MentorNet: Regularizing Very Deep Neural Networks on Corrupted Labels

Lu Jiang[1] Zhengyuan Zhou[2] Thomas Leung[1] Li-Jia Li[1] Li Fei-Fei[1,2]
[1]Google Inc. [2]Stanford University
{lujiang, leungt, lijiali, feifeili}@google.com, zyzhou@stanford.edu,

Abstract

Recent studies have discovered that deep networks are capable of memorizing the entire data even when the labels are completely random. Since deep models are trained on big data where labels are often noisy, the ability to overfit noise can lead to poor performance. To overcome the overfitting on corrupted training data, we propose a novel technique to regularize deep networks in the data dimension. This is achieved by learning a neural network called MentorNet to supervise the training of the base network, namely, StudentNet. Our work is inspired by curriculum learning and advances the theory by learning a curriculum from data by neural networks. We demonstrate the efficacy of MentorNet on several benchmarks. Comprehensive experiments show that it is able to significantly improve the generalization performance of the state-of-the-art deep networks on corrupted training data.

1. Introduction

Deep neural networks have witnessed tremendous success in various vision tasks such as object recognition [19, 15, 39] and detection [14]. State-of-the-art deep networks have hundreds of layers and far more trainable model parameters than the number of samples on which they are trained. A recent study found deep networks are capable of memorizing the entire data even on corrupted labels [45], where some or all true labels are replaced with random labels. Regularization is an effective approach to overcome overfitting. Zhang *et al*. [45] empirically demonstrated that common regularizers for neural networks such as weight decay, data augmentation [20] and dropout [36], namely model regularizers, are less effective for improving the generation performance of deep convolutional neural networks (CNNs) trained on corrupted labels, an observation also confirmed by our study. Deep CNNs are typically trained on large-scale data, the annotation on which are usually noisy [1, 11]. The ability to overfit noise in training data often leads to poor performance.

It is challenging to regularize very deep CNNs due to the formidable number of model parameters. To tackle this challenge, we propose a novel technique to regularize deep CNNs in the data dimension, which we term *data regularization*. Our goal is, by regularizing CNNs trained on corrupted labels, to improve its generalization performance on clean test data. The networks considered are deep CNNs such as Resnet [15] and Inception-resnet [39] which have a few hundred layers and orders-of-magnitude more parameters than the number of training examples. Specifically, we propose to *learn* time-varying weights for each example used to train the classification network, namely *StudentNet*. We introduce a *MentorNet* to supervise the training of the StudentNet. As shown in Fig. 1, during training, MentorNet learns to assign a weight to each training example. By learning nonuniform weights, MentorNet encourages some examples to be learned earlier with more attention, thereby prioritizing the learning effort. For MentorNet training, we first pretrain a MentorNet to approximate some predefined weighting specified in labeled data. Then we finetune it on a third dataset of clean labels. In test time, *StudentNet* makes predictions alone without MentorNet.

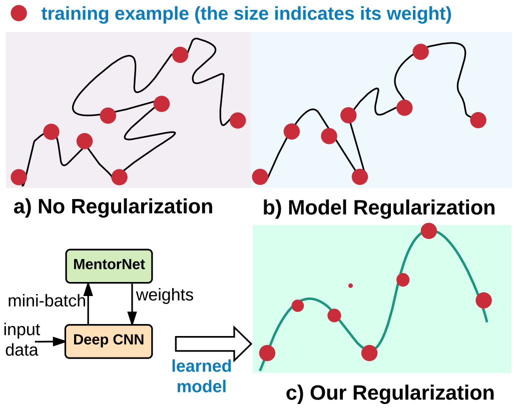


Figure 1: Illustration of existing and our regularization. Each dot is a training example, its size indicating example weight. The curve represents the learned model. Existing regularization, such as weight decay, is less effective for deep CNNs. Data regularization addresses the overfitting of deep CNNs by learning to assign appropriate weights to examples.

Our approach is inspired by curriculum learning [4]. MentorNet learns to weight training examples, resulting in a curriculum that decides the timing and attention to learn each example. Curriculum learning has been applied to find better minima in a variety of computer vision problems [38, 26, 7, 16, 25, 44], face detection [26], object detection [7], video detection [16]. Our model advances the curriculum learning methodology by learning a curriculum from data by neural networks. The proposed model enables existing weighting schemes, such as self-paced weighting [21], hard negative mining [31] and focal loss [27], to be understood and further analyzed through a common framework, and more importantly to be learned by a neural network. In addition, we discuss an algorithm to optimize MentorNet, with deep CNNs, on large-scale data. We theoretically prove its convergence and empirically evaluate its performance on the large-scale ImageNet data.

We verify MentorNet on 4 benchmarks: CIFAR-10, CIFAR-100, ImageNet, and YFCC100M. Comprehensive experiments show that the MentorNet is able to improve the performance of deep CNNs trained on both controlled and real-world noisy labels, and outperforms the state-of-the-art weakly-supervised learning methods. To summarize, the contribution of this paper is threefold:

- We discover that deep CNNs trained on corrupted labels can be improved by learning another network to weigh training examples.
- We propose an algorithm to optimize MentorNet with deep CNNs on big data and prove its convergence under standard and mild assumptions.
- We empirically verify the proposed model on 4 datasets of both controlled and real-world noisy labels.

2. Related Work

Our work is inspired by curriculum learning [4], where a model is gradually learned by incorporating samples ordered in a meaningful sequence called *curriculum*. Curriculum learning has been broadly applied to solve various vision problems such as tracking [38], face detection [26], object detection [7], video detection [16], *etc*. A common curriculum learning approach is to assign weights to examples such that examples are learned in a predefined order by example weights. For example, Kumar *et al.* jointly learned the weight with the model parameters [21], leading to a weighting scheme that assigns higher weights to examples of smaller training loss. After that, various weighting schemes have been proposed, *e.g.* in [38, 16, 26, 34, 6, 28]. Alternative minimization is commonly used in existing methods to solve the joint optimization of the model and weight parameters. Existing work using alternative minimization has not been able to train with deep CNNs. Compared to the existing work, our paper presents a first study to propose to learn a curriculum (weighting scheme) from data by neural networks.

In parallel to curriculum learning, a variety of weighting schemes, which assign predefined weights to training examples, have been studied in the computer vision community [10, 23, 31, 27, 46]. For example, Malisiewicz *et al.* [31] proposed to mine "hard-negative" examples for objection detection. Higher weights are given to hard examples, leading to a shifted training set distribution to emphasize such hard negatives. Lin *et al.* introduced a focal loss for object detection. As we shall see, the loss is equivalent to weighting examples w.r.t. the smoothed failure distribution. Different from above work, our model learns weighting schemes by neural networks.

Our work is related to weakly-supervised learning methods which model the noise distribution of corrupted labels. Among recent contributions, Reed *et al.* [33] developed a robust loss to model "prediction consistency". Sukhbaatar *et al.* [37] proposed a noise transformation to estimate the noise distribution. The transformation matrix needs to be periodically updated and is non-trivial to learn. To address the issue, Goldberger *et al.* [12] proposed to add an additional softmax layer end-to-end with the base model. Azadi *et al.* [2] tackled this problem by a regularizer called AIR. This method was shown to be effective but it relied on additional clean labels to train the representation. More recently, methods started utilizing additional clean labels. Veit *et al.* [43] proposed to learn to *clean up* the labels of noisy examples. Li *et al.* [24] trained a network using clean data, coupled with a knowledge graph, to distill soft logits to the noisy data. Different from previous work, we focus on weighting examples in corrupted labels to train very deep CNNs from scratch. Experiments show that our model is effective even without clean labeled data.

3. Model

This section discusses our model for training deep CNNs on corrupted labels. Our goal is to overcome overfitting, by introducing a regularizer that learns to weight examples. Formally, consider a classification problem with the training set $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \cdots, (\mathbf{x}_n, \mathbf{y}_n)\}$, where $\mathbf{x}_i$ denotes the i^{th} observed example and $\mathbf{y}_i \in \{0,1\}^m$ is the corresponding noisy label vector over m classes. Let $g_s(\mathbf{x}_i, \mathbf{w})$ denote the discriminative function of a neural network called *StudentNet*, parameterized by the network parameter $\mathbf{w} \in \mathbb{R}^d$. Further, let $\mathbf{L}(\mathbf{y}_i, g(\mathbf{x}_i, \mathbf{w}))$, a m-dimensional column vector, denote the loss over m classes. We introduce the latent variable, $\mathbf{v} \in \mathbb{R}^{n \times m}$, and optimize the following regularized objective:

$$\min_{\mathbf{w}\in\mathbb{R}^d, \mathbf{v}\in[0,1]^{n\times m}} \mathbb{F}(\mathbf{w},\mathbf{v}) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{v}_i^T \mathbf{L}(\mathbf{y}_i, g_s(\mathbf{x}_i, \mathbf{w})) + G(\mathbf{v}; \lambda) + \theta \|\mathbf{w}\|_2 \qquad (1)$$

where $\|\cdot\|_2$ is the l_2 norm for weight decay, and here data augmentation and dropout are subsumed inside the StudentNet g_s. $\mathbf{v}_i \in [0,1]^{m\times 1}$ is a vector to represent the weight for the i-th example [21]. The function G is the *explicit data* *regularizer*[1] and λ is the hyperparameter. G determines the complexity of a weighting scheme, and imposes an (unnormalized) weight distribution over all training examples. A model without data regularization is a special case where all the weights are equal to 1. This paper focuses on the one-hot label and the softmax cross-entropy loss. For notation convenience, we denote the loss for the i-th example as $\mathbf{L}(\mathbf{y}_i, g_s(\mathbf{x}_i, \mathbf{w})) = \ell_i$ and the weight $\mathbf{v}_i$ as a scalar v_i.

