Deep Critical Learning (i.e., Deep Robustness):

Example weighting and label correction.

Guest Speaker: (Amos) Xinshao Wang, PhD in CS/DL.
Senior ML researcher at Zenith Ai, UK.
Former Visit Scholar &
Former Postdoc at the University of Oxford.

12th Aug 2022, Southern University of Science and Technology

Outline

Outline

Why do we need deep critical learning?

To learn meaningful patterns under real-world adverse cases.

Figure: Display of abnormal training examples highlighted by red boxes.

Outline

- Some inputs contain only background and no semantic information at all.
- Some may contain a object that does not belong to any class in the training set.

Adverse cases in real-world data

In-distribution anomalies: Detect => Ignore or Correct

- Label noise arising from:
  1. Noisy annotations: e.g., some images of deer may be wrongly annotated to horse.
  2. Missing annotations: we may use some algorithms to predict their labels. If so, the predicted labels are not 100% accurate.
- When an input contains more than one object, it becomes ambiguous without any prior.

Outline

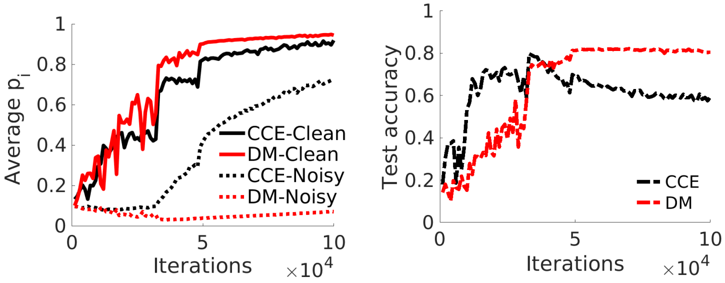
Objectives of critical/robust deep learning

What is the meaning of critical learning here?

- **To learn meaningful patterns** on semantically clean training data (where noise may exist, however, the semantic matching from observations to annotations is highly correct).
- **Without fitting wrong patterns** on semantically wrong training data, so that the learning process of a model is not contaminated.
- **Generalisation** to **unseen** data.

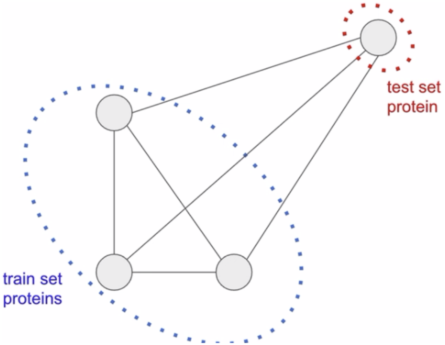
Robustness against adverse cases

What should a learning process ideally look like?



- $p_i = p(y_i|x_i)$: predicted relevance between an observation x_i and it label y_i.
- We train ResNet-56 on CIFAR-10 with 40% symmetric label noise.

Generalise to unseen data



[2] Bileschi, Maxwell L., et al. "Using deep learning to annotate the protein universe." Nature Biotechnology (2022).

Outline

Robust learning via example weighting

Example weighting is universal in deep learning

We define our interpretation of example weighting [11]:

Definition (Example Weighting). *In gradient-based optimisation, the loss's derivative of an example can be interpreted as its effect on the update of a model [4, 1]. Therefore, a derivative's magnitude function can be treated as a weighting scheme from the viewpoint of example weighting.*

Accordingly, one technique that leads to a change of the derivative magnitude function, is equivalent to, modifying an example weighting scheme.

Outline

Derivative manipulation (DM)

Fundamental research questions and conclusions

1. What kind of examples to focus on during training? How to weight them properly?
2. A loss function is okay to be non-symmetric, unbounded, or even non-differentiable.
3. Is a loss function necessary for deriving the gradient used for back-propagation?
4. When a training set contains a higher label noise rate, we should focus on easier training examples for better generalisation!

Standard practice versus DM

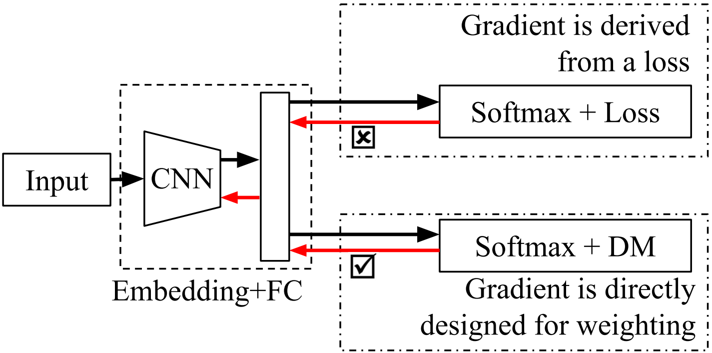


Figure: Black and red arrows denote forward process and gradient back-propagation, respectively.

