Faithful and Fast Influence Function via Advanced Sampling

Jungyeon Koh [1] **Hyeonsu Lyu** [1] **Jonggyu Jang** [1] **Hyunjong Yang** [1]

Abstract

How can we explain the influence of training data on black-box models? Influence functions (IFs) offer a post-hoc solution by utilizing gradients and Hessians. However, computing the Hessian for an entire dataset is resource-intensive, necessitating a feasible alternative. A common approach involves randomly sampling a small subset of the training data, but this method often results in highly inconsistent IF estimates due to the high variance in sample configurations. To address this, we propose two advanced sampling techniques based on features and logits. These samplers select a small yet representative subset of the entire dataset by considering the stochastic distribution of features or logits, thereby enhancing the accuracy of IF estimations. We validate our approach through class removal experiments, a typical application of IFs, using the F_1-score to measure how effectively the model forgets the removed class while maintaining inference consistency on the remaining classes. Our method reduces computation time by 30.1% and memory usage by 42.2%, or improves the F_1-score by 2.5% compared to the baseline. Our code will be available at https://github.com/jungyeonkoh/samplingIF.

1. Introduction

A comprehensive understanding of model behaviors has become paramount, particularly as ensuring alignment with human ethics and societal values emerges as a critical concern in the renaissance of hyper-scale models. The recent technical report (Park et al., 2024) exemplified such concerns, addressing the potential of AI to deceive humans. However, the paradigm shift towards deeper and larger architectures has posed significant challenges for providing explainability and interpretability.

Influence functions—originating from classical statistics (Hampel, 1974)—have revived as a crucial breakthrough in enhancing the transparency and accessibility of black-box AI models (Koh & Liang, 2017). For black-box AI models, influence functions provide a direct evaluation of how the inclusion or exclusion of data affects model parameters by leveraging only their Hessians and gradients. Thus, unlike traditional retraining-based analyses, such as leave-k-out validations, influence functions significantly reduce the costs associated with fine-tuning and retraining, thereby mitigating the carbon footprint of model analysis (Koh et al., 2019). Recent studies have demonstrated the robustness of influence functions across various domains, including model analysis (Koh & Liang, 2017; Kong et al., 2022), priori and post-hoc data processing (Lee et al., 2020; Yang et al., 2023; Cohen et al., 2020), machine unlearning (Grosse et al., 2023), and natural language processing (Jain et al., 2022; Ye et al., 2022).

While influence functions have advanced the strive for explainability in black-box model inference, they encounter two major limitations: (1) high memory and computational demands, and (2) imprecise approximations when applied to large-scale models due to the theoretical necessity for Hessian inversion. The emergence of hyperscale AI systems, such as large language models (LLMs), has worsened these issues, posing further challenges to achieving practical real-world applications. To avoid such inefficient computations, primitive influence function methods employ random sampling when computing Hessians, which still fails to accurately estimate the true leave-one-out (LOO) effect.

To resolve these challenges, we propose advanced sampling techniques designed to preserve the accuracy of influence functions while enhancing computational and memory efficiency. Figure 1 illustrates the differences between our sampling methods and conventional random sampling. Our findings confirm that employing representative data points in Hessian computations improves both the accuracy and efficiency of influence function estimations.

2. Revisiting Influence Functions

Definitions and Implications. Given a model θ of size p in the parameter space $\Theta \in \mathbb{R}^p$, and n training points $z_1, .., z_n \in Z$, the empirical risk is defined as $L(\theta) = \frac{1}{n}\sum_{i=1}^{n} l(z_i;\theta)$. Accordingly, the empirical risk minimizer is $\hat{\theta} = \arg\min_{\theta\in\Theta} L(\theta)$.

[1]Department of Electrical Engineering, Pohang University of Science and Technology, Pohang, Republic of Korea.

