Improved Influence Function for Error Detection

Thang Nguyen-Duc*, **Hoang Thanh-Tung***
FPT Software AI Center
{thangnd34, tunght18}@fsoft.com.vn

Quan Tran
Adobe Research
quan.tran@adobe.com

Anh Dau, Nghi Bui
FPT Software AI Center
{anhdtv7, nghibdq}@fsoft.com.vn

Abstract

Influence functions (IFs) are a powerful tool for estimating the influence of a data point on another data point. IFs are widely used in machine learning, especially for detecting anomalous examples. However, they are unstable when applied to deep networks. In this paper, we provide an explanation of the instability of IFs and develop a solution to this problem. We show that IFs are unreliable when the two data points belong to two different classes. Our solution leverages class information to improve the stability of IF. Extensive experiments show that our modification significantly improves the performance and stability of IFs while incurring no additional computational cost.

1 Introduction

Influence function (IF) (Koh and Liang, 2017) is an instance-based approach to understanding black-box predictors, especially deep neural networks. Different from the feature-based approach, which identifies parts of the input that are responsible for the model's prediction, instance-based approach identifies which inputs are the most influential to the model's prediction. When its assumptions are (approximately) satisfied, IF gives a reasonable estimate of the influence of a data point on another data point. This capability allows IF to detect harmful data points - data points that have a negative influence on other data points. IF is a powerful tool for detecting adversarial (Cohen et al., 2020), poisonous (Koh and Liang, 2017; Cinà et al., 2021), and erroneous (Dau et al., 2022) examples. Correcting or removing anomalous examples from the training data improves the performance and robustness of models trained on such data.

The success of IF has inspired a series of follow up works on efficient influence estimation (Charpiat et al., 2019; Khanna et al., 2019; Barshan et al., 2020; Pruthi et al., 2020). For brevity, we call IF and its variants influence functions (IFs). Basu et al. (2021) empirically observed that IFs are unstable especially when they are applied to deep networks. In this paper, we provide a deeper analysis with both empirical and theoretical explanation of the phenomenon. In Sec. 3, we show that IFs are very noisy when the two data points are in different classes, but IFs become much more stable when the two data points are in the same class.

Motivated by our analysis, we propose class-based influence functions (IFs-class), a variant of IFs that use class information to reduce the noise in influence estimation. Compared to IFs, IFs-class are much more stable and performant while having the same computational complexity (Sec. 3, 4). Our improved IFs can be used in classification problems and can be easily extended to sequential classification problems such as Named Entity Recognition (NER).

Our contributions include

1. An explanation of the instability of IFs. Our analysis sheds light on the behavior of IFs and helps the development of better IFs.

2. A stable, high performance variant of IFs that can be used to develop better anomalous data detection algorithms.

2 Background and Related work

We define the notations used in this paper. Let $\mathbf{z} = (\mathbf{x}, \mathbf{y})$ be a data point, where $\mathbf{x} \in \mathcal{X}$ is the input, $\mathbf{y} \in \mathcal{Y}$ is the target output; $\mathcal{Z} = \{\mathbf{z}^{(i)}\}_{i=1}^{n}$ be a dataset of n data points; $\mathcal{Z}_{-j} = \mathcal{Z} \backslash \mathbf{z}^{(j)}$ be the dataset $\mathcal{Z}$ with $\mathbf{z}^{(j)}$ removed; $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$ be a model with parameter $\boldsymbol{\theta}$; $\mathcal{L}_{\mathcal{Z},\boldsymbol{\theta}} = \frac{1}{n}\sum_{i=1}^{n} \ell(f_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}), \mathbf{y}^{(i)}) = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{z}^{(i)}; \boldsymbol{\theta})$ be the loss of $f_{\boldsymbol{\theta}}$ on $\mathcal{Z}$, where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ is the loss function; $\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{Z},\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\theta}}_{-i} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{Z}_{-i},\boldsymbol{\theta}}$ be the optimal parameters of the model $f_{\boldsymbol{\theta}}$ trained on $\mathcal{Z}$ and $\mathcal{Z}_{-i}$. In this

* Equal contribution

paper, $f_{\boldsymbol{\theta}}$ is a deep networks and $\hat{\boldsymbol{\theta}}$ is found by training $f_{\boldsymbol{\theta}}$ with gradient descent.

2.1 Influence function and variants

The influence of a data point $\mathbf{z}^{(i)}$ on another data point $\mathbf{z}^{(j)}$ is defined as

$$s^{(ij)} = \ell(\mathbf{z}^{(j)}; \hat{\boldsymbol{\theta}}_{-i}) - \ell(\mathbf{z}^{(j)}; \hat{\boldsymbol{\theta}}) \quad (1)$$

$s^{(ij)} < 0$ means that removing $\mathbf{z}^{(i)}$ decreases the loss at $\mathbf{z}^{(j)}$. In other words, $s^{(ij)} < 0$ means that $\mathbf{z}^{(i)}$ is harmful to $\mathbf{z}^{(j)}$. The more negative $s^{(ij)}$ is, the more harmful $\mathbf{z}^{(i)}$ is to $\mathbf{z}^{(j)}$. Naive computation of $s^{(ij)}$ requires retraining $f_{\boldsymbol{\theta}}$ on $\mathcal{Z}_{-i}$. Averaging this influence score over a dataset $\mathcal{Z}'$ gives the influence of $\mathbf{z}^{(i)}$ on $\mathcal{Z}'$

$$s^{(i)} = \frac{1}{|\mathcal{Z}'|} \sum_{\mathbf{z}^{(j)} \in \mathcal{Z}'} s^{(ij)} \quad (2)$$

