GEX: A flexible method for approximating influence via Geometric Ensemble (NeurIPS 2023)

Sung-Yub Kim (presenter), Kyungsu Kim, Eunho Yang

Oct. 6th, 2023

KAIST

MLILAB
Machine Learning & Intelligence

TL;DR

- Influence Function (IF) and its approximations suffer from high computational cost and framework dependency. Furthermore, we find & prove that these methods suffer from a distributional bias due to their bilinear form.

- To mitigate this issue, we propose a novel IF approximation method with Geometric Ensemble.

- We empirically verify that the proposed method improves the downstream task performance and relieves the framework dependency of IF (and its approximations).

MLILAB Machine Learning & Intelligence

Overview

1. Introduction
- Influence Function (IF)
- Two limitations of IF

2. Identifying distributional bias in IF and its approximations
- Distributional bias in IF and its approximations
- A simple case study in a modified two-circle dataset

3. Resolving distributional bias via Geometric Ensemble
- Step 1. Removing linearization
- Step 2. Utilization of Geometric Ensemble
- Empirical evaluations

4. Discussion points

MLILAB Machine Learning & Intelligence

Related works

[1] Cook, R. Dennis. "Detection of influential observation in linear regression." Technometrics 19.1 (1977): 15-18.
[2] Koh, Pang Wei, and Percy Liang. "Understanding black-box predictions via influence functions." International conference on machine learning. PMLR, 2017.
[3] Pruthi, Garima, et al. "Estimating training data influence by tracing gradient descent." Advances in Neural Information Processing Systems 33 (2020): 19920-19930.
[4] Schioppa, Andrea, et al. "Scaling up influence functions." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 36. No. 8. 2022.
[5] Sorscher, Ben, et al. "Beyond neural scaling laws: beating power law scaling via data pruning." Advances in Neural Information Processing Systems 35 (2022): 19523-19536.
[6] Paul, Mansheej, Surya Ganguli, and Gintare Karolina Dziugaite. "Deep learning on a data diet: Finding important examples early in training." Advances in Neural Information Processing Systems 34 (2021): 20596-20607.
[7] Song, Jaeyun, Kim, SungYub, and Yang, Eunho. "RGE: A Repulsive Graph Rectification for Node Classification via Influence." International Conference on Machine Learning. PMLR, 2023.
[8] Kim, SungYub, Kim, Kyungsu, Yang, Eunho. "GEX: A flexible method for approximating influence via Geometric Ensemble". Advances in Neural Information Processing Systems (2023)
[9] Song, Jaeyun, et al. "Post-Training Recovery from Injected Bias with Self-Influence", Under review, 2023
[10] Daxberger, Erik, et al. "Laplace redux-effortless bayesian deep learning." Advances in Neural Information Processing Systems 34 (2021): 20089-20103.
[11] Kong, Zhifeng, and Kamalika Chaudhuri. "Understanding instance-based interpretability of variational auto-encoders." Advances in Neural Information Processing Systems 34 (2021): 2400-2412.
[12] Terashita, Naoyuki, et al. "Influence Estimation for Generative Adversarial Networks." International Conference on Learning Representations. 2020.
[13] Chen, Zizhang, et al. "Characterizing the Influence of Graph Elements." The Eleventh International Conference on Learning Representations. 2022.
[14] Grosse, Roger, et al. "Studying Large Language Model Generalization with Influence Functions." arXiv preprint arXiv:2308.03296 (2023).
[15] https://medium.com/data-science-in-your-pocket/random-projection-for-dimension-reduction-27d2ec7d40cd
[16] Arnoldi, Walter Edwin. "The principle of minimized iterations in the solution of the matrix eigenvalue problem." Quarterly of applied mathematics 9.1 (1951): 17-29.
[17] Garipov, Timur, et al. "Loss surfaces, mode connectivity, and fast ensembling of dnns." Advances in neural information processing systems 31 (2018).
[18] Otsu, Nobuyuki. "A threshold selection method from gray-level histograms." IEEE transactions on systems, man, and cybernetics 9.1 (1979): 62-66.
[19] Wei, Jiaheng, et al. "Learning with Noisy Labels Revisited: A Study Using Real-World Human Annotations." International Conference on Learning Representations. 2021.
[20] Kong, Shuming, Yanyan Shen, and Linpeng Huang. "Resolving training biases via influence-based data relabeling." International Conference on Learning Representations. 2021.
[21] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
[22] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).
[23] Kim, SungYub, et al. "Scale-invariant Bayesian Neural Networks with Connectivity Tangent Kernel." The Eleventh International Conference on Learning Representations. 2023.
[24] Andriushchenko, Maksym, et al. "A modern look at the relationship between sharpness and generalization." International Conference on Machine Learning. PMLR, 2023.
[25] Cohen, Jeremy, et al. "Gradient Descent on Neural Networks Typically Occurs at the Edge of Stability." International Conference on Learning Representations. 2020.
[26] Damian, Alex, Eshaan Nichani, and Jason D. Lee. "Self-Stabilization: The Implicit Bias of Gradient Descent at the Edge of Stability." The Eleventh International Conference on Learning Representations. 2022.
[27] Song, Minhak, and Chulhee Yun. "Trajectory Alignment: Understanding the Edge of Stability Phenomenon via Bifurcation Theory." arXiv preprint arXiv:2307.04204 (2023).

