 \times 10^{-3}$ and a learning rate scheduler: ReduceLROnPlateau)
- All networks are trained for 200 epochs.
- For MR, SGD optimizer with momentum 0.9 and learning rate of 1×10^{-3} is used as the meta-optimizer. For MN, SGD optimizer with learning rate of 2×10^{-3} is used as meta-optimizer.

Experimental Setup (contd.)

Network architectures and optimizers

- For MNIST: 1-hidden layer fully-connected network with Adam (learning rate $= 2 \times 10^{-4}$ and a learning rate scheduler: ReduceLROnPlateau)
- For CIFAR10: 4-layer CNN with Adam [21] (learning rate $= 2 \times 10^{-3}$ and a learning rate scheduler: ReduceLROnPlateau)
- All networks are trained for 200 epochs.
- For MR, SGD optimizer with momentum 0.9 and learning rate of 1×10^{-3} is used as the meta-optimizer. For MN, SGD optimizer with learning rate of 2×10^{-3} is used as meta-optimizer.
- For Clothing-1M: pre-trained ResNet-50 with SGD (learning rate of 1×10^{-3} that is halved at epochs 6 and 11) with a weight decay of 1×10^{-3} and momentum 0.9 for 14 epochs.

Network Architectures

Table 2: Network Architectures used for training on MNIST and CIFAR-10 datasets

MNIST	CIFAR-10
11*DENSE $28\times28 \rightarrow 256$	3×3 CONV., 64 ReLU, STRIDE 1, PADDING 1
	BATCH NORMALIZATION
	2×2 MAX POOLING, STRIDE 2
	3×3 CONV., 128 ReLU, STRIDE 1, PADDING 1
	BATCH NORMALIZATION
	2×2 MAX POOLING, STRIDE 2
	3×3 CONV., 196 ReLU, STRIDE 1, PADDING 1
	BATCH NORMALIZATION
	3×3 CONV., 16 ReLU, STRIDE 1, PADDING 1
	BATCH NORMALIZATION
	2×2 MAX POOLING, STRIDE 2
DENSE $256 \rightarrow 10$	DENSE $256 \rightarrow 10$

Other Training Details

- For CL, soft hinge loss is used as suggested in [28] instead of cross-entropy loss. Rest of the algorithms implemented for this paper use cross-entropy loss.

Other Training Details

- For CL, soft hinge loss is used as suggested in [28] instead of cross-entropy loss. Rest of the algorithms implemented for this paper use cross-entropy loss.
- All the simulations are run for 5 trials.

Other Training Details

- For CL, soft hinge loss is used as suggested in [28] instead of cross-entropy loss. Rest of the algorithms implemented for this paper use cross-entropy loss.
- All the simulations are run for 5 trials.
- All experiments use PyTorch [36], NumPy [15], scikit-learn [38], and NVIDIA Titan X Pascal GPU with CUDA 10.0.

Performance Metrics

- For all algorithms we compare **test accuracies** on a separate test set with clean labels.

Performance Metrics

- For all algorithms we compare **test accuracies** on a separate test set with clean labels.
- The main idea in all sample selection schemes is to identify noisy labels. Hence, in addition to test accuracies, we also compare:

Performance Metrics

- For all algorithms we compare **test accuracies** on a separate test set with clean labels.
- The main idea in all sample selection schemes is to identify noisy labels. Hence, in addition to test accuracies, we also compare:
  - **precision** (# *clean labels selected* / # *of selected labels*)

Performance Metrics

- For all algorithms we compare **test accuracies** on a separate test set with clean labels.
- The main idea in all sample selection schemes is to identify noisy labels. Hence, in addition to test accuracies, we also compare:
  - **precision** (# *clean labels selected* / # *of selected labels*)
  - **recall** (# *clean labels selected* / # *of clean labels*)

Results on MNIST - Test Accuracy

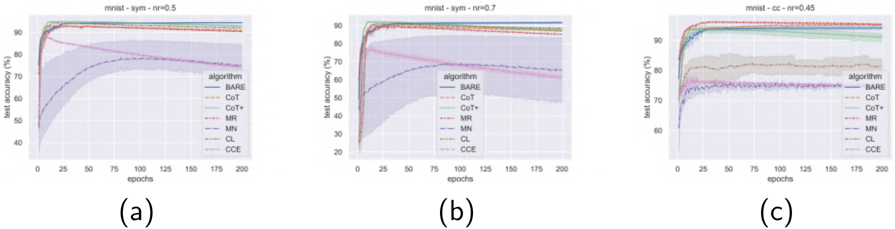
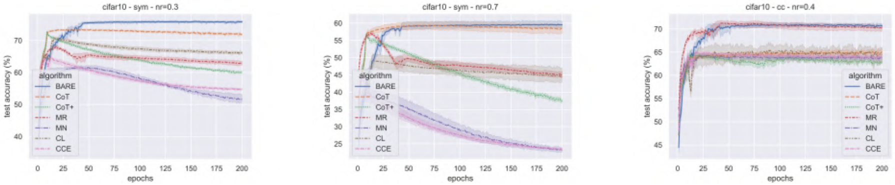


Figure 1: Test Accuracies - MNIST - Symmetric ((a) & (b)) & Class-conditional ((c)) Label Noise

- BARE outperforms the baselines for symmetric noise.
- For class-conditional noise, the test accuracy of BARE is marginally less than the best of the baselines, namely CoT and MR.

Results on CIFAR10 - Test Accuracy



(a) (b) (c)

Figure 2: Test Accuracies - CIFAR10 - Symmetric ((a) & (b)) & Class-conditional ((c)) Label Noise

- BARE outperforms the baseline schemes and its test accuracies are uniformly good for all types of label noise.

Some comments

- Test accuracies for BARE stays saturated after attaining maximum performance whereas for baselines, there's an accuracy dip towards end of training. This suggests that BARE doesn't let the network overfit even after long durations of training unlike the baselines.

Some comments

- Test accuracies for BARE stays saturated after attaining maximum performance whereas for baselines, there's an accuracy dip towards end of training. This suggests that BARE doesn't let the network overfit even after long durations of training unlike the baselines.
- All baselines have hyperparameters and the accuracies reported here are for the best possible hyperparameter values obtained through tuning. The MR and MN algorithms are particularly sensitive to hyperparameter values in the meta learning algorithm. In contrast, BARE has no hyperparameters for the sample selection and hence no such tuning is involved.

Results on Clothing-1M – Test Accuracy

Table 3: Test accuracies on Clothing-1M dataset

Algorithm	Test Accuracy (%)
CCE	68.94
D2L [31]	69.47
GCE [56]	69.75
Forward [37]	69.84
CoT [13]†	70.15
SEAL [6]	70.63
DY [2]	71.00
SCE [46]	71.02
LRT [57]	71.74
PTD-R-V [48]	71.67
Joint Opt. [45]	72.23
BARE (Ours)	**72.28**
DivideMix [26]	74.76

†as reported in [6]

Results on Clothing-1M – Test Accuracy (contd.)

- Even for real-world noisy datasets such as Clothing-1M where the label noise that isn't synthetic unlike the symmetric and class-conditional label noise used for aforementioned simulations, BARE performs better than all but one baselines.
- However, it is to be noted that DivideMix requires about 2.4 times the computation time required for BARE. In addition to this, DivideMix requires tuning of 5 hyperparameters whereas no such tuning is required for BARE.