$s^{(i)}$ is more stable than $s^{(ij)}$ and is commonly used in anomalous data detection (Cohen et al., 2020; Dau et al., 2022). Koh and Liang (2017) used first and second order derivatives to quickly estimate $s^{(ij)}$ without retraining the model $f_{\boldsymbol{\theta}}$ on $\mathcal{Z}_{-i}$

$$\begin{aligned} s^{(ij)} &\approx IF(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}) \\ &\approx \frac{1}{n} \nabla_{\hat{\boldsymbol{\theta}}} \ell(\mathbf{z}^{(i)}; \hat{\boldsymbol{\theta}})^\top H_{\hat{\boldsymbol{\theta}}}^{-1} \nabla_{\hat{\boldsymbol{\theta}}} \ell(\mathbf{z}^{(j)}; \hat{\boldsymbol{\theta}}) \end{aligned} \quad (3)$$

where $H_{\hat{\boldsymbol{\theta}}}$ is the Hessian at $\hat{\boldsymbol{\theta}}$. Exact computation of $H_{\hat{\boldsymbol{\theta}}}^{-1}$ is intractable for modern networks. Koh and Liang (2017) developed a fast algorithm for estimating $H_{\hat{\boldsymbol{\theta}}}^{-1} \nabla_{\hat{\boldsymbol{\theta}}} \ell(\mathbf{z}^{(j)}; \hat{\boldsymbol{\theta}})$ and used only the derivatives w.r.t. the last layer's parameters to improve the algorithm's speed. Charpiat et al. (2019) proposed gradient dot product (GD) and gradient cosine similarity (GC) as faster alternatives to IF. Pruthi et al. (2020) argued that the influence can be better approximated by accumulating it through out the training process (TracIn). The formula for IFs are summarized in Tab. 1 in Appendix A.

IFs can be viewed as measures of the similarity between the gradients of two data points. The intuition of IFs is that gradients of harmful examples are dissimilar from that of normal examples.

2.2 Influence functions for error detection

There are many kinds of noisy data such as outlier, mislabeled, or ambiguous data, that could have bad effects on the model.[1] Beyer et al. (2020) showed that a significant portion of the ImageNet (Deng et al., 2009) are mislabeled or have ambiguous labels. For brevity, we call these data points errors. Given a noisy dataset $\mathcal{Z}$ and we want to identify erroneous data points in $\mathcal{Z}$.

Cohen et al. (2020); Dau et al. (2022) used IFs to detect adversarial and erroneous examples in classification datasets. They used IFs to measure the influence of each data point $\mathbf{z} \in \mathcal{Z}$ on $\mathcal{Z}'$, a set of clean test data points. For the error detection task, data points in $\mathcal{Z}$ are ranked by how harmful they are to $\mathcal{Z}'$. Most harmful data points are re-examined by human or are removed from $\mathcal{Z}$ (Alg. 2 in Appendix A). In this paper, we focus on the error detection problem but IFs and IFs-class can be used to detect other kinds of anomalous data.

3 Improved influence function for error detection

3.1 Motivation

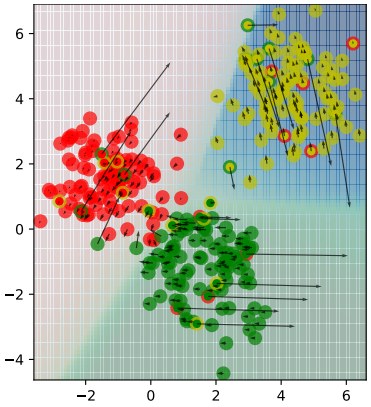


Figure 1: Gradient pattern on a classification problem. A mislabeled data point is shown by a circle with two colors, the inner color is the original (true) class, the outer color is the new (noisy) class. The gradient is computed w.r.t. the last layer's parameters. We plot only the first 2 dimensions of the gradient. See Appendix C for implementation details and other gradient dimensions.

Basu et al. (2021) attributed the instability of IFs to the non-convexity of neural networks and the errors in Taylor's expansion and Hessian-Vector product approximation. In this section, we show that the learning dynamics of neural networks makes IFs unstable and develop a solution to the problem.

Pezeshkpour et al. (2021); Hanawa et al. (2021) empirically showed that IFs with last layer gradient perform as well as or better than IFs with all layers' gradient and variants of IF behave similarly. Therefore, for simplicity, we analyze the behavior

[1] We note that there might be mislabeled/ambiguous/outlier data points that do not have bad effects on the model. In this work, we are interested in detecting and removing/correcting data points with bad effects.

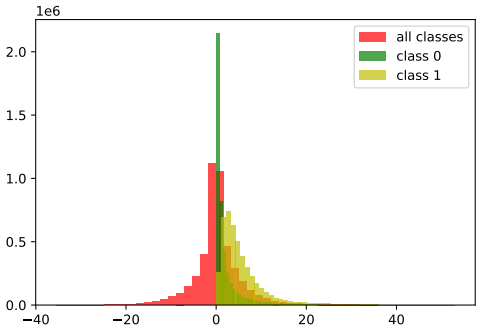


Figure 2: GD score distribution on the IMDB dataset. Results on other datasets are shown in Appendix C.

of GD with last layer's gradient and generalize our results to other IFs. Let's consider the gradient pattern of an MLP on a 3-class classification problem (Fig. 1). We observe that gradients of mislabeled examples have large magnitudes and are opposite to gradients of correct examples in the same original class. However, gradients of mislabeled data points are not necessarily opposite to that of correct data points from other classes. Furthermore, gradients of two data points from two different classes are almost perpendicular. We make the following observation. A mislabeled/correct data point often has a very negative/positive influence on data points of the true class, but its influence on other classes is noisy and small.