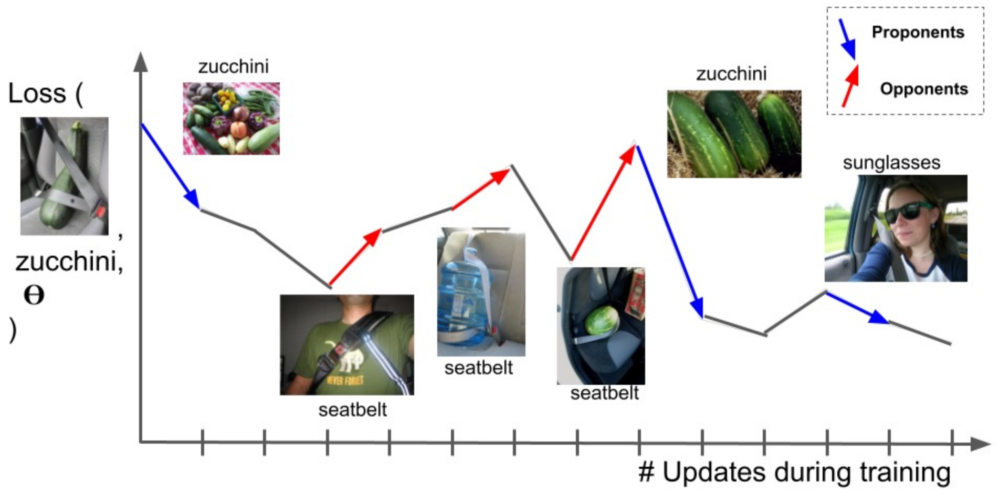
MLILAB
Machine Learning & Intelligence

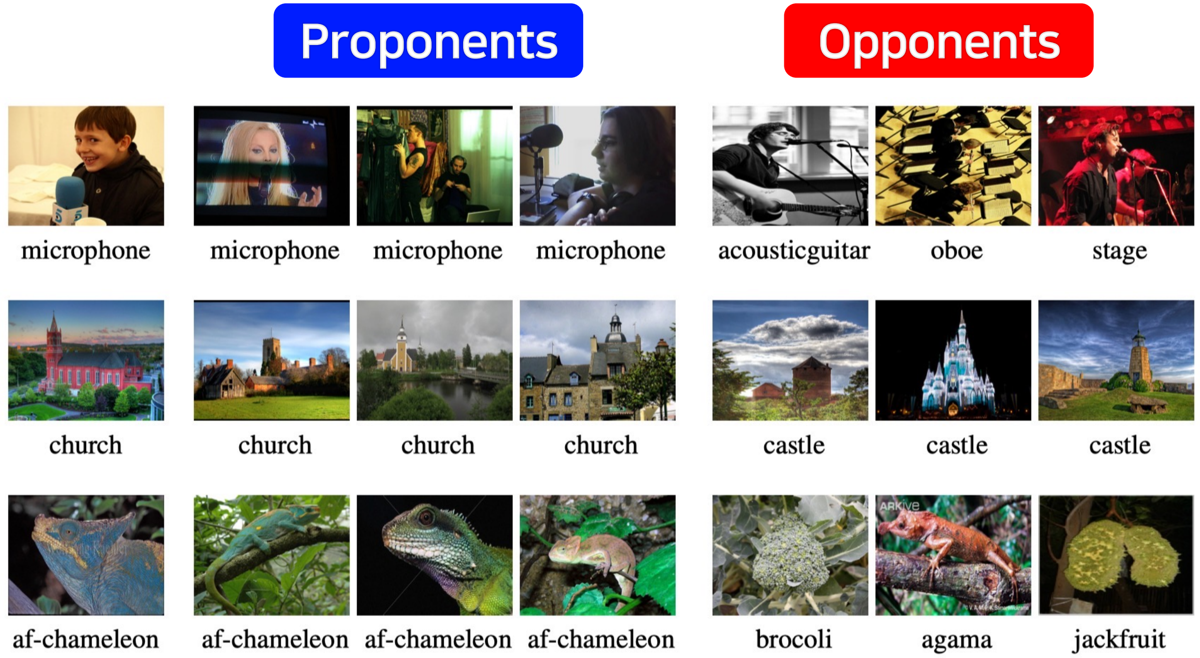
1. Introduction

- Influence Function (IF)
- Two limitations of IF

Influence Function

- Approximating the counterfactual effect of removing training samples
  - Measures how the leave-one-out (LOO) retraining of a training sample changes the loss of each sample. (similar to Cook's distance[1] in robust statistics)
  - The sign determines whether the training sample is beneficial, and the scale measures its impact.