The objective in Eq. (1) is non-trivial to optimize as the size of the latent weight variables is proportional to the total number of training examples. In the existing literature, alternating minimization [13], or its related variants, is commonly employed to minimize the training objective, e.g. in [28, 46, 9, 22, 21, 38]. This is an algorithmic paradigm where $\mathbf{w}$ and $\mathbf{v}$ are alternatively minimized, one at a time while the other is held fixed. When $\mathbf{v}$ is fixed, the weighted loss is typically minimized by stochastic gradient decent. When $\mathbf{w}$ is fixed, we compute $\mathbf{v}^{(k)} = \arg\min_{\mathbf{v}} \mathbb{F}(\mathbf{v}^{(k-1)}, \mathbf{w}^{(k)})$ using the most recently converged $\mathbf{w}^{(k)}$ at epoch k. However, in large-scale CNN training regimes, it is often infeasible to implement alternating minimization. Section 4 will discuss an algorithm to optimize Eq. (1) for large-scale training.

Similar to model regularizers, the function G can take many forms. For example, Kumar *et al.* employed $G(\mathbf{v}) = -\lambda\|\mathbf{v}\|_1$ [21]. In alternative minimization, it leads to a closed form solution for $\mathbf{v}$ when $\mathbf{w}$ is fixed, *i.e.* $v_i^* = \mathbb{1}(\ell_i \leq \lambda)$, $\forall i \in [1, n]$, where $\mathbb{1}$ is the indicator function which equals 1 when the condition is true and 0 otherwise. The weighting scheme is called self-paced learning and have been applied to many vision problems [46, 26, 38, 22]. As another example, the paper [17] introduced a regularizer $G(\mathbf{v}) = \frac{1}{2}\lambda\sum_{i=1}^{n}(v_i^2 - 2v_i)$. The closed-form solution of $\mathbf{v}$ under a fixed $\mathbf{w}$ is given by:

$$\frac{\partial\mathbb{F}(\mathbf{w},\mathbf{v})}{\partial\mathbf{v}} = 0 \Rightarrow v_i^* = \max(0, 1 - \frac{1}{\lambda}\ell_i), \forall i \in [1, m]. \quad (2)$$

In above examples, a regularizer is described in two forms: an analytic form of $G(\mathbf{v})$ and a closed-form solution for $\mathbf{v}^* = \arg\min_{\mathbf{v}} \mathbb{F}(\mathbf{w}, \mathbf{v})$. Following [9], we call the former *explicit* and the latter *implicit* data regularizer. Both forms lead to the same solution. The implicit regularizer is more convenient to use in practice as the analytic form of G can be difficult to derive. Generally, in this perspective, the method, that assigns weights to example loss or gradient [27, 6, 31], can be regarded as specifying an implicit regularizer, or a *weighting scheme*, to compute $\mathbf{v}$. Fan *et al.* [9] showed this assumption holds for a family of weighting schemes of certain properties.

This paper models these weighting schemes through a common framework. The unified framework not only improve our theoretical understanding about the learning objective of existing weighting schemes, but also offers two practical benefits: it allows them to be i) learned by a common neural network *MentorNet*; ii) optimized by the same algorithm in Section 4. Note that weighting schemes can sometimes hide in the objective function. For example, when examples are weighted by the cumulative density function of an exponential distribution:

$$v_i^* = [1 - \exp\{-\ell_i\}]^\gamma, \quad (3)$$

where γ is a hyperparameter for smoothing the distribution. When $\gamma = 1$, Eq. (3) is the common failure distribution used in reliability engineering [3], where the hazard rate is measured by example loss instead of time. It is easy to verify that Eq. (3), when optimized by our Algorithm 1, minimizes the same classification objective in the focal loss [27], an award-winning method for object detection.

In the rest of the paper, Section 4 first introduces an algorithm for optimizing Eq. (1). Section 5 discusses the *MentorNet* for learning weighting schemes.

4. The Algorithm

The alternative minimization used in related work proves to be intractable for large-scale training, mainly for two important reasons. First, in the subroutine of minimizing $\mathbf{w}$ when fixing $\mathbf{v}$, stochastic gradient descent often takes many steps before converging. This means that it can take a long time before moving past this single sub-step. However, such computation is often wasteful, particularly in the initial part of training, because, when $\mathbf{v}$ is far away from the optimal point, there is not much gain in finding the exact optimal $\mathbf{w}$ corresponding to this $\mathbf{v}$. Second, more importantly, the subroutine of minimizing $\mathbf{v}$ when fixing $\mathbf{w}$ is often impractical, because the fixed vector $\mathbf{v}$ may not even fit into memory. For example, training on 10M samples on 5K classes consumes 2TB just for storing the weight matrix. Training data regularization objective in the presence of big training data requires some thought on the algorithmic level.

Consequently, to minimize Eq. (1), we propose a novel algorithm called *SPADE* (Scholastic gradient PArtial DEscent). The algorithm minimizes $\mathbf{w}$ and $\mathbf{v}$ stochastically over mini-batches. It takes an input of a function g_m to compute example weights for each mini-batch, and outputs the optimized model parameters. As a general approach, Algorithm 1 can also take the input of an explicit regularizer G. Let $\Xi_t = \{(\mathbf{x}_j, \mathbf{y}_j)\}_{j=1}^{k}$ denotes a mini-batch fetched uniformly at random and $\mathbf{v}_{\Xi}^{(t)} = [v_1^{(t)}, ..., v_k^{(t)}]$ represent the weights of examples in Ξ_t. The function g_m computes:

$$\mathbf{v}_{\Xi}^{(t)} = g_m(\Xi_t, \mathbf{w}^{(t-1)}) = \arg\min_{\mathbf{v}_{\Xi}} \mathbb{F}(\mathbf{v}^{(t-1)}, \mathbf{w}^{(t-1)}). \quad (4)$$

In Algorithm 1, for $\mathbf{w}$, a stochastic gradient is computed (via a mini-batch) and applied (Step 9), where α_t is the learning rate. For the latent weight variables $\mathbf{v}$, gradient descent is only applied to a small subset thereof, namely

[1] We name it data regularizer as it follows a general definition in [5]: a technique adding a penalty term to the error function to prevent overfitting.

$\mathbf{v}_\Xi$, parameters corresponding only to the mini-batch (Step 6 and 7). The partial gradient update on weight parameters is performed when the explicit form of the data regularization term is used (Step 6). Otherwise, we directly apply $\arg\min_{\mathbf{v}_\Xi} \mathbb{F}(\mathbf{v}^{(t-1)}, \mathbf{w}^{(t-1)})$ using the function g_m (Step 7). Note in both cases, the weight parameters are computed on-the-fly within a mini-batch and thus do not need to be fixed nor stored. Besides, the algorithm maintains a moving average on the p-th percentile of the loss ℓ_{pt} (Step 5). The moving average will be used in the MentorNet in Section 5. The weight decay parameter θ in Eq. (1) needs to be regularized according to the sum of nonzero weights inside a mini-batch (Step 8).

Algorithm 1. SPADE Alg. for optimizing Eq. (1)

```
Input  : Input dataset $\mathcal{D}$, an explicit regularizer $G$ or a
         function $g_m$.
Output: Model parameters $\mathbf{w}$ of the StudentNet.
1  Initialize $\mathbf{w}^{(0)}, \mathbf{v}^{(0)}, \theta^{(0)}, \ell_{pt}^{(0)}, t = 0$
2  while Not Converged do
3      Fetch a mini-batch $\Xi_t$ uniformly at random;
4      Compute the loss $\ell$ for $\Xi_t$ and its percentile $\ell_{pt}^{(t)}$;
5      $\ell_{pt}^{(t)} = (1-\gamma)\ell_{pt}^{(t-1)} + \gamma\ell_{pt}^{(t)}$;
6      if $G$ is used then
           $\mathbf{v}_\Xi^{(t)} = \mathbf{v}_\Xi^{(t-1)} - \alpha_t \nabla_\mathbf{v}\mathbb{F}(\mathbf{w}^{(t-1)}, \mathbf{v}^{(t-1)})|_{\Xi_t}$ ;
7      else Update $\mathbf{v}_\Xi^{(t)} = g_m(\Xi_t, \mathbf{w}^{(t-1)})$ ;
8      $\theta^{(t)} = \theta^{(0)} \frac{1}{k}\sum_{i=1}^{k} \mathbf{v}_{\Xi_i}^{(t)}$ ;
9      $\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} - \alpha_t \nabla_\mathbf{w}\mathbb{F}(\mathbf{w}^{(t-1)}, \mathbf{v}^{(t)})|_{\Xi_t}$;
10     $t \leftarrow t + 1$
11 end
12 return $\mathbf{w}^{(t)}$
```

Algorithm 1 provides a feasible approach for stochastically optimizing the model parameters with the latent weight variables through mini-batch training. The algorithm can be used to replace the commonly used alternative minimization [13] or majorize-minimization algorithm [42], in order to learn latent variables for deep CNNs. This may have profound impacts on many vision applications [28, 9, 21, 38, 16, 32]. Under mild and standard assumptions, in Theorem 1, we prove that the algorithm stabilizes and converges to a stationary point (in mean square): a strong theoretical guarantee for a general non-convex stochastic optimization problem.

Theorem 1 *Let $\nabla_\mathbf{w}\mathbb{F}(\mathbf{w}, \mathbf{v})$ be an L-Lipschitz function in $\mathbf{w}$ (for all $\mathbf{v}$) and $\nabla_\mathbf{v}\mathbb{F}(\mathbf{w}, \mathbf{v})$ be an L-Lipschitz function in $\mathbf{v}$ (for all $\mathbf{w}$)*[2]. *Let $\mathbf{w}^{(t)}, \mathbf{v}^{(t)}$ be iterates from Algorithm 1. If $\sum_{t=0}^{\infty} \alpha_t = \infty, \sum_{t=0}^{\infty} \alpha_t^2 < \infty$, then* $\lim_{t\to\infty} \mathbf{E}[\|\nabla\mathbb{F}(\mathbf{w}^{(t)}, \mathbf{v}^{(t)})\|_2^2] = 0$.

