CHALMERS
UNIVERSITY OF TECHNOLOGY

Label noise-robust methods for supervised learning with deep neural networks

A reimplementation and review of recent methods

Master's thesis in Engineering Mathematics and Computational Science

RIGON DEMISAI

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master's thesis 2020

Label noise-robust methods for supervised learning with deep neural networks

A reimplementation and review of recent methods

RIGON DEMISAI

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Automatic Control Group
Chalmers University of Technology
Gothenburg, Sweden 2020

Label noise-robust methods for supervised learning with deep neural networks
A reimplementation and review of recent methods
RIGON DEMISAI

Supervisor: Rebecka Jörnsten, Dept. of Mathematical Sciences, Viktor Andersson, Centiro
Examiner: Balazs Kulcsar, Dept. of Electrical Engineering

Master's Thesis 2020
Department of Electrical Engineering
Division of Systems and Control
Automatic Control Group
Chalmers University of Technology
SE-412 96 Gothenburg

Typeset in LaTeX
Gothenburg, Sweden 2020

Label noise-robust methods for supervised learning with deep neural networks
A reimplementation and review of recent methods
RIGON DEMISAI
Chalmers University of Technology

Abstract

Despite their success in image recognition, image segmentation and object detection tasks, convolutional neural networks have been shown to easily memorize noisy labels. Learning from noisy labels significantly degrades model performance and poses a challenge to applications using real-world datasets where mislabeling is common.

There is a growing literature devoted to the issues of learning with label noise. Proposed approaches tend to be either label noise-robust methods such as loss correction- or label correction methods, probabilistic label-noise tolerant methods, data cleansing methods, or some combination of the above.

The main contribution of this thesis is the reimplementation, combination and review of a small but wide selection of recent learning methods that are designed to mitigate label noise. A comparison of these methods is provided for the task of classifying images in a label noise setting.

Keywords: classification, image recognition, convolutional neural networks, label noise, loss correction, label correction, data cleansing, noise modeling, robust methods, reimplementation, review.

Contents

1

Introduction

Deep neural networks have made impressive performance leaps in recent years in a wide variety of tasks. These improvements are in part due to the design of better network architectures and the availability of large-scale labeled datasets.

However, collecting labeled samples for large datasets remains a practical hurdle for the development of most real-world systems. Gathering precise labels is a time consuming process that requires careful involvement from human experts. Labeling errors are common and they can arise through systematic or random processes.

It has been observed that deep neural networks are prone to fit random labels (Zhang et al., 2016), which may cause significant reductions in performance that can be detrimental in real-world applications. This is in part, due to the ability of deep neural networks to memorize random labels when the *number* of correct labels is insufficient relative to the *proportion* of random labels to correct labels (Rolnick et al., 2017). These kinds of observations have led to the development of methods to make deep neural networks more robust to label noise. For a survey of deep learning methods that help mitigate label noise, see (Frenay & Verleysen, 2014).

This paper reimplements and compares a small but wide selection of recently proposed learning methods that aim to mitigate the effects of label noise. These methods are tested at different levels of synthetic uniform label noise such that an increase in label noise has the effect of decreasing the *proportion* of correct samples and their *absolute number*. See (Rolnick et al., 2017) for a study that investigates these effects separately.

Mixup- (Zhang et al., 2017) and *class entropy* regularization (Tanaka et al., 2018) are used in several reimplemented models to improve calibration, generalization and robustness to label noise.

Two loss correction approaches are covered. The first is called *bootstrap loss* (Reed et al., 2014) which forms a convex combination of original labels and predicted pseudo-labels. The bootstrap loss is combined with a *beta mixture model* (Arazo et al., 2019) to model the label noise and dynamically re-weight the loss if a sample is likely to be noisy. The second loss correction approach is called *forward correction* which estimates the noise associated with each class through a probability transition matrix and corrects the loss by modifying the predictions on each sample (Hendrycks et al., 2018).

A label correction and curriculum learning approach is explored, it is called *self-adaptive training* (Huang et al., 2020) which aims to gradually correct the labels throughout the training procedure and re-weights the per sample loss to emphasize confident predictions to selectively guide the loss gradient.

The final type of approach aims to identify noisy samples in a dataset such that they may be plucked out and relabeled by some other process. This is achieved by tracking the prediction confidence on a per sample basis with a statistical metric called *area under the margin ranking* (Pleiss et al., 2020). Supervised learning may proceed on the cleaner subset with reduced size- and noise proportion, or both subsets may be used in the case of semi-supervised learning where the identified noisy labels are removed.

2

Learning with label noise

To successfully train a deep neural network classifier with potentially noisy labels some understanding of label noise and its effect on the training dynamics is needed. The purpose of this chapter is to establish preliminaries and notation, explain the main causes of label noise, how it can be modeled, and how deep neural networks are affected by label noise during supervised learning.

2.1 Preliminaries and notation

For notational purposes, column vectors are written in bold (e.g. $\boldsymbol{x}$). Let $\mathcal{C} = \{1, \ldots, c\}$ be an ordered set of positive integers; $|\mathcal{C}|$ denotes the cardinality of the set. Denote $\boldsymbol{e}^i \in \{0, 1\}^{|\mathcal{C}|}$ as the ith standard canonical vector in $\mathbb{R}^{|\mathcal{C}|}$, such that $\sum_{j=1}^{|\mathcal{C}|} \boldsymbol{e}^j = 1$.

Consider a supervised c-class classification problem with an input space $\mathcal{X} \subset \mathbb{R}^d$ and a corresponding label space $\mathcal{Y} = \{\boldsymbol{e}^i : i \in \mathcal{C}\}$. Let $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N \subset (\mathcal{X} \times \mathcal{Y})$ be a labeled training set with N samples such that $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$ are drawn from an unknown probability distribution $p(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})$.

A classifier $\boldsymbol{h}_{\boldsymbol{\theta}} : \mathcal{X} \to \mathbb{R}^{|\mathcal{C}|}$, maps the input space to the label space. Here, $\boldsymbol{h}_{\boldsymbol{\theta}}$ is a neural network consisting of a composition of intermediate transformations. The vector $\boldsymbol{\theta}$ are the parameters of the neural network (its weights and biases). The estimated true label of a sample is predicted by the class-conditional probabilities $\hat{p}(\boldsymbol{y}|\boldsymbol{x})$ that are obtained by applying the softmax function on the network output

$$\hat{p}(\boldsymbol{y} = \boldsymbol{e}^i|\boldsymbol{x}) = \frac{e^{\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x})_i}}{\sum_{j=1}^{|\mathcal{C}|} e^{\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x})_j}} \quad \text{for} \quad i = 1, \ldots, c \tag{2.1}$$

such that $\sum_{j=1}^{|\mathcal{C}|} \hat{p}(\boldsymbol{y} = \boldsymbol{e}^j|\boldsymbol{x}) = 1$. The short notation $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \hat{p}(\boldsymbol{y}|\boldsymbol{x})$ will be used interchangeably to denote the estimated class-conditional probability obtained from a classifier.

A neural network is trained by iteratively updating its parameters $\boldsymbol{\theta}$ to minimize a measure of its prediction error, also known as empirical risk. A common empirical risk function used for classification is the cross-entropy loss

$$\mathcal{L}(\boldsymbol{f}; \boldsymbol{\theta}_t) = -\frac{1}{N} \sum_{i=1}^{N} \boldsymbol{y}_i^T \log \boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{\theta}_t). \tag{2.2}$$

The network parameters are updated using gradient descent by stepping a distance

$\eta > 0$ in the negative gradient direction:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_t} = \boldsymbol{\theta}_t - \eta \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}} \right)^T \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{\theta}_t}. \tag{2.3}$$

The aim of updating the weights is to iteratively tune the network into making predictions $\boldsymbol{f}(\boldsymbol{x};\boldsymbol{\theta})$ that match the ground truth labels $\boldsymbol{y}$, which consequently minimizes the loss function. One iteration t of training is called an epoch and it involves a complete forward pass on the training set, followed by backpropagation of the accumulated prediction errors, and updating of the network parameters.