MLILAB Machine Learning & Intelligence

Influence Function

- Approximating the counterfactual effect of removing training samples

The **Leave-One-Out (LOO) retraining effect** of training sample $z \in \mathcal{S}$ on another instance $z' \in \mathbb{R}^D$

$$\mathcal{I}_{\text{LOO}}(z, z') := \ell(z', \theta_z^*) - \ell(z', \theta^*)$$

ERM solution

$$\theta^* := \underset{\theta \in \mathbb{R}^P}{\text{argmin}}\ L(S, \theta)$$

Retrained parameter

$$\theta_z^* := \underset{\theta \in \mathbb{R}^P}{\text{argmin}}\ L(S, \theta) - \frac{\ell(z, \theta)}{N}$$

Since retraining is computationally intractable, [2] proposed an efficient approximation, named **Influence Function (IF)**:

$$\mathcal{I}(z, z') := g_{z'}^\top H^{-1} g_z$$

This can be interpreted as a two-step approximation of retraining effect

Loss linearization

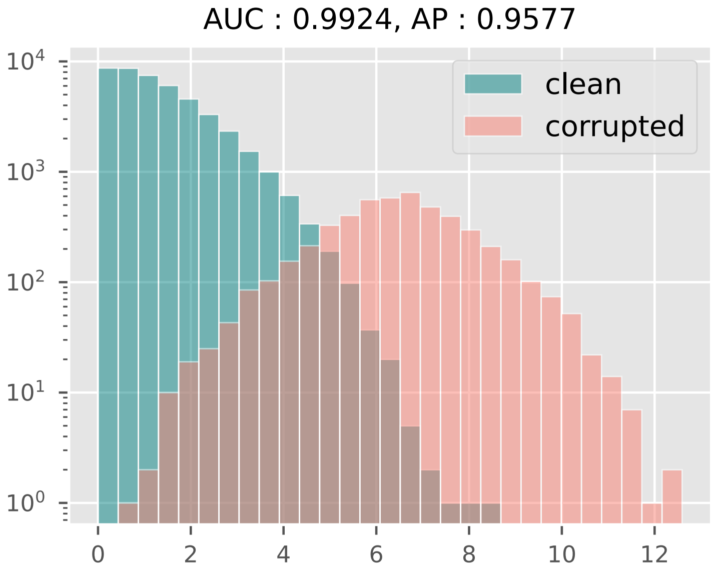
$$\begin{aligned}\mathcal{I}_{\text{LOO}}(z, z') &\approx \ell_{\theta^*}^{\text{lin}}(z', \theta_z^*) - \ell^{\text{lin}}(z', \theta^*) \\ &= g_{z'}^\top(\theta_z^* - \theta^*) \\ &\approx g_{z'}^\top H^{-1} g_z = \mathcal{I}(z, z')\end{aligned}$$

Newton ascent step

MLILAB Machine Learning & Intelligence

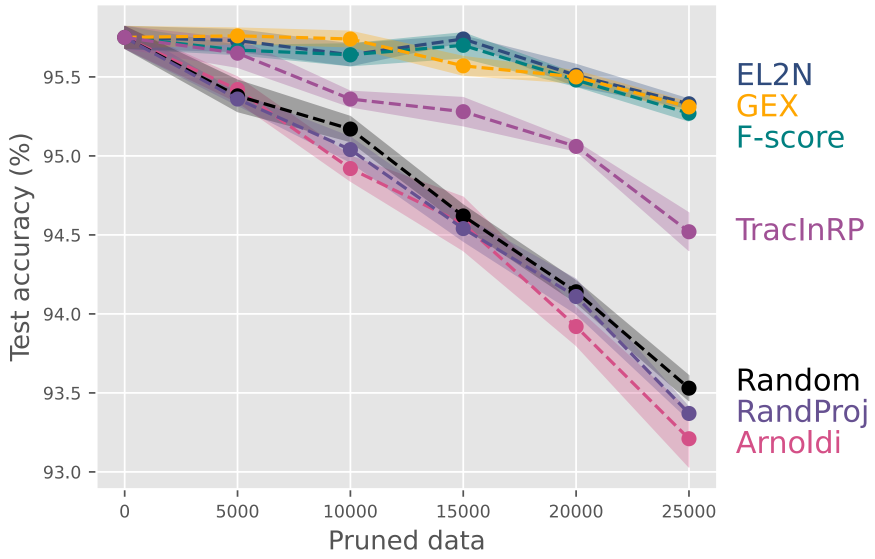
Influence Function

- Approximating the counterfactual effect of removing training samples
  - Used to noisy label detection[2, 3, 4] and dataset pruning[5, 6].