5. MentorNet

Existing weighting schemes are predefined as analytic expressions of G or closed-form solutions to computing $\mathbf{v}$. This section introduces a new way to learn weighting schemes by a neural network called *MentorNet*. The MentorNet replaces the predefined function g_m in Algorithm 1. Let Θ denote the parameters in the MentorNet g_m. Our goal is to learn an optimal Θ to compute the weight for every example and the given the fixed model parameter, *i.e.* $g_m(\mathbf{z}; \Theta) = \arg\min_\mathbf{v} \mathbb{F}(\mathbf{v}, \mathbf{w})$. Here the input $\mathbf{z}$ indicates the features about the training example.

5.1. MentorNet architecture

We first discuss the architecture of the MentorNet. An ideal MentorNet should be capable of 1) accurately approximating existing weighting schemes; 2) switching weighting schemes according to the learning progress of StudentNet, and 3) adapting the weights for different classes. To achieve these goals, we propose an architecture illustrated in Fig. 2, and empirically verify it in Section 6.1.

The input features to the MentorNet $\mathbf{z}$ include the sample loss, label, the training epoch. The loss of an example is represented as its loss in the previous few epochs, and is encoded by a bidirectional LSTM network to capture the prediction variance [6]. The input to each LSTM unit consists of an absolute loss ℓ and its difference to the moving average $\ell - \ell_{pt}$. The moving average of the loss is obtained from Step 5 in Algorithm 1. The label and epoch are encoded by two separate embedding layers. All feature inputs are fed into two fully-connected layers fc_1, fc_2 to compute the weight, where fc_2 uses the sigmoid activation to ensure the output weights are bounded between 0 and 1. The last layer in Fig. 2 is a probabilistic sampling layer. It samples the weights $\mathbf{v}$, without replacement, according to the normalized weight distribution. Note The sampling is only performed on already trained MentorNets and the sampling ratio is a hyperparameter.

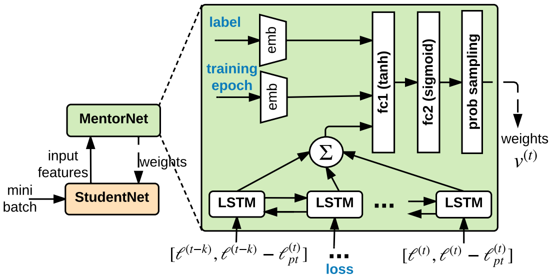


Figure 2: The proposed MentorNet architecture. The input features are the example loss, label and training epoch. The outputs are weights for every example in a mini-batch. *emb*, *fc* and *prob sampling* stand for the embedding, fully-connected and probabilistic sampling layer. $\ell^{(t)}$ and $\ell_{pt}^{(t)}$ denote the example loss and the loss moving average at the epoch t.

[2] This condition would automatically be satisfied if $\nabla_\mathbf{w} L(\cdot)$ is Lipschitz in $\mathbf{w}$ and $G(\cdot)$ is Lipschitz in $\mathbf{v}$.

5.2. MentorNet training

The parameters of MentorNet and StudentNet are not learned jointly to avoid a trivial solution of producing zero weights to all examples. We train MentorNet in two steps: pretrain and finetune on separate datasets other than $\mathcal{D}$. Once the MentorNet is pretrained or finetuned, we fix its parameter and plug it in Algorithm 1 to compute the weight.

MentorNet pretraining: In pretraining, we enumerate the input space of $\mathbf{z}$ and annotate a weight for each data point. This results in a pretraining dataset $\mathcal{D}_{pre} = \{(\mathbf{z}_i, v_i^*)\}_i$, where $\mathbf{z}_i$ the i-th input feature about loss, label and training epoch, and $v_i^* \in [0, 1]$ is a desirable weight. The weight can be derived from any weight scheme with or without closed-form solutions. A MentorNet is learned to approximate the desired weighting scheme specified in the data in $\mathcal{D}_{pre}$. If its explicit regularizer G is known, we can minimize the objective $\mathbb{F}$ in Eq. (1) by

$$\arg\min_{\Theta} \sum_{\mathbf{z}_i \in \mathcal{D}_{pre}} g_m(\mathbf{z}_i; \Theta)\ell_i + G(g_m(\mathbf{z}_i; \Theta); \lambda) \tag{5}$$

Otherwise we directly minimize the euclidean distance to the labeled weights:

$$\arg\min_{\Theta} \sum_{\mathbf{z}_i \in \mathcal{D}_{pre}} \|v_i^* - g_m(\mathbf{z}_i; \Theta)\|_2^2 \tag{6}$$

Theoretically, Eq. (5) and Eq. (6) converges to the same solution. We observed that Eq. (6) converges faster in practice.

In our experiments, we label the data point in $\mathcal{D}_{pre}$ as the following: it assigns 1.0 weights to all examples with a 0.5 sampling ratio in first 20% training epochs; then it favors examples of smaller loss, *i.e.* $v_i = \mathbb{1}(\ell_i \leq \ell_{pt})$ (ℓ_{pt} is set to the 75th-percentile) with a sampling rate of 0.95. In the last 10% training epochs, it switches to the linear weighting using the 80th-percentile and increases the sampling ratio gradually to 1.0.

MentorNet finetuning: The weighting scheme may change along with the learning progress of a StudentNet. To this end, we finetune the MentorNet on a third dataset $\mathcal{D}_{ft} = \{(\mathbf{x}_i, \mathbf{y}_i, v_i^*)\}$, where $(\mathbf{x}_i, \mathbf{y}_i)$ are examples sampled from $\mathcal{D}$. We assume that we have privileged information about v_i, a binary label indicating whether this example should be learned. In our experiment, positive weights are given to examples of correct labels. In finetuning, we also use the activation before the logit layer the as the input to the MentorNet.

We finetune the MentorNet using the sparse mixture-of-expert model [35]. For each $(\mathbf{x}_i, \mathbf{y}_i)$ in $\mathcal{D}_{ft}$ we first compute its input features $\mathbf{z}_i$. Denote $\mathbf{g_k}(\mathbf{z}_i) = [g_1(\mathbf{z}_i), ..., g_k(\mathbf{z}_i)]^{k\times 1}$ the weights obtained by k pretrained MentorNet $g_1, \cdots, g_k$. In finetuning, we minimizes the following cross-entropy loss:

$$\begin{aligned}\arg\min_{\Theta,\mathbf{w_g}} \sum_{v_i \in \mathcal{D}_{ft}} & v_i^* \log G_\sigma(\mathbf{w_g}^T \mathbf{g_k}(\mathbf{z}_i) + \epsilon)) \\ & + (1 - v_i^*) \log(1 - G_\sigma(\mathbf{w_g}^T \mathbf{g_k}(\mathbf{z}_i) + \epsilon)),\end{aligned} \tag{7}$$

where $\mathbf{w_g}$ is the parameter to learn. G_σ is the softmax sparse gating function and ϵ is the Gaussian noise in [35].

A trained MentorNet entails a curriculum. By assigning different weights to examples, it determines the timing and attention to learn each example. MentorNet offers a new flexible approach to implement curriculum learning. The MentorNet can be pretrained to approximate predefined weighting schemes and then finetuned to adjust to suitable weighting schemes.

6. Experiments

This section empirically verifies the proposed method. We seek answers to the following questions: is MentorNet accurate for approximating predefined weighting schemes (Section 6.1)? Is it effective to improve deep CNNs on corrupted labels (Section 6.2)? Can it improve deep CNNs on real-world noisy labels (Section 6.3)?

6.1. MentorNet pretraining

This subsection examines *MentorNet*'s accuracy in approximating predefined weighting schemes. To this end, we generate training data $\mathcal{D}_{pre}$ by enumerating the feasible input space of the loss, label and epoch. For each sample, we label a weight according to existing weighting schemes, including self-paced [21], hard negative mining [31], linear weighting [17], focal loss [27], prediction variance [6]. Besides, we also consider a temporal mixture weighting which employs the self-paced in the first 5 epochs and the hard negative mining thereafter.

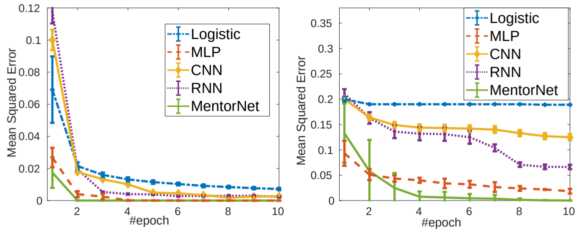
We compare the MentorNet architecture in Fig. 2 to a simple logistic regression and 3 classical networks: a 2-layer Multiple Layer Perception (MLP), a 2-layer CNN with mean pooling, and an LSTM network (RNN) to sequentially encode the features of every example in a mini-batch. The same features are used in all networks. Eq. (6) is minimized by the Adam optimizer [18].

The performance is evaluated by the Mean Squared Error (MSE) to the true weight in $\mathcal{D}_{pre}$. Each experiment is repeated 5 times using random starting values, and the average MSE (with the 90% confidence interval) is reported.

Table 1 compares the MSE. As we see, MentorNet outperforms other networks across all weighting schemes. The results validate that MentorNet can accurately approximate predefined weighting schemes. We found that MentorNet also converges faster. Fig. 3 illustrates the error curve during training, where the x-axis is the training epoch and y-axis is the MSE.

