How to train your model when you can not trust on the annotations?

Authors: Filipe Cordeiro (filipe.rolim@ufrpe.br)
Gustavo Carneiro (gustavo.carneiro@adelaide.edu.au)

SIBGRAPI 2020 33rd Conference on Graphics, Patterns and Images

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO
THE UNIVERSITY of ADELAIDE

Material

- Slides, codes, link to the paper

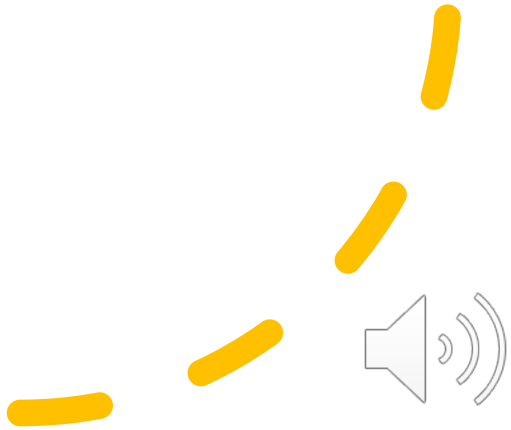
github.com/filipe-research/tutorial_noisylabels

Outline

- Motivation
- Definition and Fundaments
- Related Work
- Practical Examples
- Trends and opportunities

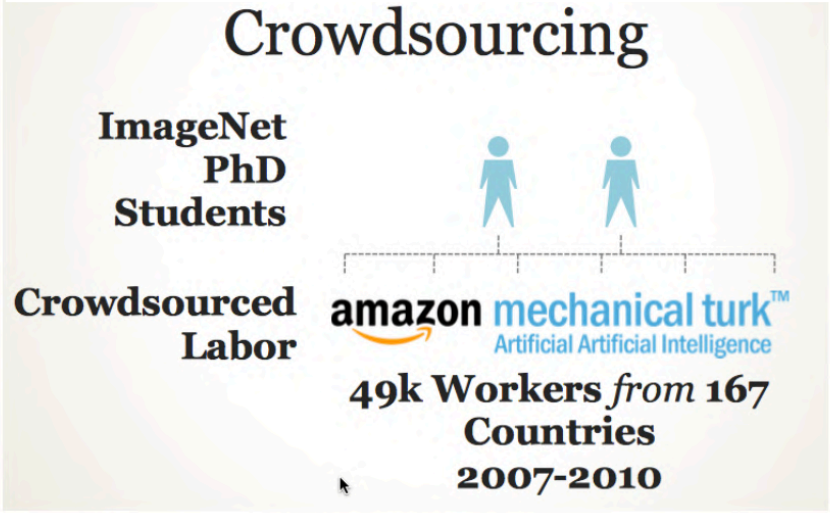
Motivation

State-of-the-art models require *large amounts* of **clean**, *annotated data*.



Motivation

- Annotation is labour intensive!!



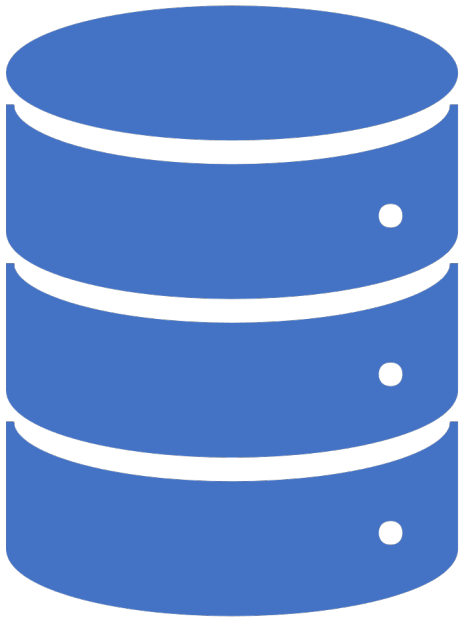
Slide from Fei-Fei Li and Jia Deng

ImageNet: 15 million labeled images; over 20,000 classes

The data that transformed AI research—and possibly the world (D. Gershgorn, quartz, magazine, 2017)

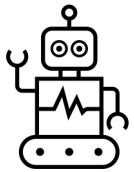
- 49k workers
- 167 countries
- 2.5 years to complete!

How difficult it is to build a clean annotated dataset?



Annotating Datasets

Human

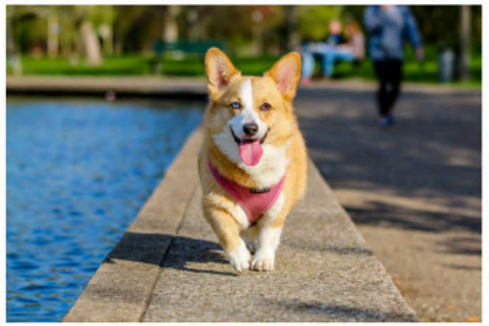


Automatic

Manual Annotation/ Crowdsourcing

What animal is this?

A: Dog
B: Cat
C: Owl



What animal is this?

A: Dog
B: Cat
C: Owl

What animal is this?

A: Dog
B: Cat
C: Owl

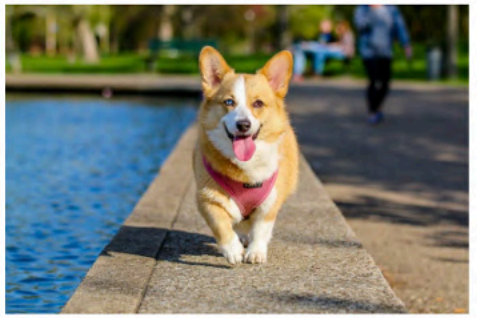
What animal is this?

A: Dog
B: Cat
C: Owl

Error in responses

What animal is this?

A: Dog
B: Cat **X**
C: Owl



What animal is this?

A: Dog **X**
B: Cat
C: Owl

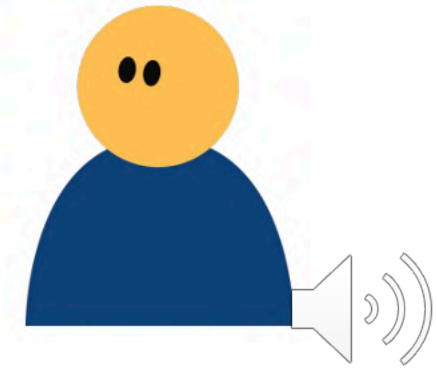
What animal is this?

A: Dog **X**
B: Cat
C: Owl

What animal is this?

A: Dog
B: Cat **X**
C: Owl

- Errors can happen when doing tasks quickly



Error in responses

- low quality data

What animal is this?
A: Dog
B: Cat
C: Deer

What animal is this?
A: Dog
B: Cat
C: Deer

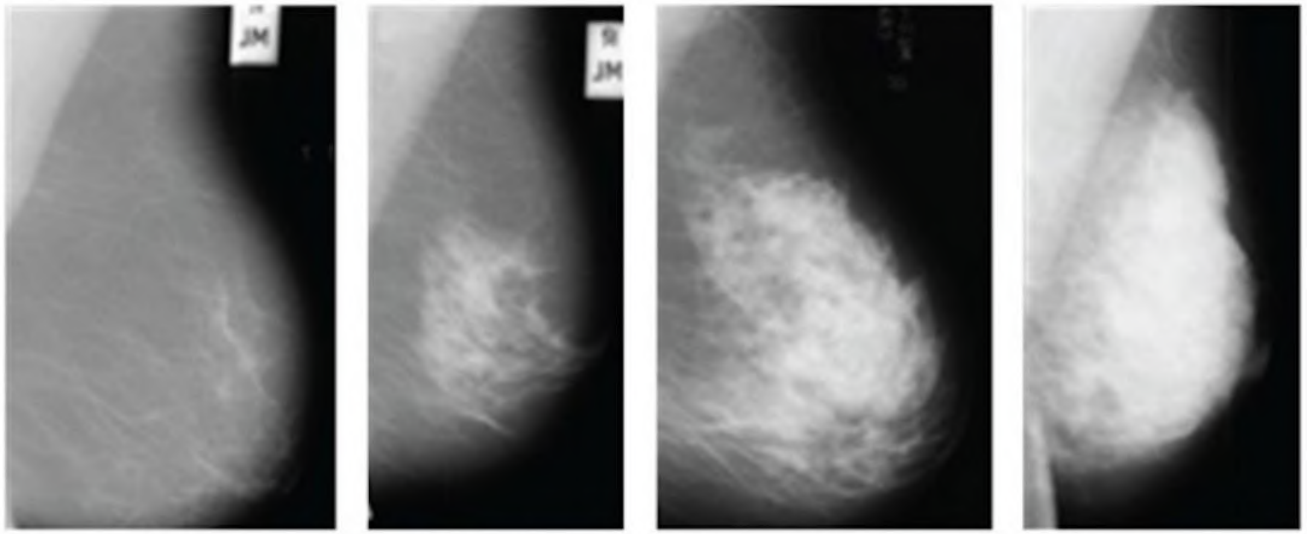
What animal is this?
A: Dog
B: Cat
C: Deer

What animal is this?
A: Dog
B: Cat
C: Deer

E.g: CIFAR-10 dataset

Error in responses

- Challenging tasks



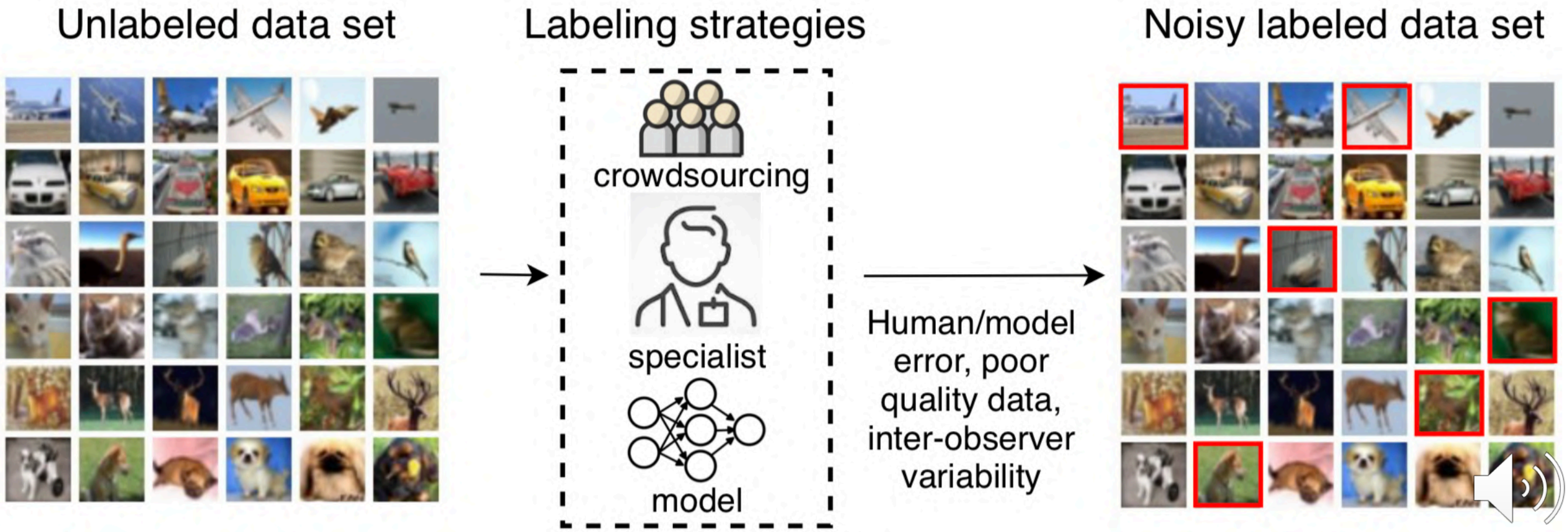
Mammography images

Dermatostopic images

Motivation

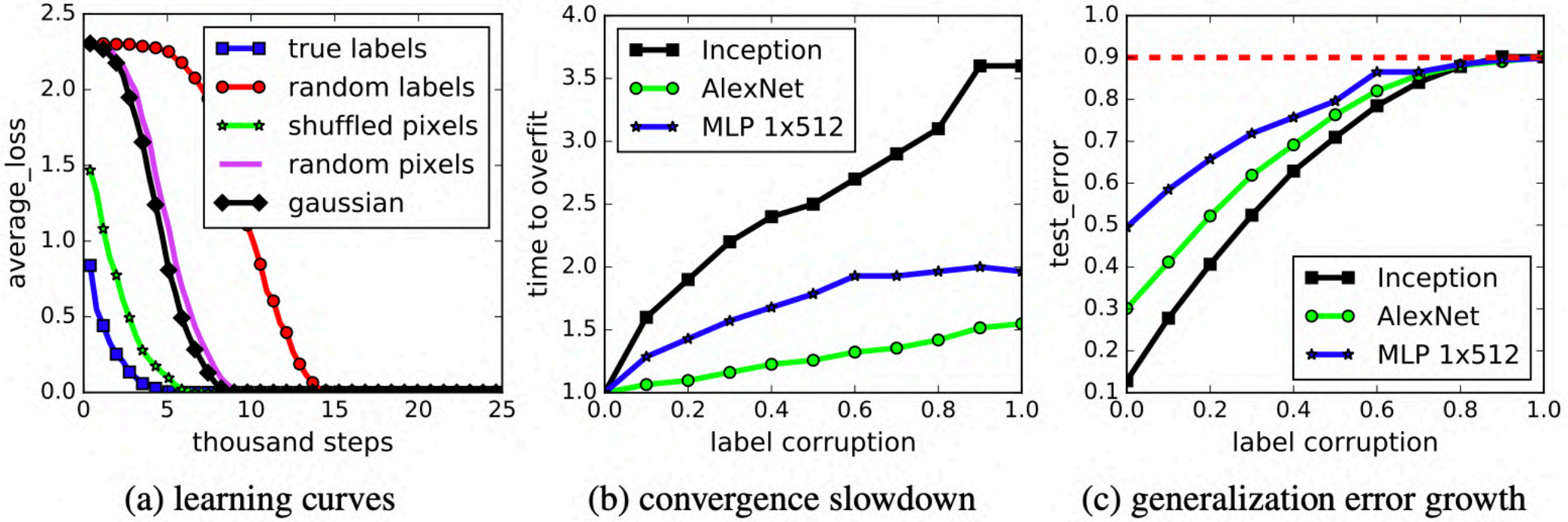
Noisy labels may occur naturally when human annotators are involved

[D. McNicol, *A primer of signal detection theory*. Psycology Press, 2005]



Why label noise is relevant?

- “Deep Neural Networks easily fit random labels” [1]



(a) learning curves (b) convergence slowdown (c) generalization error growth

[1] Zhang et al., “Understanding Deep Learning Requires Re-thinking Generalization”, ICLR 2017.

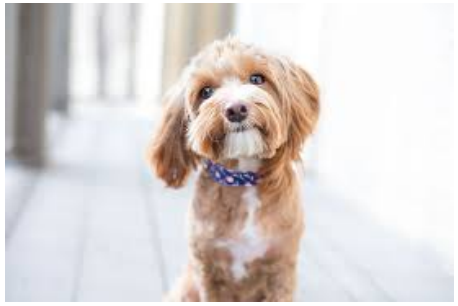
Definition and Taxonomy

Problem Definition

- Noisy labels denotes the presence of labels which are different from true class.

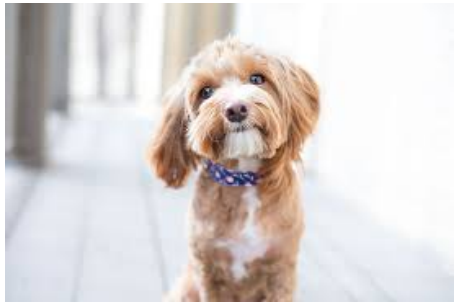
Problem Definition

- Noisy labels denotes the presence of labels which are different from true class.

Image x	Hidden clean label $\hat{y} \in \Upsilon$
	Dog
	Dog

Problem Definition

- Noisy labels denotes the presence of labels which are different from true class.

Image x	Hidden clean label $\hat{y} \in \Upsilon$	observed label $y \in \Upsilon$
	Dog	Dog
	Dog	Cat

Problem Definition

- Dataset of noisy labels: $D = \{x_i, y_i\}_{i=1}^{|D|}$
- Image: x_i
- Observed label: $\mathrm{y_i} \in \Upsilon$
- Hidden true label: $\widehat{y_i} \in \Upsilon$
- $y_i \sim p(y|x_i, \Upsilon, \widehat{y_i})$

Problem Definition

$$p(y = j | x_i, \hat{y}_i = c) = \eta_{jc}(x_i)$$

$$\sum_{j \in \Upsilon} \eta_{jc}(x_i) = 1$$

Problem Definition

$$p(y = j|x_i, \hat{y}_i = c) = \eta_{jc}(x_i)$$

$$\sum_{j \in Y} \eta_{jc}(x_i) = 1$$

Noisy Label \ True Label	0	1	2	3	4
0	0.6	0.1	0.2	0.1	0.0
1	0.1	0.8	0.1	0.0	0.0
2	0.1	0.0	0.5	0.6	0.0
3	0.1	0.1	0.2	0.2	0.0
4	0.1	0.0	0.0	0.1	1.0

Problem Definition

$$p(y = j|x_i, \hat{y}_i = c) = \eta_{jc}(x_i)$$

$$\sum_{j \in Y} \eta_{jc}(x_i) = 1$$

Noisy Label \ True Label	0	1	2	3	4
0	0.6	0.1	0.2	0.1	0.0
1	0.1	0.8	0.1	0.0	0.0
2	0.1	0.0	0.5	0.6	0.0
3	0.1	0.1	0.2	0.2	0.0
4	0.1	0.0	0.0	0.1	1.0

Problem Definition

$$p(y = j|x_i, \hat{y}_i = c) = \eta_{jc}(x_i)$$

$$\sum_{j \in \Upsilon} \eta_{jc}(x_i) = 1$$

$$\eta_{23}(x_i) = p(y = 2|x_i, \hat{y}_i = 3)$$

$$\eta_{23}(x_i) = 0.6$$

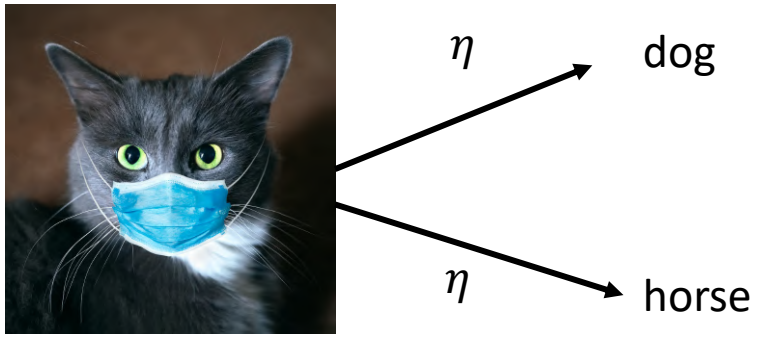
Noisy Label \ True Label	0	1	2	3	4
0	0.6	0.1	0.2	0.1	0.0
1	0.1	0.8	0.1	0.0	0.0
2	0.1	0.0	0.5	0.6	0.0
3	0.1	0.1	0.2	0.2	0.0
4	0.1	0.0	0.0	0.1	1.0

Types of Noise

- Symmetric
- Asymmetric
- Semantic
- Open-set

Types of Noise: Symmetric Noise

- Also called random or uniform noise
- A label has **equal probability** η to flip to another class

