

# Class based Influence Functions for Error Detection

Nguyen Duc-Thang\*<sup>†</sup> Hoang Thanh-Tung\*<sup>†</sup> Quan Tran\*<sup>‡</sup>  
Huu-Tien Dang<sup>†</sup> Nguyen Ngoc-Hieu<sup>†</sup> Anh Dau<sup>†</sup> Nghi Bui<sup>†</sup>

<sup>†</sup> FPT Software AI Center   <sup>‡</sup> Adobe Research  
{nguyenducthang8a2, htt210, quanthdhcn}@gmail.com

## Abstract

Influence functions (IFs) are a powerful tool for detecting anomalous examples in large scale datasets. However, they are unstable when applied to deep networks. In this paper, we provide an explanation for 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 IFs. Extensive experiments show that our modification significantly improves the performance and stability of IFs while incurring no additional computational cost.

## 1 Introduction

Deep learning models are data hungry. Large models such as transformers (Vaswani et al., 2017), BERT (Devlin et al., 2019), and GPT-3 (Brown et al., 2020) require millions to billions of training data points. However, data labeling is an expensive, time consuming, and error prone process. Popular datasets such as the ImageNet (Deng et al., 2009) contain a significant amount of errors - data points with incorrect or ambiguous labels (Beyer et al., 2020). The need for automatic error detection tools is increasing as the sizes of modern datasets grow.

Influence function (IF) (Koh and Liang, 2017) and its variants (Charpiat et al., 2019; Khanna et al., 2019; Barshan et al., 2020; Pruthi et al., 2020) are a powerful tool for estimating the influence of a data point on another data point. Researchers leveraged this capability of IFs to design or detect adversarial (Cohen et al., 2020), poisonous (Koh et al., 2022; Koh and Liang, 2017), and erroneous (Dau et al., 2022) examples in large scale datasets. The intuition is that these harmful data points usually have a negative influence on other data points and this influence can be estimated with IFs.

Basu et al. (2021) empirically observed that IFs are unstable when they are applied to deep neu-

ral networks (DNNs). The quality of influence estimation deteriorates as networks become more complex. In this paper, we provide empirical and theoretical explanations for the instability of IFs. We show that IFs scores are very noisy when the two data points belong to two different classes but IFs scores are much more stable when the two data points are in the same class (Sec. 3). Based on that finding, we propose IFs-class, variants of IFs that use class information to improve the stability while introducing no additional computational cost. IFs-class can replace IFs in anomalous data detection algorithms. In Sec. 4, we compare IFs-class and IFs on the error detection problem. Experiments on various NLP tasks and datasets confirm the advantages of IFs-class over IFs.

## 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;  $\mathcal{Z}_{-i} = \mathcal{Z} \setminus \mathbf{z}^{(i)}$  be the dataset  $\mathcal{Z}$  with  $\mathbf{z}^{(i)}$  removed;  $f_{\theta} : \mathcal{X} \rightarrow \mathcal{Y}$  be a model with parameter  $\theta$ ;  $\mathcal{L}_{\mathcal{Z}, \theta} = \frac{1}{n} \sum_{i=1}^n \ell(f_{\theta}(\mathbf{x}^{(i)}), \mathbf{y}^{(i)}) = \frac{1}{n} \sum_{i=1}^n \ell(\mathbf{z}^{(i)}; \theta)$  be the empirical risk of  $f_{\theta}$  measured on  $\mathcal{Z}$ , where  $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+$  is the loss function;  $\hat{\theta} = \arg \min_{\theta} \mathcal{L}_{\mathcal{Z}, \theta}$  and  $\hat{\theta}_{-i} = \arg \min_{\theta} \mathcal{L}_{\mathcal{Z}_{-i}, \theta}$  be the optimal parameters of the model  $f_{\theta}$  trained on  $\mathcal{Z}$  and  $\mathcal{Z}_{-i}$ . In this paper,  $f_{\theta}$  is a deep network and  $\hat{\theta}$  is found by training  $f_{\theta}$  with gradient descent on the training set  $\mathcal{Z}$ .