Weighting Scheme	Logistic	MLP	CNN	RNN	MentorNet
Self-paced [21]	8.9±0.8E-3	1.1±0.3E-5	4.9±1.0E-2	1.6±1.0E-2	**1.6±0.5E-6**
Hard negative mining [31]	7.1±0.7E-3	1.6±0.6E-5	2.7±0.6E-3	2.2±0.4E-3	**6.6±4.5E-7**
Linear weighting [17]	9.2±0.1E-4	1.2±0.4E-4	1.1±0.2E-4	2.0±0.3E-2	**4.4±1.3E-5**
Prediction variance [6]	6.8±0.1E-3	4.0±0.1E-3	2.8±0.4E-2	6.2±0.2E-3	**1.4±0.7E-3**
Focal loss [27]	1.7±0.0E-3	6.0±3.5E-5	1.2±0.3E-2	1.2±0.3E-2	**1.5±0.8E-5**
Temporal mixture weighting	1.8±0.0E-1	1.9±0.4E-2	1.2±0.6E-1	6.6±0.4E-2	**1.2±1.1E-4**

Table 1: The MSE comparison on learning predefined weighting schemes.



(a) Hard negative mining (b) Temporal mixture weighting

Figure 3: Convergence comparison of MentorNet architectures.

6.2. Experiments on controlled corrupted labels

This subsection examines MentorNet's ability in improving deep CNNs on corrupted labels. We employ the commonly used standard setting in the literature [45, 12, 2] to train CNNs on corrupted labels, where the label of each image is independently changed to a uniform random class with probability p, where p is called noise fraction. The labels of test data remain clean for evaluation. The controlled setting allows to comprehensively evaluate a model under different noise fractions.

Datasets and StudentNet: We employ the same benchmarks in [45, 12]: CIFAR-10, CIFAR-100 and ImageNet. CIFAR-10 and CIFAR-100 [19] consist of 32 × 32 color images arranged in 10 and 100 classes. Both datasets contain 50,000 training and 10,000 test images. ImageNet ILSVRC2012 [8] contain 1,281,167 training and 50,000 validation images, split into 1,000 classes. Each image is resized to 299x299 with 3 color channels.

We employ three state-of-the-art deep CNNs as our StudentNets: inception [40], resnet [15] and inception-resnet [39]. Table 2 shows their #model parameters, and training and test accuracy of the model trained on original training data (noise fraction= 0). As shown, the StudentNet achieves comparable accuracy to the state-of-the-art on each task. We use the same simplified inception in [45] for smaller input image size, with more details in [45].

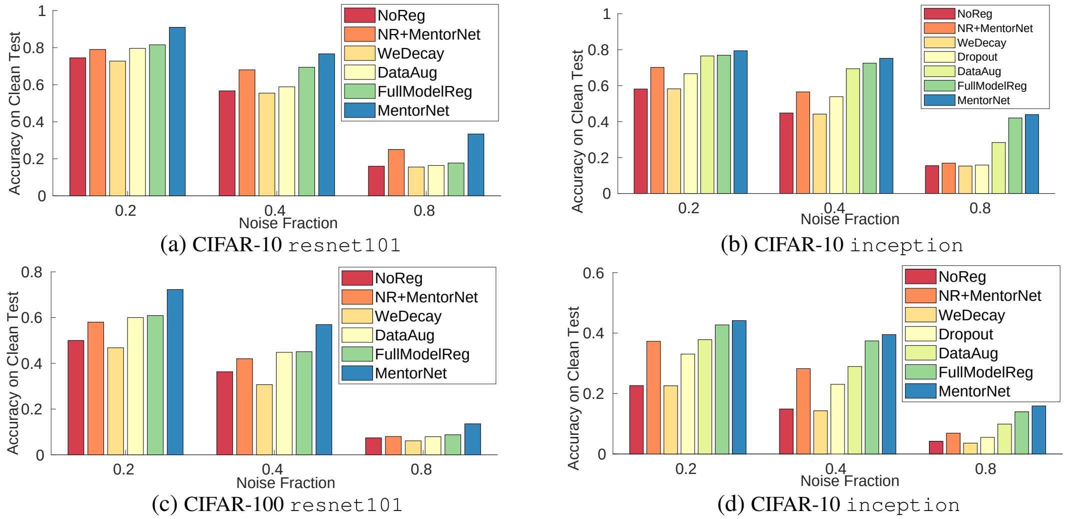
Dataset	Model	#para	train acc	test acc
CIFAR10	inception [40]	1.7M	0.83	0.81
	resnet101 [15]	84M	1.00	0.96
CIFAR100	inception [40]	1.7M	0.64	0.49
	resnet101 [15]	84M	1.00	0.79
ImageNet	inception_resnet v2 [39]	59M	0.88	0.77

Table 2: Information about the *StudentNet* used in the experiments. *train acc* and *test acc* stand for the training and test accuracy of the StudentNet trained on original clean training data.

Baselines: We compare our MentorNet with the following classical regularizers: **NoReg** indicates the vanilla StudentNet with no regularizers. **Weight Decay (WeDecay)** adds an l_2 norm of the model parameters into the learning objective. **Data Augmentation (DataAug)** [20] augments the training data via domain-specific transformations such as random cropping, perturbation and contrast. **Dropout** [36] masks out network layer outputs randomly. **FullModelReg** is the StudentNet trained using all three model regularizers. We also compare to a number of recent weakly-supervised learning methods, including **Self-paced** [21], **Reed** [33], **S-model** [2], and **AIR** [12].

Configurations and Setups: Algorithm 1 is used to optimize the FullModelReg StudentNet, where γ is set to 0.95 and the same learning rate plan in the respective StudentNet is used. Unless stated otherwise, MentorNet is trained on the weighting scheme described in Section 5 with no information about the clean labels. Therefore, it is fair to compare MentorNet with other methods. Specifically, **MentorNet** indicates the MentorNet used to regularize the FullModelReg StudentNet, and **NR+MentorNet** to regularize the NoReg StudentNet. **MentorNet w/ Finetune** indicates the MentorNet finetuned on a third dataset with clean labels using Eq. (7). The dataset is created by randomly sampling 20% data from the training data $\mathcal{D}$. Three pretrained MentorNet under different hyper-parameters are used as the experts in finetuning. For CIFAR datasets, the batch size is set to 128 and runs for 39K iterations. The Step 9 in Algorithm 1 is implemented by the momentum optimizer (momentum=0.9) on a single GPU. For ImageNet We run Algorithm 1 using a distributed asynchronized momentum optimizer on 50 GPUs. We set the batch size to 32 and train the model till they converge *i.e.* 500K steps for NoReg and 1M steps for FullModelReg.

We first show comparison results to classical regularizers. Fig. 4 plots the test accuracy on clean test data, where the x-axis denotes the noise fraction in training data. As we see on both datasets, MentorNet improves FullModelReg (full model regularization) and NoReg (no regularization) across different noise fractions. The improvement is much more significant on deeper CNN Resnet. For example, on CIFAR-10 in Fig. 4a, MentorNet yields an absolute 12% gain on NoReg and a 8% gain on FullModelReg under 40% noise fraction. The sharp accuracy decrease of Resnet along with the growth of noise fraction suggests deep CNNs are more prone to overfitting. FullModelReg is able to overcome some overfitting. However, MentorNet further improves it and achieves the best accuracy on clean test data.



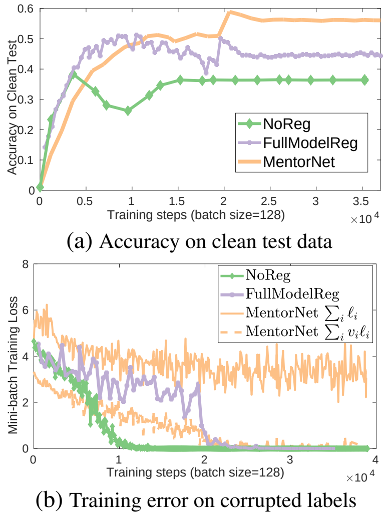
(a) CIFAR-10 `resnet101`

(b) CIFAR-10 `inception`

(c) CIFAR-100 `resnet101`

(d) CIFAR-10 `inception`

Figure 4: Comparison of classical and our regularizer on CIFAR-10 and CIFAR-100. The figure caption shows the dataset and StudentNet. The x-axis and y-axis denote the noise fraction and the classification accuracy on the clean test data.



(a) Accuracy on clean test data

(b) Training error on corrupted labels

Figure 5: Comparison of (a) training error on corrupted training data, and (b) test accuracy on clean test data. The StudentNet is the resnet101 trained on CIFAR-100 under 40% noise fraction.

Fig. 5 plots the training error on corrupted labels and the test accuracy on clean test data, under a representative setting: resnet on CIFAR-100 of 40% noise fraction. The x-axis denotes the training step and the y-axis is the accuracy on clean test data in Fig. 5a, and the total training error of a mini-batch in Fig. 5b. As we see, MentorNet improves the accuracy of FullModelReg by about 11% and exhibits a much larger training error (solid orange curve in Fig. 5b). The results indicate that MentorNet effectively overcomes overfitting and improves generation performance of deep CNNs. Though the training error of MentorNet appears to be fluctuating, it converges. To show that, we plot the error $\sum_i v_i \ell_i$ in the dashed orange curve in Fig. 5b. As we see, the objective converges to zero. This empirically verifies Theorem 1 and demonstrates the convergence of our algorithm.

Fig. 6 illustrates the learned curriculum by MentorNet on CIFAR-10 under 40% noise fraction. It shows the top 15% images of the highest weights in a mini-batch along with the training step (the x-axis). The y-axis denotes the sum of the total weights in the mini-batch. As we see, MentorNet starts assigning weights to a small number of images but many of them have incorrect labels (images with red borders). After that, *e.g.* at iteration 13k, it increases the weights to more images and the images of correct labels (with green borders) also grow. In the end, *e.g.* at iteration 38k, greater weights seem to converge to a subset images of correct labels.

In Table 3, we compare MentorNet to the state-of-the-art weakly-supervised learning method. We cite the number reported in their papers (row 3 and 5) as well as implement two methods with using the same resnet (row 1, 2 and 4). The method marked by the star utilizes additional clean labels in ImageNet, leading to a less fair comparison. Nevertheless, MentorNet significantly outperforms the state-of-the-art methods. Notably, it improves the previous best performance by 11% on CIFAR-100 under 40% noise fraction.

