Learning Advisor Networks for Noisy Image Classification

Simone Ricci, Tiberio Uricchio(✉), and Alberto Del Bimbo

University of Florence, Florence, Italy
{simone.ricci,tiberio.uricchio,alberto.bimbo}@unifi.it

Abstract. In this paper, we introduced the novel concept of advisor network to address the problem of noisy labels in image classification. Deep neural networks (DNN) are prone to performance reduction and overfitting problems on training data with noisy annotations. Weighting loss methods aim to mitigate the influence of noisy labels during the training, completely removing their contribution. This discarding process prevents DNNs from learning wrong associations between images and their correct labels but reduces the amount of data used, especially when most of the samples have noisy labels. Differently, our method weighs the feature extracted directly from the classifier without altering the loss value of each data. The advisor helps to focus only on some part of the information present in mislabeled examples, allowing the classifier to leverage that data as well. We trained it with a meta-learning strategy so that it can adapt throughout the training of the main model. We tested our method on CIFAR10 and CIFAR100 with synthetic noise, and on Clothing1M that contains real-world noise, reporting state-of-the-art results.

Keywords: Meta learning · Advisor network · Noisy labels

1 Introduction

Modern image classification systems are based on using deep neural networks models that are trained on a huge number of labeled images [11]. Due to the extreme cost of labeling such an amount of images and difficulty in covering many concepts, researchers recently have looked into methods that generate labels automatically. One significant line of research exploits available labeled images from non-experts (e.g. from social networks, online stores) that can be easily retrieved in large quantities but may have mislabeled [1].

Deep neural networks typically consist of a large number of parameters that are highly shared among feature dimensions and states, enabling flexibility in learning different tasks and classes. This flexibility has the advantage to lead to strong discriminative models unless data annotations are corrupted by noise, leading to performance reduction and overfitting problems [9]. Recent methods tried to address the problem by using curriculum learning [4], directly estimating

S. Sclaroff et al. (Eds.): ICIAP 2022, LNCS 13232, pp. 442–453, 2022.
https://doi.org/10.1007/978-3-031-06430-2_37

the labels noise in the set [8], or measuring the confidence of the network during training [12], also using another co-trained network [7]. The idea was usually to understand mislabeled samples out of distribution and reduce their influence on the learning by dampening their loss or decreasing their impact directly from the training set.

In this paper, we proposed a meta-learning approach to address the problem of noisy labels in image classification based on an advisor network, developed to help the classifier. While a standard image classification model is trained, the advisor network observes the main network activations and adjusts features at training time when noisy label images are identified as input. This allows the classifier model to get information even from mislabeled samples where some noise structure is present. We only retained the main model as the final classifier, while the advisor was discarded. Unlike the teacher-student paradigm, the advisor network was not trained to solve the image classification task, but only to help the learning process of the classifier model by its altering activations.

In summary, our contribution is:

- We propose the use of an advisor network, i.e. the use of an additional network at training time, learned by meta-learning, that can adjust activations and gradient of the main network that is being trained.
- We develop such concept for the task of image classification, allowing the training of an image classification network in presence of artificial label noise.
- We test our approach in presence of artificial label noise and on a popular noisy dataset, obtaining state-of-the-art performance.

2 Related Works

2.1 Noisy Training Labels

Numerous works deal with the problem of noisy labels in training data. It has been shown that the performance of machine learning systems degrades in the presence of label noise [18,21]. A first solution involves a loss correction to mitigate the effect of mislabeled samples on the classifier network. For example GLC [8], Reed [22], M-correction [2], F-correction [6] and S-adaptation [20] estimated the matrix of corruption probabilities used to change the wrong labels to the correct ones. Instead, [17,25,32] modeled the annotations noise distribution linearly combining the output of the network and the noisy label to estimate true labels. Another different approach was assigning a weight to each sample. A lower weight value avoids the contribution of that sample to the training of the network. In this way, it is possible to assign low values to mislabeled examples and high values to correct ones. MentorNet [10] and MentorMix [9] found the latent weights with data-driven curriculum learning and the student-teacher paradigm. Other contributions include data augmentation strategies like Mixup [33], Advaug [5] and DevideMix [13]. Differently from these methods, we modified the network activation using an advisor instead of the loss value.