### 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 the change in loss at  $\mathbf{z}^{(j)}$  when  $\mathbf{z}^{(i)}$  is removed from the training set

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

The absolute value of  $s^{(ij)}$  measures the strength of the influence of  $\mathbf{z}^{(i)}$  on  $\mathbf{z}^{(j)}$ . The sign of  $s^{(ij)}$  show

\*Joint first authors

the direction of influence. A negative  $s^{(ij)}$  means that removing  $\mathbf{z}^{(i)}$  decreases the loss at  $\mathbf{z}^{(j)}$ , i.e.  $\mathbf{z}^{(i)}$  is harmful to  $\mathbf{z}^{(j)}$ .  $s^{(ij)}$  has high variance because it depends on a single (arbitrary) data point  $\mathbf{z}^{(j)}$ . To better estimate the influence of  $\mathbf{z}^{(i)}$  on the entire data distribution, researchers average the influence scores of  $\mathbf{z}^{(i)}$  over a reference set  $\mathcal{Z}'$

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

$s^{(i)}$  is the influence of  $\mathbf{z}^{(i)}$  on the reference set  $\mathcal{Z}'$ .  $\mathcal{Z}'$  can be a random subset of the training set or a held-out dataset. Naive computation of  $s^{(ij)}$  requires retraining  $f_{\theta}$  on  $\mathcal{Z}_{-i}$ . Koh and Liang (2017) proposed the influence function (IF) to quickly estimate  $s^{(ij)}$  without retraining

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

where  $H_{\hat{\theta}} = \partial^2 \mathcal{L}_{\mathcal{Z}, \hat{\theta}} / \partial \theta^2$  is the Hessian at  $\hat{\theta}$ . Exact computation of  $H_{\hat{\theta}}^{-1}$  is intractable for modern networks. Koh and Liang (2017) developed a fast algorithm for estimating  $H_{\hat{\theta}}^{-1} \nabla_{\hat{\theta}} \ell(\mathbf{z}^{(j)}; \hat{\theta})$  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. 3 in Appx. A.

IFs can be viewed as measures of the similarity between the gradients of two data points. Intuitively, gradients of harmful examples are dissimilar from that of normal examples (Fig. 1).

## 2.2 Influence functions for error detection

In the error detection problem, we have to detect data points with wrong labels. Given a (potentially noisy) dataset  $\mathcal{Z}$ , we have to rank data points in  $\mathcal{Z}$  by how likely they are erroneous. Removing or correcting errors improves the performance and robustness of models trained on that dataset.

Traditional error detection algorithms that use hand designed rules (Chu et al., 2013) or simple statistics (Huang and He, 2018), do not scale well to deep learning datasets. Cohen et al. (2020); Dau et al. (2022) used IFs to detect adversarial and erroneous examples in deep learning datasets. Dau

et al. (2022) used IFs to measure the influence of each data point  $\mathbf{z} \in \mathcal{Z}$  on a clean reference set  $\mathcal{Z}'$ . 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 Appx. 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 Method

### 3.1 Motivation

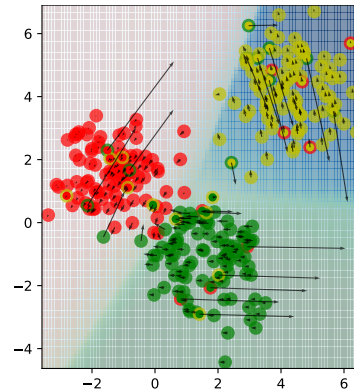


Figure 1: Gradient pattern on a classification problem. A mislabeled data point is shown by a circle with two colors, the inner color is the original (true) class, the outer color is the new (noisy) class. We plot only the first 2 dimensions of the gradient. See Appx. C for implementation details and other gradient dimensions.

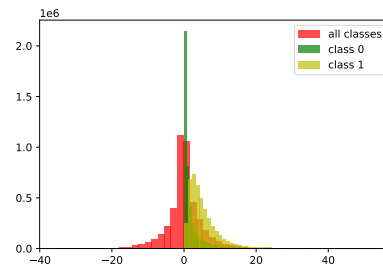


Figure 2: GD score distribution on the IMDB dataset. Results on other datasets are shown in Appx. C.

