Detecting and Rectifying Noisy Labels: Similarity-based Methods

Huu-Tien Dang[†] **Nguyen Duc-Thang**[*] **Hoang Thanh-Tung**[◇] **Naoya Inoue**[†‡]
[†]JAIST [*]FPT Software AI Center [◇]Vietnam National University [‡]RIKEN
{s2310417, naoya-i}@jaist.ac.jp

Abstract

Label noise in datasets could damage the performance of neural net training. As the size of modern deep networks grows, there is a growing demand for automated tools for detecting such errors. In this paper, we propose model-agnostic error detection and rectification methods utilizing the penultimate feature from the trained neural network. Our idea is based on an observation that the similarity of penultimate features is higher for within-class data points than that of other class data points, making the probability of label occurrence within a tight-similar cluster, informative to detect and rectify errors. Extensive experiments show our method not only demonstrates high performances across various noises but also automatically rectifies these errors to improve the quality of datasets and model generalization.

1 Introduction

While the majority of knowledge in AI systems is learned through unsupervised learning, supervised learning is an indispensable step in building strong AI systems (c.f. the LeCun's cake). For Large Language Models (LLMs) such as GPTs (Brown et al., 2020), LLaMA (Touvron et al., 2023a,b), and Gemini (Team et al., 2023), supervised learning accounts for only a small fraction of the total computation budget but has a significant impact on the models' performance. Recent research (e.g. Zhou et al., 2023; Gunasekar et al., 2023) finds that high quality training data significantly improves performance while reducing the training cost by orders of magnitude. The need for automated tools for improving the quality of supervised learning data is rising as datasets and models are getting larger at an unprecedented speed.

Real world datasets contain a notable amount of errors (Beyer et al., 2020; Northcutt et al., 2021b). Previous works (Dau et al., 2022; Nguyen-Duc et al., 2023) showed that removing errors from the training set improves the performance of AI models trained on that dataset. Automatic error rectification, however, is an underexplored topic. In this paper, we present a feature based approach for error detection and rectification in large scale datasets. We theoretically show that the similarity between the penultimate feature of a mislabeled data point and its true class data points is larger than that for data points from other classes (Sec. 3.1). Inspired by this observation, we develop simple yet effective similarity-based methods for detecting and rectifying label errors (Sec. 3.2). Extensive experiments demonstrate the superiorities of our methods across various settings (Sec. 4.2). Furthermore, our methods are *posthoc* and *model-agnostic* i.e. they can be applied to any deep neural network (DNN) architectures without the need for retraining.

2 Background and related work

Notation. Let $\mathbf{z} = (\mathbf{x}, \mathbf{y})$ be a data point, where $\mathbf{x} \in \mathcal{X}$ is an input and $\mathbf{y} \in \mathcal{Y}$ is an output. Let $\mathcal{D} = \{\mathbf{z}^{(i)}\}_{i=1}^{n}$ be a N-class noisy training dataset of n data points. Let $f : \mathcal{X} \mapsto \mathcal{Y}$ be a deep model parameterized by θ; $\hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{z}^{(i)}, \theta)$ are optimal parameters of f measured on $\mathcal{D}$, where $\ell : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}^{+}$ be the loss function. In this paper, $\mathbf{g}_{\hat{\theta}}(\mathbf{z}^{(i)}) = \nabla_{\theta} \ell(\mathbf{z}^{(i)}, \hat{\theta})$ is denoted as the gradient of the loss at $\hat{\theta}$ with respect to (w.r.t) θ.

Confident-based Error Detection Methods. Confident-based methods are based on the notion of *confident learning* (Northcutt et al., 2021a) that deriving label quality measurements by using predicted probability distribution (Wang and Mueller, 2022; Kuan and Mueller, 2022; Thyagarajan et al., 2022). Low confidence serves as a heuristic indicating the likelihood of a label noise. Given a data point $\mathbf{z}$ with output label $\mathbf{y} = (y_1, ..., y_k, ...y_N)$, the model's predicted probabilities is $\mathbf{p} = (p_1, ..., p_k, ..., p_N)$ over N classes. Northcutt et al. (2021a) proposed three label quality scoring methods:

(1) *Self-Confidence* (SC) refers to the estimated probability that the input $\mathbf{x}$ belongs to the class associated with its given label k: $\mathrm{SC}(\mathbf{z}, \mathbf{p}) = p_{y_k}$, for $k \in \{1, 2, ..., N\}$.

(2) *Normalized-Margin* (NM) is the quantified difference between the model's estimated probability of the given label and the probability of the most likely class: $\mathrm{NM}(\mathbf{z}, \mathbf{p}) = p_{y_k} - p_{y_{j^*}}$, for $j^* = \arg\max_{j \neq k \in \{1,2,...,N\}} p_{y_j}$

(3) *Confidence-Weighted Entropy* (CE) is the ratio of SC score and the normalized entropy: $\mathrm{CE}(\mathbf{z}, \mathbf{p}) = \frac{p_{y_k}}{\mathcal{H}_N(\mathbf{p})}$, where $\mathcal{H}_N(\mathbf{p}) = -\frac{1}{\log N} \sum_{n=1}^{N} p_n \log(p_n)$.

Gradient-based Error Detection Methods. Koh and Liang (2017) use Influence Function (IF)—a concept from robust statistic (Hampel, 1974)—for measuring the influence of a training data point to weights of a DNN. Dau et al. (2022) proposed a way to adapt IF and its variants i.e. *Gradient Dot Product* (GD; Charpiat et al. (2019a)), *Gradient Cosine* (GC; Charpiat et al. (2019a)), and *Tracing Gradient Decent* (TracIn; Pruthi et al. (2020)), for identifying erroneous in large-scale source code datasets. The idea is the gradients of error data points exhibit significantly large magnitudes and are opposite in direction to the gradients of normal data points. The algorithm computes the influence score of each data point in the noisy dataset with data points in a reference set. A more negative influence score means is more likely to be an error. Nguyen-Duc et al. (2023) use class information to improve the performance and stability of these gradient methods.

(1) $\mathrm{IF}(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}) = -\frac{1}{n} \mathbf{g}_{\hat{\theta}}(\mathbf{z}^{(i)})^\top \mathcal{H}_{\hat{\theta}}^{-1} \mathbf{g}_{\hat{\theta}}(\mathbf{z}^{(j)})$,

(2) $\mathrm{GD}(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}) = \langle \mathbf{g}_{\hat{\theta}}(\mathbf{z}^{(i)}), \mathbf{g}_{\hat{\theta}}(\mathbf{z}^{(j)}) \rangle$,

(3) $\mathrm{GC}(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}) = \cos(\mathbf{g}_{\hat{\theta}}(\mathbf{z}^{(i)}), \mathbf{g}_{\hat{\theta}}(\mathbf{z}^{(j)}))$,

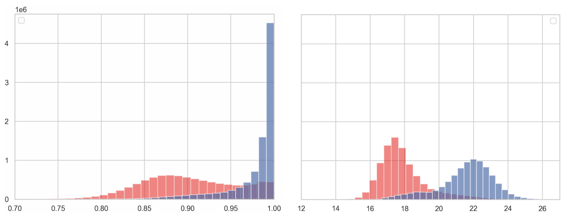
(4) $\mathrm{TracIn}(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}) = \sum_{t=1}^{T} \eta_t \mathrm{GD}(\mathbf{z}^{(i)}, \mathbf{z}^{(j)})$, where $\mathcal{H}_{\hat{\theta}}$ is the hessian matrix, T is the number of epochs, and η_t is the learning rate at epoch t.

Other Error Detection Methods. The rule-based approach (Chu et al., 2013) and statistics-based approach (Huang and He, 2018) are commonly used for structured data such as tabular data. Krishnan et al. (2016) combines active learning and convex models to detect errors on small classification datasets. These methods are not suitable for deep learning, as they assume convexity in the model, and the rules in many large scale datasets are not easy to find and describe.

3 Method

3.1 Observation

We design experiments to randomly corrupt and inject noise into datasets. We then train a deep network using gradient descent on these altered datasets to measure how noisy data points behave on other data points. As an illustrative example in Fig. 1, we observed that the similarity between the mislabeled data points and their true class data point penultimate-layer representations is often higher than other class data points. We find that this phenomenon persists across varying percentages of noises. To complement our observation, we provide a theoretical explanation in Appendix. B.



(a) Cosine similarity (b) Dot product

Figure 1: Distribution of (a) Cosine similarity and (b) Dot product over IMDB (Maas et al., 2011) with 10% noise. Blue bars represent the similarity between mislabeled data points and their true class data points, red bars represent the similarity between mislabeled data points and other class data points. Features are obtained from a trained BERT model.

