ProSelfLC: Progressive Self Label Correction for Training Robust Deep Neural Networks

Xinshao Wang[1, 2], Yang Hua[3], Elyor Kodirov[1], David A. Clifton[2, *], Neil M. Robertson[1, 3]

[1]Zenith Ai, UK
[2]Institute of Biomedical Engineering, University of Oxford, UK
[3]Institute of Electronics, Communications and Information Technology, Queen's University Belfast, UK
{xinshao, elyor}@zenithai.co.uk, {y.hua, n.robertson}@qub.ac.uk, {xinshao.wang, david.clifton}@eng.ox.ac.uk

Abstract

To train robust deep neural networks (DNNs), we systematically study several target modification approaches, which include output regularisation, self and non-self label correction (LC). Two key issues are discovered: (1) Self LC is the most appealing as it exploits its own knowledge and requires no extra models. However, how to automatically decide the trust degree of a learner as training goes is not well answered in the literature? (2) Some methods penalise while the others reward low-entropy predictions, prompting us to ask which one is better?

To resolve the first issue, taking two well-accepted propositions–deep neural networks learn meaningful patterns before fitting noise [3] and minimum entropy regularisation principle [10]–we propose a novel end-to-end method named ProSelfLC, which is designed according to learning time and entropy. Specifically, given a data point, we progressively increase trust in its predicted label distribution versus its annotated one if a model has been trained for enough time and the prediction is of low entropy (high confidence). For the second issue, according to ProSelfLC, we empirically prove that it is better to redefine a meaningful low-entropy status and optimise the learner toward it. This serves as a defence of entropy minimisation.

We demonstrate the effectiveness of ProSelfLC through extensive experiments in both clean and noisy settings. The source code is available at `https://github.com/XinshaoAmosWang/ProSelfLC-CVPR2021`.

1. Introduction

There exist many target (label) modification approaches. They can be roughly divided into two groups: (1) Output regularisation (OR), which is proposed to penalise over-confident predictions for regularising deep neural networks. It includes label smoothing (LS) [42, 29] and confidence penalty (CP) [33]; (2) Label correction (LC). On the one

hand, LC regularises neural networks by adding the similarity structure information over training classes into one-hot label distributions so that the learning targets become *structured and soft*. On the other hand, it can *correct the semantic classes* of noisy label distributions. LC can be further divided into two subgroups: Non-self LC and Self LC. The former requires extra learners, while the latter relies on the model itself. A typical approach of Non-self LC is knowledge distillation (KD), which exploits the predictions of other model(s), usually termed teacher(s) [17]. Self LC methods include Pseudo-Label [23], bootstrapping (Boot-soft and Boot-hard) [35], Joint Optimisation (Joint-soft and Joint-hard) [43], and Tf-KD$_{self}$ [55]. According to an overview in Figure 1 (detailed derivation is in Section 3 and Table 1), *in label modification, the output target of a data point is defined by combining a one-hot label distribution and its corresponding prediction or a predefined label distribution*.

Firstly, we present the drawbacks of existing approaches: (1) OR methods naively penalise confident outputs without leveraging easily accessible knowledge from other learners or itself (Figure 1a); (2) Non-self LC relies on accurate auxiliary models to generate predictions (Figure 1b). (3) Self LC is the most appealing because it exploits its own knowledge and requires no extra learners. However, there is a core question that is not well answered:

In Self LC, how much should we trust a learner to leverage its knowledge?

As shown in Figure 1b, in Self LC, for a data point, we have two labels: a predefined one-hot $\mathbf{q}$ and a predicted structured $\mathbf{p}$. Its learning target is $(1 - \epsilon)\mathbf{q} + \epsilon\mathbf{p}$, i.e., a trade-off between $\mathbf{q}$ and $\mathbf{p}$, where ϵ defines the trust score of a learner. In existing methods, ϵ is fixed without considering that a model's knowledge grows as the training progresses. For example, in bootstrapping, ϵ is fixed throughout the training process. Joint Optimisation stage-wisely trains a model. It fully trusts predicted labels and uses them to replace old ones when a stage ends, i.e., $\epsilon = 1$. Tf-KD$_{self}$ trains a model by two stages: $\epsilon = 0$ in the first one while ϵ is tuned for the second stage. Note that $\mathbf{p}$ is gen-

$$\tilde{\mathbf{q}}_{\text{LS}} = (1-\epsilon)\mathbf{q} \; \boxed{+\epsilon\mathbf{u}}$$

$\mathbf{q}$ $\quad$ $\mathbf{u}$

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} \implies \begin{bmatrix} (1-\epsilon)+\epsilon/3 \\ \epsilon/3 \\ \epsilon/3 \end{bmatrix}$$

$1-\epsilon \qquad \epsilon$

LS

$$\tilde{\mathbf{q}}_{\text{CP}} = (1-\epsilon)\mathbf{q} \; \boxed{-\epsilon\mathbf{p}}$$

$\mathbf{q}$ $\quad$ $\mathbf{p}$

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 1/2 \\ 1/3 \\ 1/6 \end{bmatrix} \implies \begin{bmatrix} (1-\epsilon)-\epsilon/2 \\ -\epsilon/3 \\ -\epsilon/6 \end{bmatrix} \Leftrightarrow \begin{bmatrix} (1-\epsilon)-\epsilon/2 \\ 0 \\ 0 \end{bmatrix}$$

$1-\epsilon \qquad \epsilon$

CP
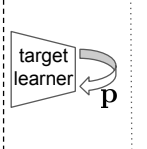
(a) OR includes LS [42] and CP [33]. LS softens a target by adding a uniform label distribution. CP changes the probability 1 to a smaller value $1-\epsilon$ in the one-hot target. The double-ended arrow means factual equivalence, because an output is definitely non-negative after a softmax layer.
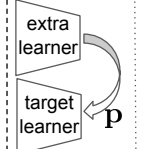
$$\tilde{\mathbf{q}}_{\text{LC}} = (1-\epsilon)\mathbf{q} \; \boxed{+\epsilon\mathbf{p}}$$

target learner $\quad \mathbf{p}$

$\mathbf{q}$ $\quad$ $\mathbf{p}$

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1/2 \\ 1/3 \\ 1/6 \end{bmatrix} \implies \begin{bmatrix} (1-\epsilon)+\epsilon/2 \\ \epsilon/3 \\ \epsilon/6 \end{bmatrix}$$

$1-\epsilon \qquad \epsilon$

Self LC

extra learner / target learner $\quad \mathbf{p}$

$\mathbf{q}$ $\quad$ $\mathbf{p}$

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1/2 \\ 1/6 \\ 1/3 \end{bmatrix} \implies \begin{bmatrix} (1-\epsilon)+\epsilon/2 \\ \epsilon/6 \\ \epsilon/3 \end{bmatrix}$$

$1-\epsilon \qquad \epsilon$

Non-self LC

(b) LC contains Self LC [23, 35, 43, 55] and Non-self LC [17]. The parameter ϵ defines how much a predicted label distribution is trusted.