Basu et al. (2021) attributed the instability of IFs to the non-convexity of DNNs and the errors in Taylor’s expansion and Hessian-Vector product approximation. In this section, we show that the learning dynamics of DNNs makes examples from different classes unrelated and can have random influence on each other.

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, we analyze the behavior of GD with last layer’s gradient and generalize our results to other IFs. Fig. 1 shows the last layer’s gradient of an MLP on a 3-class classification problem. In the figure, gradients of mislabeled data points have large magnitudes and are opposite to gradients of correct data points in the true 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 same (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 no significant influence 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.

Our theoretical analysis shows that when the two data points have different labels, then the sign of GD depends on two random variables, the sign of inner product of the features and the sign of inner product of gradients of the losses w.r.t. the logits. And as the model becomes more confident about the labels of the two data points, the magnitude of GD becomes smaller very quickly. Small perturbations to the logits or the features can flip the sign of GD. In contrast, if the two data points have the same label, then the sign of GD depends on only one random variable, the sign of the inner product of the feature, and the GD’s magnitude remains large when the model becomes more confident. Mathematical details are deferred to Appx. D.

### 3.2 Class based IFs for error detection

Our class based IFs for error detection is shown in Alg. 1. In Sec. 3.1, we see that an error has a very

---

**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 clean reference dataset
- 5:  $f_{\hat{\theta}}$ : a deep model pretrained on  $\mathcal{Z}$
- 6:  $\text{sim}(\cdot, \cdot)$ : a similarity measure in Tab. 3

**Ensure:**  $\hat{\mathcal{Z}}$ : data points in  $\mathcal{Z}$  ranked by score

- 7: **for**  $\mathbf{z}^{(i)} \in \mathcal{Z}$  **do**
  - 8:     **for**  $k = 1, \dots, C$  **do**
  - 9:          $s_k^{(i)} = \frac{\sum_{j=1}^{m_k} \text{sim}(\nabla_{\hat{\theta}} \ell(\mathbf{z}^{(i)}), \nabla_{\hat{\theta}} \ell(\mathbf{z}'^{(j_k)}))}{m_k}$
  - 10:     **end for**
  - 11:      $s^{(i)} = \min_k(s_k^{(i)})$
  - 12: **end for**
  - 13:  $\hat{\mathcal{Z}} = \text{sort}(\mathcal{Z}, \text{key} = s, \text{ascending} = \text{True})$
  - 14: **return**  $\hat{\mathcal{Z}}$
- 

strong negative influence on correct data points in the true class, and a correct data point has a positive influence on correct data points in the true class. Influence score on the true class is a stronger indicator of the harmfulness of a data point and is better at differentiating erroneous and correct data points. Because we do not know the true class of  $\mathbf{z}^{(i)}$  in advance, we compute its influence score on each class in the reference set  $\mathcal{Z}'$  and take the minimum of these influence scores as the indicator of the harmfulness of  $\mathbf{z}^{(i)}$  (line 8-11). Unlike the original IFs, IFs-class are not affected by the noise from other classes and thus, have lower variances (Fig. 4 in Appx. A). In Appx. A, we show that our algorithm has the same computational complexity as IFs based error detection algorithm.

## 4 Experiments

### 4.1 Error detection on benchmark datasets

**Experiment setup** We evaluate the error detection performance of IFs-class on 2 NLP tasks, (1) text classification on IMDB (Maas et al., 2011), SNLI (Bowman et al., 2015), and BigCloneBench (Svajlenko et al., 2014) datasets, and (2) NER on the CoNLL2003 (Tjong Kim Sang and De Meulder, 2003) dataset. For text classification tasks, we detect text segments with wrong labels. For the NER task, we detect tokens with wrong entity types. We use BERT (Devlin et al., 2019) and CodeBERT (Feng et al., 2020) in our experiments. Implementation details are located in Appx. B. To

