



# Robust Deep Learning Based on Meta-learning

Deyu Meng

Xi'an Jiaotong University

[dymeng@mail.xjtu.edu.cn](mailto:dymeng@mail.xjtu.edu.cn)

<http://gr.xjtu.edu.cn/web/dymeng>

- **Deep Learning**
- Robust
- Meta-learning

# The Success of Deep Learning Relies on **well-annotated** & **big** data sets

IMAGENET



Microsoft  
Common Objects in Context

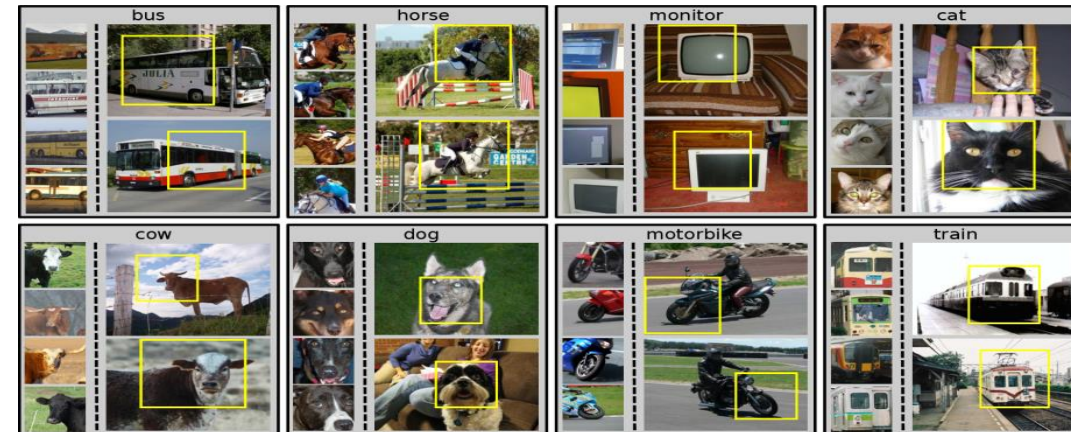
Dataset examples



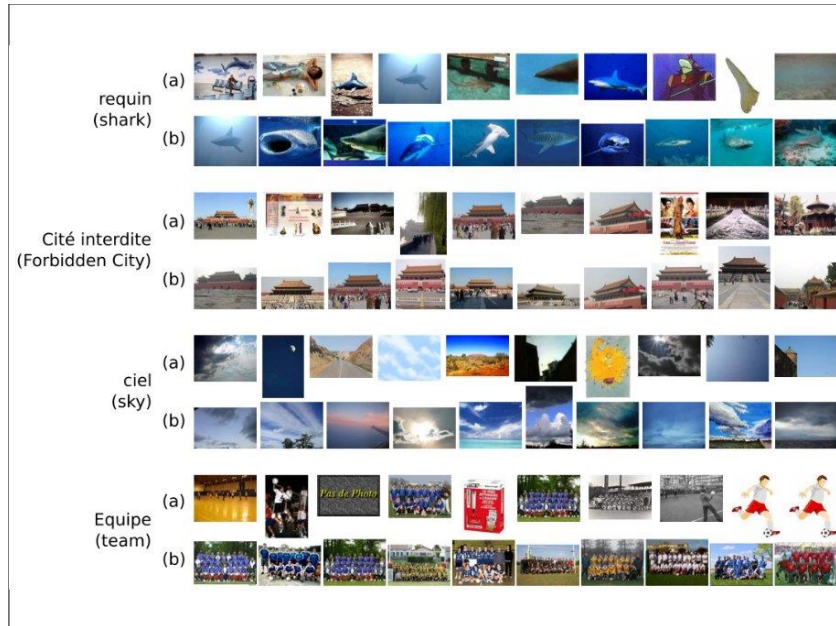
LFW



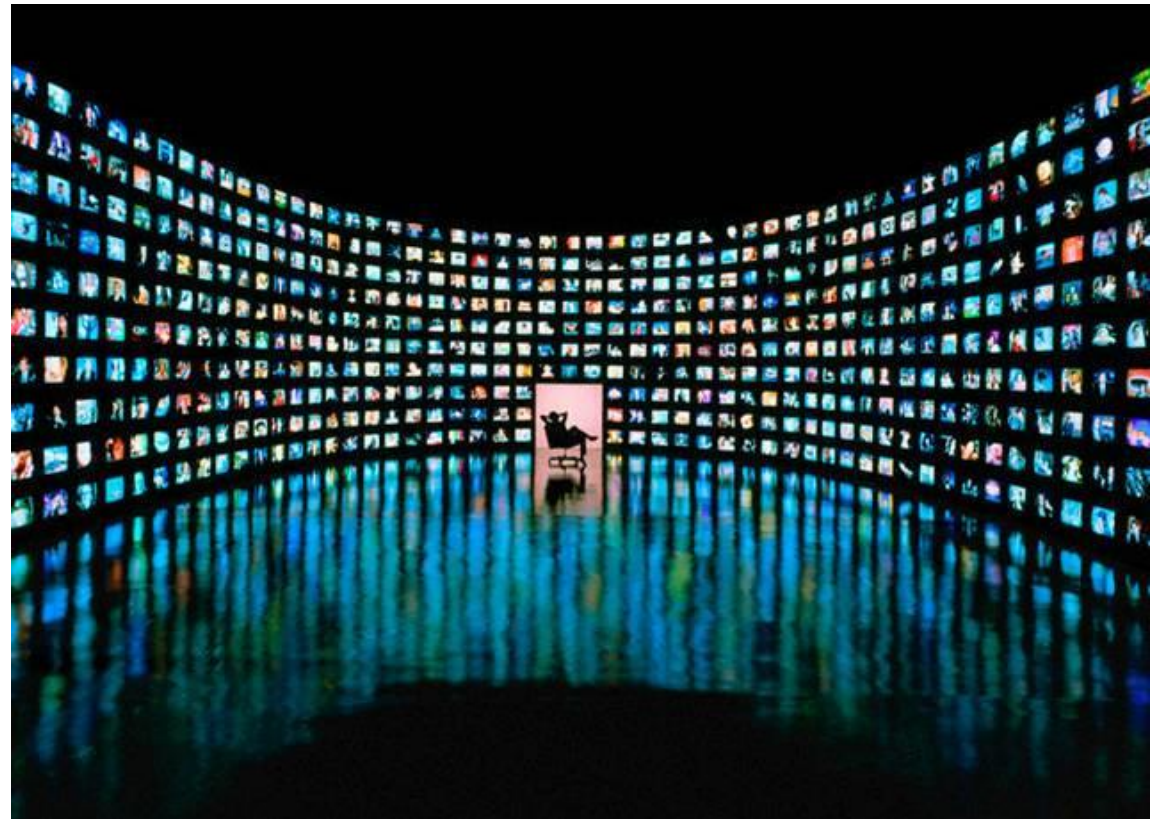
PASCAL2  
Pattern Analysis, Statistical Modelling and  
Computational Learning



What we think  
we have:

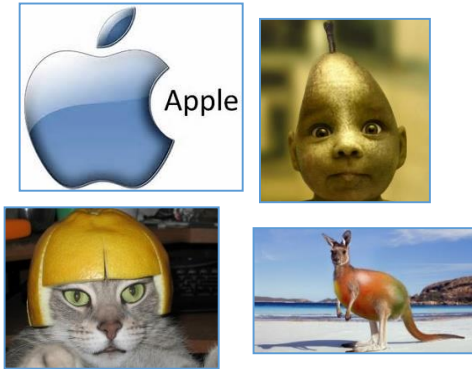


But what we really  
have is always:



# Commonly Encountered Data Bias (low quality data)

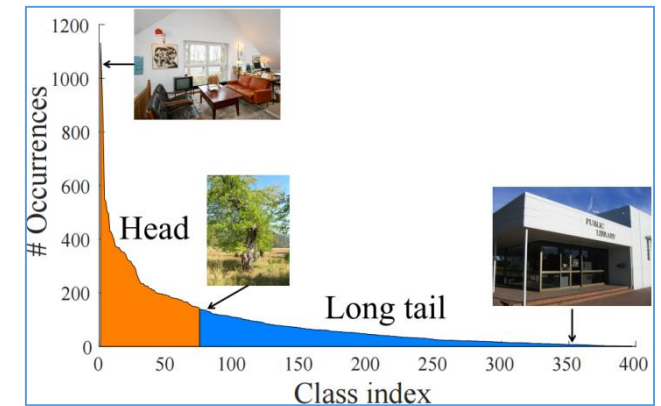
Label noise



Data noise



Class imbalance



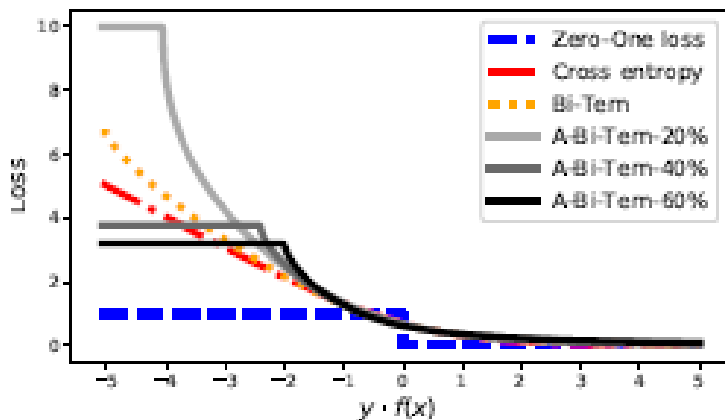
- Deep Learning
- **Robust**
- Meta-learning

# Robust Machine Learning for Data Bias

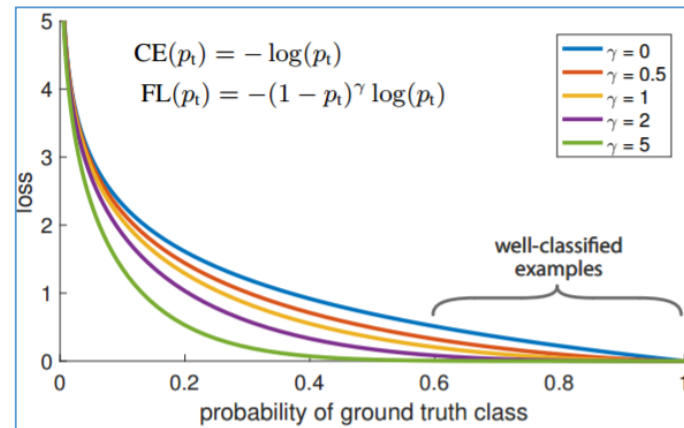
Design specific optimization objective (especially, robust loss) to make it robust to certain data bias:

