IOFM: Using the Interpolation Technique on the Over-Fitted Models to Identify Clean-Annotated Samples

Dongha Kim, Yongchan Choi, Kunwoong Kim, Ilsang Ohn, Yongdai Kim

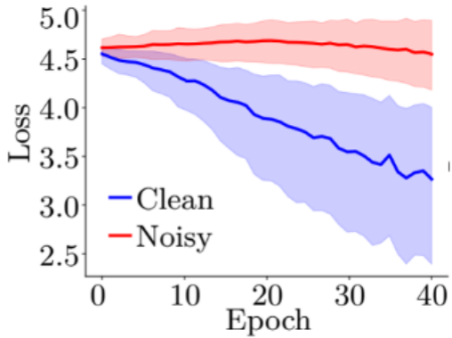
AAAI 2024

Department of Statistics, Seoul National University
Presented by Sangmoon Han

2025-12-23

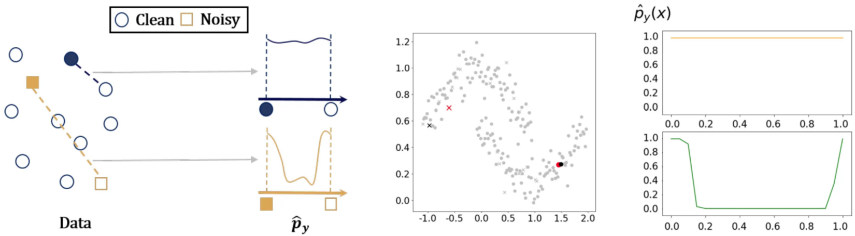
Table of Contents

The memorization effect



- The memorization effect(**ME**) is a phenomenon that deep neural networks (DNNs) memorize clean data before noisy ones.
- Recent SOTA algorithms for handling noisy label problems are based on the **ME**.
- However, the **ME** may not occur in certain scenarios,
  1. labels have an imbalanced distribution.
  2. labels are heavily contaminzted.

Overview of IOFM



- Let's consider an over-fitted DNN and its feature space, i.e., the map of the highest hidden layer.
- For a given training sample $(\mathbf{x}_*, y_*)$, $\mathbf{x}_*$ is located close to other inputs sharing the label y_* regardless of anomalousness of y_* on the feature space.
- When they consider the original input space, the similarity between x_* and its neighbors, **chosen from the feature space**, would be quite different on the original input space depending on whether y_* is clean or noisy.
- Conceptually, The IOFM measures how similar the neighbors of a given data chosen from the feature space are on the input space and decides **the data as clean when the similarity is large.**

Notation

- $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^d$: Observed input vector.
- $y \in [K]$: Observed label, where $[K] = \{1, \ldots, K\}$.
- $y^{\mathrm{gt}} \in [K]$: Ground-truth label, where $[K] = \{1, \ldots, K\}$.

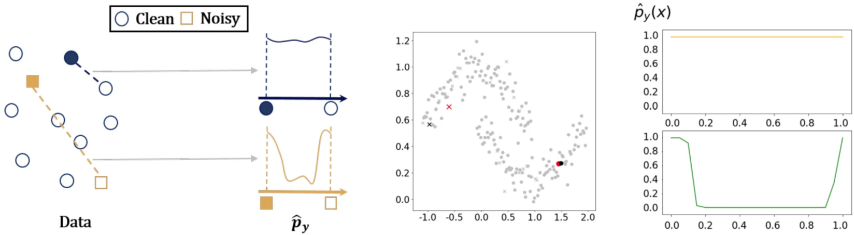
- $(\mathbf{x}, y)$ is cleanly labeled if $y = y^{\mathrm{gt}}$ and noisily labeled if $y \neq y^{\mathrm{gt}}$.
- Let $\mathcal{D}^{\mathrm{tr}} = \{(\mathbf{x}_i, y_i), i \in [n]\}$ be a training data set, and $\mathcal{C}^{\mathrm{tr}} = \left\{(\mathbf{x}, y) \in \mathcal{D}^{\mathrm{tr}} : y = y^{\mathrm{gt}}\right\}$ be the set of clean labeled samples.
- Our goal is to identify the clean labeled subset $\mathcal{C}^{\mathrm{tr}}$ from $\mathcal{D}^{\mathrm{tr}}$ accurately.

