Compressing Features for Learning With Noisy Labels

Yingyi Chen®, Shell Xu Hu, Xi Shen, Chunrong Ai, and Johan A. K. Suykens®, Fellow, IEEE

Abstract—Supervised learning can be viewed as distilling relevant information from input data into feature representations. This process becomes difficult when supervision is noisy as the distilled information might not be relevant. In fact, recent research shows that networks can easily overfit all labels including those that are corrupted, and hence can hardly generalize to clean datasets. In this article, we focus on the problem of learning with noisy labels and introduce compression inductive bias to network architectures to alleviate this overfitting problem. More precisely, we revisit one classical regularization named Dropout and its variant Nested Dropout. Dropout can serve as a compression constraint for its feature dropping mechanism, while Nested Dropout further learns ordered feature representations with respect to feature importance. Moreover, the trained models with compression regularization are further combined with co-teaching for performance boost. Theoretically, we conduct bias variance decomposition of the objective function under compression regularization. We analyze it for both single model and co-teaching. This decomposition provides three insights: 1) it shows that overfitting is indeed an issue in learning with noisy labels; 2) through an information bottleneck formulation, it explains why the proposed feature compression helps in combating label noise; and 3) it gives explanations on the performance boost brought by incorporating compression regularization into co-teaching. Experiments show that our simple approach can have comparable or even better performance than the state-of-the-art methods on benchmarks with real-world label noise including Clothing1M and ANIMAL-10N. Our implementation is available at https://yingyichen-cyy.github.io/CompressFeatNoisyLabels/.

Index Terms—Bias variance decomposition, compression, deep learning, information sorting, label noise.

The success of deep learning depends on the availability of massive and carefully labeled data. However, there is often no guarantee that all annotations are perfect, especially when the amount of data is huge and annotations are required to be fine such as optical flow and segmentation. In contrast, with the rapid development of the Internet, there are multiple ways to have inexpensive and convenient access to large but defective data, including querying commercial search engines [7], downloading images from social media [8], and various web crawling strategies [9]. Correspondingly, persistent efforts have been paid in literature to learn with imperfect data, among which learning with noisy labels has always been attached great significance.

The problem of learning with noisy labels dates back to [10] and [11]. The mainstream methods include: 1) training on reweighted samples [4], [12]–[15] where samples possibly clean are assigned larger weights than those possibly corrupted; 2) using robust loss functions to resist noise [16]–[19]; 3) conducting label correction [20]–[22] where original labels are often substituted by possible clean predictions; and 4) semisupervised learning methods [23]–[25] where samples are first identified as clean or corrupted, and then networks are trained in a semisupervised manner with only the clean labels used. Moreover, label noise itself also plays an important role in understanding the generalization puzzle of deep learning. Empirical experiments in [1] show that deep neural networks (DNNs), such as AlexNet [26], can achieve almost zero training errors on randomly labeled datasets. This analysis demonstrates that the capacities of DNNs are often high enough to memorize the entire noisy training information. Since overfitting is mainly due to the model capacities, an alternative way to address the problem of training with noisy labels is to introduce explicit compression inductive bias to the model architecture, which is the main focus of this work.

In this article, we propose to combat this overfitting problem by introducing compression inductive bias to networks. More precisely, rather than relying on the prediction of deterministic DNNs, we introduce feature compression to the hidden features in networks via Dropout [2] and its variant Nested Dropout [3]. Dropout can be served as a compression constraint for its feature dropping mechanism, while Nested Dropout further learns ordered feature representations with respect to feature importance. Leveraging Nested Dropout, we can not only constrain the model capacity but also filter out the irrelevance while preserving the relevance with respect to the learning task. Moreover, compared with Dropout, the information sorting property of Nested Dropout is particularly

I. INTRODUCTION
useful for conducting signal-to-noise separation in the feature level. Note that we may also consider other compression strategies such as principal component analysis (PCA) and kernel PCA (kernel PCA) [27], but Dropout/Nested Dropout is a plug-and-play component to networks, thus bringing much convenience to the implementation.

In addition to Dropout/Nested Dropout bringing feature-level compression to networks, we find that they are suitable for incorporating into co-teaching [4] which is a strong method for learning with noisy labels, for performance boost. Specifically, co-teaching trains two networks simultaneously where networks update themselves based on the small-loss mini-batch samples selected by their peer. Intuitively, this sample selection mechanism discards samples with possibly wrong labels and preserves those that are possibly clean. We will show in this article that sample selection during the cross-update process together with compression techniques will further prevent networks from overfitting the noisy labels. On account that good performance of co-teaching requires the two base networks to be reliable enough, we propose our two-stage method: 1) train two Dropout/Nested Dropout networks separately until convergence and 2) fine-tune these two networks with co-teaching.

Note that Dropout/Nested Dropout is maintained in the second stage for fine-tuning. The efficacy of our two-stage compression approach is validated on benchmark real-world datasets by achieving comparable or even better performance than the state-of-the-art approaches. For example, on Clothing1M [5], our method obtains 75.0% in accuracy, which achieves comparable performance to DivideMix [25] and ELR+ [28]. On ANIMAL-10N [6], we achieve 84.5% in accuracy while the state-of-the-art method PLC [22] is 83.4%.

Beyond the empirical contributions, we provide theoretical explanations on why compression can combat label noise. In particular, we conduct bias variance decomposition of the objective function where Dropout/Nested Dropout is formulated into latent variable model. This decomposition provides the following three insights.

1) It shows that overfitting is indeed an issue in learning with noisy labels. The bias term determines how close the model fits noisy labels, while the variance term promotes a consensus among individual models in the latent variable model. Deterministic DNNs have zero variance term and thus focus on minimizing the bias term during training, leading to over-fitting on noisy labels.

2) Through an information bottleneck formulation, it explains why the proposed feature compression helps in combating label noise. Dropout/Nested Dropout can serve as compression constraints since they can be formulated as optimizing an information bottleneck. These compression constraints bring nonzero variance term and thus reduce the impact of the bias term.

3) It explains the performance boost brought by incorporating compression regularization into co-teaching. The cross-update strategy of co-teaching together with the compression constraints bring larger variance term to further diminish the influence of the bias term, leading to even less overfitting on the noisy labels.

This article is based on our previous work [29] which mainly focuses on the empirical results. We enrich it with theoretical understanding of our method, the learning with noisy labels problem itself, and more detailed numerical assessments. This article is structured as follows. Section II summarizes the related works in learning with noisy labels. Section III presents our algorithm. Section IV provides a theoretical understanding of our method. Section V shows illustrative toy example and experiments on benchmark real-world datasets. Finally, we conclude this article in Section VI. Implementation is available at https://yingyichen-cy.github.io/CompressFeatNoisyLabels/.

II. RELATED WORKS

In this section, we briefly review the existing works related to learning with label noise. Extensive literature reviews can be found in [30]–[32].

A. Overfitting Prevention

The idea of preventing networks from overfitting for better generalization has been considered in [33] and [34]. In particular, Arpit et al. [33] propose that appropriately tuned explicit regularization prevents DNNs from overfitting noisy datasets while maintaining generalization on clean data, and Ma et al. [34] propose to understand the generalization of DNNs by investigating the dimensionality of the deep representation subspace of training samples. C2D [35] uses self-supervised pretraining to learn more meaningful information before overfitting to noise. ELR/ELR+ [28] proposes an early-learning regularization to resist overfitting, while AugDesc [36] achieved this using different augmentation strategies. Although starting from the point of preventing networks from overfitting, our method is different from their works mainly in that: 1) we theoretically verify that overfitting is indeed an issue by conducting bias variance decomposition while [33]–[36] are more from an experimental perspective and 2) we inject extrinsic compression to filter out noisy information, while [34] identifies network’s intrinsic compression point and adapts the corresponding loss.

B. Samples Reweighting

The samples’ reweighting scheme learns to assign small weights to those samples supposed to be corrupted. ActiveBias [37] reweights samples based on the variance of prediction probabilities and the closeness between the prediction probabilities and the decision threshold. MentorNet [12] trains its student network based on the clean samples selected by its teacher network. Co-teaching [4] cross-updates its two base models based on the small-loss samples selected by their peers. Decoupling [13] updates the networks based on samples where the predictions of the two predictors are different, that is, the “disagreement” strategy. As for co-teaching+ [14], it combines co-teaching with the “disagreement” strategy to further improve the performance. Different from the above where networks are based on “disagreement,” JoCoR [15] trains two networks as a whole by a joint loss following the “agreement” strategy and selects the small-loss examples to update themselves. This “agreement” strategy shows improvement over previous methods. Note that we still base our method on co-teaching since it is easier and also effective for both implementation and analysis. Pleiss et al. [38] propose a statistic, namely, Area Under the Margin (AUM),
which differentiates clean samples from mislabeled samples by exploiting their training dynamics. The mislabeled ones are discarded during training. Reference [38] is categorized here since discarding samples is equivalent to assigning zero weights to them.

