Google AI

Google Research

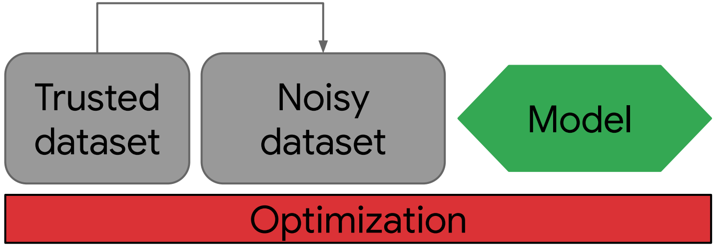
Distill Effective Supervision from Severe Label Noise

Zizhao Zhang | Han Zhang | Sercan Ö. Arık | Honglak Lee | Tomas Pfister
Google Cloud AI, Google Brain

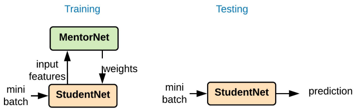
Noisy label in Practice

Practically-common scenario

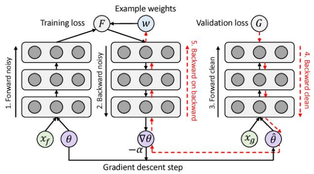
Crowd-sourcing,
web search



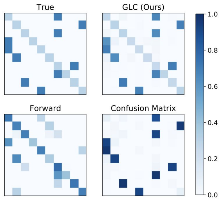
Previous work



MentorNet, Jiang et al. ICML 2018

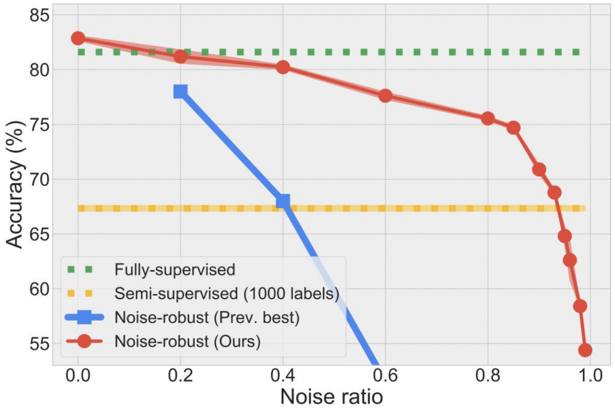


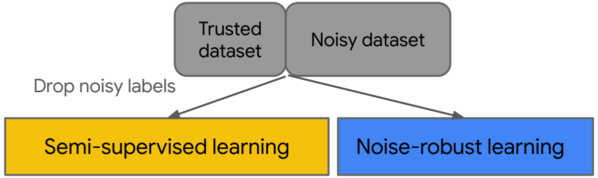
Learning-to-reweight, Ren et al, ICML 2018



TruestedData, Hendrycks et al., NeuIPS 2019

Experimented on CIFAR100 uniform noise



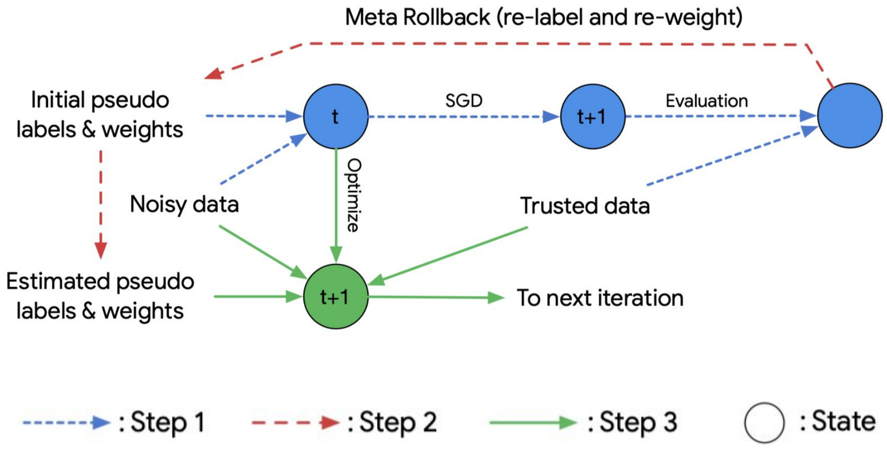


Green line: Fully-supervised baseline without label noise.
Blue line: Noise-robust methods can be severely affected if the label noise ratio is high, e.g. > 50% label noise.
Yellow line: Semi-supervised learning (SSL) methods, which discard labels of the large noisy-label dataset.
Red line: Our method significantly improves noise-robust training.