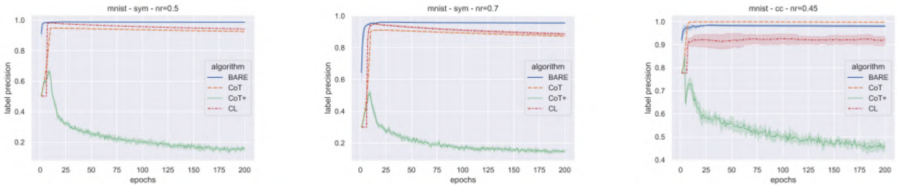
Efficiency of BARE

- Table 4 shows the typical run times for 200 epochs of training with all the algorithms.
- BARE takes roughly the same time as the usual training with CCE loss. Other baselines are significantly more expensive computationally. For MR and MN, the run times are around 8 times that of BARE for CIFAR-10.

Table 4: Algorithm run times for training (in seconds)

Algorithm	MNIST	CIFAR10
BARE	**310.64**	**930.78**
CoT	504.5	1687.9
CoT+	537.7	1790.57
MR	807.4	8130.87
MN	1138.4	8891.6
CL	730.15	1254.3
CCE	**229.27**	**825.68**

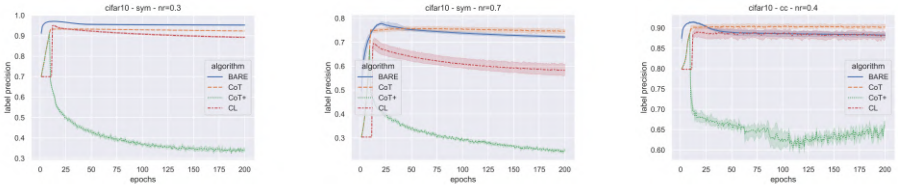
Efficacy of detecting clean samples - Label Precision



(a) (b) (c)

Figure 3: Label Precision - MNIST - Symmetric ((a) & (b)) & Class-conditional ((c)) Label Noise

Efficacy of detecting clean samples - Label Precision

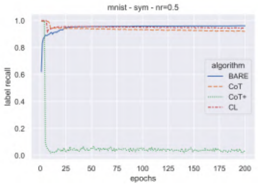


(a) (b) (c)

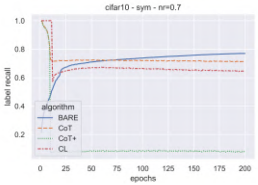
Figure 4: Label Precision - CIFAR10 - Symmetric ((a) & (b)) & Class-conditional ((c)) Label Noise

- Figures 3 and 4 show the label precision (across epochs) on MNIST and CIFAR-10 respectively.
- BARE has comparable or better precision.

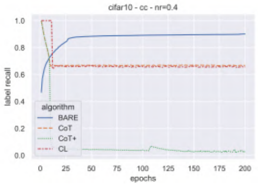
Efficiency of detecting clean samples - Label Recall



(a)



(b)



(c)

Figure 5: Label Recall - Symmetric ((a) & (b)) & Class-conditional ((c)) Label Noise

- Figure 5 show the label recall values for CoT, CoT+, CL, and BARE for MNIST (5(a)) and CIFAR-10 (5(b) & 5(c)).

Some comments

- BARE consistently achieves better recall values compared to the baselines. Higher recall values indicate that the algorithm is able to identify clean samples more reliably.

Some comments

- BARE consistently achieves better recall values compared to the baselines. Higher recall values indicate that the algorithm is able to identify clean samples more reliably.
- This is useful, for example, to employ a label cleaning algorithm on the samples flagged as noisy (i.e., not selected) by BARE.

Some comments

- BARE consistently achieves better recall values compared to the baselines. Higher recall values indicate that the algorithm is able to identify clean samples more reliably.
- This is useful, for example, to employ a label cleaning algorithm on the samples flagged as noisy (i.e., not selected) by BARE.
- CoT+ selects a fraction of samples where two networks disagree and, hence, after the first few epochs, it selects very few samples (~ 3000) in each epoch. Since these are samples in which the networks disagree, a good fraction of them may have noisy labels.

Some comments

- BARE consistently achieves better recall values compared to the baselines. Higher recall values indicate that the algorithm is able to identify clean samples more reliably.
- This is useful, for example, to employ a label cleaning algorithm on the samples flagged as noisy (i.e., not selected) by BARE.
- CoT+ selects a fraction of samples where two networks disagree and, hence, after the first few epochs, it selects very few samples ($\sim$ 3000) in each epoch. Since these are samples in which the networks disagree, a good fraction of them may have noisy labels.
- This may be the reason for the poor precision and recall values of CoT+ as seen in these figures.

Some comments

- BARE consistently achieves better recall values compared to the baselines. Higher recall values indicate that the algorithm is able to identify clean samples more reliably.
- This is useful, for example, to employ a label cleaning algorithm on the samples flagged as noisy (i.e., not selected) by BARE.
- CoT+ selects a fraction of samples where two networks disagree and, hence, after the first few epochs, it selects very few samples (~ 3000) in each epoch. Since these are samples in which the networks disagree, a good fraction of them may have noisy labels.
- This may be the reason for the poor precision and recall values of CoT+ as seen in these figures.
- In terms of fraction of samples selected also, BARE does better.

Sensitivity to noise rates

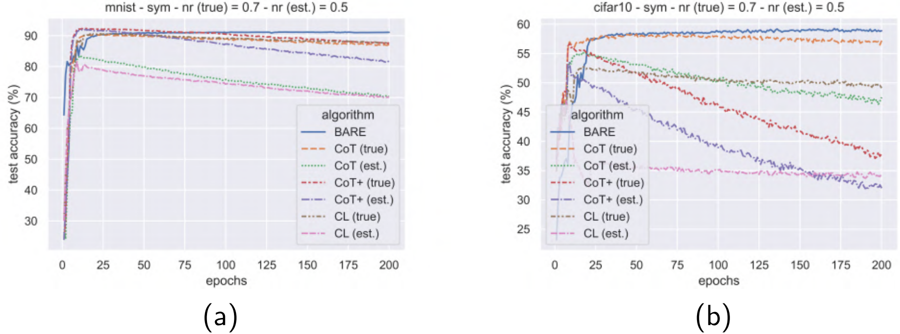


Figure 6: ((a) & (b)):Test accuracies when estimated (symmetric) noise rate, $\eta = 0.5$, and true noise rate, $\eta = 0.7$, for MNIST & CIFAR-10 resp.

- So, similar performance trends are seen for the case of mis-specified noise rates and arbitrary noise rate matrices.

BARE - Summary

- We looked at an adaptive sample selection strategy that provides robustness under label noise by relying only on batch statistics of posterior values in a mini-batch and requires no hyperparameter tuning.

BARE - Summary

- We looked at an adaptive sample selection strategy that provides robustness under label noise by relying only on batch statistics of posterior values in a mini-batch and requires no hyperparameter tuning.
- Empirical studies shows that the proposed algorithm performs better or as well as the baselines under label noise.

