A Taxonomy for Learning with Perturbation and Algorithms

RUJING YAO, Department of Information Resources Management, Nankai University; Center for Applied Mathematics, Tianjin University, China

OU WU*, Center for Applied Mathematics, Tianjin University, China

Weighting strategy prevails in machine learning. For example, a common approach in robust machine learning is to exert low weights on samples which are likely to be noisy or quite hard. This study summarizes another less-explored strategy, namely, perturbation. Various incarnations of perturbation have been utilized but it has not been explicitly revealed. Learning with perturbation is called perturbation learning and a systematic taxonomy is constructed for it in this study. In our taxonomy, learning with perturbation is divided on the basis of the perturbation targets, directions, inference manners, and granularity levels. Many existing learning algorithms including some classical ones can be understood with the constructed taxonomy. Alternatively, these algorithms share the same component, namely, perturbation in their procedures. Furthermore, a family of new learning algorithms can be obtained by varying existing learning algorithms with our taxonomy. Specifically, three concrete new learning algorithms are proposed for robust machine learning. Extensive experiments on image classification and text sentiment analysis verify the effectiveness of the three new algorithms. Learning with perturbation can also be used in other various learning scenarios, such as imbalance learning, clustering, regression, and so on. The source code is available at https://github.com/RujingYao/Learning-with-Perturbation.

Additional Key Words and Phrases: Sample weighting, Perturbation, Robust machine learning, Learning taxonomy.

ACM Reference Format:
Rujing Yao and Ou Wu. 2022. A Taxonomy for Learning with Perturbation and Algorithms. In . ACM, New York, NY, USA, 36 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

In supervised learning, a loss function is defined on the training set, and the training goal is to seek optimal models by minimizing the training loss. According to the degree of training difficulty, samples can be divided into easy, medium, hard, and noisy samples. Generally, easy and medium samples are indispensable and positively influence the training. The whole training procedure can significantly benefit from medium samples if appropriate learning manners are leveraged. However, the whole training procedure is vulnerable to noisy and partial quite hard samples.

A common practice is to introduce the sample weighting strategy if hard and noisy samples exist. Low weights are assigned to noisy and quite hard samples to reduce their negative influences during loss minimization. This strategy usually infers the weights and subsequently conducts training on the basis of the weighted loss [49]. Wang et al. [78] proposed a Bayesian method to infer the sample weights as latent variables. Kumar et al. [37] proposed a self-paced learning (SPL) manner that combines the two steps as a whole by using an added regularizer. Meta-learning [42, 63, 81] is introduced to alternately infer weights and seek model parameters with an additional validation set.

*Corresponding author.

Manuscript submitted to ACM

Various machine learning methods exist that do not rely on the weighting strategy. For example, the classical method support vector machine (SVM) [9] introduces slack variables to address possibly noisy or quite hard samples, and robust clustering [18] introduces additional vectors to cope with noises. However, a unified theory to better explain such methods and subsequently illuminate more novel methods remains lacking. In this study, another less-explored yet widely used paradigm[1], namely, perturbation, is summarized and further investigated. Mathematically, the perturbation strategy actually adds[2] a perturbation term to a feature vector, a logit vector, a loss, etc. Many existing learning methods including some classical ones can be (partially) understood in the point of the learning-with-perturbation view. Learning with perturbation is referred to as perturbation learning in the present paper.

In this study, we conduct a pilot study to construct a theoretical taxonomy for learning with perturbation. Specifically, five perturbation targets, three directions, six inference manners, and four granularity levels are defined. Several existing classical learning methods are used as illustrated examples to demonstrate the reasonableness of our constructed taxonomy, and potential research directions are presented. A close connection can be obtained among these seemingly unrelated methods and new variations of these methods can be naturally obtained. In addition, three concrete perturbation learning algorithms are proposed, namely, logit perturbation with $l1$-regularization (LogPert), mixed positive and negative perturbation (MixPert), and the meta-learning-based MixPert (Meta-MixPert). Last, the three proposed learning algorithms are evaluated on data corpora from image classification and text sentiment classification.

Our main contributions are summarized as follows:

1) A less-explored yet widely used learning scheme, namely, perturbation, is summarized and formalized in this study. A systematic taxonomy is constructed for it, which can establish an intrinsic connection among numerous seemly unrelated machine learning methods. In addition to the noisy-label learning mainly referred in this paper, other learning scenarios, such as imbalance learning, can also benefit from perturbation learning.

2) Several typical learning methods are re-explained with the viewpoint of our constructed taxonomy for learning with perturbation. A close connection can be observed for these methods and new insights can be obtained. Theoretically, various *new* methods can be generated on the basis of introducing the idea of perturbation into existing methods. Sections 5 and 6 present examples.

3) Three concrete new perturbation learning methods are proposed. Experiments on robust learning on several benchmark sets verify their effectiveness compared with several existing classical methods.

The rest of the paper is organized as follows. Section 2 briefly reviews related studies. Section 3 highlights the significance of our summarization for existing studies on learning with perturbation. Section 4 introduces our constructed taxonomy including the construction principles, details, and representative methods. Section 5 describes our proposed three new methods. Section 6 presents the experimental comparison and discussions of our methods, and conclusions are given in Section 7.

2 RELATED WORK

2.1 The Weighting Strategy in Machine Learning

Weighting is a widely used machine learning strategy in at least the following five areas: noise-aware learning [58], curriculum learning [2], crowdsourcing learning [14], cost-sensitive learning [5], and imbalance learning [31]. In noisy-aware and curriculum learning areas, weights are sample-wise; in cost-sensitive learning, weights can be sample-wise, category-wise, or mixed; in imbalance learning, weights are usually category-wise.

[1]One widely studied topic in current literature, namely, adversarial examples, is a special type of perturbation, which is discussed in Section 4.
[2]Weighting actually multiplies a term to a feature vector, a logit vector, a loss, etc.

Intuitively, the weights of medium and partial hard samples are kept or enlarged; and the weights of quite hard samples should be kept or reduced. For example, in Focal loss [45], the weights of easy samples are (relatively) reduced and those of the hard[3] samples are (relatively) enlarged. Most existing studies do not assume the above sample division. Instead, samples are usually divided into easy/non-easy or normal/noisy. For example, in Focal loss and Adaboost [19], the weights of non-easy samples are gradually increased.

In cost-sensitive learning, the weights are associated with misclassified costs. Shen et al. [66] proposed a new cost-sensitive adversarial learning framework to ensure that some special classes are less vulnerable. Indeed, perturbation learning can also be utilized in this scenario. In imbalance learning, categories with lower proportions are negatively affected. Therefore, increasing the weights of samples in the low-proportion categories is a common practice.

The perturbation strategy investigated in this study does not intend to eliminate the weighting strategy. Instead, this study summarizes various existing learning ideas which do not utilize weighting yet. These learning ideas are systematically investigated to attribute to a unified learning paradigm, namely, learning with perturbation. These two strategies can be mutually beneficial[4]. Theoretically, each concrete weighting-based learning method may correspond to a concrete perturbation-based learning method. A solid and deep investigation for the weighting strategy in machine learning will significantly benefit perturbation learning.

2.2 Noise-aware Machine Learning

This study investigates perturbation mainly in learning with noisy labels. The weighting strategy is prevailing in this area. There exist two common technical solutions.

In the first solution, noise detection is performed and noisy samples may be assigned lower weights in the successive model training. Koh and Liang [25] defined an influence function to measure the impact of each sample on the model training. Samples with higher influence scores are more likely to be noisy. Huang et al. [32] conducted a cyclical pre-training strategy and recorded the training losses for each sample in the whole cycles. The samples with higher average training losses are more likely to be noisy.

In the second solution, an end-to-end procedure is leveraged to construct a robust model. Reed et al. [62] proposed a Bootstrapping loss to reduce the negative impact of samples which may be noisy. Goldberger and Ben-Reuven [22] designed a noise adaptation layer to model the relationship between labels that may be noisy and true latent labels.

More specific methods along with the aforementioned two solutions can be found in a recent survey [70]. Perturbation can replace weighting in both above solutions. In this study, only the second solution is referred.

2.3 Robust Machine Learning

A formal definition for robust machine learning does not exist at present. There are two typical learning scenarios for robust machine learning. The first scenario refers to the robustness of a learning process, while the second scenario refers to the robustness of a trained model. In the first scenario, a robust learning method should cope well with training data that may be noisy [70, 88], imbalance [35], few-shot [11, 80], etc. In the second scenario, a robust trained model should cope well with adversarial attacks [93]. Both scenarios receive much and increasing attention in recent years. Both the weighting and the perturbation strategies are widely-used in the first scenario, whereas only the perturbation strategy is mainly utilized in the second scenario.

[3] In fact, if the weights of quite hard samples are reduced, the performance will be increased [41].

[4] For example, a sample-level weighting method (e.g., Focal loss) can be transformed into a category-level weighting method (e.g., replace the sample-level prediction y_i with the category-level average y_c) inspired by our taxonomy for learning with perturbation.

3 SIGNIFICANCE FOR THE SUMMARY OF LEARNING WITH PERTURBATION

Pure data manipulation without modifying the structure of involved DNNs has been proven to be effective in the training of DNNs. One main data manipulation strategy is sample weighting, as described in Section 2.1. Meanwhile, there are numerous other pure data manipulation strategies in previous literature. For instance, data augmentation and the perturbation of logit vectors have been widely used in imbalanced learning and noisy-label learning. Moreover, in some learning scenarios such as robust learning, the adversarial perturbations of samples or features are quite useful, whereas sample weighting is rarely employed.

An interesting and meaningful question arises: can a clear roadmap for other data manipulation methods, apart from weighting, be established from a new perspective? To address this, a taxonomy for learning with perturbation is summarized in this study. The subsequent section will demonstrate that a large number of learning methods which are derived from distinct heuristic motivations or theoretical inspirations actually perturb data in training. The construction of such a taxonomy is valuable in the following aspects:

- Connecting existing methods. Various sample weighting methods can be easily unified mathematically, differing mainly in the ways they calculate weights. However, numerous data manipulation methods, apart from sample weighting, are challenging to connect directly. To the best of our knowledge, no study has attempted to arrange such tremendous methods into a unified framework. This study constructs a taxonomy for a significant portion of these tremendous methods, which can naturally build a connection among them. The connection among the seemly unrelated methods can facilitate a better understanding and intersection of these methods. We also envision a more fundamental and deep theoretical analysis of perturbation learning based on our taxonomy.
- Promoting the importance of pure data manipulation. Our constructed taxonomy for learning with perturbation highlights a widely employed yet rarely mentioned strategy, namely, perturbation. Both perturbation and weighting cover the majority of application scenarios in deep learning. Therefore, our summary study will further demonstrate the value and importance of pure data manipulation for deep learning, which may attract more attention in both academical and industrial communities.
- Inspiring new methods and paradigms. As the application scenarios of perturbation and weighting highly overlap, their combination may yield more powerful data manipulation techniques. In addition, in terms of mathematical forms, perturbation is more flexible than weighting. Therefore, it is possible to develop more sophisticated perturbation learning methods. Section 5 provides three illustrative examples of new learning with perturbation methods. Moreover, a data manipulation agent can be designed to automatically leverage data weighting and perturbation operators on the training data of a learning task.

The next section will introduce our constructed taxonomy as well as representative methods for each division.

4 OUR CONSTRUCTED TAXONOMY

This section firstly introduces our principles for the construction of our taxonomy. Each division of the taxonomy is then elaborated in detail. Finally, several potential research directions are presented.

4.1 Principles

Perturbation can be used in many learning scenarios. This section leverages classification as the illustrative example. Given a training set $S = \{x_i, y_i\}$, $i = 1, \ldots, N$, where x_i is the i-th sample, and $y_i \in \{1, \ldots, c, \ldots, C\}$ is its categorical

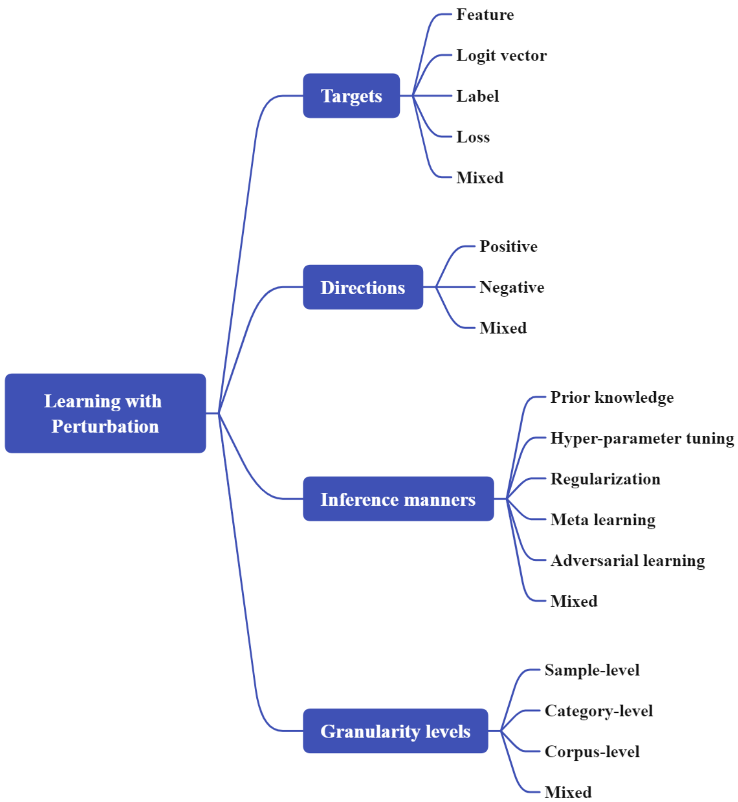

Fig. 1. Taxonomy of learning with perturbation.

label. In a standard supervised deep learning context, let u_i be the logit vector for x_i output using a deep neural network. The training loss can be written as follows:

$$\mathcal{L} = \sum\nolimits_i l(\mathbb{S}(f(x_i)), y_i) = \sum\nolimits_i l(\mathbb{S}(u_i), y_i), \tag{1}$$

where $\mathbb{S}(\cdot)$ transforms the logit vector u_i into a soft label p_i, $f(\cdot)$ represents a deep neural network, and $u_i = f(x_i)$.

Our study is firstly motivated by a widely used pure data-oriented technique, namely, weighting. In the weighting strategy, the loss function is usually defined as follows:

$$\mathcal{L} = \sum\nolimits_i w_i \cdot l(\mathbb{S}(u_i), y_i), \tag{2}$$

where w_i is the weight associated with the sample x_i. In terms of mathematical computation, "weighting" relies on the multiplication operation, whereas "perturbation" relies on adding operation. Let $\mathbf{v}$ be a variable. The perturbation for $\mathbf{v}$ means the following calculation:

$$\mathbf{v} = \mathbf{v} + \Delta\mathbf{v}, \tag{3}$$

where $\mathbf{v}$ is the perturbation. Our taxonomy is constructed according to what to perturb, the direct outcome, and how to infer the perturbation $\Delta\mathbf{v}$, which are detailed as follows:

- What to perturb. In data weighting, weights are mainly applied to the loss function. In data perturbation, there are more choices for the targets to be perturbed. Organizing different perturbation methods according to the targets facilitates the comparisons and mutual inspiration among these methods. In addition, the granularity is also about what to perturb. Therefore, both the target and its granularity are considered in our taxonomy.
- The direct outcome. It is difficult to summarize the outcomes of different perturbation learning methods into a concise division. Note that one direct outcome is the loss variation. Taking noisy-label learning as an example, a perturbation can be utilized to reduce the loss of a possibly noisy sample, and thus the negative influence of this sample will be reduced. Contrarily, when the influence should be increased for a sample, a perturbation which will increase the loss can be utilized[5]. Therefore, the direction of loss variation incurred by perturbation is chosen as one dimension.
- How to infer. This division is crucial, as the determination of the data perturbation is not a trivial task. Our arrangement along this dimension may shed light on exploring new and more effective perturbation inference methodologies.