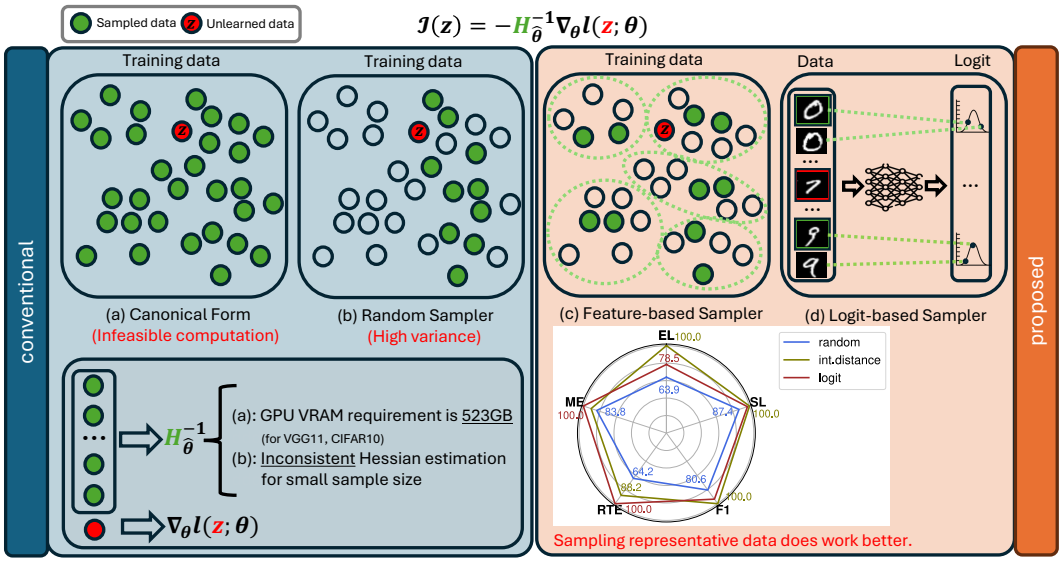

Figure 1. Overview of our approach. A quick evaluation shows the performance of three sampling methods under five metrics: exclusive-loss (EL), self-loss (SL), F1 score (F1), run-time efficiency (RTE), and memory efficiency (ME). Results show that improved samplings lead to more accurate estimations of unlearning effects within less memory and time. The influence functions require Hessian matrix of the sampled training dataset and gradient vector of the target data. In conventional methods, the Hessian matrix is (a) intractable or (b) possibly unreliable. In our method, advanced samplers can choose a small but representative subset based on (c) feature and (d) logits.

Now, influence functions compute the parameter change if a certain data point z is upweighted by some small ϵ. Hence, an ϵ-upweighted empirical risk minimizer is defined as

$$\hat{\theta}_{\epsilon,z} = \arg\min_{\theta\in\Theta} L(\theta) + \epsilon l(z;\theta). \tag{1}$$

Influence functions can be derived by using a Taylor expansion and a single Newton step as follows:

$$\mathcal{I}(z) := \left.\frac{d(\hat{\theta}_{\epsilon,z} - \hat{\theta})}{d\epsilon}\right|_{\epsilon=0} = -\mathbf{H}_{\hat{\theta}}^{-1}\nabla_\theta l(z;\theta), \tag{2}$$

where the Hessian is $\mathbf{H}_{\hat{\theta}} := \frac{1}{n}\sum_{i=1}^{n}\nabla_\theta^2 l(z_i;\hat{\theta}) \in \mathbb{R}^{p\times p}$.

IFs align well with LOO retraining for linear models, but Koh et al. (2019); Basu et al. (2020a) revealed that this breaks down when applied to larger datasets or deeper models. This discrepancy arises from a strong convexity assumption, which is often violated in modern deep neural networks. Moreover, Bae et al. (2022) showed that IFs align better with the proximal Bregman response function (PBRF), which approximates the effect of removing a data point while preserving prediction consistency on the remaining dataset. Since PBRF can effectively address questions about model behaviors, IFs remain a valuable post-hoc analysis tool, serving as a good approximation of PBRF.

LiSSA for a faster computation. Given $p = |\Theta|$, inverting $p \times p$ Hessian as in (2) imposes a huge computational bottleneck with a complexity of $O(p^3)$. Accordingly, Koh & Liang (2017) employed an iterative method using LiSSA (Agarwal et al., 2017) to compute the inverse-Hessian-vector product (iHVP) instead of directly inverting the Hessians. The iterative approximation can be represented as

$$\mathcal{I}_k = \mathcal{I}_0 + (\mathbf{I} - \mathbf{H}_{\hat{\theta}})\mathcal{I}_{k-1}, \tag{3}$$

where index k indicates the timestep in this iterative process and $\mathcal{I}_0 = \nabla l(z;\hat{\theta})$. $\mathbf{H}_{\hat{\theta}}$ is estimated using $\nabla^2 l(z_{s_i};\hat{\theta})$ from t randomly selected data samples $z_{s_1}, ..., z_{s_t}$. This recursive series converges to $\mathcal{I}(z)$ as $k \to \infty$ based on the validity of the Taylor expansion. The iteration stops when $\|\mathcal{I}_{k+1} - \mathcal{I}_k\| \leq \delta$ for a predefined threshold δ.