Noisy label detection

High self-influence indicates memorization.



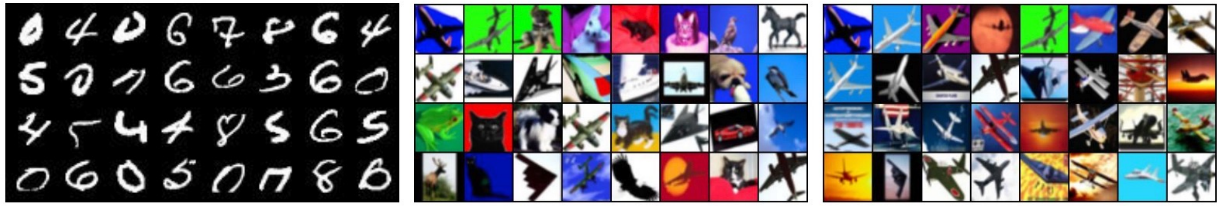
Dataset pruning

Low self-influence indicates prunable.

(easily generalizable)

Two limitations of IF

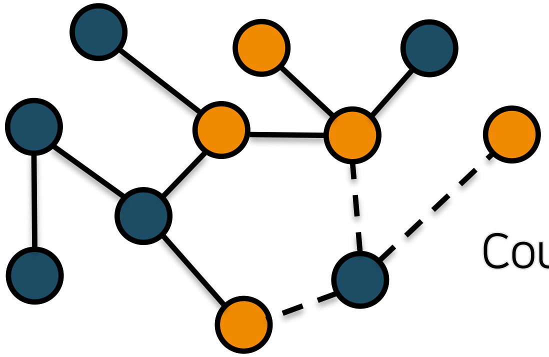
- **Limited problem structures and applications**
  - IF assumes standard supervised learning settings.
  - Therefore, formulation & intuition (e.g., interpretations of sign & scale of influence) cannot be generalized to the Generative models[11, 12], Self-Supervised Learning, and Graph Neural Networks[7, 13].



(a) MNIST (highest self-inf) (b) CIFAR (highest self-inf) (c) CIFAR-Airplane (highest self-inf)

(d) MNIST (lowest self-inf) (e) CIFAR (lowest self-inf) (f) CIFAR-Airplane (lowest self-inf)

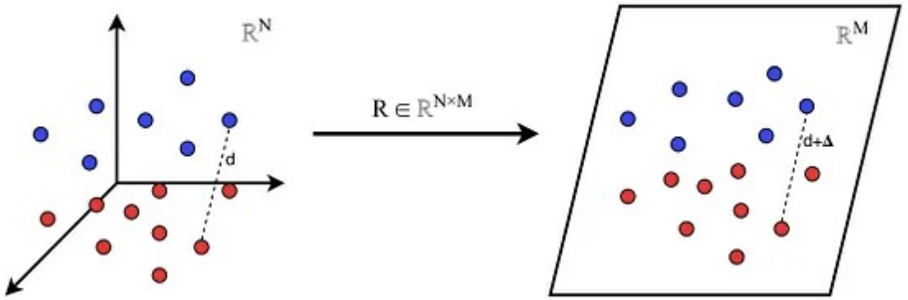
High self-influence samples are hard to recognize or high-contrast.



Counterfactual effect of removing edges?

Two limitations of IF

- High computational cost and framework dependency
- IF is still intractable to modern NN architectures (e.g., ResNet[21] and Transformer[22]) due to Hessian computation. Therefore, IF requires further approximations (e.g., stochastic conjugate gradient[2], sub-curvature approximation[14], random projection[3]).
- The Jacobian-vector product (JVP), which is only efficient for packages that support forward-mode auto-differentiation (AD), is used for the batch computation of IF (and its approximations).



Random Projection[15] can accelerate the computation of IF.

A framework supports forward-mode AD (JAX)

MLILAB Machine Learning & Intelligence

2. Identifying distributional bias in IF and its approximations

- Distributional bias in IF and its approximations
- A simple case study in a modified two-circle dataset

Distributional bias in IF and its approximations

- Efficient approximations of Influence Function

TracIn & TracInRP [3]

- By replacing the expensive inverse Hessian as an identity matrix, TracIn approximates IF as

$$\mathcal{I}_{\texttt{TracIn}}(z, z') := \frac{1}{C}\sum_{c=1}^{C} g_{z'}^{c\top} g_z^c$$

Take average over checkpoints

- Instead, the performance of TracIn is replenished by multiple checkpoints along trajectory.
- Further efficiency can be obtained by using random projection.

$$\mathcal{I}_{\texttt{TracInRP}}(z, z') := \frac{1}{C}\sum_{c=1}^{C} g_{z'}^{c\top} Q_R Q_R^\top g_z^c$$

Random projection matrix (P x R)

Arnoldi [4]

- By using the spectral decomposition of Hessian, Arnoldi approximates IF with principal components.