化腐朽为神奇

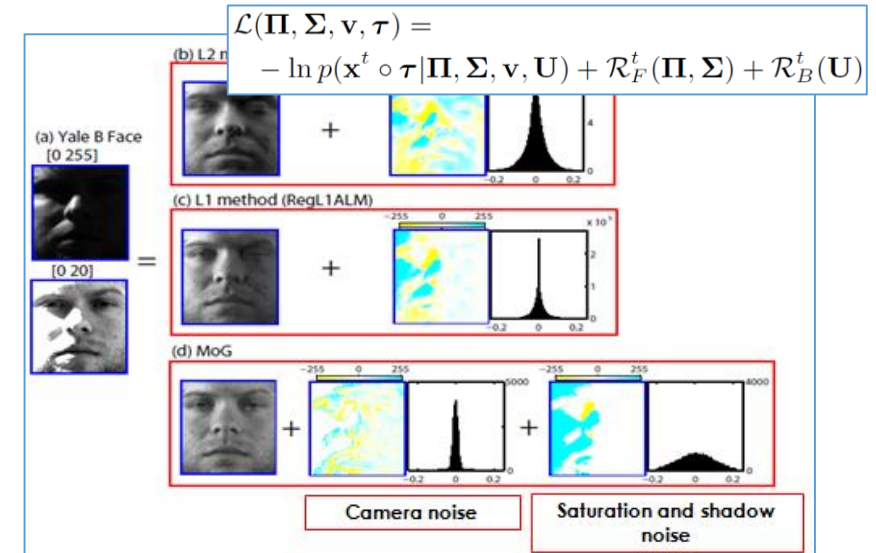
Label noise



Class imbalance



Data noise



# Two Critical Issues

Generalized Cross Entropy

Zhang, et al., NeurIPS, 2018

Symmetric Cross Entropy

Wang, et al., ICCV, 2019

Bi-Tempered logistic Loss

Amid, et al., NeurIPS, 2019

Polynomial SoftWeighting loss

Zhao, et al., AAAI, 2015

Focal loss

Lin, et al., TPAMI, 2018

CT loss

Xie, et al., TMI, 2018

$$\mathcal{L}_{GCE}(D, \mathbf{w}; q) = \frac{1}{N} \sum_{i=1}^N \frac{(1 - f_{j_i}(\mathbf{x}_i)^q)}{q}$$

$$\mathcal{L}_{SL}(D, \mathbf{w}; \gamma_1, \gamma_2) = \gamma_1 \mathcal{L}_{CE} + \gamma_2 \mathcal{L}_{RCE}$$

$$\mathcal{L}_{Bi}(D, \mathbf{w}; t_1, t_2) = -\frac{1}{N} \sum_{i=1}^N [\log_{t_1} \hat{f}_{j_i, t_2}(\mathbf{x}_i) + \frac{1}{2 - t_1} (1 - \sum_{j=1}^c \hat{f}_{j, t_2}(\mathbf{x}_i)^{2 - t_1})]$$

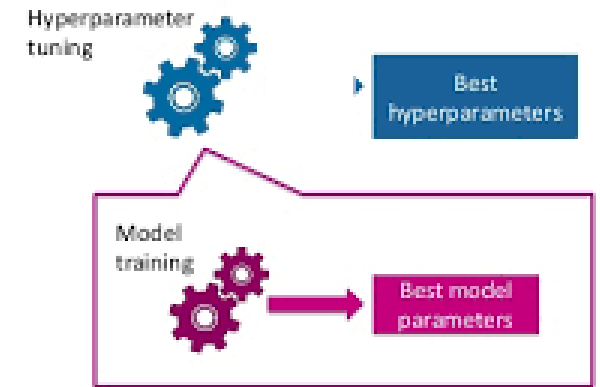
$$\mathcal{L}_{Poly}(D, \mathbf{w}; \lambda, d) = \begin{cases} \frac{(d-1)\lambda}{d} \left[ 1 - \left( 1 - \frac{\mathcal{L}_{CE}(D, \mathbf{w})}{\lambda} \right)^{\frac{d}{d-1}} \right], & \mathcal{L}_{CE} < \lambda, \\ \frac{(d-1)\lambda}{d}, & \mathcal{L}_{CE} \geq \lambda, \end{cases}$$

$$CE(p_i) = -\log(p_i)$$

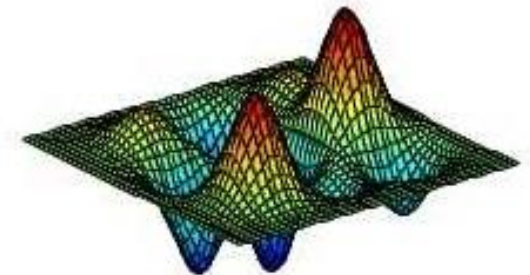
$$FL(p_i) = -(1 - p_i)^\gamma \log(p_i)$$

$$\frac{(P_i - I_i)^2}{2\sigma^2} + I_i \ln(I_{0i}) - I_i Y_i - \ln(I_i!) - I_{0i} e^{-Y_i}$$

## Hyperparameter Tuning

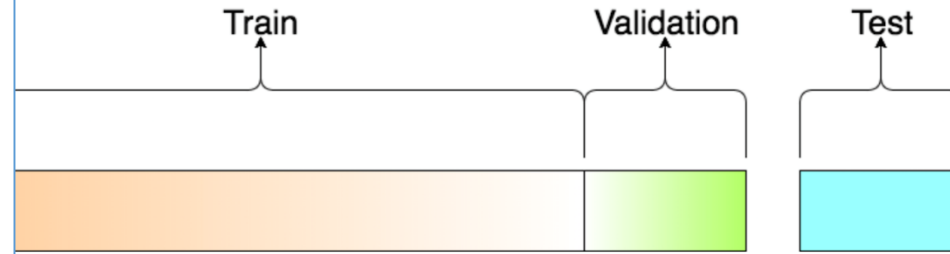
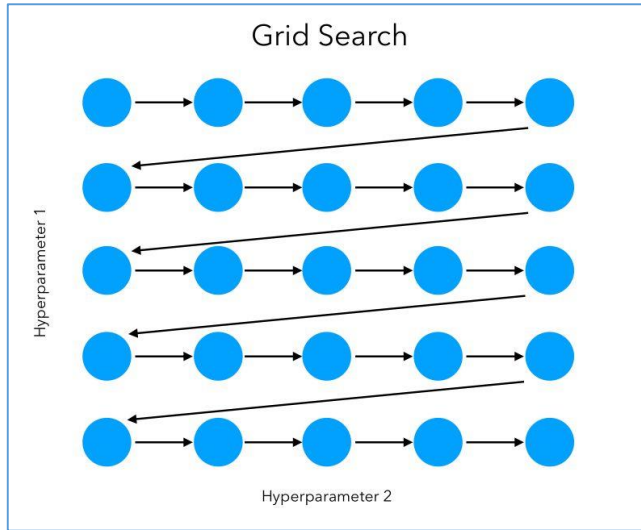


## Non-convexity



- Deep Learning
- Robust
- **Meta-learning**

# Training Data VS Validation Data



Hyperparameter tuning: by validation data

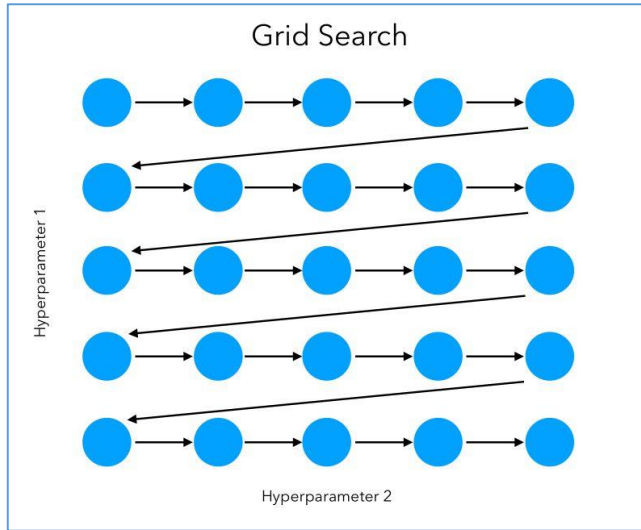
Training loss

$$\mathbf{w}^*(\Theta) = \arg \min_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^N L_i(\mathbf{w}; \Theta).$$

Validation loss

$$\begin{aligned} \Theta^* &= \arg \min_{\Theta} \frac{1}{M} \sum_{i=1}^M L_i^{(m)}(\mathbf{w}^*(\Theta)) \\ &\approx \underset{\Theta \in \{\Theta_1, \Theta_2, \dots, \Theta_s\}}{\operatorname{argmin}} \frac{1}{M} \sum_{i=1}^M L_i^{(m)}(\mathbf{w}^*(\Theta)) \end{aligned}$$

# Training Data VS Validation Data



- ✓ Low efficiency
- ✓ Low accuracy
- ✓ Search instead of optimization
- ✓ Heuristic instead of intelligent

Training loss

$$\mathbf{w}^*(\Theta) = \arg \min_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^N L_i(\mathbf{w}; \Theta).$$

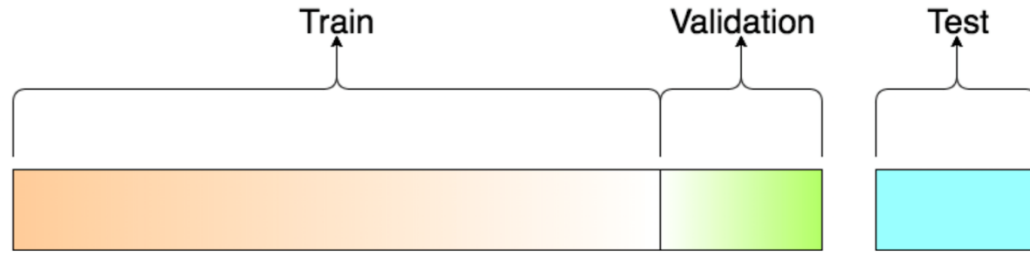
Validation loss

$$\begin{aligned} \Theta^* &= \arg \min_{\Theta} \frac{1}{M} \sum_{i=1}^M L_i^{(m)}(\mathbf{w}^*(\Theta)) \\ &\approx \arg \min_{\Theta \in \{\Theta_1, \Theta_2, \dots, \Theta_s\}} \frac{1}{M} \sum_{i=1}^M L_i^{(m)}(\mathbf{w}^*(\Theta)) \end{aligned}$$

# Intrinsic Functions of Validation Data

- *The function of validation data is higher than training data*
  - Hyper-parameter tuning VS classifier parameter learning
  - Make the model adaptable to data fit (general to specific)
- *Validation data is different from training data!*
  - Teacher vs. student
  - Ideal vs. real
  - High quality vs. low quality
  - Small scale vs. large scale
  - Fixed vs. dynamic (relatively)
- *What we should do?*
  - Lower the threshold of training data collection; higher the threshold of validation data selection

# From Validation Loss Searching to Meta Loss Training



Hyper-parameter tuning: by meta data

Training loss

$$\mathbf{w}^*(\Theta) = \arg \min_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^N L_i(\mathbf{w}; \Theta).$$

- ✓ Optimization instead of search
- ✓ Intelligent instead of heuristic (partially)

Meta loss

$$\begin{aligned} \Theta^* &= \arg \min_{\Theta} \frac{1}{M} \sum_{i=1}^M L_i^{(m)}(\mathbf{w}^*(\Theta)) \\ &= \arg \min_{\Theta \in \mathcal{G}} \frac{1}{M} \sum_{i=1}^M L_i^{(m)}(\mathbf{w}^*(\Theta)) \end{aligned}$$

# Many Recent Attempts

## ◆ Loss function.

Wu L, Tian F, Xia Y, et al. Learning to teach with dynamic loss functions. In NeurIPS, 2018: 6466-6477.