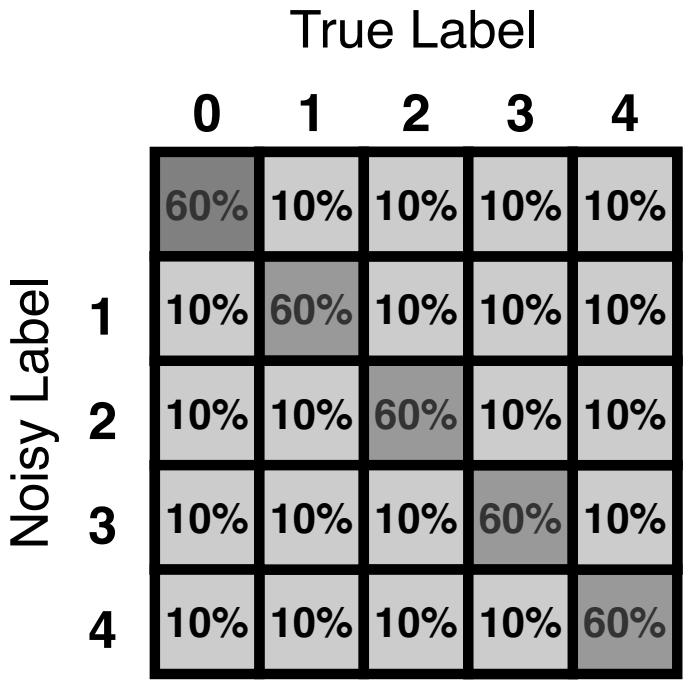


Types of Noise: Symmetric Noise

- Example

$\eta = 0.4$

$\eta_{jc}(x_i) = \frac{\eta}{C-1}$ $\eta_{cc} = 1 - \eta$



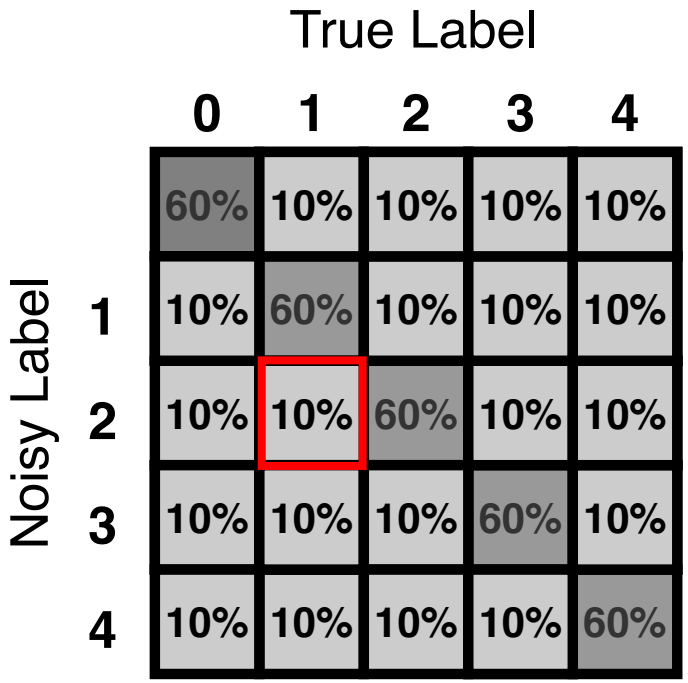
Types of Noise: Symmetric Noise

- Example

$\eta = 0.4$

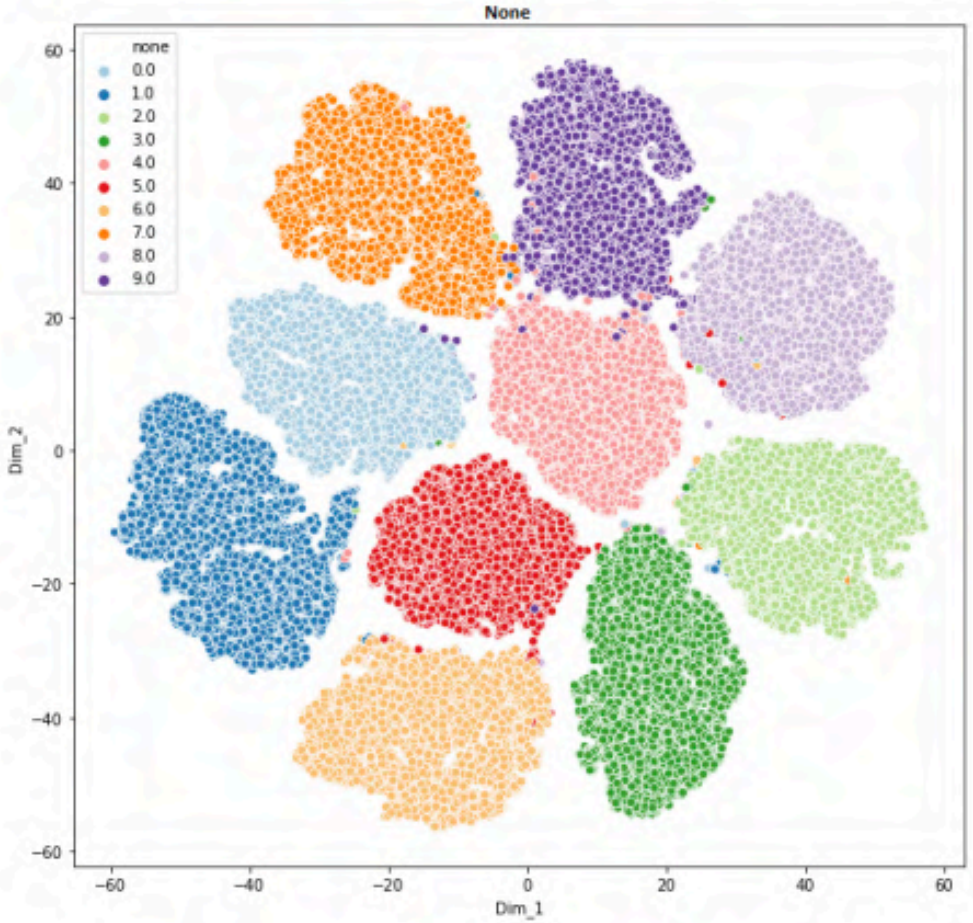
$\eta_{jc}(x_i) = \frac{\eta}{C-1}$ $\eta_{cc} = 1 - \eta$

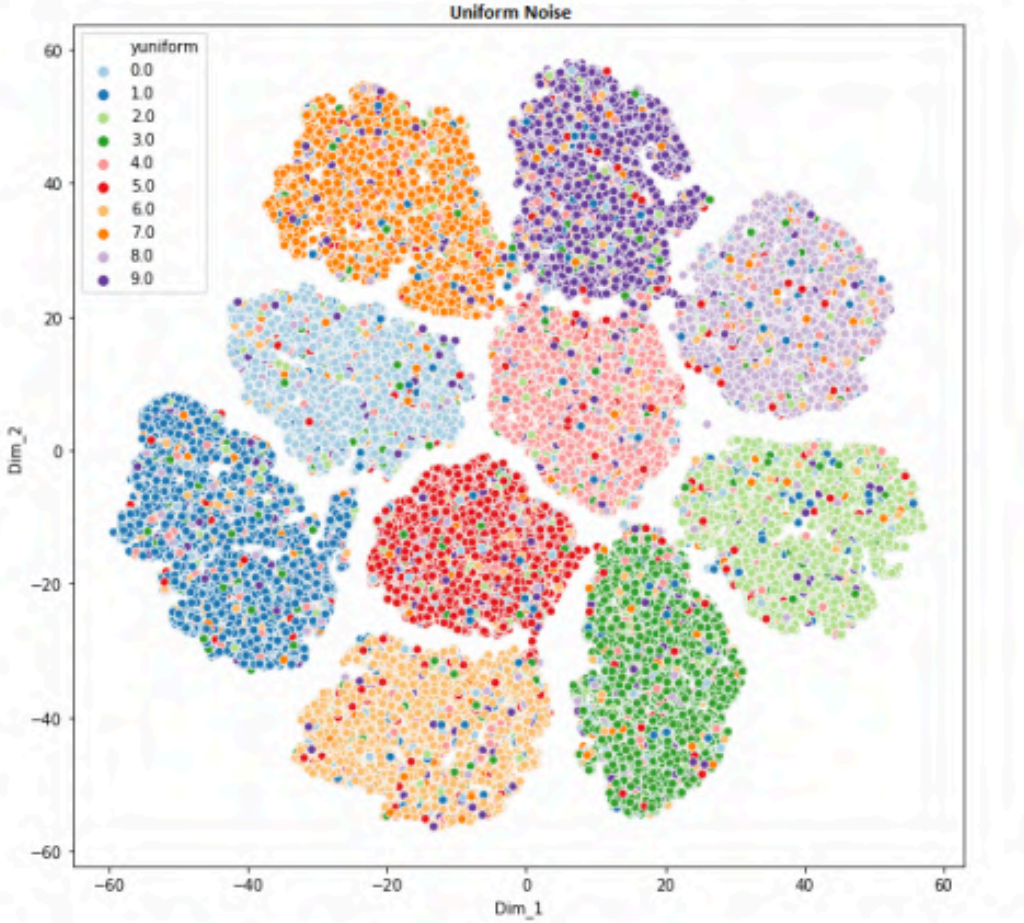
$\eta_{21}(x_i) = \frac{0.4}{5-1} = 0.1$