3.2 Algorithm

Our algorithm is detailed in Algorithm 1. It requires a small auxiliary dataset $\mathcal{D}_{\text{aux}}$, a similarity measure $\sigma(\cdot, \cdot)$. We denote $\phi^{(i)}$ and $\phi^{(j)}$ be the penultimate feature representations of $\mathbf{z}^{(i)}$ and $\mathbf{z}^{(j)}$ obtained from the trained model $f_{\hat{\theta}}$ respectively. We employ two primary similarity measures: Dot product (DOT $= \langle \phi^{(i)}, \phi^{(j)} \rangle$) and Cosine similarity (COS $= \frac{\langle \phi^{(i)}, \phi^{(j)} \rangle}{||\phi^{(i)}|| ||\phi^{(j)}||}$). We denote $\mathcal{S}(\mathcal{D}_{\text{aux}}, \mathbf{z}^{(i)})$ as k most similar to $\mathbf{z}^{(i)}$ in $\mathcal{D}_{\text{aux}}$.

Error Detection. Given a noisy dataset $\mathcal{D}$, for each data point $\mathbf{z}^{(i)} \in \mathcal{D}$ with label $\mathbf{y}^{(i)}$, our algorithm finds $\mathcal{S}(\mathcal{D}_{\text{aux}}, \mathbf{z}^{(i)})$ such that every data point in $\mathcal{D}$ but not in $\mathcal{S}(\mathcal{D}_{\text{aux}}, \mathbf{z}^{(i)})$ is at most similar to $\mathbf{z}^{(i)}$ as the least similar point in $\mathcal{S}(\mathcal{D}_{\text{aux}}, \mathbf{z}^{(i)})$ (line 6). We define a scoring function that return

Algorithm 1 Similarity-based Error Detection and Rectification

```
Require:
 1: $\mathcal{D} = \left\{\mathbf{z}^{(i)}\right\}_{i=1}^{n}$: a noisy dataset
 2: $\mathcal{D}_{\text{aux}} = \left\{\mathbf{z}^{(j)}\right\}_{j=1}^{m}$: an auxiliary dataset.
 3: $\sigma(\cdot,\cdot)$: a similarity measure.
 4: $k$: number of most similar data points.
Ensure: noisy data points in $\mathcal{D}$ are rectified.
    /* Error Detection */
 5: for $\mathbf{z}^{(i)} \in \mathcal{D}$ do
 6:     $\mathcal{S}(\mathcal{D}_{\text{aux}}, \mathbf{z}^{(i)}) = \{\mathbf{z}^{(j)} \in \mathcal{D}_{\text{aux}}\}$
        s.t. $|\mathcal{S}(\mathcal{D}_{\text{aux}}, \mathbf{z}^{(i)})| = k$, and
        $\sigma(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}) \geq \max_{\mathbf{z}'^{(i)} \in \mathcal{D} \setminus \mathcal{S}(\mathcal{D}_{\text{aux}}, \mathbf{z}^{(i)})} \sigma(\mathbf{z}^{(i)}, \mathbf{z}'^{(i)})$
 7:     $\mathbf{s}^{(i)} = \frac{1}{k} \sum_{\mathbf{z}^{(j)} \in \mathcal{S}(\mathcal{D}_{\text{aux}}, \mathbf{z}^{(i)})} \mathbb{I}(\mathbf{y}^{(j)} = \mathbf{y}^{(i)})$
 8: end for
 9: $\mathcal{D}^{\uparrow} = \text{sort}(\mathcal{D}, \text{key} = \mathbf{s}, \text{ascending} = \text{True})$
    /* Error Rectification*/
10: for $\mathbf{z}^{(i)} \in \mathcal{D}^{\uparrow}_{:p}$ do
11:     $\mathbf{z}^{(i)} = \left(\mathbf{x}^{(i)}, \textsc{Mode}(\mathcal{S}(\mathcal{D}_{\text{aux}}, \mathbf{z}^{(i)}))\right)$
12: return $\mathcal{D}$
```

$\mathbf{s}^{(i)}$—the probability of occurrence of label $\mathbf{y}^{(i)}$ in $\mathcal{S}(\mathcal{D}_{\text{aux}}, \mathbf{z}^{(i)})$ (line 7). The indicator $\mathbb{I}(\cdot)$ returns 1 if the condition holds. A lower $\mathbf{s}$ is, more likely a label error. We sort the data points in $\mathcal{D}$ in ascending order of $\mathbf{s}$ and obtain the sorted $\mathcal{D}^{\uparrow}$ (line 9).

Error Rectification. We select the first $p\%$ samples of ranked set $\mathcal{D}^{\uparrow}$ denoted as $\mathcal{D}^{\uparrow}_{:p}$ and define a class decision rule $\textsc{Mode}(\cdot)$ that selects the label in $\mathcal{S}(\mathcal{D}_{\text{aux}}, \mathbf{z}^{(i)})$ has the highest probability and greater than threshold τ. Otherwise, the label of $\mathbf{z}^{(i)}$ remains unchanged (line 11).

4 Experiment

4.1 Experiment Setting

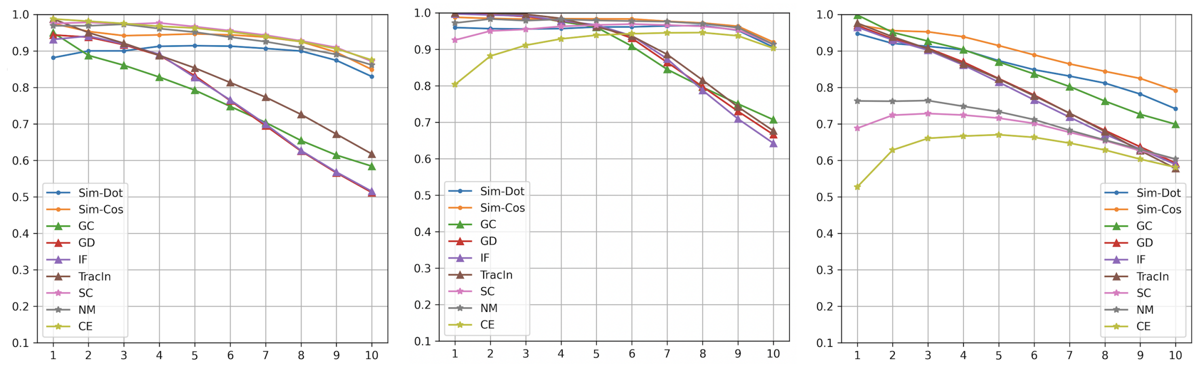
Dataset and Model. We evaluate our method on two common benchmarks: Sentiment Analysis on IMDB (Maas et al., 2011) and Short Text Classification on Snippets (Phan et al., 2008). We use BERT (Devlin et al., 2019) as the standard model for all settings.

Modeling Realistic Noise. We construct three realistic, human-originated types of noise: (1) *Uniform noise*: we randomly select data points and change the label to a different class. (2) *Systematic ambiguity noise*: we establish a rule h, which maps data points in a specific class to another fixed one. This means that the labels of selected instances in class i are flipped to $h(i)$. To ensure distinctiveness, the mapping function h adheres to the condition $h(i) \neq h(j) \forall i, j = \{1, ..., N\}$, and $i \neq j$. This noise models the situations where inputs from multiple annotators are often aggregated, the resulting differences in annotations can serve as a model of systematic noise derived from human disagreements. (3) *Concentrated noise*: we select data points that are densely clustered and change their labels to target labels. We simulate scenarios where the datasets are poisoned by malicious to evaluate the sanitization ability of methods against data poisoning attacks.

Setting. For each dataset, we construct groups of various sizes of noisy samples by corrupting the label of $p\%$ of the original training data. We construct the auxiliary dataset $\mathcal{D}_{\text{aux}}$ by randomly selecting m samples from the validation set. We fine-tune BERT on noisy dataset $\mathcal{D}$ and select the best checkpoint measure on the validation set. We select top $t\%$ ranked samples in $\mathcal{D}^{\uparrow}_{:p}$ and use error detection accuracy for evaluation. After rectifying/removing ranked samples (potentially noisy samples), we re-train the model and report the test accuracy and error reduction rate. Details of the dataset, model, and implementations are in Appendix. A.