2.2 Meta Learning

There are methods [3,15,26,27] that needs supplemental clean label to handle the noise. This assumption of clean data is also true for a solution that exploits the Meta-learning paradigm. It consists of the use of machine learning algorithms to assist the training and optimization of other machine learning models. Meta-learning [14,23,24,28] had used to address the noisy labels problem. With small clean validation data, the meta-model learns how to correct the biased training labels. For example, L2R [23] weighed each example giving less importance to the mislabels samples. MLNT [14] simulated regular training with synthetic noisy labels. MW-Net [24] learned an explicit weighting function that can be easily adapted to different types of annotations noise. MLC [28] estimated the label noise transition matrix. Contrary to all aforementioned meta-learning solutions, our method does not act by directly modifying the loss of the neural network. We applied a meta-attention layer inside a neural network. The weights of the attention are learned by the advisor network. In this way, the mislabeled data can be leveraged to improve the overall performance of the main model.

3 Method

3.1 Task

In this paper, we developed a method that can handle images with noisy labels when training networks for image classification. We started from the idea that also a mislabeled example contain information that can contribute to a greater generalization of the network. The model should concentrate only on some convenient parts of these data. Our idea was to exploit the attention mechanism to enhance the useful parts of the information and lower the rest. We made use of an auxiliary advisor network that learns automatically a function that weighs the features extracted from a DNN during its training. This advisor network should be aware of the state of the main model and the meta-learning training solves this constraint. Our method Meta Feature Re-Weighting (MFRW) acts like a meta-attention layer. Different from weighting loss methods that tend to completely remove the influence of mislabeled examples during the training our MFRW can take advantage of them.

We first introduce meta-learning basics and formulation typical of methods that learn robust deep neural networks from noisy labels. Then in Sect. 3.3, we explain our method showing the architecture of the whole process.

3.2 Meta Learning for Noisy Image Classification

In general meta-learning (ML) is referred to the process of improving a learning algorithm over multiple learning episodes, also called commonly learning to learn. Usually, ML is divided into two learning algorithms: an inner (or base) algorithm that solves a task, such as images classification, defined by a training dataset and objective function; an outer (or upper/meta) algorithm that updates the inner

one, such that the main model it learns improves an outer objective function. ML was applied to solve the problem of noisy labels in training data [23,24]. We introduce the symbols useful for understanding ML in this particular setting and the three basic steps into which the entire learning process is divided.

Let $D^{train} = \{x_i^{tra}, y_i^{tra}\}_{i=1}^{N}$ be the noisy annotated training set, where N is the total number of examples, composed of an image x_i and the correspondent one-hot label y_i over c classes. In general if we have a deep neural network (DNN) model $\Phi(\cdot; w)$, where w are its parameters and $\hat{y} = \Phi(x; w)$ is its prediction on the input image x, we can obtain the optimal parameters w^* by minimizing the softmax cross-entropy loss $\ell(\hat{y}, y)$ on the training set D^{train}. ML, applied to the Noisy Image Classification task, requires the presence of an additional verified dataset. This validation set $D^{val} = \{x_j^{val}, y_j^{val}\}_{j=1}^{M}$ is much smaller than the training set, $M \ll N$.