Figure 1: Target modification includes OR (LS and CP), and LC (Self LC and Non-self LC). Assume there are three training classes. $\mathbf{q}$ is the one-hot target. $\mathbf{u}$ is a uniform label distribution. $\mathbf{p}$ denotes a predicted label distribution. The target combination parameter is $\epsilon \in [0,1]$.

erated by a preceding-stage model in stage-wise training, which requires significant human intervention and is time-consuming in practice.

To improve Self LC, we propose a novel method named Progressive Self Label Correction (ProSelfLC), which is end-to-end trainable and needs negligible extra cost. Most importantly, ProSelfLC modifies the target progressively and adaptively as training goes. *Two design principles of ProSelfLC are*: (1) When a model learns from scratch, human annotations are more reliable than its own predictions in the early phase, during which the model is learning simple meaningful patterns before fitting noise, even when severe label noise exists in human annotations [3]. (2) As a learner attains confident knowledge as time progresses, we leverage it to revise annotated labels. This is surrounded by minimum entropy regularisation, which is widely evaluated in unsupervised and semi-supervised scenarios [9, 10].

Secondly, note that OR methods penalise low entropy while LC rewards it, intuitively leading to a second vital question:

Should we penalise a low-entropy status or reward it?

Entropy minimisation is the most widely used principle in machine learning [14, 38, 9, 10, 22]. In standard classification, minimising categorical cross entropy (CCE) optimises a model towards a low-entropy status defined by human annotations, which contain noise in very large-scale machine learning. As a result, confidence penalty becomes popular for reducing noisy fitting. In contrast, we prove that it is better to reward a meaningful low-entropy status redefined by our ProSelfLC. Therefore, our work offers a defence of

entropy minimisation against the recent confidence penalty practice [42, 29, 33, 6].

Finally, we summarise our main contributions:

- We provide a theoretical study on popular target modification methods through entropy and KL divergence [21]. Accordingly, we reveal their drawbacks and propose ProSelfLC as a solution. ProSelfLC can: (1) enhance the similarity structure information over training classes; (2) correct the semantic classes of noisy label distributions. ProSelfLC is the first method to trust self knowledge progressively and adaptively.

- Our extensive experiments: (1) defend the entropy minimisation principle; (2) demonstrate the effectiveness of ProSelfLC in both clean and noisy settings.

2. Related Work

Label noise and semi-supervised learning. We test target modification approaches in the setting of label noise because it is generic and connected with semi-supervised learning, where only a subset of training examples are annotated, leading to *missing labels*. Then the key to semi-supervised training is to reliably fill them. *When these missing labels are incorrectly filled, the challenge of semi-supervised learning changes to noisy labels.* For a further comparison, in semi-supervised learning, the annotated set is clean and reliable, because the label noise only exists in the unannotated set. While in our experimental setting, we are not given information on whether an example is trusted or not, thus being even more challenging. We summarise

existing approaches for solving label noise: (1) Loss correction, in which we are given or we need to estimate a noise-transition matrix, which defines the distribution of noise labels [26, 8, 41, 44, 52, 12, 32, 48]. A noise-transition matrix is difficult and complex to estimate in practice; (2) Exploiting an auxiliary trusted training set to differentiate examples [45, 24, 16]. This requires extra annotation cost; (3) Co-training strategies, which train two or more learners [28, 19, 13, 54, 47, 34, 25] and exploit their 'disagreement' information to differentiate data points; (4) Label engineering methods [40, 23, 35, 43, 52, 25], which relate to our focus in this work. Their strategy is to annotate unlabelled samples or correct noisy labels.

LC and knowledge distillation (KD) [5, 17]. Mathematically, we derive that some KD methods also modify labels. We use the term label correction instead of KD for two reasons: (1) label correction is more descriptive; (2) the scope of KD is not limited to label modification. For example, multiple networks are trained for KD [7]. When two models are trained, the consistency between their predictions of a data point is promoted in [4, 58], while the distance between their feature maps is reduced in [37]. Regarding self KD, two examples of the same class are constrained to have consistent output distributions [51, 56]. In another self KD [57], the deepest classifier provides knowledge for shallower classifiers. In a recent self KD method [55], Tf-KD_{self} applies two-stage training. In the second stage, a model is trained by exploiting its knowledge learned in the first stage. Our focus is to improve the end-to-end self LC. First, self KD methods [51, 56, 57] maximise the consistency of intraclass images' predictions or the consistency of different classifiers. In our view, they do not modify labels, thus being less relevant for comparison. Second, the two-stage self KD method [55] can be an add-on (i.e., an enhancement plugin) other than a competitor. E.g., in real-world practice, the first stage can be ProSelfLC instead of CCE with early stopping. Finally, we acknowledge that exploiting ProSelfLC to improve non-self KD and stage-wise approaches is an important area for future work, e.g., a better teacher model can be trained using ProSelfLC.

3. Mathematical Analysis and Theory

Let $\mathbf{X} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ represent N training examples, where $(\mathbf{x}_i, y_i)$ denotes $i-$th sample with input $\mathbf{x}_i \in \mathbb{R}^D$ and label $y_i \in \{1, 2, ..., C\}$. C is the number of classes. A deep neural network z consists of an embedding network $f(\cdot) : \mathbb{R}^D \to \mathbb{R}^K$ and a linear classifier $g(\cdot) : \mathbb{R}^K \to \mathbb{R}^C$, i.e., $\mathbf{z}_i = z(\mathbf{x}_i) = g(f(\mathbf{x}_i)) : \mathbb{R}^D \to \mathbb{R}^C$. For the brevity of analysis, we take a data point and omit its subscript so that it is denoted by $(\mathbf{x}, y)$. The linear classifier is usually the last fully-connected layer. Its output is named logit vector $\mathbf{z} \in \mathbb{R}^C$. We produce its classification probabilities $\mathbf{p}$ by normalising the logits using a softmax function:

$$\mathbf{p}(j|\mathbf{x}) = \exp(\mathbf{z}_j)/\sum\nolimits_{m=1}^{C} \exp(\mathbf{z}_m), \qquad (1)$$

where $\mathbf{p}(j|\mathbf{x})$ is the probability of $\mathbf{x}$ belonging to class j. Its corresponding ground-truth is usually denoted by a one-hot representation $\mathbf{q}$: $\mathbf{q}(j|\mathbf{x}) = 1$ if $j = y$, $\mathbf{q}(j|\mathbf{x}) = 0$ otherwise.

3.1. Semantic class and similarity structure in a label distribution

A probability vector $\mathbf{p} \in \mathbb{R}^C$ can also be interpreted as an instance-to-classes similarity vector, i.e., $\mathbf{p}(j|\mathbf{x})$ measures how much a data point $\mathbf{x}$ is similar with (analogously, likely to be) j-th class. Consequently, $\mathbf{p}$ should not be exactly one-hot, and is proposed to be corrected at training, so that it can define a more informative and structured learning target. For better clarity, we first present two definitions:

Definition 1 (*Semantic Class*). Given a target label distribution $\tilde{\mathbf{q}}(\mathbf{x}) \in \mathbb{R}^C$, the semantic class is defined by $\arg\max_j \tilde{\mathbf{q}}(j|\mathbf{x})$, i.e., the class whose probability is the largest.