Definition 1 (Emphasis Mode ψ). We define the emphasis mode to be p_i of examples whose weights are the largest, i.e., $\psi = \arg\max_{p_i} w_i, \quad \psi \in [0, 1]$.

For example, by 'emphasis mode is 0 in CCE' we mean those images with $p_i = 0$ own the highest weights.

Definition 2 (Emphasis Variance σ). $\sigma = \mathrm{E}((w_i - \mathrm{E}(w_i))^2)$, where $\mathrm{E}(\cdot)$ denotes the expectation of a variable.

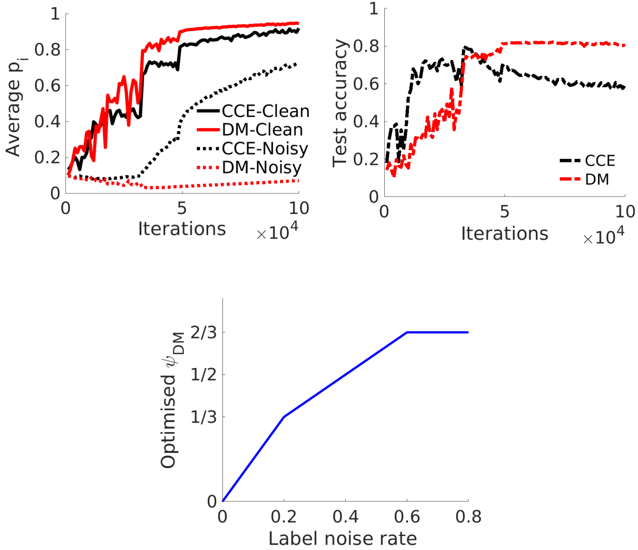


Figure: We observe noisy examples have much less p_i than clean ones, thus being more difficult examples.

Outline

Core research questions [12]

1. In Self LC, how much should we trust a learner to leverage its knowledge?
   - The trust score is fixed or updated stage-by-stage in prior work.
   - ProSelfLC modifies the target progressively, is end-to-end trainable, and requires negligible extra cost.
2. Should we penalise a low-entropy status or reward it?
   - OR methods penalise low entropy while LC rewards it.
   - ProSelfLC [12] redirects and promotes entropy minimisation.
3. Towards a low-temperature entropy state:
   - Using the standard training setting, a trained network is of low confidence when severe noise exists, making it hard to leverage its high-entropy self knowledge.
   - Therefore, we decrease the entropy of self knowledge using a low temperature before exploiting it to correct labels.

Outline

BEYOND SEMANTIC CLASS

THE SIMILARITY STRUCTURE IN A LABEL DISTRIBUTION

A label distribution defines what to learn:

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

- **Definition 2** (*Similarity Structure*). In CCE, LS and CP, a data point has an identical probability of belonging to other classes except for the semantic class. Instead, in LC, a target label distribution captures the probability difference of an example being predicted to every class. We define it to be the similarity structure of one example versus all training classes.

An overview of label (target) modification

OR(LS and CP) + LC(Self LC and Non-self LC)

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

$$\underset{1-\epsilon}{\overset{\mathbf{q}}{\begin{pmatrix}1\\0\\0\end{pmatrix}}} + \underset{\epsilon}{\overset{\mathbf{u}}{\begin{pmatrix}1/3\\1/3\\1/3\end{pmatrix}}} \Longrightarrow \begin{pmatrix}(1-\epsilon)+\epsilon/3\\ \epsilon/3\\ \epsilon/3\end{pmatrix}$$

LS

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

$$\underset{1-\epsilon}{\overset{\mathbf{q}}{\begin{pmatrix}1\\0\\0\end{pmatrix}}} - \underset{\epsilon}{\overset{\mathbf{p}}{\begin{pmatrix}1/2\\1/3\\1/6\end{pmatrix}}} \Rightarrow \begin{pmatrix}(1-\epsilon)-\epsilon/2\\ -\epsilon/3\\ -\epsilon/6\end{pmatrix} \Leftrightarrow \begin{pmatrix}(1-\epsilon)-\epsilon/2\\ 0\\ 0\end{pmatrix}$$

CP

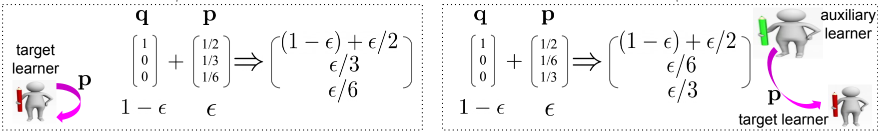
OR includes LS [9] and CP [7], which smoothes similarity structure:

- 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.