4.2 Main Result and Analysis

Error detection accuracy. (1) Fig. 2 shows the error detection accuracy of methods with three types of noise with different percentages. As a result, when t increases the performance of gradient-based methods *drastically decreases*. This pattern is observed in all three types of noise across different percentages and in both Snippets (Fig. 4) and IMDB (Fig. 5). This result shows that the gradient-based methods are unstable and less inconsistent. (2) Confident-based methods are precise with uniform noises and systematic ambiguity noise yet struggle with concentrated noise (Fig. 2c). (3) Sim-Cos and Sim-Dot have high detection accuracy and slightly decrease when t increases with difference noise. This confirmed that the Similarity-based methods are effective and more robust to ambiguity and concentrated noises than gradient-/confident-based methods. (4) We observe that Sim-Dot often has low detection accuracy on IMDB (Fig. 5). We theoretically proved that for classification datasets with N classes, the similarity of within-class data



(a) Random noise $p = 5\%$ (b) Systematic ambiguity noise $p = 10\%$ (c) Concentrated noise $p = 20\%$

Figure 2: Error detection accuracy of methods measure on Snippets. The x-axis in the figures presents the change of t from $10 \to 100\%$.

Table 1: Test accuracy after remove/rectify potential noise samples on Snippets with 20% noise. The **best** and runner-up are marked.

Method	Random noise		Ambiguity noise		Concentrated noise	
	Removed	Rectified	Removed	Rectified	Removed	Rectified
Acc. (under noise)	88.64 (+0.00)	—	82.50 (+0.00)	—	79.38 (+0.00)	—
Confident-W. Entropy	83.02 (-5.62)	—	84.38 (+1.88)	—	77.50 (-1.88)	—
Normalize-Margin	87.19 (-1.45)	—	87.01 (+4.51)	—	77.89 (-1.49)	—
Self-Confidence	86.05 (-2.59)	—	**87.36** (+4.86)	—	78.02 (-1.36)	—
Influence Function	83.24 (-5.40)	—	81.71 (-0.79)	—	79.51 (+0.13)	—
Gradient-Cosine	73.81 (-14.83)	—	83.33 (+0.83)	—	**82.23** (+2.85)	—
Gradient-Dot	86.53 (-2.11)	—	82.01 (-0.49)	—	78.68 (-0.70)	—
TracIn	85.48 (-3.16)	—	81.71 (-0.79)	—	76.67 (-2.71)	—
Sim-Cos	**89.43** (+0.79)	**87.85** (-0.79)	85.00 (+2.50)	83.73 (+1.23)	81.53 (+2.15)	**83.38** (+4.00)
Sim-Dot	86.71 (-1.93)	87.32 (-1.32)	82.98 (+0.48)	**83.95** (+1.45)	81.53 (+2.15)	81.97 (+2.59)

points is approximately $N - 1$ times larger than other class data points. For IMDB where $N = 2$, this fraction becomes approximately 1. That explains why Sim-Dot does not work well on IMDB. Mathematical details are in Appendix. B. (5) Sim-Cos, consistently outperforms Sim-Dot by a large margin for all settings. We explain from the standpoint of *feature normalization*. By definition, Cosine similarity can be seen as the normalized Dot product. In Fig. 4 (b), we empirically show that the noisy samples have L_2-norm smaller than normal samples. Therefore, when dividing the feature of data points by its norm, the similarity between noisy and normal data points tends to be larger, leading to a more distinct distribution of similarities.

Improving datasets and model generalization. Tab.1 shows a significant improvement in the test accuracy when removing/rectifying concentrated noise and systematic ambiguity noise. Nevertheless, a counter-intuitive observation regarding random noise shows that removing/rectifying noise reduces the generalization of models, even when detection accuracy is high. We posit that deep models are robust to massive random noise (Rolnick et al., 2017), then as the training process, the model also memorizes the noise, approaches the optimum, and the gradient of noise becomes smaller. The effect of noise, therefore, also decreases as the model converges. When removing noise, the model degrades the feature representation of noise samples and loses the generalization to unseen samples.

5 Conclusion

We introduce similarity-based algorithms for detecting and rectifying errors on large-scale datasets. We theoretically show that the similarity between the penultimate feature's data points is useful for detecting errors. Experiment results demonstrated the superior performance of our methods, and their capability to improve datasets quality and model generalization.

Limitations

We discuss the limitations of similarity-based methods: (1) The optimal detection accuracy of Sim-Cos and Sim-Dot unfortunately based on empirical validation, and depends on the choice of k and $\mathcal{D}_{\text{aux}}$ and also with different datasets and model architectures. (2) The generalization of models under the removal or rectification of noise remains uncertain, due to the limited exploration of datasets.

Ethics Statement

We consider only the public datasets and create artificial noises for evaluation. We do not pose any concern about the quality of the original datasets.

Acknowledgements

We thank Thu Tran and the anonymous reviewers for their constructive feedback.

References

Naman Agarwal, Brian Bullins, and Elad Hazan. 2017. Second-order stochastic optimization for machine learning in linear time. *The Journal of Machine Learning Research*, 18(1):4148–4187.

Lucas Beyer, Olivier J. Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aäron van den Oord. 2020. Are we done with imagenet? *CoRR*, abs/2006.07159.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Guillaume Charpiat, Nicolas Girard, Loris Felardos, and Yuliya Tarabalka. 2019a. Input similarity from the neural network perspective. *Advances in Neural Information Processing Systems*, 32.

Guillaume Charpiat, Nicolas Girard, Loris Felardos, and Yuliya Tarabalka. 2019b. Input similarity from the neural network perspective. *Advances in Neural Information Processing Systems*, 32.

Xu Chu, Ihab F. Ilyas, and Paolo Papotti. 2013. Holistic data cleaning: Putting violations into context. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 458–469.

Anh TV Dau, Nghi DQ Bui, Thang Nguyen-Duc, and Hoang Thanh-Tung. 2022. Towards using data-influence methods to detect noisy samples in source code corpora. In *37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–3.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. 2023. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*.

Frank R. Hampel. 1974. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69(346):383–393.

Kazuaki Hanawa, Sho Yokoi, Satoshi Hara, and Kentaro Inui. 2021. Evaluation of similarity-based explanations. In *International Conference on Learning Representations*.

Zhipeng Huang and Yeye He. 2018. Auto-detect: Data-driven error detection in tables. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1377–1392.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR.

Sanjay Krishnan, Jiannan Wang, Eugene Wu, Michael J. Franklin, and Ken Goldberg. 2016. Activeclean: Interactive data cleaning for statistical modeling. *Proc. VLDB Endow.*, 9(12):948–959.

Johnson Kuan and Jonas Mueller. 2022. Model-agnostic label quality scoring to detect real-world label errors. In *ICML DataPerf Workshop*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Thang Nguyen-Duc, Hoang Thanh-Tung, Quan Hung Tran, Dang Huu-Tien, Hieu Nguyen, Anh T. V. Dau, and Nghi Bui. 2023. Class based influence functions for error detection. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1204–1218, Toronto, Canada. Association for Computational Linguistics.

Curtis Northcutt, Lu Jiang, and Isaac Chuang. 2021a. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, 70:1373–1411.

Curtis G Northcutt, Anish Athalye, and Jonas Mueller. 2021b. Pervasive label errors in test sets destabilize machine learning benchmarks. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

Pouya Pezeshkpour, Sarthak Jain, Byron Wallace, and Sameer Singh. 2021. An empirical comparison of instance attribution methods for NLP. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 967–975, Online. Association for Computational Linguistics.

Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. 2008. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of the 17th international conference on World Wide Web*, pages 91–100.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. 2020. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33:19920–19930.

David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. 2017. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Aditya Thyagarajan, Elías Snorrason, Curtis Northcutt, and Jonas Mueller. 2022. Identifying incorrect annotations in multi-label classification data. *arXiv preprint arXiv:2211.13895*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Wei-Chen Wang and Jonas Mueller. 2022. Detecting label errors in token classification data. *arXiv preprint arXiv:2210.03920*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2023. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206*.

A Implementation detail

A.1 Dataset and Model

Snippets (Phan et al., 2008) is an open dataset of web search snippets retrieved from Google Search with 8 domains including Business, Computers, Culture-Arts-Entertainment, Education-Science, Engineering, Health, Politics-Society, and Sports. The training data and testing data include $10,060$ and $2,280$ snippets respectively. For validation purposes, we randomly split original training data into train/valid with the ratio of $8048/2012$. The dataset can be found at `http://jwebpro.sourceforge.net/data-web-snippets.tar.gz`.

IMDB (Maas et al., 2011) is the most common benchmark for Sentiment Analysis task. IMDB includes 50000 reviews from the Internet Movie Database website with original 25000 negative and 25000 positive reviews. For validation purposes, we randomly split into training, validation, and test sets of sizes 15000, 5000, and 25000. The IMDB dataset can be found at `https://ai.stanford.edu/~amaas/data/sentiment/`

BERT (Devlin et al., 2019) stands for Bidirectional Encoder Representations from Transformers. BERT is one of the most standard used pre-trained model for language understanding tasks. In all settings, we use BERT base uncased version.