In [24] a meta-model was used to implement the ML process. A multilayer perceptron network with only one hidden layer learns how to weigh each training example. Let $\Psi(\cdot; \theta)$, parameterized by θ, be the meta-model that maps a loss value to a scalar weight. In this case, the optimal parameters w^* are derived using the following weighted loss:

$$w^*(\theta) = \underset{w}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{V}_i^{tra}(\theta) \mathcal{L}_i^{tra}(w) \tag{1}$$

with $\mathcal{V}_i^{tra}(\theta) = \Psi(\mathcal{L}_i^{tra}(w); \theta)$ as the weight predicted by the meta model for the i-th training example. Instead the meta model is trained minimizing the validation loss:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{M} \sum_{j=1}^{M} \mathcal{L}_j^{val}(w^*(\theta)) \tag{2}$$

where $\mathcal{L}_j^{val}(w^*(\theta)) = \ell(\Phi(x_j^{val}; w^*(\theta), y_j^{val}))$ is the loss for the j-th validation example.

Equations (1) and (2) can be minimized alternating optimization via gradient descent. One solution that ensures the efficiency of the algorithm and its convergence [24] adopt an online strategy to update θ and w through a single optimization loop, which is divided into three main steps.

The first step is called Virtual-Train because the original DNN will not be updated and the optimization is carried out on a virtual model that is the copy of the original one. Consider the t-th iteration and associated mini batches $\mathcal{B}^{train} = \{(x_i^{tra}, y_i^{tra})\}_{i=1}^{n}$ and $\mathcal{B}^{val} = \{(x_j^{val}, y_j^{val})\}_{j=1}^{m}$, where n and m are the size of mini-batch respectively. The virtual update can be derived by:

$$\hat{w}(\theta) = w - \alpha \frac{1}{n} \sum_{i=1}^{n} \mathcal{V}_i^{tra}(\theta) \nabla_w \mathcal{L}_i^{tra}(w) \tag{3}$$

where α is the learning rate for the DNN and w is its parameter at the current iteration. Then there is the Meta-Train step, where given the optimized virtual model the meta model is updated by:

$$\theta' = \theta - \beta \frac{1}{m} \sum_{j=1}^{m} \nabla_\theta \mathcal{L}_i^{val}(\hat{w}(\theta)) \tag{4}$$

with β the learning rate for the meta model. In last step, Actual-Train, the base DNN model is optimized taking into account the previously updated meta model.

$$w' = w - \alpha \frac{1}{n} \sum_{i=1}^{n} \mathcal{V}_i^{tra}(\theta') \nabla_w \mathcal{L}_i^{tra}(w) \tag{5}$$

w' becomes the w in Eq. (3) for the $(t+1)$-th iteration.

3.3 Meta Feature Re-Weighting (MFRW)

Attention for a DNN is a mechanism that tries to mimics the cognitive attention of the human brain. It intensifies the important parts of the input and reduces the rest. In Meta Feature Re-Weighting (MFRW) the attention is applied with a Hadamard product between the feature extracted from a DNN and a vector of weights automatically learned from a meta-model. In order to get this, we separated the main model $\Phi(\cdot; w)$ in two-part: the backbone $\Phi_b(\cdot; w_b)$, that has an image x as input and gives out a feature vector f, and the classifier $\Phi_c(\cdot; w_c)$, that has f as input and a probability score vector s as output. In this way, it was possible to manipulate the feature f directly with our meta-model Ψ.

The meta-model takes two different inputs $\Psi(f, \mathcal{L})$ and gives back a vector of weights W_f. The first input f is the feature extracted from the backbone Φ_b relative to the example x. This is important for the meta-model because it makes the W_f strictly connected to the feature that needs to be modified. The other input is the loss $\mathcal{L}$ of the example x calculated from the prediction obtained by the main full model Φ. This gives the meta-model information about how much x is a "hard" or an "easy" example for the main model. The two inputs together let the meta-model differentiate a feature belonging to a noisy x from the one related to a correct x. In dot-product attention the multiplication is done element-wise, so the W_f has to be of the same size of f, and its values must be in the range $\in (0, 1)$.

MFRW is divided into 4 main phases for each iteration. Figure 1 shows the overall process of our method divided by step. We put our method at the t-th iteration and we will describe each step to reach the $(t+1)$-th.