An overview of label (target) modification

OR(LS and CP) + LC(Self LC and Non-self LC)

$$\bar{\mathbf{q}}_{\mathrm{LC}} = (1-\epsilon)\mathbf{q} + \epsilon\mathbf{p}$$



Self LC: **p** is the output of a learner itself.

Non-self LC: **p** is the output of an auxiliary learner.

- LC contains Self LC [6, 8, 10] and Non-self LC [5].
- The convex combination parameter ϵ defines how much a predicted label distribution is trusted.

Why do we choose to improve Self LC?

1. OR methods naively penalise confident outputs without leveraging easily accessible knowledge from other learners or itself.
2. Non-self LC relies on accurate auxiliary models.
3. Self LC:
    - It exploits its own knowledge;
    - It requires no extra learners;
    - However, **how much should we trust a learner to leverage its knowledge?**

Overview of existing variants of Self LC

Without considering a model's knowledge grows as time goes

1. In bootstrapping, ϵ is fixed throughout the training process.
2. Joint Optimisation fully trusts a learner by setting $\epsilon = 1$, and uses stage-wise training to gradually train the model.
   - Stage-wise training requires a significant human intervention and is time-consuming in practice.
3. **Requirements of improving Self LC**
   - End-to-end trainable.
   - Negligible extra cost.
   - Modifies the target progressively and adaptively as training goes.

ProSelfLC

Self Trust according to Training Time and Confidence

ϵ indicates how much a predicted label distribution is trusted.
For any x, we summarise the loss and modified label:

$$
\begin{aligned}
&L_{(}\tilde{q}_{\mathrm{ProSelfLC}}, p; \epsilon_{\mathrm{ProSelfLC}}) = \mathrm{H}(\tilde{q}_{\mathrm{ProSelfLC}}, p) = \mathrm{E}_{\tilde{q}_{\mathrm{ProSelfLC}}}(-\log\ p), \\
&\tilde{q}_{\mathrm{ProSelfLC}} = (1 - \epsilon_{\mathrm{ProSelfLC}})q + \epsilon_{\mathrm{ProSelfLC}} p, \\
&\epsilon_{\mathrm{ProSelfLC}} = g(t) \times l(p), \\
&g(t) = h(t/\Gamma - 0.5, B) \in (0, 1), \Rightarrow \text{Trusting learning time} \\
&l(p) = 1 - \mathrm{H}(p)/\mathrm{H}(u) \in (0, 1). \Rightarrow \text{Trusting sample confidence}
\end{aligned}
\tag{1}
$$

t and Γ are the iteration (time) counter and the number of total iterations, respectively.
$h(\cdot)$ is a logistic function where B controls its smoothness.

Outline

To penalise or reward a low-entropy status?

- OR methods penalise low entropy $\Rightarrow$ OR is against entropy minimisation principle.
- LC rewards a low-entropy status $\Rightarrow$ LC defends entropy minimisation principle.
  - LC has the same principle as the widely used expectation–maximization (EM) algorithm.

Outline

CCE+ResNet18 on CIFAR-100

The final model is used for analysis when the training stops.

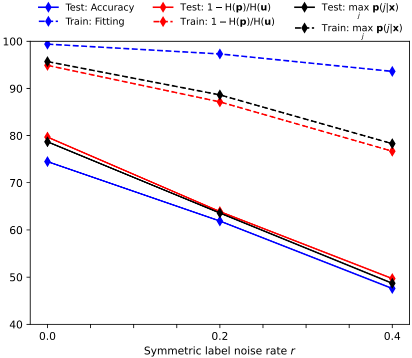


Figure: Our finding: as r increases, though the fitting of train set decreases little [13], *the confidences of both sets drop significantly*.