Shortcomings of LiSSA. The LiSSA iteration tends to produce inaccurate influence estimations. Additionally, it has a time complexity of $O(nrp)$ for n data points, r iterations, and p parameters. To address these inherent challenges, Basu et al. (2020b); Yeh et al. (2022); Koh & Liang (2017) have suggested "optimizing" the computational budget associated with n and p by sampling the dataset and freezing network layers, which still fail to enhance accuracy. This finding aligns with Feldman & Zhang (2020), who confirmed that estimation errors can occur even in simple single-layer networks. Moreover, Basu et al. (2020b) employed a second-order approximation, and Teso et al. (2021) used a Fisher information matrix (Lehmann & Casella, 2006) to improve accuracy. Nevertheless, both approaches endure a sharp increase in computational complexity, making them impractical for real-world applications.

Conversely, we believe that random sampling is responsible for inaccurate and unreliable LiSSA iterations. This is due to the high variance of the average loss associated with a limited number of sampling procedures and sampled instances. While expected Hessians and gradients from randomly sampled points are theoretically unbiased, the practical implementations suffers from this variance.

3. Sampling Methods

We assume that sampling representative data could enhance the accuracy and consistency of computing $\mathcal{I}_k$, thereby reducing the required iterations. Influence functions mostly rely on random sampling to estimate $\mathbf{H}_{\hat{\theta}} \approx \mu(\nabla^2 l(z_{s_i}; \hat{\theta}))$ (Koh & Liang, 2017), which suffers from high variance. Conversely, employing advanced sampling methods could yield a more robust Hessian approximation. In essence, we aim to "optimize" the computational complexity in terms of r by expediting the convergence of the LiSSA algorithm.

In this section, we introduce several novel sampling methods based on the features and logits of the training data.

Feature-based sampling. We assume that organizing data points within a latent feature space and selecting samples based on the space topology can avoid the unexpected variance of random samplers. Hence, we extract features in an extrinsic and intrinsic manner, then sample the features using two sampling methods.

We adopt a pre-trained Vision Transformer (ViT) model (Dosovitskiy et al., 2020) as an extrinsic feature extractor because the ViT is well-known for its effectiveness in extracting general features in diverse network architectures. However, the pre-trained ViT model takes additional time for fine-tuning; and we cannot tell how much the sampling contributes to the accuracy of influence functions when employing additional model in estimating influence functions. Accordingly, as part of an ablation approach, we design an intrinsic feature extractor, which directly uses the network being investigated to avoid transfer of trust problems.

Thereafter, we develop two sampling methods using both extrinsic and intrinsic feature extractors as follows:

- **Top-k sampling**: Compute C centroids in the feature space using the K-means algorithm for a pre-defined C. Then, select k samples that are the nearest to each centroid, resulting in a total selection of kC samples.
- **Distance-weighted sampling** (Wu et al., 2017): For each extracted feature z_i and centroid c, compute the l_2 distance $d_{z_i,c}$ and create a multinomial distribution with probability

$$p_{z_i,c} = \frac{1}{d_{z_i,c} - (\min_{z \in \mathbf{z}} d_{z,c} - \epsilon)}, \tag{4}$$

  for $\epsilon > 0$. A larger ϵ increases the probability for data points farther from the centroid, adding a certain degree of stochasticity compared to top-k sampling. Then, select k samples from the multinomial distribution for each centroid c, again resulting in kC samples in total.

Combining the two feature extractors with the two sampling methods described above, we have four feature-based samplers as follows: extrinsic Top-k sampling (**ext. top-k**), intrinsic Top-k sampling (**int. top-k**), extrinsic distance-weighted sampling (**ext. distance**), and intrinsic distance-weighted sampling (**int. distance**).

Logit-based sampling. We design another logit-based sampler (**logit**) based on a class-wise softmax score of each data point x_i across Y classes. This involves creating a multinomial distribution for each class $y \in \mathbf{y}$ with the probability of

$$p_{x_i,y} = [\text{softmax}(x_i; \theta)]_y, \tag{5}$$

Then, k samples are chosen for each class y from the multinomial, resulting in kY samples in total.

4. Experiments

We evaluate the efficacy of our sampling methods by performing a class removal task using the original influence function (Koh & Liang, 2017).

The experiments are performed on VGG11 (Simonyan & Zisserman, 2014) trained with CIFAR-10 as described in (Koh & Liang, 2017; Lyu et al., 2024), and training points labeled as "8" (horse) are removed. In addition, we evaluate the sampling methods on class removal tasks with the other influence alternatives (Agarwal et al., 2017; Guo et al., 2020; Schioppa et al., 2022; Lyu et al., 2024) and datasets in Appendix A.