Our method needs an additional initial phase Loss Pre-Calculation respect to [24] and what is described in Sect. 3.2. We must calculate in advance the value of loss $\mathcal{L}^{pre}$ related to the training batch x^{train}. This is done at the beginning to obtain a loss value dependent on the original feature and not on the weighted one. It is not an expensive step because it is a direct loss inference, without gradient calculation.

The second step is the Virtual-Train. Here Φ_b^t and Φ_c^t are temporary clone of the original ones. The train batch x^{train} is passed in Φ_b^t to obtain the features f^{train}. Then f^{train} goes inside Ψ^t with the relative loss values $\mathcal{L}^{pre}$ to get the vector of weights W_f. We multiplied element-wise f^{train} with W_f to get a new

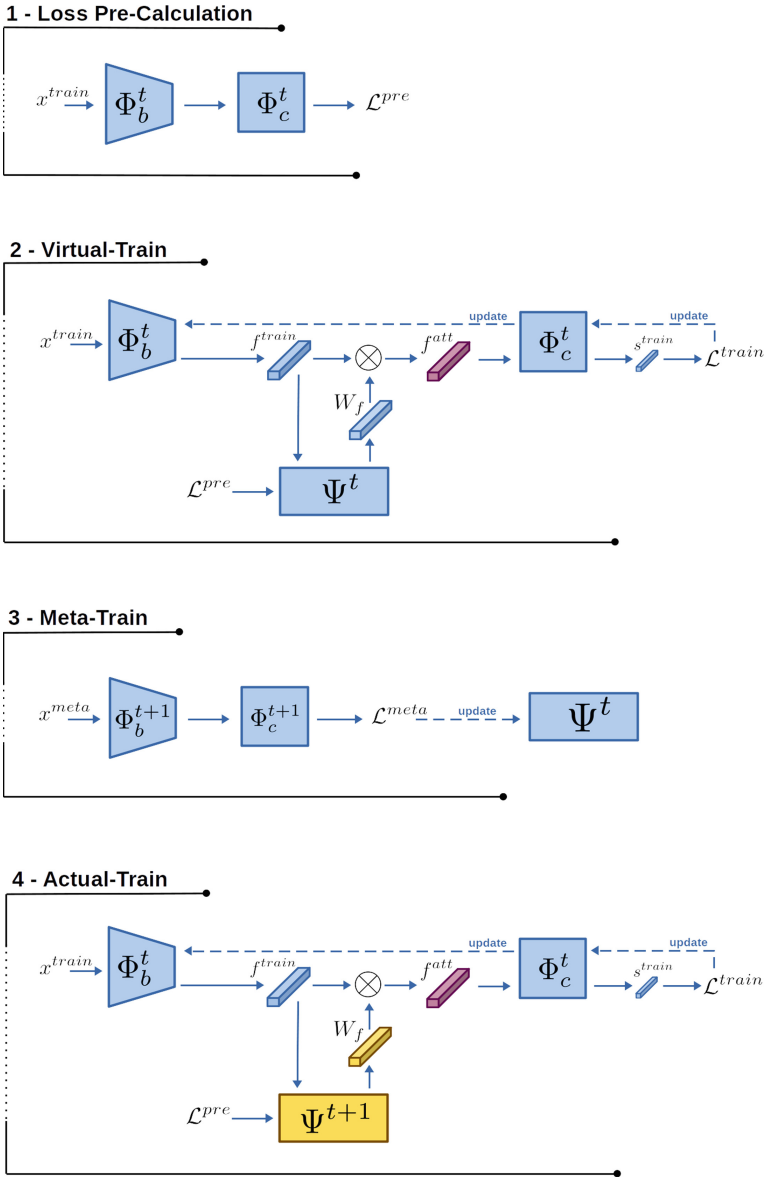

Fig. 1. Illustration of an iteration of the proposed Meta Feature Re-Weighting (MFRW) method. Each iteration is divided into four steps. First, a Loss Pre-Calculation is performed to calculate in advance the loss $\mathcal{L}^{pre}$ value of the training batch x^{train}. The second step is the Virtual-Train, where a clone of the main model is virtually updated. Here the meta-model modifies the feature of the main model multiplying it with a vector of weights. The purple color indicates the weighted features. The third step shows the Meta-Train process. With a meta batch of clean example x^{meta} the meta-model is updated minimizing the loss $\mathcal{L}^{meta}$ given by the previous virtually updated network. In the last phase Actual-Train, the real main model is trained with the meta-model optimized (yellow color). (Color figure online)