C. Robust Loss Function

Robust losses have been applied to achieve noise-tolerant classifications including ramp loss [39], unhinged loss [40], and mean absolute error [27], [41], [42]. However, the fact that DNNs can learn arbitrary labels may dampen the effectiveness of these losses in the context of deep learning. In deep learning, losses are corrected to be robust to noisy samples, or more exactly, to eliminate the influence of noisy samples. Based on the estimated noise transition matrix, forward and backward [16] modify the loss function and build an end-to-end framework. HOC [43] recently proposes to focus on clusterable feature representations so as to efficiently estimate noise transition matrix and further conduct better loss correction. Other loss correction strategies include [18] and [19]. Different from these methods, we use the cross-entropy loss albeit adapted it for latent variable models.

D. Label Correction

JO [20] is a joint optimization framework where network parameters and class labels are optimized alternatively in training. Inspired but quite unlike [20], rather than correcting labels using the running average of network predictions, PENCIL [21] corrects labels via an updating label distribution during training. SELFIE [6] selects refurbishable samples which are of low uncertainty and can be corrected with a high precision, and then replaces their labels based on past model outputs. These corrected samples together with other low-loss instances are later used to update the network. Another state-of-the-art method named PLC [22] focuses more on feature-dependent label noise where labels are progressively corrected based on the confidence of the noisy classifier. Notably, we keep using all the labels including those noisy ones instead of conducting label correction which is more complicated.

E. Semisupervised Methods

In [23], a two-stage method is proposed where samples are identified as clean or corrupted in the first stage, and then networks are trained in a semisupervised manner with only the clean labels used in stage two. Kong et al. [24] also conduct a similar two-stage method with Renyi entropy regularization used in stage two. DivideMix [25] is one of the state-of-the-art methods achieving high accuracy on real noisy datasets. Specifically, it dynamically divides training data into a labeled clean set and an unlabeled corrupted set, and then trains models on both the sets in a semisupervised manner with the improved MixMatch [44] strategy. It can be seen that these methods mainly differ in adopting different criteria for semisupervised learning step after dividing the training set into clean and corrupted subsets.

III. Method

In this section, we present our approach for learning with noisy labels. We start with recalling compression techniques including Dropout [2] and its structured variant named Nested Dropout [3] in Section III-A. Next, we combine them with one commonly accepted approach named Co-teaching [4] (see Section III-B) in Section III-C. The reason for this combination will be discussed in detail in Section IV-E.

A. Compression Regularizations

Here, we consider two compression regularizations that are plug-and-play modules, which can be inserted into common network architectures. For the sake of clarity, we summarize some necessary notations here. Let \( Z \in \mathbb{R}^{\text{channels} \times \text{height} \times \text{width}} \) be the hidden feature representation obtained by the feature network \( f, \ i.e., Z = f(X) \). Note that we set the number of channels to be \( K \) and leave out the rest for simplicity, that is, \( \mathbb{R}^K \). In this article, we treat compression methods as applying masks to the obtained feature \( Z \). In this manner, let \( M \in B \) be the feature mask where the space \( B \) can vary for different compression methods. The feature with mask applied is denoted by \( Z = M \odot Z \) where \( \odot \) is the elementwise product. Then, \( Z \) will be fed into subsequent network structures. The two compression methods are given with respect to their specific mask distributions as follows.

1) Dropout: Dropout [2] is one classical method for feature compression where each feature in the network layer it applies is dropped according to a Bernoulli distribution. The space of its feature mask \( B \) is defined by

\[
M := \{ M \in \mathbb{R}^{K \times \cdots} | \forall 1 \leq k \leq K, M_k \sim \mathcal{B}(p_{\text{drop}}) \}
\]

where \( \mathcal{B} \) is the Bernoulli distribution with \( M_k \) being either \( 1 \) or \( 0 \), and \( p_{\text{drop}} \) is the drop rate.

2) Nested Dropout: Nested Dropout [3] learns ordered representations with different dimensions having different degrees of importance. Although it is originally proposed to perform fast information retrieval and adaptive data compression, we find that it can properly regularize a network to combat label noise. In particular, while Nested Dropout is applied, meaningless representations can be dropped, which leads to a compressed network [45]. Considering above, these ordered representations can be adapted to learning with noisy labels since representations learned from noisy data are supposed to be meaningless. Consequently, Nested Dropout may serve as a strong substitute of Dropout.

To obtain an ordered feature representation, in each training iteration, we only keep the first \( k \) dimensional feature of \( Z \) and...
mask the rest to zeros, that is, \( M \in \mathcal{M} \) where
\[
\mathcal{M} := \left\{ M \in \mathbb{R}^{K \times -} \mid \forall k \sim C(p_1, \ldots, p_K)
\right\}
\forall 1 \leq i \leq k, M_i = 1 \text{ and } \forall k < i \leq K, M_i = 0 \tag{2}
\]
with 1 and 0 being all-ones and all-zeros tensors, respectively. Moreover, \( k \) is sampled from a categorical distribution denoted by \( C \) with the corresponding parameters as follows:
\[
\left\{ p_k \propto \exp \left( -\frac{k^2}{2\sigma_{\text{nest}}^2} \right) \right\} \quad \forall k = 1, \ldots, K \tag{3}
\]
where \( \sigma_{\text{nest}} \) is the major hyperparameter in Nested Dropout. In this case, smaller \( k \) is preferred if \( \sigma_{\text{nest}} \) is small. Moreover, though we could compute \( \mathbb{E}_{P_M}(Z) := \mathbb{E}_{P_M}(M \odot \tilde{Z}) \) with \( P_M \) being (3) exactly during inference, we find it more efficient to verify which \( k \) yields the best performance on the validation set, and then keep the model induced by \( k \) for testing.

Note that rather than treating Dropout and Nested Dropout merely as regularizations, we focus on their ability of inducting an ensemble of models, i.e., (6), which will be carefully discussed in Section IV-B. We underline this ensemble property in Fig. 1.

B. Co-Teaching

Co-teaching [4] is a baseline method for learning under label noise. It trains two deep networks with identical architecture, i.e., \( h_1 \) and \( h_2 \), simultaneously where each network selects its 100(1 - \( \lambda_{\text{forget}} \)) percent small-loss instances, leading to \( D_1 \) and \( D_2 \), respectively, where \( \lambda_{\text{forget}} \) is the forget rate. Note that \( \lambda_{\text{forget}} \) is a crucial hyperparameter in the co-teaching architecture. Networks update themselves based on the data subset selected by their peers. We provide an illustration for clarity in Fig. 2.

Co-teaching bases on the concept that small-loss instances are more likely to be clean [4], [12], [14], [20], [46]. Therefore, classifiers trained on them are supposed to be more resistant to noisy labels. However, one nonnegligible premise is that base models should be reliable enough to select samples which are indeed clean. To prevent constantly bad selections, a scheduling has been proposed in [4]. That is, co-teaching first keeps all the samples in the mini-batch, and then gradually decreases the sample size in \( D_1 \) and \( D_2 \) till the predefined \( N \)th epoch, after which the sample size used for training is kept fixed. Nevertheless, we experimentally find that the tuning of \( N \) is not stable since \( N \) varies with different levels of label noise. Therefore, rather than training co-teaching with random initialized base models and tuning on \( N \), we use well-trained models as initialization for better and stable performance.

C. Combination

Now we combine the compression regularizations with co-teaching in a two-stage manner: 1) train two

IV. Theoretical Analysis

This section provides the motivation and validity of our method. Notations and basic concepts in need are given in Section IV-A. We state that the key issue in combating label noise is to prevent networks from overfitting based on a bias variance decomposition in Section IV-B. To this end, we recall two compression regularizations which can be treated as implicit information bottleneck in Section IV-C. More on
Algorithm 1: Two-Stage Compression Training

**Input:** training data $D$ with size $|D|$, compression hyperparameters $(\sigma_{nest}, p_{drop})$, two initialized networks $h_1, h_2$, loss $L_q$ (8), forget rate $\lambda_{\mathrm{forget}}$.

**Ensure:** Either train with Dropout ($p_{\mathrm{drop}} > 0$) by (1), or Nested Dropout ($\sigma_{nest} > 0$) by (2), (3).

**while** $h_1, h_2$ not converge **do**

- Train $h_1, h_2$ independently on $D$ with loss $L_q$ under (Nested) Dropout;

**end**

while *Fine-tune with co-teaching* **do**

- Randomly separate mini-batch $D_m$ into two subsets: $D_{m1}, D_{m2}$ with $|D_{m1}| = |D_{m2}|$;
- $h_1$ selects $(1 − \lambda_{\mathrm{forget}})|D_{m1}|$ small-loss data $\bar{D}_{m1}$;
- $h_2$ selects $(1 − \lambda_{\mathrm{forget}})|D_{m2}|$ small-loss data $\bar{D}_{m2}$;
- Train $h_1$ on $\bar{D}_{m1}$, $h_2$ on $\bar{D}_{m2}$ independently with loss $L_q$ under (Nested) Dropout;

**end**

**Output:** $(h_1 + h_2)/2$

Nested Dropout is given in Section IV-D. Finally, we verify that the co-teaching combination leads to even less overfitting based on the bias variance decomposition in IV-E. For the sake of clarity, all the proofs in this section are given in the Appendix.