Previous methods still suffer from high label noise.

How can do better utilize the hidden correct labels in the big noisy-label datasets?

Our method estimates
Data Coefficients
with a generalized meta learning framework to distill effective supervision from label noise.



Key training steps

- Obtain initial pseudo label candidates
- Contrust meta re-labeling and re-weighting in a generalized meta learning framework. Re-labeling is formulated as a differential selection problem between estimated labels and original labels.
- Construct composed losses with estimated data coefficients.
- Train a model for one step.

Key insights (see paper):

- Better initial pseudo labels
- Better regularizations

Initial Pseudo Labels

Pseudo label estimator $g(x_i, \Phi)$ average predictions of augmentations and then apply softmax temperature calibration

$$g(x, \Phi)_i = Pr_i^{\frac{1}{\tau}} / \sum_i Pr_i^{\frac{1}{\tau}}, \text{ where } Pr = \frac{1}{K}\left(\Phi(x) + \sum_k^{K-1} \Phi(\hat{x}_k)\right)$$

For augmentation, we use AutoAugment/RandAugment:

geomatic/color transformation →flip→random crop→cutout

Inspired by MixMatch, Berthelot et al, NeurIPS, 2019

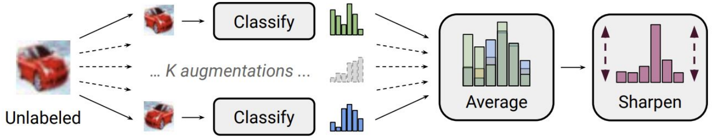
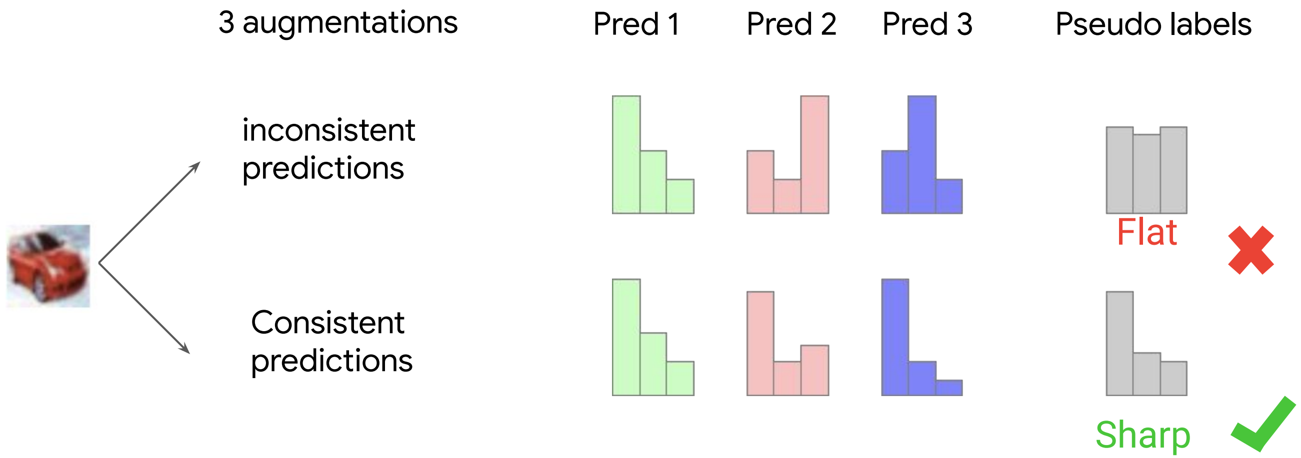


Figure 1: Diagram of the label guessing process used in MixMatch. Stochastic data augmentation is applied to an unlabeled image K times, and each augmented image is fed through the classifier. Then, the average of these K predictions is "sharpened" by adjusting the distribution's temperature. See algorithm 1 for a full description.