Evaluation metrics. We outline the following metrics to evaluate the accuracy and computational efficiency of the sampling methods.

- Self-loss (SL): The loss for the removed data, denoted as $\sum_{z \in Z'} l(z; \theta)$, where Z' is a set of all removed data points.
- F_1-score (F_1): A modified F_1 score, incorporating self-accuracy (SA) and exclusive-accuracy (EA) as $\mathrm{F}_1 = 2\frac{\mathrm{EA}(1-\mathrm{SA})}{1+\mathrm{EA}-\mathrm{SA}}$.
- Run-time efficiency (RTE): The average computing time until the influence function converges.
- Memory efficiency (ME): The peak memory consumption while computing influence functions measured by monitoring memory usage.

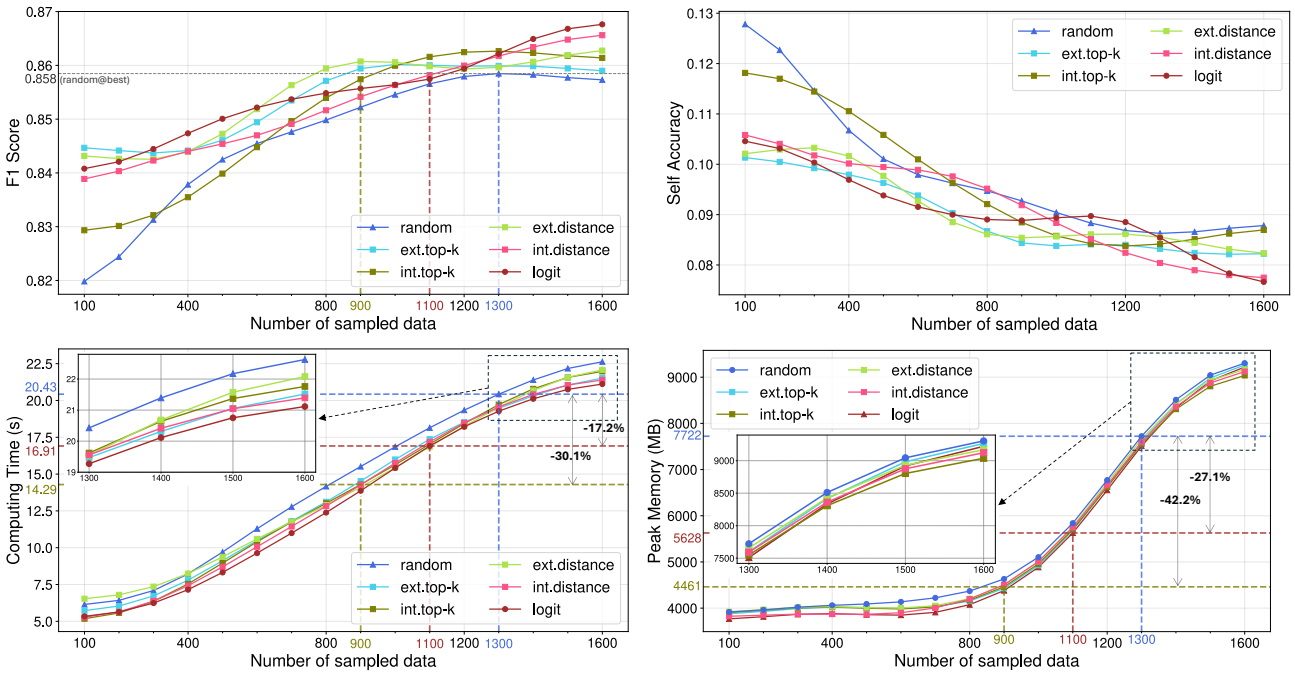

Figure 2. Evaluation results on the class removal task for VGG11 with CIFAR-10. Lower values indicate better performance, except for F_1-score.

Results. Figure 2 illustrates how the network behaves using the above metrics when the number of samples increases. The graphs are obtained by averaging the results of 25 individual experiments. Since the standard deviation (SD) is another critical indicator to verify the faithfulness of the sampling methods, we provide the corresponding SD of Fig. 2 in Appendix B.

We summarize the key findings as follows:

- **The logit yields the most accurate estimation over other methods.** Notably, both the F_1-score and self-accuracy significantly improve as the number of samples increases. We believe the superior performance of the **logit** is likely due to its utilization of the entire neural network, unlike other sampling methods. Remarkably, the **logit** takes the least compute cost as it just maps the softmax result without any intervention of external neural network or K-means algorithm.