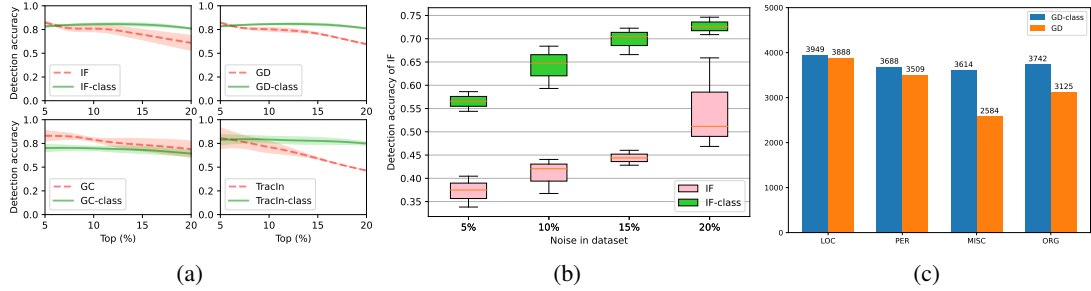


Figure 3: (a) Error detection accuracy on SNLI dataset with  $p = 20\%$ . (b) Error detection accuracy of IF and IF-class on IMDB dataset with different values of  $p$ . (c) Number of erroneous NER tokens detected by GD and GD-class at  $p = 30\%$ ,  $r = 30\%$ ,  $q = 9\%$ , grouped by entity types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

create benchmark datasets  $\mathcal{Z}'$ s, we inject random noise into the above datasets. For text classification datasets, we randomly select  $p\%$  of the data points and randomly change their labels to other classes. For the CoNLL-NER dataset, we randomly select  $p\%$  of the sentences and change the labels of  $r\%$  of the phrases in the selected sentences. All tokens in a selected phrase are changed to the same class. The reference set  $\mathcal{Z}'$  is created by randomly selecting  $m_k$  clean data points from each class in  $\mathcal{Z}$ . To ensure a fair comparison, we use the same reference set  $\mathcal{Z}'$  for both IFs and IFs-class algorithms. Models are trained on the noisy dataset  $\mathcal{Z}$ . 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).

Our code is available at <https://github.com/Fsoft-AIC/Class-Based-Influence-Functions>.

**Result and Analysis** Because results on all datasets share the same patterns, we report representative results here and defer the full results to Appx. C.

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 have higher accuracy and lower variance than the non-class-based versions. When  $q$  increases, the performance of IFs-class does not decrease as much as that of IFs. This confirms that IFs-class are less noisy than IFs. Class information fails to improve the performance of GC. To understand this, let's reconsider the similarity measure  $\text{sim}(\cdot, \cdot)$ . Let's assume that there exist some clean data points  $\mathbf{z}'^{(j)} \in \mathcal{Z}'$  with a very large gradient

$\nabla_{\hat{\theta}} \ell(\mathbf{z}'^{(j)})$ . If the similarity measure does not normalize the norm of  $\nabla_{\hat{\theta}} \ell(\mathbf{z}'^{(j)})$ , 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 both gradients,  $\nabla_{\hat{\theta}} \ell(\mathbf{z}^{(i)})$  and  $\nabla_{\hat{\theta}} \ell(\mathbf{z}'^{(j)})$ , and effectively removes such noise. However, gradients of errors tend to be larger than that of normal data points (Fig. 1). By normalizing both gradients, GC removes the valuable information about magnitudes of gradients of errors  $\nabla_{\hat{\theta}} \ell(\mathbf{z}^{(i)})$ . That lowers the 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 Appx. C).

Fig. 3(b) shows the change in detection accuracy as the level of noise  $p$  goes from 5% to 20%. 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 harder to detect that a PER or LOC tag has been changed to ORG or MISC tag than the reverse. The result shows that IFs-class is more effective than IFs in detecting hard erroneous examples.

## 4.2 The effect of data on error detection algorithms

We study the effect of the size and the cleanliness of the reference set on the performance of error detection algorithms.

**The size of the reference set.** We changed the size of classes in the reference set from 10 to 1000 to study the effect of the reference set’s size on the detection performance. We report the mean performance of GD and GC algorithms in Tab. 1. We observe no clear trend in the performance as the size of the reference set increases. Our conjecture is that gradients of clean data points from the same class have almost the same direction. Averaging the gradient direction over a small set of data points already gives a very stable gradient direction. Therefore, increasing the size of the reference set does not have much impact on detection performance.