2.2 Label noise

Given a training set $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$, traditional supervised learning aims to classify an input $\boldsymbol{x} \in \mathcal{X}$ into the ground truth class $\boldsymbol{y} \in \mathcal{Y}$. In the case of learning with label noise, there is only the noisy dataset $\tilde{\mathcal{D}} = \{(\boldsymbol{x}_i, \tilde{\boldsymbol{y}}_i)\}_{i=1}^N$ to learn from. Here $\tilde{\boldsymbol{y}} \in \mathcal{Y}$ denotes a noisy version of the ground truth label $\boldsymbol{y}$ which has been flipped with transition probability $p(\tilde{\boldsymbol{y}}|\boldsymbol{y}, \boldsymbol{x})$ conditioned on $\boldsymbol{x}$.

There are three main types of label noise: asymmetric in-distribution noise, symmetric in-distribution noise and out-of-distribution noise. In the case of asymmetric noise, training examples $(\boldsymbol{x}, \tilde{\boldsymbol{y}})$ are observed from the joint distribution

$$p(\boldsymbol{x}, \tilde{\boldsymbol{y}}) = \sum_{\boldsymbol{y} \in \mathcal{Y}} p(\tilde{\boldsymbol{y}}|\boldsymbol{y}, \boldsymbol{x}) p(\boldsymbol{y}|\boldsymbol{x}) p(\boldsymbol{x}). \tag{2.4}$$

It is frequently assumed in the literature that the transition probability is conditionally independent of the input $\boldsymbol{x}$. In that case, the transition probability simplifies into

$$p(\tilde{\boldsymbol{y}} = \boldsymbol{e}^j | \boldsymbol{y} = \boldsymbol{e}^i, \boldsymbol{x}) = p(\tilde{\boldsymbol{y}} = \boldsymbol{e}^j | \boldsymbol{y} = \boldsymbol{e}^i) = T_{ij} \tag{2.5}$$

for all $i, j \in \mathcal{C}$. The asymmetric noise model captures the labeling error that can happen when a subset of classes are very similar.

Label noise is called symmetric when the label is flipped uniformly across all classes with same probability such that

$$p(\tilde{\boldsymbol{y}} = \boldsymbol{e}^j | \boldsymbol{y} = \boldsymbol{e}^i) = p(\tilde{\boldsymbol{y}} = \boldsymbol{e}^k | \boldsymbol{y} = \boldsymbol{e}^i) = \gamma \tag{2.6}$$

for all $j, k \neq i$. The transition matrix then takes the form $T_{ij} = 1 - \gamma$ if $i = j$ and $T_{jk} = \gamma/(c-1)$ if $j \neq i$. This is the simplest noise model since it assumes that the labeling error is due to uniform random chance across all classes.

Out-of-distribution noise occurs when a noisy label represents an unknown class that does not belong to the particular dataset in question.

Figure 2.1 shows how a deep neural network classifier $\boldsymbol{h}_{\boldsymbol{\theta}}$ with standard cross-entropy loss (2.2) is non-robust to label noise and ends up memorizing the corrupted labels. Note how the loss separation between clean and noisy samples tends to zero as the model fits the noisy samples and the errors on the train and test sets diverge.

Coincidentally, the average accuracy on the train set gets maximized while it drops significantly on the test set.

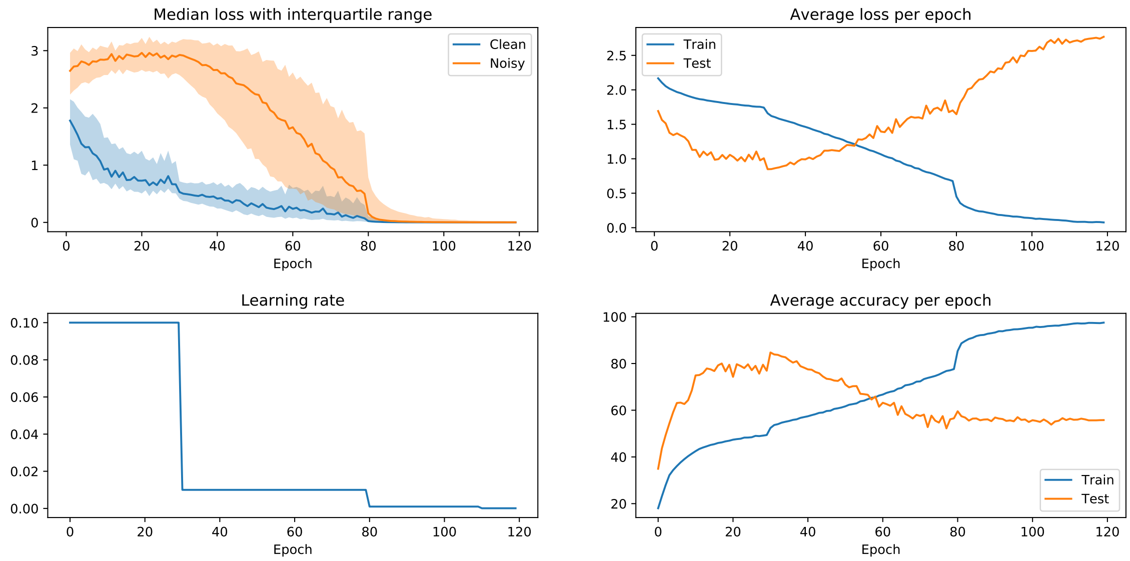

Figure 2.1: Empirical results from training a PreAct ResNet-18 with standard cross-entropy loss (2.2) on CIFAR-10 with with 50% symmetric label noise.

3

Methods to mitigate label noise

This chapter presents the theory of a selection of training methods that help to mitigate the effects of label noise during supervised learning. These methods involve some combination of regularizers, robust losses, loss correction, label correction and noise modeling techniques.

3.1 Regularizers

Regularization aims to enhance certain characteristics of a model such as robustness, calibration and generalization. There are various forms of regularization mechanisms for deep learning. Some methods aim to reduce model complexity by penalizing model parameters (Goodfellow et al., 2016), or add noise to intermediate representations or outputs (Srivastava et al., 2014), or normalize network activations (Ioffe & Szegedy, 2015). Other examples include data augmentation and transformation methods (Goodfellow et al., 2016) that modify the training set.

This section describes two regularization methods that are useful to improve robustness to label noise.

3.1.1 Mixup

Mixup (Zhang et al., 2017) is a data augmentation technique which forms virtual training examples through convex combinations of randomly selected sample pairs ($\boldsymbol{x}_p$ and $\boldsymbol{x}_q$) and their corresponding labels ($\boldsymbol{y}_p$ and $\boldsymbol{y}_q$) such that

$$\begin{aligned} \tilde{\boldsymbol{x}} &= \lambda \boldsymbol{x}_p + (1-\lambda)\boldsymbol{x}_q, \\ \tilde{\boldsymbol{y}} &= \lambda \boldsymbol{y}_p + (1-\lambda)\boldsymbol{y}_q, \end{aligned} \tag{3.1}$$

where $\lambda \sim \mathrm{Beta}(\alpha, \alpha)$, with $\alpha > 0$. The mixup parameter α controls the strength of the interpolation. The original sample is recovered when $\alpha \to 0$, while $\alpha \to \infty$ leads to a mixture of equal proportions.

Increasing the mixup interpolation parameter leads to virtual examples that are further away from the training examples. Hence interpolations between clean samples improve generalization and leads to smoother decision boundaries, while interpolations between noisy samples become difficult to memorize due to the randomness of mixed-up samples. This property makes mixup robust to label noise.