We verify the observation on real-world datasets (Fig. 2). We compute GD scores of pairs of clean data points from 2 different classes and plot the score's distribution. We repeat the procedure for pairs of data points from each class. In the 2-class case, GD scores are almost normally distributed with a very sharp peak at 0. That means, in many cases, a clean data point from one class has almost no significant effect on data points from the other class. And when it has a significant effect, the effect could be positive or negative with equal probability. In contrast, GD scores of pairs of data points from the same class are almost always positive. A clean data point almost certainly has a positive influence on clean data points of the same class.

We provide a theoretical explanation of this observation in Appendix D. If the two data points have different labels, then the sign of GD depends on two random variables. And as the model becomes more confident about the labels of the two data points, the magnitude of GD becomes smaller. If the two data points have the same label, then the sign of GD depends on only one random variable and the GD's magnitude does not decrease as the model becomes more confident about the labels.

Algorithm 1 Class based influence function for error detection.

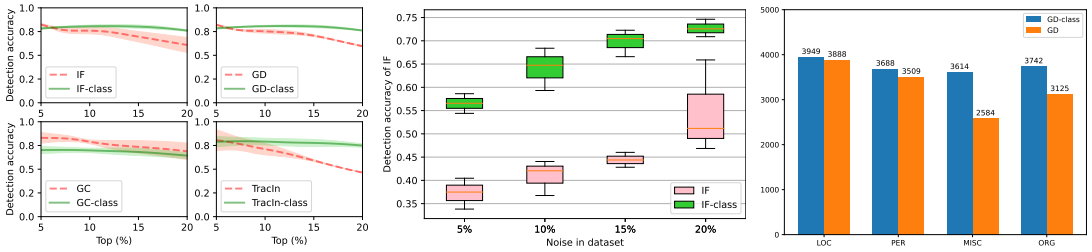
Require:
1: $\mathcal{Z} = \{\mathbf{z}^{(i)}\}_{i=1}^{n}$: a big noisy dataset
2: C: number of classes
3: $\mathcal{Z}'_k = \{\mathbf{z}'^{(j_k)}\}_{j_k=1}^{m_k}$: clean data from class k
4: $\mathcal{Z}' = \bigcup_{k=1}^{C} \mathcal{Z}'_k$: a small clean dataset
5: $f_{\hat{\boldsymbol{\theta}}}$: a deep model pretrained on $\mathcal{Z}$
6: $\text{sim}(\cdot,\cdot)$: a similarity measure in Tab. 1
Ensure: $\hat{\mathcal{Z}}$: data points in $\mathcal{Z}$ ranked by score
7: **for** $\mathbf{z}^{(i)} \in \mathcal{Z}$ **do**
8: **for** $k = 1, ..., C$ **do**
9: $s_k^{(i)} = \frac{1}{m_k} \sum_{j=1}^{m_k} \text{sim}(\nabla_{\hat{\boldsymbol{\theta}}} \ell(\mathbf{z}^{(i)}), \nabla_{\hat{\boldsymbol{\theta}}} \ell(\mathbf{z}'^{(j_k)}))$
10: **end for**
11: $s^{(i)} = \min_k(s_k^{(i)})$
12: **end for**
13: $\hat{\mathcal{Z}} = \text{sort}(\mathcal{Z}, \text{key} = \boldsymbol{s}, \text{ascending} = \text{True})$
14: **return** $\hat{\mathcal{Z}}$

3.2 Method

In the error detection problem, we have to detect examples that have negative influence on other data points. Because an erroneous data point is likely to have a very negative influence on one class and a small noisy influence on other classes, we compute the influence scores of a data point $\mathbf{z}^{(i)}$ on each class in the clean dataset $\mathcal{Z}'$ and take the minimum of these influence scores as the influence score for $\mathbf{z}^{(i)}$ (line 8-11 in Alg. 1). The resulting influence score is not affected by the noise from other classes, has lower variance, and is a stronger indicator of how harmful $\mathbf{z}^{(i)}$ is (Fig. 4 in Appendix A). In Appendix A, we show that IF-class based error detection algorithm has the same computational complexity as IF based error detection algorithm.

4 Experiments

We evaluate the error detection performance of IFs-class on 2 NLP problems, namely text classification on IMDB (Maas et al., 2011), SNLI (Bowman et al., 2015), and BigCloneBench (Svajlenko et al., 2014) datasets, and NER on the CoNLL2003 (Tjong Kim Sang and De Meulder, 2003) dataset. For the text classification problem, we detect text segments with wrong class labels. For the NER problem, we detect tokens with wrong entity types. We use BERT (Devlin et al., 2019) and CodeBERT



(a) (b) (c)

Figure 3: (a) Error detection accuracy on SNLI dataset with $p = 20\%$. (b) Error detection accuracy of IF and IF-class on IMDB dataset with different values of p. (c) Number of erroneous NER tokens detected by GD and GD-class at $p = 30\%, r = 30\%, q = 9\%$, grouped by entity types of the erroneous tokens.

(Feng et al., 2020) in our experiments. Details about models and datasets, and complete results on all datasets are located in Appendix B and C.

We inject random noise into the text classification datasets by randomly selecting $p\%$ of the data points and change their labels to other random classes. For the CoNLL-NER dataset, we randomly select $p\%$ of the sentences and change the label of $r\%$ of the phrases in the selected sentences. If a phrase belong to a class, e.g. PERSON, we change all tokens in that phrase to another class, e.g. LOCATION. To evaluate an error detection algorithm, we select top $q\%$ most harmful data points from the sorted dataset $\hat{\mathcal{Z}}$ and check how many percent of the selected data points are really erroneous. Intuitively, increasing q allows the algorithm to find more errors (increase recall) but may decrease the detection accuracy (decrease precision). We do not use Precision-Recall metric for this task because to achieve high Recall we often need to set $q = 100\%$, i.e. we have to recheck or remove the entire dataset.