Table 1: The effect of the reference set’s size.

Method	top 5%	top 10%	top 15%	top 20%
GD@10	79.18	76.95	74.90	71.58
GD@50	82.10	72.35	72.82	64.92
GD@100	92.99	88.01	82.53	77.60
GD@200	83.05	83.55	81.04	77.10
GD@500	85.36	78.08	73.22	63.96
GD@1000	83.47	82.43	82.45	77.56
GC@10	83.48	87.52	85.06	78.15
GC@50	80.63	86.73	84.77	77.96
GC@100	81.90	84.45	84.42	77.98
GC@200	79.88	83.62	84.22	77.88
GC@500	82.48	83.63	84.25	77.95
GC@1000	85.09	82.30	83.81	77.85

**The cleanliness of the reference set.** The result of GD and GD-class on SNLI dataset when the reference set is a random (noisy) subset of the training set is shown in table 2. When the reference set is noisy, the error detection performance of IF algorithms decreases significantly. IF-class algorithms are much more robust to noise in the reference set and their performance decreases only slightly. This experiment further demonstrates the advantage of IFs-class over IFs algorithms.

## 5 Conclusion

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

Table 2: The result of GD and GD-class on SNLI dataset when the reference set is a random (noisy) subset of the training set.

Method	top 5%	top 10%	top 15%	top 20%
GD	75.33	59.37	43.87	34.49
GD-Class	73.85	70.84	67.28	64.29

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



## Limitations

Our paper has the following 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  $\|\nabla_{\hat{\theta}}\ell(\mathbf{z}^{(j)})\|$  while retaining the valuable information about the norm of  $\nabla_{\hat{\theta}}\ell(\mathbf{z}^{(i)})$ .

## 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.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.
- Guillaume Charpiat, Nicolas Girard, Loris Felardos, and Yuliya Tarabalka. 2019. 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.
- 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*.
- 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.
- 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.
- Pang Wei Koh, Jacob Steinhardt, and Percy Liang. 2022. Stronger data poisoning attacks break data sanitization defenses. *Machine Learning*, 111(1):1–47.
- Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.
- 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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

## A Additional algorithms and formula

Table 3: Influence function and its variants. We drop the constant factor  $1/n$  for clarity.

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

### Computational complexity of error detection algorithms

The inner for-loop in Alg. 1 calculates  $C$  influence scores. It calls to the scoring function  $\text{sim}()$  exactly

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

**Require:**

- 1:  $\mathcal{Z} = \{\mathbf{z}^{(i)}\}_{i=1}^n$ : a big noisy dataset
- 2:  $\mathcal{Z}' = \{\mathbf{z}'^{(j)}\}_{j=1}^m$ : a clean reference dataset
- 3:  $f_{\hat{\theta}}$ : a deep model pretrained on  $\mathcal{Z}$
- 4:  $\text{sim}(\cdot, \cdot)$ : a similarity measure in Tab. 3

**Ensure:**  $\hat{\mathcal{Z}}$ : data points in  $\mathcal{Z}$  ranked by score

- 5: **for**  $\mathbf{z}^{(i)} \in \mathcal{Z}$  **do**
- 6:      $s^{(i)} = \frac{1}{m} \sum_{j=1}^m \text{sim}(\nabla_{\hat{\theta}} \ell(\mathbf{z}^{(i)}), \nabla_{\hat{\theta}} \ell(\mathbf{z}'^{(j)}))$
- 7: **end for**
- 8:  $\hat{\mathcal{Z}} = \text{sort}(\mathcal{Z}, \text{key} = s, \text{ascending} = \text{True})$
- 9: **return**  $\hat{\mathcal{Z}}$

$|\mathcal{Z}'| = m$  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.