Symmetric Noise

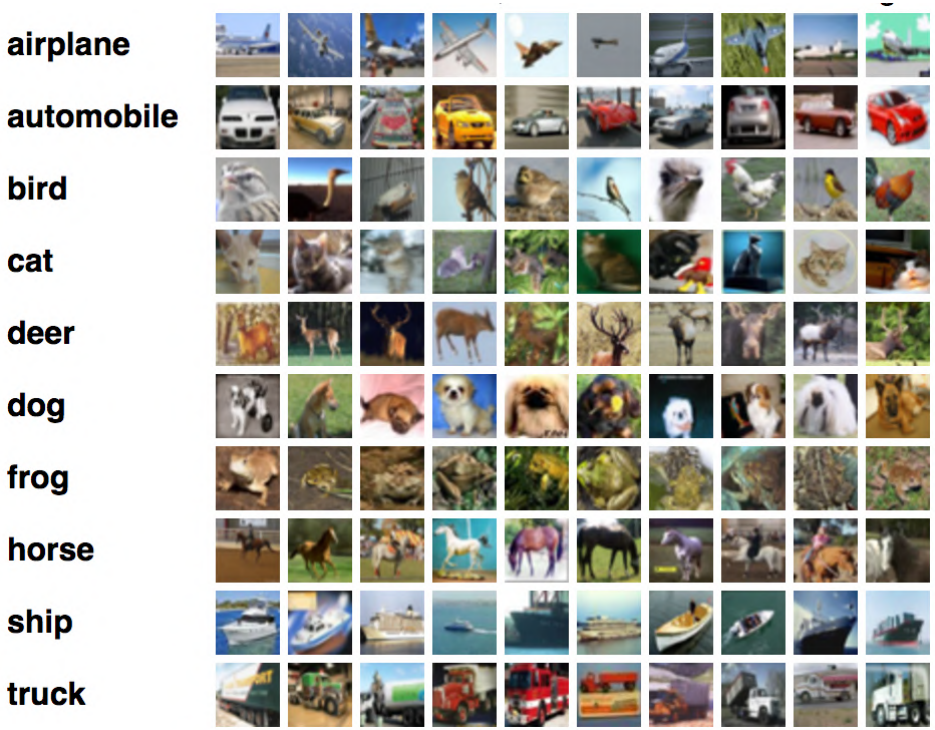
- Tsne-plot





Asymmetric Noise

- Flip labels between similar classes
  - Closer to real-world label noise



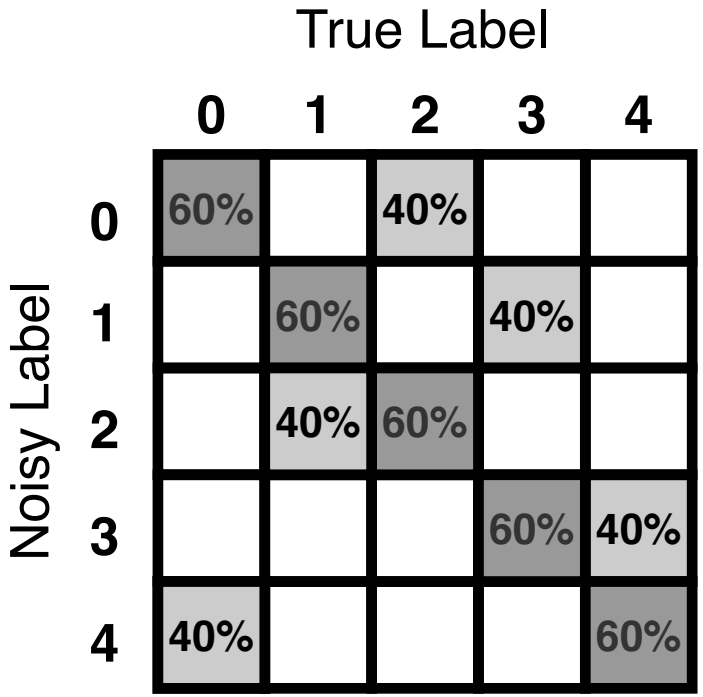
Truck→ automobile

bird→ plane

deer→ horse

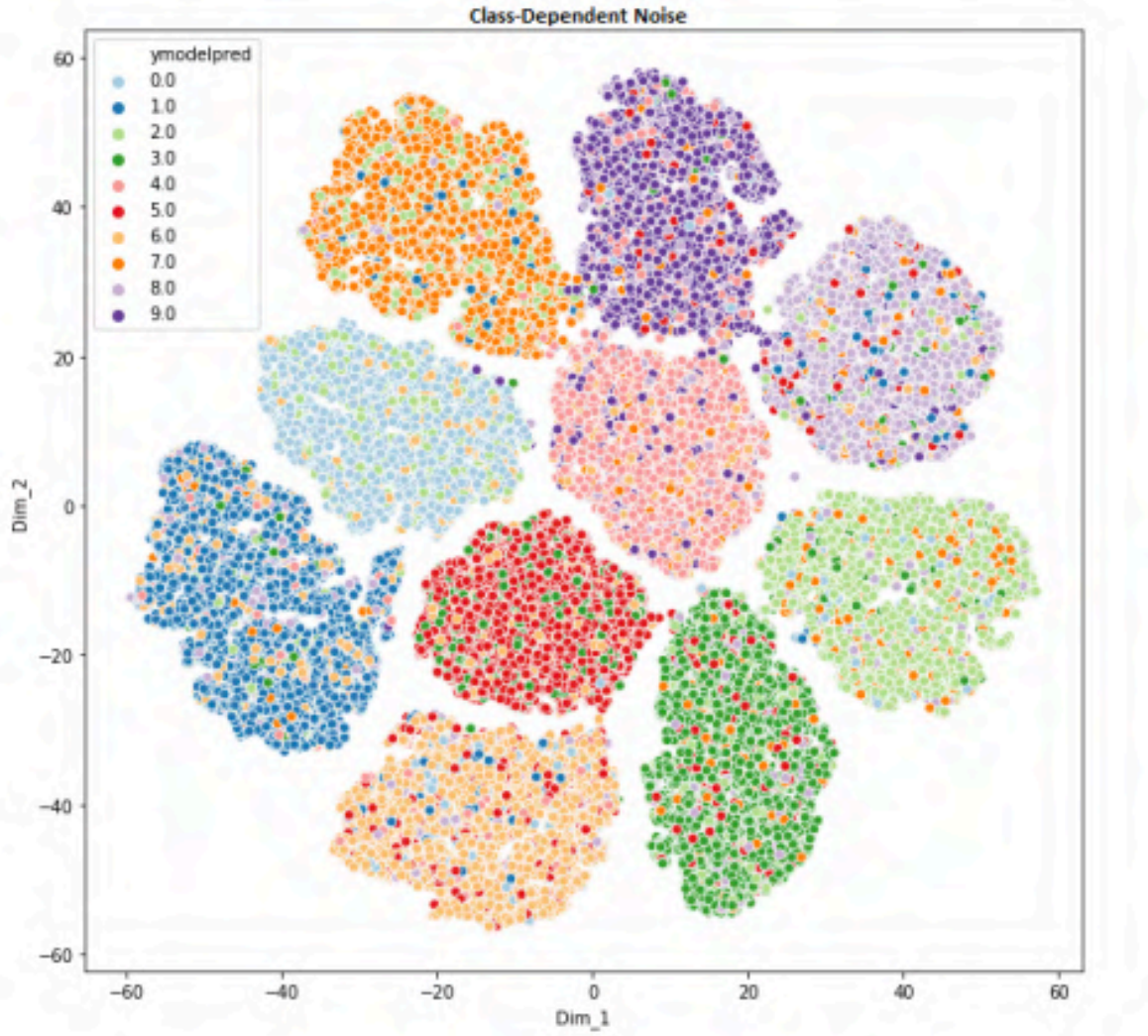
Asymmetric Noise

- η_{jc} is class conditional



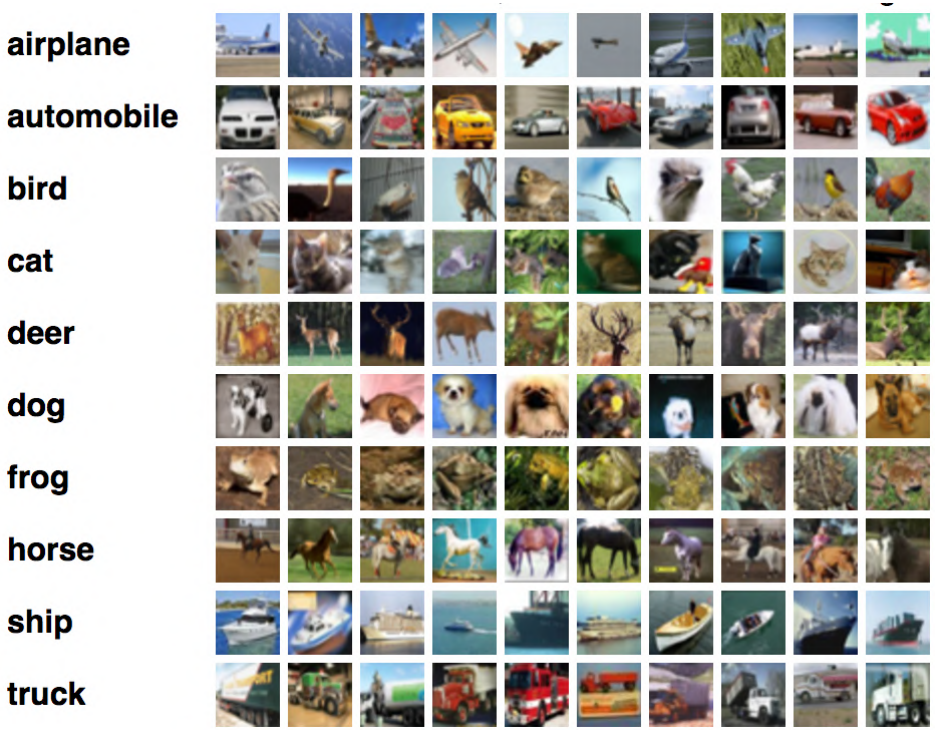
Asymmetric Noise

- Tsne plot



Semantic noise

- Depends on both classes $j, c \in \Upsilon$ and the image x_i



Truck→ automobile

bird→ plane

deer→ horse

Types of Noise

- Open-set noise

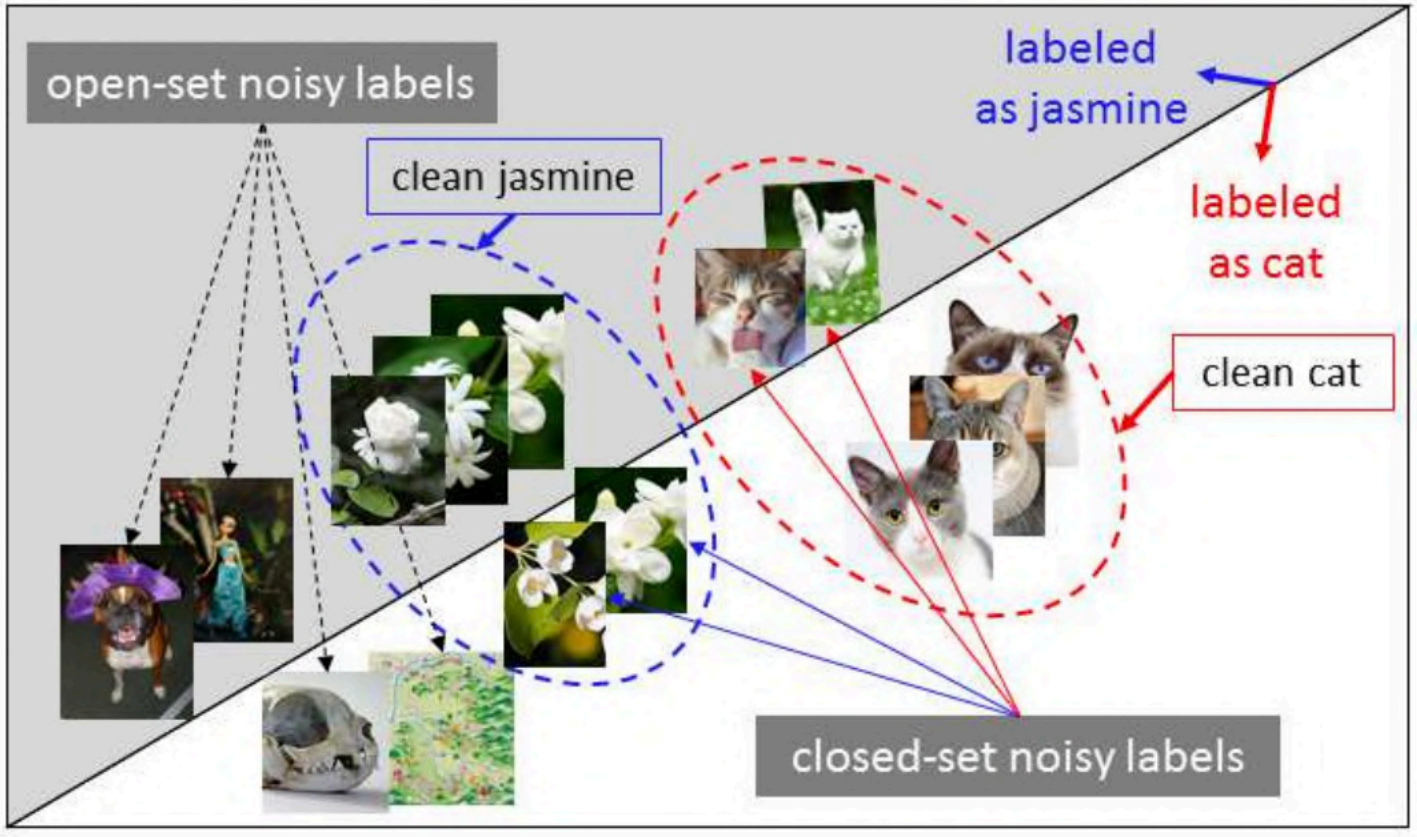
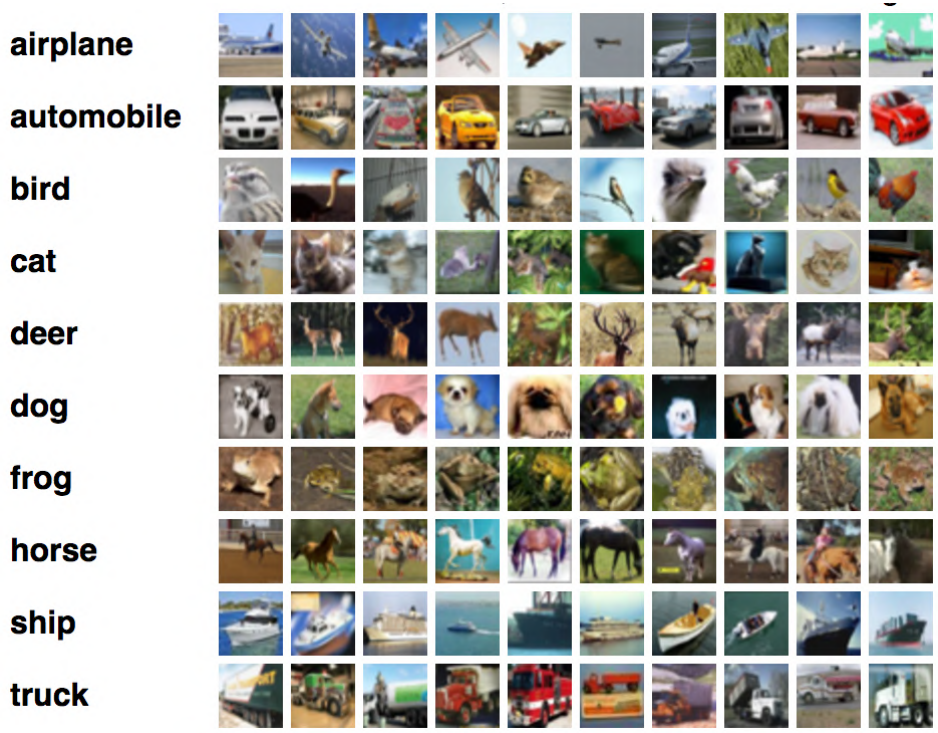


Figure from [Wang et al., Iterative Learning with Open-set Noisy Labels, CVPR, 2018]

Datasets

CIFAR-10/CIFAR-100

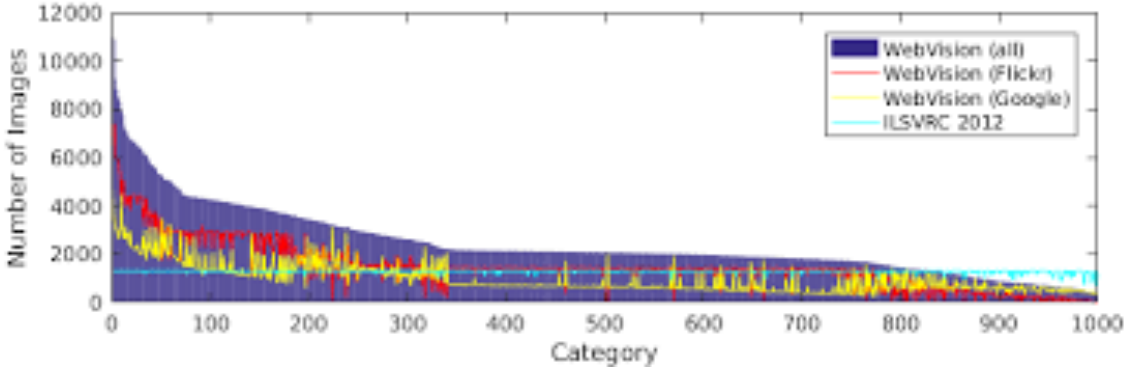


Training size	Test size	Image size	# classes	Estimated noise
50000	10000	32x32	10/100	0%

Webvision

- 2.4 million of images crawled from the Flickr website and Google Images search

Training size	Test size	# classes	Estimated noise
2.4M	50000	1000	20%



Clothing1M

- 1 million training images acquired from online shopping websites, with labels generated by surrounding texts provided by sellers

Training size	Test size	# classes	Estimated noise
1M	10000	14	38.46%

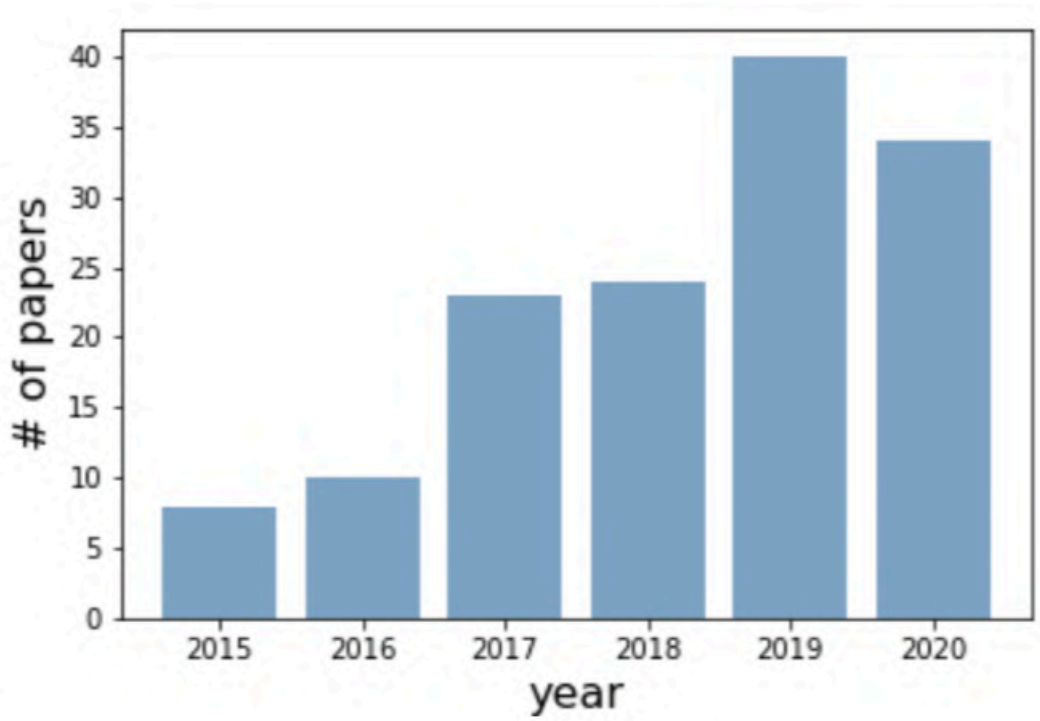
Food101-N

- Based on Food101 dataset, but noisy and with more images.

Training size	Test size	# classes	Estimated noise
310000	25000	101	19.66%

Literature Review

Growing interest



Number of publications related to noisy labels in the main conferences of computer vision and artificial intelligence, in the last five years.

Main approaches

- Transition Matrix
- Robust Losses
- Sample Weighting
- Sample Selection
- Ensemble
- Meta-Learning

Datasets, noise types, and SOTA results

[Li J, Socher R, Hoi SC. DivideMix: Learning with Noisy Labels as Semi-supervised Learning. ICLR 2020.]

- SOTA Results

Clothing 1M

Method	Test Accuracy
Cross-Entropy	69.21
F-correction (Patrini et al., 2017)	69.84
M-correction (Arazo et al., 2019)	71.00
Joint-Optim (Tanaka et al., 2018)	72.16
Meta-Cleaner (Zhang et al., 2019)	72.50
Meta-Learning (Li et al., 2019)	73.47
P-correction (Yi & Wu, 2019)	73.49
DivideMix	**74.76**

WebVision

Method	WebVision		ILSVRC12	
	top1	top5	top1	top5
F-correction (Patrini et al., 2017)	61.12	82.68	57.36	82.36
Decoupling (Malach & Shalev-Shwartz, 2017)	62.54	84.74	58.26	82.26
D2L (Ma et al., 2018)	62.68	84.00	57.80	81.36
MentorNet (Jiang et al., 2018)	63.00	81.40	57.80	79.92
Co-teaching (Han et al., 2018)	63.58	85.20	61.48	84.70
Iterative-CV (Chen et al., 2019)	65.24	85.34	61.60	84.98
DivideMix	**77.32**	**91.64**	**75.20**	**90.84**

Dataset		CIFAR-10				CIFAR-100			
Method/Noise ratio		20%	50%	80%	90%	20%	50%	80%	90%
Cross-Entropy	Best	86.8	79.4	62.9	42.7	62.0	46.7	19.9	10.1
	Last	82.7	57.9	26.1	16.8	61.8	37.3	8.8	3.5
Bootstrap (Reed et al., 2015)	Best	86.8	79.8	63.3	42.9	62.1	46.6	19.9	10.2
	Last	82.9	58.4	26.8	17.0	62.0	37.9	8.9	3.8
F-correction (Patrini et al., 2017)	Best	86.8	79.8	63.3	42.9	61.5	46.6	19.9	10.2
	Last	83.1	59.4	26.2	18.8	61.4	37.3	9.0	3.4
Co-teaching+* (Yu et al., 2019)	Best	89.5	85.7	67.4	47.9	65.6	51.8	27.9	13.7
	Last	88.2	84.1	45.5	30.1	64.1	45.3	15.5	8.8
Mixup (Zhang et al., 2018)	Best	95.6	87.1	71.6	52.2	67.8	57.3	30.8	14.6
	Last	92.3	77.6	46.7	43.9	66.0	46.6	17.6	8.1
P-correction* (Yi & Wu, 2019)	Best	92.4	89.1	77.5	58.9	69.4	57.5	31.1	15.3
	Last	92.0	88.7	76.5	58.2	68.1	56.4	20.7	8.8
Meta-Learning* (Li et al., 2019)	Best	92.9	89.3	77.4	58.7	68.5	59.2	42.4	19.5
	Last	92.0	88.8	76.1	58.3	67.7	58.0	40.1	14.3
M-correction (Arazo et al., 2019)	Best	94.0	92.0	86.8	69.1	73.9	66.1	48.2	24.3
	Last	93.8	91.9	86.6	68.7	73.4	65.4	47.6	20.5
DivideMix	Best	**96.1**	**94.6**	**93.2**	**76.0**	**77.3**	**74.6**	**60.2**	**31.5**
	Last	**95.7**	**94.4**	**92.9**	**75.4**	**76.9**	**74.2**	**59.6**	**31.0**

Table 1: Comparison with state-of-the-art methods in test accuracy (%) on CIFAR-10 and CIFAR-100 with symmetric noise. Methods marked by * denote re-implementations based on public code.

Background

- Cross-Entropy Loss:

$$\mathcal{L} = -\sum_{i=1}^{N} y_i^T \log(h_i)$$

h_i: softmax output

y_i: observed labeld

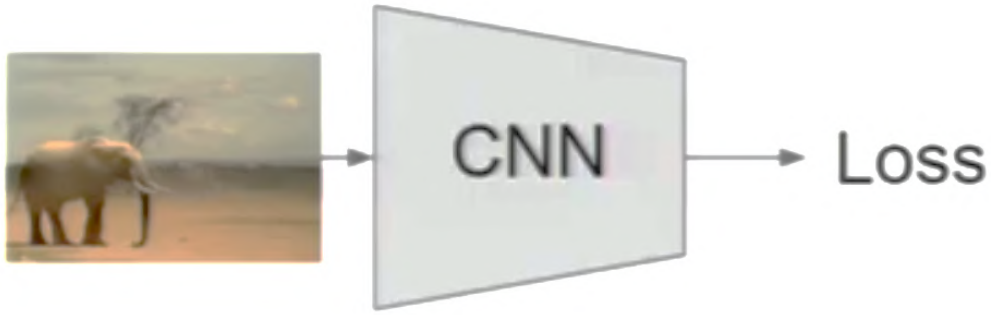
Background

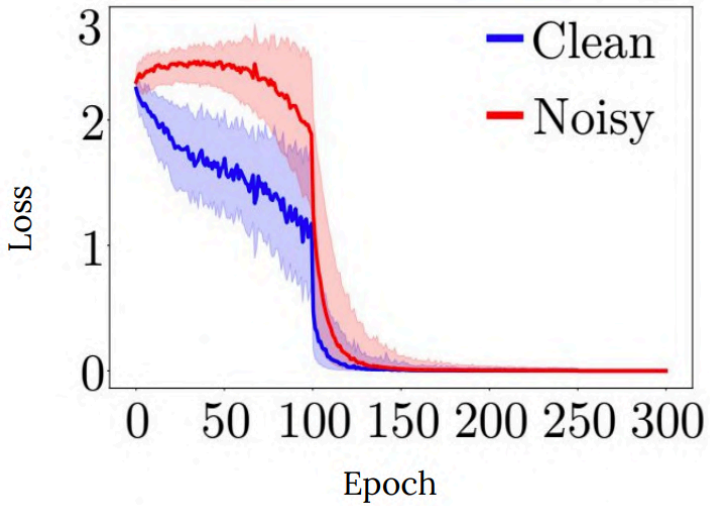
- Cross-Entropy Loss:

$$\mathcal{L} = -\sum_{i=1}^{N} y_i^T \log(h_i)$$

h_i: softmax output

y_i: observed labeld





Bootstrapping

[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 Dec 20.]

- Idea: dynamically update the targets of the prediction objective based on the current state of the model

- Cross entropy loss:

$$\mathcal{L} = -\sum_{i=1}^{N} y_i^T \log(h_i)$$

h_i: softmax output

y_i: observed labeld

Bootstrapping

[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 Dec 20.]

- Idea: dynamically update the targets of the prediction objective based on the current state of the model

- Cross entropy loss:

$$\mathcal{L} = -\sum_{i=1}^{N} y_i^T \log(h_i)$$

- Soft boostrapping:

$$\mathcal{L} = -\sum_{i=1}^{N} (\beta y_i + (1 - \beta h_i)^T \log(h_i)$$

h_i: softmax output

y_i: observed labeld

Bootstrapping

[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 Dec 20.]

- Idea: dynamically update the targets of the prediction objective based on the current state of the model

- Cross entropy loss:

$$\mathcal{L} = -\sum_{i=1}^{N} y_i^T \log(h_i)$$

h_i: softmax output

y_i: observed labeld

- Soft boostrapping:

$$\mathcal{L} = -\sum_{i=1}^{N} (\beta y_i + (1 - \beta h_i)^T \log(h_i)$$

- Hard boostrapping:

$$\mathcal{L} = -\sum_{i=1}^{N} (\beta y_i + (1 - \beta z_i)^T \log(h_i)$$

z is the argmax {0,1} model output

Boostrapping Effect

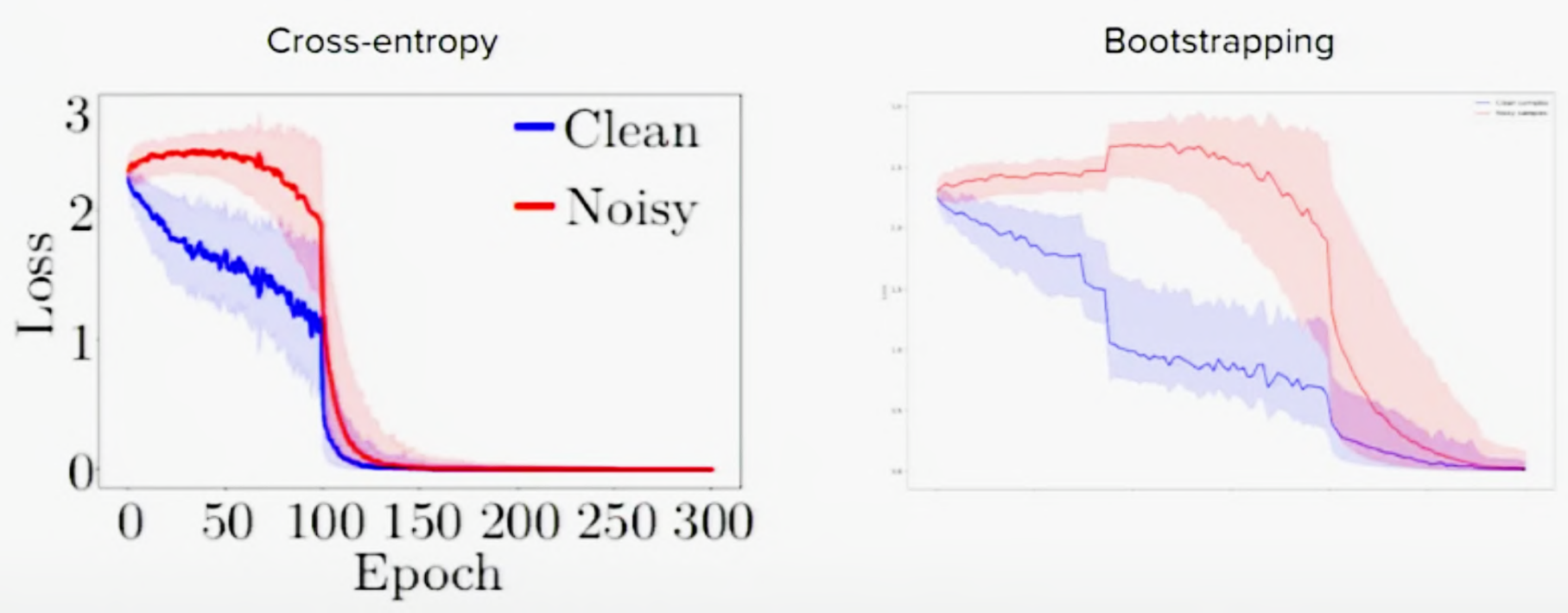


Figure from Kevin McGuinness slides

Forward/Backward approach

[Patrini G, Rozza A, Krishna Menon A, Nock R, Qu L. Making deep neural networks robust to label noise: A loss correction approach. InProceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017 (pp. 1944-1952).]