### A. Preliminaries

First, we formulate the problem of learning with noisy labels. Let $X \in \mathcal{X}$ be the input variable where $\mathcal{X}$ is the input feature space. We consider the data generation process for the training set

$$ x \sim p(x), \quad \epsilon \sim p(\epsilon), \quad y \sim p(y|x, \epsilon) \quad (4) $$

where $\epsilon$ is the noise occurring during labeling. In this manner, we denote by $Y \in \mathcal{Y}$ the contaminated label where $\mathcal{Y}$ is the corresponding signal space. The goal is to learn a model on the corrupted dataset for testing on clean data drawn from the same generative process except that $\epsilon \approx 0$.

Next, we cover some basic concepts in information theory [47]. The *entropy* gives the amount of information coded in a distribution or equivalently the uncertainty about a random variable, and it is defined as the average code length

$$ H_p(Y) := −\int_{\mathcal{Y}} p(y) \log p(y) \, dy $$

where $p(y)$ is a fixed probability measure on $\mathcal{Y}$. Similarly, *conditional entropy* gives the amount of information about one random variable given another random variable

$$ H_p(Y|X) := −\int_{\mathcal{Y}} \int_{\mathcal{X}} p(x, y) \log p(y|x) \, dx \, dy $$

where $p(x, y)$ is the joint probability measure on $\mathcal{X} \times \mathcal{Y}$ and $p(y|x)$ is the conditional p.d.f. The *cross entropy* with respect to a model distribution $q(y|x)$ is defined by

$$ H_{p,q}(Y|X) := \mathbb{E}_{p(x)} \mathbb{E}_{p(y|x)} [−\log q(y|x)] = \mathbb{E}_{p(x)} \mathbb{E}_{p(y|x)} [−\log p(y|x) + \log \frac{p(y|x)}{q(y|x)}] \geq H_p(Y|X) \quad (5) $$

which upper bounds the conditional entropy as in (5). Note that the inequality holds for that the Kullback–Leibler divergence, i.e., the second term, is always nonnegative. A related quantity is the *cross-entropy loss* $−\log q(y|x)$. The *mutual information* measures the statistical dependence between random variables $X$ and $Y$ by comparing their joint density with the product of each marginal density

$$ I(X; Y) := \int_{\mathcal{X}} \int_{\mathcal{Y}} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \, dx \, dy. $$

Note that since the mutual information is a function of $p(x, y)$, we modify the notation to $I_p(X; Y)$ for better emphasizing on the actual variable. Moreover, given a factorization $p(x, y) = p(x)p(y|x)$, we have

$$ I_p(X; Y) = H_p(Y) − H_p(Y|X) $$

which leads to another interpretation, that is, the reduction in uncertainty of $Y$ by knowing $X$. In addition, the *conditional mutual information* is defined as follows:

$$ I_p(X; Y|Z) := \int_{Z} \int_{\mathcal{Y}} \int_{\mathcal{X}} p(x, y, z) \log \frac{p(x, y|z)}{p(x|z)p(y|z)} \, dx \, dy \, dz. $$

Here and subsequently, we let $\mathbb{E}_{p(z|x)} \equiv \mathbb{E}_{p(z)}$ for the expected value of $X$ and $\mathbb{D}_{KL}(\cdot)$ for the Kullback–Leibler divergence. Besides, we denote the capital Roman alphabet for random variables or matrices and their lowercase for the values. Moreover, we denote by $Z_i$, the $i$th row if $Z$ is a matrix or the $i$th channel if $Z$ is a 3-D tensor. We also write $Z_{i,j}$ for the slice from the $i$th channel to the $j$th channel, and $\odot$ for the elementwise multiplication.

### B. Bias Variance Decomposition for Noisy Labels

Considering that deterministic networks are likely to overfit the noisy training set, we introduce an ensemble of models and rely on the intersection of these models to extract consistent information. The idea is that the information learned from the noise are less likely to be consistent across different models. This motivates us to consider the *latent variable model* since it can be treated as an ensemble of models

$$ q(y|x) := \int_{Z} q(y|z)q(z|x) \, dz \quad (6) $$

where $q(z|x)$ is the encoder, and $q(y|z)$ is the decoder or can even be an individual model induced by a particular instance of $z$. For practical reasons, we would like to use the existing network architectures, such as ResNet [48], to construct the encoder $q(z|x)$ and the decoder $q(y|z)$. Our strategy is to split an entire network architecture, e.g., a ResNet-18, into two parts as shown in Fig. 3. In this case, it is natural to take the second part plus a softmax layer to implement $q(y|z)$. The first part is, however, insufficient to implement $q(z|x)$ as we will discuss later.

Since the cross-entropy loss $−\log q(y|x)$ is intractable due to the integral in (6), we consider a surrogate quantity to $q(y|x)$ using Jensen’s inequality

$$ \tilde{q}(y|x) \propto \exp[\mathbb{E}_{q(z|x)} \log q(y|z)] \leq q(y|x) \quad (7) $$

and therefore we define the new loss function as the negative log-likelihood with respect to $\tilde{q}(y|x)$

$$ \tilde{L}_q(x, y) := \mathbb{E}_{q(z|x)} [−\log q(y|z)] \propto −\log \tilde{q}(y|x). \quad (8) $$
Based on (4), (7), and (8), we now derive the bias variance decomposition of the proposed latent variable with the new loss function under label noise as follows.

\[ E_{p(x,y)}[L_q(X, Y)] = E_{p(x)} \left[ D_{KL}(p(y|x)\|\tilde{q}(y|x)) + \text{bias} \right] + \text{const} \]  

where \( \tilde{q}(y|x) \propto \exp[E_{q(z|x)} \log q(y|z)] \) is the average or an ensemble of models.

Intuitively, the bias term in (9) determines how close the average model \( \tilde{q}(y|x) \) is to \( p(y|x) \) and \( p(y|x) \) is the conditional probability for the noisy \( Y \), while the variance term promotes a consensus among individual models. The variance term also serves as a regularization to combat label noise in the sense that the consensus downweights the influence of the incorrect labels. Unlike learning with clean data, we do not expect low bias as it indicates model’s overfitting to label noise. Instead, we rely on the variance term and early stopping to provide good training signals. In regard of above, the problem of learning with noisy labels can be simplified to how can we prevent models from overfitting the noisy training labels?

It is worth mentioning that a careful design of the encoder \( q(z|x) \) is necessary to make better use of the variance term. For instance, if we choose \( q(z|x) \) to be a Dirac delta function, i.e., a deterministic mapping from \( x \) to \( z \), the variance term is zero. As a result, the training will focus on minimizing the bias term, leading to an easy overfitting to the distribution of noisy labels. On the other hand, if the variance is too large, there will be little consensus among individual models, and therefore, no consistent information could be learned by the latent variable model. Thus in Section IV-C, we propose to design \( q(z|x) \) by incorporating a compression inductive bias for better combating label noise.

C. Compression Regularizations

We would like to create an information bottleneck for the latent variable model. In this manner, the noisy information can be filtered out systematically. To be specific, we propose to create an information bottleneck by masking and damping the output of the feature extractor \( f(X) \)

\[
\min_p -I_p(Y; Z) \\
\text{s.t. } Z = M \circ f(X), M \sim P_M
\]

where the distribution of \( M \), i.e., \( P_M \), is an extrinsic source of randomness, which is tuned on a held-out clean dataset. Here, \( M \) is also called mask as defined in (1) and (2).

However, here comes the question that why do not we use Tishby’s information bottleneck [49] directly for compression? As mentioned in [49], the relationship among the input \( X \), the label \( Y \), and the feature representation \( Z \) in the network is given by a Markov chain: \( Y \rightarrow X \rightarrow Z \). In this consideration, Tishby et al. [49] propose to learn a good feature representation by minimizing the weighted sum of the data fitting term \(-I_p(Y; Z)\) and the complexity term \(I_p(X; Z)\) with respect to the distribution \( p(x,y,z) \), that is,

\[
\min_p -I_p(Y; Z) + \beta I_p(X; Z).
\]

However, when learning with noisy labels, there is no clear causal relationship between \( Y \) and \( X \). Therefore, Tishby’s information bottleneck principle cannot be applied in this case. To be more specific, we may argue in terms of the information diagrams shown in Fig. 4. Fig. 4 (left) visualizes the Markov chain \( Y \rightarrow X \rightarrow Z \), which is called the Mickey Mouse I-diagram in [50] since \( I(Y; Z|X) = 0 \), which implies that \( I(X; Y; Z) = I(Y; Z) \). In this case, \( I(Y; Z) \) is always smaller than \( I(X; Z) \), hence we can prevent \( Z \)’s overfitting to \( Y \) by reducing \( I(X; Z) \). However, in general, \( I(Y; Z|X) \neq 0 \) as shown in Fig. 4 (middle), and we can find cases where \( I(X; Z) \) is small but \( Z \) overfits label noise. To conclude, the traditional information bottleneck (11) may not be effective when dealing with label noise.