Fig. 3(a) shows the error detection accuracy on the SNLI dataset and how the accuracy changes with q. Except for the GC algorithm, our class-based algorithms has higher accuracy and lower variance than the non-class-based versions. When q increases, the performance of our algorithms does not decrease as much as that of non-class-based algorithms. This confirms that the class-based IFs are less noisy than the original IFs.

Class information fails to improve the performance of GC. To understand this, let's reconsider the similarity measure $\text{sim}(\nabla_{\hat{\boldsymbol{\theta}}}\ell(\mathbf{z}^{(i)}), \nabla_{\hat{\boldsymbol{\theta}}}\ell(\mathbf{z}'^{(j)}))$. Let's assume that there exist some clean data points $\mathbf{z}'^{(j)} \in \mathcal{Z}'$ with a very large gradient $\nabla_{\hat{\boldsymbol{\theta}}}\ell(\mathbf{z}'^{(j)})$. If the similarity measure does not normalize the gradients then $\mathbf{z}'^{(j)}$ will have the dominant effect on the influence score. The noise in the influence score is mostly caused by these data points. GC normalizes the gradient and effectively removes such noise. From Fig. 1, we see that the gradients of erroneous data points tend to be larger than that of normal data points. By normalizing the gradients, GC removes the valuable information about magnitudes of gradients of erroneous data points $\nabla_{\hat{\boldsymbol{\theta}}}\ell(\mathbf{z}^{(i)})$. That could lower the error detection performance. In Fig. 3(a), we see that the performance of GC when $q \geq 15\%$ is lower than that of other class-based algorithms. Similar trends are observed on other datasets (Fig. 6, 7, 8 in Appendix C).

Fig. 3(b) shows the change in detection accuracy as the level of noise p goes from 5% to 20%. In this experiment, for each value of p, we set q to be equal to p. Our class-based influence score significantly improves the performance and reduces the variance. We note that when p increases, the error detection problem becomes easier as there are more errors. The detection accuracy, therefore, tends to increase with p as shown in Fig. 3(b), 9, 10.

Fig. 3(c) shows that GD-class outperforms GD on all entity types in CoNLL2003-NER. The performance difference between GD-class and GD is greater on the MISC and ORG categories. Intuitively, a person's name can likely be an organization's name but the reverse is less likely. Therefore, it is hard to detect that a PER/LOC tag has been changed to ORG/MISC tag. The result shows that IFs-class is more effective than IFs in detecting hard erroneous examples.

5 Conclusion

In this paper, we study influence functions and identify the source of their instability. We give a theoretical explanation of our observations. We introduce a stable variant of IFs and use that to develop a high performance error detection algorithm.

Our findings shed light of the development of new influence estimators and on the application of IFs in downstream tasks.

Limitations

Our paper has several limitations

1. Our class-based influence score cannot improve the performance of GC algorithm. Although class-based version of GD, IF, and TracIn outperformed the original GC, we aim to develop a stronger version of GC. From the analysis in Sec. 4, we believe that a partially normalized GC could have better performance. In partial GC, we normalize the gradient of the clean data point $\mathbf{z}'^{(j)}$ only. That will remove the noise introduced by $\nabla_{\hat{\boldsymbol{\theta}}}\ell(\mathbf{z}'^{(j)})$ while retaining the valuable information about the norm of $\nabla_{\hat{\boldsymbol{\theta}}}\ell(\mathbf{z}^{(i)})$.

Ethics Statement

Our paper consider a theoretical aspect of influence functions. It does not have any biases toward any groups of people. Our findings do not cause any harms to any groups of people.

References

Elnaz Barshan, Marc-Etienne Brunet, and Gintare Karolina Dziugaite. 2020. Relatif: Identifying explanatory training samples via relative influence. In *International Conference on Artificial Intelligence and Statistics*, pages 1899–1909. PMLR.

Samyadeep Basu, Phil Pope, and Soheil Feizi. 2021. Influence functions in deep learning are fragile. In *International Conference on Learning Representations*.

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

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

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

Antonio Emanuele Cinà, Kathrin Grosse, Sebastiano Vascon, Ambra Demontis, Battista Biggio, Fabio Roli, and Marcello Pelillo. 2021. Backdoor learning curves: Explaining backdoor poisoning beyond influence functions. *arXiv preprint arXiv:2106.07214*.

Gilad Cohen, Guillermo Sapiro, and Raja Giryes. 2020. Detecting adversarial samples using influence functions and nearest neighbors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14453–14462.

Anh T. V. Dau, Nghi D. Q. Bui, Thang Nguyen-Duc, and Hoang Thanh-Tung. 2022. Towards using data-influence methods to detect noisy samples in source code corpora. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

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.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. CodeBERT: A pre-trained model for programming and natural languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1536–1547, Online. Association for Computational Linguistics.

Daya Guo, Shuo Ren, Shuai Lu, Zhangyin Feng, Duyu Tang, Shujie Liu, Long Zhou, Nan Duan, Alexey Svyatkovskiy, Shengyu Fu, et al. 2020. Graphcodebert: Pre-training code representations with data flow. *arXiv preprint arXiv:2009.08366*.

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

Rajiv Khanna, Been Kim, Joydeep Ghosh, and Sanmi Koyejo. 2019. Interpreting black box predictions using fisher kernels. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3382–3390. PMLR.

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

Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Alexey Svyatkovskiy, Ambrosio Blanco, Colin Clement, Dawn Drain, Daxin Jiang, Duyu Tang, Ge Li, Lidong Zhou, Linjun Shou, Long Zhou, Michele Tufano, MING GONG, Ming Zhou, Nan Duan, Neel Sundaresan, Shao Kun Deng, Shengyu Fu, and Shujie LIU. 2021. CodeXGLUE: A machine learning benchmark dataset for code understanding and generation. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

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.

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.

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.