Method	CIFAR10	CIFAR100
Reed Hard (ResNet)	0.623	0.465
Reed Soft (ResNet)	0.613	0.460
S-model (Simple CNN) [12]	-	0.325
S-model (ResNet)	0.699	0.458
AIR (AlexNet) [2]	0.75*	-
MentorNet (Resnet)	**0.766**	**0.569**

Table 3: Comparison to the state-of-the art on CIFAR under 40% noise fraction. $*$ marks the method using external labeled data.

Noise Fraction	0.0	0.2	0.4	0.8	avg
Self-paced [21]	0.69	0.61	0.52	0.13	0.49
Focal loss [27]	0.77	0.59	0.44	0.09	0.47
MentorNet	0.73	0.72	0.57	0.13	0.54
MentorNet w/ Finetune	0.78	0.74	0.59	0.31	0.60

Table 4: Test accuracy of pretrained and finetuned MentorNets on CIFAR-100. The last column is the average cross noise fractions.

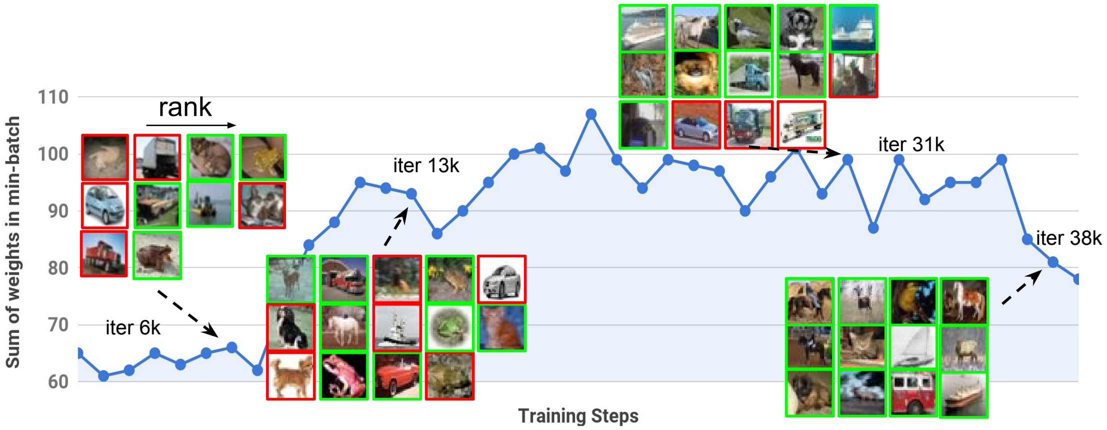


Figure 6: Illustration of the top 15% images of the highest weights on CIFAR-10. Images of clean labels have green borders and images of correlated labels are surrounded by red borders. The x-axis is the training steps and y-axis is the sum of the weights inside a mini-batch. The top 15% images of the highest nonzero weights in a mini-batch are shown.

To verify the MentorNet finetuning, we compare the MentorNet with and without finetuning on CIFAR100 datasets in Table 4. Besides, we also include two existing weighting schemes: self-paced [21] (optimized using Algorithm 1) and focal loss [27]. As shown, finetuning (on additional clean labels) on average yields an absolute 6% improvement across all noise fractions. The results show that finetuning leads to significant improvements over the pretrained MentorNets, and the proposed pretrained MentorNet achieves better performance than existing weighting schemes such as self-paced and focal loss.

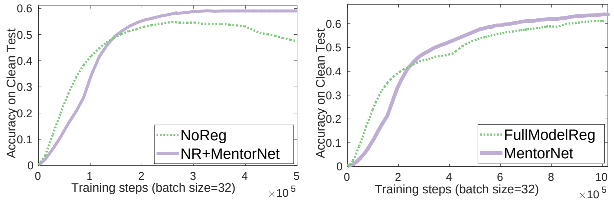
To verify MentorNet on large-scale training, we apply our MentorNet on the ImageNet [8] benchmark to improve the inception-resnet [39] model. Table 5 shows the comparison, where, as shown, MentorNet improves the performance of both the inception-resnet with no regularization (NoReg) and with full regularization (FullModelReg). It boosts the P@1 of NoReg by an absolute 5.2%, and FullModelReg by 3%. Note that the 3-5% absolute improvement is significant on the challenging ImageNet data. Fig. 7 plots the test accuracy along with the training iteration. The results substantiate that MentorNet can improve state-of-the-art deep CNNs on large-scale data.

Method	P@1	P@5
NoReg	0.538	0.770
NoReg+MentorNet	0.590	0.814
WeDecay	0.560	0.809
Dropout	0.575	0.807
DataAug	0.522	0.772
FullModelReg	0.612	0.844
MentorNet	**0.642**	**0.857**

Table 5: Comparison of the accuracy on the standard ImageNet validation set of clean labels. All models are trained on the same ImageNet data under 40% noise fraction.

6.3. Experiments with real-world noise

To test MentorNet on real-world noisy labels, we collect web images from the YFCC100M dataset [41] about the first 51 synsets (concepts) in the ImageNet ILSVRC2012 data. We download the images that contain the synset in the title, tag or description. In total, we train the inception-resenet [39] on 61,238 images about the 51 synsets. The model is evaluated on the standard ImageNet validation set with clean labels about the 51 synsets. The experiments use a similar experimental setup as in Section 6.3, except that we train inception-resenet for 50 epochs asynchronously using 15 GPUs. Note no clean labels are used in training. Table 6 compares the results. As we see, the proposed MentorNet significantly improves baseline methods on real-world noisy labels.



(a) MentorNet + NoReg (b) MentorNet + FullModelReg

Figure 7: Comparison of the accuracy on ImageNet. The x-axis denotes the training step in 100 thousand and the y-axis is the validation accuracy on the standard ImageNet validation set.

Method	P@1	p@5
FullModelReg	0.585	0.818
Reed Soft [33]	0.560	0.826
S-Model [12]	0.561	0.824
Self-paced [21]	0.543	0.798
MentorNet	**0.622**	**0.845**

Table 6: Test accuracy of CNNs trained on noisy web Images.

Fig. 9 plots the confusion matrix for all 50 classes of our best model (MentorNet + Inception-Resnet). The detailed class name and class accuracy (p@1) can be found in Table 7. The best model in the table denotes the best baseline model and its precision@1 for each class.

Fig. 8 illustrates the learned image representation of our best MentorNet model. For each image in the clean ImageNet ILSVRC2012 validation set, we extract the

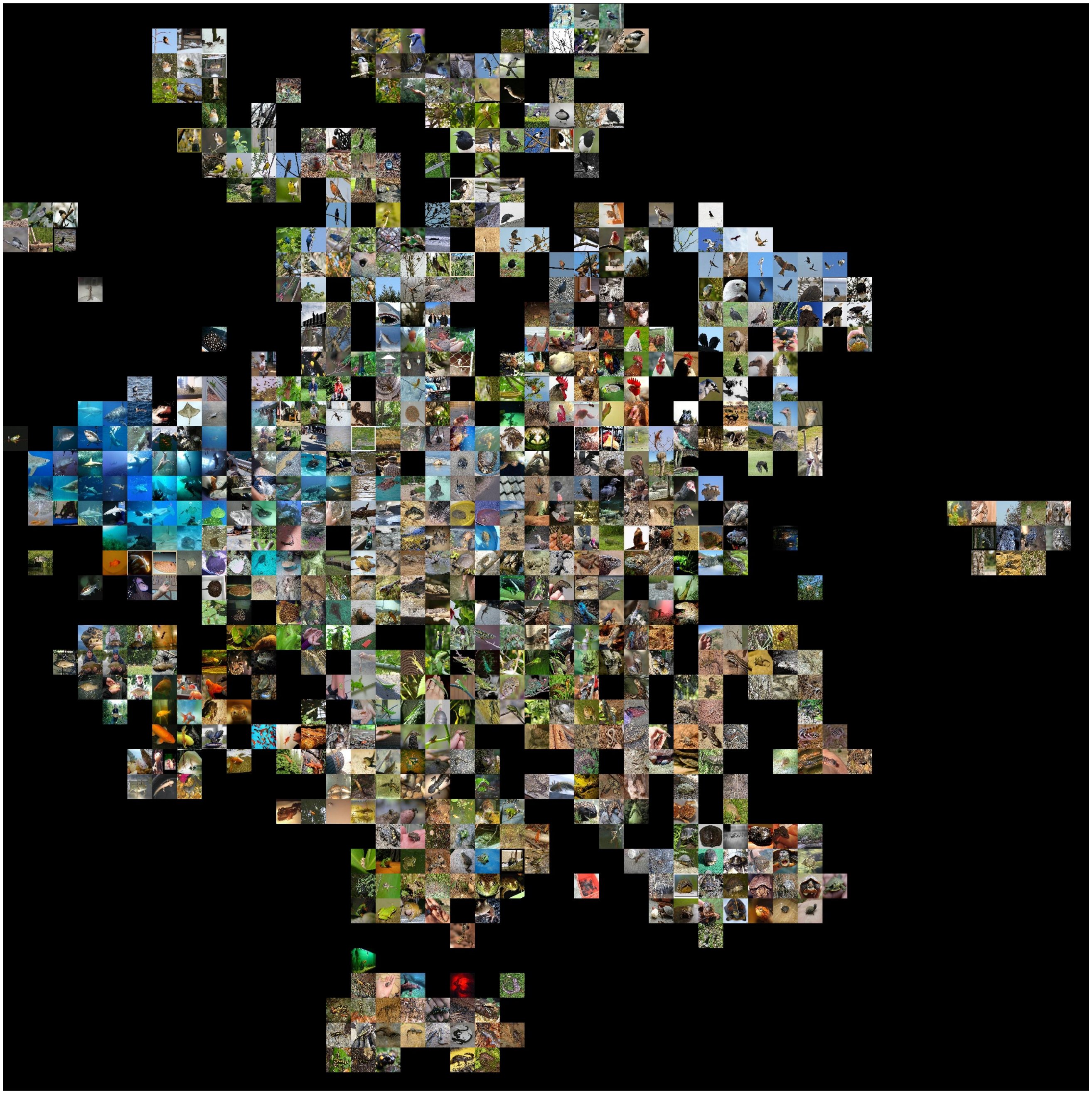


Figure 8: Visualization of the learned image feature of our best model (MentorNet + Resnet-Inception). We extract the "PreLogitsFlatten" to represent an image and plot the 2-dimensional image embedding computed by the t-SNE. Each image is the image from the ImageNet ILSVRC2012 validation set and nearby images are of similar embedding values. Note our model is trained on noisy web image data.