According to the three principles listed above, our constructed taxonomy of learning with perturbation is depicted in Fig. 1 encompassing four split items[6], namely, targets, directions, inference manners, and granularity levels. As an initial attempt for the construction of such a taxonomy, it is challenging to ensure that these four items are exhaustive. For example, perturbation can also be divided into static and dynamic. In static perturbation, perturbations remain unchanged in training, whereas in dynamic perturbation, they are changed in training. Nevertheless, this split plays a trivial role in the understanding of existing methods in current stage, so it is not included in our current taxonomy. This section introduces each item in the taxonomy.

4.2 Perturbation Targets

The perturbation target in this study denotes the variable which is designed to add a perturbation for each sample in DNN training. Eq. (1) contains four different types of variables for each sample, namely, raw feature x_i, logit vector u_i, label y_i, and sample loss l_i (=$l(\mathbb{S}(u_i), y_i)$). Therefore, perturbation targets can be further divided into four categories, namely, feature, logit vector, label, and loss.

4.2.1 *Sub-categories.*

(1) **Feature perturbation**. In this kind of perturbation, the raw feature vector (x_i) or transformed feature vector (e.g., dense feature output by the involved DNN) of each sample can have a perturbation vector (Δx_i). Eq. (1) becomes

$$\mathcal{L} = \sum\nolimits_i l(\mathbb{S}(f(x_i+\Delta x_i)), y_i) = \sum\nolimits_i l(\mathbb{S}(u_i'), y_i). \tag{4}$$

The perturbation vectors for each training sample are not set freely. Instead, they are inferred according to several manners introduced in Section 4.4. Here we provide a simple example to illustrate the usefulness of feature perturbation. Suppose there is a training sample x' with a wrong label. In sample weighting, a small weight can be assigned to this sample to reduce its negative influence in training. Let m be the center of the category of x'. Ideally, if x' is perturbed by $\Delta x' = m - x'$, then the gradient for x' will become quite small. Consequently, the negative impact of x' is also completely reduced. Indeed, the usefulness of feature perturbation is not restricted in noisy-label learning. More details will be introduced in the rest of this paper.

[5]Section 4.3.2 provides a theoretical explanation for this variation.
[6]There is no survey on sample weighting. These four terms can also be used for sample weighting.

(2) **Logit perturbation**. In this kind of perturbation, the logit vector (u_i) of each sample can have a perturbation vector (Δu_i). Eq. (1) becomes

$$\mathcal{L} = \sum\nolimits_i l(\mathbb{S}(u_i+\Delta u_i), y_i). \tag{5}$$

Likewise, Δu_i is not set freely. Compared with feature perturbation, logit perturbation receives little attention. Nevertheless, it can play similar roles with feature perturbation. Taking the illustrative task described in feature perturbation as an example, let u' and m_{log} be the logit vectors of x' and the center vector m, respectively. If u' is perturbed by $\Delta u = m_{log} - u'$, then noisy samples can also be effectively processed.

(3) **Label perturbation**. In this kind of perturbation, the label (y_i) of each sample can have a perturbation label (Δy_i). Let p_i = softmax(u_i). Eq. (1) becomes

$$\begin{aligned} &\text{(i)} \quad \mathcal{L} = \textstyle\sum_i l(p_i, y_i+\Delta y_i) \quad \text{or} \\ &\text{(ii)} \quad \mathcal{L} = \textstyle\sum_i l(p_i+\Delta y_i, y_i). \end{aligned} \tag{6}$$

In Eq. (6-i), Δy_i is added to the true label y_i, while in (ii) Δy_i is added to the predicted label p_i. Considering that labels after perturbation should be a (soft) label, Δy_i should satisfy the following requirements:

$$\sum\nolimits_c \Delta y_{ic} = 0, y_{ic} + \Delta y_{ic} \geq 0 \quad \text{or} \quad p_{ic} + \Delta y_{ic} \geq 0. \tag{7}$$

Indeed, several classical label perturbation methods are usually utilized in noisy label learning.

(4) **Loss perturbation**. In this kind of perturbation, the loss of each sample can have a perturbation loss (Δl_i). Eq. (1) becomes

$$\mathcal{L} = \sum\nolimits_i l(\mathbb{S}(u_i), y_i)+\Delta l_i. \tag{8}$$

(5) **Mix-target perturbation**. In this kind of perturbation, two or more of the aforementioned targets can have their perturbation terms, simultaneously. For example, when both feature and label perturbation are utilized, Eq. (3) becomes

$$\mathcal{L} = \sum\nolimits_i l(\mathbb{S}(f(x_i+\Delta x_i)), y_i + \Delta y_i), \tag{9}$$

where Δx_i and Δy_i are the feature and label perturbations, respectively[7].

The effectiveness of methods in each sub-category has been verified in most typical learning scenarios (e.g., standard learning, noisy-label learning, and imbalanced learning). Therefore, it is inappropriate to conclude which category is absolutely superior to others in terms of learning performance. Nevertheless, in most cases, it is relatively easy to determine the order of the four categories in terms of computational complexity, i.e., feature perturbation ≥ logit perturbation ≥ label perturbation ≈ loss perturbation. There may exist other perturbation candidates, such as view, structure (e.g., adjacency matrix in GCN), word embedding, and gradient, which will be explored in our future work.

4.2.2 Representative methods.

This part discusses a few representative methods in terms of the abovementioned subcategories. The first is robust clustering (RC) [18]. Let m_c be the cluster center of the c-th cluster. Let ω_{ic} ($\in \{0, 1\}$) denote whether x_i belongs to the c-th cluster. The optimization form of conventional data clustering can be written as follows:

$$\min_{\{m_c\},\{\omega_{ic}\}} \sum\nolimits_i \sum\nolimits_c \omega_{ic} \left\|x_i - m_c\right\|_2^2. \tag{10}$$

[7] Lee et al. [39] combine adversarial training and label smoothing, which can be considered as mix-target (feature and label) perturbation.

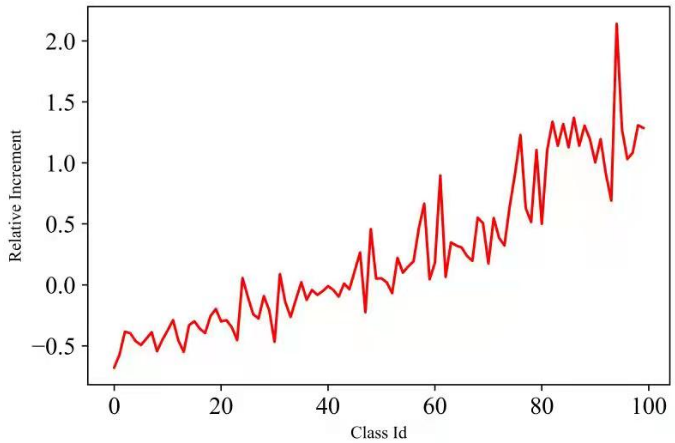

Fig. 2. The relative loss increment ($(l' - l)/l$) for LA. Head categories are in the left and tail ones are in the right. The losses of head categories are mainly decreased, while those of tail ones are increased.

Given that outlier samples may exist, Foreo et al. [18] introduced sample-level feature perturbation (denoted as o_i for x_i) with $l2$-regularization. Then robust clustering is formalized as following:

$$\min_{\{m_c\},\{\omega_{ic}\},\{o_i\}} \sum_i \sum_c \omega_{ic} \left(\|(x_i + o_i) - m_c\|_2^2 + \lambda \|o_i\|_2 \right), \tag{11}$$

Obviously, robust clustering belongs to feature perturbation.

Another typical example is logit adjustment [55], which is particularly designed for imbalanced learning. In a multi-category classification problem, let π_c be the proportion of the training samples in the c-th category. Let $\mathbf{g} = [g(\pi_1), \ldots, g(\pi_C)]$, $g(\pi_c) = \tau \log(\pi_c)$ $(\tau > 0)$. Obviously, $g(\cdot)$ is an increasing function.

Menon et al. [55] defined $\mathbf{g}$ as the logit perturbation vector and then the new cross entropy loss becomes

$$\mathcal{L} = -\sum_i \log \frac{e^{u_{i,y_i} + \tau \log \pi_{y_i}}}{\sum_c e^{u_{i,c} + \tau \log \pi_c}}. \tag{12}$$

In this loss, logit perturbations for each sample are equal.

Label smoothing is a classical noisy-label learning methods which has been proven to be effective in noisy-label learning. It is actually a type of sample-level label perturbation. Its perturbation term for a sample (x_i, y_i) is defined as follows:

$$\Delta y_i = \lambda(I/C - y_i), \tag{13}$$

where I is a C-dimensional vector and each element is equal to 1.

Knowledge distillation is widely used in many deep learning tasks [28]. In knowledge distillation, there are two deep neural networks called teacher and student, respectively. The output of the teacher model for x_i is

$$q_i = \text{softmax}(z_i/T), \tag{14}$$

where z_i is the logit vector from the teacher model and T is the temperature. q_i can be viewed as a prior knowledge about the label perturbation for the student model. Then according to Eq. (7), the training loss of the student model with label perturbation becomes

$$\mathcal{L} = \sum_i l(p_i, y_i) + \lambda(l(p_i, q_i) - l(p_i, p'_i)), \tag{15}$$

where p'_i = softmax(u_i/T). Eq. (15) is exactly the loss function of knowledge distillation. Knowledge distillation also belongs to label perturbation.

SVM [9] is one of the most classical shallow learning methods. It is based on the following hinge loss:

$$l_i = \max(0, 1 - y_i(\mathrm{w}^T x_i + b)). \tag{16}$$

To reduce the negative contributions of noisy or quite hard samples, the loss can be perturbed as follows:

$$\begin{aligned} l'_i &= \max(0, l_i - \xi_i) \\ &= \max(0, 1 - y_i(\mathbf{w}^{\mathrm{T}} x_i + b) - \xi_i) \qquad (\xi_i \geq 0) \end{aligned}, \tag{17}$$

where ξ_i is a variable for perturbation. Then the whole training loss with max margin and $l1$-norm for ξ_i becomes

$$\mathcal{L} = \frac{1}{2}\|\mathbf{w}\|^2 + \sum_i l'_i + \lambda|\xi_i| \qquad (\xi_i \geq 0). \tag{18}$$

The minimization of Eq. (18) equals to the following optimization problem:

$$\begin{aligned} &\min_{\mathbf{w}, b, \{\xi_i\}} \ \tfrac{1}{2}\|\mathbf{w}\|^2 + \lambda \textstyle\sum_i \xi_i \\ &\text{s.t.} \quad 1 - y_i(\mathbf{w}^T x_i + b) - \xi_i \leq 0 \\ &\qquad\quad \xi_i \geq 0, i = 1, \cdots, N \end{aligned}, \tag{19}$$

which is the standard form of SVM (without kernel). Alternatively, the slack variable can be seen as a loss perturbation for SVM. Naturally, other types of perturbation (e.g., label perturbation) can be considered in SVM[8].

Pereyra et al. [61] perturbed the original loss by adding a hinge function for the confidence penalty, which is defined as follows:

$$\mathcal{L} = -\sum_i l(p_i, y_i) - \beta \max(0, \tau - H(p_i)), \tag{20}$$

where β and τ are two hyper-parameters; $H(p_i)$ is information entropy of the prediction p_i, which measures the confidence of the prediction by the current model.

4.3 Perturbation Directions

4.3.1 Sub-categories.

Perturbation direction in this study denotes the loss increment or decrement after perturbation. There are two directions according to the loss variations.

(1) **Positive perturbation**. If the perturbation reduces the loss, then it is called positive perturbation. The following representative methods will indicate that positive perturbation is usually employed to reduce the influence of noisy and quite hard samples during training.

(2) **Negative perturbation**. If the perturbation increases the loss, then it is called negative perturbation. Negative perturbation can enhance the impact of the perturbed samples during training [9].

[8]We conjecture that label perturbation-based SVM may exist in the literature.

[9]As negative perturbation can explicitly or implicitly perform data augmentation, it can enhance the impact of samples including both easy and hard ones.

(3) **Mix-direction perturbation**. If the perturbation increases the losses of some training samples and decreases the losses of others simultaneously, then it is called mix-direction perturbation. The logit adjustment method actually leverages this type of perturbation, which will be discussed in Section 4.3.3.

4.3.2 *Comparison for positive and negative perturbations.*

The two directions are opposite to each other. Nevertheless, both directions have been explored in previous literature, and experiments have verified their effectiveness. We first compare them from a regularization perspective. Let $p_i = \mathbb{S}(u_i)$ represent the softmax output be the current model. Taking logit perturbation as an example, the loss in Eq. (5) can be expanded using the first-order Taylor expansion as follows:

$$\ell(\mathbb{S}(u_i+\Delta u_i), y_i) \approx \ell(p_i, y_i) + (\frac{\partial \ell}{\partial \boldsymbol{u}})^\top \Delta \boldsymbol{u} = \ell(\boldsymbol{p_i}, \boldsymbol{y_i}) + (p_i - y_i)^\top \Delta \boldsymbol{u}. \tag{21}$$

In negative augmentation, $(p_i - y_i)^\top \Delta \boldsymbol{u} > 0$, meaning that negative augmentation can be viewed as adding a regularization term to the original loss $\ell(p_i, y_i)$. There have been a lot of studies revealing that methods such as perturbation using Gaussian noise can inherently provide a regularization effect [16]. Regularization is typically utilized to prevent overfitting. In positive perturbation, $(p_i - y_i)^\top \Delta \boldsymbol{u} < 0$, which can be viewed as adding an anti-regularization term to the original loss. The concept of anti-regularization has been investigated in previous studies [12, 24, 38] and is typically utilized in learning cases when over-regularization may occur. Over-regularization means that excessive regularization is applied. Taking ridge regression as an example, its objective function is $l(X, Y; w) + \lambda ||w||_2^2$. A large value of λ will result in over-regularization. If $\lambda \to +\infty$, then $w \to \mathbf{0}$, resulting in underfitting. In other words, positive perturbation can prevent underfitting, while negative perturbation can prevent overfitting.

Both directions are useful for certain learning scenarios. We take the learning under feature noise as an illustrative example. Given a binary training dataset D, assuming it contains a certain proportion (denoted as p) of feature noise, which is exactly equivalence to an appropriate regularization. In other words, such a proportion of feature noise is useful. Therefore, if the proportion of feature noise for class '+1' is increased to $2p$ and that for class '-1' is decreased to $0.5p$, then positive feature perturbation should be exerted on samples of class '+1', and negative feature perturbation should be exerted on samples of class '-1'. This example illustrates the necessity for both directions as well as their mixture.

We then provide a statistical view to compare the two perturbation directions. It is quite challenging to conduct universal theoretical analysis for arbitrary distributions. Following some relevant theoretical studies [33, 85], we design a simple learning case with Gaussian distributions. The binary classification setting established for the theoretical analysis in [86] is adopted. The data is from two classes $\mathcal{Y} = \{-1, +1\}$ and the data from each class follows a mixture of two Gaussian distributions. In class '+1', its two distributions are centered on $\mathbf{2.5}\boldsymbol{\theta}$ and $\boldsymbol{\theta}$, respectively; in class '-1', its two distributions are centered on $\mathbf{-2.5}\boldsymbol{\theta}$ and $-\boldsymbol{\theta}$, respectively. The overall data distribution follows

$$\begin{aligned} &y \overset{u.a.r}{\sim} \{-1, +1\}, \quad \boldsymbol{\theta} = [\eta, \ldots, \eta]^T \in R^d, \\ &\boldsymbol{x} \sim \begin{cases} \gamma_1 \mathcal{N}\left(\boldsymbol{\theta}, \sigma_+^2 \boldsymbol{I}\right) + (1-\gamma_1)\mathcal{N}\left(2.5\boldsymbol{\theta}, \sigma_+^2 \boldsymbol{I}\right), & \text{if } y = +1 \\ \gamma_2 \mathcal{N}\left(-\boldsymbol{\theta}, \sigma_-^2 \boldsymbol{I}\right) + (1-\gamma_2)\mathcal{N}\left(-2.5\boldsymbol{\theta}, \sigma_-^2 \boldsymbol{I}\right), & \text{if } y = -1 \end{cases}, \end{aligned} \tag{22}$$

where γ_1 and γ_2 are two independent discrete random variables; I is an identity matrix; and σ_+^2 and σ_-^2 are two factors.