feature vector with attention f^{att}. This is given to Φ_c^t to obtain the score s^{train} and then the correspondent loss $\mathcal{L}^{train}$. We now virtually update Φ_b^t and Φ_c^t parameters, but not the one of Ψ^t.

Like [24] we have a clean and balanced meta dataset that will be used to train the meta-model Ψ in the third steps Meta-Train. Here we have Φ_b^{t+1} and Φ_c^{t+1} virtually updated from the step before. Now we pass a meta batch x^{meta} through them in order to get a validation loss $\mathcal{L}^{meta}$. Then Ψ^t is updated minimizing $\mathcal{L}^{meta}$. In this way, the meta-model is optimized to help the main model minimize its error on clean data. Here the optimization takes into consideration also the previous Virtual-Train, thus this is the most expensive part of the method.

The last phase is the Actual-Train where the original Φ_b^t and Φ_c^t are optimized taking into account the updated meta-model Ψ^{t+1}.

The meta-model is used only during the training time of the main network. It will be discarded at test time when only the main network is retained as the final model.

3.4 Meta Model Architecture

Our meta-model Ψ has a really simple architecture. The inputs of the network are a feature f and a loss value $\mathcal{L}_x$. Each input is projected in a fixed size embedding space through a separate fully connected layer. Then the embeddings are concatenated and passed to another fully connected layer that projects them into a larger common space. Its size is the sum of the dimension of each previous embedding. Finally, a linear layer is used to pass the data from the common space to a vector with a size equal to the one of the feature f, that is given as input. Because the output must be an attention weight in the range $\in (0, 1)$ we put a sigmoid activation after the last layer.

4 Experiments

To demonstrate the effectiveness of our method, we conducted experiments on synthetically generated datasets with controlled noise structure and level. Then we tested its ability to generalize with experiments on a real-world dataset.

4.1 Datasets

Following previous work [10,23,24], we used CIFAR-10 and CIFAR-100 which are the typical choice to generate synthetic datasets containing different types of noise structure. They are composed of 50K training images and 10K test images of size 32×32. Of the training set, 1000 images with clean labels are randomly selected to create the validation set for meta-training.

In addition to synthetic datasets, there is a collection of data containing real-world noise. Clothing1M [30] is a dataset that is composed of 1 million images of clothing taken from online shopping websites. There are 14 categories like T-shirt, Shirt, Knitwear, etc. The labels are obtained from the text of the images

provided by the sellers and not from expert annotators, that's why there are errors. The validation set of $7k$ clean data is as the meta dataset. This dataset allowed our strategy to be evaluated as a concrete application for fine-grained classification with noisy training annotations.

4.2 Implementation Details

We used the same settings for the experiments on CIFAR-10 and CIFAR-100. The backbone was a Resnet-32 trained through SGD with a momentum of 0.9, weight decay of 5e−4, batch size of 128, and a starting learning rate of 0.1. Learning rate decreased to its $\frac{1}{10}$ at the 50 epoch and 70 epoch, stopped at the 100 epochs.

With Clothing1M we used as backbone a ResNet-50 pre-trained on ImageNet. It was trained through SGD with a momentum of 0.9, weight decay of 1e−3, and a starting learning rate of 0.01. The batch size was 32 and it was preprocessed resizing the image to 256 × 256, crop the center 224 × 224, and performing normalization. The learning rate was divided by $\frac{1}{10}$ after 5 epochs and stops at 10 epochs.

In every experiment, the meta-model was optimized with Adam and a learning rate of 1e−4. The embedding space size was set always to 100.