Huang C, Zhai S, Talbott W, et al. Addressing the Loss-Metric Mismatch with Adaptive Loss Alignment. In ICML, 2019: 2891-2900.

Xu H, Zhang H, Hu Z, et al. AutoLoss: Learning Discrete Schedule for Alternate Optimization. In ICLR, 2019.

Li C, Yuan X, Lin C, et al. AM-LFS: AutoML for Loss Function Search. In ICCV, 2019: 8410-8419.

Grabocka J, Scholz R, Schmidt-Thieme L. Learning Surrogate Losses[J]. arXiv preprint arXiv:1905.10108, 2019.

## ◆ Regularization.

Feng J, Simon N. Gradient-based regularization parameter selection for problems with nonsmooth penalty functions[J]. Journal of Computational and Graphical Statistics, 2018, 27(2): 426-435.

Frecon J, Salzo S, Pontil M. Bilevel learning of the group lasso structure. In NeurIPS 2018: 8301-8311.

Streeter M. Learning Optimal Linear Regularizers. In ICML. 2019: 5996-6004.

## ◆ learner (NAS).

Zoph B, Le Q V. Neural architecture search with reinforcement learning. In ICLR, 2017.

Baker B, Gupta O, Naik N, et al. Designing neural network architectures using reinforcement learning. In ICLR, 2017.

Pham H, Guan M, Zoph B, et al. Efficient Neural Architecture Search via Parameter Sharing. ICML. 2018: 4092-4101.

Zoph B, Vasudevan V, Shlens J, et al. Learning transferable architectures for scalable image recognition. In CVPR, 2018: 8697-8710.

Liu H, Simonyan K, Yang Y. Darts: Differentiable architecture search. In ICLR, 2019.

Xie S, Zheng H, Liu C, et al. SNAS: stochastic neural architecture search. In ICLR, 2019.

Liu C, Zoph B, Neumann M, et al. Progressive neural architecture search. In ECCV, 2018: 19-34.

# Many Recent Attempts

## ◆ Hyper-parameters learning.

Maclaurin D, Duvenaud D, Adams R. Gradient-based hyperparameter optimization through reversible learning. In ICML, 2015: 2113-2122.

Pedregosa F. Hyperparameter optimization with approximate gradient. In ICML, 2016: 737-746.

Luketina J, Berglund M, Greff K, et al. Scalable gradient-based tuning of continuous regularization hyperparameters. In ICML. 2016: 2952-2960.

Franceschi L, Donini M, Frasconi P, et al. Forward and reverse gradient-based hyperparameter optimization. In ICML, 2017: 1165-1173.

Franceschi L, Frasconi P, Salzo S, et al. Bilevel Programming for Hyperparameter Optimization and Meta-Learning. In ICML, 2018: 1563-1572.

## ◆ Gradients and learning rate.

Andrychowicz M, Denil M, Gomez S, et al. Learning to learn by gradient descent by gradient descent. In NeurIPS, 2016.

Baydin A G, Cornish R, Rubio D M, et al. Online learning rate adaptation with hypergradient descent. In ICLR, 2018.

Jacobsen A, Schlegel M, Linke C, et al. Meta-descent for Online, Continual Prediction. In AAAI. 2019.

Metz L,, et al. Understanding and correcting pathologies in the training of learned optimizers. In ICML,2019:4556-4565.

Xu Z, Dai A M, Kemp J, et al. Learning an Adaptive Learning Rate Schedule. arXiv preprint arXiv:1909.09712, 2019.

## ◆ Sample reweighing.

Jiang L, Zhou Z, Leung T, et al. MentorNet: Learning Data-Driven Curriculum for Very Deep Neural Networks on Corrupted Labels. In ICML, 2018: 2309-2318.

Ren M, Zeng W, Yang B, et al. Learning to Reweight Examples for Robust Deep Learning. In ICML, 2018: 4331-4340.

Shu J, Xie Q, Yi L, et al. Meta-Weight-Net: Learning an Explicit Mapping For Sample Weighting. In NeurIPS, 2019.

Zhao S, Fard M M, Narasimhan H, et al. Metric-Optimized Example Weights. In ICML 2019: 7533-7542.

- Deep Learning
- **Robust**
- Meta-learning

# Adaptively Learning the Robust Loss

Generalized Cross Entropy

Zhang, et al., NeurIPS, 2018

$$\mathcal{L}_{GCE}(D, \mathbf{w}; q) = \frac{1}{N} \sum_{i=1}^N \frac{(1 - f_{j_t}(\mathbf{x}_i))^q}{q}$$

Symmetric Cross Entropy

Wang, et al., ICCV, 2019

$$\mathcal{L}_{SL}(D, \mathbf{w}; \gamma_1, \gamma_2) = \gamma_1 \mathcal{L}_{CE} + \gamma_2 \mathcal{L}_{RCE}$$

Bi-Tempered logistic Loss

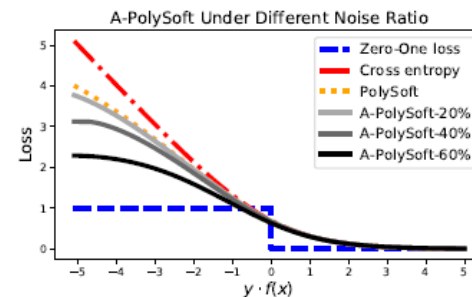
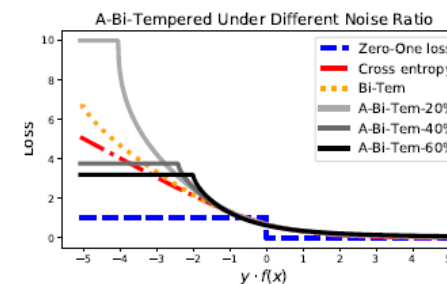
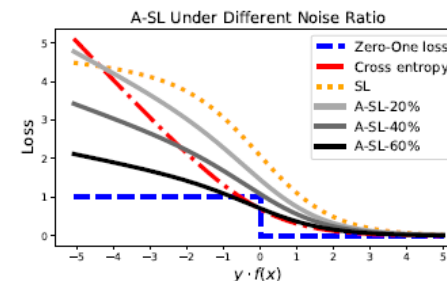
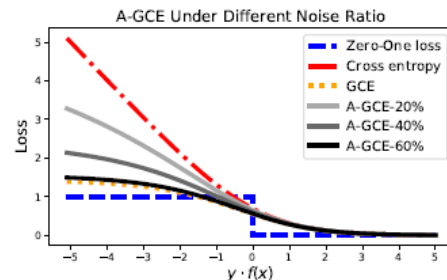
Amid, et al., NeurIPS, 2019

$$\mathcal{L}_{Bi}(D, \mathbf{w}; t_1, t_2) = -\frac{1}{N} \sum_{i=1}^N [\log_{t_1} \hat{f}_{j_t, t_2}(\mathbf{x}_i) + \frac{1}{2 - t_1} (1 - \sum_{j=1}^c \hat{f}_{j, t_2}(\mathbf{x}_i)^{2-t_1})]$$

Polynomial SoftWeighting loss

Zhao, et al., AAAI, 2015

$$\mathcal{L}_{Poly}(D, \mathbf{w}; \lambda, d) = \begin{cases} \frac{(d-1)\lambda}{d} \left[ 1 - \left( 1 - \frac{\mathcal{L}_{CE}(D, \mathbf{w})}{\lambda} \right)^{\frac{d}{d-1}} \right], & \mathcal{L}_{CE} < \lambda, \\ \frac{(d-1)\lambda}{d}, & \mathcal{L}_{CE} \geq \lambda, \end{cases}$$



# Hyperparameter Learning by Meta Learning

Training loss

$$\mathbf{w}^*(\Lambda) = \arg \min_{\mathbf{w}} \mathcal{L}_{Train}(D, \mathbf{w}; \Lambda)$$

Meta loss

$$\Lambda^* = \arg \min_{\Lambda} \mathcal{L}_{Meta}(D_{meta}, \mathbf{w}^*(\Lambda))$$

## Algorithm 1 The Adaptive Robust Loss (ARL) Algorithm

**Input:** Training data  $S$ , meta data  $S_{meta}$ , batch size  $n, m$ , max iterations  $T$ .

**Output:** Classifier network parameter  $\mathbf{w}$ , robust loss hyper-parameter  $\Lambda$ .

- 1: Initialize classifier network parameter  $\mathbf{w}^{(0)}$  and robust loss  $\mathcal{L}_R$  hyper-parameter  $\Lambda^{(0)}$ .
- 2: **for**  $t = 0$  to  $T - 1$  **do**
- 3:    $\{x, y\} \leftarrow \text{SampleMiniBatch}(S, n)$ .
- 4:    $\{x^{(m)}, y^{(m)}\} \leftarrow \text{SampleMiniBatch}(S_{meta}, m)$ .
- 5:   Update  $\Lambda^{(t)}$  by Eq. (8).
- 6:   Update  $\mathbf{w}^{(t)}$  by Eq. (10).
- 7: **end for**

$$\Lambda^{(t)} = \Lambda^{(t-1)} - \beta \nabla_{\Lambda} \mathcal{L}_{Meta}(D_m, \tilde{\mathbf{w}}^{(t)}(\Lambda)) \Big|_{\Lambda^{(t-1)}}, \quad (8)$$

$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} - \alpha \nabla_{\mathbf{w}} \mathcal{L}_{Train}(D_n, \mathbf{w}; \Lambda^{(t)}) \Big|_{\mathbf{w}^{(t-1)}}. \quad (10)$$

# Experimental Results

Table 1. Test accuracy (%) of all competing methods on CIFAR-10 and CIFAR-100 under different noise rates. The best results are in bold.