1,536 dimensional "PreLogitsFlatten" activations from the Inception-Resnet-v2 with our MentorNet model trained on noisy web images. We compute a 2-dimensional embedding using the t-SNE [29] respects to the high-dimensional (L2) distances. In other words, t-SNE arranges images that have a similar the PreLogitsFlatten activation nearby in the embedding. Every image in Fig. 8 is surrounded with its nearest neighbors. As shown, the learned image features are reasonable and place semantically relevant images close to each other.

7. Conclusion

In this paper, we presented a novel model for training deep CNNs on corrupted labels. Our work was inspired by curriculum learning and advanced the methodology by proposing to learn weighting schemes via a neural network called MentorNet. We proposed an algorithm for training MentorNet with deep CNNs on large-scale data. We conducted comprehensive experiments on four datasets of controlled and real-world noise. Our empirical results offered insightful evidence that generalization performance of deep CNNs trained on corrupted labels can be effectively improved by learning to weight examples. Directions for future work include MentorNet finetuning where clean labels are either unavailable or of limited size.

References

[1] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016. 1

[2] S. Azadi, J. Feng, S. Jegelka, and T. Darrell. Auxiliary image regularization for deep cnns with noisy labels. *ICLR*, 2016. 2, 6, 7

[3] I. Bazovsky. *Reliability theory and practice*. Courier Corporation, 2004. 3

[4] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *ICML*, 2009. 1, 2

[5] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006. 3

[6] H.-S. Chang, E. Learned-Miller, and A. McCallum. Active bias: Training a more accurate neural network by emphasizing high variance samples. *NIPS*, 2017. 2, 3, 4, 5, 6

[7] X. Chen and A. Gupta. Webly supervised learning of convolutional networks. In *ICCV*, 2015. 2

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 6, 8

[9] Y. Fan, R. He, J. Liang, and B.-G. Hu. Self-paced learning: An implicit regularization perspective. In *AAAI*, 2017. 3, 4

[10] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008. 2

[11] P. Garrigues, S. Farfade, H. Izadinia, K. Boakye, and Y. Kalantidis. Tag prediction at flickr: a view from the darkroom. *arXiv preprint arXiv:1612.01922*, 2016. 1

[12] J. Goldberger and E. Ben-Reuven. Training deep neural-networks using a noise adaptation layer. In *ICLR*, 2017. 2, 6, 7, 8

[13] J. Gorski, F. Pfeuffer, and K. Klamroth. Biconvex sets and optimization with biconvex functions: a survey and extensions. *Mathematical Methods of Operations Research*, 66(3):373–407, 2007. 3, 4

[14] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *ICCV*, 2017. 1

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 6

[16] L. Jiang, D. Meng, S.-I. Yu, Z. Lan, S. Shan, and A. Hauptmann. Self-paced learning with diversity. In *NIPS*, 2014. 2, 4

[17] L. Jiang, D. Meng, Q. Zhao, S. Shan, and A. G. Hauptmann. Self-paced curriculum learning. In *AAAI*, 2015. 3, 5, 6

[18] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[19] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009. 1, 6

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 6

[21] M. P. Kumar, B. Packer, and D. Koller. Self-paced learning for latent variable models. In *NIPS*, 2010. 2, 3, 4, 5, 6, 7, 8

[22] M. P. Kumar, H. Turki, D. Preston, and D. Koller. Learning specific-class segmentation from diverse data. In *ICCV*, 2011. 3

[23] A. Lapedriza, H. Pirsiavash, Z. Bylinskii, and A. Torralba. Are all training examples equally valuable? *arXiv preprint arXiv:1311.6510*, 2013. 2

[24] Y. Li, J. Yang, Y. Song, L. Cao, J. Li, and J. Luo. Learning from noisy labels with distillation. In *ICCV*, 2017. 2

[25] J. Liang, L. Jiang, D. Meng, and A. G. Hauptmann. Learning to detect concepts from webly-labeled video data. In *IJCAI*, 2016. 2

[26] L. Lin, K. Wang, D. Meng, W. Zuo, and L. Zhang. Active self-paced learning for cost-effective and progressive face identification. *TPAMI*, 2017. 2, 3

[27] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *ICCV*, 2017. 2, 3, 5, 6, 7, 8

[28] F. Ma, D. Meng, Q. Xie, Z. Li, and X. Dong. Self-paced co-training. In *ICML*, 2017. 2, 3, 4

[29] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008. 9

[30] J. Mairal. Stochastic majorization-minimization algorithms for large-scale optimization. In *NIPS*, 2013. 15

[31] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011. 2, 3, 5, 6

[32] D. Meng, Q. Zhao, and L. Jiang. What objective does self-paced learning indeed optimize? *arXiv preprint arXiv:1511.06049*, 2015. 4

[33] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014. 2, 6, 8

[34] E. Sangineto, M. Nabi, D. Culibrk, and N. Sebe. Self paced deep learning for weakly supervised object detection. *arXiv preprint arXiv:1605.07651*, 2016. 2

[35] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017. 5

[36] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014. 1, 6

[37] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, and R. Fergus. Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*, 2014. 2

[38] J. S. Supancic and D. Ramanan. Self-paced learning for long-term tracking. In *CVPR*, 2013. 2, 3, 4

[39] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, pages 4278–4284, 2017. 1, 6, 8

[40] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 6

[41] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016. 8

[42] F. Vaida. Parameter convergence for em and mm algorithms. *Statistica Sinica*, pages 831–840, 2005. 4

[43] A. Veit, N. Alldrin, G. Chechik, I. Krasin, A. Gupta, and S. Belongie. Learning from noisy large-scale datasets with minimal supervision. In *CVPR*, 2017. 2

[44] Y. Wu and Y. Tian. Training agent for first-person shooter game with actor-critic curriculum learning. In *ICLR*, 2016. 2

[45] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017. 1, 6

[46] D. Zhang, L. Yang, D. Meng, D. Xu, and J. Han. Spftn: A self-paced fine-tuning network for segmenting objects in weakly labelled videos. In *CVPR*, 2017. 2, 3

id	synset	label	our model	base model
1	n01440764	tench, Tinca tinca	0.82	0.8
2	n01443537	goldfish, Carassius auratus	0.58	0.58
3	n01484850	great white shark, white shark, man-eater, man-eating shark, Carcharodon carcharias	0.68	0.64
4	n01491361	tiger shark, Galeocerdo cuvieri	0.66	0.58
5	n01494475	hammerhead, hammerhead shark	0.34	0.24
6	n01496331	electric ray, crampfish, numbfish, torpedo	0.52	0.5
7	n01498041	stingray	0.3	0.22
8	n01514668	cock	0.68	0.6
9	n01514859	hen	0.24	0.26
10	n01518878	ostrich, Struthio camelus	0.84	0.72
11	n01530575	brambling, Fringilla montifringilla	0.88	0.74
12	n01531178	goldfinch, Carduelis carduelis	0.56	0.58
13	n01532829	house finch, linnet, Carpodacus mexicanus	0.8	0.74
14	n01534433	junco, snowbird	0.88	0.74
15	n01537544	indigo bunting, indigo finch, indigo bird, Passerina cyanea	0.82	0.76
16	n01558993	robin, American robin, Turdus migratorius	0.84	0.86
17	n01560419	bulbul	0.68	0.68
18	n01580077	jay	0.74	0.52
19	n01582220	magpie	0.54	0.5
20	n01592084	chickadee	0.82	0.8
21	n01601694	water ouzel, dipper	0.72	0.76
22	n01608432	kite	0.44	0.38
23	n01614925	bald eagle, American eagle, Haliaeetus leucocephalus	0.84	0.8
24	n01616318	vulture	0.5	0.42
25	n01622779	great grey owl, great gray owl, Strix nebulosa	0.96	0.94
26	n01629819	European fire salamander, Salamandra salamandra	0.94	0.86
27	n01630670	common newt, Triturus vulgaris	0.5	0.42
28	n01631663	eft	0.42	0.46
29	n01632458	spotted salamander, Ambystoma maculatum	0.78	0.82
30	n01632777	axolotl, mud puppy, Ambystoma mexicanum	0.64	0.58
31	n01641577	bullfrog, Rana catesbeiana	0.64	0.58
32	n01644373	tree frog, tree-frog	0.42	0.36
33	n01644900	tailed frog, bell toad, ribbed toad, tailed toad, Ascaphus trui	0.52	0.48
34	n01664065	loggerhead, loggerhead turtle, Caretta caretta	0.56	0.52
35	n01665541	leatherback turtle, leatherback, leathery turtle, Dermochelys coriacea	0.56	0.5
36	n01667114	mud turtle	0.28	0.2
37	n01667778	terrapin	0.26	0.3
38	n01669191	box turtle, box tortoise	0.8	0.72
39	n01675722	banded gecko	0.42	0.5
40	n01677366	common iguana, iguana, Iguana iguana	0.7	0.68
41	n01682714	American chameleon, anole, Anolis carolinensis	0.76	0.72
42	n01685808	whiptail, whiptail lizard	0.76	0.72
43	n01687978	agama	0.4	0.4
44	n01688243	frilled lizard, Chlamydosaurus kingi	0.58	0.46
45	n01689811	alligator lizard	0.34	0.32
46	n01692333	Gila monster, Heloderma suspectum	0.6	0.54
47	n01693334	green lizard, Lacerta viridis	0.26	0.22
48	n01694178	African chameleon, Chamaeleo chamaeleon	0.68	0.58
49	n01695060	Komodo dragon, Komodo lizard, dragon lizard, giant lizard, Varanus komodoensis	0.86	0.82
50	n01697457	African crocodile, Nile crocodile, Crocodylus niloticus	0.66	0.68

Table 7: Precision@1 for the first 50 classes in the ILSVRC2012 validation set. The comparison is between our best model and the second best baseline model. All models are trained on web images of noisy labels. The performance is evaluated on the clean ImageNet ILSVRC2012 validation set.