- We apply a low temperature T to decrease the entropy of self knowledge. Other than using p to correct labels, we use

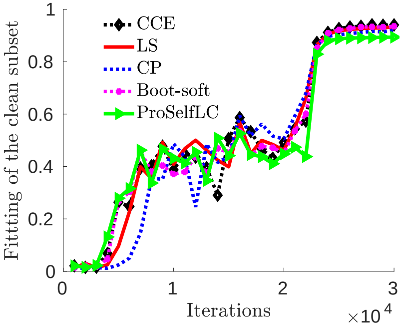
$$\mathrm{p}_T(j|\mathrm{x}) = \exp(\mathrm{z}_j/T)/\sum_{m=1}^{C}\exp(\mathrm{z}_m/T). \tag{2}$$

- An annealed temperature (denoted by AT, $0 < T < 1$) works best together with ProSelfLC.

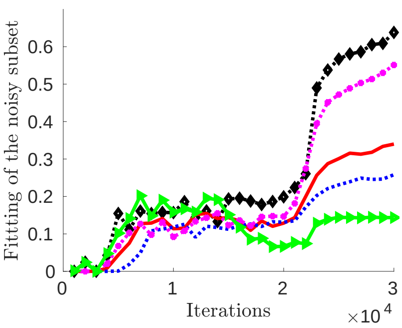
Outline

ProSelfLC

Training Dynamics On CIFAR-100 with asymmetric label noise $r = 0.4$.



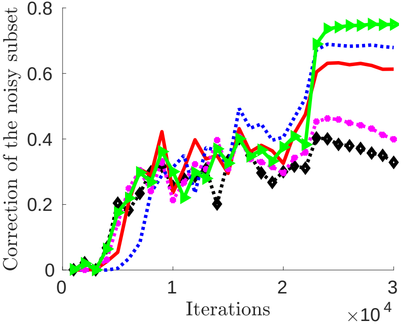
(a) Correct fitting.



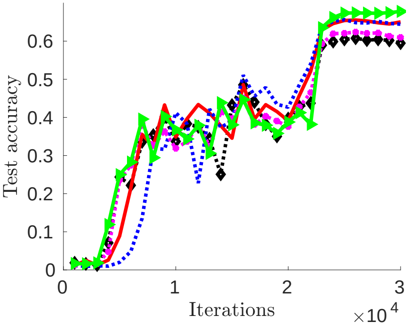
(b) Wrong fitting.

ProSelfLC

Training Dynamics On CIFAR-100 with asymmetric label noise $r = 0.4$.



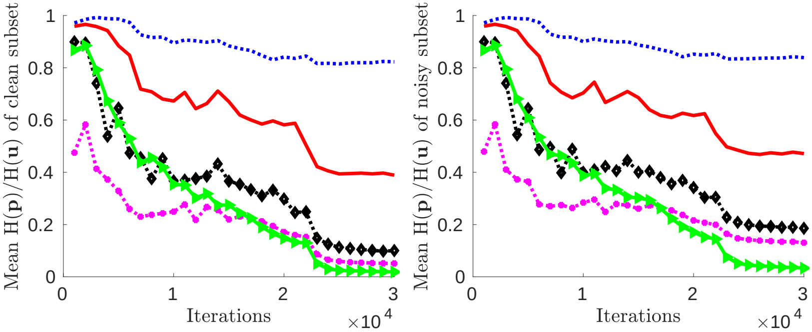
(a) Semantic class correction



(b) Generalisation.

ProSelfLC

Training Dynamics On CIFAR-100 with asymmetric label noise $r = 0.4$.



(a) Entropy of clean subset.

(b) Entropy of noisy subset.

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

Outline

Protein classification

Table: Results of robust protein classification against cropping noise and symmetric label noise. For comprehensiveness, we report accuracy and confidence metrics on both train and test sets. We show two confidence metrics: $1 - H(\mathrm{p})/H(\mathrm{u})$ and $\max_j \mathrm{p}(j|\mathrm{x})$. We use the final model when training ends without selecting the intermediate best ones. The highest accuracy or confidence in each row is green-colored.