‘Memorization’ in Deep networks

- Deep neural networks are very good at interpolating the data.

‘Memorization’ in Deep networks

- Deep neural networks are very good at interpolating the data.
- Zhang et al. [54] showed that SGD-based training of neural networks drives the training accuracy to 100% even in case of randomly labelled data.

'Memorization' in Deep networks

- Deep neural networks are very good at interpolating the data.
- Zhang et al. [54] showed that SGD-based training of neural networks drives the training accuracy to 100% even in case of randomly labelled data.
- They call this 'memorization' as the network overfits to the training data. (However, there is no precise definition in the paper).

‘Memorization’ in Deep networks

- Deep neural networks are very good at interpolating the data.
- Zhang et al. [54] showed that SGD-based training of neural networks drives the training accuracy to 100% even in case of randomly labelled data.
- They call this ‘memorization’ as the network overfits to the training data. (However, there is no precise definition in the paper).
- None of the standard regularization methods such as weight decay, drop-out, etc. seem effective in resisting such overfitting.

‘Memorization’ in Deep networks

- Deep neural networks are very good at interpolating the data.
- Zhang et al. [54] showed that SGD-based training of neural networks drives the training accuracy to 100% even in case of randomly labelled data.
- They call this ‘memorization’ as the network overfits to the training data. (However, there is no precise definition in the paper).
- None of the standard regularization methods such as weight decay, drop-out, etc. seem effective in resisting such overfitting.
- Many other studies (e.g., [3, 12, 9, 10]) throw interesting light on the dynamics of this memorization process and what it means for generalization in deep networks

Our Study

- Here, we study the effect of loss function on degree of memorization

Our Study

- Here, we study the effect of loss function on degree of memorization
- Memorization is essentially about training error.

Our Study

- Here, we study the effect of loss function on degree of memorization
- Memorization is essentially about training error.
- It is about: Can the network (always) learn a function that perfectly interpolates data?

Our Study

- Here, we study the effect of loss function on degree of memorization
- Memorization is essentially about training error.
- It is about: Can the network (always) learn a function that perfectly interpolates data?
- Depends on the kind of local minima that SGD process can take the network to

Our Study

- Here, we study the effect of loss function on degree of memorization
- Memorization is essentially about training error.
- It is about: Can the network (always) learn a function that perfectly interpolates data?
- Depends on the kind of local minima that SGD process can take the network to
- Depends on the topography of empirical risk that is minimized.

Our Study

- Here, we study the effect of loss function on degree of memorization
- Memorization is essentially about training error.
- It is about: Can the network (always) learn a function that perfectly interpolates data?
- Depends on the kind of local minima that SGD process can take the network to
- Depends on the topography of empirical risk that is minimized.
- The choice of loss function can be critical in determining this.

Our Study

- Here, we study the effect of loss function on degree of memorization
- Memorization is essentially about training error.
- It is about: Can the network (always) learn a function that perfectly interpolates data?
- Depends on the kind of local minima that SGD process can take the network to
- Depends on the topography of empirical risk that is minimized.
- The choice of loss function can be critical in determining this.
- None of the studies on memorization investigate this.

Our Results

- Choice of loss function can affect memorization

Our Results

- Choice of loss function can affect memorization
- We show empirically that a symmetric loss function can resist memorization to a good degree

Our Results

- Choice of loss function can affect memorization
- We show empirically that a symmetric loss function can resist memorization to a good degree
- We formally define what 'resisting memorization' means and provide some theoretical justification for the empirical results

Some Comments

- There are many studies on the relative efficacy of different loss functions for classification and regression (e.g., [17, 8, 39, 42])

Some Comments

- There are many studies on the relative efficacy of different loss functions for classification and regression (e.g., [17, 8, 39, 42])
- Our study is distinct – what role loss function can play in affecting the degree of memorization in overparameterized networks?

Some Comments

- There are many studies on the relative efficacy of different loss functions for classification and regression (e.g., [17, 8, 39, 42])
- Our study is distinct – what role loss function can play in affecting the degree of memorization in overparameterized networks?
- Design of algorithms for robust learning when training data has randomly corrupted labels, is also a much studied problem

Some Comments

- There are many studies on the relative efficacy of different loss functions for classification and regression (e.g., [17, 8, 39, 42])
- Our study is distinct – what role loss function can play in affecting the degree of memorization in overparameterized networks?
- Design of algorithms for robust learning when training data has randomly corrupted labels, is also a much studied problem
- Here our interest is in the inherent ability of a loss function to resist overfitting of training data when labels are randomly altered.

Notation

- $\mathcal{X} \subseteq \mathbb{R}^n$: feature space;
- $\mathcal{Y} = \{1, \ldots, K\}$ where K: number of classes
- $S = \{\mathbf{x}_i, y_i^{cl}\}_{i=1}^{\ell}$: Original training set
- $S_\eta = \{\mathbf{x}_i, y_i\}_{i=1}^{\ell}$: Training set with randomly altered labels:

$$y_i = \begin{cases} y_i^{cl} & \text{with probability } 1 - \eta \\ j \in \mathcal{Y} - \{y_i^{cl}\} & \text{with probability } \frac{\eta}{K-1} \end{cases} \tag{10}$$

  where η is referred to as the noise rate.
- h_η: classifier function (with softmax layer as output layer) learnt by an algorithm with S_η as training data
- To define loss functions we take y_i to be one-hot vector. ($\mathbf{e}^k$ represents class-k).

Notation (contd.)

- Let $J_1 = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathbb{I}_{[h_\eta(\boldsymbol{x}_i)=y_i]}$
- This is the usual training accuracy of classifier h_η on the training set with randomly altered labels.

Notation (contd.)

- Let $J_1 = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathbb{I}_{[h_\eta(\boldsymbol{x}_i)=y_i]}$
- This is the usual training accuracy of classifier h_η on the training set with randomly altered labels.
- Let $J_2 = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathbb{I}_{[h_\eta(\boldsymbol{x}_i)=y_i^{cl}]}$
- This is the accuracy of h_η on the same training set but computed with respect to the original labels.

Notation (contd.)

- Let $J_1 = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathbb{I}_{[h_\eta(\mathbf{x}_i)=y_i]}$
- This is the usual training accuracy of classifier h_η on the training set with randomly altered labels.
- Let $J_2 = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathbb{I}_{[h_\eta(\mathbf{x}_i)=y_i^{cl}]}$
- This is the accuracy of h_η on the same training set but computed with respect to the original labels.
- Relative values of J_1 and J_2 can give interesting insights into how different loss functions behave.

Loss Functions We Compare

- We consider 3 loss functions in this paper: CCE, MSE, and Robust Log Loss (RLL).

$$
\begin{aligned}
\mathcal{L}_{CCE}(\mathbf{h}(\mathbf{x}), \mathbf{e}^k) &= -\sum_i e_i^k \log\left(h_i(\mathbf{x})\right) = -\log(h_k(\mathbf{x})) \\
\mathcal{L}_{MSE}(\mathbf{h}(\mathbf{x}), \mathbf{e}^k) &= \sum_i \left(h_i(\mathbf{x}) - e_i^k\right)^2 \\
\mathcal{L}_{RLL}(\mathbf{h}(\mathbf{x}), \mathbf{e}^k) &= \log\left(\frac{\alpha+1}{\alpha}\right) - \log(\alpha + h_k(\mathbf{x})) \\
&\quad + \sum_{j \neq k} \frac{1}{K-1} \log(\alpha + h_j(\mathbf{x}))
\end{aligned}
$$

where $\alpha > 0$ is a parameter of the RLL.