Assuming that $d = 2$, γ_1 and γ_2 are uniformly distributed on $\{0, 1\}$, $\theta = [0.5, 0.5]^T$, and $\sigma_+^2 = \sigma_-^2 = 0.04$. Let D be a training set which is sampled from the above distribution. Due to possible sampling bias, the proportion of the samples from the two sub-distributions is 1:30 (C1:C2) for class '+1', and that is still 1:30 (C3:C4) for class '-1', as shown in Fig. 3. Such biased training data would result in a biased classifier. Obviously, sample weighting (e.g., importance weighting can alleviate this issue. Indeed, feature perturbation can also address this issue. Theoretically, if $\frac{14}{30}$ samples in C_1 as shown in Fig. 3 are perturbed by $1.5\boldsymbol{\theta}$ (i.e., $\boldsymbol{x} = \boldsymbol{x} + 1.5\boldsymbol{\theta}$) and $\frac{14}{30}$ samples in C_4 as shown in Fig. 3 are also perturbed by $1.5\boldsymbol{\theta}$, then the distribution on the training data is equal to the ground-truth distribution as described by (22). Consequently, an unbiased classifier would be learned.

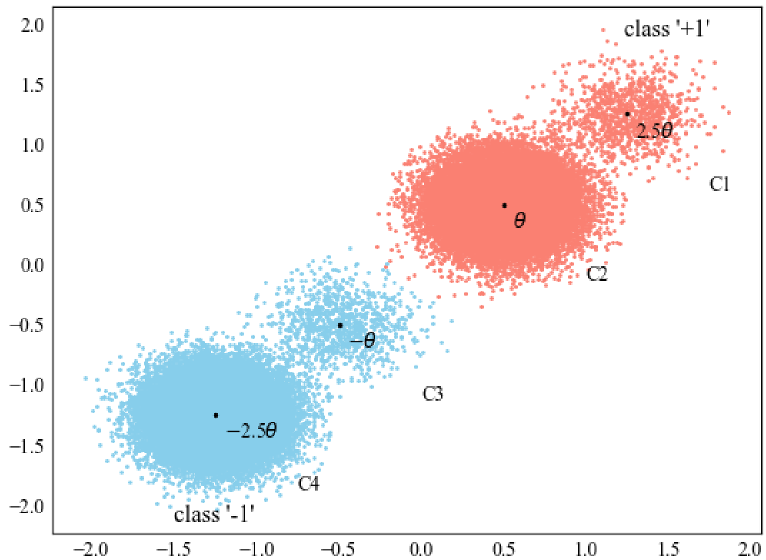

Fig. 3. A biased training set whose ground-truth distribution conforms to (22).

Obviously, the selected $\frac{14}{30}$ samples in C_1 are performed positive perturbation, whereas the selected $\frac{14}{30}$ ones in C_4 are performed negative perturbation. Several insights can be observed from this example:

- Perturbation can tune the distribution of training data like weighting.
- Although positive and negative perturbations are opposite to each other, they can cooperate to obtain better training performance for a learning task.
- Positive perturbation can reduce the proportion of hard samples (e.g., samples in C1), whereas negative perturbation can increase the proportion of hard samples (e.g., samples in C3). It is inappropriate to simply conclude that only positive perturbation or only negative perturbation is sufficient for a concrete learning task. Which direction of perturbation is required depends on the training data and the training object of a learning task.

The above theoretical comparison indicates that it is also inappropriate to directly conclude which perturbation direction is absolutely better than the other, regardless of the involved learning task. The next section will show several classical methods that employ mix-direction perturbation, i.e., both positive and negative perturbations in their approach are adopted simultaneously.

4.3.3 Representative methods.

We first revisit the three methods listed in Section 4.2.2. If the $l2$-norm distance in Eq. (10) is taken as loss, then the loss will be reduced by the perturbation o_i. As a result, the feature perturbation in robust clustering belongs to positive perturbation. In logit adjustment, Eq. (12) can be re-written to the following:

$$\mathcal{L} = -\sum_i \log \frac{e^{u_{i,y_i}}}{\sum_c e^{u_{i,c}+\tau \log \frac{\pi_c}{\pi_{y_i}}}}, \qquad \tau \geq 0. \tag{23}$$

Note that $\pi_c \leq \pi_1$ and $\pi_c \geq \pi_C$ for all the cth categories. Consequently, the losses for samples in the first head category y_1 are reduced, while the losses for samples in the last tail category y_C are increased. In other words, the first head category performs positive augmentation, while the last tail category performs negative augmentation. Existing studies reveal that overfitting occurs on tail categories [10, 30]. Therefore, negative augmentation (regularization) for tail categories is reasonable. Positive augmentation (anti-regularization) may avoid underfitting on head categories, as there are also studies that reveal that underfitting occurs on head categories [96]. Logit adjustment belongs to mix-direction augmentation. Label smoothing also belongs to mix-direction.

Adversarial samples receive great attention in recent years. It is actually a perturbed sample by the following optimization [53]:

$$\boldsymbol{x}_{\text{adv}} = \boldsymbol{x} + \arg \max_{\|\boldsymbol{\delta}\| \leq \epsilon} \ell(f(\boldsymbol{x} + \boldsymbol{\delta}), y), \tag{24}$$

where $\boldsymbol{x}_{\text{adv}}$ is the adversarial sample generated for $\boldsymbol{x}$, $\boldsymbol{\delta}$ is the perturbation term, and ϵ is the perturbation bound. Obviously, adversarial perturbation belongs to negative perturbation as the loss of new sample is larger (at least no small) than the raw sample.

Another example is adversarial label smoothing(ALS) [21]. The perturbation term of label smoothing is manually determined. Inspired by adversarial samples, ALS pursues the label perturbation in label smoothing using

$$\Delta y_i = \lambda(p_i^* - y_i), \tag{25}$$

where

$$p_i^* = \arg \max_{p_i} l(\mathbb{S}(u_i), y_i + \lambda(p_i - y_i)). \tag{26}$$

Eq. (25) has an analytic solution such that p_i^* is the one-hot vector for the category which corresponds to the minimum softmax value in $\mathbb{S}(u_i)$. It is easy to verify that $\ell(x_i, y+\Delta y_i) \geq \ell(x_i, y)$. Therefore, ALS belongs to negative perturbation.

4.4 Perturbation Inference

4.4.1 Sub-categories.

In perturbation learning, perturbation variables in losses in Eqs. (4)–(27) should be inferred during training. There are six typical manners (maybe not exhaustive) to infer their values and optimize the whole loss.

(1) **Inference with prior knowledge**. In this manner, the perturbation variables are inferred on the basis of prior knowledge. Alternatively, the perturbation variables are fixed before the optimizing of training loss.

(2) **Inference with hyper-parameter tuning**. In this manner, the perturbation variable(s) is/are taken as hyper-parameter(s). Consequently, the optimal value is determined according to the manner of hyper-parameter tuning.

(3) **Inference with regularization**. In this manner, a regularization term is added for the perturbation variables. For example, a natural assumption is that the proportion of the samples that require the perturbation variables is small. Therefore, $l1$-norm can be used. Taking the logit perturbation as examples. A loss function is defined as follows:

$$\mathcal{L} = \sum_i l(\mathbb{S}(u_i + \Delta u_i), y_i) + \lambda Reg(\Delta u_i), \tag{27}$$

where λ is a hyper-parameter and $Reg(\cdot)$ is a regularizer. This manner is similar to the self-paced learning [37, 50]. When $\lambda \rightarrow \infty$, no perturbation is allowed and perturbation learning is reduced to conventional learning.

(4) **Inference with meta-learning**. In this manner, the perturbation variables are inferred on the basis of another small clean validation set with meta-learning. Given a clean validation set Ω comprising M clean training samples and taking loss perturbation as an example. Let κ_i be the loss perturbation variable for x_i ($\in S$). We first define that

$$\mathcal{L} = \sum\nolimits_{i \in S} l(\mathbb{S}(u_i), y_i : \Theta) + \kappa_i, \tag{28}$$

where Θ is the model parameter set to be learned. Given $\boldsymbol{\kappa} = \{\kappa_i\}_{i \in S}$, Θ can be optimized on the training set S by solving

$$\Theta^*(\boldsymbol{\kappa}) = \arg\min_{\Theta} \sum\nolimits_{i \in S} l(\mathbb{S}(u_i), y_i : \Theta) + \kappa_i. \tag{29}$$

After Θ is obtained, $\boldsymbol{\kappa}$ can be optimized on the validation set Ω by solving

$$\boldsymbol{\kappa}^* = \arg\min_{\boldsymbol{\kappa}} \sum\nolimits_{j \in \Omega} l(\mathbb{S}(u_j), y_j : \Theta^*(\boldsymbol{\kappa})). \tag{30}$$

These two optimizations can be performed alternately, and finally Θ^* and $\boldsymbol{\kappa}^*$ are learned. When either logit or label perturbation is used, the above optimization procedure can also be utilized with slight variations.

The above inference manner is similar with that used in the meta-learning-based weighting strategy for robust learning [63]. Meta-learning has been widely used in robust learning and many existing meta-learning-based weighting methods [42, 81] can be leveraged for perturbation learning.

(5) **Inference with adversarial learning**. In both feature and logit perturbations, the perturbation term can be obtained by adversarial learning. Taking feature perturbation as an example, the objective function in negative perturbation is

$$\Delta x_i^* = \arg \max_{\|\Delta x_i\| \le \epsilon} l(\mathbb{S}(f(x_i + \Delta x_i)), y_i), \tag{31}$$

where ϵ is the bound. Likewise, the objective function in positive feature-level perturbation can be

$$\Delta x_i^* = \arg \min_{\|\Delta x_i\| \le \epsilon} l(\mathbb{S}(f(x_i + \Delta x_i)), y_i). \tag{32}$$

(6) **Inference with mixed manners**. Two or more of the above five manners can be combined together to infer the perturbation term in a learning task.

Remark: Existing perturbation-based learning methods adopt one of the inference manners listed above. Different manners have their own merits and defects. The prior knowledge-based manner is heuristic and thus seems quite ad hoc in some learning cases. When grid search is utilized in hyper-parameter tuning, it results in high time consumption. Regularization-based manner has good theoretical merits. However, designing a suitable regularizer is also challenging. Furthermore, as discussed previously, some learning cases may require anti-regularization. Both meta-learning-based and adversarial learning-based manners employ an optimization approach. Nevertheless, meta-learning requires an independent high-quality validation dataset, while the adversarial learning-based manner can only produce negative perturbations.

Which inference manner should be employed depends on the training data and learning object of the involved learning task. Theoretically, the inference manner can be changed from one manner to another and a new method will subsequently be obtained.

4.4.2 Representative methods.

The methods logit adjustment, label smoothing, and knowledge distillation employs prior knowledge-based inference. The methods adversarial samples and ALS employs adversarial learning.

In contrast with previous data augmentation techniques, Implicit semantic data augmentation (ISDA) [79] does not produce new samples or features. Instead, it transforms the semantic data augmentation problem into the optimization of a new loss defined as

$$\mathcal{L}= -\sum\nolimits_i \frac{e^{u_{i,y_i}}}{\sum\limits_{c=1}^{C} e^{u_{i,c}+\frac{\lambda}{2}(\mathrm{w}_c-\mathrm{w}_{y_i})^T\Sigma_{y_i}(\mathrm{w}_c-\mathrm{w}_{y_i})}}, \tag{33}$$

where Σ_{y_i} is the covariance matrix for the y_i-th category, w_c is the model parameter for the logit vectors, and $u_{i,c} = {\mathrm{w}_\mathrm{c}}^T \tilde{x}_i$ ($\tilde{x}_i$ is the output of the last feature encoding layer for x_i).

In Eq. (39), a logit perturbation term is observed as follows:

$$u_i' = u_i + \delta_{y_i}, \tag{34}$$

where

$$\delta_{y_i}=\frac{\lambda}{2}\begin{bmatrix} (\mathrm{w}_1-\mathrm{w}_{y_i})^T\Sigma_{y_i}(\mathrm{w}_1-\mathrm{w}_{y_i}) \\ \vdots \\ (\mathrm{w}_C-\mathrm{w}_{y_i})^T\Sigma_{y_i}(\mathrm{w}_C-\mathrm{w}_{y_i}) \end{bmatrix}. \tag{35}$$

Obviously, the perturbation is category-level and determined with prior knowledge. In addition, the perturbation direction is negative as the loss is increased for each training sample. The term is heavily dependent on the covariance matrix Σ_{y_i}, which can be further optimized via meta-learning by minimizing the following loss on a validation set Ω:

$$\Sigma^* = \arg\min_{\Sigma} \sum_{j\in\Omega} l\left(\mathbb{S}\left(u_j\right), y_j; \Theta^*(\Sigma)\right), \tag{36}$$

which is just the meta implicit data augmentation (MetaSAug) proposed by Li et al. [44]. MetaSAug is quite effective in long-tail classification.

Bootstrapping loss [62] is another classical label perturbation method. Given that for each sample, we can obtain a predicted label y_i' by the model trained at the previous epoch, the label perturbation can be defined as

$$\Delta y_i=\lambda(y_i' - y_i), \tag{37}$$

where λ is a hyper-parameter and locates in [0, 1]. Δy_i defined in Eq. (37) satisfies the condition given by Eq. (7). Bootstrapping loss is designed for noisy-label learning and the perturbation is infereed with prior knowledge. If y_i' is in trust, then it is highly possible that Δy_i approaches to zero if x_i is normal, and it is large if x_i is noisy. The entire Bootstrapping loss is

$$\mathcal{L} = \sum\nolimits_i l(\mathbb{S}(u_i), y_i+\lambda(y_i' - y_i)) = (1-\lambda)\sum\nolimits_i l(\mathbb{S}(u_i), y_i) + \lambda\sum\nolimits_i l(\mathbb{S}(u_i), y_i'). \tag{38}$$

Note that $l(\mathbb{S}(u_i), y_i') \le l(\mathbb{S}(u_i), y_i), \forall i$. Then Bootstrapping loss belongs to positive perturbation.

Meta adversarial perturbations [89] utilizes meta-learning to infer the adversarial perturbations for each image. In Eq. (12), the hyper-parameter τ is fixed for all categories. A category-wise setting for τ may be useful. Therefore, a new logit adjustment with meta optimization on τ is proposed and called Meta logit adjustment (Meta LA). Let Ω be the validation set for meta optimization. According to Eqs. (28–30) in the paper, the new loss is

$$\mathcal{L} = -\sum_{x_i \in S} \log \frac{e^{u_{i,y_i} + \tau_{y_i} \log \pi_{y_i}}}{\sum_y e^{u_{i,y} + \tau_{y_i} \log \pi_y}}. \tag{39}$$

Given a value for $\tau = \{\tau_1, , \tau_C\}$, the network parameter Θ can be obtained by solving

$$\Theta^*(\tau) = \arg\min_{\Theta} - \sum_{x_i \in S} \log \frac{e^{u_{i,y_i} + \tau_{y_i} \log \pi_{y_i}}}{\sum_y e^{u_{i,y} + \tau_{y_i} \log \pi_y}}. \tag{40}$$

After $\Theta^*(\tau)$ is obtained, τ can be optimized by solving

$$\tau^* = \arg\min_{\tau} - \sum_{x_i \in \Omega} l\left(\text{softmax}\left(f\left(x_i : \Theta^*\left(\tau\right)\right), y_i\right)\right). \tag{41}$$

Eqs. (40) and (41) are solved alternately. The detailed optimization steps are similar to those used in MetaSDA [44], Meta-Weight-Net [68], and other meta optimization studies.

4.5 Perturbation Granularity

4.5.1 Sub-categories.

Perturbation granularity has four levels.