- Correct loss using estimated transition matrix based on noisy samples

- Forward
  - $\mathcal{L} = -\sum_{i=1}^{N} y_i^T \log(T \times h_i)$
- Backward
  - $\mathcal{L} = -\sum_{i=1}^{N} y_i^T T^{-1} \log(h_i)$

Forward/Backward approach

[Patrini G, Rozza A, Krishna Menon A, Nock R, Qu L. Making deep neural networks robust to label noise: A loss correction approach. InProceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017 (pp. 1944-1952).]

- Correct loss using estimated transition matrix based on noisy samples

- Forward
  - $\mathcal{L} = -\sum_{i=1}^{N} y_i^T \log(T \times h_i)$
- Backward
  - $\mathcal{L} = -\sum_{i=1}^{N} y_i^T T^{-1} \log(h_i)$

Algorithm 1 Robust two-stage training

Input: the noisy training set $\mathcal{S}$, any loss ℓ
If T is unknown:
  Train a network $\boldsymbol{h}(\boldsymbol{x})$ on $\mathcal{S}$ with loss ℓ
  Obtain an unlabeled sample X'
  Estimate $\hat{T}$
Train the network $\boldsymbol{h}(\boldsymbol{x})$ on $\mathcal{S}$ with loss $\ell^{\leftarrow}$ or $\ell^{\rightarrow}$
Output: $\boldsymbol{h}(\cdot)$

Noise Transition Matrix Estimation

- For each label, take the sample with highest classification
- estimate the label column for T

True transition matrix: $Tij = p(y^{ci}|y^{cj})$

Estimated transition matrix: $Tij = h(y^{ci}|x^{cj})$

c_i: class i

h: softmax probability

x^{cj}: sample of class c_j with the highest softmax score for class c_j

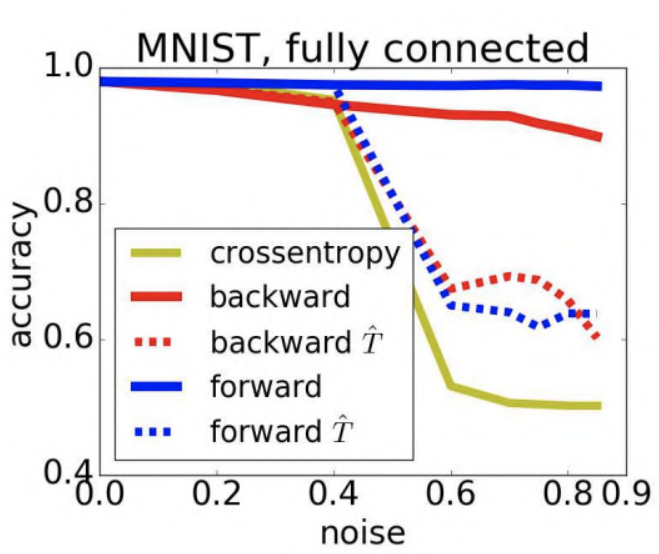
Forward/Backward approach

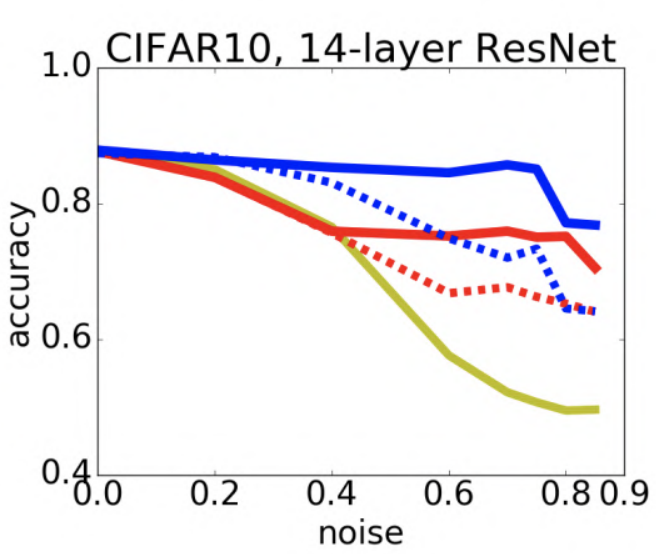
[Patrini G, Rozza A, Krishna Menon A, Nock R, Qu L. Making deep neural networks robust to label noise: A loss correction approach. InProceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017 (pp. 1944-1952).]

- Results

We artificially corrupt labels by a parametric matrix T. The rationale is to mimic some of the structure of real mistakes for similar classes, *e.g.* CAT $\rightarrow$ DOG. Transitions are parameterized by $N \in [0, 1]$ such that ground truth and wrong class have probability respectively of $1 - N, N$. An example of T used for MNIST with $N = 0.7$ is on the left:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & .3 & 0 & 0 & 0 & 0 & .7 & 0 & 0 \\ 0 & 0 & 0 & .3 & 0 & 0 & 0 & 0 & .7 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & .3 & .7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & .7 & .3 & 0 & 0 & 0 \\ 0 & .7 & 0 & 0 & 0 & 0 & 0 & .3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & 1 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & .33 & \epsilon & \epsilon & \epsilon & \epsilon & .67 & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & .35 & \epsilon & \epsilon & \epsilon & \epsilon & .65 & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & 1 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & .29 & .71 & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & .73 & .26 & \epsilon & \epsilon & \epsilon \\ \epsilon & .75 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & .25 & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & 1 & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & 1 \end{bmatrix} \quad (17)$$





Forward/Backward approach

[Patrini G, Rozza A, Krishna Menon A, Nock R, Qu L. Making deep neural networks robust to label noise: A loss correction approach. InProceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017 (pp. 1944-1952).]

- Learn hidden p(Y|X) using noisy samples
- p(Ÿ|X) = sum_Y p(Ÿ|Y)p(Y|X) or p(Ÿ|X) =T x p(Y|X)
  - where T is the label transition matrix
- Forward
  - Estimate T
  - Cross entropy of noisy label Ÿ and p(Ÿ|X) to estimate p(Y|X)
- Backward
  - Estimate T
  - Cross entropy of transformed noisy label T^{-1}Ÿ and p(Y|X)
- Estimate of T
  - For each label, take the sample with highest classification
  - estimate the label row for T

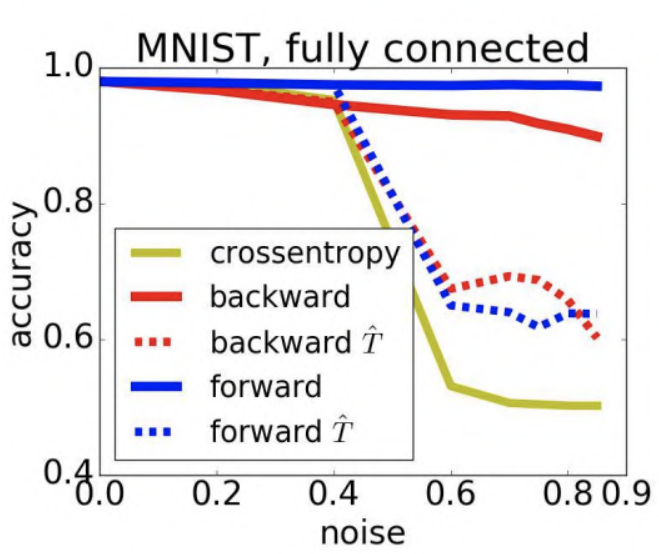
Forward/Backward approach

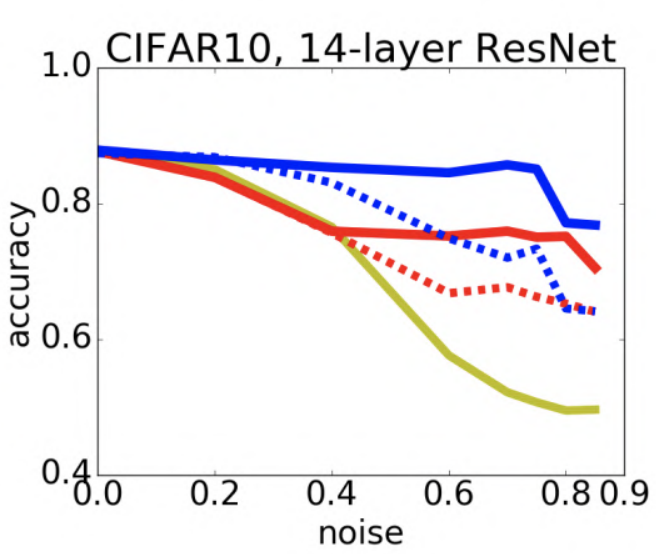
[Patrini G, Rozza A, Krishna Menon A, Nock R, Qu L. Making deep neural networks robust to label noise: A loss correction approach. InProceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017 (pp. 1944-1952).]

- Results

We artificially corrupt labels by a parametric matrix T. The rationale is to mimic some of the structure of real mistakes for similar classes, *e.g.* CAT $\rightarrow$ DOG. Transitions are parameterized by $N \in [0, 1]$ such that ground truth and wrong class have probability respectively of $1 - N, N$. An example of T used for MNIST with $N = 0.7$ is on the left:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & .3 & 0 & 0 & 0 & 0 & .7 & 0 & 0 \\ 0 & 0 & 0 & .3 & 0 & 0 & 0 & 0 & .7 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & .3 & .7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & .7 & .3 & 0 & 0 & 0 \\ 0 & .7 & 0 & 0 & 0 & 0 & 0 & .3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & 1 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & .33 & \epsilon & \epsilon & \epsilon & \epsilon & .67 & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & .35 & \epsilon & \epsilon & \epsilon & \epsilon & .65 & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & 1 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & .29 & .71 & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & .73 & .26 & \epsilon & \epsilon & \epsilon \\ \epsilon & .75 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & .25 & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & 1 & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & 1 \end{bmatrix} \quad (17)$$





Robust loss functions

[Ghosh A, Kumar H, Sastry PS. Robust loss functions under label noise for deep neural networks. InThirty-First AAAI Conference on Artificial Intelligence 2017 Feb 13.]

- Theoretical work that analyses the robustness of three loss functions to symmetric noise
  - Categorical cross entropy (CCE)
  - Mean square error (MSE)
  - Mean absolute error (MAE)

Robust loss functions

[Ghosh A, Kumar H, Sastry PS. Robust loss functions under label noise for deep neural networks. InThirty-First AAAI Conference on Artificial Intelligence 2017 Feb 13.]

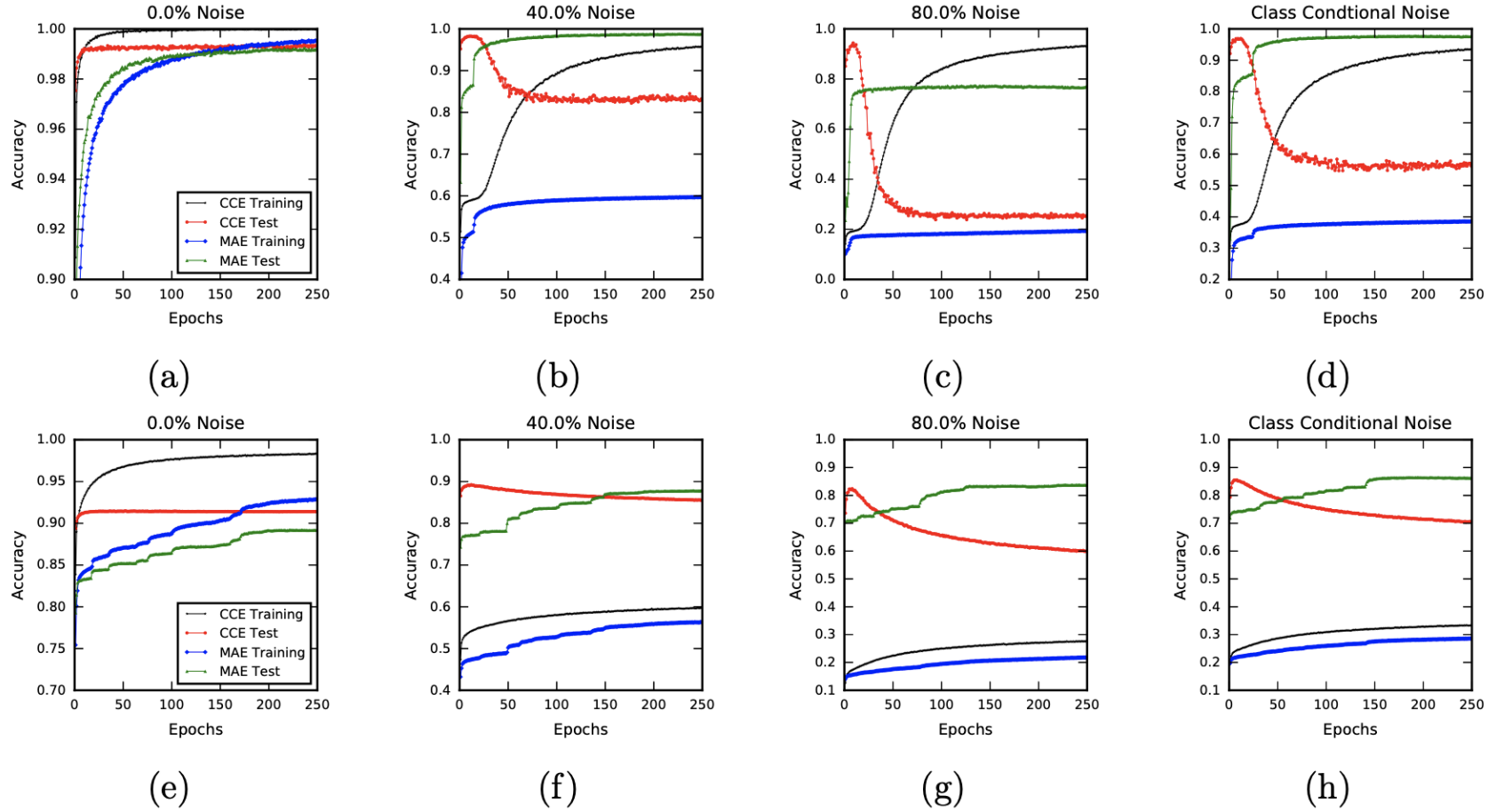


Figure 1: Train-Test Accuracies for log loss and MAE over epochs, for MNIST Datasets under (a) 0% noise (b) 40% noise (c) 80% noise (d) CC noise; and RCV1 Datasets under (e) 0% noise (f) 40% noise (g) 80% noise (h) CC noise. Legends shown in (a) and (e).

Robust loss functions

[Ghosh A, Kumar H, Sastry PS. Robust loss functions under label noise for deep neural networks. InThirty-First AAAI Conference on Artificial Intelligence 2017 Feb 13.]

Table 2: Accuracies under different noise rates (η) for all datasets (for Imdb, η's are halved). The last column gives accuracies under class conditional noise. In all the cases, standard deviation is shown only when it is more than 0.01

Data	loss	$\eta = 0\%$	$\eta = 30\%$	$\eta = 60\%$	CC
MNIST	CCE	0.9936	0.9138	0.5888	0.5775($\pm$0.0291)
	MAE	0.9916	0.9886	0.9799	0.9713
	MSE	0.9921	0.9868	0.9766	0.8505($\pm$0.0473)
RCV1	CCE	0.9126	0.8738	0.7905	0.7418($\pm$0.025)
	MAE	0.8732($\pm$0.0107)	0.8688	0.8637($\pm$0.0201)	0.8587
	MSE	0.9014	0.8943	0.8682($\pm$0.0120)	0.8315
Cifar 10	CCE	0.7812	0.5598($\pm$0.0170)	0.3083	0.4896
	MAE	0.7810($\pm$0.0190)	0.7011($\pm$0.0264)	0.5328($\pm$0.0251)	0.61425($\pm$0.0320)
	MSE	0.8074	0.7027	0.5257($\pm$0.0146)	0.6249($\pm$0.0359)
Imdb	CCE	0.8808	0.7729	0.6466	0.7858($\pm$0.0135)
	MAE	0.8813	0.8500	0.7352($\pm$0.0145)	0.8382($\pm$0.0127)
	MSE	0.8816	0.7725($\pm$0.0105)	0.6506($\pm$0.0103)	0.7874
News wire	CCE	0.7842	0.6905	0.4670	0.4973($\pm$0.0148)
	MAE	0.8081	0.7553	0.6357($\pm$0.0106)	0.6535
	MSE	0.7916	0.6626	0.4078($\pm$0.0172)	0.4377($\pm$0.0140)
News group	CCE	0.8006	0.7571	0.6435	0.5629
	MAE	0.7890	0.7749	0.7319	0.6772
	MSE	0.7999	0.7553	0.6347	0.5519

0-1 loss and domain adaptation

Curriculum Loss: Robust Learning and Generalization against Label Corruption. Lyu and Tsang. ICLR'20.

- There is a monotonic relationship between the (empirical) risk and the (empirical) adversarial risk when the 0-1 loss function is used.
  - Minimizing the empirical risk with the 0-1 loss function is equivalent to minimize the empirical adversarial risk (worst-case risk).
  - Propose a tight 0-1 loss upper-bound, called curriculum loss.

Table 1: Test accuracy(%) of DenseNet on CIFAR10 and CIFAR100.

Noise type	CIFAR10				CIFAR100			
	NPCL	CL	Softmax	RoG	NPCL	CL	Softmax	RoG
uniform (20%)	**89.49**	89.32	81.01	87.41	64.88	**67.92**	61.72	64.29
uniform (40%)	83.24	**85.57**	72.34	81.83	56.34	**58.63**	50.89	55.68
uniform (60%)	66.2	68.52	55.42	**75.45**	44.49	**46.65**	38.33	44.12

Table 2: Test accuracy(%) of DenseNet on CIFAR10 and CIFAR100 with semantic noise.

Dataset	Label generator (noise rate)	NPCL	CL	Cross-entropy	D2L
CIFAR10	DenseNet(32%)	66.5	**67.45**	67.24	66.91
	ResNet(38%)	61.88	**62.88**	62.26	59.10
	VGG(34%)	68.37	**69.61**	68.77	57.97
CIFAR100	DenseNet(34%)	**57.59**	55.14	50.72	5.00
	ResNet(37%)	**54.49**	53.20	50.68	23.71
	VGG(37%)	**55.41**	52.71	51.08	40.97

Table 3: Test accuracy(%) of DenseNet on CIFAR10 with open-set noise.