Definition 2 (*Similarity Structure*). In $\tilde{\mathbf{q}}(\mathbf{x})$, $\mathbf{x}$ has C probabilities of being predicted to C classes. The similarity structure of $\mathbf{x}$ versus C classes is defined by these probabilities and their differences.

3.2. Revisit of CCE, LS, CP and LC

Standard CCE. For any input $(\mathbf{x}, y)$, the minimisation objective of standard CCE is:

$$L_{\text{CCE}}(\mathbf{q}, \mathbf{p}) = \text{H}(\mathbf{q}, \mathbf{p}) = \text{E}_{\mathbf{q}}(-\log \mathbf{p}), \qquad (2)$$

where $\text{H}(\cdot, \cdot)$ represents the cross entropy. $\text{E}_{\mathbf{q}}(-\log \mathbf{p})$ denotes the expectation of negative log-likelihood, and $\mathbf{q}$ serves as the probability mass function.

Label smoothing. In LS [42, 17], we soften one-hot targets by adding a uniform distribution: $\tilde{\mathbf{q}}_{\text{LS}} = (1 - \epsilon)\mathbf{q} + \epsilon\mathbf{u}$, $\mathbf{u} \in \mathbb{R}^C$, and $\forall j, \mathbf{u}_j = \frac{1}{C}$. Consequently:

$$\begin{aligned} L_{\text{CCE+LS}}(\mathbf{q}, \mathbf{p}; \epsilon) &= \text{H}(\tilde{\mathbf{q}}_{\text{LS}}, \mathbf{p}) = \text{E}_{\tilde{\mathbf{q}}_{\text{LS}}}(-\log \mathbf{p}) \\ &= (1 - \epsilon)\text{H}(\mathbf{q}, \mathbf{p}) + \epsilon\text{H}(\mathbf{u}, \mathbf{p}). \end{aligned} \qquad (3)$$

Confidence penalty. CP [33] penalises highly confident predictions:

$$L_{\text{CCE+CP}}(\mathbf{q}, \mathbf{p}; \epsilon) = (1 - \epsilon)\text{H}(\mathbf{q}, \mathbf{p}) - \epsilon\text{H}(\mathbf{p}, \mathbf{p}). \qquad (4)$$

Label correction. As illustrated in Figure 1, LC is a family of algorithms, where a one-hot label distribution is modified to a convex combination of itself and a predicted distribution:

$$\begin{aligned} \tilde{\mathbf{q}}_{\text{LC}} = (1 - \epsilon)\mathbf{q} + \epsilon\mathbf{p} \Rightarrow L_{\text{CCE+LC}}(\mathbf{q}, \mathbf{p}; \epsilon) &= \text{H}(\tilde{\mathbf{q}}_{\text{LC}}, \mathbf{p}) \\ &= (1 - \epsilon)\text{H}(\mathbf{q}, \mathbf{p}) + \epsilon\text{H}(\mathbf{p}, \mathbf{p}). \end{aligned} \qquad (5)$$

We remark: (1) $\mathbf{p}$ provides meaningful information about an example's relative probabilities of being different training classes; (2) If ϵ is large, and $\mathbf{p}$ is confident in predicting a different class, i.e., $\arg\max_j \mathbf{p}(j|\mathbf{x}) \neq \arg\max_j \mathbf{q}(j|\mathbf{x})$, $\tilde{\mathbf{q}}_{\text{LC}}$ defines a different semantic class from $\mathbf{q}$.

Table 1: Summary of CCE, LS, CP and LC.

	CCE	LS	CP	LC
Learning Target	$\mathbf{q}$	$\tilde{\mathbf{q}}_{\text{LS}} = (1-\epsilon)\mathbf{q} +\epsilon\mathbf{u}$	$\tilde{\mathbf{q}}_{\text{CP}} = (1-\epsilon)\mathbf{q} -\epsilon\mathbf{p}$	$\tilde{\mathbf{q}}_{\text{LC}} = (1-\epsilon)\mathbf{q} +\epsilon\mathbf{p}$
Cross Entropy	$\mathrm{E}_{\mathbf{q}}(-\log \mathbf{p})$	$\mathrm{E}_{\tilde{\mathbf{q}}_{\text{LS}}}(-\log \mathbf{p})$	$\mathrm{E}_{\tilde{\mathbf{q}}_{\text{CP}}}(-\log \mathbf{p})$	$\mathrm{E}_{\tilde{\mathbf{q}}_{\text{LC}}}(-\log \mathbf{p})$
KL Divergence	$\mathrm{KL}(\mathbf{q}\|\mathbf{p})$	$(1-\epsilon)\mathrm{KL}(\mathbf{q}\|\mathbf{p})$ $+\epsilon\mathrm{KL}(\mathbf{u}\|\mathbf{p})$	$(1-\epsilon)\mathrm{KL}(\mathbf{q}\|\mathbf{p})$ $+\epsilon\mathrm{KL}(\mathbf{p}\|\mathbf{u})$	$(1-\epsilon)\mathrm{KL}(\mathbf{q}\|\mathbf{p})$ $-\epsilon\mathrm{KL}(\mathbf{p}\|\mathbf{u})$
Entropy minimisation	–	*Penalise* over CCE	*Penalise* over CCE	*Reward* over CCE
Semantic class	Annotated	Annotated	Annotated	Annotated and Learned
Similarity structure	No	No	No	Yes

3.3. Theory

Proposition 1. *LS, CP and LC modify the learning targets of standard CCE.*
Proof. $L_{\text{CCE+CP}}(\mathbf{q}, \mathbf{p}; \epsilon) = (1-\epsilon)\mathrm{H}(\mathbf{q}, \mathbf{p}) - \epsilon\mathrm{H}(\mathbf{p}, \mathbf{p}) = \mathrm{E}_{(1-\epsilon)\mathbf{q} - \epsilon\mathbf{p}}(-\log \mathbf{p})$. Therefore, $\tilde{\mathbf{q}}_{\text{CP}} = (1-\epsilon)\mathbf{q} - \epsilon\mathbf{p}$. Additionally, $\tilde{\mathbf{q}}_{\text{LS}} = (1-\epsilon)\mathbf{q} + \epsilon\mathbf{u}$, $\tilde{\mathbf{q}}_{\text{LC}} = (1-\epsilon)\mathbf{q} + \epsilon\mathbf{p}$. $\square$