A.2 Experiment detail

BERT was trained with AdamW (Loshchilov and Hutter, 2019) with learning rate $\eta = 5e-5$, momentum $\beta = (0.9, 0.999)$, cross entropy loss, batch-size of 16 with 15 epochs. For regularization, we use Dropout (Srivastava et al., 2014) of 0.2. We choose $p = \{5\%, 10\%, 20\%\}$, $m = 1000$, $k = \{1, 2, 5, 10, 20, 50, 100, 200\}$, $t = \{10\%, 20\%, ..., 100\%\}$, and $\tau = 0.8$. We compute the IF score for BERT with the last layer gradient as previous works (Pezeshkpour et al., 2021; Hanawa et al., 2021) and use LiSSA (Agarwal et al., 2017) to approximate the Hessian. For TracIn method, we calculate the influence score from the first epoch to the best epoch. We run experiments on 4 seeds = $\{16, 32, 64, 128\}$ and aggregate the results by taking the mean of these 4 seeds. A Nvidia RTX GeForce 3090 Ti was used to run experiments. Our implementation was attached to the supplementary materials.

B Theoretical Analysis

We consider a deep network received input $\mathbf{x} \in \mathbb{R}^d$, where d is the input dimension. Let δ be the softmax activation function, $\mathbf{b} \in \mathbb{R}^N$ be the bias. Given an output function ϕ_W parameterized by $W \in \mathbb{R}^{N \times d}$. For two inputs $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ we have two output vectors $\phi_W(\mathbf{x}^{(i)})$ and $\phi_W(\mathbf{x}^{(j)})$ respectively. As seen by a deep network, we can measure the influence between $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ by quantifying how much $\phi_W(\mathbf{x}^{(i)})$ change would change $\phi_W(\mathbf{x}^{(j)})$ as well. If $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ have high similarity, then $\mathbf{x}^{(i)}$ have high influence on $\mathbf{x}^{(j)}$ and changing $\phi_W(\mathbf{x}^{(i)})$ have large effect on $\phi_W(\mathbf{x}^{(j)})$. Otherwise, if they have low similarity, then $\mathbf{x}^{(i)}$ have low influence on $\mathbf{x}^{(j)}$ and changing $\phi_W(\mathbf{x}^{(i)})$ have small effect on $\phi_W(\mathbf{x}^{(j)})$. To measure the similarity between data points, we employ a symmetric kernel proposed by Charpiat et al. (2019b): The Inner Product

$$\mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \left\langle \nabla_W \phi_W(\mathbf{x}^{(i)}), \nabla_W \phi_W(\mathbf{x}^{(j)}) \right\rangle, \tag{1}$$

We choice ϕ is the Cross Entropy between softmax activation output $\hat{\mathbf{y}} = \delta(W\mathbf{x} + \mathbf{b})$ and true distribution $\mathbf{y}$. We have $\phi_W(\mathbf{x}) = \ell(\hat{\mathbf{y}}, \mathbf{y}; W)$. For simplicity, we remove the bias term. Let denote $\mathbf{u} = W\mathbf{x}$ and $\ell(\hat{\mathbf{y}}, \mathbf{y}) = \ell(\hat{\mathbf{y}}, \mathbf{y}; W)$. Using the chain rule:

$$\begin{aligned} \nabla_W \ell(\hat{\mathbf{y}}, \mathbf{y}) &= \text{vec}\left(\frac{\partial \ell(\hat{\mathbf{y}}, \mathbf{y})}{\partial W}\right) \\ &= \text{vec}\left(\frac{\partial \ell(\hat{\mathbf{y}}, \mathbf{y})}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial W}\right) \\ &= \nabla_{\mathbf{u}} \ell(\hat{\mathbf{y}}, \mathbf{y}) \mathbf{x}^\top, \end{aligned} \tag{2}$$

where $\text{vec}\left(\frac{\partial \ell(\hat{\mathbf{y}}, \mathbf{y})}{\partial W}\right)$ is the vectorization of the derivative of the loss ℓ with respect to W. The partial $\frac{\partial \ell(\hat{\mathbf{y}}, \mathbf{y})}{\partial \mathbf{u}}$ is

$$\frac{\partial \ell(\hat{\mathbf{y}}, \mathbf{y})}{\partial \mathbf{u}} = \frac{\partial \ell(\hat{\mathbf{y}}, \mathbf{y})}{\partial \hat{\mathbf{y}}} \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{u}} \tag{3}$$

The first term on the right hand side of Eqn. 3 is partial derivatives of the loss w.r.t the predicted output $\hat{\mathbf{y}}$. Regarding to the fact: $\mathbf{y}$ is the one-hot vector present label k has element $y_k = 1$ and $y_i = 0$ if $i \neq k$. We have:

$$\begin{aligned} \frac{\partial \ell(\hat{\mathbf{y}}, \mathbf{y})}{\partial \hat{\mathbf{y}}} &= \left[\frac{\partial \ell(\hat{\mathbf{y}}, \mathbf{y})}{\partial \hat{y}_1} \cdots \frac{\partial \ell(\hat{\mathbf{y}}, \mathbf{y})}{\partial \hat{y}_k} \cdots \frac{\partial \ell(\hat{\mathbf{y}}, \mathbf{y})}{\partial \hat{y}_N}\right] \\ &= \left[0 \cdots \frac{1}{\hat{y}_k} \cdots 0\right] \end{aligned} \tag{4}$$

The second term is the matrix comprises partial derivatives of the predicted output $\hat{\mathbf{y}}$ w.r.t $\mathbf{u}$. Regarding to the fact:

$$\frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{u}} = \begin{bmatrix} \frac{\partial \hat{y}_1}{\partial u_1} & \frac{\partial \hat{y}_1}{\partial u_2} & \cdots & \frac{\partial \hat{y}_1}{\partial u_N} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \hat{y}_k}{\partial u_1} & \frac{\partial \hat{y}_k}{\partial u_2} & \cdots & \frac{\partial \hat{y}_k}{\partial u_N} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \hat{y}_N}{\partial u_1} & \frac{\partial \hat{y}_N}{\partial u_2} & \cdots & \frac{\partial \hat{y}_N}{\partial u_N} \end{bmatrix} \tag{5}$$

Substitute Eqn. 5 and Eqn. 4 into Eqn. 3, we get

$$\frac{\partial \ell(\hat{\mathbf{y}}, \mathbf{y})}{\partial \mathbf{u}} = \left[\frac{\partial \ell(\hat{\mathbf{y}}, \mathbf{y})}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial u_1} \quad \cdots \quad \frac{\partial \ell(\hat{\mathbf{y}}, \mathbf{y})}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial u_k} \quad \cdots \quad \frac{\partial \ell(\hat{\mathbf{y}}, \mathbf{y})}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial u_N}\right] \tag{6}$$

Given a softmax activation function δ for class k: Case 1: $i = k$, we compute the derivative of softmax output $\hat{y}_k$ w.r.t u_k:

$$\begin{aligned} \frac{\partial \hat{y}_k}{\partial u_k} &= \frac{\partial}{\partial u_k}\left(\frac{e^{u_k}}{\sum_{i=1}^N e^{u_i}}\right) \\ &= \frac{e^{u_k}\left(\sum_{i=1}^N e^{u_i}\right) - e^{u_k} \cdot e^{u_k}}{\left(\sum_{i=1}^N e^{u_i}\right)^2} \\ &= \hat{y}_k(1 - \hat{y}_k) \end{aligned} \tag{7}$$

Case 2: $i \neq k$, we compute the derivative of softmax output $\hat{y}_k$ w.r.t u_i:

$$\frac{\partial \hat{y}_k}{\partial u_i} = \frac{\partial}{\partial u_i}\left(\frac{e^{u_k}}{\sum_{i=1}^N e^{u_i}}\right) \tag{8}$$

Using the chain rule, we get:

$$\frac{\partial \hat{y}_k}{\partial u_i} = -\frac{e^{u_k} \cdot e^{u_i}}{\left(\sum_{i=1}^N e^{u_i}\right)^2} = -\hat{y}_k \hat{y}_i \tag{9}$$

Substitute Eqn. 7 and Eqn. 9 into Eqn. 6, we get a column vector:

$$\nabla_{\mathbf{u}} \ell(\hat{\mathbf{y}}, \mathbf{y}) = \left[-\hat{y}_1 \cdots 1 - \hat{y}_k \cdots - \hat{y}_N\right]^\top \tag{10}$$