- **Distance-weighted samplers perform slightly worse than the logit, but still show satisfactory results.** The **int. distance** and **ext. distance** sampler also shows comparable results to the **logit** in both F_1-score and self-accuracy. The result implies that distribution-based samples from {**logit**, **int. distance**, **ext. distance**} provide a more comprehensive representation of the entire dataset than the samples from deterministic samplers {**int. distance**, **ext. distance**} and **random**.

- **Intrinsic samplers outperform extrinsic samplers.** For both top-k and distance-weighted sampling, using the model itself as an intrinsic feature extractor yields more accurate estimations than employing an additional ViT model as an extrinsic feature extractor. It indicates that using the model being investigated for feature extraction more effectively represents the true feature space than relying on an external model.

- **The random gets comparable to the other samplers as the sample count increases.** This is natural as the sample data points become sufficiently representative even selected by the **random**.

- **Remarkably, the logit and int. top-k greatly reduce both execution time and memory.** To achieve the best F_1-score of the **random**, the number of samples required is 1,300 for the **random**. Meanwhile, the **logit** and **int. top-k** only require 900 and 1,100 sample counts. As a result, the **logit** and **int. top-k** save 17.2% and 30.1% in computing time, and 22.3% and 40.6% in memory, while maintaining the same performance.

5. Discussion

Summary. This paper deals with the challenge of efficient data sampling for computing influence functions on black-box AI models. Traditional methods relying on random sampling often produce inaccurate influence estimations due to bias. To address this, we propose advanced samplers

based on features and logits, selecting a representative subset of the dataset. Our experiments show that the proposed methods improve the accuracy of influence functions even with less time and memory usage.

Limitation and future works. Our methods provide efficient sampling methods for estimating influence functions, which consistently outperform the baselines. However, our experimental analysis lacks a variety applications for influence functions. Also, as a future plan, we aim to explore the efficacy of sampling removal data in expediting class unlearning tasks. Furthermore, we plan to devise an effective update rule for influence functions with a much larger update rate than the theoretical value.

Societal impact. AIs exhibit striking capabilities beyond our imagination, but their internal mechanisms remain obscure. Influence functions, which leverage training data to explain models, serve as a cornerstone of a bottom-up approach for mechanistic understanding of AI. We anticipate that influence functions contribute to building robust AI systems through a comprehensive understanding of the system.

Acknowledgements

This research was supported by the IITP(Institute for Information & Communications Technology Planning & Evaluation), grant funded by MSIT(Ministry of Science and ICT) (RS-2023-00229541, Development of Big Data and Artificial Intelligence Based Radio Monitoring Platform). This research was also supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2024-2021-0-02048) supervised by the IITP(Institute for Information & Communications Technology Planning & Evaluation).

Impact Statement

This paper aims to advance the field of machine learning. While there are many potential societal implications of our work, we do not believe any need to be specifically highlighted here.

References

Agarwal, N., Bullins, B., and Hazan, E. Second-order stochastic optimization for machine learning in linear time. *Journal of Machine Learning Research*, 18(116): 1–40, 2017.

Bae, J., Ng, N., Lo, A., Ghassemi, M., and Grosse, R. B. If influence functions are the answer, then what is the question? *Advances in Neural Information Processing Systems*, 35:17953–17967, 2022.

Basu, S., Pope, P., and Feizi, S. Influence functions in deep learning are fragile. *arXiv preprint arXiv:2006.14651*, 2020a.

Basu, S., You, X., and Feizi, S. On second-order group influence functions for black-box predictions. In *International Conference on Machine Learning*, pp. 715–724. PMLR, 2020b.

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

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Feldman, V. and Zhang, C. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33:2881–2891, 2020.

Grosse, R., Bae, J., Anil, C., Elhage, N., Tamkin, A., Tajdini, A., Steiner, B., Li, D., Durmus, E., Perez, E., et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.

Guo, H., Rajani, N. F., Hase, P., Bansal, M., and Xiong, C. Fastif: Scalable influence functions for efficient model interpretation and debugging. *arXiv preprint arXiv:2012.15781*, 2020.

Hampel, F. R. The influence curve and its role in robust estimation. *Journal of the american statistical association*, 69(346):383–393, 1974.

Jain, S., Manjunatha, V., Wallace, B., and Nenkova, A. Influence functions for sequence tagging models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 824–839, December 2022. URL https://aclanthology.org/2022.findings-emnlp.58.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. PMLR, 2017.