Pseudo labels need consistent predictions



Consistency enforcing

$$\min_{\Theta} \; L_{KL} = \frac{1}{|D_u|} \sum_{i}^{|D_u|} \mathrm{KL}\big(\Phi(x_i) \,||\, \Phi(\hat{x}_i)\big)$$

Algorithm 1: A training step of our method at time step t

Input: Current model parameters Θ^t, A batch of training data X_u from D_u, a batch of probe data X_p from D_p, loss weight k and p, threshold T

Output: Updated model parameters Θ^{t+1}

1 Generate the augmentation $\hat{X}_u$ of X_u.

2 Estimate the pseudo labels via $g(x_u, \Phi), x_u \sim X_u \cup \hat{X}_u$ (Section 4.1 & 4.2).

3 Compute optimal data coefficients λ^* and ω^* via the meta step (Section 4.3).

4 Split the training batch X_u (also corresponding $\hat{X}_u$) to possible clean batch X_u^c and possible mislabeled batch X_u^u using the binary criterion $\mathbb{I}(\omega^* < T)$.

5 Construct the joint batch set (Section 4.4),

$$X_p \cup X_u^u \cup X_u^c \cup \hat{X}_u^u \cup \hat{X}_u^c,$$

where $\hat{X}_u^u \cup X_u^u$ uses pseudo labels estimated by $g(\cdot, \Phi)$.

6 Compute the total loss for model update

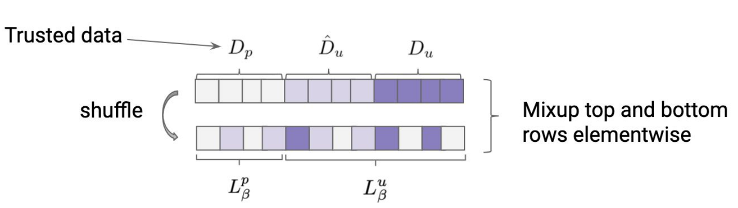
$$L_{\omega^*} + L_{\lambda^*} + L_\beta^p + p\, L_\beta^u + k\, L_{\mathrm{KL}}.$$

7 Conduct one step stochastic gradient descent to obtain Θ^{t+1}.

The training losses are composted by multiple cross-entropy losses using learned data coefficients (weights and pseudo labels)

Introduce probe data in actual updating:

MixUp is used to "gently" introduce the probe data with possibly-noisy data as training data



Experiments

State-of-the-art over many benchmarks

Two used networks: WRN28-10 (default) and ResNet29 (very light)

Method	M	Noise ratio			
		0	0.2	0.4	0.8
GCE [48]	-	93.5	89.9±0.2	87.1±0.2	67.9±0.6
MentorNet DD [17]	5k	96.0	92.0	89.0	49.0
RoG [20]	-	94.2	87.4	81.8	-
L2R [33]	1k	96.1	90.0±0.4*	86.9±0.2	73.0±0.8*
Arazo *et al.* [1]	-	93.6	94.0	92.0	86.8
Ours-RN29	0.1k	94.4	92.9±0.2	92.5±0.5	85.6+1.1
Ours	0.01k	96.8	95.4±0.6	94.5±1.0	87.9±5.1
Ours	0.05k	96.8	96.4±0.0	95.5±0.6	91.8±3.0
Ours	0.1k	96.8	**96.2±0.2**	**95.9±0.2**	**93.7±0.5**

0.01k: 1 probe image per class

Table 1: CIFAR10 with uniform noises.

- Upto 9% (86.8% -> 93.7%) improvement.
- Outperform others with a much smaller ResNet and uses 1 trusted train data/class.

Method	M	Noise ratio			
		0	0.2	0.4	0.8
GCE [48]	-	81.4	66.8±0.4	61.8±0.2	47.7±0.7
MentorNet [17]	5k	79.0	73.0	68.0	35.0
L2R [33]	1k	81.2	67.1±0.1*	61.3+2.0	35.1±1.2*
Arazo *et al.* [1]		70.3	68.7	61.7	48.2
Ours-RN29	1k	72.1	69.3±0.5	67.0±0.8	60.7±1.0
Ours	0.1k	83.0	77.4±0.4	75.1±1.1	62.1±1.2
Ours	0.5k	83.0	80.4±0.5	79.6±0.3	73.6±1.5
Ours	1k	83.0	**81.2±0.7**	**80.2±0.3**	**75.5±0.2**

0.1k: 1 probe image per class

Table 2: CIFAR100 with uniform noises.