Jeffrey Svajlenko, Judith F. Islam, Iman Keivanloo, Chanchal K. Roy, and Mohammad Mamun Mia. 2014. Towards a big data curated benchmark of inter-project code clones. In *2014 IEEE International Conference on Software Maintenance and Evolution*, pages 476–480.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

A Additional algorithms and formula

Table 1: Influence function and its variants. We drop the constant factor $1/n$ for clarity.

IF	$\nabla_{\hat{\boldsymbol{\theta}}} \ell(\mathbf{z}^{(i)}; \hat{\boldsymbol{\theta}})^\top H_{\hat{\boldsymbol{\theta}}}^{-1} \nabla_{\hat{\boldsymbol{\theta}}} \ell(\mathbf{z}^{(j)}; \hat{\boldsymbol{\theta}})$
GD	$\left\langle \nabla_{\hat{\boldsymbol{\theta}}} \ell(\mathbf{z}^{(i)}), \nabla_{\hat{\boldsymbol{\theta}}} \ell(\mathbf{z}^{(j)}) \right\rangle$
GC	$\cos\left(\nabla_{\hat{\boldsymbol{\theta}}} \ell(\mathbf{z}^{(i)}), \nabla_{\hat{\boldsymbol{\theta}}} \ell(\mathbf{z}^{(j)})\right)$
TracIn	$\sum_{t=1}^{T} \eta_t \left\langle \nabla_{\boldsymbol{\theta}^{(t)}} \ell(\mathbf{z}^{(i)}), \nabla_{\boldsymbol{\theta}^{(t)}} \ell(\mathbf{z}^{(j)}) \right\rangle$

Algorithm 2 Influence function based error detection (Dau et al., 2022)

Require:
1: $\mathcal{Z} = \left\{\mathbf{z}^{(i)}\right\}_{i=1}^{n}$: a big noisy dataset
2: $\mathcal{Z}' = \left\{\mathbf{z}'^{(j)}\right\}_{j=1}^{m}$: a small clean dataset
3: $f_{\hat{\boldsymbol{\theta}}}$: a deep model pretrained on $\mathcal{Z}$
4: $\text{sim}(\cdot, \cdot)$: a similarity measure in Tab. 1
Ensure: $\hat{\mathcal{Z}}$: data points in $\mathcal{Z}$ ranked by score
5: **for** $\mathbf{z}^{(i)} \in \mathcal{Z}$ **do**
6: $\quad s^{(i)} = \frac{1}{m} \sum_{j=1}^{m} \text{sim}(\nabla_{\hat{\boldsymbol{\theta}}} \ell(\mathbf{z}^{(i)}), \nabla_{\hat{\boldsymbol{\theta}}} \ell(\mathbf{z}'^{(j)}))$
7: **end for**
8: $\hat{\mathcal{Z}} = \text{sort}(\mathcal{Z}, \text{key} = \boldsymbol{s}, \text{ascending} = \text{True})$
9: **return** $\hat{\mathcal{Z}}$

Computational complexity of error detection algorithms

The inner for-loop in Alg. 1 calculates C influence scores. It calls to the scoring function sim() exactly $|\mathcal{Z}'| = m$ times. The complexity of the inner for-loop in Alg. 1 is equal to that of line 6 in Alg. 2. Thus, the complexity of Alg. 1 is equal to that of Alg. 2.

B Datasets and models

In this section, we describe the datasets and models in our experiments. We used standard datasets and models and experimented with 5 different random seeds and reported the mean and standard deviation. We used a Nvidia RTX 3090 to run our experiments.

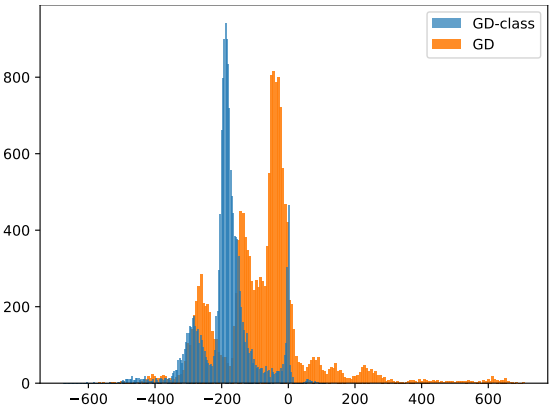


Figure 4: Distributions of GD and GD-class scores of erroneous tokens in the CoNLL2003 dataset. GD-class scores are more concentrated and have mostly negative values. GD scores are more spread out and the values are less negative. Furthermore, a significant portion of the scores are greater than 0, i.e. GD 'thinks' that these erroneous data points have positive influence on clean data points in $\mathcal{Z}'$.

B.1 Datasets

IMDB (Maas et al., 2011) The dataset includes 50000 reviews from the Internet Movie Database (IMDb) website. The task is a binary sentiment analysis task. The dataset contains an even number of positive and negative reviews. The IMDB dataset is split into training, validation, and test sets of sizes 17500, 7500, and 25000.
SNLI dataset (Standart Natural Language Inference) (Bowman et al., 2015) consists of 570k sentence pairs manually labeled as entailment, contradiction, and neutral. We convert these labels into numbers. It is geared towards serving as a benchmark for evaluating text representational systems.
BigCloneBench (Svajlenko et al., 2014) is a huge code clone benchmark that includes over 6,000,000 true clone pairs and 260,000 false clone pairs from 10 different functionality. The task is to predict whether two pieces of code have the same semantics. This dataset is commonly used in language models for code (Feng et al., 2020; Lu et al., 2021; Guo et al., 2020).
CoNLL2003 (Tjong Kim Sang and De Meulder, 2003) is one of the most influential corpora for NER model research. A large number of publications, including many landmark works, have used this corpus as a source of ground truth for NER tasks. The data consists two languages: English

and German. In this paper, we use CoNLL2003 English dataset.