Loss Functions (contd.)

- CCE and MSE are fairly commonly used loss functions for classification and regression tasks.
- RLL is a **symmetric** loss.

Symmetric Loss

A loss function, L, is **symmetric** if $\exists\ C \in \mathbb{R}$ such that:

$$\sum_{j=1}^{K} L(h(\boldsymbol{x}), j) = C, \ \forall h, \boldsymbol{x} \tag{11}$$

We next present some empirical results comparing these three losses in their ability to resist memorization

Experimental Setup

Datasets:

- MNIST [25]
- CIFAR-10 [22]

Networks:

- Inception-Lite (same as that used in [54]) for CIFAR-10
- ResNet-32 for CIFAR-10
- ResNet-18 for MNIST

Experimental Setup (contd.)

Details about network training:

Inception-Lite

- SGD(lr $= 10^{-2}$, momentum=0.9)
- number of epochs = 100
- learning rate reduced by 0.95 factor every epoch

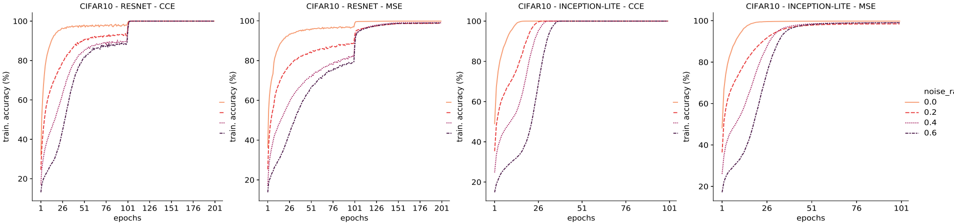
ResNet-32

- SGD(lr $= 10^{-1}$, momentum=0.9, weight_decay=10^{-4})
- number of epochs = 200
- learning rate reduced by 0.1 factor at epochs 100 and 150

ResNet-18

- Adam(lr $= 10^{-3}$)
- number of epochs = 200

Results on CIFAR-10

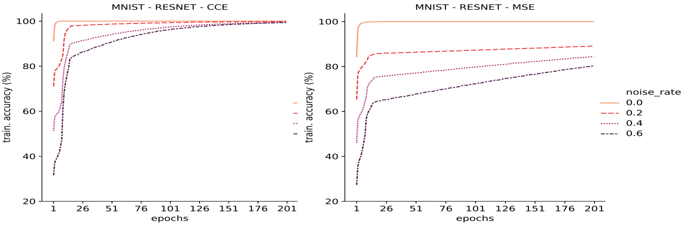


(a) *CCE* **(b)** *MSE* **(c)** *CCE* **(d)** *MSE*

Figure 7: Training set accuracies for ResNet-32 ((a) & (b)) & Inception-Lite ((c) & (d)) trained on CIFAR-10 with CCE and MSE losses for for $\eta \in \{0., 0.2, 0.4, 0.6\}$

- CCE and MSE achieve 100% training accuracy (irrespective of noise rate) thus showing they memorize random labels.

Results on MNIST

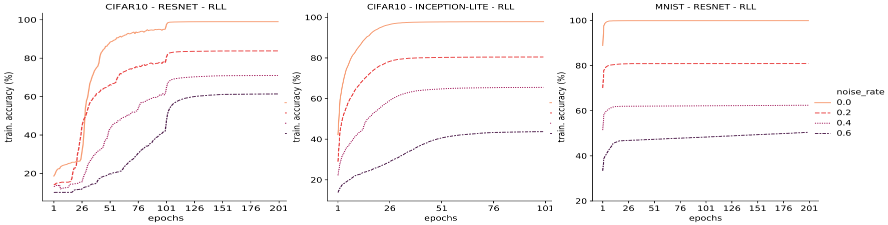


(a) *CCE* **(b)** *MSE*

Figure 8: Training set accuracies for ResNet-18 trained on MNIST with CCE and MSE losses for $\eta \in \{0., 0.2, 0.4, 0.6\}$

- CCE overfits even with this smaller network. MSE also achieves high trainig accuracy irrespective of amount of noise.

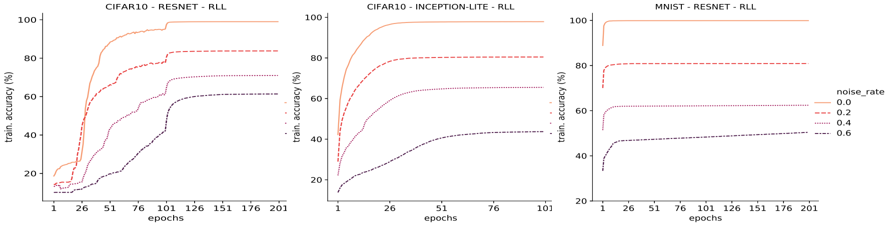
Results on CIFAR-10 & MNIST - with RLL



(a) $ResNet - 32$ **(b)** $Inception - Lite$ **(c)** $ResNet - 18$

Figure 9: Training set accuracies for networks trained on CIFAR-10 ((a) & (b)) and MNIST ((a)) with RLL for $\eta \in \{0., 0.2, 0.4, 0.6\}$

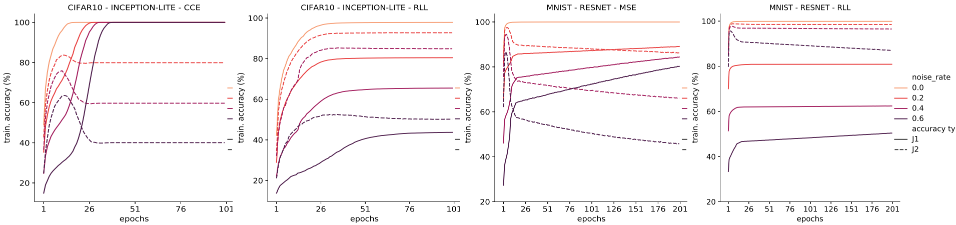
Results on CIFAR-10 & MNIST - with RLL



(a) $ResNet-32$ **(b)** $Inception-Lite$ **(c)** $ResNet-18$

- Training accuracies of RLL saturate much below 100% – the higher the noise rate the lower the training accuracy
- We can see from Figure 10 that the training accuracy of RLL saturates to almost $(1-\eta)\times 100\%$ (η :– noise rate). Now the network does not overfit. In addition, it is as if we have almost inferred the noise rate!

Results for J_1 and J_2 accuracy - with RLL



(a) *CCE* **(b)** *RLL* **(c)** *MSE* **(d)** *RLL*

Figure 11: J_1 and J_2 accuracies for $\eta \in \{0., 0.2, 0.4, 0.6\}$ (Solid lines show J_1 accuracy; dashed lines show J_2 accuracy)

- For RLL, the J_2 is always above J_1 curve – showing RLL resists overfitting to noisy labels
- For CCE and MSE, the J_1 curve eventually goes above J_2 – showing the network overfits the noisy labels as epochs progress

Resisting Memorization:

- Recall: S is the original training data and S_η is the data with randomly altered labels; $\mathcal{D}$ and $\mathcal{D}_\eta$ are the corresponding distributions.
- Let h and h_η denote the classifier learned by an algorithm when given S and S_η as training data, respectively.