Mixup has been shown to achieve a number of regularization effects such as calibration, Jacobian regularization, robustness, normalization, and generalization by extending the training distribution through sample interpolations. (Carratino et al., 2020)

In practise, mixup is performed on each mini-batch per epoch where one takes a list with the original mini-batch samples and forms a convex combination with a random permutation of that list of samples.

3.1.2 Class entropy regularization

A well calibrated classifier should spread its predictions in proportion to the expected class probabilities. To achieve this, the loss should be penalized if the classifier focuses its predictions on a small subset of classes. This objective can be achieved by *class entropy regularization* (Tanaka et al., 2018) which measures the divergence of one probability distribution compared to a reference distribution according to the equation

$$\mathcal{R}_C = \sum_{c=1}^{|\mathcal{C}|} p_c \log\left(\frac{p_c}{\bar{\boldsymbol{f}}_c}\right). \tag{3.2}$$

Here p_c is the *prior* probability that some input $\boldsymbol{x} \in \mathcal{D}$ belongs to class $c \in \mathcal{C}$, and $\bar{\boldsymbol{f}}_c$ is the mean of the class-conditional probabilities that the model assigns to class c. If the classes in the dataset are well balanced then it is assumed that $p_c = 1/|\mathcal{C}|$. In general, different datasets may require different priors. In practise, $\bar{\boldsymbol{f}}$ may be approximated using mini-batches $\mathcal{B} \subset \mathcal{D}$ such that

$$\bar{\boldsymbol{f}}(\mathcal{D}; \theta_t) = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{f}(\boldsymbol{x}_i, \theta_t) \approx \frac{1}{|\mathcal{B}|}\sum_{\boldsymbol{x}\in\mathcal{B}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}_t). \tag{3.3}$$

Note that mini-batch approximations may be of poor quality if the dataset contains either a large number of classes or extremely imbalanced classes.

The class entropy regularization is added as a penalty term to the original loss, such that

$$\mathcal{L}_{Total} = \mathcal{L} + \lambda\mathcal{R}_C \quad \text{for} \quad \lambda > 0. \tag{3.4}$$

3.2 Noise modeling

Modeling label noise may provide information about noisy samples that can be leveraged for sample identification, or for loss- and label correction during training. This section covers three types of methods to model label noise.

3.2.1 Noise transition matrix

A noise transition matrix $T \in \mathbb{R}^{|\mathcal{C}|\times|\mathcal{C}|}$ measures the probability of one label being flipped into another. See equations (2.5) and (2.6) for the respective cases of asymmetric and symmetric noise. Such a matrix can be used to correct the loss of a classifier so that it becomes more robust to label noise.

The noise transition matrix is generally unknown in practise and must be estimated. Authors differ in their ways of estimating the transition matrix and impose various assumptions which may or may not be realizable. In what follows, it will be assumed that there is access to a small dataset $\mathcal{D}$ with clean labels and a training set $\tilde{\mathcal{D}}$ with potentially noisy labels. In this setting, the authors (Hendrycks et al., 2018) propose a relatively simple method to estimate the transition matrix.

The aim is to estimate the transition probability $p(\tilde{\boldsymbol{y}}|\boldsymbol{y})$ given a classifier $\hat{p}(\tilde{\boldsymbol{y}}|\boldsymbol{x})$ trained on the noisy dataset $\tilde{\mathcal{D}}$. Using Bayes rule the following identity will do the job

$$p(\tilde{\boldsymbol{y}}|\boldsymbol{y},\boldsymbol{x})p(\boldsymbol{x}|\boldsymbol{y}) = p(\tilde{\boldsymbol{y}}|\boldsymbol{y})p(\boldsymbol{x}|\tilde{\boldsymbol{y}},\boldsymbol{y}). \tag{3.5}$$

Summing over all $\boldsymbol{x}$ gives

$$\sum_{\boldsymbol{x}\in\mathcal{D}} p(\tilde{\boldsymbol{y}}|\boldsymbol{y},\boldsymbol{x})p(\boldsymbol{x}|\boldsymbol{y}) = p(\tilde{\boldsymbol{y}}|\boldsymbol{y})\sum_{\boldsymbol{x}\in\mathcal{D}} p(\boldsymbol{x}|\tilde{\boldsymbol{y}},\boldsymbol{y}) = p(\tilde{\boldsymbol{y}}|\boldsymbol{y}). \tag{3.6}$$

Assuming that $\tilde{\boldsymbol{y}}$ and $\boldsymbol{y}$ are independent given $\boldsymbol{x}$, it follows that $p(\tilde{\boldsymbol{y}}|\boldsymbol{y},\boldsymbol{x}) = p(\tilde{\boldsymbol{y}}|\boldsymbol{x}) \approx \hat{p}(\tilde{\boldsymbol{y}}|\boldsymbol{x})$. This assumption tends to hold in practise since the mislabeling process seldom depends on the ground truth label. It is also proven in the appendix of (Hendrycks et al., 2018) that the assumption follows if the data is separable, i.e. if there exists a deterministic label $\boldsymbol{y}$ for input $\boldsymbol{x}$.

Given a subset $\mathcal{D}_i \subset \mathcal{D}$ of clean samples belonging to label $i \in \mathcal{C}$, the noise transition matrix can be estimated by

$$\begin{aligned}\hat{T}_{ij} &= \frac{1}{|\mathcal{D}_i|}\sum_{\boldsymbol{x}\in\mathcal{D}_i}\hat{p}(\tilde{\boldsymbol{y}}=\boldsymbol{e}^j|\boldsymbol{x}) = \frac{1}{|\mathcal{D}_i|}\sum_{\boldsymbol{x}\in\mathcal{D}_i}\hat{p}(\tilde{\boldsymbol{y}}=\boldsymbol{e}^j|\boldsymbol{y}=\boldsymbol{e}^i,\boldsymbol{x})\\ &\approx p(\tilde{\boldsymbol{y}}=\boldsymbol{e}^j|\boldsymbol{y}=\boldsymbol{e}^i).\end{aligned} \tag{3.7}$$

That is, the transition probability T_{ij} is estimated by the average probability predicted by the model $\hat{p}(\tilde{\boldsymbol{y}}=\boldsymbol{e}^j|\boldsymbol{x})$ for samples $\boldsymbol{x}\in\mathcal{D}_i$ with ground truth belonging to class i.

3.2.2 Beta mixture model

It has been observed by (Arazo et al., 2019) that deep neural networks learn random labels slower than clean labels. In particular, noisy samples have higher loss during the early epochs, as can be seen in Figure 3.1. This presents an opportunity to identify noisy samples by tracking the loss distribution of the training samples over time. To do this, (Arazo et al., 2019) propose to fit a beta mixture model to the loss distribution.

A mixture model is a convex combination of probability density functions

$$f(x) = \sum_{k=1}^{K}\lambda_k p(x|k), \quad \sum_{k=1}^{K}\lambda_k = 1, \tag{3.8}$$

where $K \in \mathbb{N}$ is the number of mixture components and $x \in \mathbb{R}$. Here, $p(x|k)$ is the kth component of a beta probabilty density function

$$p(x;\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}x^{\alpha-1}(1-x)^{\beta-1}, \tag{3.9}$$

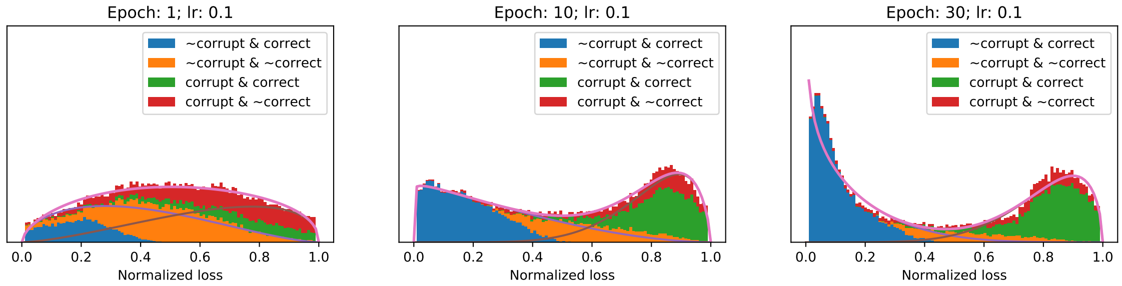

Figure 3.1: Loss distributions at different epochs when training a PreAct ResNet-18 with cross-entropy loss on CIFAR-10. The training set contains 50% symmetric label noise. The colored legend indicates whether a subset is corrupted by label noise and if the ground truth label is correctly predicted. The colored lines show the fitted curves of a two component beta mixture model.

where $\Gamma(z)$ is the Gamma function and $\alpha, \beta > 0$ are the shape parameters of the distribution.