Open-set Data	NPCL	Softmax	RoG
CIFAR100	82.85	79.01	**83.37**
ImageNet	**87.95**	86.88	87.05
CIFAR100-ImageNet	84.28	81.58	**84.35**

Passive and active normalized loss

- Normalized Loss Functions for Deep Learning with Noisy Labels. Ma et al. ICML'20.

$$\mathcal{L}_{\text{norm}} = \frac{\mathcal{L}(f(\boldsymbol{x}), y)}{\sum_{j=1}^{K} \mathcal{L}(f(\boldsymbol{x}), j)}. \quad (1)$$

A normalized loss has the property: $\mathcal{L}_{\text{norm}} \in [0, 1]$

- Normalized loss are robust

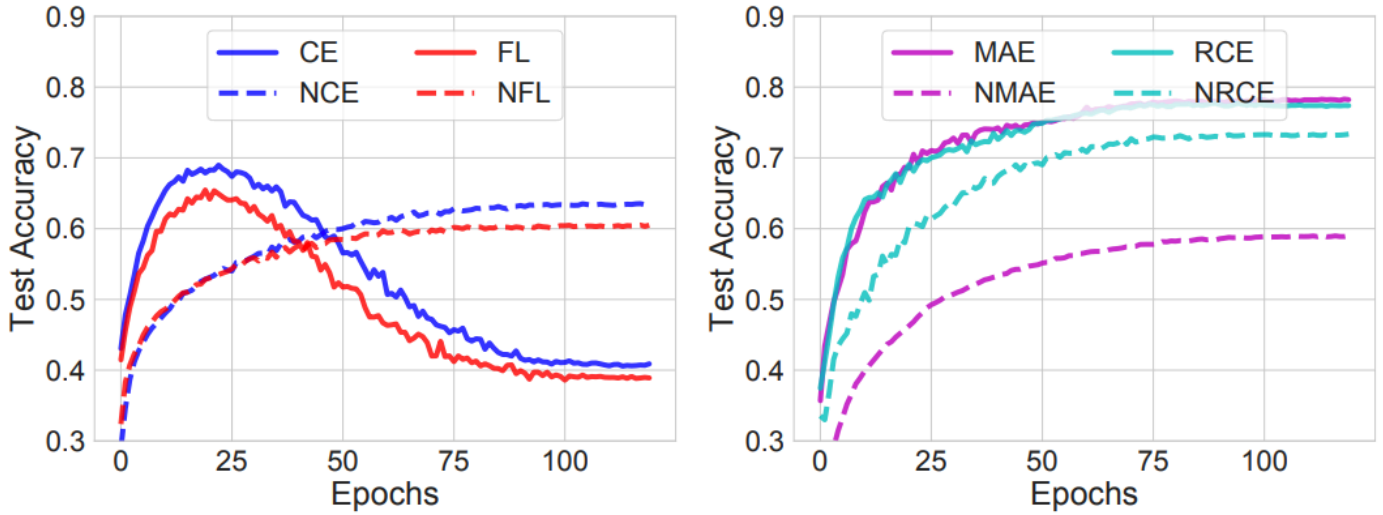


Figure 2. Test accuracies of unnormalized versus normalized loss functions on CIFAR-10 under 0.6 symmetric noise.

Passive and active normalized loss

- Normalized Loss Functions for Deep Learning with Noisy Labels. Ma et al. ICML'20.

$$\mathcal{L}_{\text{norm}} = \frac{\mathcal{L}(f(\boldsymbol{x}), y)}{\sum_{j=1}^{K} \mathcal{L}(f(\boldsymbol{x}), j)}. \tag{1}$$

A normalized loss has the property: $\mathcal{L}_{\text{norm}} \in [0, 1]$.

Definition 1. *(Active loss function)* $\mathcal{L}_{Active}$ *is an active loss function if* $\forall(\boldsymbol{x}, y) \in \mathcal{D}\ \forall k \neq y\ \ell(f(\boldsymbol{x}), k) = 0$.

Definition 2. *(Passive loss function)* $\mathcal{L}_{Passive}$ *is a passive loss function if* $\forall(\boldsymbol{x}, y) \in \mathcal{D}\ \exists k \neq y\ \ell(f(\boldsymbol{x}), k) \neq 0$.

Table 1. Examples of active and passive loss functions.

Loss Type	Active	Passive
Examples	CE, NCE, FL, NFL	MAE, NMAE, RCE, NRCE

$$\mathcal{L}_{\text{APL}} = \alpha \cdot \mathcal{L}_{\text{Active}} + \beta \cdot \mathcal{L}_{\text{Passive}}, \tag{6}$$

where, $\alpha, \beta > 0$ are parameters to balance the two terms.

Passive and active normalized loss

Table 2. Test accuracies (%) of different methods on benchmark datasets with clean or symmetric label noise ($\eta \in [0.2, 0.8]$). The res (mean±std) are reported over 3 random runs and the top 2 best results are **boldfaced**.

Datasets	Methods	Clean (η=0.0)	Symmetric Noise Rate (η) 0.2	0.4	0.6	0.8
MNIST	CE	99.25 ± 0.08	97.42 ± 0.06	94.21 ± 0.54	86.00 ± 1.48	47.08 ± 1.15
	FL	99.30 ± 0.02	97.45 ± 0.19	94.71 ± 0.25	85.76 ± 1.85	49.77 ± 2.26
	GCE	99.27 ± 0.01	99.18 ± 0.06	98.72 ± 0.05	97.43 ± 0.23	12.77 ± 2.00
	NLNL	99.27 ± 0.02	97.49 ± 0.30	96.64 ± 0.52	97.22 ± 0.06	10.32 ± 0.73
	SCE	99.24 ± 0.08	99.15 ± 0.04	98.78 ± 0.09	97.45 ± 0.29	73.70 ± 0.84
	NFL+MAE	99.39 ± 0.04	99.12 ± 0.06	98.74 ± 0.14	96.91 ± 0.09	$\mathbf{74.98 \pm 1.99}$
	NFL+RCE	99.38 ± 0.02	$\mathbf{99.19 \pm 0.06}$	$\mathbf{98.79 \pm 0.10}$	$\mathbf{97.46 \pm 0.03}$	74.59 ± 2.23
	NCE+MAE	99.37 ± 0.02	99.14 ± 0.05	98.78 ± 0.00	96.76 ± 0.34	74.66 ± 1.11
	NCE+RCE	99.37 ± 0.02	$\mathbf{99.20 \pm 0.04}$	$\mathbf{98.79 \pm 0.12}$	$\mathbf{97.48 \pm 0.13}$	$\mathbf{75.18 \pm 1.19}$
CIFAR-10	CE	90.36 ± 0.03	75.90 ± 0.28	60.28 ± 0.27	40.90 ± 0.35	19.65 ± 0.46
	FL	89.63 ± 0.25	74.59 ± 0.49	57.55 ± 0.39	38.91 ± 0.62	19.43 ± 0.27
	GCE	89.38 ± 0.23	87.27 ± 0.21	83.33 ± 0.39	72.00 ± 0.37	29.08 ± 0.80
	NLNL	91.93 ± 0.20	83.98 ± 0.18	76.58 ± 0.44	72.85 ± 0.39	51.41 ± 0.85
	SCE	91.30 ± 0.22	88.05 ± 0.26	82.06 ± 0.24	66.08 ± 0.25	30.69 ± 0.63
	NFL+MAE	89.25 ± 0.19	87.33 ± 0.14	83.81 ± 0.06	76.36 ± 0.31	45.23 ± 0.52
	NFL+RCE	90.91 ± 0.02	$\mathbf{89.14 \pm 0.13}$	$\mathbf{86.05 \pm 0.12}$	$\mathbf{79.78 \pm 0.13}$	$\mathbf{55.06 \pm 1.08}$
	NCE+MAE	88.83 ± 0.34	87.12 ± 0.21	84.19 ± 0.43	77.61 ± 0.05	49.62 ± 0.72
	NCE+RCE	90.76 ± 0.22	$\mathbf{89.22 \pm 0.27}$	$\mathbf{86.02 \pm 0.09}$	$\mathbf{79.78 \pm 0.50}$	$\mathbf{52.71 \pm 1.90}$
CIFAR-100	CE	70.89 ± 0.22	56.99 ± 0.41	41.40 ± 0.36	22.15 ± 0.40	7.58 ± 0.44
	FL	70.61 ± 0.44	56.10 ± 0.48	40.77 ± 0.62	22.14 ± 1.00	7.21 ± 0.25
	GCE	69.00 ± 0.56	65.24 ± 0.56	58.94 ± 0.50	45.18 ± 0.93	16.18 ± 0.46
	NLNL	68.72 ± 0.60	46.99 ± 0.91	30.29 ± 1.64	16.60 ± 0.90	11.01 ± 2.48
	SCE	70.38 ± 0.45	55.39 ± 0.18	39.99 ± 0.59	22.35 ± 0.65	7.57 ± 0.28
	NFL+MAE	67.98 ± 0.52	63.58 ± 0.09	58.18 ± 0.08	46.10 ± 0.50	24.78 ± 0.82
	NFL+RCE	68.23 ± 0.62	64.52 ± 0.35	58.20 ± 0.31	46.30 ± 0.45	25.16 ± 0.55
	NCE+MAE	68.75 ± 0.54	$\mathbf{65.25 \pm 0.62}$	$\mathbf{59.22 \pm 0.36}$	$\mathbf{48.06 \pm 0.34}$	$\mathbf{25.50 \pm 0.76}$
	NCE+RCE	69.02 ± 0.11	$\mathbf{65.31 \pm 0.07}$	$\mathbf{59.48 \pm 0.56}$	$\mathbf{47.12 \pm 0.62}$	$\mathbf{25.80 \pm 1.12}$

Co-teaching (1 of 4)

[Jiang L, Zhou Z, Leung T, Li LJ, Fei-Fei L. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. ICML 2018.]

- Student net (learn classifier based on weighted training samples)
- Mentor net (weight training samples)
  - Produces weights
  - Idea is to weight higher easier samples and include hard samples later in the training process

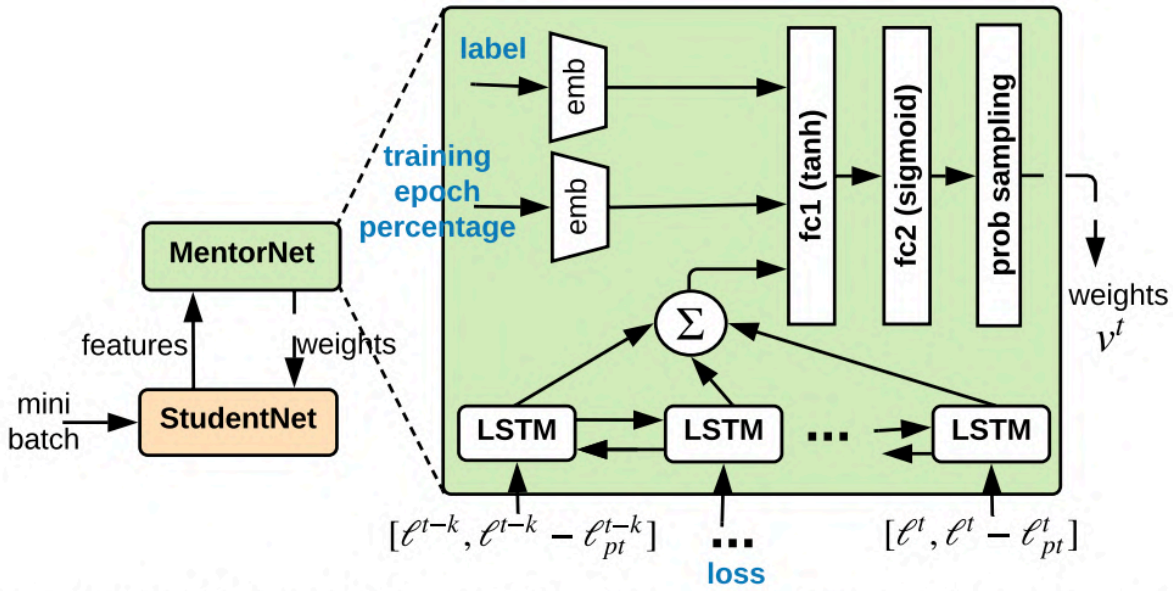


Figure 1. The MentorNet architecture used in experiments. *emb*, *fc* and *prob sampling* stand for the embedding, fully-connected and probabilistic sampling layer.

Co-teaching (1 of 4)

[Jiang L, Zhou Z, Leung T, Li LJ, Fei-Fei L. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. ICML 2018.]

Table 2. Comparison of validation accuracy on CIFAR-10 and CIFAR-100 under different noise fractions.

	Resnet-101 StudentNet						Inception StudentNet					
	CIFAR-100			CIFAR-10			CIFAR-100			CIFAR-10		
Method	0.2	0.4	0.8	0.2	0.4	0.8	0.2	0.4	0.8	0.2	0.4	0.8
FullModel	0.60	0.45	0.08	0.82	0.69	0.18	0.43	0.38	0.15	0.76	0.73	0.42
Forgetting	0.61	0.44	0.16	0.78	0.63	0.35	0.42	0.37	0.17	0.76	0.71	0.44
Self-paced	0.70	0.55	0.13	0.89	0.85	0.28	0.44	0.38	0.14	**0.80**	0.74	0.33
Focal Loss	0.59	0.44	0.09	0.79	0.65	0.28	0.43	0.38	0.15	0.77	0.74	0.40
Reed Soft	0.62	0.46	0.08	0.81	0.63	0.18	0.42	0.39	0.12	0.78	0.73	0.39
MentorNet PD	0.72	0.56	0.14	0.91	0.77	0.33	0.44	0.39	0.16	0.79	0.74	0.44
MentorNet DD	**0.73**	**0.68**	**0.35**	**0.92**	**0.89**	**0.49**	**0.46**	**0.41**	**0.20**	0.79	**0.76**	**0.46**

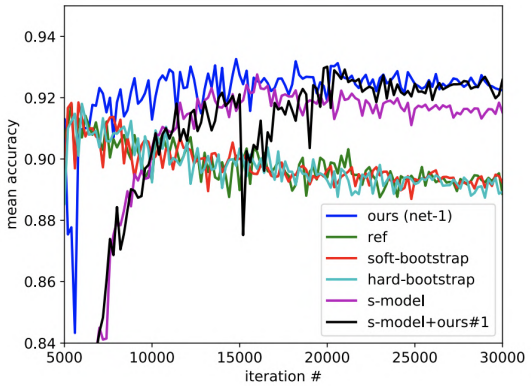
Co-teaching (2 of 4)

[Malach E, Shalev-Shwartz S. Decoupling" when to update" from" how to update". InAdvances in Neural Information Processing Systems 2017 (pp. 960-970).]

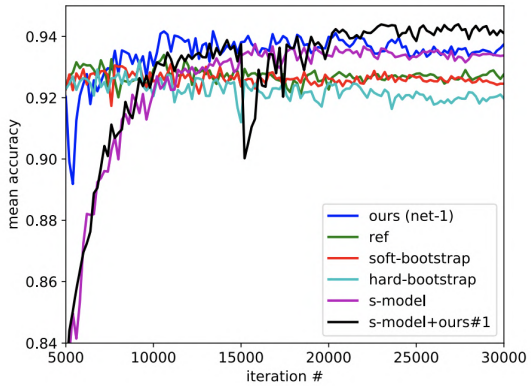
- The decision of “when to update” does not depend on the label. Instead, it depends on a disagreement between two different networks.
- Classification of gender in Labeled Faces in the Wild (LFW) dataset

Algorithm 1 Update by Disagreement

input:
an update rule U
batch size b
two initial predictors $h_1, h_2 \in \mathcal{H}$
for $t = 1, 2, \ldots, N$ **do**
draw mini-batch $(x_1, y_1), \ldots, (x_b, y_b) \sim \tilde{\mathcal{D}}^b$
let $S = \{(x_i, y_i) : h_1(x_i) \neq h_2(x_i)\}$
$h_1 \leftarrow U(h_1, S)$
$h_2 \leftarrow U(h_2, S)$
end for



Dataset #1 - more noise



Dataset #2 - less noise

Clean validation set

Co-teaching (3 of 4)

[Han B, Yao Q, Yu X, Niu G, Xu M, Hu W, Tsang I, Sugiyama M. Co-teaching: Robust training of deep neural networks with extremely noisy labels. InAdvances in neural information processing systems 2018 (pp. 8527-8537).]

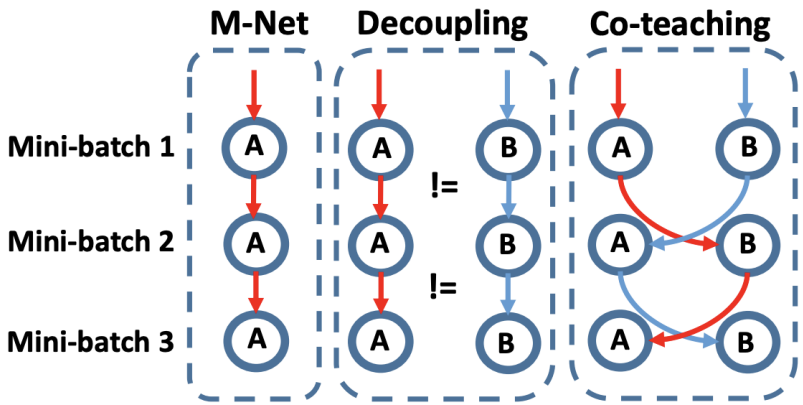


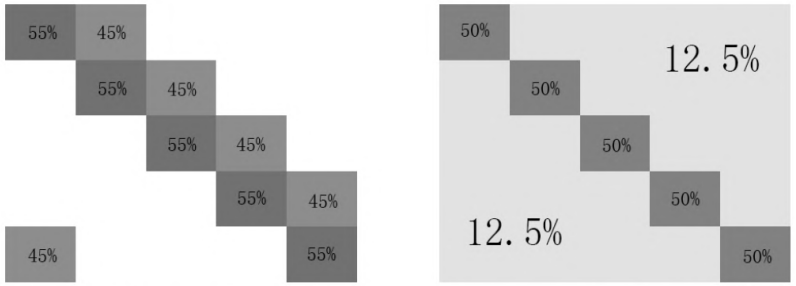
Figure 1: Comparison of error flow among MentorNet (M-Net) [17], Decoupling [26] and Co-teaching. Assume that the error flow comes from the biased selection of training instances, and error flow from network A or B is denoted by red arrows or blue arrows, respectively. **Left panel:** M-Net maintains only one network (A). **Middle panel:** Decoupling maintains two networks (A & B). The parameters of two networks are updated, when the predictions of them disagree (!=). **Right panel:** Co-teaching maintains two networks (A & B) simultaneously. In each mini-batch data, each network samples its small-loss instances as the useful knowledge, and teaches such useful instances to its peer network for the further training. Thus, the error flow in Co-teaching displays the zigzag shape.

Algorithm 1 Co-teaching Algorithm.

1: **Input** w_f and w_g, learning rate η, fixed τ, epoch T_k and $T_{\max}$, iteration $N_{\max}$;
for $T = 1, 2, \ldots, T_{\max}$ **do**
  2: **Shuffle** training set $\mathcal{D}$; //noisy dataset
  for $N = 1, \ldots, N_{\max}$ **do**
    3: **Fetch** mini-batch $\bar{\mathcal{D}}$ from $\mathcal{D}$;
    4: **Obtain** $\bar{\mathcal{D}}_f = \arg\min_{\mathcal{D}':|\mathcal{D}'| \geq R(T)|\bar{\mathcal{D}}|} \ell(f, \mathcal{D}')$; //sample $R(T)\%$ small-loss instances
    5: **Obtain** $\bar{\mathcal{D}}_g = \arg\min_{\mathcal{D}':|\mathcal{D}'| \geq R(T)|\bar{\mathcal{D}}|} \ell(g, \mathcal{D}')$; //sample $R(T)\%$ small-loss instances
    6: **Update** $w_f = w_f - \eta \nabla \ell(f, \bar{\mathcal{D}}_g)$; //update w_f by $\bar{\mathcal{D}}_g$;
    7: **Update** $w_g = w_g - \eta \nabla \ell(g, \bar{\mathcal{D}}_f)$; //update w_g by $\bar{\mathcal{D}}_f$;
  end
  8: **Update** $R(T) = 1 - \min\left\{\frac{T}{T_k}\tau, \tau\right\}$;
end
9: **Output** w_f **and** w_g.