B.2 Models

BERT (Devlin et al., 2019) stands for Bidirectional Encoder Representations from Transformers, is based on Transformers. The BERT model in this paper was pre-trained for natural language processing tasks. We use BERT for IMDB and SNLI datasets. At the same time, we also use the BERT model for the NER problem on the CoNLL2003 dataset.

CodeBert (Feng et al., 2020) is a bimodal pre-trained model for programming and natural languages. We use CodeBert for BigCloneBench dataset.

C Additional results

C.1 3-class classification experiment

We train a MLP with 2 input neurons, 100 hidden neurons, 3 output neurons with SGD for 1000 epochs. The activation function is LeakyReLU and the learning rate is $\eta = 1e-3$. The last layer has 6 parameters organized into a 3×2 matrix. The gradient of the loss with respect to the last layer's parameters is also organized into a 3×2 matrix. We visualize 3 rows of the gradient matrix in 3 subfigures (Fig. 5).

C.2 Result on IMDB, SNLI, BigClone, and CoNLL2003

To ensure a fair comparison between our class-based algorithm and algorithm 2, we use the same clean dataset $\mathcal{Z}'$ for both algorithms. The clean dataset $\mathcal{Z}'$ consists of C classes. We have $C = 2$ for the IMDB dataset, $C = 3$ for the SNLI dataset, and $C = 2$ for the BigCloneBench dataset. From each of the C classes, we randomly select $m_k = 50$ $k = 1, ..., C$ clean data points to form $\mathcal{Z}'$. We tried varying m_k from 10 to 1000 and observed no significant changes in performance.

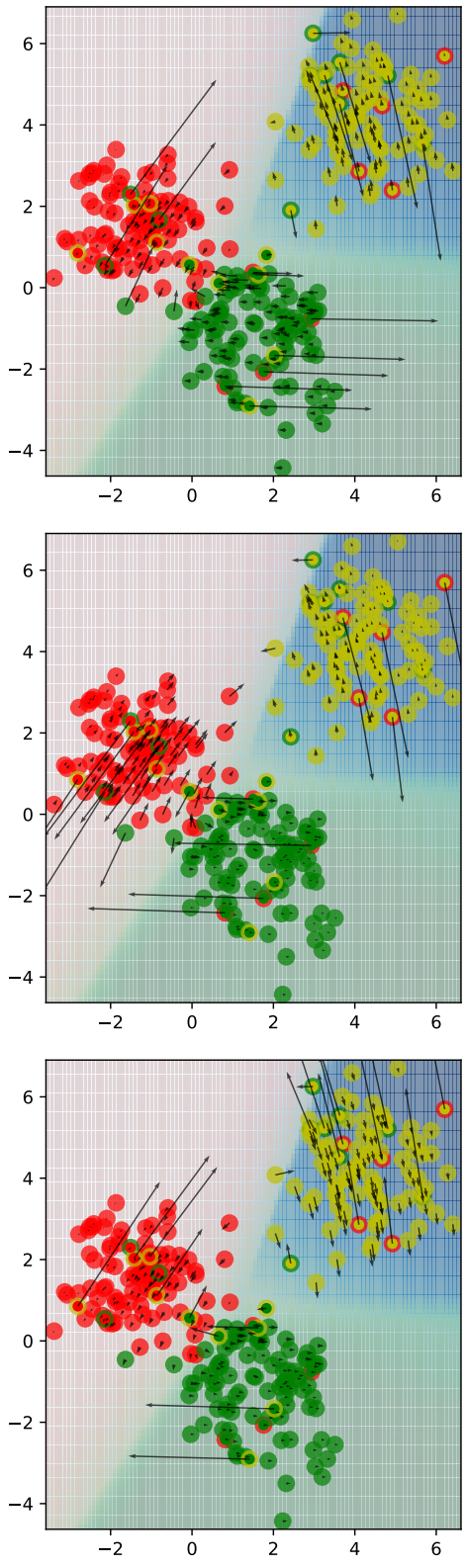


Figure 5: Gradient pattern on a classification problem. Each subfigure shows 2 dimensions of the gradient. The top subfigure shows the 1st and 2nd dimensions of the gradient. The middle subfigure shows the 3rd and 4th dimensions of the gradient. The bottom subfigure shows the 5th and 6th dimensions of the gradient.

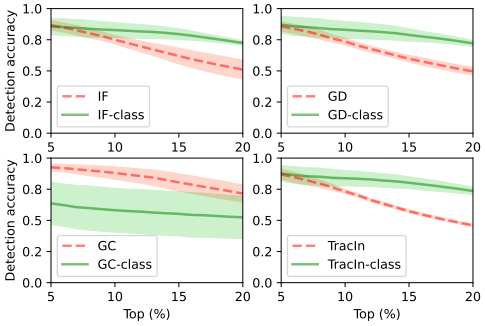


Figure 6: Error detection accuracy on IMDB dataset with $p = 20\%$.

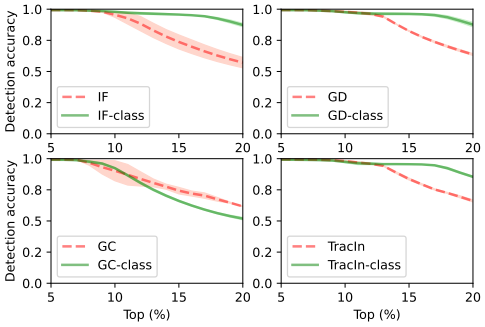


Figure 7: Error detection accuracy on BigCloneBench dataset with $p = 20\%$.