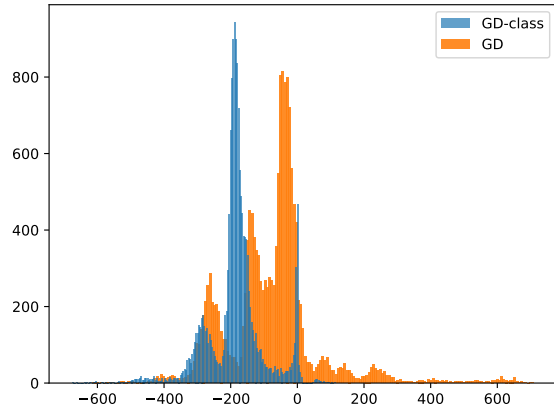


Figure 4: Distributions of GD and GD-class scores of erroneous tokens in the CoNLL2003 dataset. GD-class scores are more concentrated and have mostly negative values. GD scores are more spread out and the values are less negative. Furthermore, a significant portion of GD scores are greater than 0, i.e.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

## B Implementation details

### B.1 Experiment setup

We used standard datasets and models and experimented with 5 different random seeds and reported the mean and standard deviation. A Nvidia RTX 3090 was used to run our experiments. Models are trained with the AdamW optimizer (Loshchilov and Hutter, 2019) with learning rate  $\eta = 5e - 5$ , cross entropy loss function, and batch-size of 16.



The epoch with the best classification accuracy on the validation set was used for error detection.

Our source code and guidelines were attached to the supplementary materials.

## B.2 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. The IMDB dataset can be found at <https://ai.stanford.edu/~amaas/data/sentiment/>

**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. This dataset is available at <https://nlp.stanford.edu/projects/snli/>.

**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). This dataset is available at <https://github.com/clonebench/BigCloneBench>

**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. The sizes of training, validation, and test are 14,987, 3,466, and 3,684 sentences correspond to 203,621, 51,362, and 46,435 tokens, respectively. The dataset is available at <https://www.clips.uantwerpen.be/conll2003/ner/>

## B.3 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 in the first hidden layer, 2 hidden neurons in the second hidden layer, and 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  $3 \times 2$  matrix. The gradient of the loss with respect to the last layer’s parameters is also organized into a  $3 \times 2$  matrix. We visualize 3 rows of the gradient matrix in 3 subfigures (Fig. 5).

### C.2 Result on IMDB, SNLI, BigCloneBench, and CoNLL2003

To ensure a fair comparison between our class-based algorithm and algorithm 2, we use the same reference dataset  $\mathcal{Z}'$  for both algorithms. The reference dataset  $\mathcal{Z}'$  consists of  $C$  classes. We have  $C = 2$  for the IMDB dataset,  $C = 3$  for the SNLI dataset,  $C = 2$  for the BigCloneBench dataset, and  $C = 5$  for the CoNLL2003-NER dataset. From each of the  $C$  classes, we randomly select  $m_k = 50$   $k = 1, \dots, C$  clean data points to form  $\mathcal{Z}'$ . We tried varying  $m_k$  from 10 to 1000 and observed no significant changes in performance.

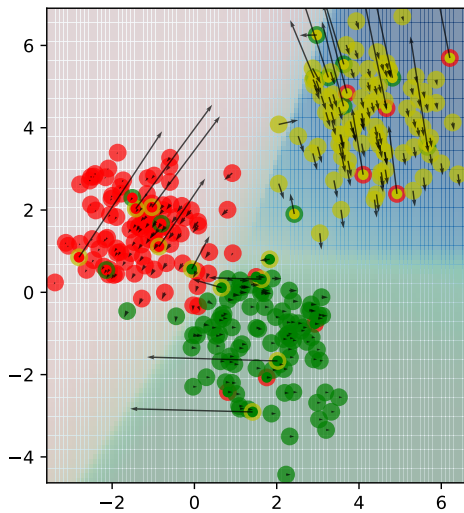
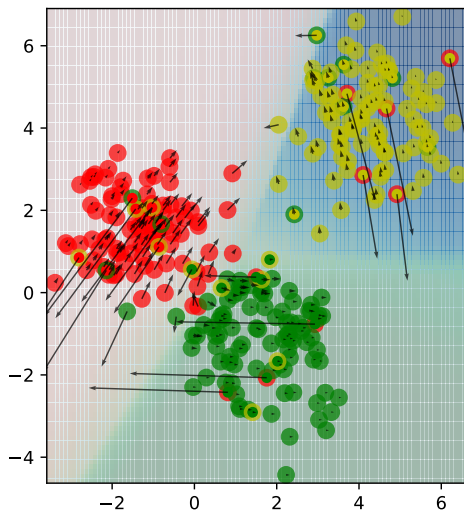
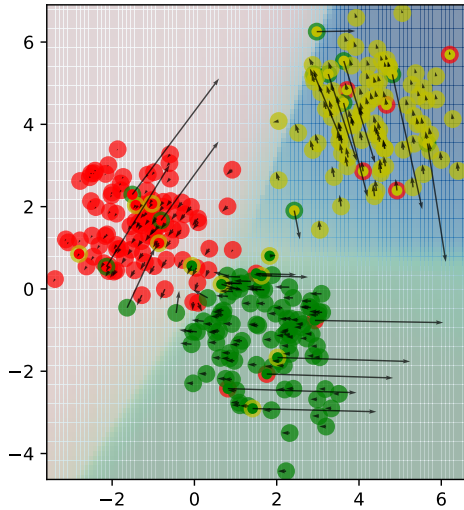