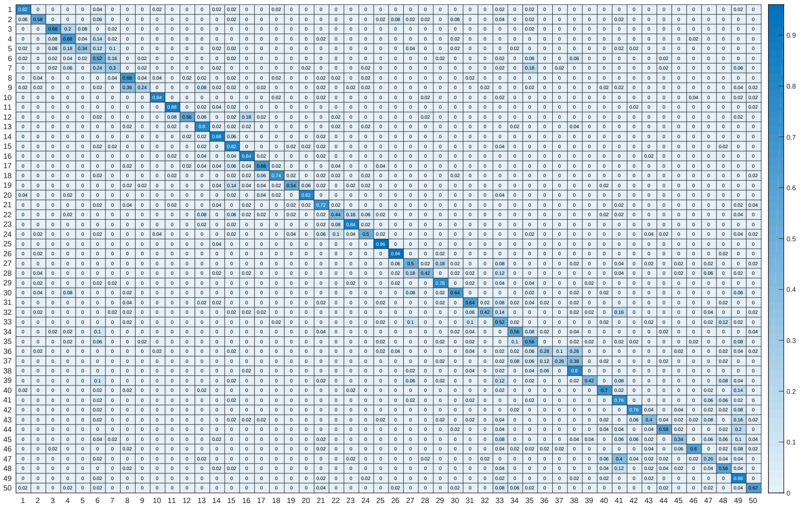


Figure 9: The confusion matrix of the prediction of our best model. The model is trained on web images of noisy labels. The performance is evaluated on the clean ImageNet ILSVRC2012 validation set.

Proof of Theorem 1

Theorem 1: Let $\nabla_{\mathbf{w}}\mathbb{F}(\mathbf{w},\mathbf{v})$ be an L-Lipschitz function in $\mathbf{w}$ (for all $\mathbf{v}$) and $\nabla_{\mathbf{v}}\mathbb{F}(\mathbf{w},\mathbf{v})$ be an L-Lipschitz function in $\mathbf{v}$ (for all $\mathbf{w}$)[3]. Let $\mathbf{w}^{(k)},\mathbf{v}^{(k)}$ be iterates from Algorithm 1. If $\sum_{k=0}^{\infty}\alpha_k = \infty, \sum_{k=0}^{\infty}\alpha_k^2 < \infty$, then $\lim_{k\to\infty}\mathbf{E}[\|\nabla\mathbb{F}(\mathbf{w}^k,\mathbf{v}^k)\|_2^2] = 0$.

Proof 1 *We first prove in full detail the case where an explicit form of G is available. The update is:*

$$\begin{aligned}\mathbf{w}^{k+1} &= \mathbf{w}^k - \alpha_k\nabla_{\mathbf{w}}\mathbb{F}(\mathbf{w}^k,\mathbf{v}^k)|_{\Xi_k},\\ \mathbf{v}^{k+1} &= \mathbf{v}^k - \alpha_k\nabla_{\mathbf{v}}\mathbb{F}(\mathbf{w}^{k+1},\mathbf{v}^k)|_{\Xi_k}.\end{aligned}$$

Define the n-dimensional vector e^k as $e_i^k = 1$ if $(x_i,y_i)\in\Xi_k$ and 0 otherwise and denote by $\otimes$ the point-wise product operation between two vectors: $[a_1,a_2]\otimes[b_1,b_2]=[a_1b_1,a_2b_2]$. Since the mini-batch is draw uniformly at random, we can rewrite the update as:

$$\begin{aligned}\mathbf{w}^{k+1} &= \mathbf{w}^k - \alpha_k[\nabla_{\mathbf{w}}\mathbb{F}(\mathbf{w}^k,\mathbf{v}^k)+\xi^k],\\ \mathbf{v}^{k+1} &= \mathbf{v}^k - \alpha_k[e^k\otimes\nabla_{\mathbf{v}}\mathbb{F}(\mathbf{w}^{k+1},\mathbf{v}^k)],\end{aligned}$$

*where ξ^k's and e^k's are zero-mean, finite-variance **iid** random variables.*

By Lipschitz continuity of $\nabla_{\mathbf{w}}F(\mathbf{w},\mathbf{v})$, we obtain the following:

$$\begin{aligned}&\mathbb{F}(\mathbf{w}^{k+1},\mathbf{v}^k)-\mathbb{F}(\mathbf{w}^k,\mathbf{v}^k)\le\langle\nabla_{\mathbf{w}}\mathbb{F}(\mathbf{w}^k,\mathbf{v}^k),\mathbf{w}^{k+1}-\mathbf{w}^k\rangle+\frac{L}{2}\|\mathbf{w}^{k+1}-\mathbf{w}^k\|_2^2\\ &=\langle\nabla_{\mathbf{w}}\mathbb{F}(\mathbf{w}^k,\mathbf{v}^k),-\alpha_k[\nabla_{\mathbf{w}}\mathbb{F}(\mathbf{w}^k,\mathbf{v}^k)+\xi^k]\rangle+\frac{L}{2}\|\mathbf{w}^{k+1}-\mathbf{w}^k\|_2^2\\ &=-\alpha_k\{\|\nabla_{\mathbf{w}}\mathbb{F}(\mathbf{w}^k,\mathbf{v}^k)\|_2^2+\langle\nabla_{\mathbf{w}}\mathbb{F}(\mathbf{w}^k,\mathbf{v}^k),\xi^k\rangle\}+\frac{L}{2}\|\mathbf{w}^{k+1}-\mathbf{w}^k\|_2^2\\ &=-\alpha_k\{\|\nabla_{\mathbf{w}}\mathbb{F}(\mathbf{w}^k,\mathbf{v}^k)\|_2^2+\langle\nabla_{\mathbf{w}}\mathbb{F}(\mathbf{w}^k,\mathbf{v}^k),\xi^k\rangle\}+\frac{L\alpha_k^2}{2}\|\nabla_{\mathbf{w}}\mathbb{F}(\mathbf{w}^k,\mathbf{v}^k)+\xi^k\|_2^2\\ &=-(\alpha_k-\frac{L\alpha_k^2}{2})\|\nabla_{\mathbf{w}}\mathbb{F}(\mathbf{w}^k,\mathbf{v}^k)\|_2^2+\frac{L\alpha_k^2}{2}\|\xi^k\|_2^2-(\alpha_k-L\alpha_k^2)\langle\nabla_{\mathbf{w}}\mathbb{F}(\mathbf{w}^k,\mathbf{v}^k),\xi^k\rangle.\end{aligned}$$

[3] This condition would automatically be satisfied if $\nabla_{\mathbf{w}}L(\cdot)$ is Lipschitz in $\mathbf{w}$ and $G(\cdot)$ is Lipschitz in $\mathbf{v}$.

Similarly, by Lipschitz continuity of $\nabla_{\mathbf{v}} F(\mathbf{w}, \mathbf{v})$, we have:

$$
\begin{aligned}
&\mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k+1}) - \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k}) \leq \langle \nabla_{\mathbf{v}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k}), \mathbf{v}^{k+1} - \mathbf{v}^{k} \rangle + \frac{L}{2} \| \mathbf{v}^{k+1} - \mathbf{v}^{k} \|_2^2 \\
&= \langle \nabla_{\mathbf{v}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k}), -\alpha_k e^k \otimes \nabla_{\mathbf{v}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k}) \rangle + \frac{L}{2} \| \mathbf{v}^{k+1} - \mathbf{v}^{k} \|_2^2 \\
&= -\alpha_k \langle \nabla_{\mathbf{v}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k}), e^k \otimes \nabla_{\mathbf{v}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k}) \rangle + \frac{L\alpha_k^2}{2} \| e^k \otimes \nabla_{\mathbf{v}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k}) \|_2^2.
\end{aligned}
$$

Combining the above two equations, we then have;

$$
\begin{aligned}
&\mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k+1}) - \mathbb{F}(\mathbf{w}^{k}, \mathbf{v}^{k}) = \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k+1}) - \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k}) + \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k}) - \mathbb{F}(\mathbf{w}^{k}, \mathbf{v}^{k}) \\
&\leq -(\alpha_k - \frac{L\alpha_k^2}{2}) \| \nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^{k}, \mathbf{v}^{k}) \|_2^2 + \frac{L\alpha_k^2}{2} \| \xi^k \|_2^2 - (\alpha_k - L\alpha_k^2) \langle \nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^{k}, \mathbf{v}^{k}), \xi^k \rangle \\
&- \alpha_k \langle \nabla_{\mathbf{v}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k}), e^k \otimes \nabla_{\mathbf{v}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k}) \rangle + \frac{L\alpha_k^2}{2} \| e^k \otimes \nabla_{\mathbf{v}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k}) \|_2^2.
\end{aligned}
$$