Models	Datasets	Methods	Symmetric Noise				Asymmetric Noise		
			Noise Rate $\eta$				Noise Rate $\eta$		
			0	0.2	0.4	0.6	0.2	0.4	
ResNet-32	CIFAR-10	CE	92.89±0.32	76.83±2.30	70.77±2.31	63.21±4.22	76.83±2.30	70.77±2.31	
		Forward	<b>93.03±0.11</b>	86.49±0.15	80.51±0.28	75.55±2.25	87.38±0.48	78.98±0.35	
		DMI	90.91±0.20	87.59±0.21	85.13±0.10	80.23±0.39	89.08±0.49	79.33±0.65	
		Meta-Weight-Net	92.04±0.15	89.19±0.57	86.10±0.18	81.31±0.37	90.33±0.61	87.54±0.23	
		PolySoft	91.40±0.39	87.53±0.48	81.49±0.34	75.87±0.25	85.99±1.77	82.71±0.99	
		<b>A-PolySoft</b>	92.12±0.12	<b>89.73±0.20</b>	<b>87.22±0.36</b>	<b>82.49±0.30</b>	<b>90.41±0.16</b>	<b>87.75±0.23</b>	
		GCE	90.03±0.30	88.51±0.37	85.48±0.16	81.29±0.23	88.55±0.22	83.31±0.14	
		<b>A-GCE</b>	91.47±0.19	89.07±0.27	86.36±0.14	81.64±0.11	89.51±0.07	86.35±0.17	
		SL	89.37±0.13	88.76±0.56	85.84±0.74	81.38±1.39	87.63±0.34	83.48±0.48	
		<b>A-SL</b>	91.50±0.16	89.53±0.22	86.36±0.41	82.19±0.30	89.54±0.28	86.45±0.20	
		Bi-Tempered	90.11±0.23	88.51±0.31	84.93±0.67	77.82±0.79	88.23±0.23	82.43±0.23	
		<b>A-Bi-Tempered</b>	92.24±0.20	89.37±0.09	86.32±0.28	81.70±0.21	89.88±0.30	86.86±0.28	
		CIFAR-100	CE	70.50±0.12	50.86±0.27	43.01±1.16	34.43±0.94	50.86±0.27	43.01±1.16
			Forward	67.81±0.61	63.75±0.38	57.53±0.15	46.44±1.03	64.28±0.23	57.90±0.57
	DMI		68.40±0.23	62.66±0.05	56.95±0.11	46.30±0.10	64.05±0.18	58.08±0.22	
	Meta-Weight-Net		69.13±0.33	64.22±0.28	58.64±0.47	47.43±0.76	64.22±0.28	58.64±0.47	
	PolySoft		68.26±0.25	62.41±0.38	56.16±0.30	45.23±0.47	63.05±0.61	56.09±0.26	
	<b>A-PolySoft</b>		68.92±0.41	<b>65.37±1.43</b>	<b>61.38±0.47</b>	<b>52.23±0.63</b>	<b>64.42±0.26</b>	<b>58.73±0.17</b>	
	GCE		67.39±0.12	63.97±0.43	58.33±0.35	41.73±0.36	62.07±0.41	55.25±0.09	
	<b>A-GCE</b>		67.57±0.32	64.58±0.30	58.50±0.15	42.16±0.63	62.46±0.52	56.75±0.44	
	SL		66.43±0.43	52.46±0.18	51.28±0.73	38.39±1.53	52.04±0.89	44.01±1.91	
	<b>A-SL</b>		68.07±0.51	63.73±0.27	57.99±0.37	45.75±0.66	63.25±0.33	56.83±0.19	
	Bi-Tempered		67.68±0.25	63.45±0.48	57.25±0.16	44.72±0.39	63.12±0.28	55.37±0.56	
	<b>A-Bi-Tempered</b>		<b>69.32±0.19</b>	64.48±0.53	59.26±0.12	48.62±0.32	63.78±0.27	56.56±0.08	

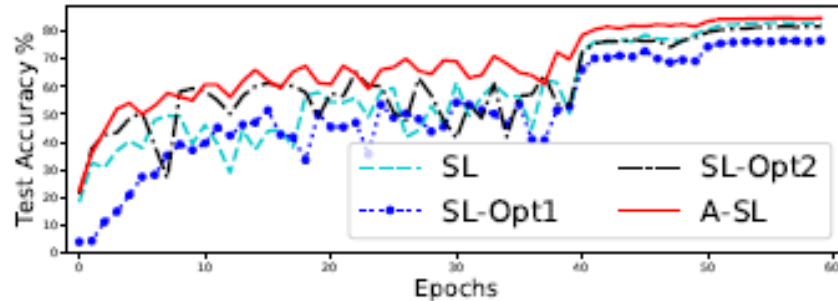
Table 3. Test accuracy (%) on T-ImageNet under different noise fractions. The best results are in bold.

Methods	Symmetric Noise				Asymmetric Noise	
	Noise Rate $\eta$				Noise Rate $\eta$	
	0	0.2	0.4	0.6	0.2	0.4
CE	55.01	43.94	35.14	20.45	42.12	33.58
Forward	<b>55.29</b>	46.57	38.01	24.43	44.98	36.99
DMI	54.50	46.10	40.35	25.23	44.82	36.68
MW-Net	53.58	48.31	43.33	28.23	45.17	37.72
PolySoft	52.18	46.86	40.76	21.48	43.99	36.11
<b>A-PolySoft</b>	54.18	<b>49.24</b>	<b>43.67</b>	<b>28.46</b>	<b>48.65</b>	<b>40.50</b>
GCE	53.11	47.72	38.96	23.93	45.62	35.32
<b>A-GCE</b>	53.46	48.22	41.40	24.11	46.18	36.47
SL	52.48	44.33	35.18	21.82	44.18	34.69
<b>A-SL</b>	53.34	48.99	38.29	22.39	47.68	37.78
Bi-Tem	52.09	45.90	35.36	21.32	44.14	34.37
<b>A-Bi-Tem</b>	54.22	46.67	37.36	22.10	46.91	35.43

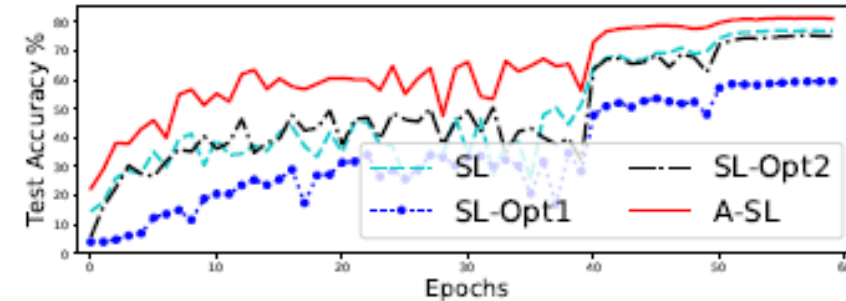
Table 4. Test accuracy (%) of different models on real-world noisy dataset Clothing1M. The best results are in bold.

Methods	CE	Forward	DMI	MN-Net	PolySoft	<b>A-PolySoft</b>	GCE	<b>A-GCE</b>	SL	<b>A-SL</b>	Bi-Tem	<b>A-Bi-Tem</b>
Accuracy	68.94	70.83	72.46	73.72	69.96	73.76	69.75	70.55	71.02	71.83	69.89	70.14

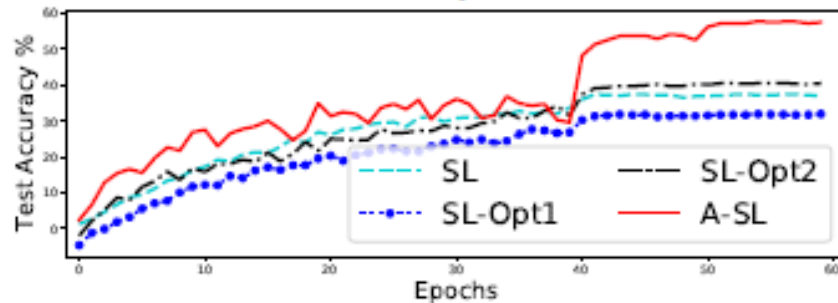
# Experimental Results



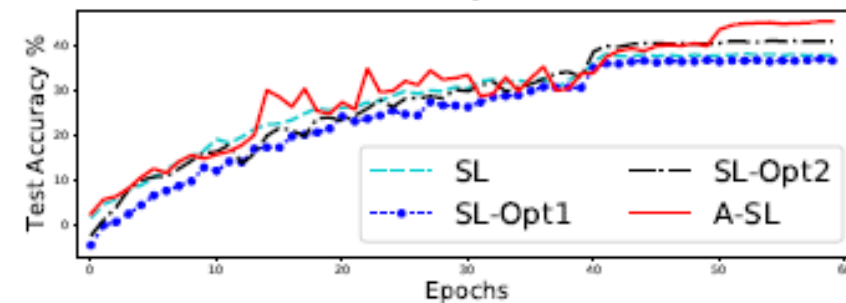
(a) CIFAR-10 40% Symmetric Noise



(b) CIFAR-10 60% Symmetric Noise



(c) CIFAR-100 40% Symmetric Noise



(d) CIFAR-100 60% Symmetric Noise

- ✓ The hyper-parameter adaptively learned by meta-learning actually not the optimal one for the original loss, with fixed hyper-parameter throughout its iteration.
- ✓ Meta learning adaptively finds a proper hyper-parameter and simultaneously explores a good initialization network parameter under its current hyper-parameter in a dynamical way.
- ✓ Such adaptive learning manner should be more suitable for simultaneously obtain optimal values for both of them rather than only updating one under the other fixed.

# When Model Contains Large Amount of Hyperparameters?

Training loss

$$\mathbf{w}^*(\Lambda) = \arg \min_{\mathbf{w}} \mathcal{L}_{Train}(D, \mathbf{w}; \Lambda)$$

Meta loss

$$\Lambda^* = \arg \min_{\Lambda} \mathcal{L}_{Meta}(D_{meta}, \mathbf{w}^*(\Lambda))$$

- Overfitting issue easily occurs (similar to conventional machine learning)
- How to alleviate this issue?
- Build parametric prior representation (neither too large nor too small) for hyperparameters (similar to conventional machine learning)
- Learner VS meta-learner
- Need to deeply understand the data as well as the learning problem!
  - ✓ Multi-view learning, multi-task learning (parameter - similar)
  - ✓ Subspace learning (matrix – low rank)

# When Model Contains Large Amount of Hyperparameters?

E.g. DNN

## Model Design

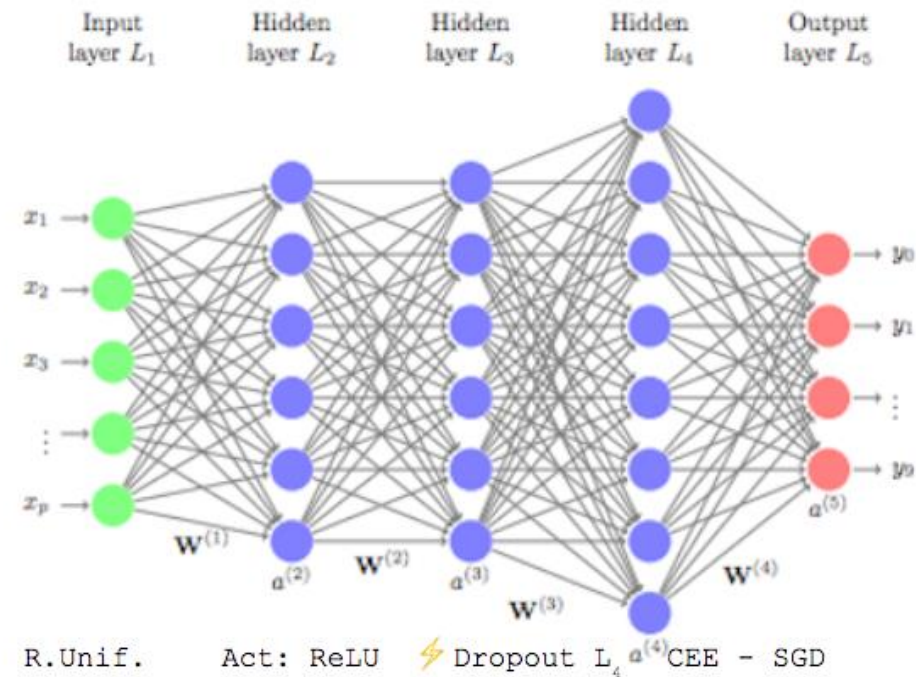
- Weight init.: Random Uniform
- Act.: ReLU
- Loss: CEE
- # Hidden Layers: 3
- # Units per layer  $\{p, p+1, p+1, p+3, 10\}$
- Optimizer: SGD
- Dropout layer:  $L_4$

## Hyperparameters

- Learning rate
- Dropout Rate
- Batch size

## Model Parameters

- $W^{(1)} \Rightarrow W^{(4)}$



- **Deep Learning**
- Robust
- Meta-learning

# Deep Learning with Training Data Bias

**Problem:** big data often come with noisy labels or class imbalance.



YFCC100M

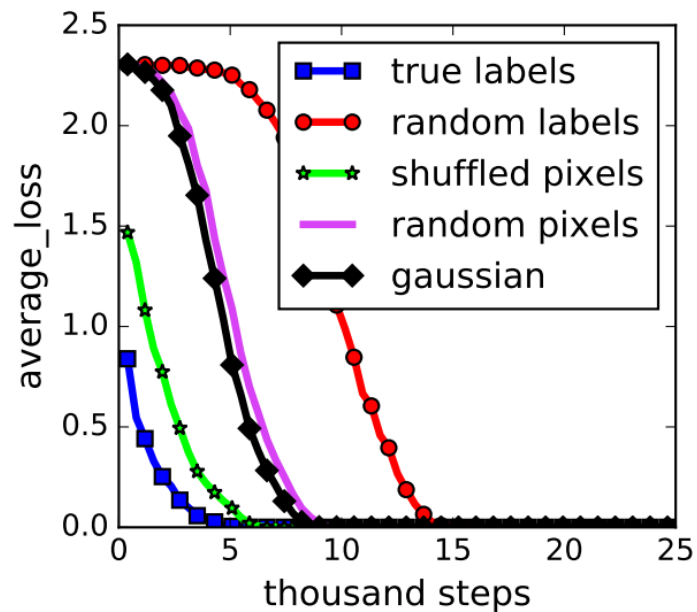
A background image for the Open Images Dataset showing a person in a red shirt in a room with metal frames.