Proposed Method: IOFM

- Let $p(\mathbf{x};\theta) : \mathbb{R}^d \to \mathbb{R}^K$ be a discriminative DNN parametrized by θ.
- Also, let $h(\mathbf{x};\theta)$ be the feature vector of $p(\mathbf{x};\theta)$, the output of the DNN's highest hidden layer.
- Furthermore, let $p(\mathbf{x};\hat{\theta})(\hat{p}(\mathbf{x}))$ be a DNN that perfectly memorizes $\mathcal{D}^{\mathrm{tr}}$ and $h(\mathbf{x};\hat{\theta})(\hat{h}(\mathbf{x}))$ be its feature vector.
- For a given training sample $(\mathbf{x}_*, y_*) \in \mathcal{D}^{\mathrm{tr}}$, let $(\mathrm{x}_{\mathrm{nbd}}, y_{\mathrm{nbd}}) \in \mathcal{D}^{\mathrm{tr}}$ be the nearest to $(\mathbf{x}_*, y_*)$ on the feature space, defined by

$$\mathbf{x}_{\mathrm{nbd}} = \underset{\mathbf{x}\in\mathcal{D}^{\mathrm{tr}}\setminus\{\mathbf{x}*\}}{\operatorname{argmin}} \left\| \hat{h}(\mathbf{x}) - \hat{h}\left(\mathbf{x}_*\right) \right\|_2 .$$

Proposed Method: IOFM



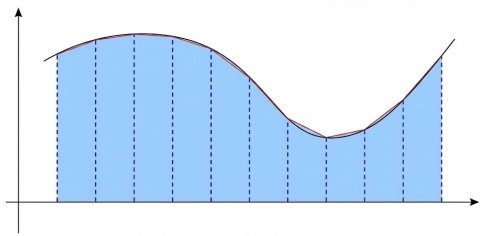
- The authors have noticed a distinct difference in the behavior of $\hat{p}_{y_*}(\mathbf{x})$ depending on its cleanness, which will be utilized for developing the new score.
- That is, the area under $\hat{p}_{y_*}(\mathbf{x})$ over the intervals between $\mathbf{x}_*$ and $\mathbf{x}_{\mathrm{nbd}}$, defined as

$$\int_0^1 \hat{p}_{y_*}\left(\alpha \mathbf{x}_* + (1-\alpha)\mathbf{x}_{\mathrm{nbd}}\right) d\alpha$$

  is large when y_* is clean while it is relatively small when y_* is corrupted.
- Let $\{\mathbf{x}_{\mathrm{nbd},l}\}_{l=1}^{L} \subset \{\mathbf{x}_i : y_i = y_*, i \in [n]\} \setminus \{\mathbf{x}_*\}$ be L neighborhood training inputs of $\mathbf{x}_*$ in the feature space. Then, they proposed the following averaged score

$$\frac{1}{L}\sum_{l=1}^{L}\int_0^1 \hat{p}_{y_*}\left(\alpha \mathbf{x}_* + (1-\alpha)\mathbf{x}_{\mathrm{nbd},l}\right) d\alpha. \qquad (1)$$

Proposed Method: IOFM



- Finally, The authors approximate the integration in (1) by the trapezoidal rule as follows:

$$s^{\mathrm{IOFM}}(\mathbf{x}_*, y_*) = \frac{1}{L}\sum_{l=1}^{L}\sum_{h=1}^{H}\frac{1}{2H}\left(\hat{p}_y\left(\mathbf{x}_{l,h-1}\right) + \hat{p}_y\left(\mathbf{x}_{l,h}\right)\right),$$

where $\mathbf{x}_{l,h} = \frac{H-h}{H}\mathbf{x}_* + \frac{h}{H}\mathbf{x}_{\mathrm{nbd},l}$ and H is the number of trapezoids, to have the IOFM score.