- Upto 56% (48.2% -> 75.5%) improvement.
- Outperform others with a much smaller ResNet and uses 1 trusted train data/class.

Method	Noise ratio		
	0.2	0.4	0.8
GCE [48]	89.5±0.3	82.3±0.7	-
LC [30]	89.1±0.5	83.6±0.3	-
Ours-RN29	92.7±0.2	90.2±0.5	78.9±3.5
Ours	**96.5±0.2**	**94.9±0.1**	**79.3±2.4**

Table 1: Asymmetric noise on CIFAR10.

Method	CIFAR10 (34%)	CIFAR100 (37%)
RoG [20]	70.0	53.6
L2R* [33]	71.0	56.9
Ours-RN29	81.8	65.1
Ours	**88.3**	**73.7**

* Trained by us

Table 2: Experiments with semantic noise where labels are generated by a neural network trained on limited data. The resulting noise ratio is shown in parentheses.

Method	mini	full
Co-teaching [13]	61.5/84.7	-
Chen *el al.* [5]	61.6/85.0	-
MentorNet [17]	63.8/85.8	64.2/84.8
Ours-RN50	78.0/94.4	65.8/85.8
Ours	**80.0/94.9**	**69.0/88.3**

mini: 60k (50 class) full: 2M (1000 class)

Table 1: WebVision 2M comparison the on min and full version (10 clean ImageNet training images per class is used).

- Upto 25% (63.8% -> 80.0%) improvement.
- Outperform MentorNet even with a much smaller ResNet50 compared with default InceptionResNetv2.

Method	Accuracy
ResNet50 [22]	81.44
CleanNet [22]	83.95
Self-Learning [14]	85.11
Ours-RN50	**87.57**

Table 2: Food101N comparison.

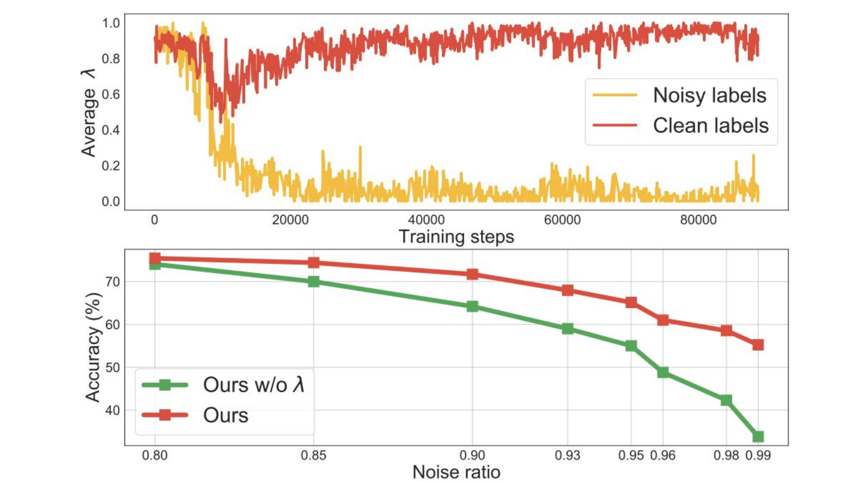
Data coefficients: exemplar weights and labels

$$\Theta^*(\boxed{\omega}, \boxed{\lambda}) = \arg\min_{\Theta} \sum_{i=1} \omega_i L(\mathcal{P}(\lambda_i), \Phi(x_i; \Theta)),$$

$$\mathcal{P}(\lambda_i) = \lambda_i y_i + (1 - \lambda_i) g(x_i, \Phi) \ \ s.t.\ 0 \leq \lambda_i \leq 1$$

Binary selection formulation: Smaller \lambda favors pseudo labels

Study on CIFAR100



Our method

- Estimates Data Coefficients, exemplar weights and labels, to distill effective supervision for noise-robust model training.
- Significantly outperforms previous methods and sets new state of the arts on most benchmarks.

https://github.com/google-research/google-research/tree/master/ieg