Principal eigenvectors of Hessian

$$\mathcal{I}_{\texttt{Arnoldi}}(z, z') := g_{z'}^\top U_R \Lambda_R^{-1} U_R^\top g_z$$

Principal eigenvalues of Hessian

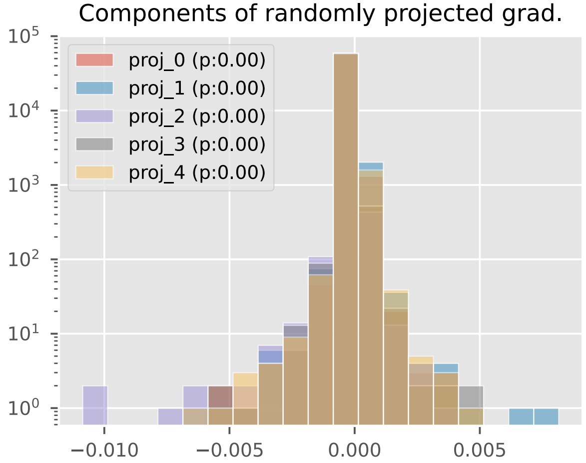
- They use Arnoldi[41] iteration to estimate principal components.
- Note that IF, TracIn(RP), and Arnoldi are all **bilinear w.r.t. sample-wise gradients**.

MLILAB Machine Learning & Intelligence

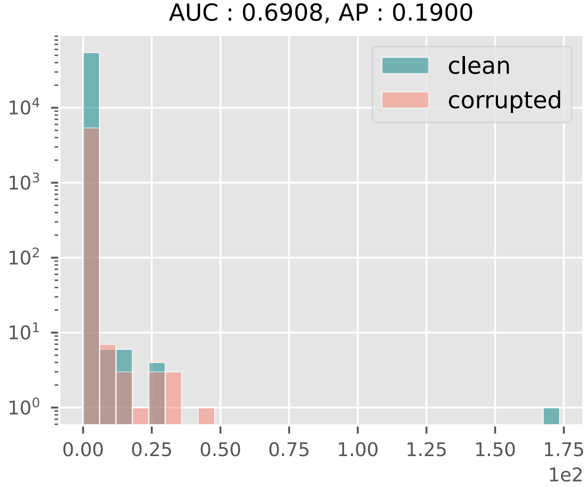
Distributional bias in IF and its approximations

- Consequently, self-influence estimated by these methods is quadratic for sample-wise gradients.

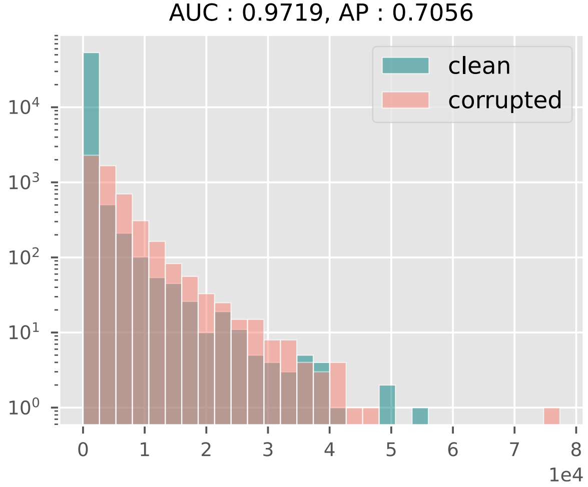
Proposition 3.1 (Distributional bias in bilinear self-influence). Let us assume g_z follows a P-dimensional stable distribution (e.g., Gaussian, Cauchy, and Lévy distribution) and $M \in \mathbb{R}^{P \times P}$ is a positive (semi-)definite matrix. Then, self-influence in the form of $\mathcal{I}_M(z, z) = g_z^\top M g_z$ follows a unimodal distribution. Furthermore, if g_z follows a Gaussian distribution, then the self-influence follows a generalized χ^2-distribution.