Improvement of IOFM

- **MixUp Loss Function**: To enhance the smoothness of a trained DNN more while keeping memorizing training data, it would be helpful to use [2, Mixup], whose loss function is given as:

$$\underset{(\mathbf{x}_1,y_1),(\mathbf{x}_2,y_2)\sim\mathcal{D}^{\mathrm{tr}}}{\mathbb{E}} \underset{\lambda\sim B(\alpha,\alpha)}{\mathbb{E}} \mathrm{CE}\left(\mathrm{Mix}_\lambda\left(y_1,y_2\right), p\left(\mathrm{Mix}_\lambda\left(\mathbf{x}_1,\mathbf{x}_2\right);\theta\right)\right)$$

- **Use of Multiple IOFM Scores**: During the training process of the DNN until over-fitting, the authors calculate the IOFM scores at different training epochs and take the average for the final score.
- **Computation Time Reduction in Searching Neighbors**: This issue can be addressed by only restricting the neighbor search among a small subset of randomly sampled training data that share the same label.

Experiment

Dataset	CIFAR100					
Imbalance type	Step			Long-tail		
Noise type	symm.		Asymm.	symm.		Asymm.
Noise rate (r)	0.3	0.5	0.2	0.3	0.5	0.2
Loss	0.958(0.698)	0.938(0.632)	0.826(0.674)	0.939(0.670)	0.896(0.609)	0.765(0.667)
Ens-Loss	0.969(0.945)	0.951(0.912)	0.829(0.816)	0.954(0.918)	0.915(0.873)	0.796(0.791)
Margin	0.952(0.689)	0.927(0.626)	0.847(0.684)	0.929(0.660)	0.887(0.603)	0.793(0.676)
AUM	0.966(0.851)	0.947(0.779)	0.871(0.802)	0.951(0.798)	0.911(0.727)	0.827(0.775)
sinIOFM	0.965(0.902)	0.949(0.874)	0.862(0.766)	0.953(0.872)	0.909(0.833)	0.826(0.723)
IOFM	**0.973(0.958)**	**0.958(0.936)**	**0.911(0.910)**	**0.961(0.942)**	**0.927(0.905)**	**0.875(0.874)**

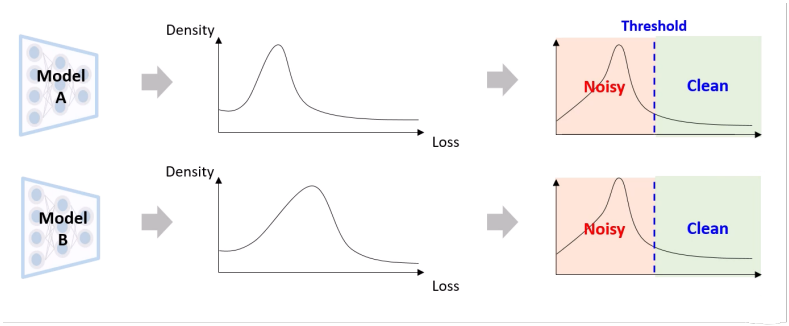
Table 1: Comparison of the clean/noisy classification AUC values on the imbalanced CIFAR100. We list the best and final (in the parentheses) results.

Dataset	CIFAR10			CIFAR100		
Noise type	Symm.		Asymm.	Symm.		Asymm.
Noise rate (r)	0.8	0.9	0.4	0.8	0.9	0.4
Loss	0.919(0.569)	0.836(0.556)	0.933(0.753)	0.847(0.557)	0.708(0.521)	0.621(0.552)
Ens-Loss	0.947(0.895)	0.867(0.763)	0.908(0.901)	0.873(0.704)	0.733(0.660)	0.642(0.638)
Margin	0.918(0.567)	0.834(0.554)	**0.940**(0.756)	0.833(0.553)	0.704(0.520)	0.634(0.553)
AUM	0.948(0.809)	0.869(0.687)	0.927(0.874)	0.874(0.693)	0.734(0.588)	0.677(0.637)
sinIOFM	0.924(0.699)	0.842(0.622)	0.912(0.825)	0.859(0.684)	0.714(0.567)	0.664(0.607)
IOFM	**0.954(0.907)**	**0.887(0.811)**	0.934(**0.921**)	**0.890(0.806)**	**0.746(0.653)**	**0.713(0.712)**