Figure 5: Gradient pattern on a classification problem. Each subfigure shows 2 dimensions of the gradient. The top subfigure shows the 1st and 2nd dimensions of the gradient. The middle subfigure shows the 3rd and 4th dimensions of the gradient. The bottom subfigure shows the 5th and 6th dimensions of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

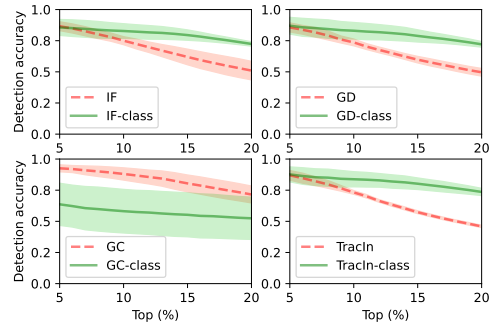


Figure 6: Error detection accuracy on IMDB dataset with  $p = 20\%$ .

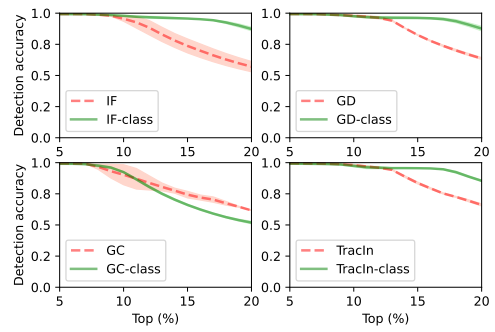


Figure 7: Error detection accuracy on BigCloneBench dataset with  $p = 20\%$ .

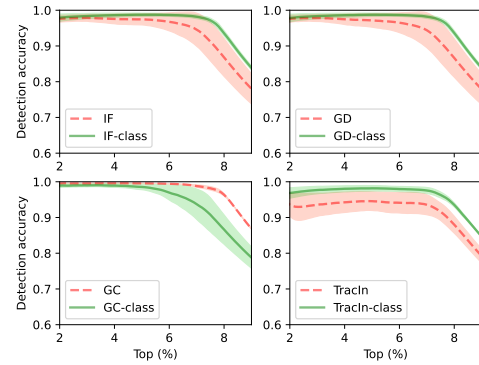


Figure 8: Error detection accuracy on CoNLL2003 dataset with  $p = 30\%$  and  $r = 30\%$

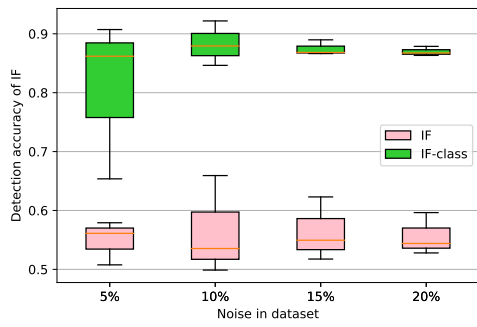


Figure 9: Change in error detection accuracy on the BigCloneBench dataset as the level of noise changes.