Open Images Dataset

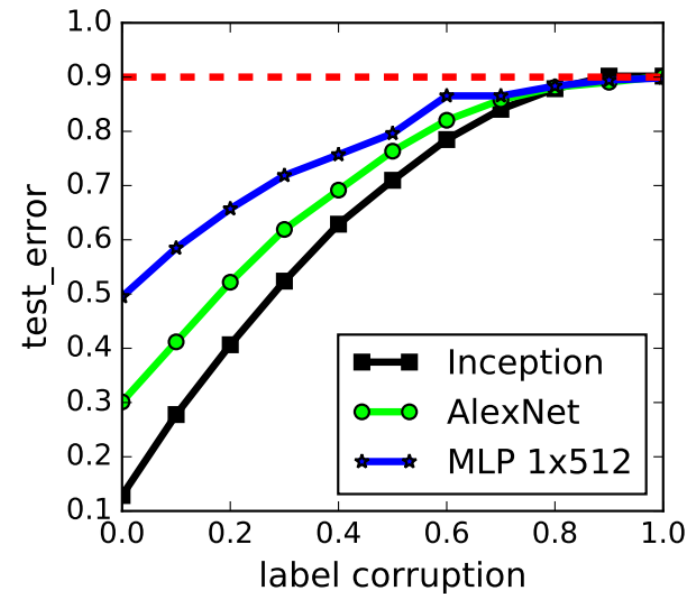
# Deep Networks tend to overfit to Training Data!

Zhang et al. (2017) found that:

*Deep neural networks easily fit(memorizing) random labels.*



(a) learning curves



(c) generalization error growth

How to robustly train deep networks on training data bias to improve the generalization performance?

# Related work: Learning with Training Data Bias

## ◆ Sample weighting methods

- ✓ dataset resampling (Chawla et al., 2002)
- ✓ instance re-weight (Zadrozny, 2004)
- ✓ AdaBoost method (Freund & Schapire, 1997)
- ✓ Hard example mining (Malisiewicz et al., 2011)
- ✓ focal loss (Lin et al., 2018)
- ✓ self-paced learning (Kumar et al., 2010)
- ✓ Iterative reweighting strategy (Fernando & Mikchaoui, 2003; Zhang & Sabuncu, 2018)
- ✓ prediction variance method (Chang et al., 2017)

## ◆ Meta learning methods

- ✓ FWL (Dehghani et al., 2018)
- ✓ learning to teach (Fan et al., 2018; Wu et al., 2018)
- ✓ MentorNet (Jiang et al., 2018)
- ✓ L2RW (Ren et al., 2018)

## ◆ Other methods

- ✓ GLC (Hendrycks et al., 2018)
- ✓ Reed (Reed et al., 2015)
- ✓ Co-teaching (Han et al., 2018)
- ✓ D2L (Ma et al., 2018)
- ✓ S-Model (Goldberger & Ben-Reuven, 2017)

# Sample weighting methods

Existing studies define a curriculum as a **function** (hand-design) for specific tasks and extra hyper-parameter setting.

Strategy	Regularizer $G$	Weight $v^*$
Self-paced [ <i>Kumar et al. NIPS 2010</i> ]	$-\ v\ _1$	$v^* = \mathbb{I}(l_i \leq \lambda)$
Linear weighting [ <i>Jiang et al. AAAI 2015</i> ]	$\frac{1}{2} \sum_{i=1}^n (v_i^2 - 2v_i)$	$v^* = \max(0, 1 - \frac{1}{\lambda} l_i)$
Focal Loss [ <i>Lin et al., ICCV 2017</i> ]	—	$v^* = [1 - \exp(-l_i)]^\alpha$
Hard example mining [ <i>Malisiewicz et al., ICCV 2011</i> ]	—	$v^* = \mathbb{I}(l_i > \lambda(1 - y_i))$
Prediction variance [ <i>Chang et al., NIPS 2017</i> ]	—	$v^* = \frac{1}{Z} \sqrt{\text{Var}(l_i) + \frac{\text{Var}(l_i)}{ l_i }}$

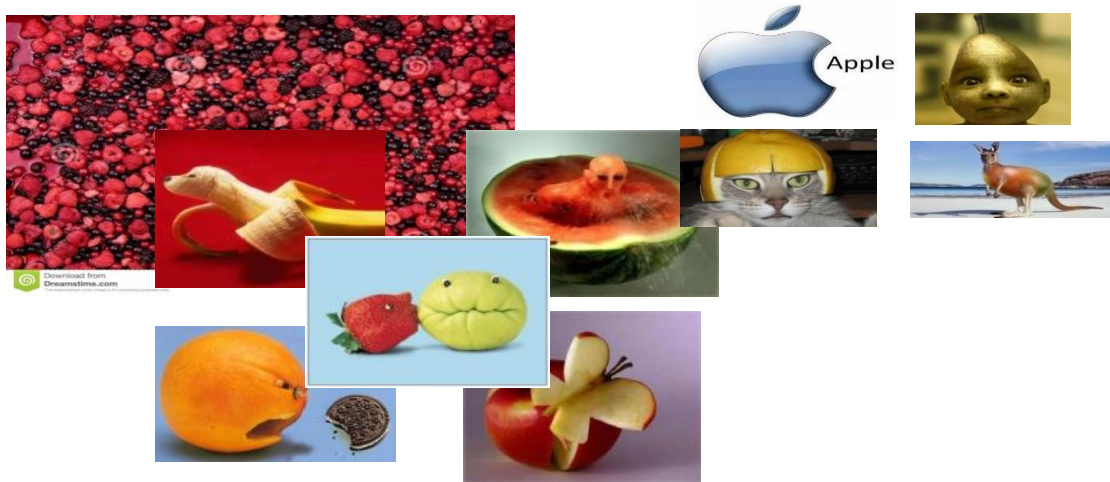
# Sample weighting methods

Strategy	Regularizer $G$	Weight $v^*$
Self-paced [ <i>Kumar et al. NIPS 2010</i> ]	$\ v\ _1$	$v^* = \mathbb{I}(l_i \leq \lambda)$
Linear weighting [ <i>Jiang et al. AAAI 2015</i> ]	$\frac{1}{2} \sum_{i=1}^n (v_i^2 - v_i)$	$v^* = \max(0, 1 - \frac{1}{\lambda} l_i)$
Focal Loss [ <i>Lin et al., ICCV 2017</i> ]	—	$v^* = [1 - \exp(-l_i)]^\alpha$
Hard example mining [ <i>Malisiewicz et al., ICCV 2011</i> ]	—	$v^* = \mathbb{I}(l_i > \lambda(1 - y_i))$
Prediction variance [ <i>Chang et al., NIPS 2017</i> ]	—	$v^* = \frac{1}{Z} \sqrt{\text{Var}(l_i) + \frac{\text{Var}(l_i)}{ l_i }}$

- Need to pre-specify the form of weighting function
- Need to manually set hyper-parameters

# Meta Data and Meta Loss

## Training Data



## Meta Data



## Training Loss

$$\mathbf{w}^*(V) = \arg \min_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^N \mathbf{v}_i L_i(\mathbf{w})$$

## Meta Loss

$$V^* = \arg \min_V \frac{1}{M} \sum_{i=1}^M L_i^{(m)}(\mathbf{w}^*(V))$$

# L2RW [*Ren et al., ICML 2018*]

Key idea: Minimize validation loss **one-step ahead**.

$$\theta_{t+1}(\epsilon) = \theta_t - \alpha \nabla \sum_{i=1}^n \epsilon_i \mathcal{L}_i^{\text{train}}(\theta) \Big|_{\theta=\theta_t}.$$

Ideally, want to find the optimal perturbation:

$$\epsilon^* = \underset{\epsilon}{\operatorname{argmin}} \frac{1}{M} \sum_{i=1}^M \mathcal{L}_i^{\text{val}}(\theta_{t+1}(\epsilon)).$$

Instead, take a single gradient descent step from zero:

$$u_{i,t} = -\eta \frac{\partial}{\partial \epsilon_{i,t}} \frac{1}{m} \sum_{j=1}^m \mathcal{L}_j^{\text{val}}(\theta_{t+1}(\epsilon)) \Big|_{\epsilon_{i,t}=0}.$$

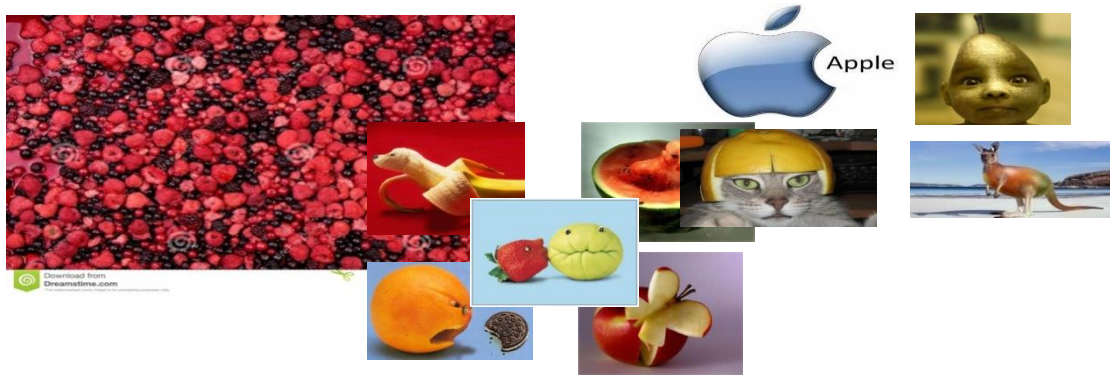
**Notation:**

- $\epsilon_i$ : Perturbation of the weights on  $i$ -th example

**Directly learning weights  
from training and meta  
data**

# Meta Data and Meta Loss

## Training Data



## Meta Data



## Training Loss

$$\mathbf{w}^*(\Theta) = \arg \min_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^N \mathbf{v}(\text{Input}; \Theta) L_i(\mathbf{w})$$