4.3 Results

Flip (or asymmetric) is a noise that is designed to mimic the structure where labels are only replaced by similar classes, e.g. dog ↔ cat. We choose to test our method on that type of noise because it usually appends that the label error could depend on the ambiguity between classes and similar visual patterns [30]. We created a synthetic version of CIFAR-10 and CIFAR-100. The noise ratio was controlled by a parameter p, which represents the probability that a clean example is contaminated by noise. In this way we could test our method on different level of noise, from $p = 0.0$ (no noise), to $p = 0.8$ (heavy noise).

Table 1 shows the accuracy results on the test set of CIFAR-10 and CIFAR-100 datasets with flip label noises. The compared methods are directly cited with the result on their paper. For MW-Net [24] and the direct training (CrossEntropy) we report also our reproduced results. The accuracy gained over the other methods were significant. We can see that at a higher noise rate our result outperforms MW-Net and CrossEntropy by a large margin, indicating the effectiveness of our method on the synthetic Flip noise. From the results of Table 1 is possible to notice a limitation of our strategy that occurs when there is no noise ($p = 0.0$) in the training annotations. We obtained worse accuracy values than the training with the classic softmax cross-entropy loss on both CIFAR-10 and CIFAR-100. The advisor network introduces a bias from the distribution of the meta set to the training data. Because the training annotations are completely correct the introduction of this meta bias makes the accuracy a little worse than without.

Table 1. Top-1 accuracy on CIFAR10 and CIFAR100 dataset with Flip noise. The backbone used was a ResNet-32. p denotes the different levels of noise. The results for the cited method are reported directly from their original papers. † indicates the results obtained by our implementation. The first and the second best results are respectively marked with bold and underline.

Dataset	Flip CIFAR-10					Flip CIFAR-100				
Noise p	0.0	0.2	0.4	0.6	0.8	0.0	0.2	0.4	0.6	0.8
CrossEntropy [24]	92.89	76.83	70.77	–	–	70.50	50.86	43.01	–	–
Reed-Hard [22]	92.31	88.28	81.06	–	–	69.02	60.27	50.40	–	–
S-Model [6]	83.61	79.25	75.73	–	–	51.46	45.45	43.8	–	–
Self-paced [12]	88.52	87.03	81.63	–	–	67.55	63.63	53.51	–	–
Focal Loss [16]	93.03	86.45	80.45	–	–	70.02	61.87	54.13	–	–
Co-teaching [7]	89.87	82.83	75.41	–	–	63.31	54.13	44.85	–	–
D2L [17]	92.02	87.66	83.89	–	–	68.11	63.48	51.83	–	–
Fine-tuning [24]	**93.23**	82.47	74.07	–	–	**70.72**	56.98	46.37	–	–
MentorNet [10]	92.13	86.3	81.76	–	–	70.24	61.97	52.66	–	–
L2RW [23]	89.25	87.86	85.66	–	–	64.11	57.47	50.98	–	–
GLC [8]	91.02	89.68	88.92	–	–	65.42	63.07	**62.22**	–	–
MW-net [24]	92.04	90.33	87.54	–	–	70.11	64.22	58.64	–	–
CrossEntropy†	92.33	90.56	86.25	26.67	13.58	70.18	**65.02**	50.25	18.67	4.32
MW-net† [24]	92.19	90.74	87.63	42.41	27.19	70.57	64.13	51.23	19.89	7.42
Ours	91.87	**91.09**	**90.26**	**89.34**	**82.47**	68.93	63.54	59.07	**56.13**	**20.29**

We introduced also two new noise settings, namely Flip2 and Flip3. The difference from Flip is that the noise is equally distributed over multiple similar classes, two and three respectively. Table 2 and 3 show respectively the result for noise of type Flip2 and Flip3. We can see how our method performs better than the others, especially in very noisy situations.

Table 2. Accuracy result on CIFAR10 and CIFAR100 dataset with Flip2 noise. p denotes the different level of noise. † indicates the results obtained by our implementation. The first and the second best results are respectively marked with bold and underline.