Fitting a probabilty density function to a loss distribution seen in Figure 3.1 requires the estimation of its parameters α and β. These parameters happen to define the first and second moments of the beta distribution in the following way

$$\mu = \frac{\alpha}{\alpha + \beta}, \quad s^2 = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}. \tag{3.10}$$

Solving the coupled equations (3.10) gives the shape parametets as functions of the mean and variance

$$\alpha = \mu \left(\frac{\mu(1-\mu)}{s^2} - 1 \right), \quad \beta = (1-\mu) \left(\frac{\mu(1-\mu)}{s^2} - 1 \right), \tag{3.11}$$

which can be estimated empirically from all the sample losses in one epoch.

The mixture model parameters α, β and λ can be estimated using the *expectation maximization* method in order to fit a two-component beta mixture model to loss distributions of the types observed in Figure 3.1.

Let $l_i \in \mathbb{R}$ for $i \in \{1, \dots, N\}$ be the sample losses that have been computed during one training epoch. In the *expectation* step, the parameters λ_k, α_k and β_k for each mixture component $k \in \{0, 1\}$ are fixed with some reasonable initial values. Then the posterior probability of each sample l_i is updated using Bayes rule

$$p(k|l_i) = \frac{\lambda_k p(l_i|\alpha_k, \beta_k)}{\sum_{j=1}^{K} \lambda_j p(l_i|\alpha_j, \beta_j)}. \tag{3.12}$$

In the *maximization* step, the posterior probabilites are used to compute the estimated weighted mean- and variance of each component distribution

$$\bar{l}_k = \frac{\sum_{i=1}^{N} p(k|l_i) l_i}{\sum_{i=1}^{N} p(k|l_i)}, \quad \bar{s}_k^2 = \frac{\sum_{i=1}^{N} p(k|l_i)(l_i - \bar{l}_k)^2}{\sum_{i=1}^{N} p(k|l_i)}. \tag{3.13}$$

Given the parametrization in equation (3.11) the parameters of the mixture component are updated such that

$$\alpha_k = \bar{l}_k \left(\frac{\bar{l}_k(1-\bar{l}_k)}{s_k^2} - 1 \right), \quad \beta_k = \frac{\alpha_k(1-\bar{l}_k)}{\bar{l}_k}. \tag{3.14}$$

The mixture coefficients are then updated according to

$$\lambda_k = \frac{1}{N} \sum_{i=1}^{N} p(k|l_i).$$

The algorithm may terminate after some criteria has been met, e.g. when a maximum number of iterations has passed, or the relative change of the parameters between iterations has become small enough. It is possible that the algorithm becomes numerically unstable in practise when for example the losses l_i are too close to zero or one. This issue is avoided by constraining the losses such that $l_i \in [\epsilon, 1-\epsilon]$ for $\epsilon > 0$.

The probability that a sample is noisy is finally given by the posterior probability of the fitted beta mixture model

$$p(k|l_i) = \frac{p(l_i|k)p(k)}{p(l_i)} \tag{3.15}$$

for $k \in \{0, 1\}$. The mixture component $k = 0$ denotes clean samples and $k = 1$ denotes noisy samples. A sample is thus estimated to be noisy with probability $p(k = 1|l_i) \in [0, 1]$.

3.2.3 Area under the margin ranking

The area under the margin (Pleiss et al., 2020) is a ranking statistic that measures the average gap between the logit prediction of the assigned class compared the largest other logit.

The margin at epoch t is thus defined as

$$M^{(t)}(\boldsymbol{x}, \boldsymbol{y} = \boldsymbol{e}^k) = \boldsymbol{z}_k^{(t)}(\boldsymbol{x}) - \max_{i \neq k} \boldsymbol{z}_i^{(t)}(\boldsymbol{x}), \tag{3.16}$$

where $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$ and $\boldsymbol{z}_c^{(t)}(\boldsymbol{x}) \in \mathbb{R}^{|\mathcal{C}|}$ is the logit output at epoch t corresponding to class $c \in \mathcal{C}$.

The area under the margin is the average of the margins across all training epochs for each pair $(\boldsymbol{x}, \boldsymbol{y})$, such that

$$AUM_{\boldsymbol{x},\boldsymbol{y}} = \frac{1}{T} \sum_{t=1}^{T} M^{(t)}(\boldsymbol{x}, \boldsymbol{y}), \tag{3.17}$$

where T is the total number of epochs trained.

The AUM ranking can be intepreted in terms of how easy or difficult it is for a classifier to learn the assigned label of a sample. Knowing the AUM rankings of noisy samples allows for a suitable threshold value to be choosen to separate such samples from the dataset. However, which subset of samples that are noisy is usually not known beforehand. It is therefore required to use fake mislabeled data (called indicator samples) to simulate real mislabeled data.

The process of identifying mislabeled samples is broken down into four steps.

1. Let $\tilde{\mathcal{D}}$ be a training set with some proportion of noisy labels. Randomly select a subset $\mathcal{D}' \subset \tilde{\mathcal{D}}$ of data to be indicator samples. Let $c+1$ be a new fake class which is assigned to all indicator samples. Then form the new training set
$$\mathcal{D}_{Train} = \{(\boldsymbol{x}, \boldsymbol{e}^{c+1}) \in \mathcal{D}'\} \cup (\tilde{\mathcal{D}} \setminus \mathcal{D}')$$
2. Train a neural network on the training set $\mathcal{D}_{Train}$ until the first learning rate drop. While training compute the AUM of all samples.
3. Let the threshold α be the 99th percentile AUM ranking from the indicator sample subset.
4. Identify a set $\mathcal{M}$ of mislabeled samples using α as a threshold:
$$\mathcal{M} = \{(\boldsymbol{x}, \boldsymbol{y}) \in (\mathcal{D}_{Train} \setminus \mathcal{D}') : AUM_{\boldsymbol{x},\boldsymbol{y}} \leq \alpha\}.$$
5. Repeat the process once more with a second indicator sample subset
$$\mathcal{D}'' \subset (\tilde{\mathcal{D}} \setminus \mathcal{D}'), \tag{3.18}$$
such that all samples in the training dataset get their estimated AUM ranking.

Figure (3.2) shows how a majority of noisy and difficult samples can be identified with the above method. The correctly labeled difficult samples (in blue) are those with near zero and negative AUM ranking. What constitutes a difficult sample by AUM ranking may vary with network architecture and noise level.