Koh, P. W. W., Ang, K.-S., Teo, H., and Liang, P. S. On the accuracy of influence functions for measuring group effects. *Advances in neural information processing systems*, 32, 2019.

Kong, S., Shen, Y., and Huang, L. Resolving training biases via influence-based data relabeling. In *The 10th International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=EskfH0bwNVn.

Lee, D., Park, H., Pham, T., and Yoo, C. D. Learning augmentation network via influence functions. In *Proceedings of 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2020.

Lehmann, E. L. and Casella, G. *Theory of point estimation*. Springer Science & Business Media, 2006.

Lyu, H., Jang, J., Ryu, S., and Yang, H. J. Deeper understanding of black-box predictions via generalized influence functions, 2024.

Park, P. S., Goldstein, S., O'Gara, A., Chen, M., and Hendrycks, D. Ai deception: A survey of examples, risks, and potential solutions. *Patterns*, 5(5), 2024.

Schioppa, A., Zablotskaia, P., Vilar, D., and Sokolov, A. Scaling up influence functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8179–8186, 2022.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Teso, S., Bontempelli, A., Giunchiglia, F., and Passerini, A. Interactive label cleaning with example-based explanations. *Advances in Neural Information Processing Systems*, 34:12966–12977, 2021.

Wu, C.-Y., Manmatha, R., Smola, A. J., and Krahenbuhl, P. Sampling matters in deep embedding learning. In *Proceedings of the IEEE international conference on computer vision*, pp. 2840–2848, 2017.

Yang, S., Xie, Z., Peng, H., Xu, M., Sun, M., and Li, P. Dataset pruning: Reducing training data by examining generalization influence. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=4wZiAXD29TQ.

Ye, J., Gao, J., Wu, Z., Feng, J., Yu, T., and Kong, L. ProGen: Progressive zero-shot dataset generation via in-context feedback. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3671–3683, December 2022. URL https://aclanthology.org/2022.findings-emnlp.269.

Yeh, C.-K., Taly, A., Sundararajan, M., Liu, F., and Ravikumar, P. First is better than last for language data influence. *Advances in Neural Information Processing Systems*, 35: 32285–32298, 2022.

A. Accuracy Evaluation Details

All experiments are performed on Linux with an NVIDIA Geforce RTX 3080Ti (12GB) GPU. CUDA version is 11.6 and Driver Version is 510.108.03. All codes are written under Python 3.10.10 and PyTorch 2.3.0.

Altogether, for each dataset and model pair, both a target classification model and a ViT model, used as an extrinsic feature extractor, are initially well-trained. Subsequently, samplers generate sampled data based on its feature or logit distribution. To evaluate the performance of our sampling methods among different influence function approaches, we utilize five benchmarks: LiSSA-based IF (IF) (Koh & Liang, 2017), projected IF (PIF), generalized IF (GIF), freezed IF (FIF) (Lyu et al., 2024), and second-order IF (SIF) (Basu et al., 2020b). Using these benchmarks alongside our sampling methods, we conduct a class removal experiment on: (1) Alexnet with the MNIST dataset, and (2) VGG11 with the CIFAR-10 dataset. The accuracy and consistency of our sampling methods on all datasets are presented in Table 1 and Table 2, respectively.

Table 1. Overall performance benchmark of sampling methods for various influence function methods. The Alexnet model and the MNIST dataset are used. The best and the second-best performing methods are highlighted in bold, with the best-performing ones also marked with a superscript asterisk.