Compared with Tishby’s IB (11), we discard the term \( I(X; Z) \) in (10) completely and rely only on the held-out clean dataset to remove noisy information. Since \( I_p(Y; Z) \) is intractable, we further adjust it to be computationally available. Instead of estimating the joint distribution \( p(x,y,z) \), we consider a surrogate joint distribution \( q(x,y,z) \approx p(x,y,z) \) and access to \( p(x,y) \) only through its samples. Note that the idea is similar to the variational information bottleneck in [51]. Specifically, we first identify that \(-I_p(Y; Z) = H_p(Y|Z) - H_p(Y)\) where \( H_p(Y) \) is a constant for the representation learning, and hence we approximate \( H_p(Y|Z) \) by

\[
H_p(Y|Z) \leq H_{p,q}(Y|Z) = E_{p(x,y,z)}[-\log q(y|z)] \\
\approx E_{p(x,y)}E_{q(z|x)}[-\log q(y|z)] = E_{p(x,y)}[L_q(X, Y)].
\]

Now, we rewrite our learning with noisy labels as follows:

\[
\min_q E_{p(x,y)}[L_q(X, Y)] \\
\text{s.t. } Z = M \circ f(X), M \sim P_M
\]

where \( L_q(x, y) := E_{q(z|x)}[-\log q(y|z)] \) as in (8). As such, we only need to learn the proposed \( q(z|x) \) and \( q(y|z) \). In particular, we propose an implicit parameterization for \( q(z|x) \) as specified in (12), which involves a feature extractor \( f(X) \) and an external random variable \( M \). We also proposed to learn \( q(z|x) \) and \( q(y|z) \) jointly by minimizing \( E_{p(x,y)}[L_q(X, Y)] \).
which will be optimized by stochastic gradient descent (SGD) as we only have access to the samples of $X$, $Y$, and $Z$.

1) Dropout: If we set $P_{M}$ to be a Bernoulli distribution as in (1), then (12) exactly covers Dropout. However, different from the original formulation in [2], we formulate Dropout into the framework of latent variable models with our loss (8). In this manner, Dropout is the baseline of incorporating compression inductive bias for combating label noise.

2) Nested Dropout: If we specify $M \in \mathcal{M}$ where $\mathcal{M}$ is (2), and $P_{M}$ as (3), then (12) exactly covers the Nested Dropout. More properties are provided in Section IV-D.

D. Nested Dropout

Nested Dropout is a variant of Dropout where the importance of each feature channel is sorted from high to low, while channels in the Dropout model are of equal importance. For better understanding of Nested Dropout’s sorting property, we theoretically work on it through mutual information.

Before presenting the theorem on the information sorting property, we need the following Assumption 1 where hidden features are supposed to be exchangeable [53].

Assumption 1: Let $X$ be the input, $f$ be the feature extractor, $d$ be the subsequent network structure including the classification head, and $Z = f(X) \equiv [\hat{Z}_{k}]_{k=1}^{K}$ be the hidden feature representation. The hidden feature representation $Z$ is exchangeable, and $d$ is also exchangeable. That is, for any permutation $\pi$, the model satisfies that

\begin{align}
(Z_{1}, Z_{2}, \ldots, Z_{K}) &\overset{D}{=} (\hat{Z}_{\pi(1)}, \hat{Z}_{\pi(2)}, \ldots, \hat{Z}_{\pi(K)}) \quad (13) \\
Y|Z_{1}, Z_{2}, \ldots, Z_{K} &\overset{D}{=} Y|\hat{Z}_{\pi(1)}, \hat{Z}_{\pi(2)}, \ldots, \hat{Z}_{\pi(K)} \quad (14)
\end{align}

where $\overset{D}{=}$ denotes equivalence in distribution.

We find this can serve as a valid assumption considering the network architectures. According to de Finetti’s theorem [54], a sequence of random variables $(Z_{1}, Z_{2}, \ldots)$ are infinitely exchangeable iff, $p(\hat{z}_{1}, \hat{z}_{2}, \ldots, \hat{z}_{n}) = \int \Pi_{n=1}^{n} p(\hat{z}_{i}|\theta) P(d\theta)$, for all $n \in \mathbb{N}_{+}$ and some measure $P$ on $\theta$. If we consider a simple multilayer perceptron (MLP) where $\hat{z} = f(x) = W_{1}x + b_{1}, \hat{y} = d(\hat{z}) = W_{2}\hat{z} + b_{2}$ and denote $\theta = [W_{1}, b_{1}, x]$. In this way, for any permutation of the hidden feature $Z_{\pi} := \pi(\hat{z})$, it can be obtained by $\pi(W_{1})\pi(x) + \pi(b_{1})$. As it requires to integrate all the possible $\theta$ to obtain $p(\hat{z})$ and $p(\hat{z}_{\pi})$, we then have $p(\hat{z}) = p(\hat{z}_{\pi})$. Since $\hat{y}_{\pi} = \pi(W_{2})\hat{z}_{\pi} + \pi(b_{2}) = \hat{y}, d$ is exchangeable. Moreover, a similar argument can be derived by considering the features after global average pooling in CNNs.

Theorem 2: Let $X$ be the input, $f$ be the feature extractor, and $\hat{Z} = f(X) \equiv [\hat{Z}_{k}]_{k=1}^{K}$ be the hidden feature representation. Suppose that the model satisfies Assumption 1. Then, we have $D = M_{K} \odot \hat{Z}_{K}$

\begin{align}
I(Y; \hat{Z}_{1}) &= \cdots = I(Y; \hat{Z}_{K}) \quad (15) \\
I(Y; Z_{1}) &\geq \cdots \geq I(Y; Z_{K}). \quad (16)
\end{align}

This sorting property is also discussed in [3], although from the perspective of $I(Y; Z_{1:k}) \leq I(Y; Z_{1:(k+1)})$ with $Z_{1:k} := [Z_{1}, Z_{2}, \ldots, Z_{k}]$. In addition to the theoretical analysis, we conduct an experiment on CIFAR-10 [52] using ResNet-18 to verify Theorem 2 empirically. We plot the empirical estimate of the variational lower bound of $I(Y; Z)$, i.e., $H(Y|Z)$, for $Z$ computed by (12) and $Z = \tilde{Z} = f(X)$. The comparison is shown in Fig. 5 where the one without Nested Dropout is tagged as baseline. A clear information sorting has been achieved compared with the baseline training.

E. Combination With Co-Teaching

In this section, we consider stage two in our method where two networks are further fine-tuned with co-teaching. Recall that during the cross-update state, one network selects its small-loss instances $D_{1}$ and sends them to its peer. Intuitively, the above process resembles the teacher and student mechanism where the teacher selects possibly clean instances for the student to learn. In this regard, let $q_{t}(y|x)$ be the teacher network and $q_{y}(y|x)$ in (6) be the student network. The sample selection mechanism only preserves those with small loss $-\log q_{t}(y|x)$, i.e., large $q_{t}(y|x)$. If we consider this selection with respect to probability together with the following student network training, we reformulate the student’s loss as:

\begin{align}
L_{s}^{t}(x, y) &:= q_{t}(y|x)L_{q}(x, y) \\
&= \mathbb{E}_{q(z|x)}[-q_{t}(y|x)\log q(y|z)] \quad (17)
\end{align}

where $q_{t}(y|x)$ represents $(x, y)$’s probability to be selected. Moreover, by regarding sample selection and student network training as a whole, we redefine student network’s decoder by $q_{co}(y|x, z) \propto \exp[q_{t}(y|x)\log q(y|z)]$, to distinguish it from the original student network’s decoder $q(y|z)$, we call $q_{co}(y|x, z)$ the taught student decoder. The following Theorem 3 gives the bias variance decomposition when networks are further fine-tuned with Co-teaching.

Theorem 3: Let $q_{t}(y|x)$ be the Co-teaching teacher network, and $q_{co}(y|x, z) = \exp[q_{t}(y|x)\log q(y|z)]/C_{t}(x, z)$ where $C_{t}(x, z) := \int_{\mathcal{Y}}\exp[q_{t}(y|x)\log q(y|z)]dy \leq 1$ be the taught student decoder. For the Co-teaching student loss $L_{s}^{t}$ defined in (17), the risk has a bias variance decomposition

\begin{align}
\mathbb{E}_{p(x, y)}[L_{s}^{t}(X, Y)] &\approx \mathbb{E}_{p(x)} \left[ D_{KL}(p(y|x)\|q_{co}(y|x)) \right. \\
&+ \mathbb{E}_{q(z|x)}[D_{KL}q_{co}(y|x)\|q_{co}(y|x, z))] + \text{const} \quad (18)
\end{align}
where \( \tilde{q}_{co}(y|x) \propto \exp[\mathbb{E}_{q(z|x)} \log q_{co}(y|x,z)] \) is the average or an ensemble of models. Moreover, by defining \( \alpha(y|x) := \mathbb{E}_{q(y|x)}[\exp[q(y|x)]]/\exp[q_{i}(y|x)] \), we then have the following conditions.