(1) **Sample-level perturbation**. All the perturbation variables discussed above are for samples. Each sample has its own perturbation variable.

(2) **Category-level perturbation**. In this level, samples within the same category share the same perturbation. Taking the logit vector-based perturbation as an example, when category-level perturbation is utilized, the loss in Eq. (5) becomes

$$\mathcal{L} = \sum\nolimits_i l(\mathbb{S}(u_i + \Delta u_{y_i}), y_i). \tag{42}$$

Category-level perturbation mainly solves the problem when the impact of all the samples of a category should be increased or decreased. For example, in long-tail classification, the tail category should be emphasized in learning.

(3) **Corpus-level perturbation**. In this level, samples within the whole training corpus share the same perturbation. Take the negative perturbation described in Eq. (31) as an example, the objective function becomes

$$\Delta x^* = \arg\max_{\|\Delta x\| \le \epsilon} l(\mathbb{S}(f(x_i + \Delta x)), y_i), \tag{43}$$

which means that all samples share the same term Δx^*. Δx^* is exactly the universal adversarial perturbation [57].

(4) **Mix-level perturbation**. In this level, more than one of the aforementioned three levels are utilized simultaneously. This case occurs in complex contexts, e.g., when both noisy labels and category imbalance exist. Taking label-based perturbation as an example. The loss in Eq. (6) can be written as

$$\mathcal{L} = \sum\nolimits_i l(p_i, y_i + \Delta y_i + \Delta y_{y_i}), \tag{44}$$

where Δy_{y_i} is the category-level label perturbation.

Remark: Most methods belong to sample-level, which has been applied in most learning scenarios. Corpus-level is a special case of category-level, and category-level is also a special case of sample-level. Therefore, sample-level should outperform the other granularity levels theoretically. Nevertheless, it is still inappropriate to conclude that which level is absolutely the best choice.

4.5.2 *Representative methods.*

Adversarial perturbation introduced previously is in the sample level. Training with adversarial samples (i.e., adversarial training) is proven to be useful in many applications and various methods are proposed [53]. Shafahi et al. [65] proposed universal adversarial training (UAT) which is actually based on a corpus-level negative feature perturbation. The loss on adversarial samples is

$$\mathcal{L}_{corpus-adv} = \max_{\|\delta\| \leq \epsilon} \sum\nolimits_i l\left(\mathbb{S}(f(x_i + \delta)), y_i\right). \tag{45}$$

Benz et al. [3] observed that universal adversarial perturbation does not attack all classes equally. They proposed a category-wise universal adversarial training (C-UAT) method and the loss on adversarial samples is

$$\mathcal{L}_{category-adv} = \max_{\|\delta_{y_i}\| \leq \epsilon} \sum\nolimits_i l\left(\mathbb{S}(f(x_i + \delta_{y_i})), y_i\right), \tag{46}$$

which belongs to the category-level negative feature perturbation.

Conventional sample-level adversarial samples and the corpus-level UAP are in the two extremes. Nevertheless, both are demonstrated to be quite useful in adversarial training. There are also a large number of studies on UAP [6], which partially reflects that each perturbation level has its own value.

Motivated by our taxonomy, mix-level adversarial perturbation can subsequently be generated. A mixed corpus/sample-level adversarial perturbation is described as an example:

$$\begin{aligned} \delta^* &= \arg\max_{\delta} \sum_i l(S(f(x_i + \delta)), y_i) \\ \mathcal{L}_{mixed-adv} &= \max_{\delta_i} \sum_i l(S(f(x_i + \delta^* + \delta_i)), y_i), \end{aligned} \tag{47}$$

where δ^* and δ_i are the corpus-level and sample-level perturbations, respectively. A further statistical analysis for the two levels of adversarial perturbations may illuminate us to better understand the adversarial characteristics of the data. Some other variations of adversarial perturbation (e.g., hash adversarial perturbation [87]) can also benefit from our taxonomy for learning with perturbation.

Arcface [48] is a classical face recognition loss defined as follows:

$$\mathcal{L} = -\sum\nolimits_i \frac{e^{s_i(cos(\theta_{i,y_i} + m))}}{e^{s_i(cos(\theta_{i,y_i} + m))} + \sum_{c \neq y_i} e^{s_i(cos(\theta_{i,c}))}}, \tag{48}$$

where $s_i = ||\mathbf{w}_{y_i}||||\tilde{x}_i||$, $\theta_{i,c}$ is the angle between the weight $\mathbf{w}_c$ and the feature $\tilde{x}_i$ which are defined in the description for ISDA, and m is a hyper-parameter. Indeed, m does not strictly belong to the five perturbation targets in our taxonomy. It is simply placed in the category of logit perturbation in this paper. It is a corpus-level term and determined via hyper-parameter tuning.

Wang et al. [76] proposed a new Arcface loss, namely, Balanced loss, with the category-level perturbation. The loss is defined as

$$\mathcal{L} = -\sum\nolimits_i \frac{e^{s_i(cos(\theta_{i,y_i} + m_{g_i}))}}{e^{s_i(cos(\theta_{i,y_i} + m_{g_i}))} + \sum_{c \neq y_i} e^{s_i(cos(\theta_{i,c}))}}, \tag{49}$$

where g_i is the skin-tone category of the j-th sample. Obviously, m_{g_i} is a category-level term. It can be optimized via meta-learning:

$$m_g^* = \arg\min_{\{m_{g_j}\}} \sum_{j \in \Omega} l(\Theta(m_{g_j})), \tag{50}$$

which is proven to be quite effective in the experiments conducted by Wang et al. [76].

4.6 Several potential directions for perturbation learning

There are numerous open problems for learning with perturbation. This part casts a vision toward the future, contemplating the promising research directions deserving further investigation listed below:

- **Deep representation of training characteristics**. Training characteristics of a sample denotes the static or dynamic quantities that can characterize information such as distribution, geometry, and neighborhood of the sample. For example, the categorical proportion, margin, loss, and gradient norm are typical training characteristics. In most existing perturbation learning methods, the perturbation direction, granularity, and inference manner (especially the manner with prior knowledge) heavily depend on the quantification of the training characteristics of training samples. Nevertheless, existing methods utilize no more than three raw training characteristics. A deep representation for the overall training characteristics of a training sample would be quite useful.
- **Unified theoretical basis for data perturbation**. Our taxonomy summarizes a wide range of methods which are based on distinct heuristic observation or theoretical inspirations. Constructing a unified theoretical basis for perturbation learning can establish a more fundamental connection among these seemingly irrelevant methods. This connection will contribute to answering the question of which target, direction, inference manner, or granularity level should be employed when facing a concrete learning task.
- **Data perturbation agent**. AI agent is a hot topic in current AI community. If the training characteristics of samples can be well represented and the theoretical basis is well constructed, then automatic data perturbation may be achieved. Indeed, it is feasible to compile more than thousands of learning tasks and train a data perturbation agent with a proper learning procedure.
- **Theoretical comparison with data weighting**. The weighting strategy is straightforward and quite intuitive; hence, it has been widely used in the machine learning community. Perturbation does not seem as straightforward as weighting. However, the former can play the same/similar role as weighting in machine learning. They both have their own merits. Perturbation is more flexible than weighting, while weighting is usually more efficient than perturbation. A theoretical comparison between them is beneficial for both strategies and their cooperation.

5 THREE NEW LEARNING METHOD EXAMPLES

In addition to the representative methods mentioned in the previous section, there are also other numerous typical methods such as Robust nonrigid ICP (RNICP) [29], D2L [51], DAC [73], Deep self-learning (DSL) [26], LDAM [4], MRFL [94], Robust regression (RR) [69], MAT [56], MSLC [83], PD-UA [47], AKD [8], Mixup [92], MetaMixup [54], MetaDistil [95], ZLA [7], Adaptive Face loss (AFL) [48], Robust LASSO (RLASSO) [60], Bootstrapping loss [62], online label smoothing (OLS) [91], AutoBalance [43], PolyLoss [40], DEFENSE_GEN [59], v-SVM [64], and RLR [17] can also be explained with perturbation learning. For example, Wang et al. [75] formulated the adversarial attack in object detection as a p-norm optimization problem, which can be seen as a regularization-based perturbation (called RegPert for brevity). Table 1 shows the coordinates of these methods according to our constructed taxonomy. The perturbation direction is not presented due to space limitation. The arrangement of numerous typical machine learning methods leveraging a general learning with perturbation taxonomy facilitates better understanding these methods and enlightens new inspirations for the design of more effective methods.

Table 1. The coordinates of several typical methods according to our constructed taxonomy.

Target	Prior knowledge			Hyper-parameter tuning			Regularization			Meta learning			Adversarial learning		
	Sample	Category	Corpus	Sample	Category	Corpus	Sample	Category	Corpus	Sample	Category	Corpus	Sample	Category	Corpus
Feature	Mixup [92]						RC [18], RegPert [75], RR [69], RNICP [29], DEFENSE_GEN [59]	MRFL [94]	PD-UA [47]	MetaMixUp [54]		MAT [56]	AT [53]	C-UAT [3]	UAT [65]
Logit	ZLA [7]	ISDA [79], LDAM [4]	LA [55]			Arcface [48]		AFL [48]			MetaSAug [44], Balanced loss [76], Meta LA	AutoBalance [43]			
Label	LS [71], Bootstrapping loss [62], D2L [51], Mixup [92]	OLS [91]					RLR [17], RLASSO [60]			MetaMixUp [54], MSLC [83]			ALS [21]		
Loss	KD [28], DSL [26], DAC [73], PolyLoss [40]					v-SVM [64]	SVM [9]			MetaDistil [95]			AKD [8]		

The empty lattices of Table 1 inspire us to explore new learning with perturbation methods[10]. In addition, our theoretical analysis shows that the combination of positive and negative augmentation has theoretical merits. To this end, this section shows three examples[11]. The first is the lattice for intersection of logit, sample, and regularization. The second is the mixed positive and negative perturbation. The third is the meta-learning version of the second one.

5.1 Sample-level Logit Perturbation

According to Table 1, sample-level logit perturbation receives litter attention in previous literature[12]. An example is given to explain how logit perturbation works. Assume that the inferred logit vector of a noisy sample x_i and its (noisy) label y_i are as follows:

$$\begin{aligned} u_i &= [3.0, 0.8, 0.2]^T \\ y_i &= [0, 1, 0]^T. \end{aligned} \tag{51}$$

The cross-entropy loss incurred by this training sample is $-log[e^{0.8}/(e^{3.0}+e^{0.8}+e^{0.2})] = 2.36$. This loss negatively affects training because y_i is noisy. To reduce the negative influence, if a perturbation vector (e.g., $[-1, 2, 0]^T$) is added, then the new logit vector becomes $[2.0, 2.8, 0.2]^T$. Consequently, the new loss of x_i is $-log[e^{2.8}/(e^{2.0}+e^{2.8}+e^{0.2})] = 0.42$, which is much lower than 2.36[13]. The negative influence of this noisy sample will be reduced significantly.

In actual learning tasks, samples with noisy labels are inevitable in the corresponding training corpora. However, which samples are truly noisy is unknown during training. Motivated by robust clustering (RC) [18] and robust LASSO [60], a regularized logit perturbation learning method is proposed with the following new loss:

$$\mathcal{L} = \sum\nolimits_i l(\mathbb{S}(u_i + v_i), y_i) + \lambda Reg(v_i), \tag{52}$$

where v_i is the logit perturbation vector for the i-th training sample and λ is a hyper-parameter. v_i is trainable during training. According to our taxonomy for learning with perturbation, the perturbation target, direction, inference manner, and granularity are logit, positive, regularization, and sample level, respectively, for the new loss. Naturally, extensions such as category-level logit perturbation and meta logit perturbation can be generated based on the proposed algorithm. We leave these new extensions as our future work.

When l_1-norm is used, the training loss becomes

$$\mathcal{L} = \sum\nolimits_i l(\mathbb{S}(u_i + v_i), y_i) + \lambda ||v_i||_1. \tag{53}$$

[10] It is worth noting that the empty lattices of Table 1 by no means indicate that there are no corresponding methods for each empty lattice in previous literature, as some studies may be not covered in our literature summary

[11] A family of new learning algorithms can be obtained by plugging the perturbation learning into existing learning algorithms with our constructed taxonomy by introducing the idea of perturbation learning into existing algorithms.

[12] The ZLA method in Table 1 is designed particularly for zero-shot learning.

[13] Indeed, the gradient norm for the new logit vector is also much smaller than that for the original vector.

Following self-paced learning [37], the alternative convex search (ACS) [1], which alternately optimizes the network and v, is used. We found that ACS obtained more stable results in our experiments. Let $\tilde{\mathbf{w}}$ be the model parameters in the current training epoch. v_i is achieved with the following optimization problem:

$$v_i^* = \arg\min_{v_i} l(\mathbb{S}(u_i + v_i), y_i : \tilde{\mathbf{w}}) + \lambda ||v_i||_1 \tag{54}$$

Then, the model parameters are updated with the following optimization problem:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} l(\mathbb{S}(u_i + v_i^*), y_i : \mathbf{w}) \tag{55}$$

Ideally, if neither noisy nor quite hard samples exist, then λ will be set to a large value. Consequently, v_i will approach to zero for all training samples. This method is called *LogPert* for brevity. The detailed steps are described in Algorithm 1.

Algorithm 1 LogPert

Input: Training set $S = \{x_i, y_i\}$, $i = 1, \cdots, N$; hyper-parameters λ; #Epoch; #Batch; and learning rate.

Output: Model $f(x, \mathbf{w})$.

```
Initialization: v = 0 for each training sample, w as w^(0);
repeat
    t = 1, ··· , #Epoch
        k = 1, ··· , #Batch
            Generate mini-batch D_k from S;
            Pursue v_i for each sample in D_k by solving (54);
            Update w by solving (55) using SGD;
until stable accuracy in the validation set.
```

5.2 Mixed Positive and Negative Perturbation

We observed that large perturbations (i.e., v_i) concentrate in samples with large losses during the running of LogPert in the experiments. Intuitively, we can only perturb the logit vectors of samples with large losses as the positive perturbations on samples with small losses are useless or even harmful. Let $l_i = l(\mathbb{S}(u_i), y_i)$. Motivated by adversarial training, (53) is modified into the following form

$$\mathcal{L} = \sum_{i:l_i \geq \tau} \min_{\|v_i\| \leq \epsilon} l\left(\mathbb{S}\left(u_i + v_i\right), y_i\right) + \sum_{i:l_i < \tau} l_i, \tag{56}$$

where ϵ is the perturbation bound and τ is the loss threshold. Compared with (53), (56) has one more hyper-parameter. Nevertheless, (56) is more flexible than (53). The results on image classification show that (56) is better than (53) if appropriate τ and ϵ are used. The discussion part will show that the classical self-paced learning manner [37] can be implemented by (56) with an increasing value of τ.

The proposed LogPert method relies on the positive perturbation to reduce the negative influence of samples which are noisy or quite hard. In our taxonomy, there is another perturbation direction, namely, negative perturbation which increases the losses of training samples. Typical negative perturbation methods such as adversarial training are considered as a useful technique, namely, data augmentation in previous studies. Our theoretical analysis in Section 4.3.2

reveals that the cooperation of positive and negative perturbations may yield better results, so mixed positive (to reduce the influence of noisy samples) and negative (to augment clean samples) perturbations are considered. Fig. 4 illuminates the influence of our proposed mixed positive and negative perturbation on training data.

On the basis of (56), a mixed perturbation is subsequently obtained with the following loss:

$$\mathcal{L} = \sum_{i:l_i \geq \tau} \min_{\|v_i\| \leq \epsilon_1} l\left(\mathbb{S}(u_i + v_i), y_i\right) + \sum_{i:l_i < \tau} \max_{\|\delta_i\| \leq \epsilon_2} l\left(\mathbb{S}(f\left(x_i + \delta_i\right)), y_i\right). \tag{57}$$

The main difference between (57) and the adversarial training loss [53] is that the losses of quite hard (including noisy) samples are not increased any more in (57). Instead, the losses of these samples are reduced as in (57). When $\tau > \max_i \{l_i\}$, only the maximization part exists and the whole loss becomes the adversarial training loss; when $\epsilon_2 = 0$, (57) is reduced to (56).