Table 2: Clean/noisy classification AUC results on heavily noisy CIFAR10&100. The best and final (in the parentheses) results are listed.

DivideMix in semi-supervised learning

- The IOFM can be applied to learn deep classification models with noisy labeled data.
- In this paper, the authors consider a combination of the IOFM with the [1, DivideMix], one of the state-of-the-art methods for learning classification models in the presence of noisy labels.
- To combine the IOFM and DivideMix, the authors simply substitute the per-sample losses used in the DivideMix with the IOFM scores.

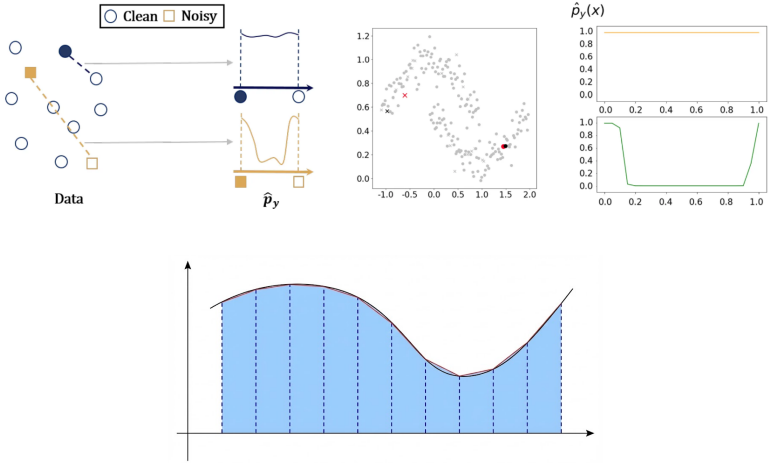


Experiment

Dataset	CIFAR10			
Imbalance type	Step			
Noise type	symm.		Asymm.	
Noise rate (r)	0.3	0.5	0.2	0.4
Cross-Entropy	72.72	68.99	81.19	74.95
Mixup	74.81	65.63	81.97	75.93
DivideMix	85.76	85.16	87.15	78.35
IOFM+DivideMix	**88.19**	**88.07**	**87.35**	**78.73**

Table 3: Comparison of the best test accuracies(%) of various methods on the imbalanced CIFAR10.

Appendix: Proposal



We consider two versions of the IOFM: 1) the original IOFM (IOFM) based on the ensemble of multiple IOFM scores from multiple epochs, and 2) the IOFM based on the single score at the last epoch (sinIOFM). We consider