1) If \( \alpha(y|x) \leq 1 \), then
\[
D_{KL}(p(y|x)\|\tilde{q}_{co}(y|x)) \leq D_{KL}(p(y|x)\|\tilde{q}(y|x)). \tag{18}
\]

2) If \( \alpha(y|x) \leq C_1(x,z) \), then
\[
\mathbb{E}_{q(z|x)}D_{KL}(q_{co}(y|x)\|q_{co}(y|x,z))
\geq \mathbb{E}_{q(z|x)}D_{KL}(\tilde{q}(y|x)\|q(y|x)). \tag{19}
\]

**Remark 1:** The condition \( \alpha(y|x) \leq 1 \) equals to \( q_i(y|x) \geq \log[\mathbb{E}_{q(y|x)} \exp(q_i(y|x))] \) where the right-hand side measures the difference between a single training network \( \tilde{q}(y|x) \) and the teacher network \( q_i(y|x) \) of Co-teaching. The larger the difference, the smaller the value, and vice versa. Hence, to obtain smaller bias term than in (9), the sample selection of Co-teaching only chooses those \( (x,y) \) with large \( q_i(y|x) \) so as to meet the condition. As \( C_1(x,z) \leq 1 \) by definition, if further \( \alpha(y|x) \leq C_1(x,z) \), i.e., \( q_i(y|x) \geq \log[\mathbb{E}_{q(y|x)} \exp(q_i(y|x))/C_1(x,z)] \) with larger right-hand side value, then we will have larger variance term than that in (9).

Theorem 3 and Remark 1 together demonstrate that choosing samples with large \( q_i(y|x) \) during selection of Co-teaching leads to smaller bias term and larger variance term than those in (9). That is, the impact of the bias term can be even diminished. Consequently, the sample selection mechanism during Co-teaching’s cross-update process helps in further preventing networks from overfitting on noisy labels, thus achieving better performance on clean datasets.

### V. Experiments

In this section, we present our experimental results. First, we focus on how Dropout and Nested Dropout cope with the regression noise by a toy example in Section V-A. For better understanding of our methods, we assess them on real datasets albeit with synthetic label noise in Section V-B. In Section V-C, we compare our method with the state-of-the-art methods on two real-world datasets: Clothing1M [5] and ANIMAL-10N [6]. Finally, we conduct ablation study on ANIMAL-10N and Clothing1M in Section V-D.

#### A. Toy Example: A Simple Regression With Noise

This section provides an intuitive better understanding on the reason why Nested Dropout [3] and Dropout [2] are able to resist label noise. To this end, we give a simulated regression experiment. Specifically, we generate a dataset of noisy observations from \( y_i = x_i + \epsilon_i \) for \( i = 1, \ldots, 64 \) where \( x_i \) is evenly spaced between \([0, 10]\) and \( \epsilon_i \sim \mathcal{N}(0, 1) \) are i.i.d samples. We use an MLP consisting of three linear layers with input and output dimensions being \( 1 \rightarrow 64 \rightarrow 128 \rightarrow 1 \). Moreover, we add Rectified Linear Unit (ReLU) activations to all the layers except the last one. When training model with Nested Dropout/Dropout, we only apply it to the last layer of the MLP, and the corresponding model is denoted by MLP+Nested and MLP+Dropout, respectively. Note that we follow (3) where \( \sigma_{nest} = 200 \). Fig. 6 gives the results after 100k epochs. The drop ratio \( p_{drop} \) of Dropout varies in \([0.9, 0.7, 0.5, 0.3]\) where the compression ratio decreases. As in Fig. 6, MLP overfits the label noise, while MLP+Nested with the first \( k = 1 \), \( k = 10 \) channels recover the ground-truth \( y = x \) better. Nevertheless, MLP+Nested gradually overfits the label noise due to over-parameterization as the number of channels increases. As for MLP+Dropout, with \( p_{drop} \) decreasing, the models become overfitting the noisy labels. However, MLP+Nested with \( k = 1 \) still gives the best performance. To conclude, both the compression methods prevent networks from overfitting the noisy patterns. Notably, for MLP+Nested, the main data structure information is contained in the first few channels, while noisy information is likely to be encoded in channels toward the end.

#### B. Model Analysis on Synthetic Noise

1) **Datasets:** We evaluate our methods on CIFAR-10 [52] and CIFAR-100 [52] with synthetic label noise following [20], [25], and [57]. For the training data, we manually corrupt the label according to a transition matrix \( Q \) with \( Q_{ij} = \Pr(y = j | y_{clean} = i) \), \( i, j \in \{1, \ldots, C\} \) denoting the probability of flipping clean \( y_{clean} \) to noisy \( y \). One representative structure of the matrix \( Q \) is the symmetric flipping [40], that is, \( P(y = i | y_{clean} = i) = 1 - \tau, P(y \neq i | y_{clean} = i) = \tau/(C - 1) \) where \( C \) is the number of classes and \( \tau \) is called the noise ratio. The other representative structure is the asymmetric (or pair) flipping [16] where label mistakes only happen within very similar classes, and therefore should be tailored for different datasets. For example, in CIFAR-10, the asymmetric flippings follow: truck \( \rightarrow \) auto-mobile, bird \( \rightarrow \) airplane, deer \( \rightarrow \) horse, and cat \( \leftrightarrow \) dog. The probability is \( \tau \) for flipping from ground truth to inaccurate class, while \( 1 - \tau \) for the remaining uncorrupted.

2) **Implementation Details:** Our methods are implemented with PyTorch. Following previous works [25], [58], the experiments on CIFAR-10 and CIFAR-100 are with PreAct ResNet-18 [55] trained from scratch. In the first stage, we use SGD optimizer with a momentum of 0.9, a weight decay of \( 1e-4 \), an initial learning rate of 0.1, and a batch size of 128. We apply learning rate warm-up with 6000 iterations, and the number of epochs is 200 with learning rate decayed by 0.1 at 100 and 150 epochs. Mixup data augmentation [56] is adopted in the first stage for better performance as in [25] and [58]. We apply Dropout/Nested Dropout on the average pooled Conv5 features. In the second stage, two well-trained models are set as base models for co-teaching. The initial learning rate is \( 1e-3 \) and we still use SGD as optimizer. Moreover, \( \lambda_{forget} \) is tuned under different noise ratios, batch norm is frozen, and no warm-up is applied. The models are trained for 100 epochs with the learning rate decayed by 0.1 after 50 epochs. We set \( \sigma_{nest} = 50, p_{drop} = 0.5 \) for cases on CIFAR-10 and and \( \sigma_{nest} = 100, p_{drop} = 0.3 \) on CIFAR-100. Note that when training with Nested Dropout, we record the optimal number of channels \( k^* \) of the model and use only these first \( k^* \) channels when testing.

We further show that our methods can also serve as good complementary strategies to other state-of-the-art methods to achieve even better performance. In particular, we propose an additional data preprocessing step before training our methods, which is named “precleaning.” During this precleaning, for example, we substitute the original labels of the dataset with
the predictions of a well-trained DivideMix [25] model, resulting in a precleaned dataset. We later train our methods on this precleaned dataset so as to further exceed the performance of DivideMix. Note that we can use any state-of-the-art methods to conduct “precleaning.”

3) Results on CIFAR-10 and CIFAR-100 [52]: We compare our two-stage methods with multiple state-of-the-art methods on CIFAR-10 and CIFAR-100 under different types and levels of synthetic label noise in Table I. We consider the performance of our methods with and without the “precleaning” step separately. Without the precleaning step, our simple Nested+Co-teaching and Dropout+Co-teaching achieve the top three performance for all except the extreme 90% label noise ratio cases. Moreover, by precleaning with DivideMix, our methods achieve the best performance for all the cases. Note that we also compare with M-correction [58] and MLNT [57] with the precleaning step in Table I. It can be seen that although both M-correction and MLNT improve upon their own results with the help of precleaning, they fail to surpass the performance of DivideMix. Therefore, they cannot serve as complementary strategies to DivideMix to enhance performance. In contrast, our Dropout+Co-teaching improves upon DivideMix by a maximum of 5.9% in accuracy under 90% symmetric noise on CIFAR-100. We also consider the training and inference time of our methods. It takes 2.7 h for training the complete two-stage model on a single NVIDIA V100 GPU, and both Nested+Co-teaching and Dropout+Co-teaching take 0.24 ms/image for inference. In regard of above, our methods not only perform well on their own but also serve as effective and efficient complementary strategies to other state-of-the-art methods with only a little extra time. Moreover, we provide additional insight that using ImageNet [60] pretrained models as in Table II, and we can achieve even better performance.

C. Comparison With State-of-the-Art Methods on Real Datasets

1) Datasets: The following experiments are conducted on two real-world datasets with real label noise: Clothing1M [5]
and ANIMAL-10N [6]. Clothing1M is a benchmark dataset containing 1 million clothing images with 14 categories from online shopping websites, and its overall estimated noise is 38.5% according to [5]. Moreover, this dataset provides 50k, 14k, and 10k manually verified clean data for training, validation, and testing, respectively. Note that we do not use the clean training set during training. In our experiment, we randomly sample a balanced subset that includes 260k images with 18.5k images per category, from the noisy training set as in [21] and [22]. This balanced subset is used as our training set, and classification accuracies are reported on the 10k clean test data. We follow data augmentations in [25], [28], and [58], which includes Mixup data augmentation [56]. ANIMAL-10N is another benchmark dataset recently proposed by Song et al. [6]. It contains ten animal classes with confusing appearance. There are 50k training and 5k testing images, with an estimated label noise rate of 8%. No data augmentation is applied so as to follow the settings in [6].