Randomly projected gradients and the results of normality test



Self-influence hist. of Arnoldi



Self-influence hist. of TracInRP

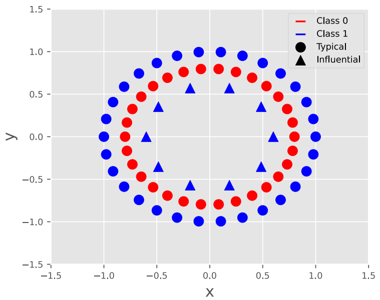
MLILAB Machine Learning & Intelligence

A simple case study in a modified two-circle dataset

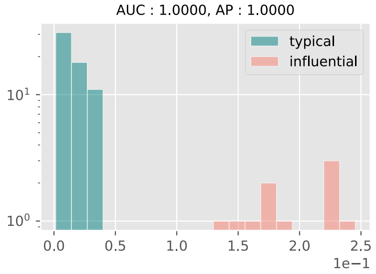
- To verify that this bias can hurt the separability of typical and influential samples, we consider the following setting.
  - We add 10 influential training samples at the center of the two-circle dataset (30 typical samples per circle).
  - $\mathcal{I}_{LOO}$: High influential samples form a separate mode of high self-influence.
  - $\mathcal{I}$: For both damping scales, influential samples are mixed with typical samples due to the distributional bias of the bilinear form.

$$H(\alpha) = H + \alpha \mathbf{I}$$

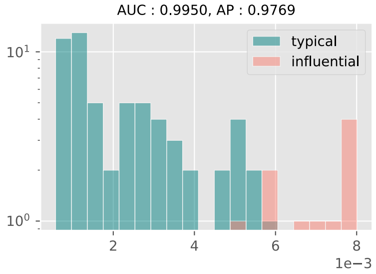
Damping scale



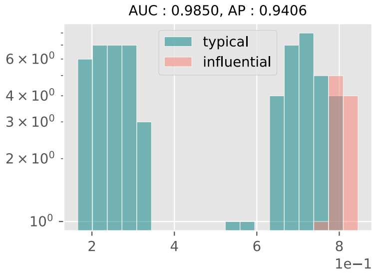
Three-circle dataset



$\mathcal{I}_{\text{LOO}}$



$\mathcal{I}\ (\alpha = 100)$



$\mathcal{I}\ (\alpha = 0.01)$

3. Resolving distributional bias via Geometric Ensemble

- Step 1. Removing linearization
- Step 2. Utilization of Geometric Ensemble
- Empirical evaluations

Removing Linearization

- Key Idea 1: Removing Linearization
  - To mitigate the distributional bias in IF, we propose an alternative view of IF through Laplace Approximation.

Theorem 4.1 (Connection between IF and LA). $\mathcal{I}$ in [28] can be expressed as

$$\mathcal{I}(z, z') = \mathbb{E}_{\psi \sim p_{\mathtt{LA}}} \left[\Delta\ell^{\text{lin}}_{\theta^*}(z, \psi) \cdot \Delta\ell^{\text{lin}}_{\theta^*}(z', \psi) \right]$$

where $\Delta\ell^{\text{lin}}_{\theta^*}(z, \psi) := \ell^{\text{lin}}_{\theta^*}(z, \psi) - \ell^{\text{lin}}_{\theta^*}(z, \theta^*) = g_z^\top(\psi - \theta^*)$ and $p_{\mathtt{LA}}$ is the Laplace approximated posterior

$$p_{\mathtt{LA}}(\psi) := \mathcal{N}\left(\psi | \theta^*, H^{-1}\right).$$

Linearized loss deviation

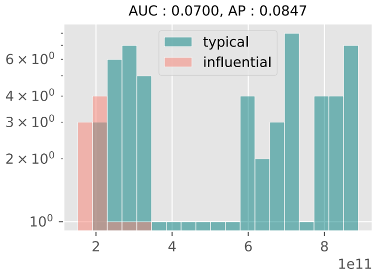
Curvature term is dominant (or uninformative prior)

- By using (non-linear) loss deviation ($\Delta\ell_{\theta^*}(z, \psi) := \ell(z, \psi) - \ell(z, \theta^*)$), we have

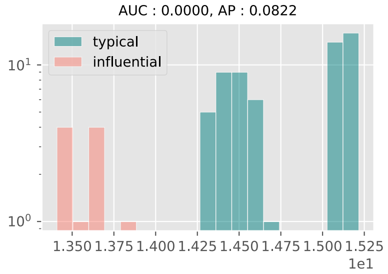
$$\mathcal{I}_{\mathtt{LA}}(z, z') := \mathbb{E}_{\psi \sim p_{\mathtt{LA}}} \left[\Delta\ell_{\theta^*}(z, \psi) \cdot \Delta\ell_{\theta^*}(z', \psi) \right].$$

Utilization of Geometric Ensemble

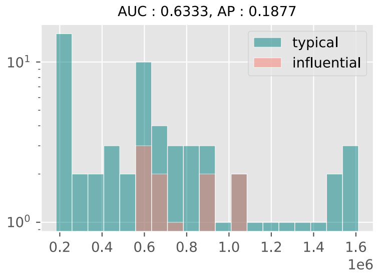
- Key Idea 2: Replace Laplace Approximation to Geometric Ensemble (GE)
  - While $\mathcal{I}_{LA}$ does not suffer from the distributional bias in theory, it still mixes typical and influential (memorized) samples and even underperforms its liner counterpart.