Proposition 2. *Some KD methods, which aim to minimise the KL divergence between predictions of a teacher and a student, belong to the family of label correction.*
Proof. In general, a loss function of such methods can be defined to be $L_{\text{KD}}(\mathbf{q}, \mathbf{p}_t, \mathbf{p}) = (1-\epsilon)\mathrm{H}(\mathbf{q}, \mathbf{p}) + \epsilon\mathrm{KL}(\mathbf{p}_t\|\mathbf{p})$ [55]. $\mathrm{KL}(\cdot\|\cdot)$ denotes the KL divergence. As $\mathrm{KL}(\mathbf{p}_t\|\mathbf{p}) = \mathrm{H}(\mathbf{p}_t, \mathbf{p}) - \mathrm{H}(\mathbf{p}_t, \mathbf{p}_t)$, $\mathbf{p}_t$ is from a teacher and fixed when training a student. We can omit $\mathrm{H}(\mathbf{p}_t, \mathbf{p}_t)$:

$$L_{\text{KD}}(\mathbf{q}, \mathbf{p}_t, \mathbf{p}) = (1-\epsilon)\mathrm{H}(\mathbf{q}, \mathbf{p}) + \epsilon\mathrm{H}(\mathbf{p}_t, \mathbf{p})$$
$$= \mathrm{E}_{(1-\epsilon)\mathbf{q} + \epsilon\mathbf{p}_t}(-\log \mathbf{p}) \qquad (6)$$
$$\Rightarrow \tilde{\mathbf{q}}_{\text{KD}} = (1-\epsilon)\mathbf{q} + \epsilon\mathbf{p}_t.$$

Consistent with LC in Eq (5), $L_{\text{KD}}(\mathbf{q}, \mathbf{p}_t, \mathbf{p})$ revises a label using $\mathbf{p}_t$. $\square$

Proposition 3. *Compared with CCE, LS and CP penalise entropy minimisation while LC reward it.*

Proposition 4. *In CCE, LS and CP, a data point $\mathbf{x}$ has the same semantic class. In addition, $\mathbf{x}$ has an identical probability of belonging to other classes except for its semantic class.*

The proof of propositions 3 and 4 is presented in the Appendix A. Only LC exploits informative information and has the ability to correct labels, while LS and CP only relax the hard targets. We summarise CCE, LS, CP and LC in Table 1. Constant terms are ignored for concision.

4. ProSelfLC: Progressive and Adaptive Label Correction

In standard CCE, a semantic class is considered while the similarity structure is ignored. It is mainly due to the difficulty of annotating the similarity structure for every data point, especially when C is large [50]. Fortunately, recent progress demonstrates that there are some effective approaches to define the similarity structure of data points without annotation: (1) In KD, an auxiliary teacher model can provide a student model the similarity structure information [17, 29]; (2) In Self LC, e.g., Boot-soft, a model helps itself by exploiting the knowledge it has learned so far. We focus on studying the end-to-end Self LC.

In Self LC, ϵ indicates how much a predicted label distribution is trusted. In ProSelfLC, we propose to set it automatically according to learning time t and prediction entropy $\mathrm{H}(\mathbf{p})$, i.e., ProSelfLC trusts self knowledge according to training time and confidence. For any $\mathbf{x}$, we summarise:

$$\begin{cases} \text{Loss: } L_{(\tilde{\mathbf{q}}_{\text{ProSelfLC}}, \mathbf{p}; \epsilon_{\text{ProSelfLC}}) = \mathrm{H}(\tilde{\mathbf{q}}_{\text{ProSelfLC}}, \mathbf{p})} \\ \qquad\qquad\qquad\qquad = \mathrm{E}_{\tilde{\mathbf{q}}_{\text{ProSelfLC}}}(-\log \mathbf{p}). \\ \\ \text{Label: } \tilde{\mathbf{q}}_{\text{ProSelfLC}} = (1-\epsilon_{\text{ProSelfLC}})\mathbf{q} + \epsilon_{\text{ProSelfLC}}\mathbf{p}. \\ \\ \epsilon_{\text{ProSelfLC}} = g(t) \times l(\mathbf{p}) \begin{cases} g(t) = h(t/\Gamma - 0.5, B) \in (0,1), \\ l(\mathbf{p}) = 1 - \mathrm{H}(\mathbf{p})/\mathrm{H}(\mathbf{u}) \in (0,1). \end{cases} \end{cases}$$
$$(7)$$

t and Γ are the iteration counter and the number of total iterations, respectively. $h(\eta, B) = 1/(1 + \exp(-\eta \times B))$. Here, $\eta = t/\Gamma - 0.5$. B, Γ are task-dependent and searched on a validation set.

4.1. Self trust scores

Global trust score $g(t)$ denotes how much we trust a learner. It is independent of data points, thus being global. $g(t)$ grows as t rises. B adjusts the exponentiation's base and growth speed of $g(t)$.

Local trust score $l(\mathbf{p})$ indicates how much we trust an output distribution $\mathbf{p}$, which is data-dependent. $l(\mathbf{p})$ rises as $\mathrm{H}(\mathbf{p})$ becomes lower, rewarding a confident distribution.

4.2. Design reasons

Regarding $g(t)$, in the earlier learning phase, i.e., $t < \Gamma/2$, $g(t) < 0.5 \Rightarrow \epsilon_{\text{ProSelfLC}} < 0.5, \forall\mathbf{p}$, so that the human annotations dominate and ProSelfLC only modifies the similarity structure. When a learner has not seen the training data for enough time at the earlier stage, its knowledge is less reliable and a wrong confident prediction may occur. Our design assuages the bad impact of such unexpected cases. When it comes to the later training phase, i.e., $t > \Gamma/2$, we have $g(t) > 0.5$ as it has been trained for more than half of entire iterations.

Table 2: The values of $g(t)$, $l(\mathbf{p})$ and $\epsilon_{\text{ProSelfLC}} = g(t) \times l(\mathbf{p})$ under different cases. We use concrete values for concise interpretation. We bold the special case when the semantic class is changed.

		$l(\mathbf{p})$: Consistency is defined by whether $\mathbf{p}$ and $\mathbf{q}$ share the semantic class or not.		
		0.1(non-confident)	0.9(confidently consistent)	0.9(confidently inconsistent)
Earlier phase	$g(t) = 0.1$	0.01	0.09	0.09
Later phase	$g(t) = 0.9$	0.09	0.81	**0.81**