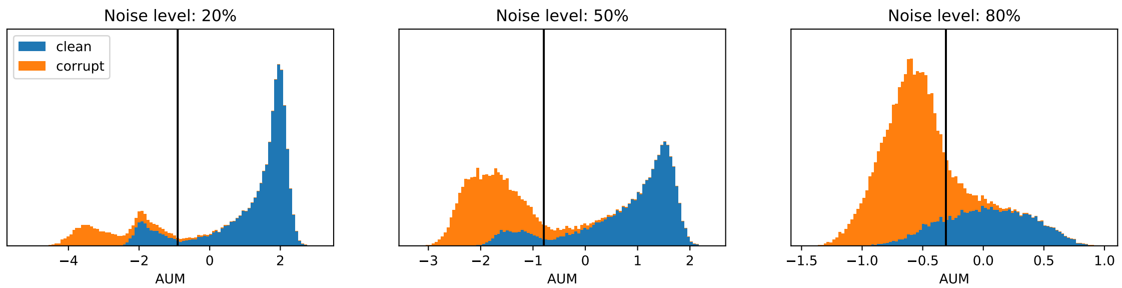

Figure 3.2: Empirical results illustrating the identification of noisy and difficult samples on CIFAR-10 for three levels of label noise. The images are histograms of the AUM rankings for each training sample after 100 epochs of training with standard cross-entropy loss. The vertical line represents the α-threshold obtained from the 99th percentile AUM ranking of the indicator samples.

3.3 Robust learning

This section covers a selection of loss- and label correction methods for robust learning that combine many of the methods and techniques of previous sections to help mitigate the effects of label noise. The following methods are implemented in the label noise experiments in section 4.

3.3.1 Mixup with cross-entropy loss

Mixup regularization (3.1) can be applied to the data and be combined with the cross-entropy (2.2) loss to reduce overfitting to label noise. Mixup is performed on the inputs ($\boldsymbol{x}_p$ and $\boldsymbol{x}_q$) and on the labels ($\boldsymbol{y}_p$ and $\boldsymbol{y}_q$) such that the cross-entropy loss takes the form

$$\begin{aligned}\tilde{\mathcal{L}}(\boldsymbol{f};\boldsymbol{\theta}_t) &= \sum_{p\in\mathcal{I}} \tilde{\boldsymbol{y}}_p^T \log \boldsymbol{f}(\tilde{\boldsymbol{x}};\boldsymbol{\theta}_t)\\ &= -\lambda \sum_{p\in\mathcal{I}} \boldsymbol{y}_p^T \log \boldsymbol{f}(\tilde{\boldsymbol{x}};\boldsymbol{\theta}_t) - (1-\lambda) \sum_{q\in\pi(\mathcal{I})} \boldsymbol{y}_q^T \log \boldsymbol{f}(\tilde{\boldsymbol{x}};\boldsymbol{\theta}_t),\end{aligned} \tag{3.19}$$

where π is a permutation mapping of the ordered index set $\mathcal{I} = \{1, ..., |\mathcal{D}|\}$.

Figure 3.3 shows that the mixed-up cross-entropy loss function maintains a high degreee of loss separation between clean and noisy samples. This implies that the model is less prone to overfit the noise, and this can be observed by the much improved generalization on the clean test set.

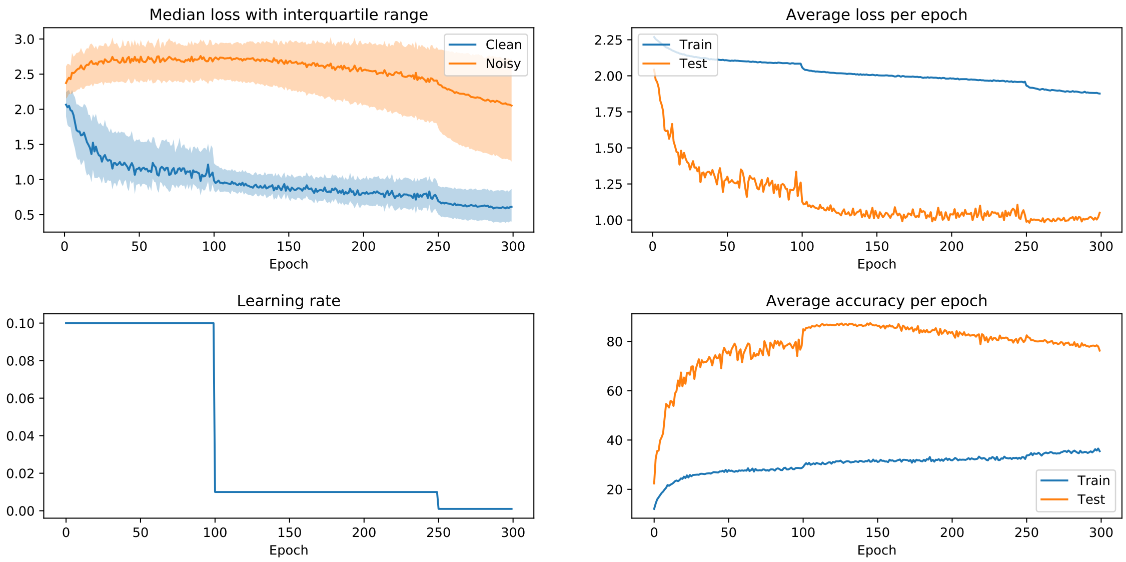

Figure 3.3: Experiment results from a PreAct ResNet-18 with mixed-up cross-entropy loss (3.19) on CIFAR-10 with with 50% symmetric label noise.

3.3.2 Loss correction with bootstrap loss

The *bootstrap loss* (Reed et al., 2014) avoids overfitting noisy labels through convex combinations of original labels and pseudo-labels based on the current state of the model, such that

$$\mathcal{L}(\boldsymbol{f}, \boldsymbol{z}; \boldsymbol{\theta}_t) = -\frac{1}{N}\sum_{i=1}^{N} [(1-\omega_i)\boldsymbol{y}_i + \omega_i \boldsymbol{z}(\boldsymbol{x}_i;\boldsymbol{\theta}_t))]^T \log(\boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{\theta}_t)). \tag{3.20}$$

Here $\boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{\theta}_t)$ is the per sample class-conditional probability, $\boldsymbol{y}_i$ is a potentially noisy original label, $\boldsymbol{z}_i^t = \arg\max_j \boldsymbol{f}_j(\boldsymbol{x}_i;\boldsymbol{\theta}_t)$ is a pseudo-label derived from the softmax prediction, and $w_i \in [0, 1]$ is a sample weight which puts emphasis on either the original label or the predicted pseudo-label. The aim is that clean samples train with emphasis on the original label $\boldsymbol{y}_i$ and noisy samples on the predicted label $\boldsymbol{z}(\boldsymbol{x}_i; \boldsymbol{\theta}_t)$.

The authors (Arazo et al., 2019) have proposed a significant improvement to the original bootstrap loss by dynamically assigning the predictions of the beta mixture model (3.15) to the weights such that $\omega_i = p(k = 1|l_i)$. The noise model gets updated after each epoch to reflect the changes of the sample losses during training.

It is also possible to combine the label selection mechanism of the bootstrap loss together with mixup. Similar to (3.19), the mixed-up bootstrap loss takes the form

$$\begin{aligned} \tilde{\mathcal{L}}(\boldsymbol{f}; \boldsymbol{\theta}_t) = -\lambda \sum_{p \in \mathcal{I}} \left[(1-\omega_p)\tilde{\boldsymbol{y}}_p + \omega_i \boldsymbol{z}(\boldsymbol{x}_p; \boldsymbol{\theta}_t)\right]^T \log \boldsymbol{f}(\tilde{\boldsymbol{x}}; \boldsymbol{\theta}_t) \\ - (1-\lambda) \sum_{q \in \pi(\mathcal{I})} \left[(1-\omega_q)\tilde{\boldsymbol{y}}_q + \omega_i \boldsymbol{z}(\boldsymbol{x}_q; \boldsymbol{\theta}_t)\right]^T \log \boldsymbol{f}(\tilde{\boldsymbol{x}}; \boldsymbol{\theta}_t), \end{aligned} \tag{3.21}$$

where π is a permutation mapping of the ordered index set $\mathcal{I} = \{1, ..., |\mathcal{D}|\}$, and $\omega_p, \omega_q \in [0, 1]$ are the corresponding weights for the labels $\tilde{\boldsymbol{y}}_p, \tilde{\boldsymbol{y}}_q \in \mathcal{D}$.