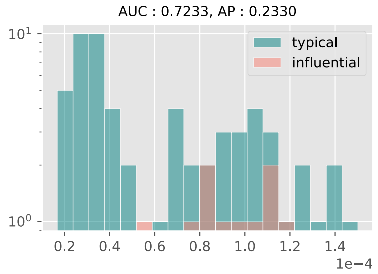
$\mathcal{I}_{\texttt{LA}}\ (\alpha = 0.01)$



$\mathcal{I}_{\texttt{LA}}\ (\alpha = 100)$



$\mathcal{I}_{\texttt{LA}}\ (R = 50)$



$\mathcal{I}\ (R = 50)$

- This performance drop is originated from singularity of Hessian for over-parameterized NNs.

Proposition (Hessian singularity for over-parameterized NNs). Let us assume a pre-trained parameter $\theta^* \in \mathbb{R}^P$ achieves zero training loss $L(S, \theta^*) = 0$ for squared error. Then, H has at least $P - NK$ zero-eigenvalues for NNs such that $NK < P$. Furthermore, if x is an input of training sample $z \in S$, then the following holds for the eigenvectors $\{u_i\}_{i=NK+1}^{P}$

$$g_z^\top u_i = \nabla_{\hat{y}}^\top \ell(z, \theta^*) \underbrace{\nabla_\theta^\top f(x, \theta^*) u_i}_{\mathbf{0}} = 0$$

The singularity does not affect linearized NNs, but it severely degrade original NNs.

MLILAB Machine Learning & Intelligence

Utilization of Geometric Ensemble

- Key Idea 2: Replace Laplace Approximation to Geometric Ensemble (GE)
  - Consequently, we modify the parameter distribution as follows:

$$\mathcal{I}_{\texttt{LA}}(z, z') := \mathbb{E}_{\psi\sim p_{\texttt{LA}}}\left[\Delta\ell_{\theta^*}(z,\psi)\cdot\Delta\ell_{\theta^*}(z',\psi)\right].$$

↓

$$\mathcal{I}_{\texttt{GEX}}(z, z') = \mathbb{E}_{\psi\sim p_{\texttt{GE}}}\left[\Delta\ell_{\theta^*}(z,\psi)\cdot\Delta\ell_{\theta^*}(z',\psi)\right]$$

Empirical distribution of Geometric Ensemble[42]
(Fine-tuning trajectory of SGD)

- By this modification, GEX
  1. can **capture the local geometry around θ^*** similar to LA,
  2. can avoiding **overestimating loss deviations caused by the singularity** of the Hessian.
     (∵ GE finds diverse & low loss checkpoints around θ^* [42]. Therefore, GE does not yield underfitting issues.)

MLILAB Machine Learning & Intelligence

Utilization of Geometric Ensemble

- Pseudocode of GEX

Algorithm 2 $\mathcal{I}_{\texttt{GEX}}$

```
Input: training data S, pre-trained parameter θ*, number of LA samples M, number of fine-tuning steps T,
two data samples z, z'

# Generating Geometric Ensemble (GE; [16])
for m = 1, ..., M (This computation can be parallelized for multiple devices) do
    Initialized the m-th checkpoint θ_m^0 ← θ* (or θ_{m-1}^T)
    for t = 1, ..., T do
        Apply stochastic optimization update (e.g., SGD with momentum): θ_m^t ← θ_m^{t-1}
    end for{End fine-tuning the m-th checkpoint θ_m^T}
end for{End generation of GE {θ_m^T}_{m=1}^M}

# Compute the non-linear IF approximation (10)
Î_GEX(z, z') ← Σ_{m=1}^M [Δℓ_θ*(z, θ_m^T) · Δℓ_θ*(z', θ_m^T)]

Output: Î_GEX(z, z')
```

Pure post-hoc implementation

Does not require forward-mode AD!
(Framework independent)

Empirical evaluations

- **Treatment of Noisy label detection**
  - GEX can distinguish mislabeled samples better than other influence-based methods on both synthetic and real-world label noise[19].
  - Otsu algorithm[18] is used to find a threshold between noisy and clean labels.
  - For relabeling, we follow the relabeling function in [20].
  - Metrics
    - Noisy label detection: Area under the ROC curve (AUC), Average Precision (AP)
    - Noisy label relabeling: Test accuracy after relabeling.