The inner product kernel in Eqn. 2 become:

$$\mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \overbrace{\nabla_{\mathbf{u}}\ell(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)})^\top \nabla_{\mathbf{u}}\ell(\hat{\mathbf{y}}^{(j)}, \mathbf{y}^{(j)})}^{\mathbf{G}^{(ij)}} \cdot (\mathbf{x}^{(j)\top}\mathbf{x}^{(i)}) \tag{11}$$

We denote $\mathbf{G}^{(ij)}$ as the dot product of two gradients of the loss at $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$. Suppose input $\mathbf{x}^{(i)}$ and input $\mathbf{x}^{(j)}$ have label k and k' corresponding. We have

$$\mathbf{G}^{(ij)} = \begin{bmatrix} -\hat{y}_1^{(i)} \cdots 1 - \hat{y}_k^{(i)} \cdots -\hat{y}_N^{(i)} \end{bmatrix} \begin{bmatrix} -\hat{y}_1^{(j)} \\ \vdots \\ 1 - \hat{y}_{k'}^{(j)} \\ \vdots \\ -\hat{y}_N^{(j)} \end{bmatrix} \tag{12}$$

we consider 2 cases:
If $k = k'$:

$$\begin{aligned} \mathbf{G}_{k=k'}^{(ij)} &= \hat{y}_1^{(i)}\hat{y}_1^{(j)} + \cdots + (1 - \hat{y}_k^{(i)})(1 - \hat{y}_k^{(j)}) \\ &\quad + \cdots + \hat{y}_N^{(i)}\hat{y}_N^{(j)} \\ &= (1 - \hat{y}_k^{(i)})(1 - \hat{y}_k^{(j)}) + \sum_{n=1, n\neq k}^{N} \hat{y}_n^{(i)}\hat{y}_n^{(j)} \end{aligned} \tag{13}$$

If $k \neq k'$:

$$\begin{aligned} \mathbf{G}_{k\neq k'}^{(ij)} &= \hat{y}_1^{(i)}\hat{y}_1^{(j)} + \cdots + (1 - \hat{y}_k^{(i)})(-\hat{y}_k^{(j)}) + \\ &\quad (1 - \hat{y}_{k'}^{(j)})(-\hat{y}_{k'}^{(i)}) + \cdots + \hat{y}_N^{(i)}\hat{y}_N^{(j)} \\ &= \hat{y}_k^{(j)}(\hat{y}_k^{(i)} - 1) + \hat{y}_{k'}^{(i)}(\hat{y}_{k'}^{(j)} - 1) \\ &\quad + \sum_{n=1, n\neq k, n\neq k'}^{N} \hat{y}_n^{(i)}\hat{y}_n^{(j)} \end{aligned} \tag{14}$$

During the training process, the model is more confident about the labels of data points, indicating the value of $\hat{y}_k^{(i)}$ and $\hat{y}_{k'}^{(j)}$ being closer to 1. Assume a well-trained model, and $\hat{y}_k^{(i)} \approx \hat{y}_{k'}^{(j)} = \alpha$; $\hat{y}_n^{(i)} = \hat{y}_n^{(j)} \approx \epsilon = \frac{1-\alpha}{N-1}$. ($n \neq k$ and $n \neq k'$). Substitute these values into Eqn. 13 and Eqn. 14, we get :

$$\mathbf{G}_{k=k'}^{(ij)} \approx (1-\alpha)^2 + \epsilon^2(N-1) \tag{15}$$

$$\mathbf{G}_{k\neq k'}^{(ij)} \approx -\frac{N(1-\alpha)^2}{(N-1)^2} = -\epsilon^2 N \tag{16}$$

As N become very large with deep learning dataset and ϵ small, the magnitude of $\mathbf{G}_{k\neq k'}^{(ij)}$ is close to 0 for $k \neq k'$. That means data points in different classes tend to be pushed into different orthogonal sub-spaces. Let's consider the the magnitude of $\mathbf{G}_{k=k'}^{(ij)}$ and $\mathbf{G}_{k\neq k'}^{(ij)}$, divide $|\mathbf{G}_{k=k'}^{(ij)}|$ by $|\mathbf{G}_{k\neq k'}^{(ij)}|$, we get:

$$\begin{aligned} \frac{\mathcal{K}_{k=k'}^{(ij)}}{\mathcal{K}_{k\neq k'}^{(ij)}} = \frac{|\mathbf{G}_{k=k'}^{(ij)}|}{|\mathbf{G}_{k\neq k'}^{(ij)}|} &\approx \frac{|(1-\alpha)^2 + \epsilon^2(N-1)|}{|-\epsilon^2 N|} \\ &\approx \frac{\epsilon^2(N-1)^2 + \epsilon^2(N-1)}{\epsilon^2 N} \\ &\approx N - 1 \end{aligned} \tag{17}$$

Here, the kernel $\mathcal{K}_{k=k'}^{(ij)}$ ($\mathcal{K}_{k\neq k'}^{(ij)}$) represents the similarity between $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ when they share the same label (different labels). This explains why mislabeled data points are often more similar to true class data points than data points in other classes.

Similarity-based methods do not work well on IMDB (Fig. 5). For the IMDB dataset with $N = 2$, the fraction are approximately 1. Hence, the similarity between data points within the same class and those in the remaining class does not have a significant gap.

C Ablation studies

C.1 The effect of the size of $\mathcal{D}_{\text{aux}}$, the number of k, and τ.

The effect of the size of $\mathcal{D}_{\text{aux}}$. We change the size of $\mathcal{D}_{\text{aux}}$ from 100 to 1500, fix $k = 100$. Tab. 2 and Tab. 3 show the change in error detection accuracy as the size of $\mathcal{D}_{\text{aux}}$ changes. We observed that: (1) the detection accuracy of the methods increases as the size of $\mathcal{D}_{\text{aux}}$ increases. This is true for both Snippets and IMDB, Sim-Cos and Sim-Dot, and for all levels of noise $\{5\%, 10\%, 20\%\}$. (2) The accuracy gains are larger for smaller $\mathcal{D}_{\text{aux}}$ values. (3) Sim-Cos outperforms Sim-Dot for all noise levels and most $\mathcal{D}_{\text{aux}}$ values.

The effect of k. Tab. 4 and Tab. 5 show that Sim-Cos and Sim-Dot have accuracy increase as k increases for all noise levels $\{5\%, 10\%, 20\%\}$ and for most values of k. The accuracy gains are often larger for smaller values of k. Sim-Cos outperforms Sim-Dot, especially for larger values of k for all noise levels and most values of k. The impact of noise on detection accuracy with Snippets is generally small, but it increases with k. For Sim-Cos with 20% noise in Tab. 4, the accuracy starts to

Table 2: The effect of the size of $\mathcal{D}_{\text{aux}}$, setting with Snippets.

Method	5% noise	10% noise	20% noise
Sim-Cos@100	16.91	22.63	31.69
Sim-Cos@200	51.99	40.67	65.25
Sim-Cos@300	57.46	56.96	80.23
Sim-Cos@400	60.45	72.38	86.45
Sim-Cos@500	75.37	78.60	86.88
Sim-Cos@600	77.36	78.60	89.93
Sim-Cos@700	82.58	78.48	93.10
Sim-Cos@800	85.07	78.60	92.91
Sim-Cos@900	86.07	79.72	93.16
Sim-Cos@1000	86.56	80.01	93.28
Sim-Cos@1500	88.06	80.97	94.28
Sim-Dot@100	16.91	22.63	31.69
Sim-Dot@200	51.74	40.92	64.26
Sim-Dot@300	57.96	56.59	79.42
Sim-Dot@400	61.19	70.64	86.82
Sim-Dot@500	74.62	78.23	87.01
Sim-Dot@600	76.36	78.10	88.87
Sim-Dot@700	81.34	77.36	92.91
Sim-Dot@800	82.83	77.36	92.54
Sim-Dot@900	83.83	78.10	92.17
Sim-Dot@1000	84.08	78.73	92.41
Sim-Dot@1500	85.32	80.34	93.78

Table 3: The effect of the size of $\mathcal{D}_{\text{aux}}$ on detection performance with IMDB.