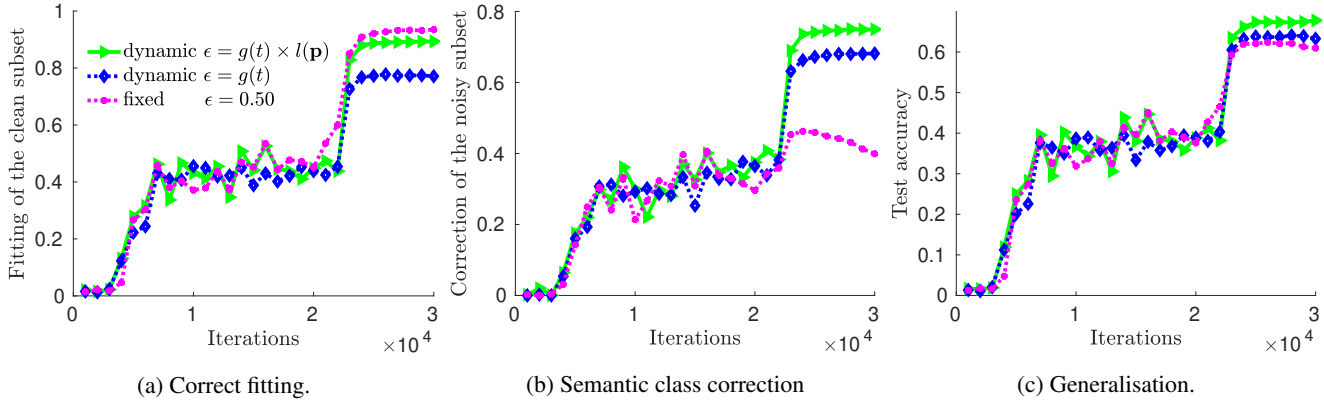(a) Correct fitting.   (b) Semantic class correction   (c) Generalisation.

Figure 2: Study of setting ϵ using three schemes: global trust and local trust, merely global trust, and fixed ϵ. Experiments are done on CIFAR-100 with 40% asymmetric label noise. Vertical axes denote evaluation metrics. Mean results are displayed.

Regarding $l(\mathbf{p})$, it affects the later learning phase. If $\mathbf{p}$ is less confident, $l(\mathbf{p})$ will be smaller, then $\epsilon_{\text{ProSelfLC}}$ will be smaller, hence we trust $\mathbf{p}$ less when it is of higher uncertainty. If $\mathbf{p}$ is highly confident, we trust its confident knowledge. *Ablation study of our design is in Figure 2, where three variants of ϵ are presented.* In our experiments, note that when ϵ is fixed, we try three values (0.125, 0.25, 0.50) and display the best instantiation, i.e., $\epsilon = 0.50$.

4.3. Case analysis

Due to the potential memorisation in the earlier phase (which is more rare), we may get unexpected confident wrong predictions for noisy labels, but their trust scores are small as $g(t)$ is small. We conduct the case analysis of ProSelfLC in Table 2 and summarise its core tactics as follows:

(1) Correct the similarity structure for every data point in all cases, thanks to exploiting the self knowledge of a learner, i.e., $\mathbf{p}$.

(2) Revise the semantic class when t is large enough and $\mathbf{p}$ is confidently inconsistent. As highlighted in Table 2, when two conditions are met, we have $\epsilon_{\text{ProSelfLC}} > 0.5$ and $\arg\max_j \mathbf{p}(j|\mathbf{x}) \neq \arg\max_j \mathbf{q}(j|\mathbf{x})$, then $\mathbf{p}$ redefines the semantic class. For example, if $\mathbf{p} = [0.95, 0.01, 0.04]$, $\mathbf{q} = [0, 0, 1]$, $\epsilon_{\text{ProSelfLC}} = 0.8 \Rightarrow \tilde{\mathbf{q}}_{\text{ProSelfLC}} = (1 - \epsilon_{\text{ProSelfLC}})\mathbf{q} + \epsilon_{\text{ProSelfLC}}\mathbf{p} = [0.76, 0.008, 0.232]$. Note that ProSelfLC also becomes robust against lengthy exposure to the training data, as demonstrated in Figures 2, 3.

5. Experiments

In deep learning, small differences (e.g., random accelerators like cudnn and different frameworks like Caffe [18],

Tensorflow [1] and PyTorch [31]) may lead to a large gap of final performance. Therefore, to compare more properly, we re-implement CCE, LS and CP. Regarding Self LC methods, we re-implement Boot-soft [35], where ϵ is fixed throughout training. We do not re-implement stage-wise Self LC and KD methods, e.g., Joint Optimisation and Tf-KD$_{self}$ respectively, because time-consuming tuning is required. We fix the random seed and do not use any random accelerator. In standard and synthetic cases, we train on 80% training data (corrupted in synthetic cases) and use 20% trusted training data as a validation set to search all hyperparameters, e.g., ϵ, Γ, B and settings of an optimiser. *Note that Γ and an optimiser's settings are searched first and then fixed for all methods.* Finally, we retrain a model on the entire training data (corrupted in synthetic cases) and report its accuracy on the test data to fairly compare with prior results. In real-world label noise, the used dataset has a separate clean validation set. Here, a clean dataset is used only for validation, which is generally necessary for any method and differs from the methods [44, 45, 19, 36, 26, 16, 53, 60] which use a clean dataset to train a network's learning parameters.

5.1. Standard image classification

Datasets and training details. (1) CIFAR-100 [20] has 20 coarse classes, each containing 5 fine classes. There are 500 and 100 images per class in the training and testing sets, respectively. The image size is 32×32. We apply simple data augmentation [15], i.e., we pad 4 pixels on every side of the image, and then randomly crop it with a size of 32×32. Finally, this crop is horizontally flipped with a

Table 3: Test accuracy (%) in the standard setting. We report three settings of hyperparameters.

Dataset	CCE	LS (ϵ)			CP (ϵ)			Boot-soft (ϵ)			ProSelfLC (B)		
		0.125	0.25	0.50	0.125	0.25	0.50	0.125	0.25	0.50	8	10	12
CIFAR-100	69.0	69.9	69.6	68.4	69.5	69.3	68.7	68.9	69.1	69.1	70.1	**70.3**	69.8
ImageNet 2012	75.5	75.3	75.2	74.9	75.2	74.8	74.6	75.7	75.8	75.8	**76.0**	**76.0**	75.9

(a) Correct fitting.

(b) Wrong fitting.