2) Implementation Details: The following experiments are implemented on PyTorch. Experiments on Clothing1M [5] are with ResNet-50 [48] pretrained on ImageNet [60] following [21], [25], and [57]. Dropout/Nested Dropout is applied right after the linear classifier in the network with $p_{\text{nest}} = 250$, $p_{\text{drop}} = 0.5$. First, the SGD optimizer is used for stage one model training with momentum 0.9, weight decay 5e−4, initial learning rate 2e−2, and batch size 96. Learning rate warm-up is used for 6000 iterations in stage one, and the model is later trained for 30 epochs with the learning rate decayed by 0.1 after the tenth epoch. Second, the two models trained in stage one are fine-tuned through co-teaching in stage two. SGD optimizer is used with the same settings and follows a cosine learning rate decay [61] with a maximum learning rate of 0.001 and without learning rate warm-up. The forget rate $\lambda_{\text{forget}}$ is set to be 0.5, batch norms are frozen, and no warm-up is applied. The initial learning rate is $4e^{-3}$ and is decayed by 0.2 after the 5th epoch with 30 epochs in total.

3) Results on the Clothing1M [5]: We compare our methods with the state-of-the-art methods in Table III. Notably, a single model trained with Nested Dropout or Dropout can not only surpass M-correction [58], JO [20], ELR [28] but also achieve comparable performance to HOC [43] and PENCIL [21]. The combination of Dropout+Co-teaching boosts the performance of a single Dropout model by 1.1%. Moreover, the combination of Nested+Co-teaching boosts from 73 to 74.1%. Moreover, the combination of Nested+Co-teaching boosts from 73.2 to 74.0%.

The learning rate is later decayed by 0.2 at 50th and 75th epochs. Moreover, the models are trained with learning rate warm-up for 6000 iterations. In stage two, the forget rate $\lambda_{\text{forget}}$ is set to be 0.2, batch norms are frozen, and no warm-up is applied. The initial learning rate is $4e^{-3}$ and is decayed by 0.2 after the 5th epoch with 30 epochs in total.

3) Results on the Clothing1M [5]: We compare our methods with the state-of-the-art methods in Table III. Notably, a single model trained with Nested Dropout or Dropout can not only surpass M-correction [58], JO [20], ELR [28] but also achieve comparable performance to HOC [43] and PENCIL [21]. The combination of Dropout+Co-teaching boosts the performance of a single Dropout model by 1.1%. Moreover, the combination of Nested+Co-teaching boosts from 73.2% of a single model to 75.0%, achieving the best among all the methods.

4) Results on ANIMAL-10N [6]: Table IV gives the results on ANIMAL-10N. It can be seen that our single Dropout model can achieve comparable performance to SELFIE [6]. Moreover, the combination with co-teaching provides a consistent performance boost, which is in line with the results on Clothing1M [5]. Notably, our best performance using Dropout+Co-teaching achieves 84.5% accuracy outperforming the recent approach PLC [22] by 1.1%.

| Methods | Acc. (%) |
|---------|----------|
| Cross-Entropy [5] | 69.2 |
| M-correction [58] | 71.0 |
| ELR [28] | 72.9 |
| M-correction [58] | 73.4 |
| PENCIL [21] | 73.5 |
| MLNT [57] | 73.5 |
| PLC* [22] | 74.0 |
| DivideMix [25] | 74.8 |

**Table IV**

**Average Test Accuracy (%) With Standard Deviation (Three Runs) of State-of-the-Art Methods on ANIMAL-10N [6]. All Approaches Are Implemented With VGG-19 [62] Architecture. Results With “*” Use Two Networks for Training

| Methods | Acc. (%) |
|---------|----------|
| Nested | 81.3 ± 0.6 |
| Nested+Co-teaching* | 84.1 ± 0.1 |
| Dropout | 81.6 ± 0.2 |
| Dropout+Co-teaching* | 84.5 ± 0.1 |
D. Ablation Study With Real Label Noise

This section provides ablation study of \( \sigma_{\text{nest}} \), \( p_{\text{drop}} \), and \( \lambda_{\text{forget}} \) on ANIMAL-10N [6]. Note that same as many state-of-the-art methods [6], [25], [28], [57], our hyperparameters need to be tuned on a clean validation set. Moreover, we also evaluate our methods using different backbones on Clothing1M [5].

1) Ablation on \( \sigma_{\text{nest}} \): As in Table V(a), Nested Dropout provides consistent improvement compared with training with standard cross-entropy loss, and the performance gain is also robust to the choices of the hyperparameter \( \sigma_{\text{nest}} \). Moreover, fine-tuning through co-teaching provides clear performance boost for all the models. We also present the optimal number of channels of each model (entry "k^*"). Although there are two layers of Nested Dropout applied to the classifier of VGG-19, the optimal number of channels \( k^* \) is recorded with respect to the last Nested Dropout layer for simplicity. Interestingly, models trained with Nested Dropout achieve better performance with only less than 1% of channels compared with their counterparts with cross-entropy loss.

2) Ablation on \( p_{\text{drop}} \): We experiment on different \( p_{\text{drop}} \) with the results given in Table V(a). Note that the results with \( p_{\text{drop}} \geq 0.5 \) are not given in Table V(a) since the training of single VGG-19 fails on ANIMAL-10N [6]. The performance under different choices of \( p_{\text{drop}} \) is less robust compared with those of Nested Dropout. However, what is in common is that the combination with co-teaching again brings significant performance boost for all the models.

3) Ablation on \( \lambda_{\text{forget}} \): We focus on how forget rate \( \lambda_{\text{forget}} \) of co-teaching influences the performance in Table V(b).

We present the settings where Nested Dropout and Dropout have the best performance. To be specific, we set \( \sigma_{\text{nest}} = 100 \) and \( p_{\text{drop}} = 0.1 \) for training, respectively. For Nested+Co-teaching, the performance drops with \( \lambda_{\text{forget}} \) increasing, and the accuracy remains 83.3% for \( \lambda_{\text{forget}} \geq 0.5 \). The performance with \( \lambda_{\text{forget}} \geq 0.5 \) is not given in Table V(b) for simplicity. Moreover, this performance boost of 83.3% compared with a single Nested Dropout network actually results from the ensemble estimation, not the cross-update mechanism of co-teaching. Therefore, considering that the estimated label noise ratio of ANIMAL-10N is 8%, co-teaching’s core mechanism is not suitable for cases where the difference between \( \lambda_{\text{forget}} \) and the ground-truth noise ratio is large. Furthermore, performance of Dropout+Co-teaching again verifies the above analysis where the best performance is achieved by \( \lambda_{\text{forget}} = 0.2 \).

4) Ablation on Different Backbones: As given in Table VI, the best performance is achieved using ResNet-50 as backbone, which is 75.0% in accuracy. Even though EfficientNet-B2 [64] cannot achieve the best performance possibly due to its capacity issue, Nested+Co-teaching indeed improves upon a single Nested Dropout model. The same applied to Dropout+Co-teaching. These results suggest the effectiveness of our approach with different backbones.

VI. CONCLUSION

In this article, we investigate the problem of image classification in the presence of label noise. In particular, we find that preventing networks from overfitting the corrupted labels is one key problem in learning with noisy labels based on a bias variance decomposition. To this end, we introduce compression inductive bias to networks to increase the variance term so as to weaken the influence of the bias term which is associated with overfitting. This inductive bias is realized by applying simple compression regularizations such as Dropout [2] and its variant named Nested Dropout [3] to networks. Notably, Nested Dropout is proven to learn ordered feature representations in this article. Therefore, this information sorting property can bring interpretability with respect to channel importance to networks while filtering out the noisy patterns. Moreover, we combine these compression regularizations with co-teaching [4], leading to a two-stage method. We then theoretically verify that this combination is in

\[ \lambda_{\text{forget}} \]

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
line with our bias variance trade-off since co-teaching further increases the variance term, hence further preventing networks from overfitting. Our method is validated on benchmark real datasets under synthetic label noise and real-world label noise including Clothing1M [5] and ANIMAL-10N [6]. Our method achieves comparable or even better performance than the state-of-the-art approaches. Our approach is simple compared with many existing methods. Therefore, we hope that our approach can serve as a strong baseline for future research on learning with noisy labels.

APPENDIX

This appendix provides the proofs of this article.

Proof of Theorem 1: Since $H(Y|X, e)$ is a constant with respect to the model distribution $q(y|x)$, for a given input $x$, we have

$$
E_{p(y|x)}\left[L_q(x, Y)\right] = E_{p(y|x)}E_{q(z|x)}\left[-\log q(y|z)\right] = E_{p(y|x)}E_{q(z|x)}\left[-\log q(y|z)\right] = q(y|x)\log q(y|x) - \log p(y|x, e) = E_{p(y|x)}D_{KL}(p(y|x)\|q(y|x)) + H(Y|X = x, e).
$$

(20)

The following result from [65, Th. 3.1] is in need.