Resisting Memorization:

- Recall: S is the original training data and S_η is the data with randomly altered labels; $\mathcal{D}$ and $\mathcal{D}_\eta$ are the corresponding distributions.
- Let h and h_η denote the classifier learned by an algorithm when given S and S_η as training data, respectively.
- We say that an algorithm **resists memorization** if

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{I}_{\{h(\boldsymbol{x}_i)=y_i^{cl}\}} = \frac{1}{m}\sum_{i=1}^{m}\mathbb{I}_{\{h_\eta(\boldsymbol{x}_i)=y_i^{cl}\}} \tag{12}$$

Resisting Memorization:

- Recall: S is the original training data and S_η is the data with randomly altered labels; $\mathcal{D}$ and $\mathcal{D}_\eta$ are the corresponding distributions.
- Let h and h_η denote the classifier learned by an algorithm when given S and S_η as training data, respectively.
- We say that an algorithm **resists memorization** if

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{I}_{\{h(\boldsymbol{x}_i)=y_i^{cl}\}} = \frac{1}{m}\sum_{i=1}^{m}\mathbb{I}_{\{h_\eta(\boldsymbol{x}_i)=y_i^{cl}\}} \tag{12}$$

- This is a reasonable formalization for 'resisting memorization'

Resisting Memorization:

- Recall: S is the original training data and S_η is the data with randomly altered labels; $\mathcal{D}$ and $\mathcal{D}_\eta$ are the corresponding distributions.
- Let h and h_η denote the classifier learned by an algorithm when given S and S_η as training data, respectively.
- We say that an algorithm **resists memorization** if

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{I}_{\{h(\boldsymbol{x}_i)=y_i^{cl}\}} = \frac{1}{m}\sum_{i=1}^{m}\mathbb{I}_{\{h_\eta(\boldsymbol{x}_i)=y_i^{cl}\}} \tag{12}$$

- This is a reasonable formalization for 'resisting memorization'
- Note that the J_2 accuracy defined earlier is the quantity on RHS above. The accuracy for $\eta = 0$ would be the quantity on LHS above.

Resisting Memorization:

- Recall: S is the original training data and S_η is the data with randomly altered labels; $\mathcal{D}$ and $\mathcal{D}_\eta$ are the corresponding distributions.
- Let h and h_η denote the classifier learned by an algorithm when given S and S_η as training data, respectively.
- We say that an algorithm **resists memorization** if

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{I}_{\{h(\boldsymbol{x}_i)=y_i^{cl}\}} = \frac{1}{m}\sum_{i=1}^{m}\mathbb{I}_{\{h_\eta(\boldsymbol{x}_i)=y_i^{cl}\}} \tag{12}$$

- This is a reasonable formalization for 'resisting memorization'
- Note that the J_2 accuracy defined earlier is the quantity on RHS above. The accuracy for $\eta = 0$ would be the quantity on LHS above.
- For RLL, as we saw, the J_2 accuracy is mostly close to the accuracy achieved when $\eta = 0$

Resisting Memorization:

- Recall: S is the original training data and S_η is the data with randomly altered labels; $\mathcal{D}$ and $\mathcal{D}_\eta$ are the corresponding distributions.
- Let h and h_η denote the classifier learned by an algorithm when given S and S_η as training data, respectively.
- We say that an algorithm **resists memorization** if

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{I}_{\{h(\boldsymbol{x}_i)=y_i^{cl}\}} = \frac{1}{m}\sum_{i=1}^{m}\mathbb{I}_{\{h_\eta(\boldsymbol{x}_i)=y_i^{cl}\}} \tag{13}$$

- Note that the definition stated above is related to the definition of robustness that we had defined in Equation 8.

above using training data. The accuracy for $\eta = 0$ would be an estimate of LHS above.

- For RLL, as we saw, the J_2 accuracy is mostly close to the accuracy achieved when $\eta = 0$

Resisting Memorization: Formulation

- As RLL is a symmetric loss and the noise model we considered is of symmetric noise, the following theorem gives some idea of why RLL has this interesting behaviour.

Theorem

Let $\mathcal{L}$ be a symmetric loss, $\mathcal{D}$ and $\mathcal{D}_\eta$ as defined above. Assume $\eta < \frac{K-1}{K}$. Let $y_{\boldsymbol{x}}^{cl}$ and $y_{\boldsymbol{x}}$ denote the original and noisy labels corresponding to the pattern $\boldsymbol{x}$. The risk of h over $\mathcal{D}$ and $\mathcal{D}_\eta$ is $R_{\mathcal{L}}(h) = \mathbb{E}_{\mathcal{D}}[\mathcal{L}(h(\boldsymbol{x}), y_{\boldsymbol{x}}^{cl})]$ and $R_{\mathcal{L}}^{\eta}(h) = \mathbb{E}_{\mathcal{D}_\eta}[\mathcal{L}(h(\boldsymbol{x}), y_{\boldsymbol{x}})]$ resp. Then, given any two classifiers h_1 and h_2, if $R_{\mathcal{L}}(h_1) < R_{\mathcal{L}}(h_2)$, then $R_{\mathcal{L}}^{\eta}(h_1) < R_{\mathcal{L}}^{\eta}(h_2)$ and vice versa.

Resisting Memorization: Formulation

- As RLL is a symmetric loss and the noise model we considered is of symmetric noise, the following theorem gives some idea of why RLL has this interesting behaviour.

Theorem

Let $\mathcal{L}$ be a symmetric loss, $\mathcal{D}$ and $\mathcal{D}_\eta$ as defined above. Assume $\eta < \frac{K-1}{K}$. Let $y_{\boldsymbol{x}}^{cl}$ and $y_{\boldsymbol{x}}$ denote the original and noisy labels corresponding to the pattern $\boldsymbol{x}$. The risk of h over $\mathcal{D}$ and $\mathcal{D}_\eta$ is $R_{\mathcal{L}}(h) = \mathbb{E}_{\mathcal{D}}[\mathcal{L}(h(\boldsymbol{x}), y_{\boldsymbol{x}}^{cl})]$ and $R_{\mathcal{L}}^{\eta}(h) = \mathbb{E}_{\mathcal{D}_\eta}[\mathcal{L}(h(\boldsymbol{x}), y_{\boldsymbol{x}})]$ resp. Then, given any two classifiers h_1 and h_2, if $R_{\mathcal{L}}(h_1) < R_{\mathcal{L}}(h_2)$, then $R_{\mathcal{L}}^{\eta}(h_1) < R_{\mathcal{L}}^{\eta}(h_2)$ and vice versa.

- For symmetric losses relative risks of two classifiers are same both with and without noise.

Resisting Memorization: Formulation

- As RLL is a symmetric loss and the noise model we considered is of symmetric noise, the following theorem gives some idea of why RLL has this interesting behaviour.