Dataset	Flip2 CIFAR-10				Flip2 CIFAR-100			
Noise p	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8
CrossEntropy†	90.71	87.83	75.83	11.86	64.91	57.7	36.55	7
MW-net† [24]	**90.93**	88.83	86.85	27.49	**65.37**	**59**	36.97	7.99
Ours	90.66	**89.72**	**87.75**	**73.83**	63.07	57.96	**45.35**	**22.41**

Table 4 shows the results on Clothing1M. As we can see our method outperforms the current state-of-the-art result.

Table 3. Result for Flip3 noise on CIFAR10 and CIFAR100 dataset. p denotes the different level of noise. † indicates the results obtained by our implementation. The first and the second best results are respectively marked with bold and underline.

Dataset	Flip3 CIFAR-10				Flip3 CIFAR-100			
Noise p	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8
CrossEntropy†	90.13	88.44	82.31	20.34	<u>65.29</u>	<u>59.35</u>	44	<u>11.07</u>
MW-net† [24]	**90.56**	<u>88.49</u>	<u>85.65</u>	<u>22.69</u>	**65.33**	**62.74**	<u>45.77</u>	10.33
Ours	<u>90.31</u>	**88.96**	**87.73**	**75.53**	62.98	59.08	**52.28**	**25.72**

Table 4. Comparison with state-of-the-art methods in test accuracy (%) on Clothing1M dataset. Results for baselines are copied from original papers.

Method	Accuracy (%)
CrossEntropy [24]	68.94
F-correction [20]	69.84
JoCoR [29]	70.30
S-Model [6]	70.36
M-correction [2]	71.00
MLC [28]	71.06
Joint-Optim [25]	72.16
MLNT [14]	73.47
P-correction [32]	73.49
MW-Net [24]	73.72
MentorMix [9]	74.30
FaMUS [31]	74.43
DivideMix [13]	74.76
AugDesc [19]	75.11
Ours	**75.35**

5 Conclusions

In this paper, we introduced Meta Feature Re-Weighting (MFRW), which makes use of a novel concept of advisor network to mitigate the problem of training DNNs on corrupted labels. We empirically show the effectiveness of our method on a synthetic and real-world noisy dataset for the classification task. The experimental results demonstrate that advisor strategy can leverage information present in noisy data helping the main network to achieve a better generalization performance. Our method yields state-of-the-art performance on the Clothing1M dataset. Future research in this area may include adapting the advisor network to different problems than noise, like class imbalance.