Method	5% noise	10% noise	20% noise
Sim-Cos@100	6.40	8.55	20.10
Sim-Cos@200	58.10	74.00	78.55
Sim-Cos@300	60.30	75.60	78.55
Sim-Cos@400	60.10	75.65	78.55
Sim-Cos@500	60.20	75.55	78.55
Sim-Cos@600	59.90	75.70	78.52
Sim-Cos@700	60.20	75.55	78.45
Sim-Cos@800	60.10	75.60	78.47
Sim-Cos@900	60.20	75.60	78.42
Sim-Cos@1000	59.90	75.50	78.52
Sim-Cos@1500	60.10	75.55	78.50
Sim-Dot@100	6.40	8.55	20.10
Sim-Dot@200	57.80	66.65	77.72
Sim-Dot@300	58.00	65.70	77.80
Sim-Dot@400	58.10	68.20	78.47
Sim-Dot@500	57.90	67.10	77.95
Sim-Dot@600	58.00	73.65	78.35
Sim-Dot@700	58.50	69.45	78.52
Sim-Dot@800	57.90	65.70	77.87
Sim-Dot@900	58.20	68.65	77.47
Sim-Dot@1000	59.90	68.60	77.55
Sim-Dot@1500	57.90	68.05	78.10

decrease after $k = 20$. This suggests that using larger k can harm the accuracy when the data is very noisy. For Sim-Dot with 10% and 20% noise in Tab. 4, the accuracy changes are less consistent across different values of k. This suggests that Sim-Dot may be more sensitive to the choice of k.

Table 4: Error detection accuracy of Sim-Cos and Sim-Dot changes as k changes with Snippets.

Method	5% noise	10% noise	20% noise
Sim-Cos@k=1	77.61	82.46	87.69
Sim-Cos@k=2	88.55	85.57	92.23
Sim-Cos@k=5	89.30	86.31	94.53
Sim-Cos@k=10	89.30	86.94	94.65
Sim-Cos@k=20	89.55	86.69	94.53
Sim-Cos@k=50	88.06	82.09	94.46
Sim-Cos@k=100	86.56	80.10	93.28
Sim-Cos@k=200	75.12	78.23	86.76
Sim-Dot@k=1	80.09	82.09	88.75
Sim-Dot@k=2	81.34	83.45	90.05
Sim-Dot@k=5	82.58	84.70	91.29
Sim-Dot@k=10	84.57	85.32	92.60
Sim-Dot@k=20	85.57	84.57	93.10
Sim-Dot@k=50	86.07	81.96	93.59
Sim-Dot@k=100	84.08	78.73	92.41
Sim-Dot@k=200	75.12	77.73	86.76

Table 5: The effect of k on detection performance with IMDB.

Method	5% noise	10% noise	20% noise
Sim-Cos@k=1	28.50	50.85	66.15
Sim-Cos@k=2	49.10	66.80	69.72
Sim-Cos@k=5	57.80	72.05	75.65
Sim-Cos@k=10	59.50	74.10	76.77
Sim-Cos@k=20	59.50	74.90	77.42
Sim-Cos@k=50	60.20	75.55	78.42
Sim-Cos@k=100	59.90	75.50	78.52
Sim-Cos@k=200	60.00	75.45	78.57
Sim-Dot@k=1	57.50	67.25	75.17
Sim-Dot@k=2	57.00	67.30	75.30
Sim-Dot@k=5	57.20	67.95	75.80
Sim-Dot@k=10	57.10	67.75	76.00
Sim-Dot@k=20	57.10	68.15	76.27
Sim-Dot@k=50	57.90	68.55	76.95
Sim-Dot@k=100	59.90	68.60	77.55
Sim-Dot@k=200	58.20	68.25	78.40

The effect of τ on error reduction rate. We analyze the effect of τ on the error reduction rate. We vary the number of $\tau = \{0.5, 0.6, 0.7, 0.8, 0.9, 0.99\}$. From results in Tab. 6 and Tab. 7, we see that for both Snippets and IMDB, the change of τ has minimal impact on the error reduction rate. Generally, the reduction rate is higher when the noise level is higher.

C.2 Correlation.

We calculate the Spearman correlation between ranking scores assigned to samples by detection methods. Fig. 3a shows that gradient-based, confident-based, and similarity-based methods have low Spearman correlation with each other. Confident-based and Gradient-based methods have

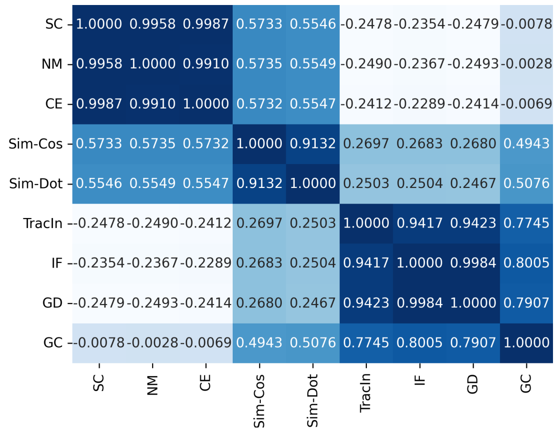
Table 6: Error Reduction Rate on Snippets.

Method	5% noise	10% noise	20% noise
Sim-Cos@τ=0.5	66.16	58.95	80.98
Sim-Cos@τ=0.6	50.74	58.83	77.00
Sim-Cos@τ=0.7	48.00	58.20	76.75
Sim-Cos@τ=0.8	46.51	57.83	76.69
Sim-Cos@τ=0.9	46.51	57.46	76.69
Sim-Cos@τ=0.99	46.51	57.46	76.69
Sim-Dot@τ=0.5	42.03	55.72	76.25
Sim-Dot@τ=0.6	43.03	56.34	76.38
Sim-Dot@τ=0.7	45.52	57.08	76.63
Sim-Dot@τ=0.8	46.26	57.33	76.69
Sim-Dot@τ=0.9	46.51	57.46	76.69
Sim-Dot@τ=0.99	46.51	57.46	76.69

Table 7: Error Reduction Rate on IMDB.

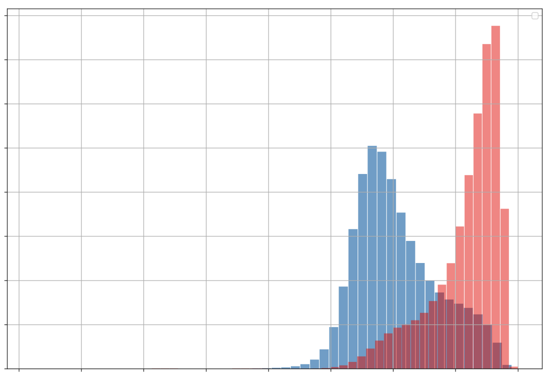
Method	5% noise	10% noise	20% noise
Sim-Cos@τ=0.5	28.70	51.00	59.37
Sim-Cos@τ=0.6	28.40	41.15	59.12
Sim-Cos@τ=0.7	28.40	41.15	59.12
Sim-Cos@τ=0.8	28.40	41.15	59.12
Sim-Cos@τ=0.9	28.40	41.15	59.12
Sim-Cos@τ=0.99	28.40	41.15	59.12
Sim-Dot@τ=0.5	28.40	41.15	59.12
Sim-Dot@τ=0.6	28.40	41.15	59.12
Sim-Dot@τ=0.7	28.40	41.15	59.12
Sim-Dot@τ=0.8	28.40	41.15	59.12
Sim-Dot@τ=0.9	28.40	41.15	59.12
Sim-Dot@τ=0.99	28.40	41.15	59.12

negative Spearman correlation, indicating they are very different in ranking. We observed the same phenomenon in datasets across levels and types of noise.

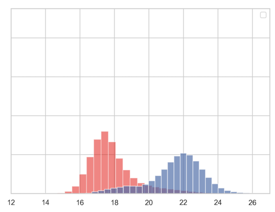


	SC	NM	CE	Sim-Cos	Sim-Dot	TracIn	IF	GD	GC
SC	1.0000	0.9958	0.9987	0.5733	0.5546	-0.2478	-0.2354	-0.2479	-0.0078
NM	0.9958	1.0000	0.9910	0.5735	0.5549	-0.2490	-0.2367	-0.2493	-0.0028
CE	0.9987	0.9910	1.0000	0.5732	0.5547	-0.2412	-0.2289	-0.2414	-0.0069
Sim-Cos	0.5733	0.5735	0.5732	1.0000	0.9132	0.2697	0.2683	0.2680	0.4943
Sim-Dot	0.5546	0.5549	0.5547	0.9132	1.0000	0.2503	0.2504	0.2467	0.5076
TracIn	-0.2478	-0.2490	-0.2412	0.2697	0.2503	1.0000	0.9417	0.9423	0.7745
IF	-0.2354	-0.2367	-0.2289	0.2683	0.2504	0.9417	1.0000	0.9984	0.8005
GD	-0.2479	-0.2493	-0.2414	0.2680	0.2467	0.9423	0.9984	1.0000	0.7907
GC	-0.0078	-0.0028	-0.0069	0.4943	0.5076	0.7745	0.8005	0.7907	1.0000

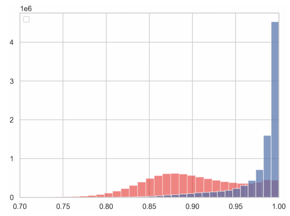
(a) Spearman correlation of methods on Snippets with random noise 20%.