Co-teaching (3 of 4)

[Han B, Yao Q, Yu X, Niu G, Xu M, Hu W, Tsang I, Sugiyama M. Co-teaching: Robust training of deep neural networks with extremely noisy labels. InAdvances in neural information processing systems 2018 (pp. 8527-8537).]



(a) Pair ($\epsilon = 45\%$). (b) Symmetry ($\epsilon = 50\%$).

Table 4: Average test accuracy on *MNIST* over the last ten epochs.

Flipping-Rate	Standard	Bootstrap	S-model	F-correction	Decoupling	MentorNet	Co-teaching
Pair-45%	56.52% ±0.55%	57.23% ±0.73%	56.88% ±0.32%	0.24% ±0.03%	58.03% ±0.07%	80.88% ±4.45%	**87.63%** ±0.21%
Symmetry-50%	66.05% ±0.61%	67.55% ±0.53%	62.29% ±0.46%	79.61% ±1.96%	81.15% ±0.03%	90.05% ±0.30%	**91.32%** ±0.06%
Symmetry-20%	94.05% ±0.16%	94.40% ±0.26%	98.31% ±0.11%	**98.80%** ±0.12%	95.70% ±0.02%	96.70% ±0.22%	97.25% ±0.03%

Table 5: Average test accuracy on *CIFAR-10* over the last ten epochs.

Flipping,Rate	Standard	Bootstrap	S-model	F-correction	Decoupling	MentorNet	Co-teaching
Pair-45%	49.50% ±0.42%	50.05% ±0.30%	48.21% ±0.55%	6.61% ±1.12%	48.80% ±0.04%	58.14% ±0.38%	**72.62%** ±0.15%
Symmetry-50%	48.87% ±0.52%	50.66% ±0.56%	46.15% ±0.76%	59.83% ±0.17%	51.49% ±0.08%	71.10% ±0.48%	**74.02%** ±0.04%
Symmetry-20%	76.25% ±0.28%	77.01% ±0.29%	76.84% ±0.66%	**84.55%** ±0.16%	80.44% ±0.05%	80.76% ±0.36%	82.32% ±0.07%

Co-teaching (4 of 4)

[Yu X, Han B, Yao J, Niu G, Tsang IW, Sugiyama M. How does disagreement help generalization against label corruption?. ICML 2019]

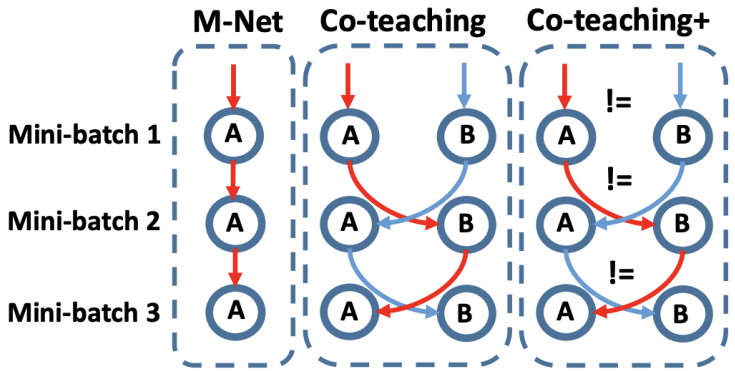


Figure 2. Comparison of error flow among MentorNet (M-Net), Co-teaching and Co-teaching+. Assume that the error flow comes from the selection of training instances, and the error flow from network A or B is denoted by red arrows or blue arrows, respectively. **Left panel:** M-Net maintains only one network (A). **Middle panel:** Co-teaching maintains two networks (A & B) simultaneously. In each mini-batch data, each network selects its small-loss data to teach its peer network for the further training. **Right panel:** Co-teaching+ also maintains two networks (A & B). However, two networks feed forward and predict each mini-batch data first, and keep prediction disagreement data (!=) only. Based on such disagreement data, each network selects its small-loss data to teach its peer network for the further training.

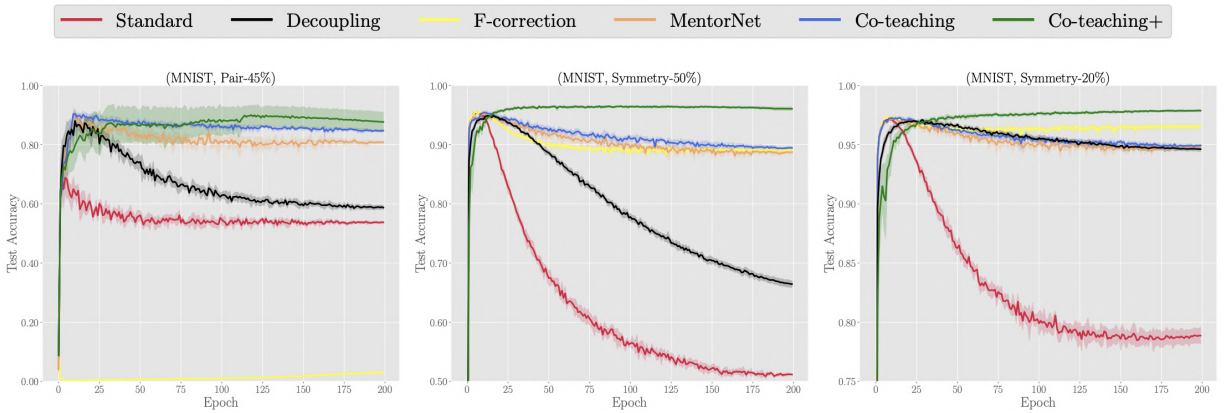
Algorithm 1 Co-teaching+. Step 4: disagreement-update; Step 5-8: cross-update.

1: **Input** $w^{(1)}$ and $w^{(2)}$, training set $\mathcal{D}$, batch size B, learning rate η, estimated noise rate τ, epoch E_k and $E_{\max}$;
for $e = 1, 2, \ldots, E_{\max}$ **do**
  2: **Shuffle** $\mathcal{D}$ into $\frac{|\mathcal{D}|}{B}$ mini-batches; //noisy dataset
  for $n = 1, \ldots, \frac{|\mathcal{D}|}{B}$ **do**
    3: **Fetch** n-th mini-batch $\bar{\mathcal{D}}$ from $\mathcal{D}$;
    4: **Select** prediction disagreement $\bar{\mathcal{D}}'$ by Eq. (1);
    5: **Get** $\bar{\mathcal{D}}'^{(1)} = \arg\min_{\mathcal{D}':|\mathcal{D}'|\geq\lambda(e)|\bar{\mathcal{D}}'|} \ell(\mathcal{D}'; w^{(1)})$; //sample $\lambda(e)\%$ small-loss instances
    6: **Get** $\bar{\mathcal{D}}'^{(2)} = \arg\min_{\mathcal{D}':|\mathcal{D}'|\geq\lambda(e)|\bar{\mathcal{D}}'|} \ell(\mathcal{D}'; w^{(2)})$; //sample $\lambda(e)\%$ small-loss instances
    7: **Update** $w^{(1)} = w^{(1)} - \eta\nabla\ell(\bar{\mathcal{D}}'^{(2)}; w^{(1)})$; //update $w^{(1)}$ by $\bar{\mathcal{D}}'^{(2)}$;
    8: **Update** $w^{(2)} = w^{(2)} - \eta\nabla\ell(\bar{\mathcal{D}}'^{(1)}; w^{(2)})$; //update $w^{(2)}$ by $\bar{\mathcal{D}}'^{(1)}$;
  end
  9: **Update** $\lambda(e) = 1 - \min\{\frac{e}{E_k}\tau, \tau\}$ or $1 - \min\{\frac{e}{E_k}\tau, (1 + \frac{e-E_k}{E_{\max}-E_k})\tau\}$;
end
10: **Output** $w^{(1)}$ **and** $w^{(2)}$.

$$\bar{\mathcal{D}}' = \{(x_i, y_i) : \bar{y}_i^{(1)} \neq \bar{y}_i^{(2)}\}, \quad (1)$$

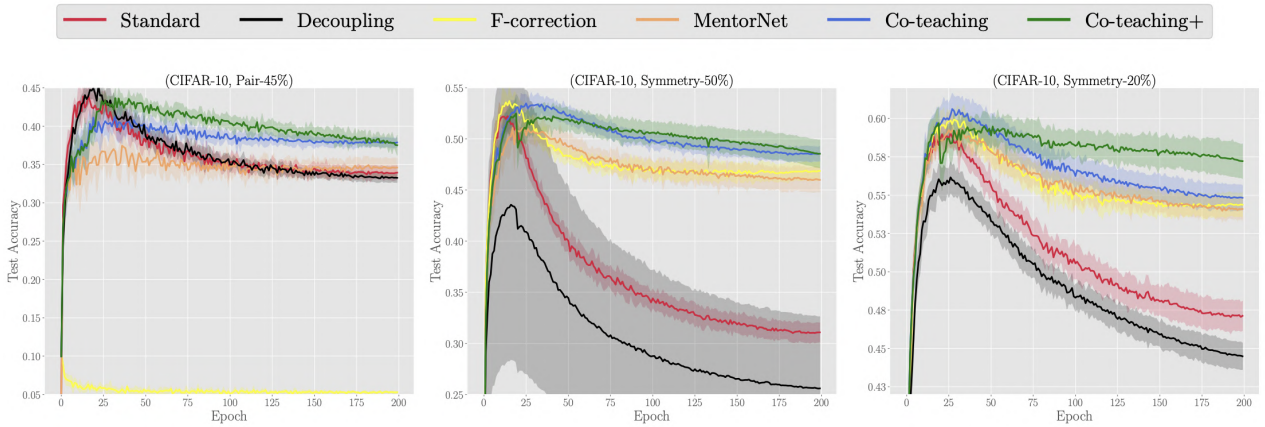
Co-teaching (4 of 4)

[Yu X, Han B, Yao J, Niu G, Tsang IW, Sugiyama M. How does disagreement help generalization against label corruption?. ICML 2019]



(a) Pair-45%. (b) Symmetry-50%. (c) Symmetry-20%.

Figure 3. Test accuracy vs. number of epochs on *MNIST* dataset.



(a) Pair-45%. (b) Symmetry-50%. (c) Symmetry-20%.

Figure 4. Test accuracy vs. number of epochs on *CIFAR-10* dataset.

Contrastive loss + CE loss

- Combating Noisy Labels by Agreement: A Joint Training Method with Co-Regularization. Wei et al. CVPR'20

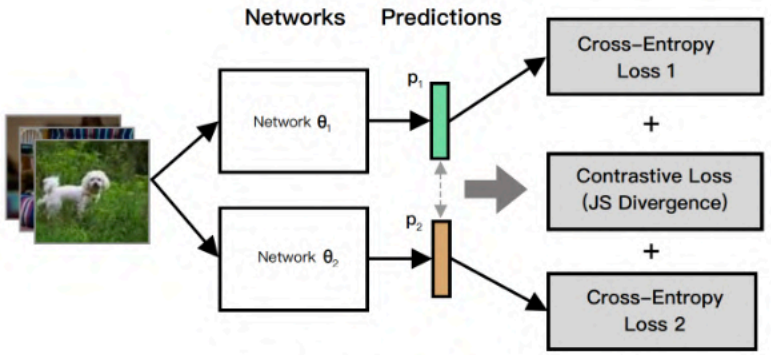


Figure 2. JoCoR schematic.

$$\begin{aligned}\ell_{\text{sup}}(\boldsymbol{x}_i, y_i) &= \ell_{\text{C1}}(\boldsymbol{x}_i, y_i) + \ell_{\text{C2}}(\boldsymbol{x}_i, y_i) \\ &= -\sum_{i=1}^{N}\sum_{m=1}^{M} y_i \log(p_1^m(\boldsymbol{x}_i)) \\ &\quad -\sum_{i=1}^{N}\sum_{m=1}^{M} y_i \log(p_2^m(\boldsymbol{x}_i))\end{aligned} \tag{2}$$

7-layer CNN

Table 3. Average test accuracy (%) on *CIFAR-10* over the last 10 epochs.

Flipping-Rate	Standard	F-correction	Decoupling	Co-teaching	Co-teaching+	JoCoR
Symmetry-20%	69.18 ± 0.52	68.74 ± 0.20	69.32 ± 0.40	78.23 ± 0.27	78.71 ± 0.34	$\mathbf{85.73} \pm 0.19$
Symmetry-50%	42.71 ± 0.42	42.19 ± 0.60	40.22 ± 0.30	71.30 ± 0.13	57.05 ± 0.54	$\mathbf{79.41} \pm 0.25$
Symmetry-80%	16.24 ± 0.39	15.88 ± 0.42	15.31 ± 0.43	26.58 ± 2.22	24.19 ± 2.74	$\mathbf{27.78} \pm 3.06$
Asymmetry-40%	69.43 ± 0.33	70.60 ± 0.40	68.72 ± 0.30	73.78 ± 0.22	68.84 ± 0.20	$\mathbf{76.36} \pm 0.49$

Table 5. Classification accuracy (%) on the *Clothing1M* test set

ResNet18

Methods	*best*	*last*
Standard	67.22	64.68
F-correction	68.93	65.36
Decoupling	68.48	67.32
Co-teaching	69.21	68.51
Co-teaching+	59.32	58.79
JoCoR	**70.30**	**69.79**

Negative Learning

[Kim Y, Yim J, Yun J, Kim J. Nlnl: Negative learning for noisy labels. InProceedings of the IEEE International Conference on Computer Vision 2019 (pp. 101-110).]

- Learn what is not instead of what it is

Algorithm 1 Complementary label generation

Input: Training label $y \in \mathcal{Y}$

while iteration **do**

$\overline{y}$ = Randomly select from $\{1, ..., C\} \setminus \{y\}$

Output: Complementary label $\overline{y}$

$$\mathcal{L}(f, y) = -\sum_{k=1}^{c} \boldsymbol{y}_k \log \boldsymbol{p}_k \qquad (1)$$

$$\mathcal{L}(f, \overline{y}) = -\sum_{k=1}^{c} \overline{\boldsymbol{y}}_k \log(1 - \boldsymbol{p}_k) \qquad (2)$$

Algorithm 2 Overall process of SelNLPL

Input: Training data $(\boldsymbol{x}, y) \in (\mathcal{X}, \mathcal{Y})$, network $f(\boldsymbol{x}; \boldsymbol{\theta})$, total epoch T

for $i \leftarrow 1$ to T **do** ▷ NL

Batch $\leftarrow$ Sample $\boldsymbol{x}$

Update f by minimizing Eq. 2

for $i \leftarrow 1$ to T **do** ▷ SelNL

Batch $\leftarrow$ Sample $\boldsymbol{x}$ if $\boldsymbol{p}_y > 1/\mathrm{c}$

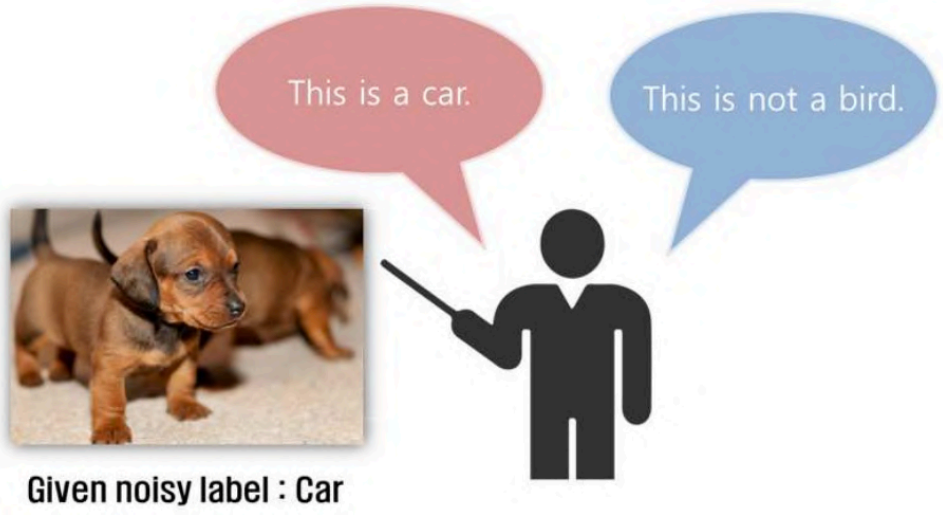
Update f by minimizing Eq. 2

for $i \leftarrow 1$ to T **do** ▷ SelPL

Batch $\leftarrow$ Sample $\boldsymbol{x}$ if $\boldsymbol{p}_y > \gamma$

Update f by minimizing Eq. 1

Output: Network $f(\boldsymbol{x}; \boldsymbol{\theta})$



Negative Learning

[Kim Y, Yim J, Yun J, Kim J. Nlnl: Negative learning for noisy labels. InProceedings of the IEEE International Conference on Computer Vision 2019 (pp. 101-110).]

- Learn what is not instead of what it is

Algorithm 1 Complementary label generation

Input: Training label $y \in \mathcal{Y}$

while iteration **do**

$\overline{y}$ = Randomly select from $\{1, ..., C\} \setminus \{y\}$

Output: Complementary label $\overline{y}$

$$\mathcal{L}(f, y) = -\sum_{k=1}^{c} \boldsymbol{y}_k \log \boldsymbol{p}_k \qquad (1)$$

$$\mathcal{L}(f, \overline{y}) = -\sum_{k=1}^{c} \overline{\boldsymbol{y}}_k \log(1 - \boldsymbol{p}_k) \qquad (2)$$

Algorithm 2 Overall process of SelNLPL

Input: Training data $(\boldsymbol{x}, y) \in (\mathcal{X}, \mathcal{Y})$, network $f(\boldsymbol{x}; \boldsymbol{\theta})$, total epoch T

for $i \leftarrow 1$ to T **do** ▷ NL

Batch $\leftarrow$ Sample $\boldsymbol{x}$

Update f by minimizing Eq. 2

for $i \leftarrow 1$ to T **do** ▷ SelNL

Batch $\leftarrow$ Sample $\boldsymbol{x}$ if $\boldsymbol{p}_y > 1/\mathrm{c}$

Update f by minimizing Eq. 2

for $i \leftarrow 1$ to T **do** ▷ SelPL

Batch $\leftarrow$ Sample $\boldsymbol{x}$ if $\boldsymbol{p}_y > \gamma$

Update f by minimizing Eq. 1

Output: Network $f(\boldsymbol{x}; \boldsymbol{\theta})$

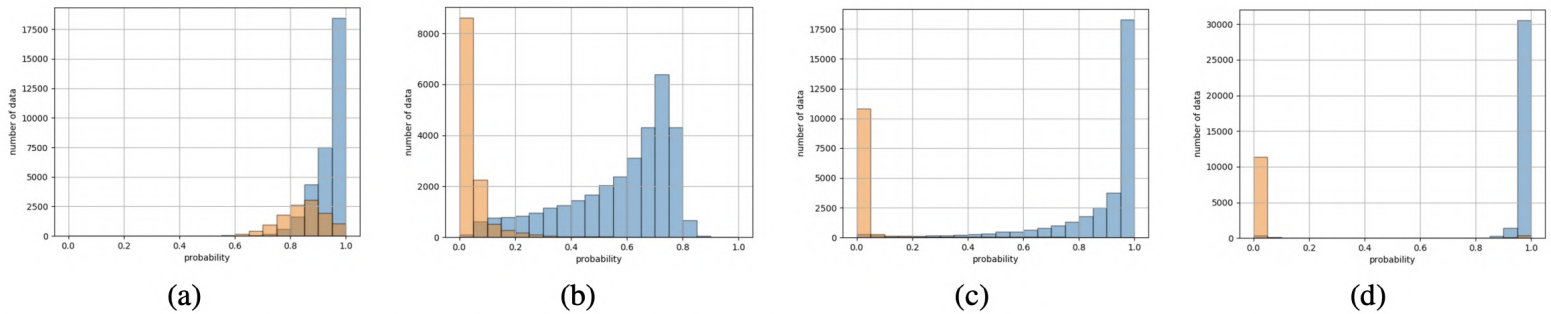


Figure 3: Histogram showing the distribution of CIFAR10 training data with 30% *symm-inc* noise, according to probability $\boldsymbol{p}_y$ (confidence). Blue indicates clean data, whereas orange indicates noisy data. (a): PL. (b): NL. (c): NL→SelNL. (d): NL→SelNL→SelPL (SelNLPL).