Figure 3.4 shows the effects of training with mixed-up cross-entropy loss for a warm-up period of 105 epochs and then proceeding to train with mixed-up bootstrap loss until the end of the training period.

The increased separation between clean and noisy losses makes it easier for the beta mixture model to identify and re-weight noisy labels in the bootstrap loss. This allows the model to iteratively adjust towards the correct labels which further increases robustness and performance on the test set.

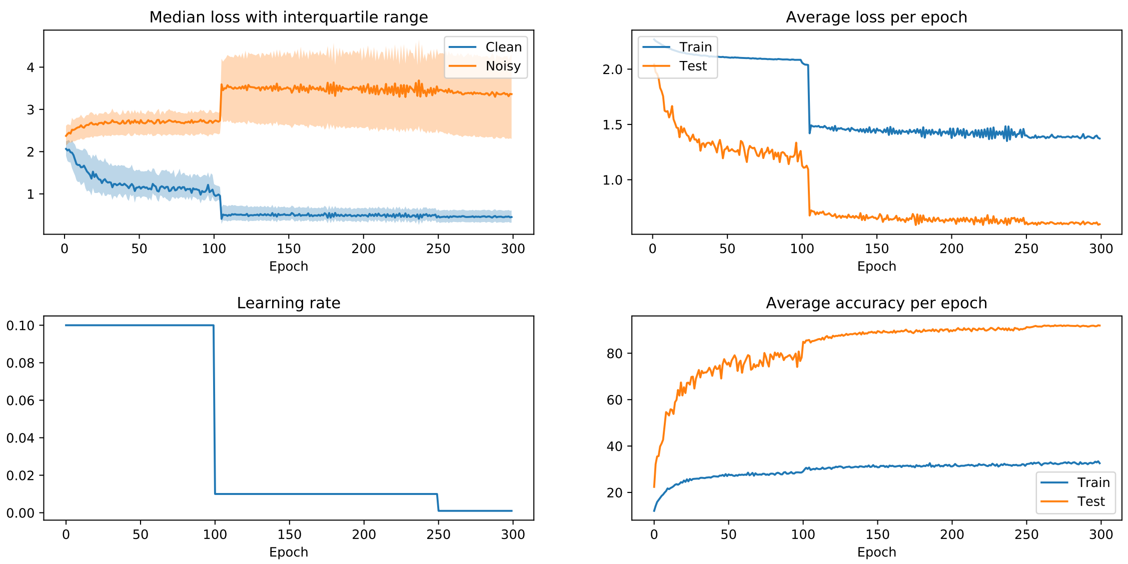

Figure 3.4: Experiment results from PreAct ResNet-18 with mixed-up bootstrap loss (3.21) and the beta mixture model (3.21) on CIFAR-10 with with 50% symmetric label noise.

3.3.3 Forward loss correction with a noise transition matrix

Loss correction methods that involve a noise transition matrix T typically come in two flavours. T is used either to transform the predictions by $T^\top \boldsymbol{f}$ (called *forward* correction) in order to match the noisy labels, or to transform the noisy labels by $T^{-1}\tilde{\boldsymbol{y}}$ (called *backward* correction) to match the predictions, given that T is non-singular. It is shown in both cases (Patrini et al., 2016), that the the minimzer of the corrected loss under the noisy distribution is the same as the minimizer of the loss under the clean distribution. Thus robustness can be attained by the model learned with the corrected loss.

To show how forward correction works consider a sample with its corresponding noisy label $(\boldsymbol{x}, \tilde{\boldsymbol{y}} = \boldsymbol{e}^k) \in \tilde{D}$. Its class-conditional probability estimate after softmax is $\hat{p}(\tilde{\boldsymbol{y}}|\boldsymbol{x})$. Using the standard cross-entropy loss the per sample loss takes the form

$$\begin{aligned} l(\hat{p}(\boldsymbol{y}|\boldsymbol{x}), \boldsymbol{e}^k) &= -\log \hat{p}(\tilde{\boldsymbol{y}} = \boldsymbol{e}^k|\boldsymbol{x}) \\ &= -\log \sum_{j=1}^{|\mathcal{C}|} p(\tilde{\boldsymbol{y}} = \boldsymbol{e}^k|\boldsymbol{y} = \boldsymbol{e}^j)\hat{p}(\boldsymbol{y} = \boldsymbol{e}^j|\boldsymbol{x}) \\ &= -\log \sum_{j=1}^{|\mathcal{C}|} T_{jk}\, \hat{p}(\boldsymbol{y} = \boldsymbol{e}^j|\boldsymbol{x}), \end{aligned} \tag{3.22}$$

where T is the true noise transition matrix.

The cross-entropy loss becomes robust to label noise by using T to map supposedly correct sample predictions to match the noisy label $\tilde{\boldsymbol{y}} = \boldsymbol{e}^k$ and thus minimize the loss. However, T is not known in practise and the success of the loss correction is dependent on the ability of the estimated transition matrix $\hat{T}$ to map correct predictions to noisy labels.

3.3.4 Label correction by self-adaptive training

Self-adaptive training (Huang et al., 2020) is a label correction method which gradually updates the original labels $\tilde{\boldsymbol{y}} \in \mathcal{Y}$ with the aim to alter the training dynamic and correct noisy labels.

The training schedule is divided into two phases. In the initial warm-up phase the model trains on the original labels to capture informative signals from the corrupted training set. The second phase begins to accumulatively update the original labels by incorporating model predictions in an exponential moving average

$$\tilde{\boldsymbol{y}}_i \leftarrow \alpha\tilde{\boldsymbol{y}}_i + (1-\alpha)\boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{\theta}_t), \tag{3.23}$$

where α is a weight parameter that controls the influence of the softmax prediction. The updating scheme smoothens the labels and allows them to change cumulatively if the model predictions remain confident.

It is possible to tune-out noisy samples during training by re-weighting the sample losses by their corresponding label confidence

$$\omega_i = \max_j \tilde{\boldsymbol{y}}_{ij} \tag{3.24}$$

such that the objective function to minimize takes the form

$$\mathcal{L}(\boldsymbol{f};\boldsymbol{\theta}_t) = -\frac{1}{\sum_{i=1}^{N}\omega_i}\sum_{i=1}^{N}\omega_i\tilde{\boldsymbol{y}}_i \log \boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{\theta}_t). \tag{3.25}$$

Samples are treated equally in the warm-up phase of the training schedule. Then as the labels are updated the training focuses on the confident samples. Noisy samples get reconsidered once their label confidence gets sufficiently high.

4

Experiments

This chapter contains details of experiments with deep convolutional neural network classifiers trained on CIFAR-10 with various levels of label noise. The classifiers are based on the learning methods presented in section 3.3.

4.1 Models

The following classifier models are evaluated.