Noise type	Metric	Data	CCE	LS	CP	Boot-soft	ProSelfLC
Cropping noise	Accuracy	Test	88.7	90.4	90.9	90.4	92.4
		Train	99.3	99.3	98.8	95.9	91.8
	$1 - \frac{H(\mathbf{p})}{H(\mathbf{u})}$	Test	86.9	59.5	83.8	89.0	96.4
		Train	89.0	63.3	82.0	92.6	97.6
	$\max_j \mathbf{p}(j\|\mathbf{x})$	Test	96.4	90.1	95.5	96.8	99.2
		Train	97.6	92.2	95.7	98.5	99.7
Cropping noise + Label noise	Accuracy	Test	84.2	89.4	89.4	89.6	92.2
		Train	84.8	75.2	72.8	77.2	75.7
	$1 - \frac{H(\mathbf{p})}{H(\mathbf{u})}$	Test	65.3	38.7	37.4	57.6	96.1
		Train	42.0	20.0	18.0	31.3	95.2
	$\max_j \mathbf{p}(j\|\mathbf{x})$	Test	88.9	80.4	78.4	86.0	99.2
		Train	80.7	70.0	67.5	74.9	99.1

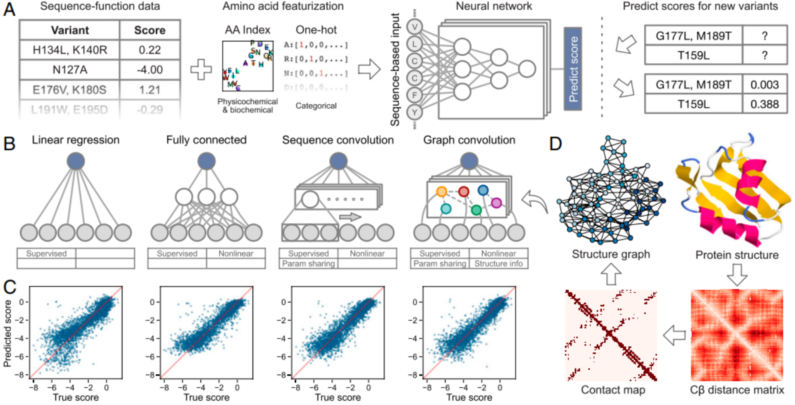
Outline

Summary

1. ProSelfLC:
    - enhances the similarity structure information over training classes.
    - corrects the semantic classes of noisy label distributions.
    - is the first method to trust self knowledge progressively and adaptively.
    - learns towards a low-temperature entropy state.
2. Our extensive experiments:
    - defend the entropy minimisation principle.
3. Code: https://github.com/XinshaoAmosWang/ProSelfLC-AT

Outline

Protein understanding example



[3] Gelman, Sam, et al. "Neural networks to learn protein sequence–function relationships from deep mutational scanning data." Proceedings of the National Academy of Sciences (2021).

Thanks for your attention.

Questions and discussions are very welcome.

Personal Info: Homepage and Google Scholar

References

[1] Barron, J. T. A general and adaptive robust loss function. In *CVPR*, 2019.

[2] Bileschi, M. L., Belanger, D., Bryant, D. H., Sanderson, T., Carter, B., Sculley, D., Bateman, A., DePristo, M. A., and Colwell, L. J. Using deep learning to annotate the protein universe. *Nature Biotechnology*, pp. 1–6, 2022.

[3] Gelman, S., Fahlberg, S. A., Heinzelman, P., Romero, P. A., and Gitter, A. Neural networks to learn protein sequence–function relationships from deep mutational scanning data. *Proceedings of the National Academy of Sciences*, 2021.

[4] Hampel, F. R., Ronchetti, E. M., Rousseeuw, P. J., and Stahel, W. The approach based on influence functions. In *Robust Statistics*. Wiley, 1986.

[5] Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. In *NeurIPS Deep Learning and Representation Learning Workshop*, 2015.

[6] Lee, D.-H. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. 2013.

[7] Pereyra, G., Tucker, G., Chorowski, J., Kaiser, Ł., and Hinton, G. Regularizing neural networks by penalizing confident output distributions. In *ICLR Workshop*, 2017.

[8] Reed, S., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., and Rabinovich, A. Training deep neural networks on noisy labels with bootstrapping. In *ICLR Workshop*, 2015.

[9] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

[10] Tanaka, D., Ikami, D., Yamasaki, T., and Aizawa, K. Joint optimization framework for learning with noisy labels. In *CVPR*, 2018.

[11] Wang, X., Kodirov, E., Hua, Y., and Robertson, N. M. Derivative manipulation for general example weighting. *arXiv preprint arXiv:1905.11233*, 2019.

[12] Wang, X., Hua, Y., Kodirov, E., Clifton, D. A., and Robertson, N. M. Proselflc: Progressive self label correction for training robust deep neural networks. In *CVPR*, 2021.

[13] Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.