Negative Learning

[Kim Y, Yim J, Yun J, Kim J. Nlnl: Negative learning for noisy labels. InProceedings of the IEEE International Conference on Computer Vision 2019 (pp. 101-110).]

Datasets	Model	Methods	*Symm*				*Asymm*			
			20	40	60	80	10	20	30	40
FashionMNIST	ResNet18	CE	93.24	92.09	90.29	86.20	94.06	93.72	92.72	89.82
		MAE [3]	80.39	79.30	82.41	74.73	74.03	63.03	58.14	56.04
		Forward T [25]	93.64	92.69	91.16	87.59	94.33	94.03	93.91	93.65
		Forward $\hat{T}$ [25]	93.26	92.24	90.54	85.57	94.09	93.66	93.52	88.53
		L_q [39]	93.35	92.58	91.30	88.01	93.51	93.24	92.21	89.53
		Truncated L_q [39]	93.21	92.60	91.56	88.33	93.53	93.36	92.76	91.62
		Ours	**94.82**	**94.16**	**92.78**	-	**95.10**	**94.88**	**94.66**	**93.96**
CIFAR10	ResNet34	CE	86.98	81.88	74.14	53.82	90.69	88.59	86.14	80.11
		MAE [3]	83.72	67.00	64.21	38.63	82.61	52.93	50.36	45.52
		Forward T [25]	88.63	85.07	79.12	64.30	91.32	90.35	89.25	88.12
		Forward $\hat{T}$ [25]	87.99	83.25	74.96	54.64	90.52	89.09	86.79	83.55
		L_q [39]	89.83	87.13	82.54	64.07	90.91	89.33	85.45	76.74
		Truncated L_q [39]	89.70	87.62	82.70	67.92	90.43	89.45	87.10	82.28
		Ours	**94.23**	**92.43**	**88.32**	-	**94.57**	**93.35**	**91.80**	**89.86**
CIFAR100	ResNet34	CE	58.72	48.20	37.41	18.10	66.54	59.20	51.40	42.74
		MAE [3]	15.80	9.03	7.74	3.76	13.38	11.50	8.91	8.20
		Forward T [25]	63.16	54.65	44.62	24.83	**71.05**	**71.08**	**70.76**	**70.82**
		Forward $\hat{T}$ [25]	39.19	31.05	19.12	8.99	45.96	42.46	38.13	34.44
		L_q [39]	66.81	61.77	53.16	29.16	68.36	66.59	61.45	47.22
		Truncated L_q [39]	67.61	62.64	54.04	29.60	68.86	66.59	61.87	47.66
		Ours	**71.52**	**66.39**	**56.51**	-	70.35	63.12	54.87	45.70

Table 3: Comparison with results reported by Zhang *et al.* [39]

Mixup

[Zhang H, Cisse M, Dauphin YN, Lopez-Paz D. mixup: Beyond empirical risk minimization. ICLR 2018.]

- Avoid overfitting by data augmentation based on convex combinations of pairs of examples and their labels

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j, \qquad \text{where } x_i, x_j \text{ are raw input vectors}$$
$$\tilde{y} = \lambda y_i + (1-\lambda)y_j, \qquad \text{where } y_i, y_j \text{ are one-hot label encodings}$$

Mixup

[Zhang H, Cisse M, Dauphin YN, Lopez-Paz D. mixup: Beyond empirical risk minimization. ICLR 2018.]

- Avoid overfitting by data augmentation based on convex combinations of pairs of examples and their labels

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j, \qquad \text{where } x_i, x_j \text{ are raw input vectors}$$
$$\tilde{y} = \lambda y_i + (1-\lambda)y_j, \qquad \text{where } y_i, y_j \text{ are one-hot label encodings}$$

Mixup

[Zhang H, Cisse M, Dauphin YN, Lopez-Paz D. mixup: Beyond empirical risk minimization. ICLR 2018.]

- Avoid overfitting by data augmentation based on convex combinations of pairs of examples and their labels

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j, \qquad \text{where } x_i, x_j \text{ are raw input vectors}$$
$$\tilde{y} = \lambda y_i + (1-\lambda)y_j, \qquad \text{where } y_i, y_j \text{ are one-hot label encodings}$$

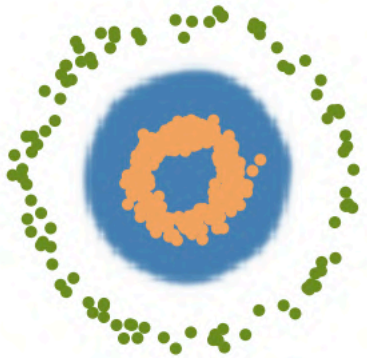
```
# y1, y2 should be one-hot vectors
for (x1, y1), (x2, y2) in zip(loader1, loader2):
    lam = numpy.random.beta(alpha, alpha)
    x = Variable(lam * x1 + (1. - lam) * x2)
    y = Variable(lam * y1 + (1. - lam) * y2)
    optimizer.zero_grad()
    loss(net(x), y).backward()
    optimizer.step()
```

(a) One epoch of *mixup* training in PyTorch.

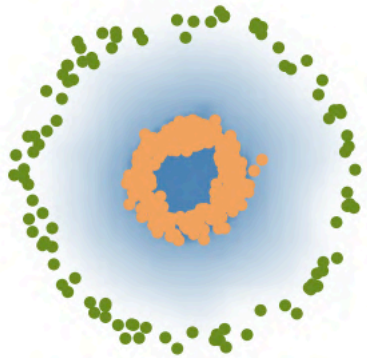
Mixup

- Data augmentation (2 classes)

No MixUp



MixUp



Encourages the model to behave linearly in-between training examples

Reduces oscillations when predicting outside the training examples

Mixup

- CIFAR-10

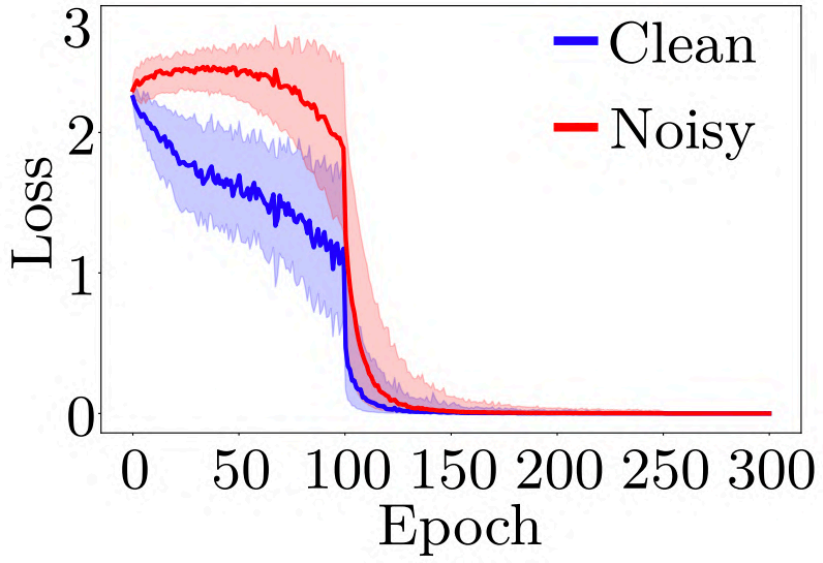
Label corruption	Method	Test error		Training error	
		Best	Last	Real	Corrupted
20%	ERM	12.7	16.6	0.05	0.28
	ERM + dropout ($p = 0.7$)	8.8	10.4	5.26	83.55
	mixup ($\alpha = 8$)	**5.9**	6.4	2.27	86.32
	mixup + dropout ($\alpha = 4, p = 0.1$)	6.2	**6.2**	1.92	85.02
50%	ERM	18.8	44.6	0.26	0.64
	ERM + dropout ($p = 0.8$)	14.1	15.5	12.71	86.98
	mixup ($\alpha = 32$)	11.3	12.7	5.84	85.71
	mixup + dropout ($\alpha = 8, p = 0.3$)	**10.9**	**10.9**	7.56	87.90
80%	ERM	36.5	73.9	0.62	0.83
	ERM + dropout ($p = 0.8$)	30.9	35.1	29.84	86.37
	mixup ($\alpha = 32$)	25.3	30.9	18.92	85.44
	mixup + dropout ($\alpha = 8, p = 0.3$)	**24.0**	**24.8**	19.70	87.67

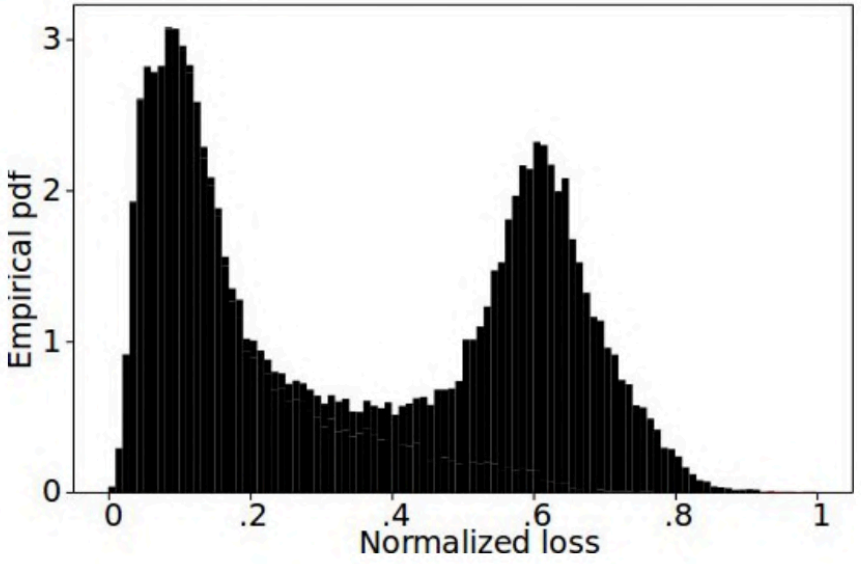
Table 2: Results on the corrupted label experiments for the best models.

Mixture Model and Mixup

[Eric Arazo *et al*. Unsupervised Label Noise Modeling and Loss Correction, ICML 2019]

- Before label noise memorization: clean and noisy samples are (to some extent) distinguishable in the loss

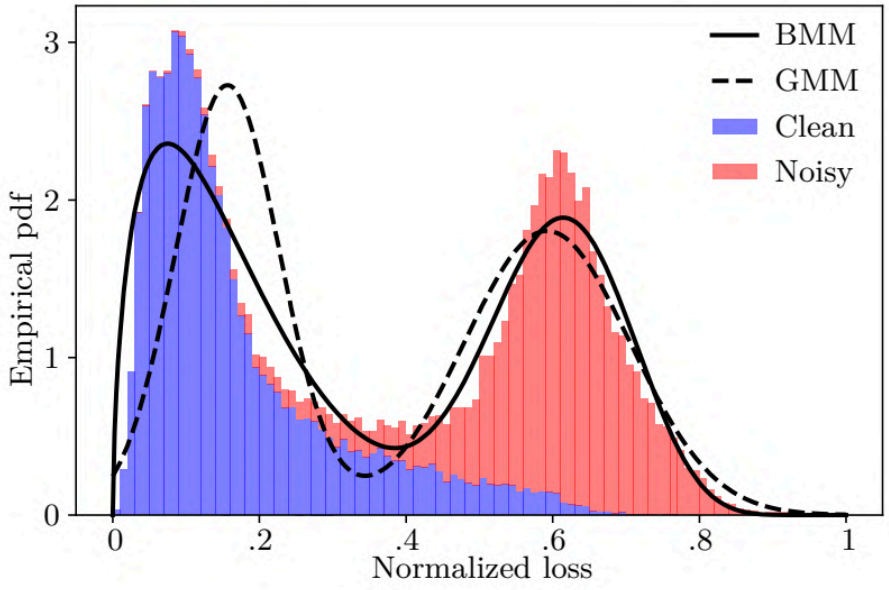




Mixture Model and Mixup

[Eric Arazo *et al*. Unsupervised Label Noise Modeling and Loss Correction, ICML 2019]

- Two-component mixture model suits the problem



$$\mathcal{L} = -\sum_{i=1}^{N} \big((1 - w_i) y_i + w_i z_i\big)^T \log(h_i)$$

label

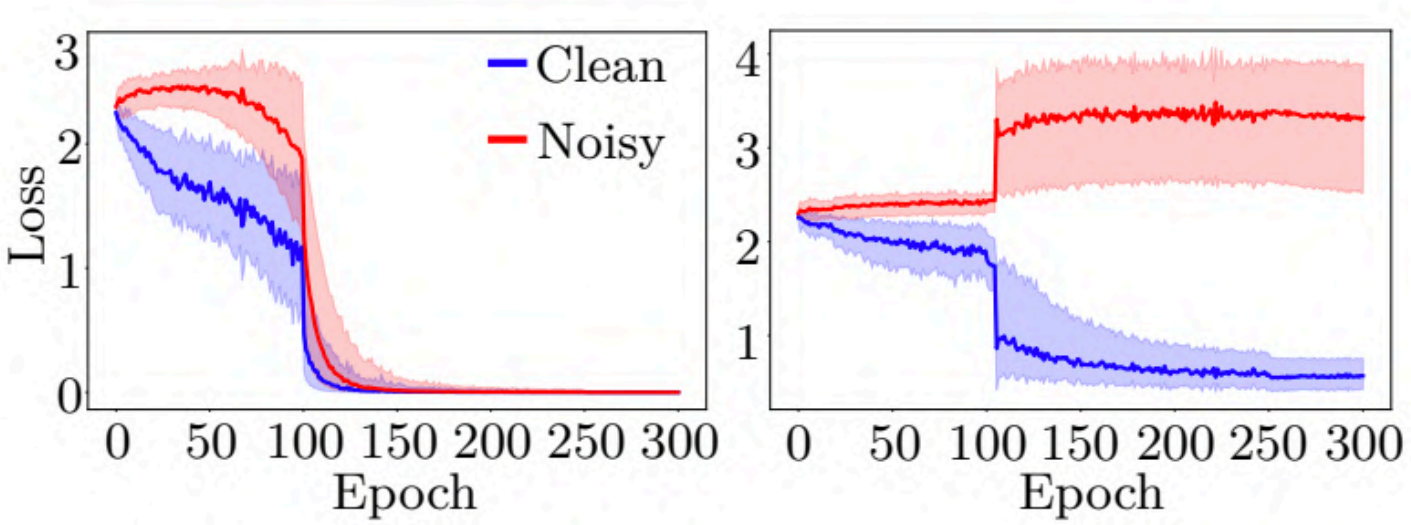
prediction

w_i: wight for prediction z_i of sample x_i

Mixture Model and Mixup

[Eric Arazo *et al*. Unsupervised Label Noise Modeling and Loss Correction, ICML 2019]

- They also add mixup data augmentation



CE loss

Arazo

MixMatch [Semi-Supervised context]

- Why Semi-supervised solutions are interesting to noisy labels?

MixMatch: A Holistic Approach to Semi-Supervised Learning

NIPS 2019

David Berthelot
Google Research
dberth@google.com

Nicholas Carlini
Google Research
ncarlini@google.com

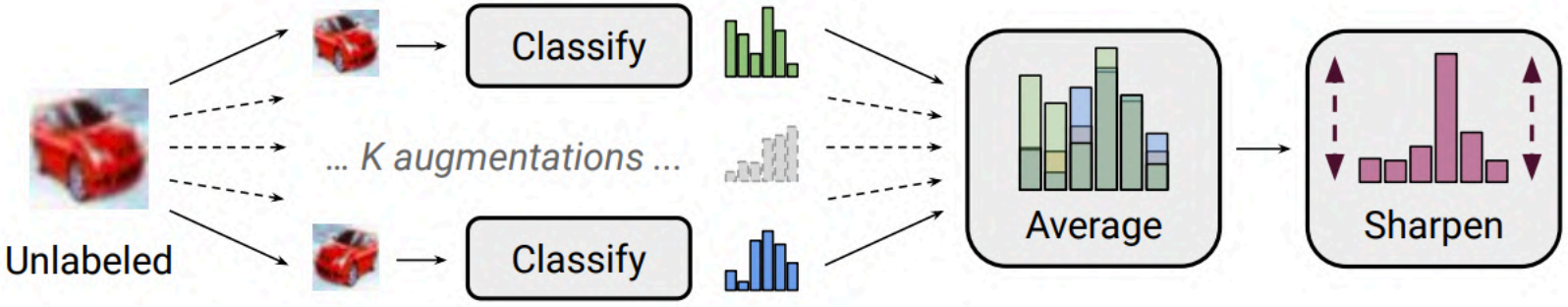
Ian Goodfellow
Work done at Google
ian-academic@mailfence.com

Avital Oliver
Google Research
avitalo@google.com

Nicolas Papernot
Google Research
papernot@google.com

Colin Raffel
Google Research
craffel@google.com

MixMatch



$$\mathcal{X}', \mathcal{U}' = \text{MixMatch}(\mathcal{X}, \mathcal{U}, T, K, \alpha)$$

$$\mathcal{L}_{\mathcal{X}} = \frac{1}{|\mathcal{X}'|} \sum_{x,p \in \mathcal{X}'} \text{H}(p, \text{p}_{\text{model}}(y \mid x; \theta))$$

$$\mathcal{L}_{\mathcal{U}} = \frac{1}{L|\mathcal{U}'|} \sum_{u,q \in \mathcal{U}'} \|q - \text{p}_{\text{model}}(y \mid u; \theta)\|_2^2$$

$$\mathcal{L} = \mathcal{L}_{\mathcal{X}} + \lambda_{\mathcal{U}} \mathcal{L}_{\mathcal{U}}$$

MixMatch

Algorithm 1 MixMatch takes a batch of labeled data $\mathcal{X}$ and a batch of unlabeled data $\mathcal{U}$ and produces a collection $\mathcal{X}'$ (resp. $\mathcal{U}'$) of processed labeled examples (resp. unlabeled with guessed labels).