(b) Distribution of L_2-norm ($||\phi||_2$) of features



(c) Distribution of similarity of $\langle \phi^{(i)}, \phi^{(j)} \rangle$



(d) Distribution of similarity of $\frac{\langle \phi^{(i)}, \phi^{(j)} \rangle}{||\phi^{(i)}||_2 ||\phi^{(j)}||_2}$

Figure 3: Visualization of (a) Spearman correlation, (b) the L_2-norm of noisy data points (presented by blue bars) and normal data points (presented by red bars). Using normalization (Cosine; (d)) enables the differentiation of the similarity distribution, therefore, enhancing detection accuracy compared to the unnormalized approach (Dot; (c)).

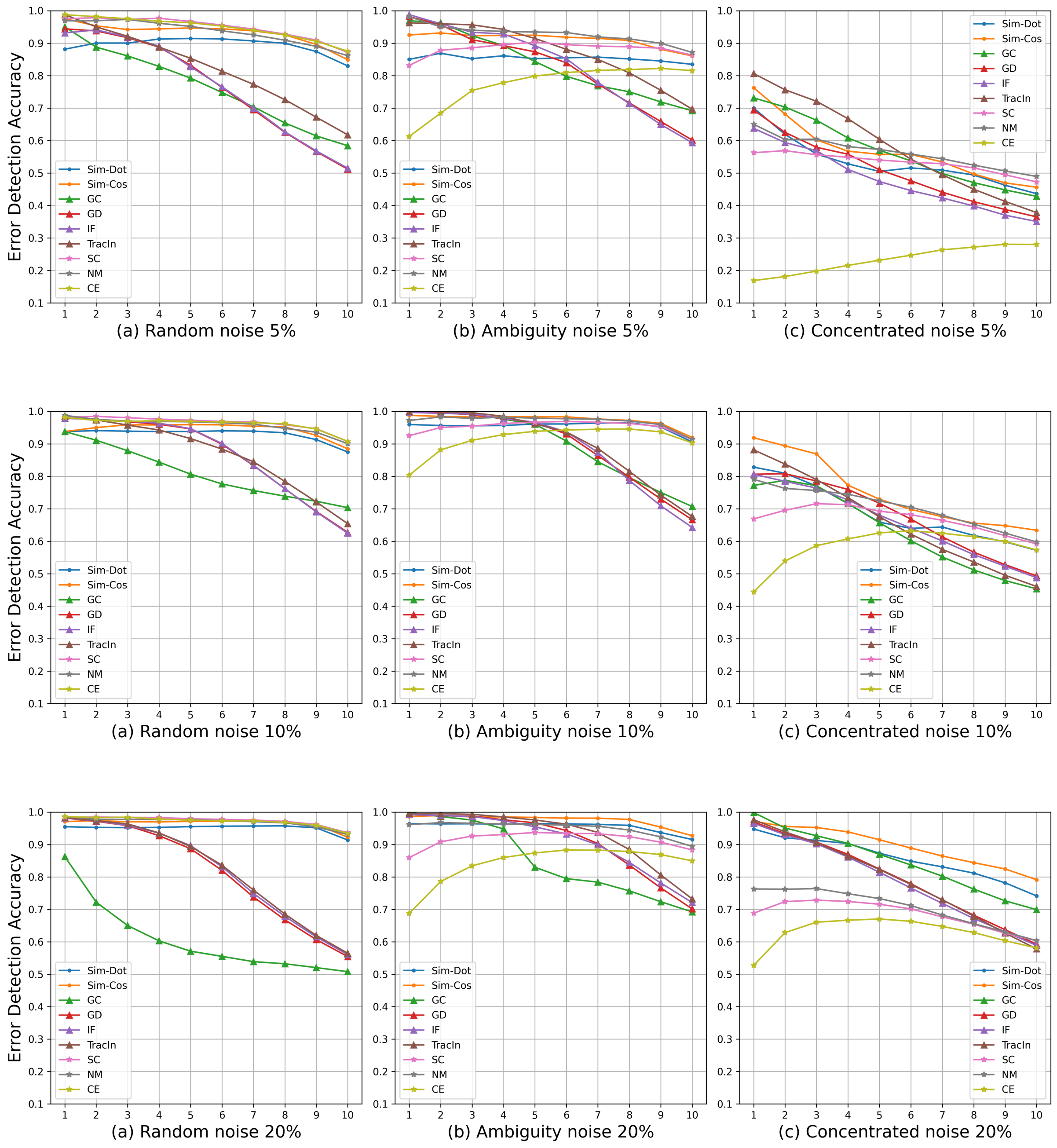
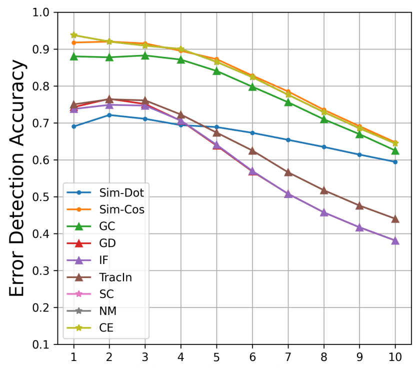
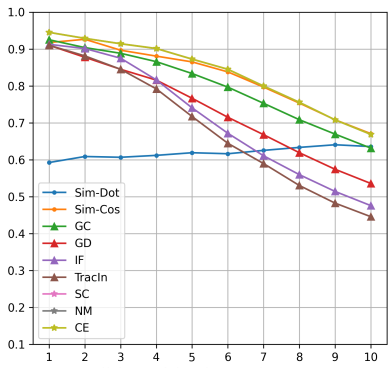


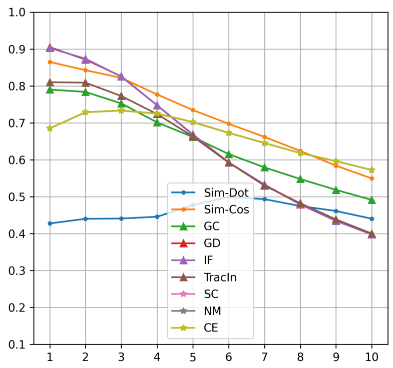
Figure 4: Detection accuracy of methods measured on Snippets with different levels of noise.



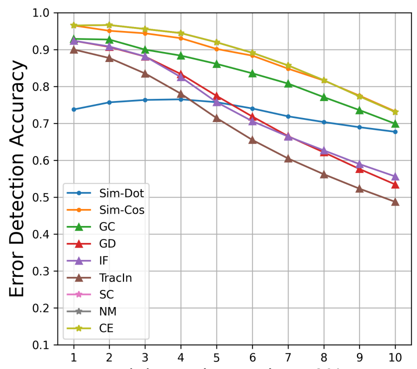
(a) Random noise 5%



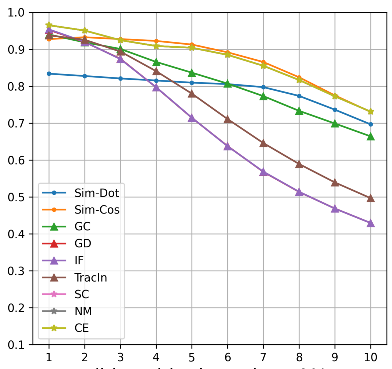
(b) Ambiguity noise 5%



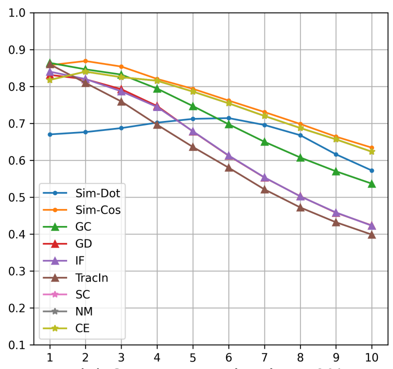
(c) Concentrated noise 5%



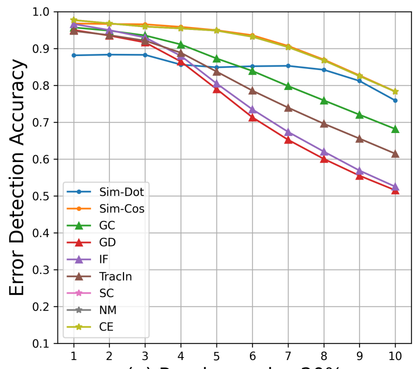
(a) Random noise 10%



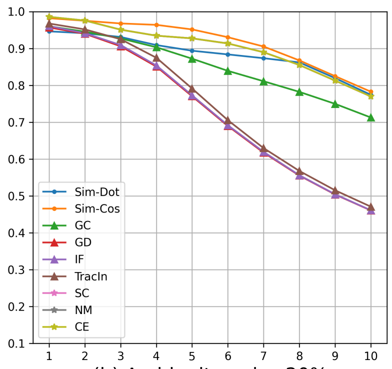
(b) Ambiguity noise 10%



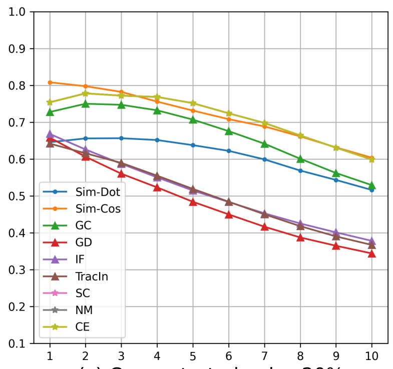
(c) Concentrated noise 10%



(a) Random noise 20%



(b) Ambiguity noise 20%



(c) Concentrated noise 20%

Figure 5: Detection accuracy of methods measured on IMDB with different levels of noise.