(c) Semantic class correction

(d) Entropy of clean subset.
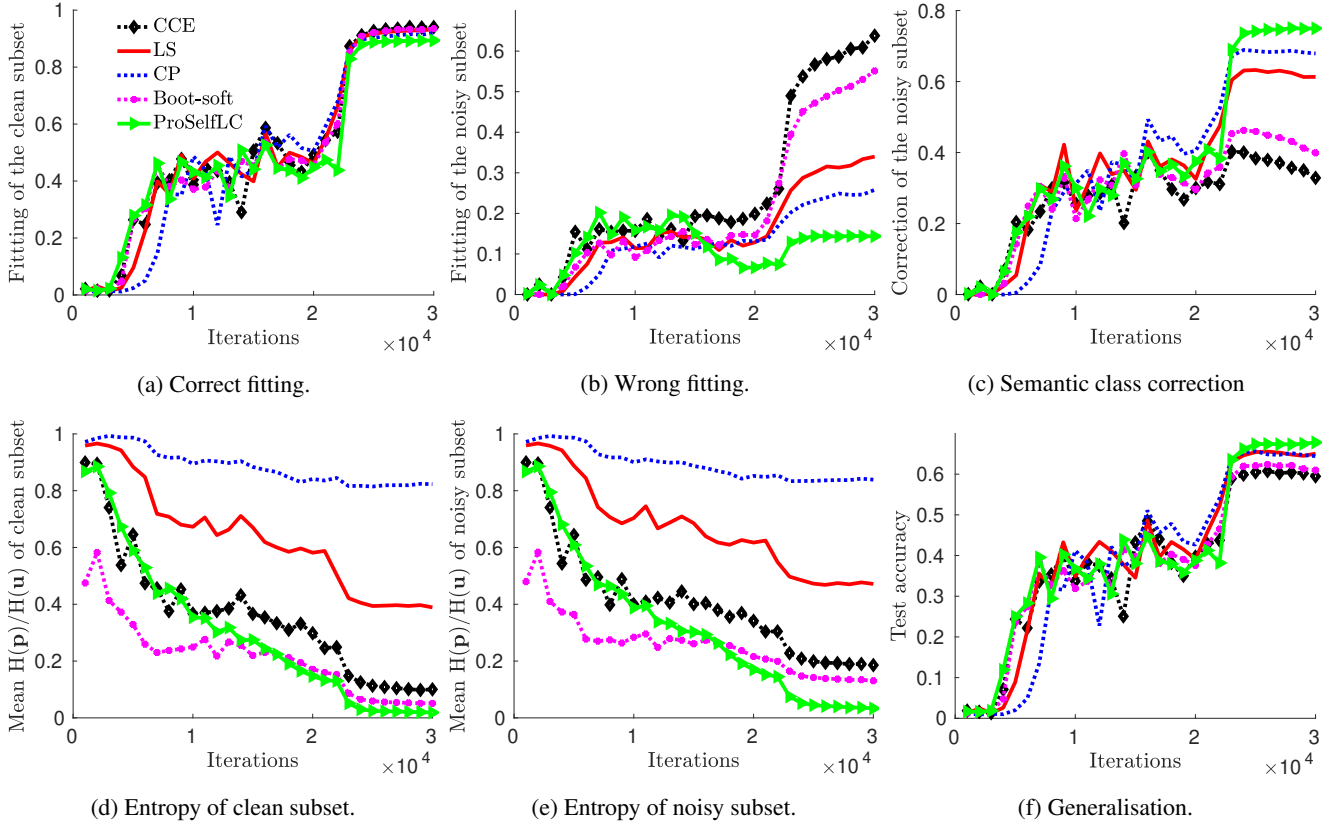
(e) Entropy of noisy subset.

(f) Generalisation.

Figure 3: Comprehensive learning dynamics on CIFAR-100 with 40% asymmetric label noise. Vertical axes denote evaluation metrics. Their mean results are displayed. *At training, a learner is NOT GIVEN whether a label is trusted or not.* We store intermediate models and analyse them when the training ends.

probability of 0.5. We choose SGD with its settings as: (a) a learning rate of 0.1; (b) a momentum of 0.9; (c) a weight decay of $5e-4$; (d) the batch size is 256 and the number of training iterations is 30k. We divide the learning rate by 10 at 15k and 22k iterations, respectively. (2) We train ResNet-50 [15] on ImageNet 2012 classification dataset, which has 1k classes and 50k images in the test set [39]. We use SGD with a start learning rate of $2e-3$. A polynomial learning rate decay with a power of 2 is used. We set the momentum to 0.95 and the weight decay to $1e-4$. We train on a single V100 GPU and the batch size is 64. We report the final test accuracy when the training ends at 500k iterations. We use the standard data augmentation: an original image is warped to 256×256, followed by a random crop of 224×224. This crop is randomly flipped. We fix common settings to fairly compare CCE, LS, CP, Boot-soft and ProSelfLC.

Result analysis. In Table 3, we observe the superiority

of ProSelfLC in standard setting without considering label noise. Being probably surprising, LS and CP reduce the performance consistently as ϵ increases on ImageNet. Instead, Boot-soft and ProSelfLC improve versus CCE. We remark that both test sets are large so that their differences are noticeable.

5.2. Synthetic label noise

Noise generation. (1) Symmetric label noise: the original label of an image is uniformly changed to one of the other classes with a probability of r; (2) Asymmetric label noise: we follow [46] to generate asymmetric label noise to fairly compare with their reported results. Within each coarse class, we randomly select two fine classes A and B. Then we flip $r \times 100\%$ labels of A to B, and $r \times 100\%$ labels of B to A. We remark that the overall label noise rate is smaller than r.

Table 4: Accuracy (%) on the CIFAR-100 clean test set. All compared methods use ResNet-44.

	Method	Asymmetric Noisy Labels			Symmetric Noisy Labels		
		r=0.2	r=0.3	r=0.4	r=0.2	r=0.4	r=0.6
Results From SL [46]	Boot-hard	63.4	63.2	62.1	57.9	48.2	12.3
	Forward	64.1	64.0	60.9	59.8	53.1	24.7
	D2L	62.4	63.2	61.4	59.2	52.0	35.3
	GCE	63.0	63.2	61.7	59.1	53.3	36.2
	SL	65.6	65.1	63.1	60.0	53.7	41.5
Our Trained Results	CCE	66.6	63.4	59.5	58.0	50.1	37.9
	LS	67.9	66.4	65.0	63.8	57.2	46.5
	CP	67.7	66.0	64.4	64.0	56.8	44.1
	Boot-soft	66.9	65.3	61.0	63.2	59.0	44.8
	ProSelfLC	**68.7**	**68.5**	**67.9**	**64.8**	**59.3**	**47.7**

Table 5: The results of different hyperparameters on CIFAR-100 using ResNet-44. Under different noise rates, the best instantiation of each approach is bolded except for CCE.

Method (hyperparameter)	Value of hyperparameter	Asymmetric label noise			Symmetric label noise			Clean
		20%	30%	40%	20%	40%	60%	
CCE	None or $\epsilon = 0$	66.6	63.4	59.5	58.0	50.1	37.9	69.0
LS (ϵ)	0.125	66.4	65.6	63.1	61.7	52.5	39.1	**69.9**
	0.25	**67.9**	**66.4**	**65.0**	62.8	55.9	40.9	69.6
	0.50	66.8	65.8	64.6	**63.8**	**57.2**	**46.5**	68.4
CP (ϵ)	0.125	65.7	64.2	60.3	59.8	52.3	39.6	**69.5**
	0.25	66.8	65.1	61.6	61.0	53.3	40.9	69.3
	0.50	**67.7**	**66.0**	**64.4**	**64.0**	**56.8**	**44.1**	68.7
Boot-soft (ϵ)	0.125	65.8	64.1	60.7	59.7	51.2	40.6	68.9
	0.25	66.2	64.1	60.3	61.1	54.4	43.3	**69.1**
	0.50	**66.9**	**65.3**	**61.0**	**63.2**	**59.0**	**44.8**	**69.1**
ProSelfLC (B)	8	67.8	67.4	**67.9**	64.7	57.7	**47.7**	70.1
	10	68.5	**68.5**	66.8	63.9	59.0	47.5	**70.3**
	12	68.6	67.9	67.4	64.0	**59.3**	47.5	69.8
	14	**68.7**	68.0	67.8	**64.8**	59.0	47.4	69.6
	16	68.4	67.2	67.3	63.7	59.0	32.3	69.9

Baselines.[1] We compare with the results reported recently in SL [46]. Forward is a loss correction approach that uses a noise-transition matrix [32]. D2L monitors the subspace dimensionality change at training [27]. GCE denotes generalised cross entropy [59] and SL is symmetric cross entropy [46]. They are robust losses designed for solving label noise. Training details are the same as Section 5.1.