Input

Structure

## Meta Loss

$$\Theta^* = \arg \min_{\Theta} \frac{1}{M} \sum_{i=1}^M L_i^{(m)}(\mathbf{w}^*(\Theta))$$

# MentorNet [Jiang et al., ICML 2018]

---

**Algorithm 1** SPADE for minimizing Eq. (1)

---

**Input** : Dataset  $\mathcal{D}$ , a predefined  $G$  or a learned  $g_m(\cdot; \Theta)$

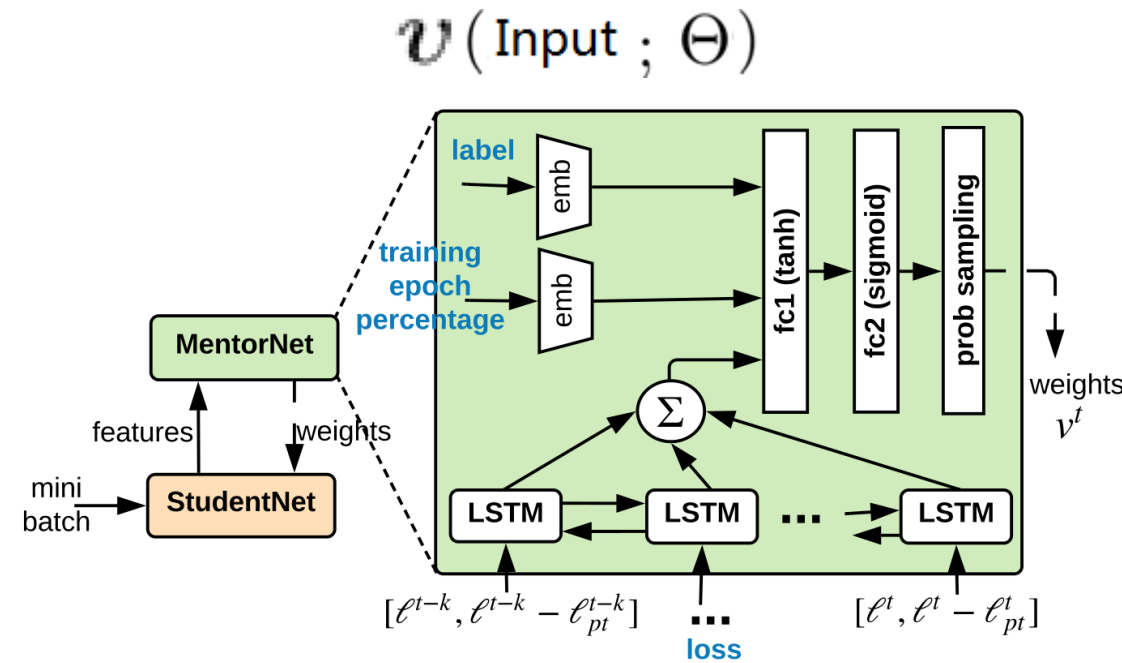
**Output** : The model parameter  $\mathbf{w}$  of StudentNet.

```

1 Initialize  $\mathbf{w}^0, \mathbf{v}^0, t = 0$ 
2 while Not Converged do
3   Fetch a mini-batch  $\Xi_t$  uniformly at random
4   For every  $(\mathbf{x}_i, y_i)$  in  $\Xi_t$  compute  $\phi(\mathbf{x}_i, y_i, \mathbf{w}^t)$ 
5   if update curriculum then
6      $\Theta \leftarrow \Theta^*$ , where  $\Theta^*$  is learned in Sec. 3.1
7   end
8   if  $G$  is used then
9      $\mathbf{v}_{\Xi}^t \leftarrow \mathbf{v}_{\Xi}^{t-1} - \alpha_t \nabla_{\mathbf{v}} \mathbb{F}(\mathbf{w}^{t-1}, \mathbf{v}^{t-1})|_{\Xi_t}$ 
10  end
11  else  $\mathbf{v}_{\Xi}^t \leftarrow g_m(\phi(\Xi_t, \mathbf{w}^{t-1}); \Theta)$ ;
12   $\mathbf{w}^t \leftarrow \mathbf{w}^{t-1} - \alpha_t \nabla_{\mathbf{w}} \mathbb{F}(\mathbf{w}^{t-1}, \mathbf{v}^t)|_{\Xi_t}$ 
13   $t \leftarrow t + 1$ 
14 end
15 return  $\mathbf{w}^t$ 

```

---



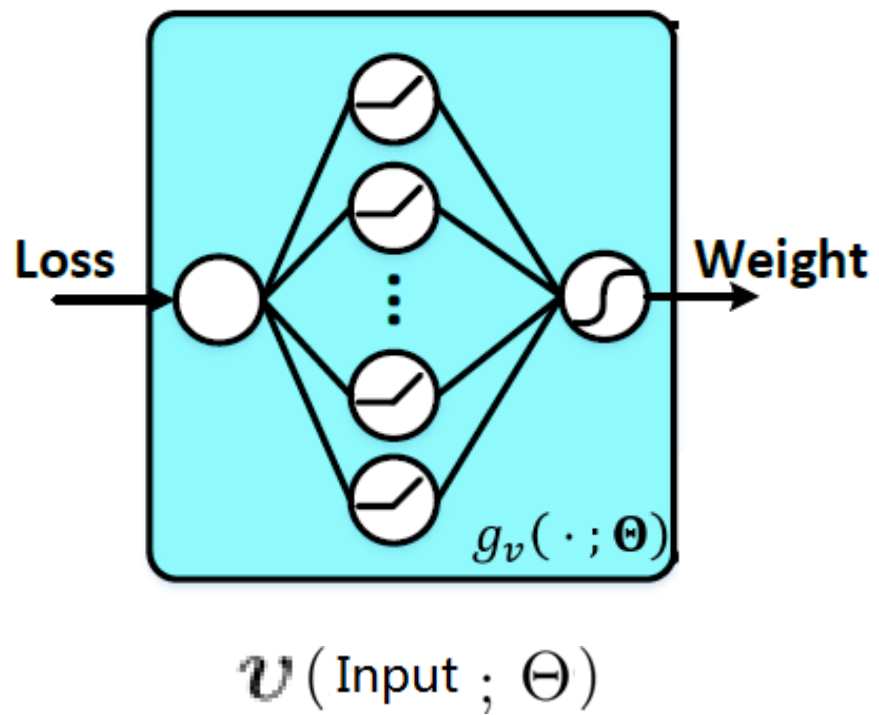
The meta-learner is complex, hard to be reproduced.

**Very Complex Input**  
**Very Complex Theta**

# Our work



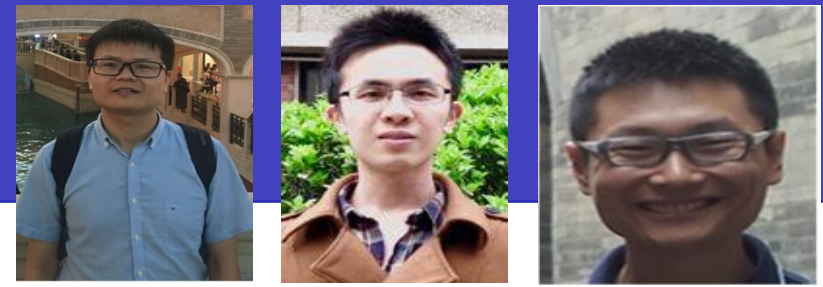
## Meta-Weight-Net



Input: Loss  
Theta: MLP



# Our work



## Inner loop:

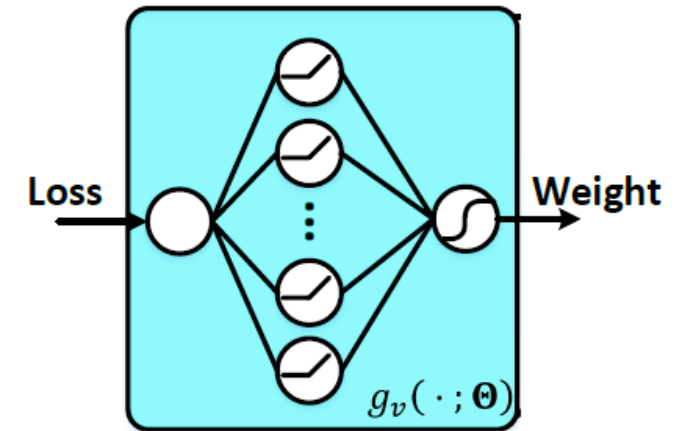
$$\mathbf{w}^*(\Theta) = \arg \min_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^N \mathcal{V}(L_i^{train}(\mathbf{w}); \Theta) L_i^{train}(\mathbf{w})$$

## Outer loop:

$$\Theta^* = \arg \min_{\Theta} \frac{1}{M} \sum_{i=1}^M L_i^{(m)}(\mathbf{w}^*(\Theta)).$$

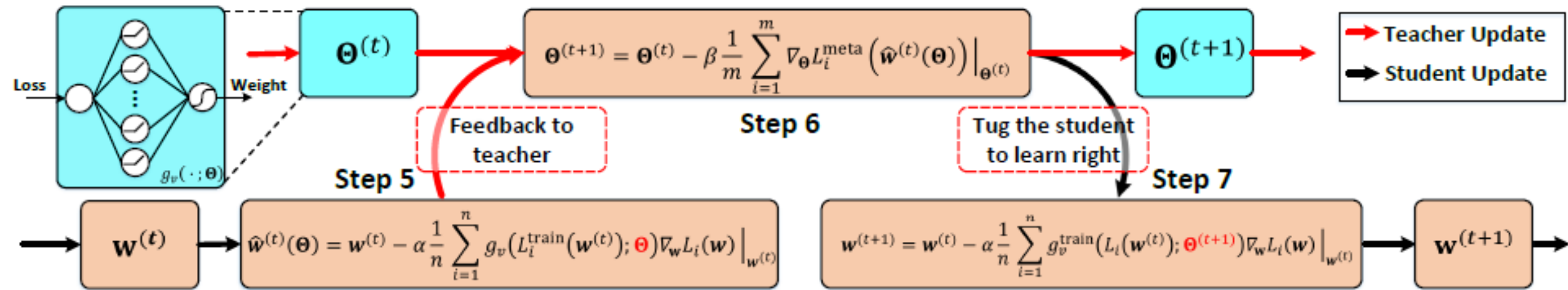
## Notation:

- ◆  $\Theta$ : Parameters of teacher
- ◆  $w$ : Parameters of student



Meta-Weight-Net

# Our work



## Algorithm 1 The MW-Net Learning Algorithm

**Input:** Training data  $\mathcal{D}$ , meta-data set  $\hat{\mathcal{D}}$ , batch size  $n, m$ , max iterations  $T$ .