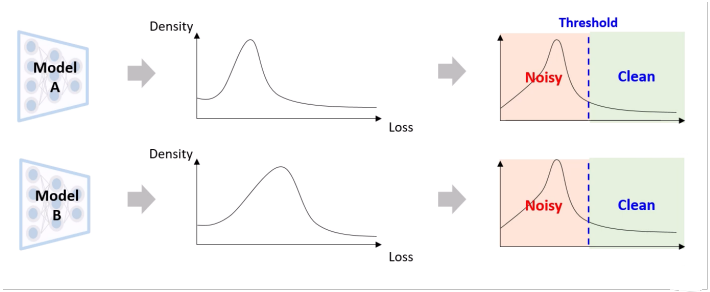
Appendix: Algorithm

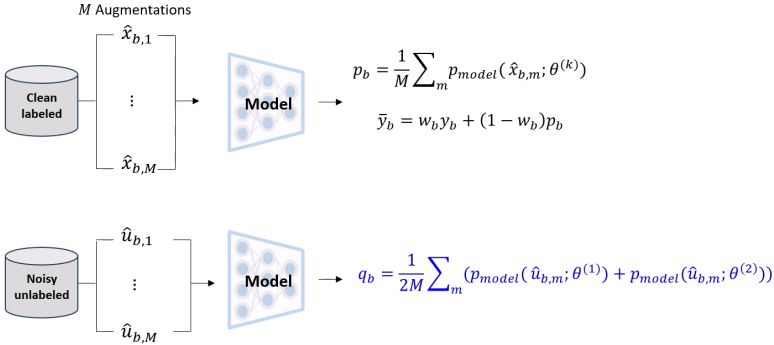
Algorithm 1: IOFM
In practice, we set $(T_1, T_2, L, H) = (150, 10, 10, 3)$.

input Training data: $\mathcal{D}^{\text{tr}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, a prediction model and its feature function: $p(\cdot;\theta)$ and $h(\cdot;\theta)$, an optimizer $\mathcal{O}$, four integers: T_1, T_2, L, and H.

$\mathcal{S}^{\text{ens}} \leftarrow \emptyset$ // Ensemble IOFM score set
for $(ep = 1$ **to** $T_1)$ **do**
  $MixUp(f(\cdot;\theta), \mathcal{D}^{\text{tr}}, \mathcal{O})$ // train $p(\cdot;\theta)$ using MixUp
  if (ep mod T_2 = 0) **then**
    $\mathcal{S}^{\text{tmp}} \leftarrow \emptyset$
    for $(i = 1$ **to** $n)$ **do**
      $s_i \leftarrow s^{\text{IOFM}}(\mathbf{x}_i, y_i)$ //IOFM score of $(\mathbf{x}_i, y_i)$
      $\mathcal{S}^{\text{tmp}} \leftarrow \text{append}(\mathcal{S}^{\text{tmp}}, s_i)$ // append s_i to $\mathcal{S}^{\text{tmp}}$
    end for
    $\mathcal{S}^{\text{ens}} \leftarrow \mathcal{S}^{\text{ens}} + \mathcal{S}^{\text{tmp}}$
  end if
end for
output $\mathcal{S}^{\text{ens}}$

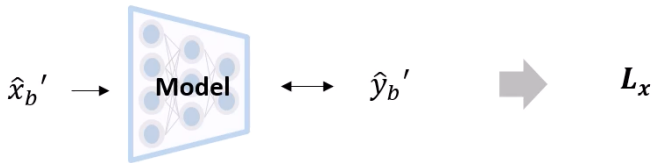
Appendix: Dividemix & Mixmatch

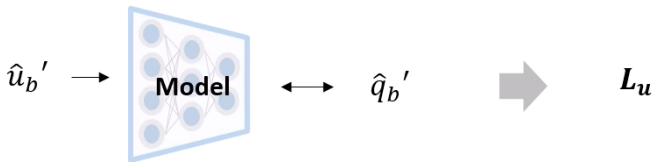




Appendix: Dividemix & Mixmatch

$$\hat{x}_b{}' = \lambda' \hat{x}_1{}' + (1-\lambda')\hat{x}_2{}'$$
$$\hat{y}_b{}' = \lambda' \hat{y}_1{}' + (1-\lambda')\hat{y}_2{}'$$
$$\hat{u}_b{}' = \lambda' \hat{u}_1{}' + (1-\lambda')\hat{u}_2{}'$$
$$\hat{q}_b{}' = \lambda' \hat{q}_1{}' + (1-\lambda')\hat{q}_2{}'$$





References I

Junnan Li, Richard Socher, and Steven C. H. Hoi. *DivideMix: Learning with Noisy Labels as Semi-supervised Learning*. 2020. arXiv: 2002.07394 [cs.CV]. URL: https://arxiv.org/abs/2002.07394.

Hongyi Zhang et al. *mixup: Beyond Empirical Risk Minimization*. 2018. arXiv: 1710.09412 [cs.LG]. URL: https://arxiv.org/abs/1710.09412.