		Random	Ext. Top-k	Int. Top-k	Ext. Distance	Int. Distance	Logit
IF	EL(↓)	0.051 ± 0.008	0.058 ± 0.014	0.063 ± 0.021	0.052 ± 0.012	0.055 ± 0.008	0.050 ± 0.005
	SL(↑)	7.77 ± 1.74	8.71 ± 2.62	8.95 ± 1.21	7.72 ± 1.82	8.36 ± 1.69	7.93 ± 0.85
	F1(↑)	0.963 ± 0.016	0.967 ± 0.012	0.967 ± 0.012	0.955 ± 0.029	**0.968 ± 0.011**	**0.967 ± 0.006***
PIF	EL(↓)	0.097 ± 0.039	0.076 ± 0.043	0.192 ± 0.023	0.066 ± 0.014	0.120 ± 0.019	0.078 ± 0.047
	SL(↑)	8.95 ± 2.30	9.34 ± 2.52	8.21 ± 1.12	9.01 ± 1.51	9.73 ± 1.14	8.41 ± 0.34
	F1(↑)	0.979 ± 0.011	0.964 ± 0.046	0.958 ± 0.005	**0.990 ± 0.002***	0.981 ± 0.003	**0.988 ± 0.006**
FIF	EL(↓)	0.140 ± 0.056	0.085 ± 0.036	0.182 ± 0.080	0.066 ± 0.027	0.147 ± 0.050	0.082 ± 0.040
	SL(↑)	9.88 ± 1.98	9.34 ± 2.53	8.91 ± 1.47	8.99 ± 2.01	9.89 ± 0.97	8.13 ± 0.61
	F1(↑)	0.980 ± 0.006	0.960 ± 0.050	0.965 ± 0.014	**0.989 ± 0.004***	0.979 ± 0.003	**0.987 ± 0.006**
GIF	EL(↓)	0.100 ± 0.031	0.091 ± 0.029	0.152 ± 0.065	0.067 ± 0.026	0.126 ± 0.026	0.106 ± 0.022
	SL(↑)	8.06 ± 1.89	9.16 ± 2.18	8.88 ± 1.97	9.55 ± 1.84	8.79 ± 1.57	8.12 ± 0.85
	F1(↑)	0.978 ± 0.010	0.953 ± 0.062	0.956 ± 0.019	**0.990 ± 0.003***	0.979 ± 0.003	**0.982 ± 0.002**
SIF	EL(↓)	0.040 ± 0.058	0.084 ± 0.044	0.048 ± 0.042	0.076 ± 0.008	0.058 ± 0.021	0.020 ± 0.024
	SL(↑)	10.05 ± 2.39	13.14 ± 3.01	10.44 ± 2.81	10.12 ± 2.13	10.36 ± 1.96	10.50 ± 1.68
	F1(↑)	0.948 ± 0.055	**0.976 ± 0.030**	0.958 ± 0.047	0.951 ± 0.034	0.958 ± 0.081	**0.978 ± 0.030***

Table 2. Overall performance benchmark of sampling methods for various influence function methods. The VGG11 model and the CIFAR-10 dataset are used. The best and the second-best performing methods are highlighted in bold, with the best-performing ones also marked with a superscript asterisk.

		Random	Ext. Top-k	Int. Top-k	Ext. Distance	Int. Distance	Logit
IF	EL(↓)	0.777 ± 0.138	0.646 ± 0.089	0.628 ± 0.090	0.614 ± 0.078	0.608 ± 0.050	0.678 ± 0.070
	SL(↑)	7.83 ± 1.27	8.24 ± 1.35	8.04 ± 1.15	7.80 ± 0.73	8.14 ± 0.32	8.27 ± 0.87
	F1(↑)	0.869 ± 0.012	**0.874 ± 0.018**	0.874 ± 0.019	0.871 ± 0.017	0.867 ± 0.013	**0.881 ± 0.012** *
PIF	EL(↓)	1.189 ± 0.382	1.580 ± 0.194	1.184 ± 0.270	1.245 ± 0.162	1.438 ± 0.222	0.936 ± 0.081
	SL(↑)	7.93 ± 1.10	7.45 ± 0.58	8.56 ± 0.97	7.10 ± 0.63	8.24 ± 1.03	8.84 ± 0.56
	F1(↑)	0.826 ± 0.049	0.781 ± 0.030	**0.828 ± 0.047**	0.800 ± 0.031	0.809 ± 0.054	**0.863 ± 0.015***
FIF	EL(↓)	1.271 ± 0.359	0.980 ± 0.227	0.952 ± 0.233	0.916 ± 0.170	0.950 ± 0.351	1.154 ± 0.185
	SL(↑)	7.04 ± 1.44	7.90 ± 0.54	7.72 ± 1.14	8.43 ± 0.43	7.98 ± 0.93	8.36 ± 0.83
	F1(↑)	0.800 ± 0.055	0.852 ± 0.025	0.857 ± 0.038	**0.859 ± 0.020**	**0.860 ± 0.043***	0.835 ± 0.029
GIF	EL(↓)	1.231 ± 0.483	1.261 ± 0.261	1.064 ± 0.240	1.175 ± 0.256	1.091 ± 0.356	1.030 ± 0.344
	SL(↑)	7.33 ± 1.14	8.13 ± 0.93	8.28 ± 0.68	8.47 ± 1.11	7.87 ± 0.92	8.40 ± 0.64
	F1(↑)	0.811 ± 0.066	0.814 ± 0.039	**0.847 ± 0.035**	0.829 ± 0.026	0.828 ± 0.059	**0.852 ± 0.048***
SIF	EL(↓)	0.984 ± 0.242	1.297 ± 0.228	1.045 ± 0.249	1.380 ± 0.287	0.987 ± 0.341	0.589 ± 0.320
	SL(↑)	7.26 ± 1.39	6.95 ± 1.26	9.62 ± 1.44	6.90 ± 1.28	7.64 ± 1.52	8.11 ± 1.30
	F1(↑)	0.839 ± 0.052	0.819 ± 0.060	**0.845 ± 0.027**	0.834 ± 0.059	0.826 ± 0.032	**0.897 ± 0.021***