Input: Batch of labeled examples and their one-hot labels $\mathcal{X} = \big((x_b, p_b); b \in (1, \dots, B)\big)$, batch of unlabeled examples $\mathcal{U} = \big(u_b; b \in (1, \dots, B)\big)$, sharpening temperature T, number of augmentations K, Beta distribution parameter α for MixUp.
for $b = 1$ **to** B **do**
  $\hat{x}_b = \text{Augment}(x_b)$ *// Apply data augmentation to* x_b
  for $k = 1$ **to** K **do**
    $\hat{u}_{b,k} = \text{Augment}(u_b)$ *// Apply* k^{th} *round of data augmentation to* u_b
  end for
  $\bar{q}_b = \frac{1}{K} \sum_k \text{p}_{\text{model}}(y \mid \hat{u}_{b,k}; \theta)$ *// Compute average predictions across all augmentations of* u_b
  $q_b = \text{Sharpen}(\bar{q}_b, T)$ *// Apply temperature sharpening to the average prediction (see eq. (7))*
end for
$\hat{\mathcal{X}} = \big((\hat{x}_b, p_b); b \in (1, \dots, B)\big)$ *// Augmented labeled examples and their labels*
$\hat{\mathcal{U}} = \big((\hat{u}_{b,k}, q_b); b \in (1, \dots, B), k \in (1, \dots, K)\big)$ *// Augmented unlabeled examples, guessed labels*
$\mathcal{W} = \text{Shuffle}\big(\text{Concat}(\hat{\mathcal{X}}, \hat{\mathcal{U}})\big)$ *// Combine and shuffle labeled and unlabeled data*
$\mathcal{X}' = \big(\text{MixUp}(\hat{\mathcal{X}}_i, \mathcal{W}_i); i \in (1, \dots, |\hat{\mathcal{X}}|)\big)$ *// Apply* MixUp *to labeled data and entries from* $\mathcal{W}$
$\mathcal{U}' = \big(\text{MixUp}(\hat{\mathcal{U}}_i, \mathcal{W}_{i+|\hat{\mathcal{X}}|}); i \in (1, \dots, |\hat{\mathcal{U}}|)\big)$ *// Apply* MixUp *to unlabeled data and the rest of* $\mathcal{W}$
return $\mathcal{X}', \mathcal{U}'$

MixMatch

Ablation	250 labels	4000 labels
MixMatch	11.80	6.00
MixMatch without distribution averaging ($K = 1$)	17.09	8.06
MixMatch with $K = 3$	11.55	6.23
MixMatch with $K = 4$	12.45	5.88
MixMatch without temperature sharpening ($T = 1$)	27.83	10.59
MixMatch with parameter EMA	11.86	6.47
MixMatch without MixUp	39.11	10.97
MixMatch with MixUp on labeled only	32.16	9.22
MixMatch with MixUp on unlabeled only	12.35	6.83
MixMatch with MixUp on separate labeled and unlabeled	12.26	6.50
Interpolation Consistency Training [45]	38.60	6.81

Table 4: Ablation study results. All values are error rates on CIFAR-10 with 250 or 4000 labels.

SOTA: DivideMix

[Li J, Socher R, Hoi SC. DivideMix: Learning with Noisy Labels as Semi-supervised Learning. ICLR 2020.]

- Co-teaching
- Noisy label identification using loss and GMM
- Sophisticated data augmentation (MixMatch)
- Similar to [Arazo E, Ortego D, Albert P, O'Connor NE, McGuinness K. Unsupervised label noise modeling and loss correction. ICML 2019]

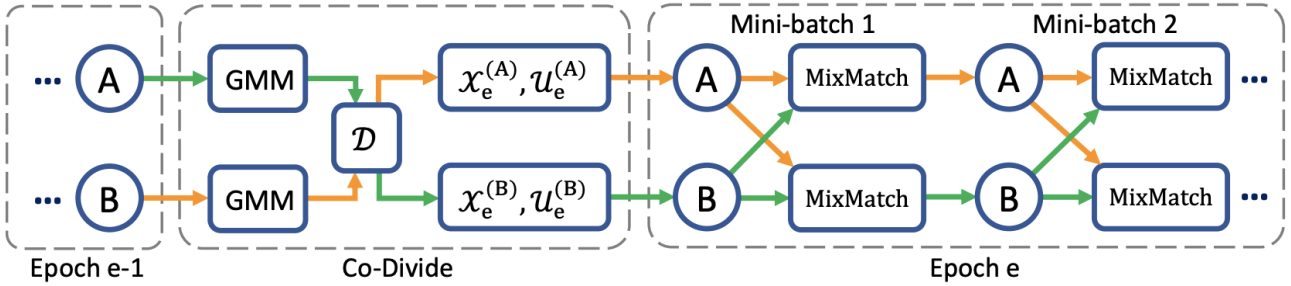


Figure 1: DivideMix trains two networks (A and B) simultaneously. At each epoch, a network models its per-sample loss distribution with a GMM to divide the dataset into a labeled set (mostly clean) and an unlabeled set (mostly noisy), which is then used as training data for the other network (*i.e.* co-divide). At each mini-batch, a network performs semi-supervised training using an improved MixMatch method. We perform label co-refinement on the labeled samples and label co-guessing on the unlabeled samples.

SOTA: DivideMix

[Li J, Socher R, Hoi SC. DivideMix: Learning with Noisy Labels as Semi-supervised Learning. ICLR 2020.]

Algorithm 1: DivideMix. Line 4-8: co-divide; Line 17-18: label co-refinement; Line 20: label co-guessing.

Input: $\theta^{(1)}$ and $\theta^{(2)}$, training dataset $(\mathcal{X}, \mathcal{Y})$, clean probability threshold τ, number of augmentations M, sharpening temperature T, unsupervised loss weight λ_u, Beta distribution parameter α for MixMatch.

$\theta^{(1)}, \theta^{(2)} = \text{WarmUp}(\mathcal{X}, \mathcal{Y}, \theta^{(1)}, \theta^{(2)})$ *// standard training (with confidence penalty)*

while $e < \text{MaxEpoch}$ **do**

  $\mathcal{W}^{(2)} = \text{GMM}(\mathcal{X}, \mathcal{Y}, \theta^{(1)})$ *// model per-sample loss with $\theta^{(1)}$ to obtain clean proabability for $\theta^{(2)}$*

  $\mathcal{W}^{(1)} = \text{GMM}(\mathcal{X}, \mathcal{Y}, \theta^{(2)})$ *// model per-sample loss with $\theta^{(2)}$ to obtain clean proabability for $\theta^{(1)}$*

  for $k = 1, 2$ **do** *// train the two networks one by one*

    $\mathcal{X}_e^{(k)} = \{(x_i, y_i, w_i) | w_i \geq \tau, \forall (x_i, y_i, w_i) \in (\mathcal{X}, \mathcal{Y}, \mathcal{W}^{(k)})\}$ *// labeled training set for $\theta^{(k)}$*

    $\mathcal{U}_e^{(k)} = \{x_i | w_i < \tau, \forall (x_i, w_i) \in (\mathcal{X}, \mathcal{W}^{(k)})\}$ *// unlabeled training set for $\theta^{(k)}$*

    for iter $= 1$ **to** num_iters **do**

      From $\mathcal{X}_e^{(k)}$, draw a mini-batch $\{(x_b, y_b, w_b); b \in (1, ..., B)\}$

      From $\mathcal{U}_e^{(k)}$, draw a mini-batch $\{u_b; b \in (1, ..., B)\}$

      for $b = 1$ **to** B **do**

        for $m = 1$ **to** M **do**

          $\hat{x}_{b,m} = \text{Augment}(x_b)$ *// apply m^{th} round of augmentation to x_b*

          $\hat{u}_{b,m} = \text{Augment}(u_b)$ *// apply m^{th} round of augmentation to u_b*

        end

        $p_b = \frac{1}{M} \sum_m \text{p}_{\text{model}}(\hat{x}_{b,m}; \theta^{(k)})$ *// average the predictions across augmentations of x_b*

        $\bar{y}_b = w_b y_b + (1 - w_b) p_b$

        // refine ground-truth label guided by the clean probability produced by the other network

        $\hat{y}_b = \text{Sharpen}(\bar{y}_b, T)$ *// apply temperature sharpening to the refined label*

        $\bar{q}_b = \frac{1}{2M} \sum_m \left(\text{p}_{\text{model}}(\hat{u}_{b,m}; \theta^{(1)}) + \text{p}_{\text{model}}(\hat{u}_{b,m}; \theta^{(2)})\right)$

        // co-guessing: average the predictions from both networks across augmentations of u_b

        $q_b = \text{Sharpen}(\bar{q}_b, T)$ *// apply temperature sharpening to the guessed label*

      end

      $\hat{\mathcal{X}} = \{(\hat{x}_{b,m}, \hat{y}_b); b \in (1, ..., B), m \in (1, ..., M)\}$ *// augmented labeled mini-batch*

      $\hat{\mathcal{U}} = \{(\hat{u}_{b,m}, q_b); b \in (1, ..., B), m \in (1, ..., M)\}$ *// augmented unlabeled mini-batch*

      $\mathcal{L}_{\mathcal{X}}, \mathcal{L}_{\mathcal{U}} = \text{MixMatch}(\hat{\mathcal{X}}, \hat{\mathcal{U}})$ *// apply MixMatch*

      $\mathcal{L} = \mathcal{L}_{\mathcal{X}} + \lambda_u \mathcal{L}_{\mathcal{U}} + \lambda_r \mathcal{L}_{\text{reg}}$ *// total loss*

      $\theta^{(k)} = \text{SGD}(\mathcal{L}, \theta^{(k)})$ *// update model parameters*

    end

  end

end

Having acquired $\hat{\mathcal{X}}$ (and $\hat{\mathcal{U}}$) which consists of multiple augmentations of labeled (unlabeled) samples and their refined (guessed) labels, we follow MixMatch to "mix" the data, where each sample is interpolated with another sample randomly chosen from the combined mini-batch of $\hat{\mathcal{X}}$ and $\hat{\mathcal{U}}$. Specifically, for a pair of samples (x_1, x_2) and their corresponding labels (p_1, p_2), the mixed (x', p') is computed by:

$$\lambda \sim \text{Beta}(\alpha, \alpha), \tag{5}$$

$$\lambda' = \max(\lambda, 1 - \lambda), \tag{6}$$

$$x' = \lambda' x_1 + (1 - \lambda') x_2, \tag{7}$$

$$p' = \lambda' p_1 + (1 - \lambda') p_2. \tag{8}$$

MixMatch transforms $\hat{\mathcal{X}}$ and $\hat{\mathcal{U}}$ into $\mathcal{X}'$ and $\mathcal{U}'$. Equation 6 ensures that $\mathcal{X}'$ are "closer" to $\hat{\mathcal{X}}$ than $\hat{\mathcal{U}}$. The loss on $\mathcal{X}'$ is the cross-entropy loss and the loss on $\mathcal{U}'$ is the mean squared error:

$$\mathcal{L}_{\mathcal{X}} = -\frac{1}{|\mathcal{X}'|} \sum_{x,p \in \mathcal{X}'} \sum_c p_c \log(\text{p}_{\text{model}}^c(x; \theta)), \tag{9}$$

$$\mathcal{L}_{\mathcal{U}} = \frac{1}{|\mathcal{U}'|} \sum_{x,p \in \mathcal{U}'} \|p - \text{p}_{\text{model}}(x; \theta)\|_2^2. \tag{10}$$

Under high levels of noise, the network would be encouraged to predict the same class to minimize the loss. To prevent assigning all samples to a single class, we apply the regularization term used by Tanaka et al. (2018) and Arazo et al. (2019), which uses a uniform prior distribution π (*i.e.* $\pi_c = 1/C$) to regularize the model's average output across all samples in the mini-batch:

$$\mathcal{L}_{\text{reg}} = \sum_c \pi_c \log\left(\pi_c \Big/ \frac{1}{|\mathcal{X}'| + |\mathcal{U}'|} \sum_{x \in \mathcal{X}' + \mathcal{U}'} \text{p}_{\text{model}}^c(x; \theta)\right). \tag{11}$$

Finally, the total loss is:

$$\mathcal{L} = \mathcal{L}_{\mathcal{X}} + \lambda_u \mathcal{L}_{\mathcal{U}} + \lambda_r \mathcal{L}_{\text{reg}}. \tag{12}$$

SOTA: DivideMix

[Li J, Socher R, Hoi SC. DivideMix: Learning with Noisy Labels as Semi-supervised Learning. ICLR 2020.]

Dataset		CIFAR-10				CIFAR-100			
Method/Noise ratio		20%	50%	80%	90%	20%	50%	80%	90%
Cross-Entropy	Best	86.8	79.4	62.9	42.7	62.0	46.7	19.9	10.1
	Last	82.7	57.9	26.1	16.8	61.8	37.3	8.8	3.5
Bootstrap	Best	86.8	79.8	63.3	42.9	62.1	46.6	19.9	10.2
(Reed et al., 2015)	Last	82.9	58.4	26.8	17.0	62.0	37.9	8.9	3.8
F-correction	Best	86.8	79.8	63.3	42.9	61.5	46.6	19.9	10.2
(Patrini et al., 2017)	Last	83.1	59.4	26.2	18.8	61.4	37.3	9.0	3.4
Co-teaching+*	Best	89.5	85.7	67.4	47.9	65.6	51.8	27.9	13.7
(Yu et al., 2019)	Last	88.2	84.1	45.5	30.1	64.1	45.3	15.5	8.8
Mixup	Best	95.6	87.1	71.6	52.2	67.8	57.3	30.8	14.6
(Zhang et al., 2018)	Last	92.3	77.6	46.7	43.9	66.0	46.6	17.6	8.1
P-correction*	Best	92.4	89.1	77.5	58.9	69.4	57.5	31.1	15.3
(Yi & Wu, 2019)	Last	92.0	88.7	76.5	58.2	68.1	56.4	20.7	8.8
Meta-Learning*	Best	92.9	89.3	77.4	58.7	68.5	59.2	42.4	19.5
(Li et al., 2019)	Last	92.0	88.8	76.1	58.3	67.7	58.0	40.1	14.3
M-correction	Best	94.0	92.0	86.8	69.1	73.9	66.1	48.2	24.3
(Arazo et al., 2019)	Last	93.8	91.9	86.6	68.7	73.4	65.4	47.6	20.5
DivideMix	Best	**96.1**	**94.6**	**93.2**	**76.0**	**77.3**	**74.6**	**60.2**	**31.5**
	Last	**95.7**	**94.4**	**92.9**	**75.4**	**76.9**	**74.2**	**59.6**	**31.0**

Table 1: Comparison with state-of-the-art methods in test accuracy (%) on CIFAR-10 and CIFAR-100 with symmetric noise. Methods marked by * denote re-implementations based on public code.

Method	Best	Last
Cross-Entropy	85.0	72.3
F-correction (Patrini et al., 2017)	87.2	83.1
M-correction (Arazo et al., 2019)	87.4	86.3
Iterative-CV (Chen et al., 2019)	88.6	88.0
P-correction (Yi & Wu, 2019)	88.5	88.1
Joint-Optim (Tanaka et al., 2018)	88.9	88.4
Meta-Learning (Li et al., 2019)	89.2	88.6
DivideMix	**93.4**	**92.1**

Table 2: Comparison with state-of-the-art methods in test accuracy (%) on CIFAR-10 with 40% asymmetric noise. We re-implement all methods under the same setting.

Method	Test Accuracy
Cross-Entropy	69.21
F-correction (Patrini et al., 2017)	69.84
M-correction (Arazo et al., 2019)	71.00
Joint-Optim (Tanaka et al., 2018)	72.16
Meta-Cleaner (Zhang et al., 2019)	72.50
Meta-Learning (Li et al., 2019)	73.47
P-correction (Yi & Wu, 2019)	73.49
DivideMix	**74.76**

Table 3: Comparison with state-of-the-art methods in test accuracy (%) on Clothing1M. Results for baselines are copied from original papers.

Method	WebVision		ILSVRC12	
	top1	top5	top1	top5
F-correction (Patrini et al., 2017)	61.12	82.68	57.36	82.36
Decoupling (Malach & Shalev-Shwartz, 2017)	62.54	84.74	58.26	82.26
D2L (Ma et al., 2018)	62.68	84.00	57.80	81.36
MentorNet (Jiang et al., 2018)	63.00	81.40	57.80	79.92
Co-teaching (Han et al., 2018)	63.58	85.20	61.48	84.70
Iterative-CV (Chen et al., 2019)	65.24	85.34	61.60	84.98
DivideMix	**77.32**	**91.64**	**75.20**	**90.84**

Table 4: Comparison with state-of-the-art methods trained on (mini) WebVision dataset. Numbers denote top-1 (top-5) accuracy (%) on the WebVision validation set and the ImageNet ILSVRC12 validation set. Results for baseline methods are copied from Chen et al. (2019).

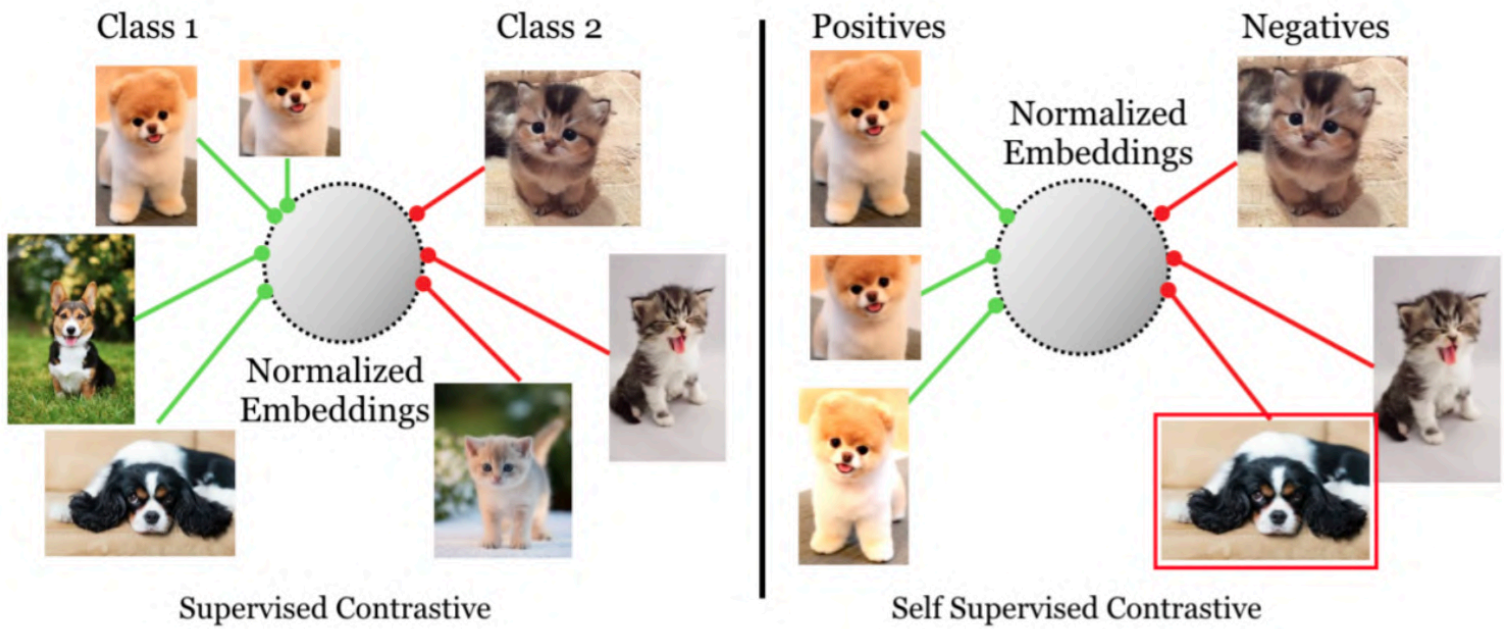
Practice with noisy labels

Code

github.com/filipe-research/tutorial_noisylabels

Trends and Conclusion

SupContrast: Supervised Contrastive Learning



This repo covers an reference implementation for the following papers in PyTorch, using CIFAR as an illustrative example:

(1) Supervised Contrastive Learning. Paper

(2) A Simple Framework for Contrastive Learning of Visual Representations. Paper

Trends

Under review as a conference paper at ICLR 2021

CONTRAST TO DIVIDE: SELF-SUPERVISED PRE-TRAINING FOR LEARNING WITH NOISY LABELS

Anonymous authors
Paper under double-blind review

Lessons Learned

- First approach tried to solve the problem in a supervised way
- Recent approaches have used semi-supervised strategies
- Recent trends are using unsupervised warmup, followed by supervised stage