Result analysis. For all methods, we directly report their final results when training terminates. Therefore, *we test the robustness of a model against not only label noise, but also a long time being exposed to the data.* In Table 4, we observe that: (1) ProSelfLC outperforms all baselines, which is significant in most cases; (2) In both implementation, Boot-hard and Boot-soft perform worse than the others. However, our ProSelfLC makes Self LC the best solution. Furthermore, learning dynamics are visualised in Figure 3, which helps to understand why ProSelfLC works bet-

ter. Figure 4 shows the learning dynamics when r changes. Results of different B, ϵ are in Table 5. We note that when the noise rate is higher, a smaller B performs better.

We further discuss their differences on model calibration [11] in Appendix B, the changes of entropy and $\epsilon_{\text{ProSelfLC}}$ during training in Appendix C.

Revising the semantic class and similarity structure. In Figures 3b and 3c, we show dynamic statistics of different approaches on fitting wrong labels and correcting them. ProSelfLC is much better than its counterparts. Semantic class correction reflects the change of similarity structure.

To redefine and reward a low-entropy status. On the one hand, we observe that LS and CP work well, being consistent with prior claims. In Figures 3d and 3e, the entropies of both clean and noisy subsets are much higher in LS and CP, correspondingly their generalisation is the best except for ProSelfLC in Figure 3f. On the other hand, ProSelfLC has the lowest entropy while performs the best, which proves that a learner's confidence does not necessarily weaken its generalisation performance. Instead, a model needs to be careful with what to be confident in. As shown

---

[1]We do not consider DisturbLabel [49], which flips labels randomly and is counter-intuitive. It weakens the generalisation because generally the accuracy drops as the uniform label noise increases.

Table 6: Test accuracy (%) on the real-world noisy dataset Clothing 1M.

Boot-hard	Forward	D2L	GCE	SL	S-adaptation	Masking	MD-DYR-SH	Joint-soft	Our Trained Results				
									CCE	LS	CP	Boot-soft	ProSelfLC
68.9	69.8	69.5	69.8	71.0	70.3	71.1	71.0	72.2	71.8	72.6	72.4	72.3	**73.4**

(a) $r = 20\%$.　　　　(b) $r = 30\%$.　　　　(c) $r = 40\%$.
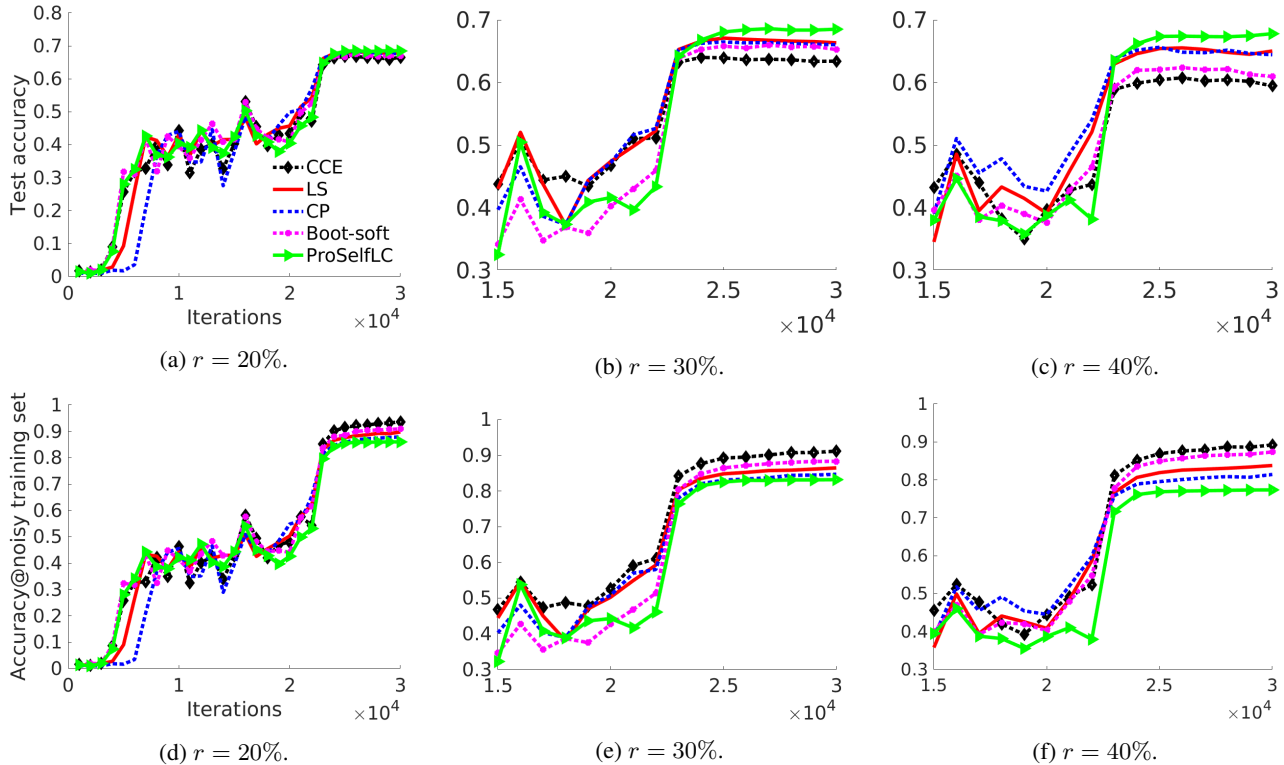
(d) $r = 20\%$.　　　　(e) $r = 30\%$.　　　　(f) $r = 40\%$.

Figure 4: Learning dynamics on CIFAR-100 under asymmetric noisy labels. We show all iterations only in (a) and (d). In the others, we show the second half iterations, which are of higher interest. As the noise rate increases, the superiority of ProSelfLC becomes more significant, i.e., avoiding fitting noise in the 2nd row and better generalisation in the 1st row.

by Figures 3b and 3c, ProSelfLC has the least wrong fitting and most semantic class correction, which indicates that a meaningful low-entropy status is redefined.

5.3. Real-world label noise

Clothing 1M [48] has around 38.46% label noise in the training data and about 1 million images of 14 classes from shopping websites. Its internal noise structure is agnostic.