Lemma 1: Let $\theta \sim \pi$ and $\omega \sim \pi'$ and let $f_\theta$ be a distribution with the same supports as a random probability distribution $P_\omega$ for all $\theta$ and $\omega$. Then, if $Y \sim \tilde{P} := E_{\omega}P_\omega$

$$
E_{\omega}E^D_{KL}(P_\omega\|f_\theta) = D_{KL}(\tilde{P}\|P) + E_{\omega}D_{KL}(\tilde{P}\|f_\theta) + I(Y; \omega)
$$

where $\tilde{P}(\cdot) \propto \exp[\tilde{E}(\log f_\theta(\cdot))]$.

Now by applying Lemma 1, we find that the first term in (20) can be decomposed as

$$
E_{p(y|x)}E_{q(z|x)}D_{KL}(p(y|x)\|q(y|z)) = D_{KL}(p(y|x)\|q(y|z)) + E_{q(z|x)}D_{KL}(\tilde{q}(y|x)\|q(y|z)) + I(Y; e)
$$

where $p(y|x) = E_{p(y|x)}p(y|x, e)$, and also $\tilde{q}(y|x) = \exp[E_{q(z|x)}\log q(y|z)]/\int \exp[E_{q(z|x)}\log q(y|z)] dy$. Note the proof follows if we plug in the above decomposition when computing $E_{p(y|x)}[L_q(X, Y)]$. Note that both $H(Y|X, e)$ and $I(Y; e)$ are constant with respect to the model $q(y|x)$. This completes the proof. □

Proof of Theorem 2: We split the proof into two parts.

1) Proof of (15): For the hidden representation $Z$, since we have (13) held, $\forall i, j \in \{1, \ldots, 2\}$, and $i < j$ without loss of generality, we interchange the $i$th and $j$th arguments

$$
p_{\hat{Z}_i}(\hat{z}_i) = \int \cdots \int p_{\hat{Z}_i, \hat{Z}_{i+1}, \ldots, \hat{Z}_K} (\hat{z}_1, \ldots, \hat{z}_i, \ldots, \hat{z}_j, \ldots, \hat{z}_K) \times d\hat{z}_1 \cdots d\hat{z}_{i-1}d\hat{z}_{i+1} \cdots d\hat{z}_K
$$

$$
= \int \cdots \int p_{\hat{Z}_i, \hat{Z}_{i+1}, \ldots, \hat{Z}_K} (\hat{z}_1, \ldots, \hat{z}_i, \ldots, \hat{z}_j, \ldots, \hat{z}_K) \times d\hat{z}_1 \cdots d\hat{z}_{i-1}d\hat{z}_{i+1} \cdots d\hat{z}_K = p_{\hat{Z}_j}(\hat{z}_j).
$$

Therefore, $p_{\hat{Z}_i}(\hat{z}_i) = p_{\hat{Z}_j}(\hat{z}_i)$ and $\hat{Z}_i$ and $\hat{Z}_j$ are identically distributed. Since $i$ and $j$ are arbitrarily chosen, we have $\{\hat{Z}_j\}_{j=1}^2$ identically distributed. Similarly, for the joint distributions of $Y, \hat{Z}_i$, and $Y, \hat{Z}_j$ with $i$ and $j$ arbitrarily chosen, we have

$$
p_{Y, \hat{Z}_i}(y, \hat{z}_i) = \int \cdots \int p_{Y, \hat{Z}_i, \hat{Z}_{i+1}, \ldots, \hat{Z}_K} (y, \hat{z}_1, \ldots, \hat{z}_i, \ldots, \hat{z}_j, \ldots, \hat{z}_K) \times d\hat{z}_1 \cdots d\hat{z}_{i-1}d\hat{z}_{i+1} \cdots d\hat{z}_K
$$

$$
= \int \cdots \int p_{Y, \hat{Z}_i, \hat{Z}_{i+1}, \ldots, \hat{Z}_K} (y, \hat{z}_1, \ldots, \hat{z}_i, \ldots, \hat{z}_j, \ldots, \hat{z}_K) \times d\hat{z}_1 \cdots d\hat{z}_{i-1}d\hat{z}_{i+1} \cdots d\hat{z}_K = p_{Y, \hat{Z}_j}(y, \hat{z}_i)
$$

where the third equivalence holds for the permutation invariant (13) and (14). Hence, $p_{Y, \hat{Z}_i}(y, \hat{z}_i) = p_{Y, \hat{Z}_j}(y, \hat{z}_i)$ for arbitrary $i, j$, that is, $(Y, \hat{Z}_i)_{i=1}^K$ are identically distributed.

Considering the formulation of mutual information, we have $\forall i, j \in \{1, \ldots, K\}$

$$
I(Y; \hat{Z}_i) := \int \int p_{Y, \hat{Z}_i}(y, \hat{z}_i) \log \frac{p_{Y, \hat{Z}_i}(y, \hat{z}_i)}{p_Y(y)p_{\hat{Z}_i}(\hat{z}_i)} dy d\hat{z}_i
$$

$$
= \int \int p_{Y, \hat{Z}_j}(y, \hat{z}_i) \log \frac{p_{Y, \hat{Z}_j}(y, \hat{z}_i)}{p_Y(y)p_{\hat{Z}_j}(\hat{z}_i)} dy d\hat{z}_j
$$

$$
= I(Y; \hat{Z}_j).
$$

This completes the proof of (15).

2) Proof of (16): By the definition of Nested Dropout in (2) and (3), we have

$$
Z = [\hat{Z}_{1k}, 0, \ldots, 0] \quad \text{with} \quad k \sim C(p_1, \ldots, p_K).
$$

We first rewrite the above calculation equivalently as

$$
T_k^i = \hat{Z}_i \quad \forall k = 1, \ldots, K
$$

$$
T_k^{i+1} = \begin{cases} 
T_k^i, & k \leq i \\
\epsilon_i T_k^i, & k > i 
\end{cases}
$$

$$
\forall i = 1, \ldots, K - 1 \quad \forall k = 1, \ldots, K
$$

where

$$
\epsilon_i = \begin{cases} 
b \sim Bernoulli(\eta_1), & \epsilon_{i-1} = 1 \\
0, & \epsilon_{i-1} = 0 
\end{cases}
$$

and $\epsilon_0 = 1$, $Z_k = T_k^K \quad \forall k = 1, \ldots, K$.

The above recursion can be solved and specified by

$$
Z_1 = \hat{Z}_1, \quad Z_k = \left(\prod_{j=1}^{k-1} \epsilon_j\right) \hat{Z}_k \quad \forall k = 2, \ldots, K
$$

which suggests that

$$
P(Z = [\hat{Z}_1, 0, \ldots, 0]) = P(Z_1 = \hat{Z}_1, Z_2 = 0) = P(\epsilon_1 = 0) = 1 - \eta_1
$$

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
and for $k = 2, \ldots, K - 1$, we have
\[
P(Z = [Z_{1:k}, 0, \ldots, 0]) = P(Z_k = \hat{Z}_k, Z_{k+1} = 0) = P(e_1 = 1, \ldots, e_{k-1} = 1, e_k = 0) = \left(\prod_{j=1}^{k-1} \eta_j\right) (1 - \eta_k)
\]
and also have
\[
P(Z = Z) = P(Z_{k-1} = Z_{k-1}, Z_K = Z_K) = P(e_1 = 1, \ldots, e_{K-1} = 1) = \prod_{j=1}^{K-1} \eta_j.
\]
Hence, the equivalence holds if
\[
p_1 \equiv 1 - \eta_1, \quad p_k \equiv \left(\prod_{j=1}^{k-1} \eta_j\right) (1 - \eta_k)
\]
for $k = 2, \ldots, K - 1$, and $p_K \equiv \prod_{j=1}^{K-1} \eta_j$.

More precisely, the equivalence holds if
\[
\eta_1 = 1 - p_1, \quad \eta_k = \frac{1 - \sum_{j=1}^{k} p_j}{1 - \sum_{j=1}^{k+1} p_j}
\]
for $k = 2, \ldots, K - 1$.

Given the construction of the network, we obtain a Markov chain $Y \to X \to T_k^1 \to \cdots \to T_k^k$, which simply means that $T_k^j$ is conditionally independent of $Y$ given $X$ or $T_j^i$ for $j < i$. Now, we need the following data processing inequality:

**Lemma 2 (Data Processing Inequality):** Let three random variables form the Markov chain $A \to B \to C$, meaning that $C$ is conditionally independent of $A$ given $B$. Then
\[
I(A; B) \geq I(A; C).
\]
The equality holds if and only if $I(A; B | C) = 0$.

Combining the above data processing inequality with both (15) and (21), we have
\[
I(Y; Z_{k+1}) = I(Y; T_{k+1}) \leq I(Y; T_k^k) = I(Y; Z_k).
\]

Note that $I(Y; T_{k+1}^k) = I(Y; T_k^k)$ is due to the fact that
\[
T_{k+1}^k = \prod_{j=1}^{k+1} e_j = \begin{cases} \hat{Z}_{k+1}, & \text{if } \prod_{j=1}^{k-1} e_j = 1 \\ 0, & \text{o.w.} \end{cases}
\]
and
\[
T_k^k = \prod_{j=1}^{k} e_j = \begin{cases} \hat{Z}_k, & \text{if } \prod_{j=1}^{k-1} e_j = 1 \\ 0, & \text{o.w.} \end{cases}
\]
and $I(Y; Z_{k+1}) = I(Y; \hat{Z}_k)$. This completes the proof.