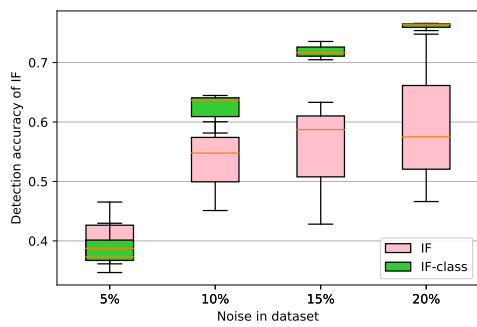


Figure 10: Change in error detection accuracy on the SNLI dataset as the level of noise changes.

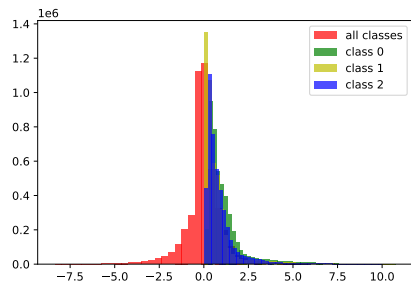


Figure 11: GD score distribution on the SNLI dataset.

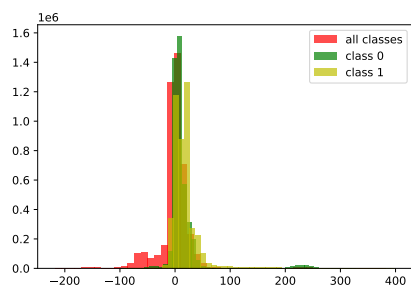


Figure 12: GD score distribution on the BigCloneBench dataset.

## 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  $f_{\theta}$  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

$$\begin{aligned} \mathbf{a} &= W\mathbf{u} \\ \hat{\mathbf{y}} &= \delta(\mathbf{a}) \end{aligned}$$

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

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

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

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

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

$$= \frac{\partial \ell}{\partial \hat{\mathbf{y}}} \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{a}} \quad (7)$$

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

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

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 that 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.

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

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  $\hat{\mathbf{y}}'$ .

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

$\alpha \approx 1$  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 at a quadratic rate. The means, as the training progresses, data points from different classes

become more and more independent. The gradients of data points from different classes also become more and more perpendicular.

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  $f_\theta$  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. Because  $|\nabla_{\mathbf{a}}\ell^\top\nabla_{\mathbf{a}'}\ell|$  is small, a tiny noise in the logits  $\mathbf{a}$  and  $\mathbf{a}'$  can flip the sign of  $\nabla_{\mathbf{a}}\ell^\top\nabla_{\mathbf{a}'}\ell$  and change the direction of influence.

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 and almost always is positive.

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

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

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  $k' \neq k$ .



## ACL 2023 Responsible NLP Checklist

---

### A For every submission:

- A1. Did you describe the limitations of your work?  
*Limitations section*
- A2. Did you discuss any potential risks of your work?  
*Ethics Statement*
- A3. Do the abstract and introduction summarize the paper’s main claims?  
*Left blank.*
- A4. Have you used AI writing assistants when working on this paper?  
*Left blank.*

### B Did you use or create scientific artifacts?

*Left blank.*

- B1. Did you cite the creators of artifacts you used?  
*Not applicable. Left blank.*
- B2. Did you discuss the license or terms for use and / or distribution of any artifacts?  
*Not applicable. Left blank.*
- B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?  
*Not applicable. Left blank.*
- B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?  
*Not applicable. Left blank.*
- B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?  
*Not applicable. Left blank.*
- B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.  
*Not applicable. Left blank.*

### C Did you run computational experiments?

*Left blank.*

- C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?  
*Left blank.*

---

*The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.*

- C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?

*Left blank.*

- C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?

*Left blank.*

- C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?

*Left blank.*

**D  Did you use human annotators (e.g., crowdworkers) or research with human participants?**

*Left blank.*

- D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?

*Not applicable. Left blank.*

- D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?

*Not applicable. Left blank.*

- D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?

*Not applicable. Left blank.*

- D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?

*Not applicable. Left blank.*

- D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?

*Not applicable. Left blank.*