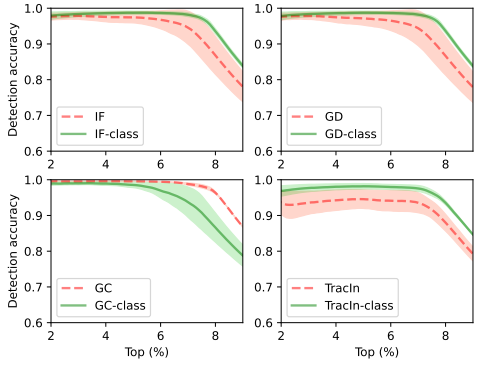


Figure 8: Error detection accuracy on CoNLL2003 dataset with $p = 30\%$ and $r = 30\%$

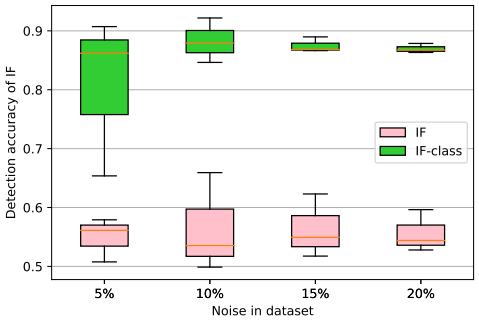


Figure 9: Change in error detection accuracy on the BigCloneBench dataset as the level of noise changes.

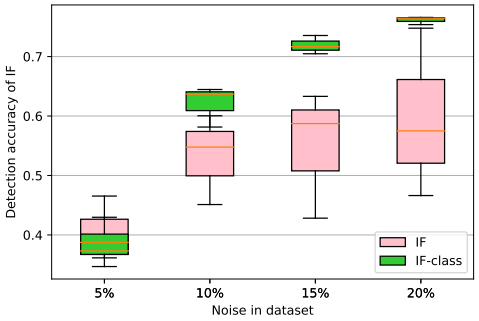


Figure 10: Change in error detection accuracy on the SNLI dataset as the level of noise changes.

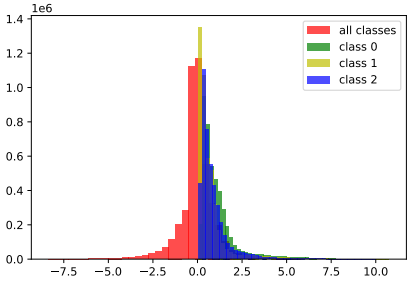


Figure 11: GD score distribution on the SNLI dataset.

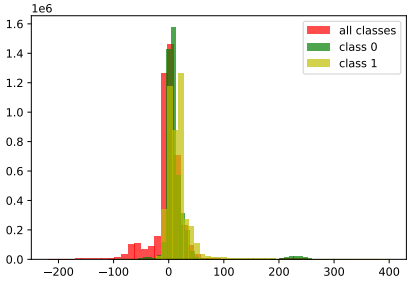


Figure 12: GD score distribution on the BigClone dataset.

D Explanation of the observation in Sec. 3

Let's consider a classification problem with cross entropy loss function

$$\ell(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{i=1}^{d_y} y_i \log \hat{y}_i$$

where d_y is the number of classes. Let $\mathbf{z} = (\mathbf{x}, \mathbf{y})$ be a data point with label k, i.e. $y_k = 1,\ y_i = 0 \ \forall\ i \neq k$. The model $f_{\boldsymbol{\theta}}$ is a deep network with last layer's parameter $W \in \mathbb{R}^{d_y \times d_h}$, where d_h is the number of hidden neurons. Let $\mathbf{u} \in \mathbb{R}^{d_h}$ be the activation of the penultimate layer. The output is computed as follow

$$\mathbf{a} = W\mathbf{u}$$
$$\hat{\mathbf{y}} = \delta(\mathbf{a})$$

where δ is the softmax output function. The derivative of the loss at $\mathbf{z}$ w.r.t. W is

$$\frac{\partial \ell(\mathbf{z})}{\partial W} = \nabla_{\mathbf{a}} \ell(\mathbf{z})\ \mathbf{u}^\top \tag{4}$$

$$= \begin{bmatrix} \nabla_{\mathbf{a}} \ell(\mathbf{z})_1 \mathbf{u}^\top \\ \vdots \\ \nabla_{\mathbf{a}} \ell(\mathbf{z})_{d_y} \mathbf{u}^\top \end{bmatrix} \tag{5}$$

The gradient $\nabla_{\mathbf{a}} \ell(\mathbf{z})$ is

$$(\nabla_{\mathbf{a}} \ell)^\top = \frac{\partial \ell}{\partial \mathbf{a}} \tag{6}$$

$$= \frac{\partial \ell}{\partial \hat{\mathbf{y}}} \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{a}} \tag{7}$$

$$= \begin{bmatrix} \frac{\partial \ell}{\partial \hat{y}_1} & \cdots & \frac{\partial \ell}{\partial \hat{y}_k} & \cdots & \frac{\partial \ell}{\partial \hat{y}_{d_y}} \end{bmatrix} \times \begin{bmatrix} \frac{\partial \hat{y}_1}{\partial a_1} & \frac{\partial \hat{y}_1}{\partial a_2} & \cdots & \frac{\partial \hat{y}_1}{\partial a_{d_h}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \hat{y}_k}{\partial a_1} & \frac{\partial \hat{y}_k}{\partial a_2} & \cdots & \frac{\partial \hat{y}_k}{\partial a_{d_h}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \hat{y}_{d_y}}{\partial a_1} & \frac{\partial \hat{y}_{d_y}}{\partial a_2} & \cdots & \frac{\partial \hat{y}_{d_y}}{\partial a_{d_h}} \end{bmatrix} \tag{8}$$

$$= \begin{bmatrix} \frac{\partial \ell}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial a_1} & \cdots & \frac{\partial \ell}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial a_k} & \cdots & \frac{\partial \ell}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial a_{d_h}} \end{bmatrix} \tag{9}$$