- CE is a model with standard cross-entropy loss (2.2).
- CE-DBL (Arazo et al., 2019) is loss correction model based on the cross-entropy loss (2.2), the bootstrap loss (3.20) and the beta mixture model (3.15). This model is trained in two phases: a warm-up phase with cross-entropy loss and a loss correction phase with the bootstrap loss where the weights are dynamically assigned during training by a beta mixture model. The mixture model is refitted each epoch on the new loss distribution after the model has been trained.
- CE-SAL (Huang et al., 2020) is a label correction model based on the cross-entropy loss (2.2) and the self-adaptive loss (3.25). This model is trained in two phases: a warm-up phase with cross-entropy loss and a label correction phase with self-adaptive loss.
- M (Zhang et al., 2017) is a model based on the mixed-up cross-entropy loss (3.19).
- M-DBL (Arazo et al., 2019) is a loss correction model based on the mixed-up cross-entropy loss (3.19), the mixed-up bootstrap loss (3.21) and the beta mixture model (3.15). This model in trained in two phases: a warm-up phase with mixed-up cross-entropy loss and a loss correction phase with mixed-up bootstrap loss where the weights are dynamically assigned during training by a beta mixture model the same way as CE-DBL.
- M-DBL-RC (Arazo et al., 2019) is the same as M-DBL with the addition of class entropy regularization (3.2).
- AUM (Pleiss et al., 2020) is the implementation of the area under the margin ranking (3.17). A subset of $\tilde{\mathcal{D}}$ is selected to be indicator samples. Then a CE model is trained on the modified dataset until it begins to fit the noise. This procedure is repeated once more with a different subset of indicator samples such that AUM rankings for the entire dataset are obtained. The indices of noisy samples are subsequently identified.

- AUM-CE is the CE model trained on the cleaned subset filtered by AUM score.
- AUM-CE-SAL is the CE-SAL model trained on the cleaned subset filtered by AUM score.
- AUM-CE-FCL (Hendrycks et al., 2018) is a loss correction model based on the cross-entropy loss (2.2) and the forward correction loss (3.22). Two subsets of the original training set are initially obtained by AUM rankings: a clean subset $\mathcal{D}$ and a noisy subset $\tilde{\mathcal{D}}$.

  The training proceeds in three phases. In the first phase a CE model is trained on the noisy dataset $\tilde{\mathcal{D}}$ such that it becomes an estimator of noisy labels. The model is used in the second phase to estimate a noise transition matrix (3.7) by predicting the class-conditional probabilities of samples on the clean dataset $\mathcal{D}$. In the third phase, a new model is initialized and trained on the clean dataset with cross-entropy loss and on the noisy dataset with forward loss correction using the estimated noise transition matrix.

4.2 Dataset and implementation details

The models are trained and tested on the CIFAR-10 dataset (Krizhevsky et al.) which contains 60K color images with a resolution of 32×32 pixels. The training set contains 50K images and the remaining 10K images are for validation. The images are evenly distributed across 10 classes. The network architecture is a PreAct ResNet-18 (He et al., 2015) which is optimized using stochastic gradient descent (SGD). All models are implemented in PyTorch (Paszke et al., 2019) and are trained using CUDA (NVIDIA et al., 2020) on a single Tesla T4 GPU.

The preprocessing steps, hyperparameter settings and training schedules are mostly based on (Arazo et al., 2019), with a few exceptions.

Random seed The seed for all random number generators is set to 1 for all experiments. Only one seed was used in experiments due to time constraints.

Preprocessing All images are augmented by 32×32 random crops with 4-pixel padding, random horizontal flipping, and normalization.

Synthetic label noise is applied such that a percentage of labels are randomly relabeled. Relabeling is done with replacement which means that there is a chance that the true label could be kept. Only symmetric label noise was tested in experiments due to time constraints.

Optimizer Losses are optimized using stochastic gradient descent (SGD) with momentum (0.9), weight decay (10^{-4}) and training batch size of 128 samples.

Training schedule without mixup All models that are without mixup (except AUM and CE-SAL-COSINELR) are trained for 120 epochs. The initial learning rate is (0.1) which is reduced by a factor of 10 after 30, 80 and 110 epochs. Warm-up lasts for 30 epochs.

Training schedule with mixup All models with mixup are trained for 300 epochs. The initial learning rate is (0.1) and gets reduced by a factor of 10 after 100 and 250 epochs. Warm-up lasts for 105 epochs. The mixup parameter is $\alpha = 32$ in all use-cases. The class entropy regularization coefficient is set to $\lambda = 1$.

Training schedule for AUM AUM is performed by training a CE model for 100 epochs with a fixed learning rate of (0.1). The batch size is 64 to increase variance in SGD which reduces memorization.

Training schedule for CE-SAL-COSINELR The model is trained for 300 epochs. The initial learning rate is (0.1) which is reduced by cosine annealing with a maximum range of 200 epochs. The batch size is 128. Warm-up lasts for 60 epochs.

4.3 Results and discussion

The following results are collected from experiments conducted under the conditions explained in section 4.2. Due to time costraints and the number of models that were reimplemented and tested, there is only one accuracy score recorded for each model and level of label noise in the dataset. The numbers should be interpreted with this knowledge in mind.

Table 4.1 shows the accuracy score for the models that were trained without mixup. The gap between the best and last accuracy score indicates whether a model overfits the noise during later stages of training. Results show that the CE model with the standard cross-entropy loss is clearly not robust and fits the noise.

However, the other models show that they are indeed robust to label noise and display improved performance compared to the baseline CE model.

Table 4.1: Validation accuracy on CIFAR-10. Key: CE (cross-entropy), DBL (dynamic bootstrap loss), SAL (self-adaptive loss), H (hard labels), S (soft labels).

Noise level (%)		20	50	80	Avg. runtime
CE	Best	89.69	84.73	66.19	2.77h
	Last	82.51	55.78	23.95	
CE-DBL-S	Best	90.1	84.88	**70.14**	3.2h
	Last	83.68	71.05	**70.12**	
CE-DBL-H	Best	89.8	84.61	67.84	3.25h
	Last	83.64	67.14	60.95	
CE-SAL-S	Best	**91.59**	**87.68**	69.0	2.71h
	Last	**91.47**	**87.47**	68.94	

Table 4.2 shows the performance results for the models that have been trained with mixup regularization (with the exception of CE-SAL-S-COSINELR). The significance of mixup is noted by looking at the much improved accuracy scores and reduced score gap between the best and last epoch.

Note that the M-DBL-H model completely degenerates and predicts only one class at the final epoch. This is remedied by the class entropy regularization which consequently gives a huge performance boost when computing the loss with hard labels. The same regularization appears not to have the same effect when training with soft labels.

The CE-SAL-S-COSINELR model doesn't use mixup regularization but is still included in Table 4.2 because it is trained for the same duration as the other models. Its performance is comparable to M-DBL-S while being less complex to implement and less time consuming to train.

Table 4.2: Validation accuracy on CIFAR-10. Key: M (mixup), DBL (dynamic bootstrap loss), SAL (self-adaptive loss), RC (class entropy regularization), H (hard labels), S (soft labels).

Noise level (%)		20	50	80	Avg. runtime
M	Best	93.0	87.43	70.31	7.12h
	Last	92.74	76.27	48.04	
M-DBL-S	Best	94.14	90.71	76.88	8.9h
	Last	93.97	89.97	74.04	
M-DBL-H	Best	93.85	91.93	67.6	8.75h
	Last	93.76	91.8	10.0	
M-DBL-RC-S	Best	**94.23**	90.62	78.63	8.83h
	Last	**94.06**	89.88	74.84	
M-DBL-RC-H	Best	94.11	**92.06**	**87.19**	8.9h
	Last	93.98	**91.93**	**86.88**	
CE-SAL-S-COSINELR	Best	93.37	89.28	74.13	5.45h
	Last	93.19	89.28	74.05	

Table 4.3 shows the effectiveness of cleansing a mislabeled dataset by thresholding with AUM ranking. Both precision and recall are consistently high across different levels of label noise. However, the percentage of noise remaining in the filtered dataset increases with higher noise proportions in the training set. This may be caused by an increased likelihood of fitting noisy samples early on during training since the number of clean samples are small. Such samples may then end up on the right side of the threshold value.

Table 4.4 shows the results of training on the smaller and cleaner subset that remains after removing samples that have been identified as noisy or difficult by AUM ranking. The tight gap between the best and last epoch for the AUM-CE model indicates that the subset has less label noise than before. On the other hand, the accuracy scores on the test set are comparatively lower than the mixup based models that were trained on the full training set with noisy labels. This could be because there are samples missing in the reduced training set which might help the models perform better on the test set.