Table 8: Test accuracy on Snippets with 10% noise.

Method	Random noise		Ambiguity noise		Concentrated noise	
	Removed	Rectified	Removed	Rectified	Removed	Rectified
Acc. (under noise)	85.43 (+0.00)	—	88.24 (+0.00)	—	85.08 (+0.00)	—
Confidence-W. Entropy	88.07 (+2.64)	—	85.04 (-3.20)	—	84.56 (-0.52)	—
Normalize-Margin	81.75 (-3.68)	—	86.09 (-2.15)	—	79.73 (-5.35)	—
Self-Confidence	84.47 (-0.96)	—	87.06 (-1.18)	—	82.06 (-3.02)	—
Influence Function	90.04 (+4.61)	—	85.92 (-2.32)	—	81.36 (-3.72)	—
Gradient-Cosine	82.94 (-2.49)	—	81.18 (-7.06)	—	80.48 (-4.60)	—
Gradient-Dot	86.01 (+0.58)	—	85.13 (-3.11)	—	83.29 (-1.79)	—
TracIn	85.74 (+0.31)	—	86.88 (-1.36)	—	77.98 (-7.10)	—
Sim-Cos	79.73 (-5.70)	87.85 (+2.42)	85.48 (-2.76)	85.57 (-2.67)	81.58 (-3.23)	82.06 (-3.02)
Sim-Dot	88.29 (+2.86)	87.32 (+1.89)	84.34 (-3.90)	86.58 (-1.66)	81.62 (-3.46)	80.83 (-4.25)

Table 9: Test accuracy on Snippets with 5% noise.

Method	Random noise		Ambiguity noise		Concentrated noise	
	Removed	Rectified	Removed	Rectified	Removed	Rectified
Acc. (under noise)	87.14 (+0.00)	—	87.36 (+0.00)	—	87.93 (+0.00)	—
Confidence-W. Entropy	84.86 (-2.28)	—	86.57 (-0.79)	—	87.80 (-0.13)	—
Normalize-Margin	84.65 (-2.49)	—	84.82 (-2.54)	—	83.77 (-4.16)	—
Self-Confidence	85.35 (-1.79)	—	83.99 (-3.37)	—	86.09 (-1.84)	—
Influence Function	85.30 (-1.84)	—	86.00 (-1.36)	—	85.30 (-2.63)	—
Gradient-Cosine	85.43 (-1.71)	—	85.13 (-2.23)	—	79.51 (-8.42)	—
Gradient-Dot	84.56 (-2.58)	—	89.21 (+1.85)	—	87.06 (-0.87)	—
TracIn	84.65 (-2.49)	—	87.71 (+0.35)	—	81.88 (-6.05)	—
Sim-Cos	86.45 (-0.69)	85.17 (-1.97)	87.10 (-0.26)	85.48 (-1.88)	85.39 (-2.54)	87.19 (-0.74)
Sim-Dot	87.15 (+0.01)	87.06 (-0.08)	83.37 (-3.99)	84.34 (-3.02)	84.12 (-3.81)	79.82 (-8.11)

Table 10: Test accuracy on IMDB with 20% noise.

Method	Random noise		Ambiguity noise		Concentrated noise	
	Removed	Rectified	Removed	Rectified	Removed	Rectified
Acc. (under noise)	89.92	—	89.36	—	85.98	—
Confidence-W. Entropy	91.00 (+1.08)	—	90.73 (+1.37)	—	86.66 (+0.68)	—
Normalize-Margin	91.00 (+1.08)	—	90.73 (+1.37)	—	86.66 (+0.68)	—
Self-Confidence	91.00 (+1.08)	—	90.73 (+1.37)	—	86.66 (+0.68)	—
Influence Function	90.95 (+1.03)	—	90.51 (+1.15)	—	87.09 (+1.11)	—
Gradient-Cosine	90.59 (+0.67)	—	90.93 (+1.57)	—	86.78 (+0.80)	—
Gradient-Dot	90.43 (+0.51)	—	89.45 (+0.09)	—	82.87 (-3.20)	—
TracIn	90.19 (+0.27)	—	91.09 (+1.73)	—	82.87 (-3.20)	—
Sim-Cos	90.70 (+0.78)	88.73 (-1.19)	90.78 (+1.42)	90.83 (+1.47)	87.76 (+1.78)	84.73 (-1.25)
Sim-Dot	91.49 (+1.57)	90.46 (+0.54)	91.58 (+2.22)	90.13 (+0.77)	87.25 (+1.27)	87.46 (+1.48)

Table 11: Test accuracy on IMDB with 10% noise.

Method	Random noise		Ambiguity noise		Concentrated noise	
	Removed	Rectified	Removed	Rectified	Removed	Rectified
Acc. (under noise)	91.76 (+0.00)	—	91.68 (+0.00)	—	89.41 (+0.00)	—
Confidence-W. Entropy	91.79 (+0.03)	—	92.48 (+0.20)	—	90.95 (+1.54)	—
Normalize-Margin	91.79 (+0.03)	—	92.48 (+0.20)	—	90.95 (+1.54)	—
Self-Confidence	91.79 (+0.03)	—	92.48 (+0.20)	—	90.95 (+1.54)	—
Influence Function	91.65 (-0.11)	—	90.97 (-0.71)	—	86.26 (-3.15)	—
Gradient-Cosine	91.72 (-0.04)	—	91.29 (-0.39)	—	88.05 (-1.36)	—
Gradient-Dot	90.81 (-0.95)	—	92.10 (+0.42)	—	88.15 (-1.26)	—
TracIn	91.75 (-0.01)	—	91.66 (-0.02)	—	88.86 (-0.55)	—
Sim-Cos	91.12 (-0.64)	91.98 (+0.22)	92.52 (+0.84)	92.17 (+0.49)	90.15 (+0.74)	89.20 (-0.21)
Sim-Dot	91.38 (-0.38)	91.51 (-0.25)	91.52 (-0.16)	91.61 (-0.07)	89.78 (+0.37)	88.37 (-1.04)

Table 12: Test accuracy on IMDB with 5% noise.

Method	Random noise		Ambiguity noise		Concentrated noise	
	Removed	Rectified	Removed	Rectified	Removed	Rectified
Acc. (under noise)	91.67 (+0.00)	—	92.07 (+0.00)	—	91.52 (+0.00)	—
Confidence-W. Entropy	92.35 (+0.68)	—	92.58 (+0.51)	—	91.94 (+0.42)	—
Normalize-Margin	92.35 (+0.68)	—	92.58 (+0.51)	—	91.94 (+0.42)	—
Self-Confidence	92.35 (+0.68)	—	92.58 (+0.51)	—	91.94 (+0.42)	—
Influence Function	92.55 (+0.88)	—	91.76 (-0.31)	—	90.77 (-0.75)	—
Gradient-Cosine	92.46 (+0.79)	—	90.88 (-1.19)	—	89.78 (-1.74)	—
Gradient-Dot	91.79 (+0.12)	—	92.58 (+0.51)	—	90.32 (-1.20)	—
TracIn	92.62 (+0.95)	—	91.97 (-0.10)	—	91.45 (-0.07)	—
Sim-Cos	92.10 (+0.43)	92.10 (+0.43)	**93.09** (+1.02)	92.50 (+0.43)	**92.11** (+0.59)	91.74 (+0.22)
Sim-Dot	**92.69** (+1.02)	92.06 (+0.39)	91.63 (-0.44)	91.09 (-0.98)	91.59 (+0.07)	90.96 (-0.56)