**Proof of Theorem 3:** Recall that the taugh student decoder is defined by $q_{co}(y|x, z) = \exp[q_c(y|x) \log q(y|z)]/C_1(x, z)$, where $C_1(x, z) := \int_Y \exp[q_c(y|x) \log q(y|z)] dy$. For a given input $x$
\[
E_{p(y|x)}[L_q'(x, Y)]
\]
\[
= E_{p(y|x)} E_{q_{co}(y|x)}[-q_c(y|x) \log q(y|z)]
\]
\[
= E_{p(y|x)} E_{q_{co}(y|x)}[-\log q_{co}(y|x, z) - E_{q_{co}(y|x)}[\log C_1(x, z)]
\]
\[
= E_{p(y|x)} E_{q_{co}(y|x)} D_{KL}(p(y|x|\epsilon) || q_{co}(y|x, z)) + \text{const} \quad (22)
\]
where $\text{const} = H(Y|X = x, \epsilon) - E_{q_{co}(y|x)}[\log C_1(X = x, z)]$, and the last equation is according to (20). With Lemma 1 applied, the first term in (22) can be decomposed as
\[
E_{p(y|x)} E_{q_{co}(y|x)} D_{KL}(p(y|x|\epsilon) || q_{co}(y|x, z))
\]
\[
= D_{KL}(p(y|x) || \tilde{q}_{co}(y|x))
\]
\[
+ E_{q_{co}(y|x)} D_{KL}(q_{co}(y|x) || q_{co}(y|x, z)) + I(Y; \epsilon)
\]
where we have $p(y|x) = E_{p(y|x)} p(y|x, \epsilon)$, and also
\[
\tilde{q}_{co}(y|x) \propto \exp[E_{q_{co}(y|x)} \log q_{co}(y|x, z)]
\]
\[
\propto \exp[E_{q_{co}(y|x)} q_{co}(y|x) \log q(y|z)]
\]
\[
= \exp[q_c(y|x) E_{q_{co}(y|x)} \log q(y|z)]
\]
\[
\propto \exp[q_c(y|x)] \tilde{q}(y|x).
\]
Since $\tilde{q}_{co}(y|x) \propto \exp[q_c(y|x)] \tilde{q}(y|x)$, we have $q_{co}(y|x)$ follows
\[
q_{co}(y|x) = \tilde{q}(y|x)/C_2(x) \text{ with } C_2(x) := \int_Y \exp[q_c(y|x)] \tilde{q}(y|x) dy.
\]
The proof follows if we plug in the above decomposition when computing $E_{p(y|x)}[L_q'(x, Y)]$.

Note that both $H(Y|X, \epsilon)$ and $I(Y; \epsilon)$ are constant with respect to the model $q(y|x)$.

1) For the bias term, we have
\[
D_{KL}(p(y|x) || \tilde{q}_{co}(y|x))
\]
\[
= \int_Y p(y|x) \log \frac{p(y|x)}{\tilde{q}_{co}(y|x)} dy
\]
\[
= \int_Y p(y|x) \log \frac{C_2(x) \cdot p(y|x)}{\exp[q_c(y|x)] \tilde{q}(y|x)} dy
\]
\[
= \int_Y p(y|x) \log \left[\frac{\alpha(y|x) \cdot p(y|x)}{\tilde{q}(y|x)}\right] dy
\]
where $\alpha(y|x) := E_{q_{co}(y|x)}[\exp[q_c(y|x)]] / \exp[q_c(y|x)]$. For $\alpha(y|x) \leq 1$, that is, $q_c(y|x) \geq \log[E_{q_{co}(y|x)}[\exp[q_c(y|x)]]]$, we have the co-teaching bias term satisfying
\[
D_{KL}(p(y|x) || \tilde{q}_{co}(y|x)) \leq D_{KL}(p(y|x) || \tilde{q}(y|x)).
\]

2) For the variance term, we have
\[
E_{q_{co}(y|x)} D_{KL}(\tilde{q}_{co}(y|x) || q_{co}(y|x, z))
\]
\[
= E_{q_{co}(y|x)} \int_Y \tilde{q}_{co}(y|x) \log \frac{\tilde{q}_{co}(y|x)}{q_{co}(y|x, z)} dy
\]
\[
= E_{q_{co}(y|x)} \int_Y \tilde{q}(y|x) \log \frac{\tilde{q}(y|x)}{\tilde{q}(y|x)} dy
\]
\[
\times \log \frac{\tilde{q}(y|x) / \alpha(y|x)}{\exp[q_c(y|x) \log q(y|z)] / C_1(x, z)} dy
\]
\[
\geq E_{q_{co}(y|x)} \int_Y \tilde{q}(y|x) \log \frac{C_1(x, z) \cdot \tilde{q}(y|x)}{\alpha(y|x) \cdot \tilde{q}(y|x)} dy
\]
where the last inequality is due to $\exp[q_c(y|x) \log q(y|z)] \leq q(y|z)$ as the output.
of the teacher network is derived by the softmax layer with \( 0 \leq q_i(x|y) \leq 1 \). Similarly, \( C_1(x, z) := \int \frac{\log q(y|x)}{\log q(y|z)} dy \leq \int \frac{\log q(y|z)}{\log q(y|x)} dy = 1 \). In these regards, if we further have \( a(x|y) \leq C_1(x, z) \), then
\[
\mathbb{E}_{q(z|x)} \int \frac{\tilde{q}(y|x)}{a(x|y)} \log \left[ \frac{C_1(x, z) \tilde{q}(y|x)}{a(x|y) q(y|z)} \right] dy \\
\geq \mathbb{E}_{q(z|x)} \int \frac{\tilde{q}(y|x)}{q(y|z)} \log \left( \frac{\tilde{q}(y|x)}{q(y|z)} \right) dy \\
= \mathbb{E}_{q(z|x)} \mathbb{D}_{KL}[\tilde{q}(y|x) || q(y|z)].
\]
To conclude, if \( a(x|y) \leq C_1(x, z) \), then
\[
\mathbb{E}_{q(z|x)} \mathbb{D}_{KL}[q_{\alpha}(y|x) || q_{\alpha}(y|x, z)] \\
\geq \mathbb{E}_{q(z|x)} \mathbb{D}_{KL}[\tilde{q}(y|x) || q(y|z)].
\]
This completes the proof. \( \square \)

ACKNOWLEDGMENT

The authors appreciate Qinghua Tao for helpful discussions, and also the anonymous reviewers for their insightful comments.

REFERENCES

[1] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, “Understanding deep learning requires rethinking generalization,” 2016, arXiv:1611.03530.
[2] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” J. Mach. Learn. Res., vol. 15, no. 56, pp. 1929–1958, 2014.
[3] O. Rippel, M. Gelbart, and R. Adams, “Learning ordered representations with nested dropout,” in Proc. Int. Conf. Mach. Learn., vol. 32, 2014, pp. 1746–1754.
[4] B. Han et al., “Co-teaching: Robust training of deep neural networks with extremely noisy labels,” in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 8536–8546.
[5] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, “Learning from massive noisy labeled data for image classification,” in Proc. IEEE Comput. Conf. Mach. Learn. Pattern Recognit. (CVPR), Jun. 2015, pp. 2691–2699.
[6] H. Song, M. Kim, and J. Lee, “SELFIE: Refurbishing unclean samples for robust deep learning,” in Proc. Int. Conf. Mach. Learn., vol. 97, 2019, pp. 5907–5915.
[7] W. Li, L. Wang, W. Li, E. Agustsson, and L. Van Gool, “WebVision database: Visual learning and understanding from web data,” 2017, arXiv:1708.02862.
[8] D. Mahajan et al., “Exploring the limits of weakly supervised pretraining,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 181–196.
[9] C. Olston and M. Najork, Web Crawling. Boston, MA, USA: Now, 2010.
[10] D. Angluin and P. Laird, “Learning from noisy examples,” Mach. Learn., vol. 2, no. 4, pp. 343–370, 1988.
[11] J. R. Quinlan, “Induction of decision trees,” in Proc. Int. Conf. Mach. Learn., vol. 1, no. 1, pp. 81–106, 1986.
[12] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei, “MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels,” in Proc. Int. Conf. Mach. Learn., vol. 30, 2018, pp. 2304–2313.
[13] E. Malach and S. Shalev-Shwartz, “Decoupling ‘when to update’ from ‘how to update,’” in Proc. Adv. Neural Inf. Process. Syst., vol. 30, 2017, pp. 960–970.
[14] X. Yu, B. Han, J. Yao, G. Niu, I. Tsang, and M. Sugiyama, “How does disagreement help generalization against label corruption?” in Proc. Int. Conf. Mach. Learn., 2019, pp. 7164–7173.
[15] H. Wei, L. Feng, X. Chen, and B. An, “Combating noisy labels by agreement: A joint training method with co-regularization,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 13726–13735.