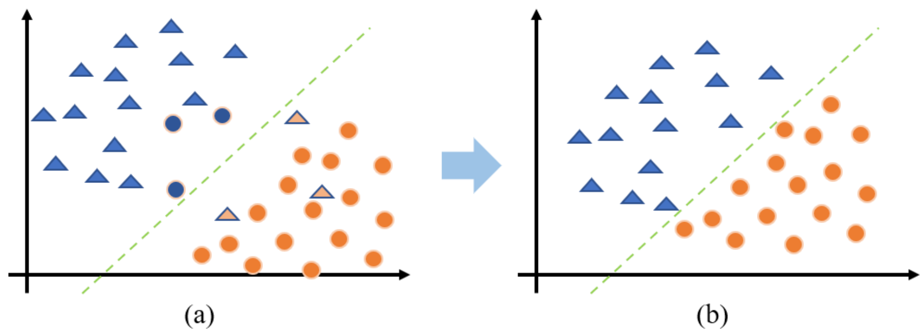

Fig. 4. An illustration of the effects of mixed positive and negative perturbation. The raw training data is shown in (a). The positive perturbation implicitly deletes the six noisy-label samples; while the negative perturbation implicitly pushes the normal training data toward the decision boundary. The perturbed training data, which can be viewed as augmented samples, is shown in (b).

Algorithm 2 MixPert

Input: Training set $S = \{x_i, y_i\}$, $i = 1, \cdots, N$; #Epoch; #Batch; learning rate; ϵ_1; ϵ_2; and τ.

Output: Model $f(x, \mathbf{w})$.

Initialization: $\mathbf{w}$ as $\mathbf{w}^{(0)}$;
repeat
  $t = 1, \cdots$, #Epoch
    $k = 1, \cdots$, #Batch
      Generate mini-batch D_k from S;
      Infer v_i according to Eq. (59) for samples with a lower loss than τ;
      Infer δ_i for the rest samples according to PGD optimization;
      Calculate loss based on Eq. (57);
      Update $\mathbf{w}$ using SGD;
until stable accuracy in the validation set.

The minimization part in both (56) and (57) can be solved with an optimization approach similar to PGD [53]. This method is called *MixPert* for brevity. The PGD-like optimization for the minimization part in (62) is as follows. First, we have

$$\frac{\partial l(\mathbb{S}(u_i+v_i), y_i)}{\partial v_i}|_{v_i=0} = \mathbb{S}(u_i) - \hat{y}_i, \tag{58}$$

where $\hat{y}_i$ is the one-hot vector of y_i. Therefore, v_i can be calculated by

$$v_i = \eta(\hat{y}_i - \mathbb{S}(u_i)), \tag{59}$$

where η is a hyper-parameter. Accordingly, the updating of u_i is

$$u_i' = u_i+\eta(\hat{y}_i - \mathbb{S}(u_i)). \tag{60}$$

In our implementation, only one updating step is used. Consequently, if ∞-norm is used, then we have

$$|v_i| = |\eta(\mathbb{S}(u_i) - \hat{y}_i)| \leq |\eta||(\mathbb{S}(u_i) - \hat{y}_i)| \leq \eta. \tag{61}$$

Therefore, we use η to control the bound (i.e., ϵ_1) of v_i. The detailed steps of MixPert are described in Algorithm 2.

Eq. (57) determines which direction of perturbation is performed for a training sample solely based on the loss in the current epoch. As introduced in Section 4.6, training loss is a typical quantity for training characteristics. If an independent dataset is available, meta-learning can be employed to automatically determine the positive or negative direction based on more training characteristics. Let α_i ($\in \{0, 1\}$) be a binary variable to denote the choice of positive or negative perturbation for a sample x_i. The loss in Eq. (57) becomes

$$\mathcal{L} = \sum_i \alpha_i \min_{\|v_i\| \leq \epsilon_1} l(\mathbb{S}(u_i+v_i), y_i) + (1-\alpha_i) \max_{\|\delta_i\| \leq \epsilon_2} l(\mathbb{S}(f(x_i+\delta_i)), y_i), \quad \alpha_i \in \{0, 1\}. \tag{62}$$

If $\alpha_i = 1(l_i \geq \tau)$, then (62) becomes (57). Here we employ three widely used training characteristics including training loss, gradient norm, and functional margin to infer the value of α_i via meta-learning. Following Meta-Weight-Net [68], we employ a multi-layer perceptron (MLP) with 100 hidden nodes, taking the three training characteristics as input. The output is α_i, which is transformed into a real number through a Sigmoid function. Similar optimization steps with those of Meta-Weight-Net used in [68] are leveraged. This method is called Meta-MixPert. The main pipeline of Meta-MixPert is shown in Fig. (5).

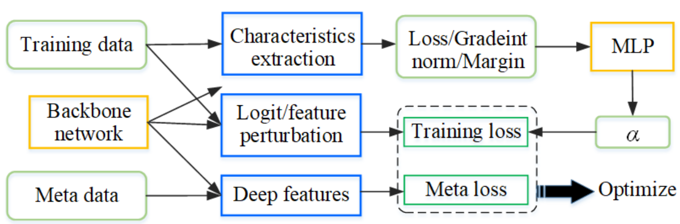

Fig. 5. The main pipeline of the proposed Meta-MixPert.

6 EXPERIMENTS

This section evaluates our methods (LogPert, MixPert, and Meta-MixPert) in image classification and text sentiment analysis, and experiments when the datasets have noises are also considered.

6.1 Competing Methods

As our proposed methods belong to the end-to-end noise-aware solution, the following methods are compared: soft/hard Bootstrapping [62], label smoothing (LS) [71], online label smoothing (OLS) [91], progressive self label correction (ProSelfLC) [77], PGD-based adversarial training (PGD-AT) [53], Self-Distillation from Last Mini-Batch (DLB) [67], and Margin-based Label Smoothing (MbLS) [46].

The parameter settings are detailed in the corresponding subsections. All the results are the average values of three repeated runs.

6.2 Image Classification

Four benchmark image classification datasets, namely, CIFAR-10, CIFAR-100 [36], ImageNet [13], and Clothing-1M [84] are used. To simulate the noisy-label learning scenario, label noise should be added in the training data [14]. The synthetic label noises are simulated on the basis of the two common schemes used in [23, 25, 34]. The first is the random scheme in which each training sample is assigned to a uniform random label with a probability p. The second is the pair scheme in which each training sample is assigned to the category next to its true category on the basis of the category list with a probability p. The value of p is set to 10%, 20%, and 30%.

6.2.1 Experiments on CIFAR-10. CIFAR-10 consists of 50k training images and 10k test images in 10 classes. Resnet-20, ResNet-32, ResNet-44, ResNet-56, and ResNet-110 [27] are used as base neural networks to evaluate our proposed LogPert and MixPert on the dataset.

For all the neural networks, the #epochs are set to 300 and batch size is set to 128. SGD is uesd as the optimizer. The initial learning rate is set to 0.1 and decayed by a factor of 0.1 at the 150th and 225th epochs. In LogPert, λ is searched in {0.175, 0.35} and the learning rate for the perturbation variable is searched in {1.5, 3, 6, 12}. In MixPert, $\epsilon_1 (i.e., \eta)$ is searched in {0.5, 1, 2, 4}, and ϵ_2 is searched in {0, 8/255, 10/255, 12/255}. τ is determined according to the top-pro percent of ordered losses, and the value of pro is searched in {0, 15, 25, 35, 45, 50}. In PGD-AT, ϵ_2 is is searched in {8/255, 10/255, 12/255}. For other competing methods, namely, soft/hard Bootstrapping, LS, OLS, DLB, and MbLS, we follow the parameter settings in their original papers.

The results are shown in Table 2 when ResNet-20 is used as the base neural network. For 0% noise, our proposed method LogPert achieves the highest accuracy, and for other noises, our proposed method MixPert achieves the best performance. The results of MixPert are obtained when ϵ_2 equals to 0, indicating that only positive perturbation is useful for the (clean) accuracy. Indeed, both the hyper-parameters ϵ_2 and τ balance the trade-off between the positive and negative perturbations. Comparisons on other base networks, namely, ResNet-32, ResNet-44, ResNet-56, and ResNet-110, are also conducted. Tables 3 and 4 present the classification accuracies of the competing methods with the above four base networks on partial noisy rates. Our proposed method MixPert achieves the best results.

When LogPert is used, some original labels with high average perturbation terms are found to be erroneous. Fig. 6 shows two samples from CIFAR-10. Their labels seem wrong.

In addition, we plot the distribution of $l1$-norm of perturbed logit vectors when using LogPert on CIFAR-10 dataset without label noises (0%). The results are shown in Fig. 7. The distribution curve shows a long-tail trend, which is quite reasonable.

To verify the performance of the Meta-MixPert algorithm we proposed and to ensure fair evaluation, the settings in [74, 81] are followed. We randomly select 1000 images in training set as the small clean validation set for CIFAR-10.

[14] If label noises are added into the training data, then performance will drop seriously for conventional learning methods. Accordingly, noisy-label learning methods are designed to reduce the serious performance drop.

Table 2. Classification accuracies (%) and standard deviations on CIFAR-10 (ResNet-20).

		Random noise			Pair noise		
	0%	10%	20%	30%	10%	20%	30%
Base (ResNet-20)	91.79±0.31	88.78±0.33	87.55±0.32	85.85±0.37	90.32±0.19	89.28±0.14	87.06±0.23
Soft Bootstrapping	91.83±0.12	89.37±0.18	87.52±0.37	85.59±0.33	90.44±0.23	89.16±0.22	87.08±0.25
Hard Bootstrapping	92.06±0.10	89.61±0.20	88.07±0.32	86.37±0.26	90.34±0.18	89.54±0.25	86.86±0.19
Label Smoothing	92.12±0.14	90.15±0.09	88.54±0.18	86.82±0.16	90.63±0.22	90.12±0.06	88.28±0.42
Online Label Smoothing	92.18±0.15	89.84±0.14	88.19±0.15	86.08±0.22	90.65±0.18	89.52±0.08	87.68±0.16
ProSelfLC	91.80±0.16	89.90±0.16	88.84±0.22	86.78±0.31	90.40±0.23	89.76±0.17	87.11±0.20
PGD-AT	89.90±0.08	87.56±0.13	86.87±0.13	84.80±0.17	88.90±0.15	88.38±0.07	86.79±0.13
DLB	91.87±0.13	89.59±0.21	87.92±0.26	85.90±0.28	90.35±0.20	89.32±0.19	87.13±0.28
MbLS	92.20±0.12	90.18±0.14	88.93±0.21	86.89±0.22	90.63±0.18	90.10±0.12	88.31±0.17
LogPert	**93.04±0.07**	91.07±0.05	90.42±0.13	88.86±0.16	91.74±0.10	91.29±0.07	89.95±0.11
MixPert	92.94±0.06	**91.09±0.11**	**90.63±0.12**	**88.98±0.15**	**92.18±0.06**	**91.41±0.08**	**90.01±0.16**

Table 3. Classification accuracies (%) and standard deviations on CIFAR-10 (0% noise) when using different base neural networks.

	ResNet-32	ResNet-44	ResNet-56	ResNet-110
Base	92.50±0.26	92.82±0.15	93.03±0.34	93.51±0.18
Soft Bootstrapping	92.40±0.17	92.83±0.16	93.43±0.27	94.08±0.29
Hard Bootstrapping	92.19±0.23	92.94±0.11	93.38±0.25	94.02±0.23
Label Smoothing	92.75±0.24	92.89±0.18	93.05±0.23	93.92±0.43
Online Label Smoothing	92.61±0.19	92.93±0.34	93.41±0.20	93.54±0.18
ProSelfLC	92.87±0.22	92.98±0.28	93.21±0.19	93.58±0.37
PGD-AT	90.66±0.16	91.31±0.19	91.80±0.22	91.98±0.15
DLB	92.51±0.24	92.85±0.27	93.44±0.26	94.04±0.22
MbLS	92.90±0.19	93.01±0.22	93.48±0.27	94.09±0.24
LogPert	93.71±0.10	94.07±0.07	94.39±0.15	94.91±0.11
MixPert	**93.95±0.14**	**94.15±0.13**	**94.52±0.14**	**95.14±0.10**

Table 4. Classification accuracies (%) and standard deviations on CIFAR-10 (20% pair noise) when using different base neural networks.

	ResNet-32	ResNet-44	ResNet-56	ResNet-110
Base	89.66±0.32	89.83±0.25	90.11±0.31	90.55±0.27
Soft Bootstrapping	89.79±0.28	89.98±0.24	90.17±0.25	90.59±0.30
Hard Bootstrapping	89.94±0.29	90.06±0.27	90.21±0.23	90.66±0.33
Label Smoothing	90.52±0.15	90.83±0.19	91.05±0.22	91.31±0.24
Online Label Smoothing	90.65±0.13	90.81±0.16	90.95±0.21	91.16±0.20
ProSelfLC	90.58±0.17	91.01±0.19	91.16±0.24	91.48±0.27
PGD-AT	89.07±0.12	89.37±0.15	89.94±0.17	90.41±0.22
DLB	90.01±0.26	90.19±0.25	90.38±0.19	91.01±0.19
MbLS	90.68±0.26	91.03±0.29	91.15±0.25	91.51±0.22
LogPert	91.74±0.12	92.04±0.14	92.26±0.13	92.79±0.16
MixPert	**91.87±0.13**	**92.13±0.12**	**92.44±0.15**	**93.16±0.11**

If the compared methods do not rely on the small clean validation set, both the training and the small clean validation sets are merged as training set. MbLS [46], which performed better in the aforementioned experiments, is used for comparison. ResNet-56 and ResNet-110 are used as the base neural networks. The results are shown in Table 5. When employing meta-learning strategies, our proposed MixPert shows further improvement, and Meta-MixPert achieves the best results.

Fig. 6. Samples with high average perturbation terms whose labels seem erroneous.

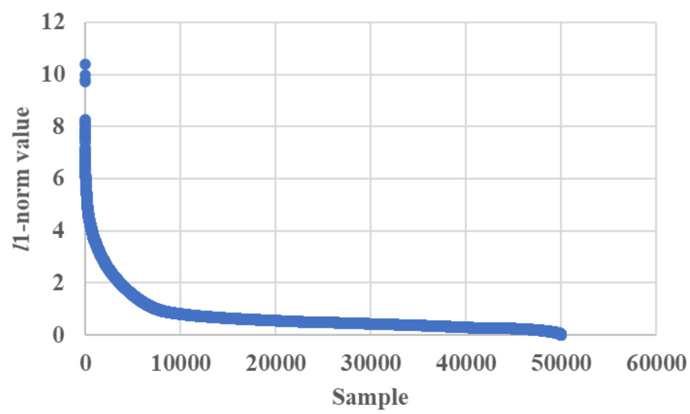

Fig. 7. Distribution of $l1$-norm of perturbed logit vectors on CIFAR-10.

Table 5. Classification accuracies (%) and standard deviations on CIFAR-10 (ResNet-56 and ResNet-110).

	ResNet-56		ResNet-110	
	0%	20% pair noise	0%	20% pair noise
Base	93.03±0.34	90.13±0.28	93.51±0.18	90.60±0.26
MbLS	93.48±0.27	91.20±0.33	94.09±0.24	91.46±0.19
MixPert	94.52±0.14	92.47±0.16	95.14±0.10	93.22±0.20
Meta-MixPert	**95.19±0.17**	**93.86±0.11**	**95.83±0.13**	**94.80±0.18**

6.2.2 Experiments on CIFAR-100. CIFAR-100 consists of 50k training images and 10k test images in 100 classes. ResNet-20, ResNet-32, ResNet-44, ResNet-56, and ResNet-110 are also used as base neural networks. Training details are the same as the experiments on CIFAR-10. In addition, Wide-ResNet-28-10 (WRN-28-10) [90] is also employed. We follow the same experimental setting as [79].