Theorem

Let $\mathcal{L}$ be a symmetric loss, $\mathcal{D}$ and $\mathcal{D}_\eta$ as defined above. Assume $\eta < \frac{K-1}{K}$. Let $y_{\boldsymbol{x}}^{cl}$ and $y_{\boldsymbol{x}}$ denote the original and noisy labels corresponding to the pattern $\boldsymbol{x}$. The risk of h over $\mathcal{D}$ and $\mathcal{D}_\eta$ is $R_{\mathcal{L}}(h) = \mathbb{E}_{\mathcal{D}}[\mathcal{L}(h(\boldsymbol{x}), y_{\boldsymbol{x}}^{cl})]$ and $R_{\mathcal{L}}^{n}(h) = \mathbb{E}_{\mathcal{D}_\eta}[\mathcal{L}(h(\boldsymbol{x}), y_{\boldsymbol{x}})]$ resp. Then, given any two classifiers h_1 and h_2, if $R_{\mathcal{L}}(h_1) < R_{\mathcal{L}}(h_2)$, then $R_{\mathcal{L}}^{\eta}(h_1) < R_{\mathcal{L}}^{\eta}(h_2)$ and vice versa.

- For symmetric losses relative risks of two classifiers are same both with and without noise.
- So, symmetric losses can resist memorization (if we can get minimum of risk)

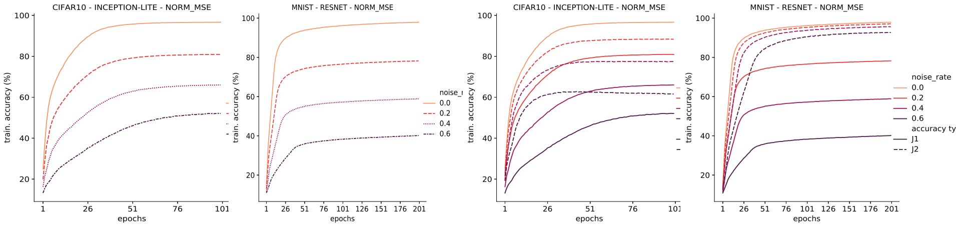
Resisting Memorization: Symmetric Losses

- As is easy to see, the symmetry condition implies that the loss function is bounded.
- Given a bounded loss function we can satisfy the symmetry condition by 'normalizing' it. Given a bounded loss, L, define $\bar{L}$, by

$$\bar{L}(h(X), j) = \frac{L(h(X), j)}{\sum_s L(h(X), s)} \tag{14}$$

- $\bar{L}$ satisfies the symmetry condition (Equation 11). As mentioned earlier, CCE loss is unbounded and hence normalization would not turn it into a symmetric loss. However, we can normalize MSE loss.

Results for J_1 and J_2 accuracy - with Norm. MSE



(a) *Norm.MSE* **(b)** *Norm.MSE* **(c)** *Norm.MSE* **(d)** *Norm.MSE*

Figure 12: Train. accuracy and J_1 & J_2 accuracies for Inception-Lite ((a) & (c)) & ResNet-18 ((b) & (d)) trained on CIFAR-10 and MNIST resp. for $\eta \in \{0., 0.2, 0.4, 0.6\}$ (Solid lines show J_1 accuracy; dashed lines show J_2 accuracy)

- Once we normalize MSE, it no longer overfits the data with random labels; it behaves more like RLL now.

Conclusions I

- We first looked at problem of robust learning under label noise from a risk minimization perspective.
- We also defined the relation between the noisy and clean labels through conditional probabilities and categorized label noise into different types.
- Next, we looked at the existing approaches in literature and categorized them accordingly.
- Motivated by the limitations of existing (sample reweighting) methods, viz. knowledge of noise rates, small set of extra clean data, and additional computation resources, we propose an adaptive, data-dependent sample selection scheme, BARE, for robust learning under label noise.

Conclusions II

- BARE relies on statistics of assigned posterior probabilities of all samples in a minibatch to select samples from it. The mini-batch statistics are used as proxies for determining current state of learning here.
- Unlike other algorithms, BARE neither needs an extra data set with clean labels nor does it need any knowledge of the noise rates. Further it has no hyperparameters in the selection algorithm. Comparisons with baseline schemes on benchmark datasets show the effectiveness of the proposed algorithm both in terms of performance metrics and computational complexity.
- Since DNNs can interpolate training data even in case of randomly labelled data, the phenomenon of memorization in deep networks has received a lot of attention.
- However, role of loss function on memorization in deep networks hasn't been studied.

Conclusions III

- We showed through empirical studies that changing the loss function alone can significantly change this memorization.
- We showed this with the symmetric loss function, RLL, and we have provided some theoretical analysis to explain the empirical results.
- The results presented suggest that choice of loss function can play a critical role in overfitting by deep networks.
- It also highlights the need to further investigate the nature of different (symmetric) loss functions for a better understanding of robust learning.

Publications based on thesis

- **Patel, D.** and Sastry, P.S. *Memorization in Deep Neural Networks: Does the Loss Function matter?*, accepted in PAKDD 2021
- **Patel, D.** and Sastry, P.S. *Adaptive Sample Selection for Robust Learning under Label Noise*, under review

Acknowledgements

- Nisha and Vidyadhar

Acknowledgements

- Nisha and Vidyadhar
- Prof. Sastry

Acknowledgements

- Nisha and Vidyadhar
- Prof. Sastry
- IISc

Acknowledgements

- Nisha and Vidyadhar
- Prof. Sastry
- IISc
- Pavan, Aravind, and Karthik (SPIRE Lab)
- Dr. Prasanta Kumar Ghosh

Acknowledgements

- Nisha and Vidyadhar
- Prof. Sastry
- IISc
- Pavan, Aravind, and Karthik (SPIRE Lab)
- Dr. Prasanta Kumar Ghosh
- Lalit

Acknowledgements

- Nisha and Vidyadhar
- Prof. Sastry
- IISc
- Pavan, Aravind, and Karthik (SPIRE Lab)
- Dr. Prasanta Kumar Ghosh
- Lalit
- Avijith, Vishnu, and Kartik

Acknowledgements

- Nisha and Vidyadhar
- Prof. Sastry
- IISc
- Pavan, Aravind, and Karthik (SPIRE Lab)
- Dr. Prasanta Kumar Ghosh
- Lalit
- Avijith, Vishnu, and Kartik
- NoteBook Drive (NBD) family

Acknowledgements

- Nisha and Vidyadhar
- Prof. Sastry
- IISc
- Pavan, Aravind, and Karthik (SPIRE Lab)
- Dr. Prasanta Kumar Ghosh
- Lalit
- Avijith, Vishnu, and Kartik
- NoteBook Drive (NBD) family
- Maqbool Sir

Thank You

Any Questions?

References I

Eric Arazo, Diego Ortego, Paul Albert, Noel O'Connor, and Kevin McGuinness.
Unsupervised label noise modeling and loss correction.
In *International Conference on Machine Learning*, pages 312–321. PMLR, 2019.

Eric Arazo, Diego Ortego, Paul Albert, Noel O'Connor, and Kevin McGuinness.
Unsupervised label noise modeling and loss correction.
In *International Conference on Machine Learning*, pages 312–321. PMLR, 2019.