B. Consistency Evaluation Details

We assess the standard deviation (SD) of evaluation results to measure the consistency of influence function estimations with our novel sampling methods. The standard deviation of Fig. 2 is shown in Figure 3. Based on this figure, the following observation can be additionally made:

- **Proposed samplers provide more faithful evaluations with smaller deviations compared to the random sampler.** In particular, the logit-based sampler shows the smallest standard deviation, while extrinsic and intrinsic distance-weighted based samplers rank the second and third-smallest. This result strengthens our previous observation that stochastic sampling methods provide more accurate and consistent results than deterministic sampling methods.

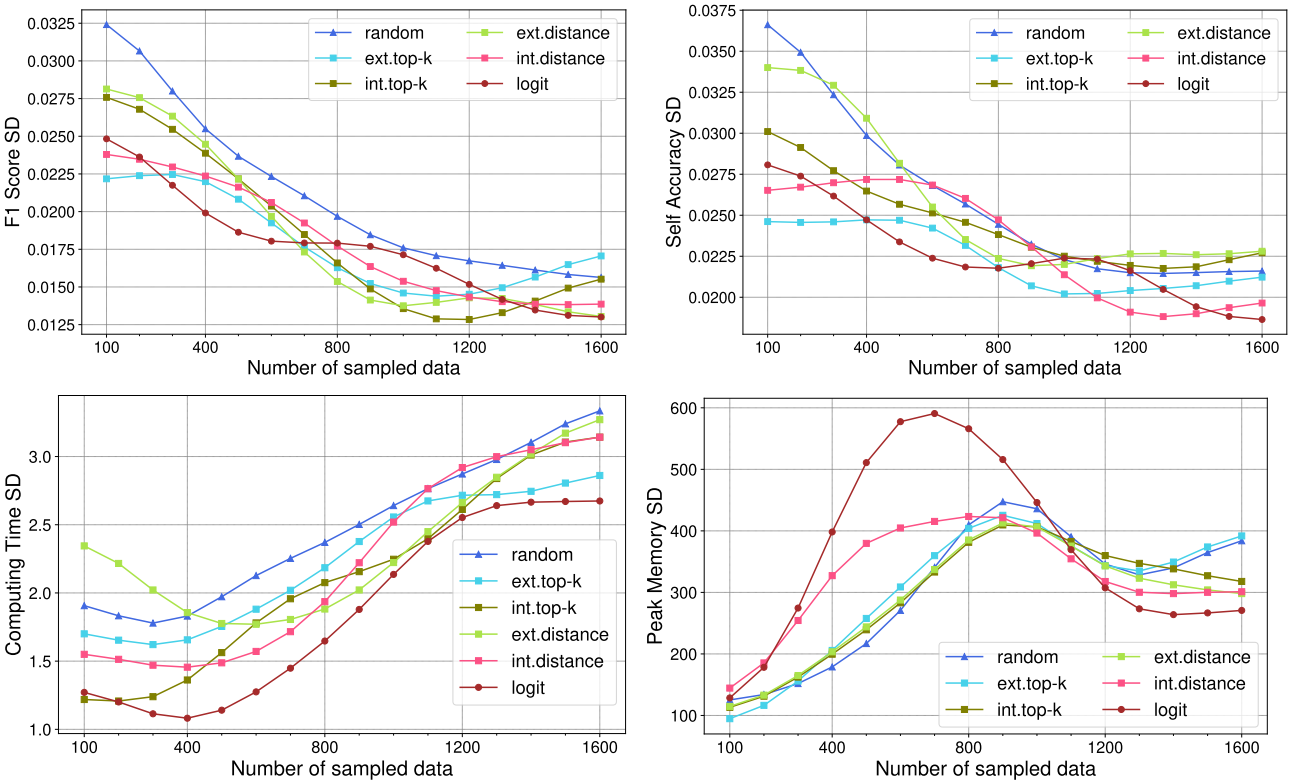

Figure 3. Standard deviation of evaluation metrics presented in Fig. 2.