The experimental results on CIFAR-100 are shown in Tables 6, 7, 8, and 9. LogPert and MixPert outperform all competing methods. Compared with the base neural network ResNet20, the maximum improvement of the LogPert is 6.01%, and the minimum improvement is 1.53%. The maximum improvement of the MixPert is 7.81%, and the minimum improvement is 1.32%. In 20% pair noise experiments, compared with ResNet-32, ResNet-44, ResNet-56, and ResNet-110, MixPert improves the accuracy by 4.68%, 6.34%, 5.98%, and 4.35%, respectively. Compared with the base neural network WRN-28-10, the maximum improvement of the LogPert is 3.72%, and the minimum improvement is 2.15%. The maximum improvement of the MixPert is 4.10%, and the minimum improvement is 2.26%. All the experimental results show that our proposed methods significantly improve the classification performance. In addition, ResNet-110 and WRN-28-10 are used as the base neural networks to evaluate our proposed Meta-MixPert method. The experimental results are shown in Tables 10. Our proposed Meta-MixPert achieves the best results.

Table 6. Classification accuracies (%) on CIFAR-100 (ResNet-20).

		Random noise			Pair noise		
	0%	10%	20%	30%	10%	20%	30%
Base (ResNet-20)	67.81±0.08	63.67±0.29	60.63±0.33	57.82±0.35	63.94±0.29	61.22±0.03	55.74±0.22
Soft Bootstrapping	68.38±0.24	64.01±0.23	60.66±0.28	57.97±0.23	64.29±0.31	60.71±0.23	56.27±0.26
Hard Bootstrapping	67.62±0.29	64.28±0.33	60.32±0.22	58.09±0.19	63.96±0.26	60.69±0.29	56.18±0.17
Label Smoothing	67.54±0.10	65.04±0.18	61.84±0.27	59.06±0.08	65.43±0.24	62.71±0.24	58.92±0.19
Online Label Smoothing	67.80±0.19	64.55±0.15	61.53±0.22	59.19±0.13	64.70±0.28	62.54±0.19	57.44±0.25
ProSelfLC	68.37±0.22	64.64±0.28	62.14±0.17	58.93±0.24	65.36±0.18	62.57±0.16	59.08±0.27
PGD-AT	64.37±0.17	60.39±0.24	57.38±0.21	54.23±0.16	60.41±0.20	58.08±0.13	54.37±0.22
DLB	68.33±0.19	64.30±0.33	61.02±0.27	58.05±0.24	64.33±0.25	61.41±0.15	57.09±0.19
MbLS	68.40±0.27	65.09±0.15	62.17±0.28	59.21±0.22	65.39±0.17	62.76±0.14	59.22±0.21
LogPert	**69.34±0.08**	65.63±0.12	62.64±0.14	59.70±0.13	66.59±0.17	64.81±0.09	61.75±0.15
MixPert	69.13±0.12	**65.79±0.14**	**62.76±0.20**	**60.17±0.12**	**66.81±0.16**	**64.83±0.11**	**63.55±0.13**

Table 7. Classification accuracies (%) and standard deviations on CIFAR-100 (0% noise) when using different base neural networks.

	ResNet-32	ResNet-44	ResNet-56	ResNet-110
Base	69.16±0.19	70.02±0.19	70.38±0.34	73.18±0.12
Soft Bootstrapping	69.76±0.25	70.76±0.34	71.01±0.40	74.19±0.24
Hard Bootstrapping	69.37±0.24	70.06±0.29	70.26±0.31	73.35±0.18
Label Smoothing	69.91±0.27	70.52±0.51	71.49±0.29	74.01±0.44
Online Label Smoothing	69.53±0.22	70.05±0.79	71.06±0.26	73.59±0.19
ProSelfLC	69.54±0.29	70.39±0.35	70.49±0.32	73.42±0.24
PGD-AT	65.94±0.18	66.55±0.26	67.58±0.29	70.83±0.17
DLB	69.74±0.27	70.44±0.32	70.79±0.35	73.87±0.18
MbLS	69.93±0.26	70.59±0.31	71.50±0.25	74.19±0.23
LogPert	71.55±0.16	72.01±0.15	72.79±0.23	75.72±0.13
MixPert	**71.68±0.14**	**72.18±0.19**	**72.93±0.24**	**75.80±0.14**

6.2.3 Experiments on ImageNet. LogPert, MixPert, and Meta-MixPert are also evaluated on a large-scale dataset ImageNet which consists of 1.2M training images and 50K validation images in 1K categories. Resnet-50 and ResNet-101 [27] are used as base neural networks.

We train all the neural networks for 250 epochs using a batch size of 256. SGD is uesd as the optimizer. The initial learning rate is 0.1 and decayed by a factor of 0.1 at the 75th, 150th, and 225th epochs. The parameter settings are the same as the settings on CIFAR-10.

Top-1 and top-5 errors are used to assess the classification performance. The experimental results are shown in Tabels 11, 12 and 13. MixPert and LogPert still achieve the best and the second-best performance, respectively. The meta-learning strategies further improve the performance of the MixPert. Experiments on ImageNet still demonstrate the effectiveness of our methods LogPert, MixPert and Meta-MixPert.

Table 8. Classification accuracies (%) and standard deviations on CIFAR-100 (20% pair noise) when using different base neural networks.

	ResNet-32	ResNet-44	ResNet-56	ResNet-110
Base	62.46±0.54	62.73±0.64	63.37±0.22	67.51±0.19
Soft Bootstrapping	63.09±0.33	63.69±0.39	64.06±0.28	67.87±0.26
Hard Bootstrapping	63.03±0.41	63.57±0.32	63.99±0.34	67.40±0.23
Label Smoothing	64.45±0.28	65.72±0.27	66.50±0.74	69.43±0.36
Online Label Smoothing	63.94±0.66	65.18±0.70	65.45±0.52	68.38±0.34
ProSelfLC	64.04±0.37	65.04±0.44	65.48±0.46	68.86±0.26
PGD-AT	60.13±0.31	60.58±0.28	60.96±0.29	65.62±0.20
DLB	63.11±0.37	64.01±0.33	64.26±0.45	68.13±0.35
MbLS	64.55±0.24	65.74±0.29	66.61±0.37	69.39±0.26
LogPert	66.67±0.25	67.16±0.24	68.69±0.23	71.83±0.19
MixPert	**67.14±0.23**	**69.07±0.26**	**69.35±0.20**	**71.86±0.18**

Table 9. Classification accuracies (%) and standard deviations on CIFAR-100 (WRN-28-10).

		Random noise			Pair noise		
	0%	10%	20%	30%	10%	20%	30%
Base (WRN-28-10)	81.53±0.09	78.51±0.18	75.84±0.23	72.95±0.31	78.92±0.16	76.37±0.25	72.31±0.29
Soft Bootstrapping	81.74±0.26	78.73±0.33	75.92±0.29	72.98±0.29	78.98±0.22	76.41±0.33	72.74±0.27
Hard Bootstrapping	81.86±0.34	78.90±0.29	75.89±0.32	73.06±0.28	78.95±0.18	76.44±0.25	72.65±0.24
Label Smoothing	82.07±0.16	79.03±0.24	76.77±0.27	73.54±0.19	79.34±0.17	76.82±0.28	73.11±0.35
Online Label Smoothing	82.02±0.23	78.97±0.19	76.79±0.22	73.62±0.21	79.18±0.24	76.79±0.33	73.02±0.31
ProSelfLC	81.95±0.14	79.12±0.21	76.81±0.17	73.31±0.15	79.52±0.26	76.79±0.34	73.27±0.28
PGD-AT	79.42±0.12	76.23±0.19	73.48±0.24	70.48±0.17	76.51±0.25	74.25±0.20	70.52±0.24
DLB	82.05±0.25	79.01±0.27	76.54±0.28	73.29±0.22	79.11±0.21	76.67±0.31	72.86±0.27
MbLS	82.24±0.23	79.59±0.24	76.93±0.25	73.77±0.27	79.63±0.17	77.41±0.29	73.49±0.23
LogPert	83.32±0.10	81.05±0.22	78.01±0.19	75.11±0.24	81.07±0.15	79.20±0.26	76.03±0.20
MixPert	**83.39±0.07**	**81.14±0.15**	**78.10±0.26**	**75.27±0.21**	**81.29±0.11**	**79.25±0.27**	**76.41±0.18**

Table 10. Classification accuracies (%) and standard deviations on CIFAR-100 (ResNet-110 and WRN-28-10).

	ResNet-110		WRN-28-10	
	0%	20% pair noise	0%	20% pair noise
Base	73.18±0.12	67.47±0.22	81.53±0.09	76.41±0.24
MbLS	74.19±0.23	69.42±0.21	82.24±0.23	77.38±0.27
MixPert	75.80±0.14	71.88±0.16	83.39±0.07	79.30±0.29
Meta-MixPert	**76.55±0.15**	**73.27±0.13**	**83.97±0.12**	**80.42±0.25**

6.2.4 Experiments on Clothing 1M. The experiments are also conducted on Clothing 1M, which is a real-world noisy dataset. The Clothing 1M dataset consists of 1M images with noisy labels and additional 50k, 14k, 10k of clean data for training, validation and testing, respectively.

To evaluate LogPert and MixPert, we follow the same experimental settings as previous studies [42, 72]. The ResNet-50 pre-trained on ImageNet is used as the base neural network. We use SGD with a momentum of 0.9 and a weight decay of 10^{-3}. The batch size is set to 32. For LogPert, MixPert, and other competing methods, the parameter settings are the same as the settings in 6.2.1. The results are shown in Table 14. The proposed MixPert achieves the best results, and LogPert achieves the second-best results. Compared with the base neural network, the MixPert and LogPert improved by 5.70% and 5.46%, respectively. To evaluate Meta-MixPert, the settings in [81] are followed. The results shown in Table 15 demonstrate meta-learning strategies further improve the performance of the MixPert method.

Table 11. Top-1 and Top-5 Errors (%) on ImageNet (0% noise). * denotes the results reported in online label smoothing [91].

	Top-1 Error(%)	Top-5 Error(%)
Base (ResNet-50)	23.68*	7.05*
Soft Bootstrapping	23.49*	6.85*
Hard Bootstrapping	23.85*	7.07*
Label Smoothing	22.82*	6.66*
Online Label Smoothing	22.28*	6.39*
ProSelfLC	23.15	6.74
PGD-AT	24.93	7.33
DLB	23.42	6.79
MbLS	22.26	6.36
LogPert	21.82	6.13
MixPert	**21.79**	**6.10**

Table 12. Top-1 and Top-5 Errors (%) on ImageNet (0% noise).* denotes the results reported in online label smoothing [91].

	Top-1 Error(%)	Top-5 Error(%)
Base (ResNet-101)	21.87*	6.29*
Soft Bootstrapping	21.61	6.18
Hard Bootstrapping	21.92	6.30
Label Smoothing	21.27*	5.85*
Online Label Smoothing	20.85*	5.50*
ProSelfLC	21.43	5.97
PGD-AT	22.92	6.53
DLB	21.59	6.15
MbLS	20.83	5.46
LogPert	20.48	5.35
MixPert	**20.43**	**5.31**

Table 13. Top-1 and Top-5 Errors (%) on ImageNet.

	ResNet-50		ResNet-101	
	Top-1 Error(%)	Top-5 Error(%)	Top-1 Error(%)	Top-5 Error(%)
Base	23.68	7.05	21.87	6.29
MbLS	22.26	6.36	20.83	5.46
MixPert	21.79	6.10	20.43	5.31
Meta-MixPert	**21.42**	**5.89**	**20.02**	**5.14**

6.3 Text Sentiment Analysis

A benchmark dataset is used, namely, IMDB [52]. It is a large internet movie dataset for binary classification tasks with 50k labeled reviews. The proportion of training, validation, and test data we used is 4:1:5. Two types of label noises are added. In the first type (symmetric), the labels of the former 5%, 10%, and 20% (according to their indexes in the corpus) training samples are flipped to simulate the label noises; in the second type (asymmetric), the labels of the former 5%, 10%, and 20% (according to their indexes in the corpus) positive samples are flipped to negative.

BiLSTM with attention and BERT-Base are used as base models. For BiLSTM with attention, the 300-D Glove [94] embedding is used; the embedding dropout and the dimension of hidden vectors are set to 0.5 and 100, respectively. The learning rates for BiLSTM with attention and BERT-Base are set to 1e-3 and 2e-5, respectively. For both models, the batch size is set to 64 and the #epochs is set to 6. AdamW is uesd as the optimizer. In LogPert, the learning rate for the perturbation variable is searched in {0.75, 0.8, 0.85}, and the λ is searched in {0.75, 1}. In MixPert, ϵ_1 ($i.e.$, η)

Table 14. Classification accuracies (%) on Clothing 1M.

	Accuracy
Base(ResNet-50)	69.19
Soft Bootstrapping	70.26
Hard Bootstrapping	70.77
Label Smoothing	71.81
Online Label Smoothing	71.79
ProSelfLC	71.80
PGD-AT	68.02
DLB	71.76
MbLS	72.14
LogPert	74.65
MixPert	**74.89**

Table 15. Classification accuracies (%) on Clothing 1M.

	Accuracy
Base(ResNet-50)	69.31
MbLS	72.32
MixPert	74.92
Meta-MixPert	**75.41**

is searched in {0, 0.05, 0.1, 0.15}, and ϵ_2 is searched in {0, 0.005, 0.01, 0.02}. τ is determined by the top-*pro* percent of ordered losses, and the value of *pro* is searched in {0, 5, 15, 30, 60}. In PGD-AT, ϵ_2 is is searched in {0.005, 0.01, 0.02}. For other competing methods, namely, soft/hard Bootstrapping, LS, OLS, DLB, and MbLS, we follow the parameter settings in the original papers.

Table 16. Classification accuracies (%) on IMDB.

		Symmetric noise			Asymmetric noise		
	0%	5%	10%	20%	5%	10%	20%
Base (BiLSTM+attention)	84.39±0.34	83.04±0.17	81.90±0.61	78.13±0.13	82.35±0.88	79.53±2.68	73.74±1.14
Soft Bootstrapping	84.79±0.87	83.87±0.13	81.11±0.62	79.60±1.78	83.36±1.11	80.70±2.19	73.52±2.65
Hard Bootstrapping	84.44±0.93	84.10±0.54	83.01±0.70	80.84±1.07	82.48±1.72	81.42±1.55	75.26±1.02
Label Smoothing	84.62±0.18	83.14±0.24	82.41±0.51	80.73±0.20	82.75±0.29	82.28±0.33	74.70±0.48
Online Label Smoothing	84.83±0.51	84.14±0.37	82.09±0.54	80.91±1.17	83.78±0.77	81.35±0.92	73.75±1.38
ProSelfLC	84.79±0.39	83.21±0.44	82.17±0.47	80.42±0.41	83.22±0.91	81.58±0.85	74.96±3.01
PGD-AT	85.82±0.10	84.12±0.37	83.53±0.44	81.48±0.18	82.41±0.98	80.75±0.73	73.85±2.33
DLB	84.77±0.41	83.95±0.22	83.16±0.59	80.87±0.67	83.39±0.81	81.37±0.76	75.01±1.12
MbLS	84.85±0.26	84.06±0.17	83.45±0.31	81.49±0.46	83.81±0.29	81.87±0.33	75.29±1.09
LogPert	85.91±0.11	84.57±0.15	83.81±0.24	81.75±0.18	84.64±0.29	82.43±0.31	77.16±0.28
MixPert	**85.96±0.07**	**85.21±0.12**	**84.45±0.21**	**82.74±0.15**	**85.37±0.22**	**83.24±0.23**	**77.83±0.25**

The results of the competing methods on the IMDB for the symmetric and asymmetric label noises are shown in Table 16, when BiLSTM with attention [20] is used as the base network. Our proposed method, MixPert, achieves the overall best results. When no added label noises are present (0%), MixPert and LogPert still outperform the base model BiLSTM with attention by 1.57% and 1.52%, respectively.