Baselines. For loss correction and estimating the noise-transition matrix, S-adaption [8] uses an extra softmax layer, while Masking [12] exploits human cognition. MD-DYR-SH [2] is a combination of three techniques: dynamic mixup (MD), dynamic bootstrapping together with label regularisation (DYR) and soft to hard (SH). The other baselines have been introduced heretofore.

Training details. We follow [43] to train ResNet-50 and initialise it by a trained model on ImageNet. We follow Section 5.1 with small changes: the initial learning rate is 0.01 and we train 10k iterations. They are searched on the separate clean validation set.

Result analysis. In Table 6, analogously to CIFAR-100, we report our trained results of CCE, LS, CP, Boot-soft and ProSelfLC for an entirely fair comparison. ProSelfLC has the highest accuracy, which demonstrates its effectiveness again.

6. Conclusion

We present a thorough mathematical study on several target modification techniques. Through analysis of entropy and KL divergence, we reveal their relationships and limitations. To improve and endorse self label correction, we propose ProSelfLC. Extensive experiments prove its superiority over existing methods under standard and noisy settings. ProSelfLC enhances the similarity structure information over classes, and rectifies the semantic classes of noisy label distributions. ProSelfLC is the first approach to trust self knowledge progressively and adaptively.

ProSelfLC redirects and promotes entropy minimisation, which is in marked contrast to recent practices of confidence penalty [42, 33, 6].

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation*.

[2] Eric Arazo, Diego Ortego, Paul Albert, Noel O'Connor, and Kevin Mcguinness. Unsupervised label noise modeling and loss correction. In *ICML*, 2019.

[3] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. In *ICML*, 2017.

[4] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *NeurIPS*, 2014.

[5] Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *KDDM*, 2006.

[6] Abhimanyu Dubey, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. Maximum-entropy fine grained classification. In *NeurIPS*, 2018.

[7] Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *ICML*, 2018.

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

[9] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *NeurIPS*, 2005.

[10] Yves Grandvalet and Yoshua Bengio. Entropy regularization. *Semi-supervised learning*, pages 151–168, 2006.

[11] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *ICML*, 2017.

[12] Bo Han, Jiangchao Yao, Gang Niu, Mingyuan Zhou, Ivor Tsang, Ya Zhang, and Masashi Sugiyama. Masking: A new perspective of noisy supervision. In *NeurIPS*, 2018.

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

[14] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

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

[16] Dan Hendrycks, Mantas Mazeika, Duncan Wilson, and Kevin Gimpel. Using trusted data to train deep networks on labels corrupted by severe noise. In *NeurIPS*, 2018.

[17] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NeurIPS Deep Learning and Representation Learning Workshop*, 2015.

[18] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACMMM*, 2014.

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

[20] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[21] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, pages 79–86, 1951.

[22] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, page 436, 2015.

[23] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. 2013.

[24] Kuang-Huei Lee, Xiaodong He, Lei Zhang, and Linjun Yang. Cleannet: Transfer learning for scalable image classifier training with label noise. In *CVPR*, 2018.

[25] Junnan Li, Richard Socher, and Steven CH Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. In *ICLR*, 2020.

[26] Yuncheng Li, Jianchao Yang, Yale Song, Liangliang Cao, Jiebo Luo, and Li-Jia Li. Learning from noisy labels with distillation. In *ICCV*, 2017.

[27] Xingjun Ma, Yisen Wang, Michael E Houle, Shuo Zhou, Sarah M Erfani, Shu-Tao Xia, Sudanthi Wijewickrema, and James Bailey. Dimensionality-driven learning with noisy labels. In *ICML*, 2018.

[28] Eran Malach and Shai Shalev-Shwartz. Decoupling "when to update" from "how to update". In *NeurIPS*, 2017.

[29] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? In *NeurIPS*, 2019.

[30] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *AAAI*, 2015.

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[32] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *CVPR*, 2017.

[33] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. In *ICLR Workshop*, 2017.

[34] Siyuan Qiao, Wei Shen, Zhishuai Zhang, Bo Wang, and Alan Yuille. Deep co-training for semi-supervised image recognition. In *ECCV*, 2018.

[35] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. In *ICLR Workshop*, 2015.

[36] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *ICML*, 2018.

[37] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *ICLR*, 2015.

[38] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, pages 533–536, 1986.

[39] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, pages 211–252, 2015.

[40] Hwanjun Song, Minseok Kim, and Jae-Gil Lee. Selfie: Refurbishing unclean samples for robust deep learning. In *ICML*, 2019.

[41] Sainbayar Sukhbaatar and Rob Fergus. Learning from noisy labels with deep neural networks. *arXiv preprint arXiv:1406.2080*, 2014.

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

[43] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning with noisy labels. In *CVPR*, 2018.

[44] Arash Vahdat. Toward robustness against label noise in training deep discriminative neural networks. In *NeurIPS*, 2017.

[45] Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge Belongie. Learning from noisy large-scale datasets with minimal supervision. In *CVPR*, 2017.

[46] Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey. Symmetric cross entropy for robust learning with noisy labels. In *ICCV*, 2019.

[47] Hongxin Wei, Lei Feng, Xiangyu Chen, and Bo An. Combating noisy labels by agreement: A joint training method with co-regularization. In *CVPR*, 2020.

[48] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *CVPR*, 2015.

[49] Lingxi Xie, Jingdong Wang, Zhen Wei, Meng Wang, and Qi Tian. Disturblabel: Regularizing cnn on the loss layer. In *CVPR*, 2016.

[50] Ning Xu, Jun Shu, Yun-Peng Liu, and Xin Geng. Variational label enhancement. In *ICML*, 2020.

[51] Ting-Bing Xu and Cheng-Lin Liu. Data-distortion guided self-distillation for deep neural networks. In *AAAI*, 2019.

[52] Jiangchao Yao, Hao Wu, Ya Zhang, Ivor W Tsang, and Jun Sun. Safeguarded dynamic label regression for noisy supervision. In *AAAI*, 2019.

[53] Serena Yeung, Vignesh Ramanathan, Olga Russakovsky, Liyue Shen, Greg Mori, and Li Fei-Fei. Learning to learn from noisy web videos. In *CVPR*, 2017.

[54] Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor W Tsang, and Masashi Sugiyama. How does disagreement help generalization against label corruption? In *ICML*, 2019.

[55] Li Yuan, Francis EH Tay, Guilin Li, Tao Wang, and Jiashi Feng. Revisiting knowledge distillation via label smoothing regularization. In *CVPR*, 2020.

[56] Sukmin Yun, Jongjin Park, Kimin Lee, and Jinwoo Shin. Regularizing class-wise predictions via self-knowledge distillation. In *CVPR*, 2020.

[57] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *ICCV*, 2019.

[58] Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep mutual learning. In *CVPR*, 2018.

[59] Zhilu Zhang and Mert R Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *NeurIPS*, 2018.

[60] Zizhao Zhang, Han Zhang, Sercan O Arik, Honglak Lee, and Tomas Pfister. Distilling effective supervision from severe label noise. In *CVPR*, 2020.