**Output:** Classifier network parameter  $\mathbf{w}^{(T)}$

- 1: Initialize classifier network parameter  $\mathbf{w}^{(0)}$  and Meta-Weight-Net parameter  $\Theta^{(0)}$ .
- 2: **for**  $t = 0$  **to**  $T - 1$  **do**
- 3:  $\{x, y\} \leftarrow \text{SampleMiniBatch}(\mathcal{D}, n)$ .
- 4:  $\{x^{(meta)}, y^{(meta)}\} \leftarrow \text{SampleMiniBatch}(\hat{\mathcal{D}}, m)$ .
- 5: Formulate the classifier learning function  $\hat{\mathbf{w}}^{(t)}(\Theta)$  by Eq. (3).
- 6: Update  $\Theta^{(t+1)}$  by Eq. (4).
- 7: Update  $\mathbf{w}^{(t+1)}$  by Eq. (5).
- 8: **end for**

Step 5:

$$\hat{\mathbf{w}}^{(t)}(\Theta) = \mathbf{w}^{(t)} - \alpha \frac{1}{n} \times \sum_{i=1}^n \mathcal{V}(L_i^{train}(\mathbf{w}^{(t)}); \Theta) \nabla_{\mathbf{w}} L_i^{train}(\mathbf{w}) \Big|_{\mathbf{w}^{(t)}}$$

Step 6:

$$\Theta^{(t+1)} = \Theta^{(t)} - \beta \frac{1}{m} \sum_{i=1}^m \nabla_{\Theta} L_i^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta)) \Big|_{\Theta^{(t)}}$$

Step 7:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \alpha \frac{1}{n} \times \sum_{i=1}^n \mathcal{V}(L_i^{train}(\mathbf{w}^{(t)}); \Theta^{(t+1)}) \nabla_{\mathbf{w}} L_i^{train}(\mathbf{w}) \Big|_{\mathbf{w}^{(t)}}$$

# Our work

---

## Algorithm 1 The MW-Net Learning Algorithm

---

**Input:** Training data  $\mathcal{D}$ , meta-data set  $\widehat{\mathcal{D}}$ , batch size  $n, m$ , max iterations  $T$ .

**Output:** Classifier network parameter  $\mathbf{w}^{(T)}$

- 1: Initialize classifier network parameter  $\mathbf{w}^{(0)}$  and Meta-Weight-Net parameter  $\Theta^{(0)}$ .
- 2: **for**  $t = 0$  **to**  $T - 1$  **do**
- 3:    $\{x, y\} \leftarrow \text{SampleMiniBatch}(\mathcal{D}, n)$ .
- 4:    $\{x^{(meta)}, y^{(meta)}\} \leftarrow \text{SampleMiniBatch}(\widehat{\mathcal{D}}, m)$ .
- 5:   Formulate the classifier learning function  $\widehat{\mathbf{w}}^{(t)}(\Theta)$  by Eq. (3).
- 6:   Update  $\Theta^{(t+1)}$  by Eq. (4).
- 7:   Update  $\mathbf{w}^{(t+1)}$  by Eq. (5).
- 8: **end for**

**Theorem 1.** Suppose  $\mathcal{V}(\cdot)$  is a differential function with a  $\rho$ -bounded gradient, and the loss function  $\ell$  have  $\rho$ -bounded gradients with respect to training/meta data. Let the learning rate  $\beta_t, 1 \leq t \leq N$  is a monotone descent sequence, and satisfy  $\beta_1 = \min\{\frac{1}{L}, \frac{c}{\sigma\sqrt{T}}\}$  for some  $c > 0$ , such that  $\frac{\sigma\sqrt{T}}{c} \geq L$ . Then the proposed algorithm can achieve  $\mathbb{E}[\|\nabla G(\Theta^{(t)})\|_2^2] \leq \epsilon$  in  $\mathcal{O}(1/\epsilon^2)$  steps. More specifically,

$$\min_{0 \leq t \leq T} \mathbb{E}[\|\nabla \mathcal{L}^{meta}(\Theta^{(t)})\|_2^2] \leq \mathcal{O}\left(\frac{C}{\sqrt{T}}\right), \quad (7)$$

where  $C$  is some constant independent of the convergence process, and  $\sigma$  is the variance of drawing uniformly mini-batch sample at random.

**Theorem 2.** Suppose the loss function  $\ell$  is Lipschitz-smooth with constant  $L$ , and have  $\rho$ -bounded gradients with respect to training/meta data. Let the learning rate  $\alpha_t$  satisfy  $\sum_{t=0}^{\infty} \alpha_t = \infty, \sum_{t=0}^{\infty} \alpha_t^2 < \infty$ , and  $\beta_t, 1 \leq t \leq N$  is a monotone descent sequence. Then

$$\lim_{t \rightarrow \infty} \mathbb{E}[\|\nabla \mathcal{L}^{train}(\mathbf{w}^{(t)}; \Theta^{(t+1)})\|_2^2] = 0. \quad (8)$$

# Experiments

# Experimental Setup: Class Imbalance

## Datasets: CIFAR-10 & CIFAR-100

Table 1: Test accuracy (%) of ResNet-32 on long-tailed CIFAR-10 and CIFAR-100, and the best and the second best results are highlighted in **bold** and ***italic bold***, respectively.

Dataset Name	Long-Tailed CIFAR-10						Long-Tailed CIFAR-100					
	200	100	50	20	10	1	200	100	50	20	10	1
Base Model	65.68	70.36	74.81	82.23	86.39	92.89	34.84	38.32	43.85	51.14	55.71	70.50
Focal Loss	65.29	70.38	76.71	82.76	86.66	<b>93.03</b>	35.62	38.41	44.32	51.95	55.78	<b>70.52</b>
Class-Balanced	<b>68.89</b>	<b>74.57</b>	<b>79.27</b>	<b>84.36</b>	<b>87.49</b>	92.89	36.23	39.60	45.32	<b>52.59</b>	<b>57.99</b>	70.50
Fine-tuning	66.08	71.33	77.42	83.37	86.42	<b>93.23</b>	<b>38.22</b>	<b>41.83</b>	<b>46.40</b>	52.11	57.44	<b>70.72</b>
L2RW	66.51	74.16	78.93	82.12	85.19	89.25	33.38	40.23	44.44	51.64	53.73	64.11
Ours	<b>68.91</b>	<b>75.21</b>	<b>80.06</b>	<b>84.94</b>	<b>87.84</b>	92.66	<b>37.91</b>	<b>42.09</b>	<b>46.74</b>	<b>54.37</b>	<b>58.46</b>	70.37

# Experimental Setup: Noisy Label

## Datasets: CIFAR-10 & CIFAR-100

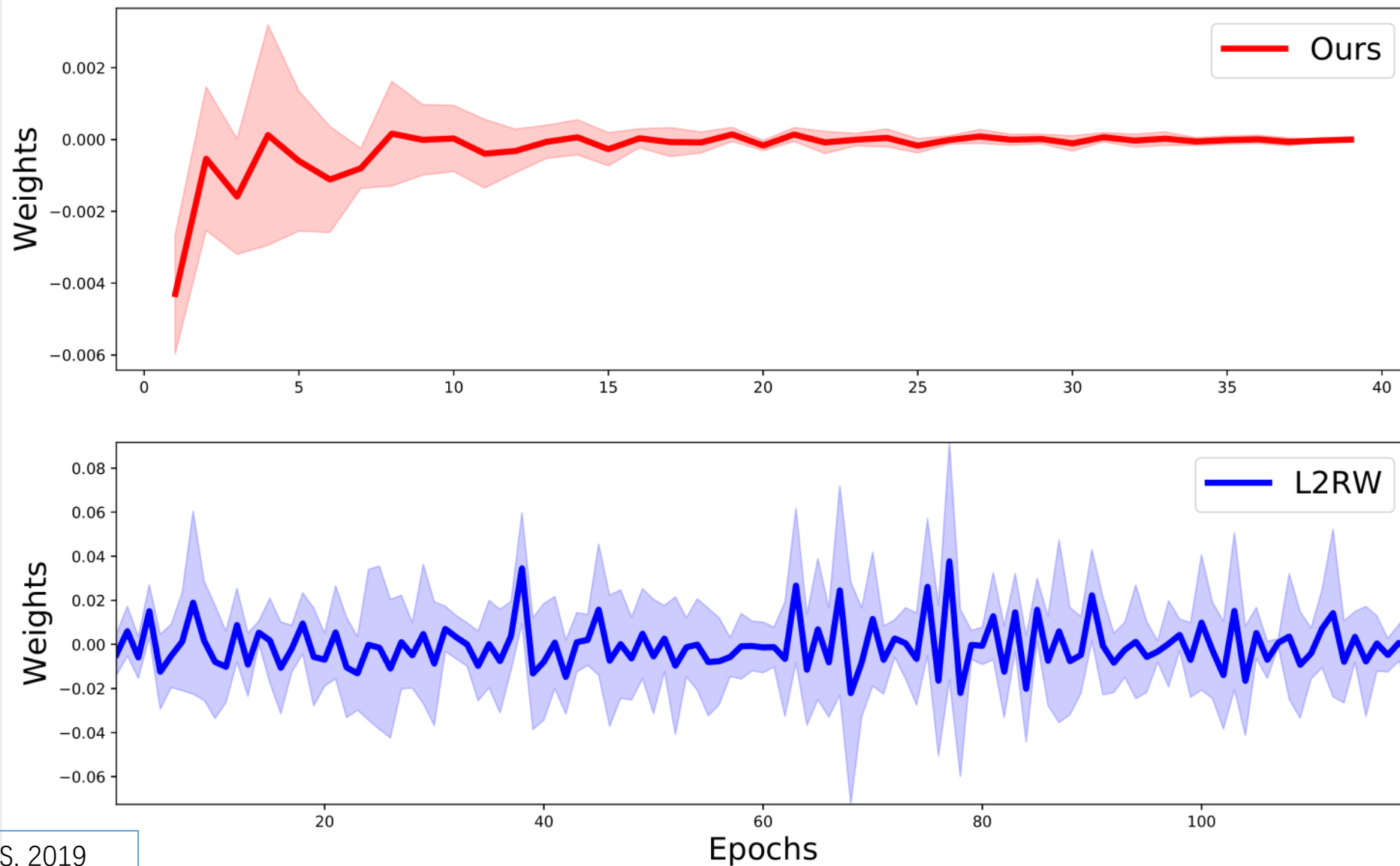
Table 2: Test accuracy comparison on CIFAR-10 and CIFAR-100 of WRN-28-10 with varying noise rates under uniform noise. Mean accuracy ( $\pm$ std) over 5 repetitions are reported (‘—’ means the method fails).