On IMDB, the base model is usually converged in the second epoch. However, LogPert is usually converged in the third or the fifth epoch. The validation accuracies of the six epochs for the base model and our LogPert are shown in Fig. 8. LogPert can decelerate the convergence speed leading that the training data can be more fully trained.

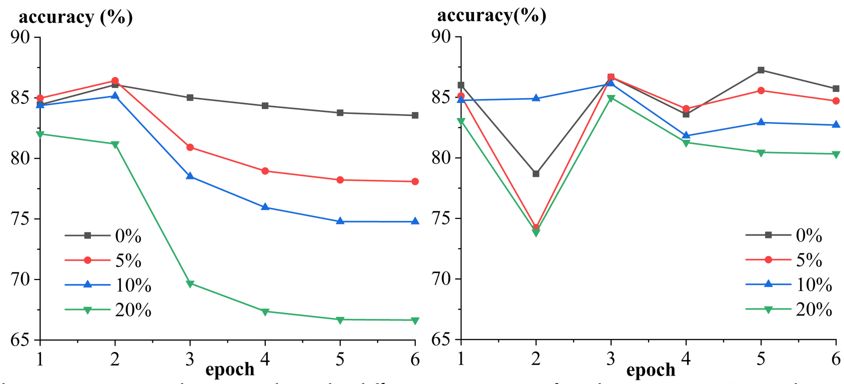

Fig. 8. The validation accuracies in the six epochs under different proportions of random noises on IMDB when using Base (left) and LogPert (right), respectively.

When LogPert is used, some original labels with high average perturbation terms are found to be erroneous. For example, the sentence "*this is a great movie. I love the series on tv and so I loved the movie. One of the best things in the movie is that Helga finally admits her deepest darkest secret to Arnold!!! that was great. i loved it it was pretty funny too. It's a great movie! Doy!!!*" is labeled as negative in the original set.

When BERT-Base [15] is used as the base model, we conduct experiments on the IMDB dataset with 0% noise, 10% symmetric noise, and 10% asymmetric noise. The experimental results are shown in Table 17. Our proposed method, MixPert, still achieves the overall best results. LogPert achieves the second-best results.

Table 17. Classification accuracies (%) on IMDB with BERT.

	0% noise	10% symmetric noise	10% asymmetric noise
Base (BERT)	90.61±0.04	89.26±0.20	89.04±0.37
Soft Bootstrapping	90.72±0.13	89.47±0.27	89.28±0.39
Hard Bootstrapping	90.70±0.09	89.38±0.29	89.26±0.40
Label Smoothing	90.89±0.11	89.61±0.17	89.35±0.21
Online Label Smoothing	90.95±0.07	89.59±0.22	89.42±0.27
ProSelfLC	90.90±0.08	89.54±0.19	89.37±0.18
PGD-AT	91.51±0.06	89.34±0.15	89.30±0.20
DLB	90.77±0.10	89.49±0.24	89.32±0.29
MbLS	91.13±0.09	89.81±0.22	89.68±0.24
LogPert	91.69±0.05	90.53±0.14	90.38±0.18
MixPert	**91.83±0.04**	**90.59±0.11**	**90.50±0.16**

6.4 Ablation Study

6.4.1 Ablation Study for MixPert. An ablation study is conducted for MixPert on CIFAR-10 (random noises) as MixPert involves both positive and negative perturbations. The results in Table 18 indicate that negative perturbation (i.e., adversarial training) does not improve the performance yet the positive perturbation achieves the best performance. Table 19 lists the clean and adversarial accuracies of MixPert under different values of ϵ_2 on the CIFAR-10 (10% random noises). The increase of ϵ_2 improves the adversarial accuracies yet reduces the clean accuracies. Although negative perturbation in MixPert does not improve the clean accuracies, it benefits the adversarial accuracies.

Table 18. An ablation study of MixPert on CIFAR-10 (%).

Random noise	0%	10%	20%	30%
Baseline (ResNet-20)	91.79±0.31	88.78±0.33	87.55±0.32	85.85±0.37
Only pos. pert. ($\epsilon_2 = 0$)	**92.94±0.06**	**91.09±0.11**	**90.63±0.12**	**88.98±0.15**
Only neg. pert. ($\epsilon_1 = 0$)	91.66±0.12	88.69±0.22	87.33±0.10	85.71±0.31
Both directions	92.02±0.15	90.11±0.27	89.75±0.17	88.17±0.16

Table 19. Performance variations under different values of ϵ_2.

	0	2/255	4/255	6/255	8/255
Clean accuracy(%)	91.09±0.11	90.11±0.27	89.77±0.18	88.67±0.15	88.30±0.19
Adversarial accuracy(%)	11.57±0.36	53.38±0.31	64.95±0.24	68.20±0.16	70.25±0.14

An ablation study is also conducted for MixPert on IMDB. The results are shown in Table 20. Each perturbation is useful and their combination achieves the best performance.

Table 20. An ablation study of MixPert on IMDB (%).

Symmetric noise	0%	5%	10%	20%
Baseline (BiLSTM+attention)	84.39±0.34	83.04±0.17	81.90±0.61	78.13±0.13
Only pos. pert. ($\epsilon_2 = 0$)	85.92±0.16	84.66±0.13	83.86±0.24	81.71±0.23
Only neg. pert. ($\epsilon_1 = 0$)	85.84±0.36	84.87±0.22	83.89±0.27	81.73±0.24
Both directions	**85.96±0.07**	**85.21±0.12**	**84.45±0.21**	**82.74±0.15**

6.4.2 Ablation Study for Meta-MixPert. An ablation study is conducted for Meta-MixPert on CIFAR-100. The results presented in Table 21 indicate that the training loss, gradient norm, and functional margin all contribute significantly to the performance of Meta-MixPert.

Table 21. An ablation study of Meta-MixPert on CIFAR-100 (%).

	0%	20% pair noise
Meta-MixPert(WRN-28-10)	83.97±0.12	80.42±0.25
Meta-MixPert without training loss	83.26±0.27	79.43±0.21
Meta-MixPert without gradient norm	83.39±0.18	79.66±0.23
Meta-MixPert without functional margin	83.11±0.13	79.35±0.19

6.4.3 Impact of Hyper-Parameters. In LogPert, the effect of λ on the results is analyzed on CIFAT-100 (pair noises), and the results are shown in Fig. 9. The best results are obtained when λ is set to 0.175 or 0.35. When the value of λ is greater than 0.35, the accuracy gradually decreases. A moderate value of λ can balance the original loss and the degree of logit perturbation.

In MixPert, the effect of ϵ_1 ($i.e.$, η) and τ (the value of pro) on the results is analyzed on IMDB (asymmetric noises), and the results are shown in Fig. 10. We observe that when ϵ_1 is greater than 0.1, the accuracy gradually decreases. As the noise percentage increases, the value of pro for the best results is larger.

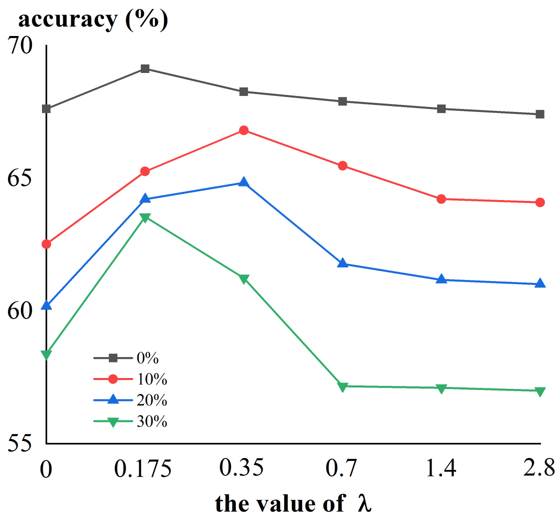

Fig. 9. Accuracies under different λ values in LogPert.

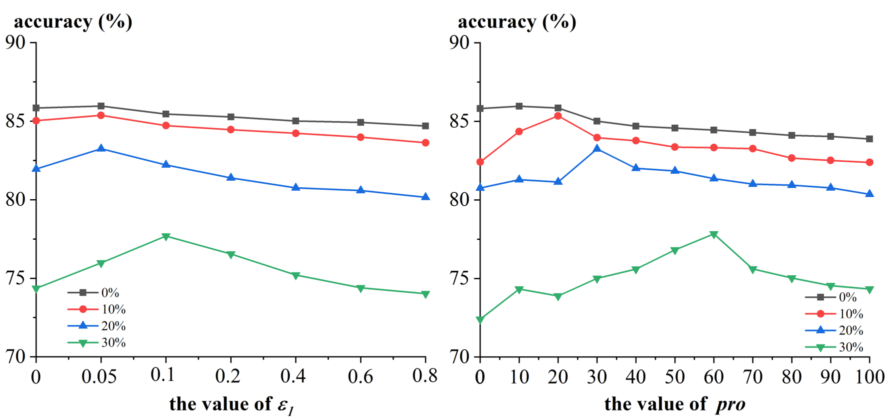

Fig. 10. Accuracies under different values of ϵ_1 (left) and pro in MixPert.

6.5 Discussion

The results indicate that our proposed two methods LogPert and MixPert achieve competitive performances among the competing methods. MixPert is superior to LogPert in most cases. The reason lies in that LogPert only implements positive perturbation in order to reduce the negative influence of samples with noisy or quite hard labels. However, MixPert can implement both positive and negative perturbations. Its positive perturbation part plays a quite similar role as LogPert, whereas its negative perturbation part plays a role of implicit data augmentation. Further, the extent of negative perturbation is controlled by the value of ϵ_2. When $\epsilon_2 = 0$, MixPert is approximately reduced to LogPert. Naturally, MixPert can achieve better results than LogPert in real use.

More extensions and new methods can be obtained based on our taxonomy.

(1) The extension of the logit perturbation (described in Eq. (56)). As previously mentioned, each weighting method may correspond to a perturbation method. Self-paced learning (SPL) [37] is a classical sample weighting strategy in

machine learning. The weights are obtained with the following objective function:

$$\min_{w_i \in \{0,1\}} \sum\nolimits_i w_i l(\mathbb{S}(u_i), y_i) - \lambda w_i. \tag{63}$$

The solution is

$$w_i = \begin{cases} 1 & \text{if } l(\mathbb{S}(u_i), y_i) \leqslant \lambda \\ 0 & \text{otherwise} \end{cases}, \tag{64}$$

which indicates that the weights of samples with larger losses than λ are set to 0. When the value of λ is increased, more samples will participate in the model training.

Fig. 11 shows the curves of weights for the original SPL and its variants. LogPert can be used to implement the SPL with (56) and (65) when the hyper-parameters ϵ and τ satisfy the following conditions:

$$\tau^{t+1} > \tau^t \text{ and } \epsilon > 2\max_i\{||u_i||\}, \tag{65}$$

where t is the index of the current epoch. A new method is obtained and can be called self-paced logit perturbation.

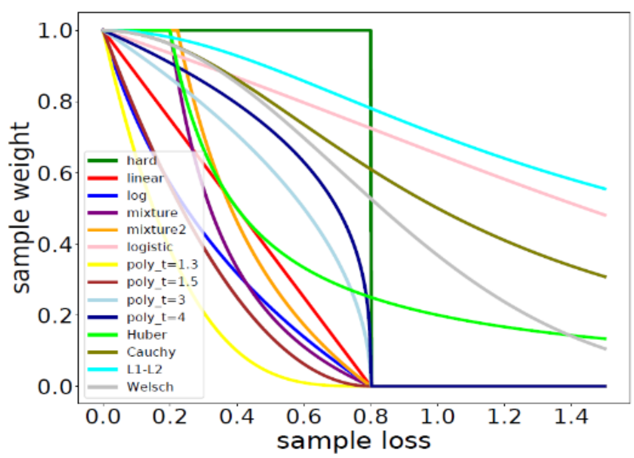

Fig. 11. The curves of weights under different losses in SPL. "Hard" represents the original SPL [82].

With Eq. (65), similar curves to those of SPL can also be obtained. Fig. 12 shows the curve of loss ratios (perturbed loss : original loss) when $\epsilon > 2\max_i\{||u_i||\}$ on the CIFAR-100 dataset. The curve indicates that our strategy can also exert higher weights (= 1) to samples with low losses and lower weights (≈ 0) to samples with high losses.

(2) The extension of MixPert. Indeed, the parameters ϵ_1 and ϵ_2 characterize the extent of positive and negative perturbations, respectively. Intuitively, a sample with a larger loss should have a greater positive perturbation; while a sample with a lower loss should have a greater negative perturbation. Therefore, the constrains for the perturbation terms in (52) can be redefined as follows:

$$\|v_i\| \leq \epsilon_1[1 + (l_i - \tau)/\tau] \text{ and } \|\delta_i\| \leq \epsilon_2[1 + (\tau - l_i)/\tau]. \tag{66}$$

(3) The extension of Bootstrapping. The Bootstrapping loss and the online label smoothing can be unified into the following new loss:

$$\mathcal{L} = \sum\nolimits_i l(p_i, y_i + \alpha(\beta \widetilde{p}_{y_i} + (1-\beta)p_i - y_i)), \tag{67}$$

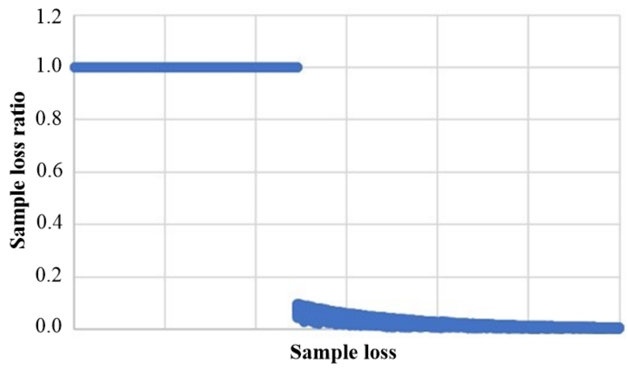

Fig. 12. Loss ratio curve of self-paced logit perturbation given a fixed η (ϵ) and τ.

where $\widetilde{p}_{y_i}$ is the category-level average prediction in the previous epoch; α and β are hyper-parameters and are located in [0, 1]. When β equals 0, the above loss becomes the soft Bootstrapping loss. When β equals 1, the loss becomes the online label smoothing loss with a little difference. Specifically, $\tilde{p}_{y_i}$ is defined as follows:

$$\tilde{p}_{y_i} = \frac{1}{Z_{y_i}} \sum_{j: y_j = y_i} (conf_j \times p_j), \tag{68}$$

where $conf_j$ is the prediction confidence of the prediction p_j, and Z_{y_i} is the normalizer. Two typical definitions of $conf_j$ are

$$\begin{aligned} conf_j &= 1 \;\; or \\ conf_j &= \begin{cases} 1 & \text{if the prediction is correct} \\ 0 & \text{otherwise} \end{cases}. \end{aligned} \tag{69}$$

When the second definition is used and $\beta = 1$, the unified loss becomes the online label smoothing. Nevertheless, in most datasets, the values of $\tilde{p}_{y_i}$ obtained by the above two definitions are close to each other as the index of the current epoch gradually increases according to our observations. The unified new method can be called mixBootstrapping.

7 CONCLUSIONS

This study reveals a widely used yet less-explored machine learning strategy, namely, perturbation. Machine learning methods leveraging or partially leveraging perturbation comprise a new learning paradigm called learning with perturbation. To solidify the theoretical basis of perturbation learning, a systematic taxonomy is constructed on the basis of which to perturb, the direction of loss variation, how to infer, and the granularity. To demonstrate the universality of perturbation learning, several existing learning methods are explained within our constructed taxonomy. Furthermore, three concrete perturbation learning methods (i.e., LogPert, MixPert, and Meta-MixPert) are proposed. Extensive experiments suggest that our proposed methods are effective in robust learning tasks. It is believable that our constructed taxonomy can build intrinsic connections among a large number of seemly unrelated learning methods, enlighten the deep understanding of these methods, and inspire the design of more effective methods.