We go from Eqn. 8 to Eqn. 9 by using the following fact

$$\frac{\partial \ell}{\partial \hat{y}_i} = \begin{cases} 0 \text{ if } i \neq k \\ \frac{1}{\hat{y}_i} \text{ if } i = k \end{cases}$$

We also have

$$\frac{\partial \hat{y}_k}{\partial a_i} = \begin{cases} \hat{y}_k(1 - \hat{y}_k) \text{ if } i = k \\ -\hat{y}_k \hat{y}_i \text{ if } i \neq k \end{cases}$$

Substitute this into Eqn. 9 we have

$$\nabla_{\mathbf{a}} \ell = \begin{bmatrix} -\hat{y}_1 \\ \vdots \\ 1 - \hat{y}_k \\ \vdots \\ -\hat{y}_{d_y} \end{bmatrix}$$

Because $1 - \hat{y}_k = \sum_{j \neq k} \hat{y}_j$, $1 - \hat{y}_k$ is much greater than $\hat{y}_j$ in general. Substitute this into Eqn. 5, we see that the magnitude of the k-th row is much larger than than of other rows. We also note that the update for the k-th row of W has the opposite direction of the updates for other rows.

Let's consider the inner product of the gradients of two data points $\mathbf{z}$ and $\mathbf{z}'$ with label k and k'. Let's consider the case where $k' \neq k$ first.

$$\text{vec}\left(\frac{\partial \ell(\mathbf{z})}{\partial W}\right)^\top \text{vec}\left(\frac{\partial \ell(\mathbf{z}')}{\partial W}\right) = (\nabla_{\mathbf{a}} \ell^\top \nabla_{\mathbf{a}'} \ell)(\mathbf{u}^\top \mathbf{u}') \tag{10}$$

Intuitively, the product $\nabla_{\mathbf{a}} \ell^\top \nabla_{\mathbf{a}'} \ell$ is small because the large element $\nabla_{\mathbf{a}} \ell_k = 1 - \hat{y}_k$ is multiplied to the small element $\nabla_{\mathbf{a}'} \ell_k = \hat{y}'_k$ and the large element $\nabla_{\mathbf{a}'} \ell_{k'} = 1 - \hat{y}'_{k'}$ is multiplied to the small element $\nabla_{\mathbf{a}} \ell_{k'} = \hat{y}_{k'}$. To make it more concrete, let's assume that $\hat{y}_k = \alpha \approx 1$ and $\hat{y}_i = \frac{1-\alpha}{d_y - 1} = \beta$ for $i \neq k$. We assume the same condition for $\mathbf{y}'$.

$$\begin{aligned} \nabla_{\mathbf{a}} \ell^\top \nabla_{\mathbf{a}'} \ell &= (\hat{y}_k - 1)\hat{y}'_k + (\hat{y}'_{k'} - 1)\hat{y}_{k'} + \sum_{i=1, i \neq k, k'}^{d_y} \hat{y}_i \hat{y}'_i \\ &= (d_y - 2)\beta^2 - 2(d_y - 1)\beta^2 \\ &= -d_y \beta^2 \\ &= -\frac{d_y (1 - \alpha)^2}{(d_y - 1)^2} \end{aligned} \tag{11}$$

$\alpha \approx 1$ implies $1 - \alpha \approx 0$ and $\beta \approx 0$. Eqn. 11 implies that as the model is more confident about the label of $\mathbf{z}$ and $\mathbf{z}'$, the product $\nabla_{\mathbf{a}} \ell^\top \nabla_{\mathbf{a}'} \ell$ tends toward 0 with a quadratic rate.

The sign of the gradient product depends on the sign of $\nabla_{\mathbf{a}} \ell^\top \nabla_{\mathbf{a}'} \ell$ and $\mathbf{u}^\top \mathbf{u}'$. The signs of $\nabla_{\mathbf{a}} \ell^\top \nabla_{\mathbf{a}'} \ell$ and $\mathbf{u}^\top \mathbf{u}'$ are random variables that depend on the noise in the features $\mathbf{u}$ and $\mathbf{u}'$ and the

weight matrix W. If the model $f_{\boldsymbol{\theta}}$ cannot learn a good representation of the input then the feature $\mathbf{u}$ and the sign of $\mathbf{u}^\top \mathbf{u}'$ could be very noisy. $\text{sign}(\mathbf{u}^\top \mathbf{u}')$ is even noisier if $\mathbf{z}$ and $\mathbf{z}'$ are from different classes.

We now consider the case where $k' = k$. When $k' = k$, $\nabla_{\mathbf{a}} \ell^\top \nabla_{\mathbf{a}'} \ell$ is always positive. The sign of the gradient product only depends on $\mathbf{u}^\top \mathbf{u}'$. That explains why the product of gradients of data points from the same class is much less noisy.

Furthermore, the magnitude of $\nabla_{\mathbf{a}} \ell^\top \nabla_{\mathbf{a}'} \ell$ is larger than that in the case $k' \neq k$ because the large element $1 - \hat{y}_k$ is multiplied to the large element $1 - \hat{y}'_k$. More concretely, under the same assumption as in the case $k' \neq k$, we have

$$\begin{aligned} \nabla_{\mathbf{a}} \ell^\top \nabla_{\mathbf{a}'} \ell &= (1 - \hat{y}_k)(1 - \hat{y}'_k) + \sum_{i=1, i \neq k}^{d_y} \hat{y}_i \hat{y}'_i \\ &= (1 - \alpha)^2 + (d_y - 1)\beta^2 \end{aligned} \tag{12}$$

From Eqn. 12, we see that when $k' = k$, the magnitude of $\nabla_{\mathbf{a}} \ell^\top \nabla_{\mathbf{a}'} \ell$ is approximately d_y times larger than that when $k' \neq k$.