References II

Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al.
A closer look at memorization in deep networks.
In *International Conference on Machine Learning*, pages 233–242. PMLR, 2017.

Carla E Brodley and Mark A Friedl.
Identifying mislabeled training data.
Journal of artificial intelligence research, 11:131–167, 1999.

Nontawat Charoenphakdee, Jongyeong Lee, and Masashi Sugiyama.
On symmetric losses for learning from corrupted labels.
In *International Conference on Machine Learning*, pages 961–970. PMLR, 2019.

References III

Pengfei Chen, Junjie Ye, Guangyong Chen, Jingwei Zhao, and Pheng-Ann Heng.
Beyond class-conditional assumption: A primary attempt to combat instance-dependent label noise.
arXiv preprint arXiv:2012.05458, 2020.

Luis Daza and Edgar Acuna.
An algorithm for detecting noise on supervised classification.
In *Proceedings of WCECS-07, the 1st World Conference on Engineering and Computer Science*, pages 701–706, 2007.

A. Demirkaya, J. Chen, and S. Oymak.
Exploring the role of loss functions in multiclass classification.
In *2020 54th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–5, 2020.

References IV

Vitaly Feldman.
Does learning require memorization? a short tale about a long tail.
In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959, 2020.

Vitaly Feldman and Chiyuan Zhang.
What neural networks memorize and why: Discovering the long tail via influence estimation.
Advances in Neural Information Processing Systems, 33, 2020.

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

Jindong Gu and Volker Tresp.
Neural network memorization dissection, 2019.

References V

Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama.
Co-teaching: Robust training of deep neural networks with extremely noisy labels.
In *Advances in neural information processing systems*, pages 8527–8537, 2018.

Sariel Har-Peled, Dan Roth, and Dav Zimak.
Maximum margin coresets for active and noise tolerant learning.
In *IJCAI*, pages 836–841, 2007.

References VI

Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant.
Array programming with NumPy.
Nature, 585(7825):357–362, Sept. 2020.

Dan Hendrycks, Mantas Mazeika, Duncan Wilson, and Kevin Gimpel.
Using trusted data to train deep networks on labels corrupted by severe noise.
In *Advances in neural information processing systems*, pages 10456–10465, 2018.

References VII

Like Hui and Mikhail Belkin.
Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks, 2020.

Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei.
Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels.
In *International Conference on Machine Learning*, pages 2304–2313. PMLR, 2018.

George H John.
Robust decision trees: Removing outliers from databases.
In *KDD*, volume 95, pages 174–179, 1995.

References VIII

Amitava Karmaker and Stephen Kwek.
A boosting approach to remove class label noise.
International Journal of Hybrid Intelligent Systems, 3(3):169–177, 2006.

Diederik P Kingma and Jimmy Ba.
Adam: A method for stochastic optimization.
arXiv preprint arXiv:1412.6980, 2014.

Alex Krizhevsky.
Learning Multiple Layers of Features from Tiny Images.
PhD thesis, University of Toronto, 2009.

H. Kumar and P. S. Sastry.
Robust loss functions for learning multi-class classifiers.
In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 687–692, 2018.

References IX

M Pawan Kumar, Benjamin Packer, and Daphne Koller.
Self-paced learning for latent variable models.
In *Proceedings of the 23rd International Conference on Neural Information Processing Systems-Volume 1*, pages 1189–1197, 2010.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner.
Gradient-based learning applied to document recognition.
Proceedings of the IEEE, 86(11):2278–2324, 1998.

Junnan Li, Richard Socher, and Steven C.H. Hoi.
Dividemix: Learning with noisy labels as semi-supervised learning.
In *International Conference on Learning Representations*, 2020.

Junnan Li, Yongkang Wong, Qi Zhao, and Mohan S Kankanhalli.
Learning to learn from noisy labeled data.
In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5051–5059, 2019.

References X

Yueming Lyu and Ivor W. Tsang.
Curriculum loss: Robust learning and generalization against label corruption.
In *International Conference on Learning Representations*, 2020.

Xingjun Ma, Hanxun Huang, Yisen Wang, Simone Romano, Sarah Erfani, and James Bailey.
Normalized loss functions for deep learning with noisy labels.
In *International Conference on Machine Learning*, pages 6543–6553. PMLR, 2020.

Xingjun Ma, Yisen Wang, Michael E Houle, Shuo Zhou, Sarah Erfani, Shutao Xia, Sudanthi Wijewickrema, and James Bailey.
Dimensionality-driven learning with noisy labels.
In *International Conference on Machine Learning*, pages 3355–3364. PMLR, 2018.

References XI

Xingjun Ma, Yisen Wang, Michael E. Houle, Shuo Zhou, Sarah Erfani, Shutao Xia, Sudanthi Wijewickrema, and James Bailey.
Dimensionality-driven learning with noisy labels.
In *Proceedings of the 35th International Conference on Machine Learning*, pages 3355–3364, 2018.

Eran Malach and Shai Shalev-Shwartz.
Decoupling" when to update" from" how to update".
In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 961–971, 2017.

Naresh Manwani and PS Sastry.
Noise tolerance under risk minimization.
IEEE transactions on cybernetics, 43(3):1146–1151, 2013.

References XII

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii.
Virtual adversarial training: a regularization method for supervised and semi-supervised learning.
IEEE transactions on pattern analysis and machine intelligence, 41(8):1979–1993, 2018.

Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari.
Learning with noisy labels.
In *Advances in neural information processing systems*, pages 1196–1204, 2013.

References XIII

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala.
Pytorch: An imperative style, high-performance deep learning library.
In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alche-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

References XIV

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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay.
Scikit-learn: Machine learning in Python.
Journal of Machine Learning Research, 12:2825–2830, 2011.

References XV

Qichao Que and Mikhail Belkin.
Back to the future: Radial basis function networks revisited.
In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 1375–1383, Cadiz, Spain, 09–11 May 2016. PMLR.

Scott E Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich.
Training deep neural networks on noisy labels with bootstrapping.
In *ICLR (Workshop)*, 2015.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun.
Learning to reweight examples for robust deep learning.
In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4334–4343, 2018.

References XVI

Ryan Michael Rifkin.
Everything old is new again: a fresh look at historical approaches in machine learning.
PhD thesis, Massachussets Insitute of Technology, 2002.

Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng.
Meta-weight-net: Learning an explicit mapping for sample weighting.
In *Advances in Neural Information Processing Systems*, pages 1919–1930, 2019.

Hwanjun Song, Minseok Kim, and Jae-Gil Lee.
Selfie: Refurbishing unclean samples for robust deep learning.
In *ICML*, pages 5907–5915, 2019.

References XVII

Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa.
Joint optimization framework for learning with noisy labels.
In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5552–5560, 2018.

Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey.
Symmetric cross entropy for robust learning with noisy labels.
In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 322–330, 2019.

Zhen Wang, Guosheng Hu, and Qinghua Hu.
Training noise-robust deep neural networks via meta-learning.
In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4524–4533, 2020.

References XVIII

Xiaobo Xia, Tongliang Liu, Bo Han, Nannan Wang, Mingming Gong, Haifeng Liu, Gang Niu, Dacheng Tao, and Masashi Sugiyama.
Part-dependent label noise: Towards instance-dependent label noise.
Advances in Neural Information Processing Systems, 33, 2020.

Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang.
Learning from massive noisy labeled data for image classification.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2691–2699, 2015.

Linli Xu, Koby Crammer, and Dale Schuurmans.
Robust support vector machine training via convex outlier ablation.
In *AAAI*, volume 6, pages 536–542, 2006.

References XIX

Quanming Yao, Hansi Yang, Bo Han, Gang Niu, and James Tin-Yau Kwok.
Searching to exploit memorization effect in learning with noisy labels.
In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 10789–10798. PMLR, 2020.

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

Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor Tsang, and Masashi Sugiyama.
How does disagreement help generalization against label corruption?
In *Proceedings of the 36th International Conference on Machine Learning*, pages 7164–7173, 2019.

References XX

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals.
Understanding deep learning requires rethinking generalization.
arXiv preprint arXiv:1611.03530, 2016.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz.
mixup: Beyond empirical risk minimization.
arXiv preprint arXiv:1710.09412, 2017.

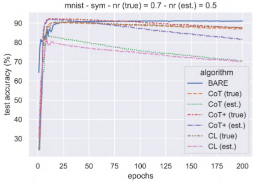
Zhilu Zhang and Mert R Sabuncu.
Generalized cross entropy loss for training deep neural networks with noisy labels.
arXiv preprint arXiv:1805.07836, 2018.

References XXI

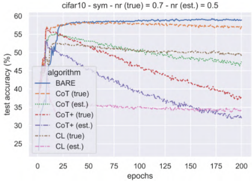
Songzhu Zheng, Pengxiang Wu, Aman Goswami, Mayank Goswami, Dimitris Metaxas, and Chao Chen.
Error-bounded correction of noisy labels.
In *International Conference on Machine Learning*, pages 11447–11457. PMLR, 2020.

Xingquan Zhu, Peng Zhang, Xindong Wu, Dan He, Chengqi Zhang, and Yong Shi.
Cleansing noisy data streams.
In *2008 Eighth IEEE International Conference on Data Mining*, pages 1139–1144. IEEE, 2008.

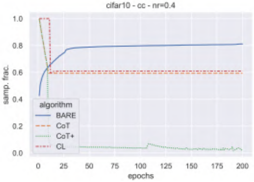
Some comments (contd.)



(a)



(b)



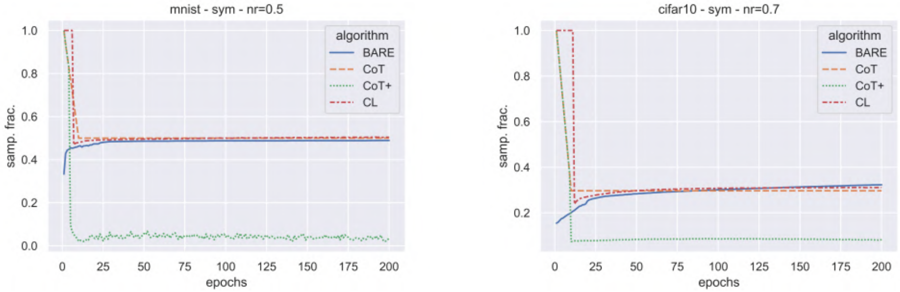
(c)

Figure 13: ((a) & (b)):Test accuracies when estimated (symmetric) noise rate, $\eta = 0.5$, and true noise rate, $\eta = 0.7$, for MNIST & CIFAR-10 resp.; (c): sample fraction values for $\eta = 0.4$ (class-conditional noise) on CIFAR-10

Some comments (contd.)

- This can be seen from Figure 13(c) as well which shows the fraction of samples chosen by the sample selection algorithms as epochs go by for $\eta = 0.4$ (class-conditional noise) on CIFAR-10 dataset.
- As noise rate is to be supplied to CoT and CL, they select $1 - \eta = 0.6$ fraction of data with every epoch. Whereas, in case of CoT+, the samples where the networks disagree is small because of the training dynamics and as a result, after a few epochs, it consistently selects very few samples. Since the noise is class-conditional, even though $\eta = 0.4$, the actual amount of label flipping is $\sim 20\%$. And this is why it's interesting to note that BARE leads to an approximate sample selection ratio of 80%.

Some comments (contd.)



(a) (b)

Figure 14: (a): Sample fraction values for $\eta = 0.5$ (symmetric noise) on MNIST, (b): Sample fraction values for $\eta = 0.7$ (symmetric noise) on CIFAR-10

Some comments (contd.)

- Figures 14(a) and 14(b) show the fraction of samples selected by different algorithms in each epoch for one case of symmetric noise. As is evident from these figures, BARE is able to identify higher fraction of clean samples effectively even without the knowledge of noise rates.

Sensitivity to noise rates (contd.)

Table 5: Test Accuracy (%) for MNIST - $\eta_{est} = 0.45$ (arbitrary noise matrix)

Algorithm	Test Accuracy
CoT [13]	**95.3**
CoT+ [53]	93.07
CL [28]	88.41
BARE (Ours)	**95.02**

Sensitivity to noise rates (contd.)

Table 6: Avg. Test Accuracy (last 10 epochs) (%) for MNIST - $\eta_{est} = 0.45$ (arbitrary noise matrix)

Algorithm	Avg. Test Accuracy (last 10 epochs)
CoT [13]	**95.22**
CoT+ [53]	93.08
CL [28]	88.56
BARE (Ours)	**95.03**

Sensitivity to noise rates (contd.)

Table 7: Test Accuracy (%) for CIFAR10 - $\eta_{est} = 0.4$ (arbitrary noise matrix)

Algorithm	Test Accuracy
CoT [13]	71.92
CoT+ [53]	68.56
CL [28]	**72.12**
BARE (Ours)	**76.22**

Sensitivity to noise rates (contd.)

Table 8: Avg. Test Accuracy (last 10 epochs) (%) for CIFAR10 - $\eta_{est} = 0.4$ (arbitrary noise matrix)

Algorithm	Avg. Test Accuracy (last 10 epochs)
CoT [13]	71.86
CoT+ [53]	68.99
CL [28]	**72.27**
BARE (Ours)	**75.96**

Arbitrary Noise Rate Matrix for MNIST

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.6 & 0 & 0 & 0 & 0 & 0.3 & 0 & 0.1 \\ 0 & 0 & 0 & 0.5 & 0 & 0.1 & 0 & 0 & 0.4 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.15 & 0.55 & 0.3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.35 & 0.55 & 0.10 & 0 & 0 \\ 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0.25 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Arbitrary Noise Rate Matrix for MNIST

Arbitrary Noise rate Matrix for CIFAR-10

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.2 & 0 & 0.7 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 \\
0.1 & 0 & 0 & 0.6 & 0 & 0.1 & 0 & 0 & 0.2 & 0 \\
0 & 0.1 & 0.1 & 0 & 0.7 & 0 & 0 & 0.1 & 0 & 0 \\
0 & 0 & 0 & 0.1 & 0 & 0.6 & 0 & 0 & 0 & 0.3 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0.8
\end{bmatrix}$$

Arbitrary Noise rate Matrix for CIFAR-10