ACKNOWLEDGEMENT

We thank Mr. Mengyang Li for his useful suggestions on the experiments.

REFERENCES

[1] M. Bazaraa, H. Sherali, and C. Shetty. 1993. *Nonlinear Programming - Theory and Algorithms*. John Wiley and Sons, Inc.

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML*. 41–48.

[3] Philipp Benz, Chaoning Zhang, Adil Karjauv, and In So Kweon. 2021. Universal adversarial training with class-wise perturbations. In *ICME*. 1–6.

[4] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. 2019. Learning imbalanced datasets with label-distribution-aware margin loss. In *NeurIPS*. 1565–1576.

[5] Kuang-Yu Chang, Chu-Song Chen, and Yi-Ping Hung. 2011. Ordinal hyperplanes ranker with cost sensitivities for age estimation. In *CVPR*. 585–592.

[6] Ashutosh Chaubey, Nikhil Agrawal, Kavya Barnwal, Keerat K. Guliani, and Pramod Mehta. 2020. Universal Adversarial Perturbations: A Survey. In *arXiv2005.08087*.

[7] Dubing Chen, Yuming Shen, Haofeng Zhang, and Philip HS Torr. 2022. Zero-Shot Logit Adjustment. *arXiv preprint arXiv:2204.11822* (2022).

[8] Yoojin Choi, Jihwan Choi, Mostafa El-Khamy, and Jungwon Lee. 2020. Data-free network quantization with adversarial knowledge distillation. In *CVPR Workshops*. 3047–3057.

[9] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning* 20, 3 (1995), 273–297.

[10] Jiequan Cui, Shu Liu, Zhuotao Tian, Zhisheng Zhong, and Jiaya Jia. 2023. ResLT: Residual Learning for Long-Tailed Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 3 (2023), 3695–3706.

[11] D. Das and Csg Lee. 2019. A Two-Stage Approach to Few-Shot Learning for Image Recognition. *IEEE Transactions on Image Processing* 29, 99 (2019), 3336–3350.

[12] Antoine de Mathelin, Francois Deheeger, Mathilde Mougeot, and Nicolas Vayatis. 2023. Deep Anti-Regularized Ensembles provide reliable out-of-distribution uncertainty quantification. In *arXiv:2304.04042*.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *CVPR*. 248–255.

[14] Jia Deng, Jonathan Krause, and Li Fei-Fei. 2013. Fine-grained crowdsourcing for fine-grained recognition. In *CVPR*. 580–587.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*. 4171–4186.

[16] Oussama Dhifallah and Yue Lu. 2021. On the Inherent Regularization Effects of Noise Injection During Training. In *ICML*. 2665–2675.

[17] Xiaozhao Fang, Yong Xu, Xuelong Li, Zhihui Lai, Wai Keung Wong, and Bingwu Fang. 2017. Regularized label relaxation linear regression. *IEEE Transactions on neural networks and learning systems* 29, 4 (2017), 1006–1018.

[18] Pedro A Forero, Vassilis Kekatos, and Georgios B Giannakis. 2012. Robust clustering using outlier-sparsity regularization. *IEEE Transactions on Signal Processing* 60, 8 (2012), 4163–4177.

[19] Yoav Freund and Robert E Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences* 55, 1 (1997), 119–139.

[20] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with LSTM. *Neural Computation* 12, 10 (2000), 2451–2471.

[21] Morgane Goibert and Elvis Dohmatob. 2019. Adversarial robustness via label-smoothing. *arXiv preprint arXiv:1906.11567* (2019).

[22] Jacob Goldberger and Ehud Ben-Reuven. 2017. Training deep neural-networks using a noise adaptation layer. In *ICLR*.

[23] Sheng Guo, Weilin Huang, Haozhi Zhang, Chenfan Zhuang, Dengke Dong, Matthew R Scott, and Dinglong Huang. 2018. Curriculumnet: Weakly supervised learning from large-scale web images. In *ECCV*. 139–154.

[24] Y. Hamamoto, Y. Mitani, H. Ishihara, T. Hase, and S. Tomita. 1996. Evaluation of an anti-regularization technique in neural networks. In *ICPR*. 205–208.

[25] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor W Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *NeurIPS*. 8536–8546.

[26] Jiangfan Han, Ping Luo, and Xiaogang Wang. 2019. Deep self-learning from noisy labels. In *ICCV*. 5137–5146.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. 770–778.

[28] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[29] Hidekata Hontani, Takamiti Matsuno, and Yoshihide Sawada. 2012. Robust nonrigid ICP using outlier-sparsity regularization. In *CVPR*. 174–181.

[30] Ruibing Hou, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. 2023. Dual Compensation Residual Networks for Class Imbalanced Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 10 (2023), 11733–11752.

[31] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. 2016. Learning deep representation for imbalanced classification. In *CVPR*. 5375–5384.

[32] Jinchi Huang, Lie Qu, Rongfei Jia, and Binqiang Zhao. 2019. O2u-net: A simple noisy label detection approach for deep neural networks. In *ICCV*. 3325–3333.

[33] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. 2019. Adversarial Examples Are Not Bugs, They Are Features. In *NeurIPS*.

[34] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*. 2309–2318.

[35] Justin M Johnson and Taghi M Khoshgoftaar. 2019. Survey on deep learning with class imbalance. *Journal of Big Data* 6, 1 (2019), 1–54.

[36] Alex Krizhevsky and Geoffrey Hinton. 2009. *Learning multiple layers of features from tiny images.* Technical Report.
[37] M Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. In *NeurIPS*. 1189–1197.
[38] Pablo Lanillos, Daniel Oliva, Anja Philippsen, Yuichi Yamashita, Yukie Nagai, and Gordon Cheng. 2020. A review on neural network models of schizophrenia and autism spectrum disorder. *Neural Networks* 122 (2020), 338–363.
[39] Wonseok Lee, Hanbit Lee, and Sang-goo Lee. 2020. Semantics-Preserving Adversarial Training. *arXiv preprint arXiv:2009.10978* (2020).
[40] Zhaoqi Leng, Mingxing Tan, Chenxi Liu, Ekin Dogus Cubuk, Jay Shi, Shuyang Cheng, and Dragomir Anguelov. 2021. PolyLoss: A Polynomial Expansion Perspective of Classification Loss Functions. In *ICLR*.
[41] Buyu Li, Yu Liu, and Xiaogang Wang. 2019. Gradient harmonized single-stage detector. In *AAAI*. 8577–8584.
[42] Junnan Li, Yongkang Wong, Qi Zhao, and Mohan S Kankanhalli. 2019. Learning to learn from noisy labeled data. In *CVPR*. 5051–5059.
[43] Mingchen Li, Xuechen Zhang, Christos Thrampoulidis, Jiasi Chen, and Samet Oymak. 2021. AutoBalance: Optimized Loss Functions for Imbalanced Data. *NeurIPS* 34 (2021).
[44] Shuang Li, Kaixiong Gong, Chi Harold Liu, Yulin Wang, Feng Qiao, and Xinjing Cheng. 2021. MetaSAug: Meta Semantic Augmentation for Long-Tailed Visual Recognition. In *CVPR*. 5212–5221.
[45] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *ICCV*. 2999–3007.
[46] Bingyuan Liu, Ismail Ben Ayed, Adrian Galdran, and Jose Dolz. 2022. The Devil is in the Margin: Margin-based Label Smoothing for Network Calibration. In *CVPR*. 80–88.
[47] Hong Liu, Rongrong Ji, Jie Li, Baochang Zhang, Yue Gao, Yongjian Wu, and Feiyue Huang. 2019. Universal adversarial perturbation via prior driven uncertainty approximation. In *ICCV*. 2941–2949.
[48] Hao Liu, Xiangyu Zhu, Zhen Lei, and Stan Z Li. 2019. Adaptiveface: Adaptive margin and sampling for face recognition. In *CVPR*. 11947–11956.
[49] Tongliang Liu and Dacheng Tao. 2016. Classification with noisy labels by importance reweighting. *IEEE Transactions on pattern analysis and machine intelligence* 38, 3 (2016), 447–461.
[50] Gengyu Lyu, Songhe Feng, Tao Wang, and Congyan Lang. 2022. A self-paced regularization framework for partial-label learning. *IEEE Transactions on Cybernetics* 52, 2 (2022), 899–911.
[51] Xingjun Ma and et al. 2018. Dimensionality-driven learning with noisy labels. In *ICML*. 3361–3370.
[52] Andrew Maas and et al. 2011. Learning word vectors for sentiment analysis. In *ACL*. 142–150.
[53] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *ICLR*.
[54] Zhijun Mai, Guosheng Hu, Dexiong Chen, Fumin Shen, and Heng Tao Shen. 2021. Metamixup: Learning adaptive interpolation policy of mixup with metalearning. *IEEE Transactions on Neural Networks and Learning Systems* (2021).
[55] Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. 2021. Long-tail learning via logit adjustment. In *ICLR*.
[56] Jan Hendrik Metzen, Nicole Finnie, and Robin Hutmacher. 2021. Meta Adversarial Training against Universal Patches. In *ICML 2021 Workshop on Adversarial Machine Learning*.
[57] Seyed-Mohsen Moosavi-Dezfooli, Omar Fawzi, Alhussein amd Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *CVPR*. 86–94.
[58] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep Ravikumar, and Ambuj Tewari. 2013. Learning with noisy labels. In *NeurIPS*. 1196–1204.
[59] Federico Nesti, Alessandro Biondi, and Giorgio Buttazzo. 2021. Detecting adversarial examples by input transformations, defense perturbations, and voting. *IEEE Transactions on Neural Networks and Learning Systems* (2021).
[60] Nam H Nguyen and Trac D Tran. 2012. Robust lasso with missing and grossly corrupted observations. *IEEE transactions on information theory* 59, 4 (2012), 2036–2058.
[61] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing Neural Networks by Penalizing Confident Output Distributions. In *arXiv:1701.06548*.
[62] Scott Reed and et al. 2015. Training deep neural networks on noisy labels with bootstrapping. In *ICLR*.
[63] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2018. Learning to reweight examples for robust deep learning. In *ICML*. 4331–4340.
[64] Bernhard Schölkopf, Alex J Smola, Robert C Williamson, and Peter L Bartlett. 2000. New support vector algorithms. *Neural computation* 12, 5 (2000), 1207–1245.
[65] Ali Shafahi, Mahyar Najibi, Zheng Xu, John Dickerson, Larry S Davis, and Tom Goldstein. 2020. Universal adversarial training. In *AAAI*. 5636–5643.
[66] Haojing Shen, Sihong Chen, Ran Wang, and Xizhao Wang. 2022. Adversarial Learning with Cost-Sensitive Classes. *IEEE Transactions on Cybernetics* (2022).
[67] Yiqing Shen, Liwu Xu, Yuzhe Yang, Yaqian Li, and Yandong Guo. 2022. Self-Distillation from the Last Mini-Batch for Consistency Regularization. In *CVPR*. 11943–11952.
[68] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. 2019. Meta-Weight-Net: Learning an explicit mapping For sample weighting. In *NeurIPS*. 1917–1928.
[69] Martin Slawski, Emanuel Ben-David, et al. 2019. Linear regression with sparsely permuted data. *Electronic Journal of Statistics* 13, 1 (2019), 1–36.
[70] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. 2022. Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[71] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *CVPR*. 2818–2826.

[72] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. 2018. Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5552–5560.

[73] Sunil Thulasidasan, Tanmoy Bhattacharya, Jeff Bilmes, Gopinath Chennupati, and Jamal Mohd-Yusof. 2019. Combating label noise in deep learning using abstention. In *ICML*. 6234–6243.

[74] Yuanpeng Tu, Boshen Zhang, Yuxi Li, Liang Liu, Jian Li, Yabiao Wang, Chengjie Wang, and Cai Rong Zhao. 2023. Learning from noisy labels with decoupled meta label purifier. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 19934–19943.

[75] Derui Wang, Chaoran Li, Sheng Wen, Qing-Long Han, Surya Nepal, Xiangyu Zhang, and Yang Xiang. 2021. Daedalus: Breaking nonmaximum suppression in object detection via adversarial examples. *IEEE Transactions on Cybernetics* (2021).

[76] Mei Wang, Yaobin Zhang, and Weihong Deng. 2021. Meta Balanced Network for Fair Face Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[77] Xinshao Wang, Yang Hua, Elyor Kodirov, David A Clifton, and Neil M Robertson. 2021. Proselflc: Progressive self label correction for training robust deep neural networks. In *CVPR*. 752–761.

[78] Yixin Wang, Alp Kucukelbir, and David M Blei. 2017. Robust probabilistic modeling with bayesian data reweighting. In *ICML*. 3646–3655.

[79] Yulin Wang, Xuran Pan, Shiji Song, Hong Zhang, Cheng Wu, and Gao Huang. 2019. Implicit semantic data augmentation for deep networks. In *NeurIPS*. 12614–12623.

[80] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. 2020. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)* 53, 3 (2020), 1–34.

[81] Zhen Wang, Guosheng Hu, and Qinghua Hu. 2020. Training noise-robust deep neural networks via meta-Learning. In *CVPR*. 4523–4532.

[82] Xiaoxia Wu, Ethan Dyer, and Behnam Neyshabur. 2021. When Do Curricula Work?. In *ICLR*.

[83] Yichen Wu, Jun Shu, Qi Xie, Qian Zhao, and Deyu Meng. 2021. Learning to purify noisy labels via meta soft label corrector. In *AAAI*. 10388–10396.

[84] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. 2015. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2691–2699.

[85] Han Xu, Xiaorui Liu, Yaxin Li, Anil Jain, and Jiliang Tang. 2021. To be Robust or to be Fair: Towards Fairness in Adversarial Training. In *ICML*. 11492–11501.

[86] Han Xu, Xiaorui Liu, Yaxin Li, Anil K. Jain, and Jiliang Tang. 2021. To be Robust or to be Fair: Towards Fairness in Adversarial Training. In *ICML*. 11492–11501.

[87] Erkun Yang, Tongliang Liu, Cheng Deng, and Dacheng Tao. 2020. Adversarial examples for hamming space search. *IEEE transactions on cybernetics* 50, 4 (2020), 1473–1484.

[88] J. Yao, J. Wang, I. W. Tsang, Y. Zhang, J. Sun, C. Zhang, and R. Zhang. 2019. Deep Learning From Noisy Image Labels With Quality Embedding. *IEEE Transactions on Image Processing* 28 (2019), 1909–1922.

[89] Chia-Hung Yuan, Pin-Yu Chen, and Chia-Mu Yu. 2022. Meta Adversarial Perturbations. In *AAAI Workshops*.

[90] Sergey Zagoruyko and Nikos Komodakis. 2016. Wide Residual Networks. In *Procedings of the British Machine Vision Conference 2016*. British Machine Vision Association.

[91] Chang-Bin Zhang, Peng-Tao Jiang, Qibin Hou, Yunchao Wei, Qi Han, Zhen Li, and Ming-Ming Cheng. 2021. Delving deep into label smoothing. *IEEE Transactions on Image Processing* 30 (2021), 5984–5996.

[92] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2018. mixup: Beyond Empirical Risk Minimization. In *ICLR*.

[93] Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. 2020. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)* 11, 3 (2020), 1–41.

[94] Liang Zhao, Tianyang Zhao, Tingting Sun, Zhuo Liu, and Zhikui Chen. 2020. Multi-view robust feature learning for data clustering. *IEEE Signal Processing Letters* 27 (2020), 1750–1754.

[95] Wangchunshu Zhou, Canwen Xu, and Julian McAuley. 2022. BERT learns to teach: Knowledge distillation with meta learning. In *ACL*. 7037–7049.

[96] B. Zhu, Y. Niu, X.-S. Hua, and H. Zhang. 2022. Cross-Domain Empirical Risk Minimization for Unbiased Long-Tailed Classification. In *CVPR AAAI*. 3589–3597.