References

1. Algan, G., Ulusoy, I.: Image classification with deep learning in the presence of noisy labels: a survey. Knowl.-Based Syst. **215**, 106771 (2021)
2. Arazo, E., Ortego, D., Albert, P., O'Connor, N., McGuinness, K.: Unsupervised label noise modeling and loss correction. In: International Conference on Machine Learning, pp. 312–321. PMLR (2019)
3. Azadi, S., Feng, J., Jegelka, S., Darrell, T.: Auxiliary image regularization for deep CNNs with noisy labels. arXiv preprint arXiv:1511.07069 (2015)
4. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 41–48 (2009)
5. Cheng, Y., Jiang, L., Macherey, W., Eisenstein, J.: AdvAug: robust adversarial augmentation for neural machine translation. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 5961–5970 (2020)
6. Goldberger, J., Ben-Reuven, E.: Training deep neural-networks using a noise adaptation layer (2016)
7. Han, B., et al.: Co-teaching: robust training of deep neural networks with extremely noisy labels. In: Advances in Neural Information Processing Systems (2018)
8. Hendrycks, D., Mazeika, M., Wilson, D., Gimpel, K.: Using trusted data to train deep networks on labels corrupted by severe noise. In: Advances in Neural Information, vol. 31, pp. 10456–10465 (2018)
9. Jiang, L., Huang, D., Liu, M., Yang, W.: Beyond synthetic noise: deep learning on controlled noisy labels. In: International Conference on Machine Learning, pp. 4804–4815. PMLR (2020)
10. Jiang, L., Zhou, Z., Leung, T., Li, L.J., Fei-Fei, L.: MentorNet: learning data-driven curriculum for very deep neural networks on corrupted labels. In: International Conference on Machine Learning, pp. 2304–2313. PMLR (2018)
11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, vol. 25 (2012)
12. Kumar, M., Packer, B., Koller, D.: Self-paced learning for latent variable models. In: Advances in Neural Information Processing Systems, vol. 23, pp. 1189–1197 (2010)
13. Li, J., Socher, R., Hoi, S.C.: DivideMix: learning with noisy labels as semi-supervised learning. In: International Conference on Learning Representations (2019)
14. Li, J., Wong, Y., Zhao, Q., Kankanhalli, M.S.: Learning to learn from noisy labeled data. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5051–5059 (2019)
15. Li, Y., Yang, J., Song, Y., Cao, L., Luo, J., Li, L.J.: Learning from noisy labels with distillation. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1910–1918 (2017)
16. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2980–2988 (2017)
17. Ma, X., et al.: Dimensionality-driven learning with noisy labels. In: International Conference on Machine Learning, pp. 3355–3364. PMLR (2018)
18. Nettleton, D.F., Orriols-Puig, A., Fornells, A.: A study of the effect of different types of noise on the precision of supervised learning techniques. Artif. Intell. Rev. **33**(4), 275–306 (2010)

19. Nishi, K., Ding, Y., Rich, A., Hollerer, T.: Augmentation strategies for learning with noisy labels. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8022–8031 (2021)
20. Patrini, G., Rozza, A., Krishna Menon, A., Nock, R., Qu, L.: Making deep neural networks robust to label noise: a loss correction approach. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1944–1952 (2017)
21. Pechenizkiy, M., Tsymbal, A., Puuronen, S., Pechenizkiy, O.: Class noise and supervised learning in medical domains: the effect of feature extraction. In: 19th IEEE Symposium on Computer-Based Medical Systems (CBMS 2006), pp. 708–713. IEEE (2006)
22. Reed, S., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., Rabinovich, A.: Training deep neural networks on noisy labels with bootstrapping. arXiv preprint arXiv:1412.6596 (2014)
23. Ren, M., Zeng, W., Yang, B., Urtasun, R.: Learning to reweight examples for robust deep learning. In: International Conference on Machine Learning, pp. 4334–4343. PMLR (2018)
24. Shu, J., et al.: Meta-Weight-Net: learning an explicit mapping for sample weighting. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems, pp. 1919–1930 (2019)
25. Tanaka, D., Ikami, D., Yamasaki, T., Aizawa, K.: Joint optimization framework for learning with noisy labels. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5552–5560 (2018)
26. Vahdat, A.: Toward robustness against label noise in training deep discriminative neural networks. In: Advances in Neural Information Processing Systems, vol. 30, pp. 5596–5605 (2017)
27. Veit, A., Alldrin, N., Chechik, G., Krasin, I., Gupta, A., Belongie, S.: Learning from noisy large-scale datasets with minimal supervision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 839–847 (2017)
28. Wang, Z., Hu, G., Hu, Q.: Training noise-robust deep neural networks via meta-learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4524–4533 (2020)
29. Wei, H., Feng, L., Chen, X., An, B.: Combating noisy labels by agreement: a joint training method with co-regularization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13726–13735 (2020)
30. Xiao, T., Xia, T., Yang, Y., Huang, C., Wang, X.: Learning from massive noisy labeled data for image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2691–2699 (2015)
31. Xu, Y., Zhu, L., Jiang, L., Yang, Y.: Faster meta update strategy for noise-robust deep learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 144–153 (2021)
32. Yi, K., Wu, J.: Probabilistic end-to-end noise correction for learning with noisy labels. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7017–7025 (2019)
33. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: beyond empirical risk minimization. In: International Conference on Learning Representations (2018)