Taking expectation of both sides and noting that $\mathbf{E}[\xi^k] = 0$, we have:

$$
\begin{aligned}
&\mathbf{E}[\mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k+1})] - \mathbf{E}[\mathbb{F}(\mathbf{w}^{k}, \mathbf{v}^{k})] \\
&= -(\alpha_k - \frac{L\alpha_k^2}{2}) \mathbf{E}[\| \nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^{k}, \mathbf{v}^{k}) \|_2^2] + \frac{L\alpha_k^2}{2} \mathbf{E}[\| \xi^k \|_2^2] - \alpha_k \mathbf{E}[\langle \nabla_{\mathbf{v}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k}), e^k \otimes \nabla_{\mathbf{v}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k}) \rangle] \\
&+ \frac{L\alpha_k^2}{2} \mathbf{E}[\| e^k \otimes \nabla_{\mathbf{v}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k}) \|_2^2] \\
&= -(\alpha_k - \frac{L\alpha_k^2}{2}) \mathbf{E}[\| \nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^{k}, \mathbf{v}^{k}) \|_2^2] + \frac{L\alpha_k^2}{2} \mathbf{E}[\| \xi^k \|_2^2] - \frac{\alpha_k m}{n} \mathbf{E}[\| \nabla_{\mathbf{v}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k}) \|_2^2] \\
&+ \frac{L\alpha_k^2}{2} \sum_{i=1}^{n} \mathbf{E}[\| e_i^k \frac{\partial_{v_i} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k})}{\partial v_i} \|_2^2] \\
&\leq -(\alpha_k - \frac{L\alpha_k^2}{2}) \mathbf{E}[\| \nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^{k}, \mathbf{v}^{k}) \|_2^2] + \frac{L\alpha_k^2}{2} \mathbf{E}[\| \xi^k \|_2^2] - \frac{\alpha_k m}{n} \mathbf{E}[\| \nabla_{\mathbf{v}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k}) \|_2^2] \\
&+ \frac{L\alpha_k^2}{2} \sum_{i=1}^{n} \mathbf{E}[\| \frac{\partial_{v_i} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k})}{\partial v_i} \|_2^2] \\
&= -\alpha_k \mathbf{E}[\| \nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^{k}, \mathbf{v}^{k}) \|_2^2] - \frac{\alpha_k m}{n} \mathbf{E}[\| \nabla_{\mathbf{v}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k}) \|_2^2] \\
&+ \frac{L\alpha_k^2}{2} \{ \mathbf{E}[\| \nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^{k}, \mathbf{v}^{k}) \|_2^2] + \mathbf{E}[\| \nabla_{\mathbf{v}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k}) \|_2^2] + \mathbf{E}[\| \xi^k \|_2^2] . \}
\end{aligned}
$$

where the second equality follows from $\mathbf{E}[\langle \nabla_{\mathbf{v}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k}), e^k \otimes \nabla_{\mathbf{v}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k}) \rangle] = \frac{m}{n} \mathbf{E}[\| \nabla_{\mathbf{v}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k}) \|_2^2]$, which can be checked by a straightforward combinatorial argument.

Since $\mathbf{E}[\| \nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^{k}, \mathbf{v}^{k}) \|_2^2] < \infty, \mathbf{E}[\| \nabla_{\mathbf{v}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k}) \|_2^2] < \infty, \mathbf{E}[\| \xi^k \|_2^2] < \infty$, let B be an upper bound on their sum. By Telescoping, we have:

$$
\begin{aligned}
&\mathbf{E}[\mathbb{F}(\mathbf{w}^{T+1}, \mathbf{v}^{T+1})] - \mathbb{F}(\mathbf{w}^{0}, \mathbf{v}^{0}) = \sum_{k=0}^{T} \left\{ \mathbf{E}[\mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k+1})] - \mathbf{E}[\mathbb{F}(\mathbf{w}^{k}, \mathbf{v}^{k})] \right\} \\
&\leq -\sum_{k=0}^{T} \alpha_k \mathbf{E}[\| \nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^{k}, \mathbf{v}^{k}) \|_2^2] - \sum_{k=0}^{T} \frac{\alpha_k m}{n} \mathbf{E}[\| \nabla_{\mathbf{v}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k}) \|_2^2] + \frac{LB}{2} \sum_{k=0}^{T} \alpha_k^2.
\end{aligned}
$$

Taking the limit $T \to \infty$ of both sides, we obtain:

$$
\begin{aligned}
&-\sum_{k=0}^{\infty} \alpha_k \mathbf{E}[\|\nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^k, \mathbf{v}^k)\|_2^2] - \sum_{k=0}^{\infty} \frac{\alpha_k m}{n} \mathbf{E}[\|\nabla_{\mathbf{v}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^k)\|_2^2] + \frac{LB}{2} \sum_{k=0}^{\infty} \alpha_k^2 \\
&\geq \lim_{T \to \infty} \mathbf{E}[\mathbb{F}(\mathbf{w}^{T+1}, \mathbf{v}^{T+1})] - \mathbf{E}[\mathbb{F}(\mathbf{w}^0, \mathbf{v}^0)] \geq \min_{\mathbf{w}, \mathbf{v}} \mathbb{F}(\mathbf{w}, \mathbf{v})] - \mathbb{F}(\mathbf{w}^0, \mathbf{v}^0) > -\infty.
\end{aligned}
$$

Since $\sum_{k=0}^{\infty} \alpha_k^2 < \infty$, the above inequality immediately implies that $\sum_{k=0}^{\infty} \alpha_k \mathbf{E}[\|\nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^k, \mathbf{v}^k)\|_2^2] < \infty$ and $\sum_{k=0}^{\infty} \alpha_k \mathbf{E}[\|\nabla_{\mathbf{v}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^k)\|_2^2] < 2$.

By Lemma A.5 in [30], to show $\lim_{k \to \infty} \mathbf{E}[\|\nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^k, \mathbf{v}^k)\|_2^2] = 0$, since α_k is not summable, it suffices to show $\Big|\mathbf{E}[\|\nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k+1})\|_2^2] - \mathbf{E}[\|\nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^k, \mathbf{v}^k)\|_2^2]\Big| \leq C\alpha_k$ for some constant C. To do so, we first recall a useful fact: for any two vectors $\mathbf{a}, \mathbf{b}$ and any finite-dimensional vector norm $\|\cdot\|$,

$$
\Big|(\|a\| + \|b\|)(\|a\| - \|b\|)\Big| \leq \|a + b\| \|a - b\|. \tag{8}
$$

We have:

$$
\begin{aligned}
&\Big|\mathbf{E}[\|\nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k+1})\|_2^2] - \mathbf{E}[\|\nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^k, \mathbf{v}^k)\|_2^2]\Big| \\
&= \Big|\mathbf{E}\Big[\big(\|\nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k+1})\|_2 + \|\nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^k, \mathbf{v}^k)\|_2\big)\big(\|\nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k+1})\|_2 - \|\nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^k, \mathbf{v}^k)\|_2\big)\Big]\Big| \\
&\leq \mathbf{E}\Big[\Big|\|\nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k+1})\|_2 + \|\nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^k, \mathbf{v}^k)\|_2\Big| \cdot \Big|\|\nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k+1})\|_2 - \|\nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^k, \mathbf{v}^k)\|_2\Big|\Big] \\
&\leq \mathbf{E}\Big[\Big\|\nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k+1}) + \|\nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^k, \mathbf{v}^k)\Big\|_2 \cdot \Big\|\nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^{k+1})\|_2 - \|\nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^k, \mathbf{v}^k)\|_2\Big\|\Big] \\
&\leq 2LB\mathbf{E}\Big[\|(\mathbf{w}^{k+1}, \mathbf{v}^{k+1})_{-}(\mathbf{w}^k, \mathbf{v}^k)\|_2\Big] \\
&\leq 2LB\alpha_k \mathbf{E}\left[\left\|\Big(\nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^k, \mathbf{v}^k) + \xi^k, e^k \otimes \nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^k)\Big)\right\|_2\right] \\
&= 2LB\alpha_k \mathbf{E}\left[\sqrt{\left\|\nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^k, \mathbf{v}^k) + \xi^k\right\|_2^2} + \sqrt{\left\|e^k \otimes \nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^k)\right\|_2^2}\right] \\
&\leq 2LB\alpha_k \sqrt{\mathbf{E}\left[\left\|\nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^k, \mathbf{v}^k) + \xi^k\right\|_2^2\right] + \mathbf{E}\left[\left\|e^k \otimes \nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^k)\right\|_2^2\right]} \\
&\leq 2LB\alpha_k \sqrt{2\mathbf{E}\left[\left\|\nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^k, \mathbf{v}^k)\right\|_2^2\right] + 2\mathbf{E}\left[\left\|\xi^k\right\|_2^2\right] + \mathbf{E}\left[\left\|\nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^{k+1}, \mathbf{v}^k)\right\|_2^2\right]} \\
&\leq 2LB\alpha_k \sqrt{5B^2} = 2\sqrt{5}B^2 L\alpha_k,
\end{aligned}
$$

where the first inequality is an application of Jensen's inequality, the second inequality follows from Equation (8) the third inequality follows from Lipschitz continuity and the sixth inequality follows from another application of Jensen's inequality. Consequently, by Lemma and the above chain of inequalities, it follows that $\lim_{k \to \infty} \mathbf{E}[\|\nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^k, \mathbf{v}^k)\|_2^2] = 0$. By a similar argument, it follows that $\lim_{k \to \infty} \mathbf{E}[\|\nabla_{\mathbf{v}} \mathbb{F}(\mathbf{w}^k, \mathbf{v}^k)\|_2^2] = 0$. Consequently, $\lim_{k \to \infty} \mathbf{E}[\|\nabla \mathbb{F}(\mathbf{w}^k, \mathbf{v}^k)\|_2^2] = 0$.

For the implicit form case, note that a direct minimization is equivalent to repeatedly gradient descent. The proof continues to carry through with much more complicated algebra, so we omit the details due to space consideration.