Datasets / Noise Rate	BaseModel	Reed-Hard	S-Model	Self-paced	Focal Loss	Co-teaching	D2L	Fine-tuning	MentorNet	L2RW	GLC	Ours	
CIFAR-10	0%	95.60 $\pm$ 0.22	94.38 $\pm$ 0.14	83.79 $\pm$ 0.11	90.81 $\pm$ 0.34	<b>95.70<math>\pm</math>0.15</b>	88.67 $\pm$ 0.25	94.64 $\pm$ 0.33	<u>95.65<math>\pm</math>0.15</u>	94.35 $\pm$ 0.42	92.38 $\pm$ 0.10	94.30 $\pm$ 0.19	94.52 $\pm$ 0.25
	40%	68.07 $\pm$ 1.23	81.26 $\pm$ 0.51	79.58 $\pm$ 0.33	86.41 $\pm$ 0.29	75.96 $\pm$ 1.31	74.81 $\pm$ 0.34	85.60 $\pm$ 0.13	80.47 $\pm$ 0.25	87.33 $\pm$ 0.22	86.92 $\pm$ 0.19	<u>88.28<math>\pm</math>0.03</u>	<b>89.27<math>\pm</math>0.28</b>
	60%	53.12 $\pm$ 3.03	73.53 $\pm$ 1.54	—	53.10 $\pm$ 1.78	51.87 $\pm$ 1.19	73.06 $\pm$ 0.25	68.02 $\pm$ 0.41	78.75 $\pm$ 2.40	82.80 $\pm$ 1.35	82.24 $\pm$ 0.36	<u>83.49<math>\pm</math>0.24</u>	<b>84.07<math>\pm</math>0.33</b>
CIFAR-100	0%	79.95 $\pm$ 1.26	64.45 $\pm$ 1.02	52.86 $\pm$ 0.99	59.79 $\pm$ 0.46	<b>81.04<math>\pm</math>0.24</b>	61.80 $\pm$ 0.25	66.17 $\pm$ 1.42	<u>80.88<math>\pm</math>0.21</u>	73.26 $\pm$ 1.23	72.99 $\pm$ 0.58	73.75 $\pm$ 0.51	78.76 $\pm$ 0.24
	40%	51.11 $\pm$ 0.42	51.27 $\pm$ 1.18	42.12 $\pm$ 0.99	46.31 $\pm$ 2.45	51.19 $\pm$ 0.46	46.20 $\pm$ 0.15	52.10 $\pm$ 0.97	52.49 $\pm$ 0.74	<u>61.39<math>\pm</math>3.99</u>	60.79 $\pm$ 0.91	61.31 $\pm$ 0.22	<b>67.73<math>\pm</math>0.26</b>
	60%	30.92 $\pm$ 0.33	26.95 $\pm$ 0.98	—	19.08 $\pm$ 0.57	27.70 $\pm$ 3.77	35.67 $\pm$ 1.25	41.11 $\pm$ 0.30	38.16 $\pm$ 0.38	36.87 $\pm$ 1.47	48.15 $\pm$ 0.34	<u>50.81<math>\pm</math>1.00</u>	<b>58.75<math>\pm</math>0.11</b>

Table 3: Test accuracy comparison on CIFAR-10 and CIFAR-100 of ResNet-32 with varying noise rates under flip noise.

Datasets / Noise Rate	BaseModel	Reed-Hard	S-Model	Self-paced	Focal Loss	Co-teaching	D2L	Fine-tuning	MentorNet	L2RW	GLC	Ours	
CIFAR-10	0%	<b>92.89<math>\pm</math>0.32</b>	92.31 $\pm$ 0.25	83.61 $\pm$ 0.13	88.52 $\pm$ 0.21	93.03 $\pm$ 0.16	89.87 $\pm$ 0.10	92.02 $\pm$ 0.14	<b>93.23<math>\pm</math>0.23</b>	92.13 $\pm$ 0.30	89.25 $\pm$ 0.37	91.02 $\pm$ 0.20	92.04 $\pm$ 0.15
	20%	76.83 $\pm$ 2.30	88.28 $\pm$ 0.36	79.25 $\pm$ 0.30	87.03 $\pm$ 0.34	86.45 $\pm$ 0.19	82.83 $\pm$ 0.85	87.66 $\pm$ 0.40	82.47 $\pm$ 3.64	86.36 $\pm$ 0.31	87.86 $\pm$ 0.36	<u>89.68<math>\pm</math>0.33</u>	<b>90.33<math>\pm</math>0.61</b>
	40%	70.77 $\pm$ 2.31	81.06 $\pm$ 0.76	75.73 $\pm$ 0.32	81.63 $\pm$ 0.52	80.45 $\pm$ 0.97	75.41 $\pm$ 0.21	83.89 $\pm$ 0.46	74.07 $\pm$ 1.56	81.76 $\pm$ 0.28	85.66 $\pm$ 0.51	<u>88.92<math>\pm</math>0.24</u>	<b>87.54<math>\pm</math>0.23</b>
CIFAR-100	0%	<b>70.50<math>\pm</math>0.12</b>	69.02 $\pm$ 0.32	51.46 $\pm$ 0.20	67.55 $\pm$ 0.27	70.02 $\pm$ 0.53	63.31 $\pm$ 0.05	68.11 $\pm$ 0.26	<b>70.72<math>\pm</math>0.22</b>	70.24 $\pm$ 0.21	64.11 $\pm$ 1.09	65.42 $\pm$ 0.23	70.11 $\pm$ 0.33
	20%	50.86 $\pm$ 0.27	60.27 $\pm$ 0.76	45.45 $\pm$ 0.25	63.63 $\pm$ 0.30	61.87 $\pm$ 0.30	54.13 $\pm$ 0.55	63.48 $\pm$ 0.53	56.98 $\pm$ 0.50	61.97 $\pm$ 0.47	57.47 $\pm$ 1.16	<u>63.07<math>\pm</math>0.53</u>	<b>64.22<math>\pm</math>0.28</b>
	40%	43.01 $\pm$ 1.16	50.40 $\pm$ 1.01	43.81 $\pm$ 0.15	53.51 $\pm$ 0.53	54.13 $\pm$ 0.40	44.85 $\pm$ 0.81	51.83 $\pm$ 0.33	46.37 $\pm$ 0.25	52.66 $\pm$ 0.56	50.98 $\pm$ 1.55	<u>62.22<math>\pm</math>0.62</u>	<b>58.64<math>\pm</math>0.47</b>

# Stable analysis of Meta-Weight-Net

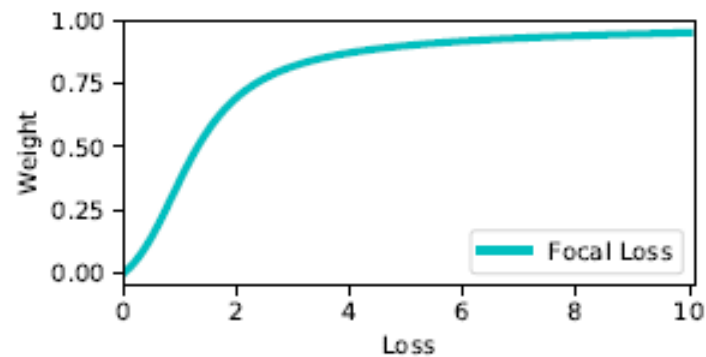


# Real Data Experiment

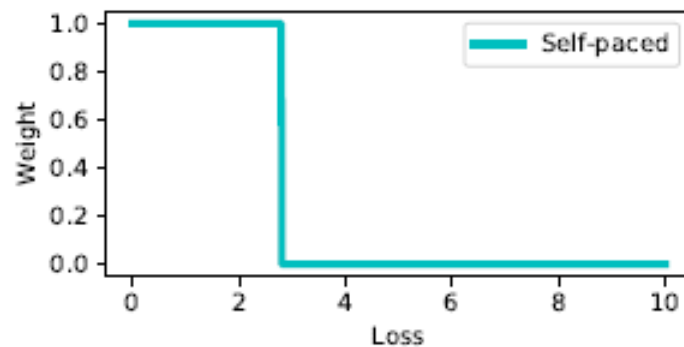
Table 4: Classification accuracy (%) of all competing methods on the Clothing1M test set.

#	Method	Accuracy	#	Method	Accuracy
1	Cross Entropy	68.94	5	Joint Optimization [66]	72.23
2	Bootstrapping [58]	69.12	6	LCCN [67]	73.07
3	Forward [65]	69.84	7	MLNT [68]	73.47
4	S-adaptation [15]	70.36	8	Ours	73.72

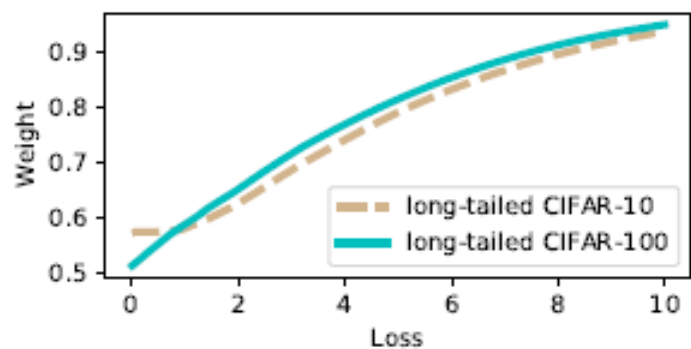
# Insight: Adaptively Learn the Weight Function



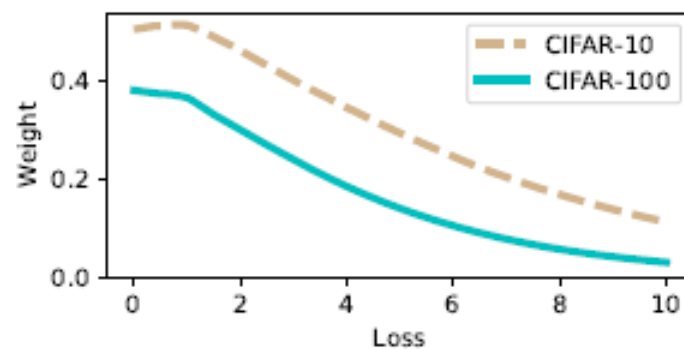
(a) Focal loss case



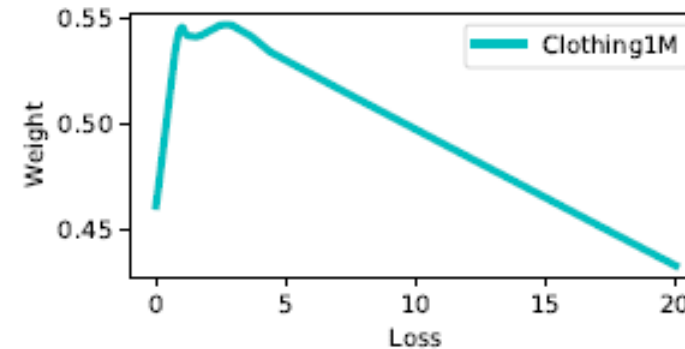
(b) Self-paced case



(c) Class imbalance case



(d) Corrupter labels case

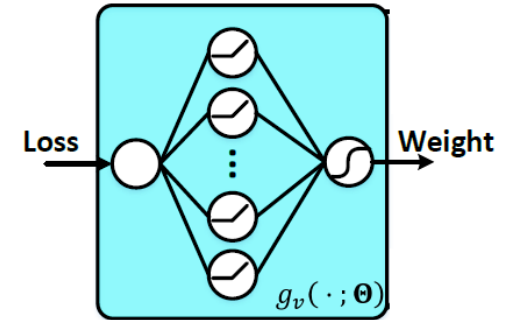


(e) Real Clothing1M dataset

# Future research

◆ General hyper-parameter learning (meta-learner designing)

◆ More amelioration to the Meta-Weight-Net



◆ Extension to other semi/weakly-supervised learning problems

◆ Multi-view learning, ensemble learning, domain adaptation

**Jun Shu, Qian Zhao, Keyu Chen, Zongben Xu, Deyu Meng.  
Learning Adaptive Loss for Robust Learning with Noisy Labels.  
arXiv:2002.06482, 2020.**

**Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben  
Xu, Deyu Meng. Meta-Weight-Net: Learning an Explicit  
Mapping For Sample Weighting. NeurIPS, 2019.**