The AUM-CE-SAL-S model doesn't perform any better than AUM-CE on the reduced training set. This is not surprising because robustness is partly achieved by underfitting the data. Thus the model has no edge when the proportion of label noise is negligible.

The AUM-CE-FCL model has a special training schedule compared to the other models in this line of experiments since it utilizes both the cleaner subset and

Table 4.3: Precision- and recall scores when identifying mislabeled samples using area under the margin ranking (3.17) on CIFAR-10 at different proportions of label noise. The following rows give additional details about the sizes of the resuting subsets and their proportions of noisy or clean samples.

Noise level (%)	20	50	80	90
AUM Precision	0.933	0.966	0.945	0.914
AUM Recall	0.939	0.928	0.917	0.870
Effective label corruption (%)	18	45	72	81
Filtered subset size	40947	28939	15058	11436
Corrupt subset size	9053	21607	34942	38564
Perc. noisy samples in filtered subset (%)	1.34	5.66	19.86	45.86
Perc. clean samples in corrupt subset (%)	6.38	3.39	5.53	8.61

the more noisy subset when training. The extra work pays off in marginally improved accuracy scores but it is not enough compared to the M-DBL type models which achieve the best results overall. The small performance gains raises doubts on whether the noise transition matrix is well estimated.

Table 4.4: Validation accuracy after training on AUM filtered subsets of CIFAR-10 (see Table 4.3). Key: AUM (area under the margin), CE (cross-entropy), SAL (self-adaptive loss), FCL (forward correction loss).

Noise level (%)		20	50	80	Runtime
AUM-CE	Best	91.82	88.05	72.58	0.75-1.8h
	Last	91.78	87.78	71.28	
AUM-CE-SAL-S	Best	91.59	87.68	69	0.75-1.8h
	Last	91.47	87.47	68.94	
AUM-CE-FCL	Best	**92.33**	**88.37**	**74.91**	2.6-3.7h
	Last	**92.32**	**88.27**	**74.16**	

5 Conclusion

Supervised learning methods designed to mitigate label noise have been reviewed under synthetic uniform label noise conditions.

Mixup has proven to be an effective regularization method which substantially increases robustness to label noise. It is easy to combine mixup with the standard cross-entropy- and the bootstrap loss. In the latter case, the combination of mixup and bootstrap loss yield positive synergy effects by increasing and maintaining the loss separation of clean and noisy samples throughout training.

Combining mixup, bootstrap loss and the beta mixture model results in an impressive model that is robust and can recover a large proportion of corrupt labels given high proportions of label noise. However, performance has to be traded for longer training times. It was also shown that the model is susceptible to get stuck in local minima and it degenerates without class entropy regularization to adjust model calibration. Class entropy regularization is most effective when the prior class-probability is well known and the classes are well balanced. This is often not the case in real-world datasets which makes this form of regularization less feasible to use. More general ways of improving model calibration ought to be explored.

The self-adaptive training method has shown remarkable robustness for such a simple implementation. Its performance is comparable to the soft-labeled dynamic bootstrap models. However, the careful label correcting procedure doesn't seem to have a natural extension with mixup regularization from which it could benefit. Attempts at such an implementation by the author did not yield satisfactory results.

The forward correction method with a noise transition matrix has shown good robustness but achieves only mediocre performance scores compared to the other models. Assuming that the transition matrix was not badly estimated, the ability of the model to recover corrupted labels was not as effective as the other models considering the amount of preprocessing and training time required.

The area under the margin ranking has proven to be an effective method of identifying difficult and noisy samples. However, its benefits diminish as the proportion label noise increases, causing a relatively large proportion of mislabeled samples to remain in the filtered subset. Despite this, all models that were trained on the filtered subset generalized well on the test set. Another drawback of this method is that it requires to run twice with different subsets of indicator samples. However, its simplicity makes it a good diagnostic and filtering tool in a data processing pipeline. It would seem a worthwhile task to devise a method that would increase the AUM ranking separation between clean and noisy samples even for high proportions of label noise. Such a method could significantly improve both precision and recall.

Bibliography

Arazo, E., Ortego, D., Albert, P., O'Connor, N. E., & McGuinness, K. (2019). Unsupervised Label Noise Modeling and Loss Correction. *arXiv e-prints*, arXiv 1904.11238, arXiv:1904.11238 (cit. on pp. 1, 9, 14, 17, 18).

Carratino, L., Cissé, M., Jenatton, R., & Vert, J.-P. (2020). On Mixup Regularization. *arXiv e-prints*, arXiv 2006.06049, arXiv:2006.06049 (cit. on p. 8).

Frenay, B., & Verleysen, M. (2014). Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(5), 845–869 (cit. on p. 1).

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning* [http://www.deeplearningbook.org]. MIT Press. (Cit. on p. 7).

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *arXiv e-prints*, arXiv 1512.03385, arXiv:1512.03385 (cit. on p. 18).

Hendrycks, D., Mazeika, M., Wilson, D., & Gimpel, K. (2018). Using Trusted Data to Train Deep Networks on Labels Corrupted by Severe Noise. *arXiv e-prints*, arXiv 1802.05300, arXiv:1802.05300 (cit. on pp. 1, 9, 18).

Huang, L., Zhang, C., & Zhang, H. (2020). Self-Adaptive Training: beyond Empirical Risk Minimization. *arXiv e-prints*, arXiv 2002.10319, arXiv:2002.10319 (cit. on pp. 1, 15, 17).

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv e-prints*, arXiv 1502.03167, arXiv:1502.03167 (cit. on p. 7).

Krizhevsky, A., Nair, V., & Hinton, G. Cifar-10 (canadian institute for advanced research). http://www.cs.toronto.edu/~kriz/cifar.html (cit. on p. 18)

NVIDIA, Vingelmann, P., & Fitzek, F. H. (2020). Cuda, release: 10.2.89. https://developer.nvidia.com/cuda-toolkit. (Cit. on p. 18)

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., ... Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett, Eds.). In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32*. Curran Associates, Inc. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf. (Cit. on p. 18)

Patrini, G., Rozza, A., Menon, A., Nock, R., & Qu, L. (2016). Making Deep Neural Networks Robust to Label Noise: a Loss Correction Approach. *arXiv e-prints*, arXiv 1609.03683, arXiv:1609.03683 (cit. on p. 15).

Pleiss, G., Zhang, T., Elenberg, E. R., & Weinberger, K. Q. (2020). Identifying Mislabeled Data using the Area Under the Margin Ranking. *arXiv e-prints*, arXiv 2001.10528, arXiv:2001.10528 (cit. on pp. 2, 11, 17).

Reed, S., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., & Rabinovich, A. (2014). Training Deep Neural Networks on Noisy Labels with Bootstrapping. *arXiv e-prints*, arXiv 1412.6596, arXiv:1412.6596 (cit. on pp. 1, 13).

Rolnick, D., Veit, A., Belongie, S., & Shavit, N. (2017). Deep Learning is Robust to Massive Label Noise. *arXiv e-prints*, arXiv 1705.10694, arXiv:1705.10694 (cit. on p. 1).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html (cit. on p. 7)

Tanaka, D., Ikami, D., Yamasaki, T., & Aizawa, K. (2018). Joint Optimization Framework for Learning with Noisy Labels. *arXiv e-prints*, arXiv 1803.11364, arXiv:1803.11364 (cit. on pp. 1, 8).

Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *arXiv e-prints*, arXiv 1611.03530, arXiv:1611.03530 (cit. on p. 1).

Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2017). mixup: Beyond Empirical Risk Minimization. *arXiv e-prints*, arXiv 1710.09412, arXiv:1710.09412 (cit. on pp. 1, 7, 17).