	Synthetic label noise				Real-world label noise			
	CIFAR-10		CIFAR-100		CIFAR-10-N		CIFAR-100-N	
Detection method	AUC	AP	AUC	AP	AUC	AP	AUC	AP
Deep-KNN	92.51 ± 0.19	69.93 ± 0.71	84.00 ± 0.14	40.17 ± 0.23	78.32 ± 0.19	72.60 ± 0.33	71.59 ± 0.21	59.76 ± 0.25
CL	57.60 ± 0.30	16.27 ± 0.20	84.16 ± 0.10	35.76 ± 0.50	75.94 ± 0.02	66.50 ± 0.09	69.49 ± 0.15	58.69 ± 0.23
F-score	73.34 ± 0.07	16.27 ± 0.09	59.18 ± 0.21	11.04 ± 0.05	69.39 ± 0.06	52.89 ± 0.06	68.95 ± 0.11	52.29 ± 0.14
EL2N	98.29 ± 0.03	95.82 ± 0.06	96.42 ± 0.05	73.82 ± 0.42	93.57 ± 0.17	91.26 ± 0.13	84.65 ± 0.08	77.26 ± 0.06
$\mathcal{I}_{\text{RandProj}}$	62.70 ± 0.19	17.90 ± 0.17	79.96 ± 0.32	26.25 ± 0.47	56.75 ± 0.38	45.61 ± 0.38	67.25 ± 0.09	54.14 ± 0.09
$\mathcal{I}_{\text{TracIn}}$	89.89	43.21	75.53	22.25	76,48	64.69	68.91	55.86
$\mathcal{I}_{\text{TracInRP}}$	89.56 ± 0.14	44.26 ± 0.37	74.99 ± 0.25	21.62 ± 0.26	77.24 ± 0.45	65.17 ± 0.68	69.04 ± 0.28	56.41 ± 0.31
$\mathcal{I}_{\text{Arnoldi}}$	61.64 ± 0.13	17.05 ± 0.18	77.20 ± 0.35	22.61 ± 0.42	56.83 ± 0.40	45.63 ± 0.40	66.57 ± 0.12	53.26 ± 0.11
$\mathcal{I}_{\text{GEX-lin}}$	64.11 ± 0.34	18.34 ± 0.36	76.06 ± 0.36	22.26 ± 0.47	56.88 ± 0.29	45.67 ± 0.33	65.68 ± 0.15	52.66 ± 0.13
$\mathcal{I}_{\text{GEX}}$	**99.74 ± 0.02**	**98.31 ± 0.06**	**99.33 ± 0.03**	**96.08 ± 0.12**	**96.20 ± 0.03**	**94.89 ± 0.04**	**89.76 ± 0.01**	**86.30 ± 0.01**

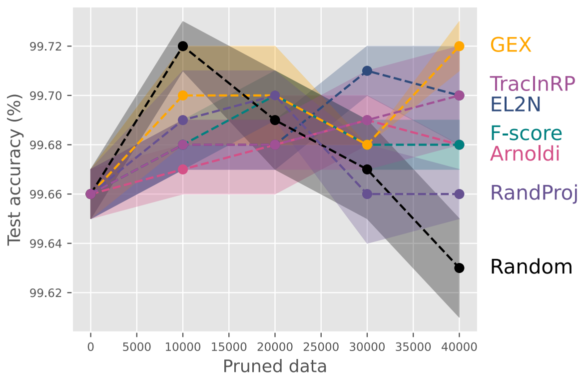
Results on noisy label detection

	Synthetic label noise		Real-world label noise	
	CIFAR-10	CIFAR-100	CIFAR-10-N	CIFAR-100-N
Clean label acc.	95.75 ± 0.06	79.08 ± 0.05	95.75 ± 0.06	79.08 ± 0.05
Noisy label acc.	90.94 ± 0.15	72.35 ± 0.17	68.63 ± 0.32	55.50 ± 0.09
Detection method	Relabeled acc.			
Deep-KNN	91.58 ± 0.10	66.12 ± 0.27	69.12 ± 0.25	50.03 ± 0.19
CL	91.11 ± 0.10	72.55 ± 0.13	30.52 ± 0.02	33.17 ± 0.02
F-score	78.94 ± 0.39	58.67 ± 0.18	53.50 ± 0.28	44.34 ± 0.21
EL2N	89.40 ± 0.10	61.72 ± 0.18	72.01 ± 0.51	47.58 ± 0.22
$\mathcal{I}_{\text{RandProj}}$	90.94 ± 0.09	72.42 ± 0.16	68.55 ± 0.17	55.47 ± 0.08
$\mathcal{I}_{\text{TracIn}}$	91.24	72.07	68.36	54.87
$\mathcal{I}_{\text{TracInRP}}$	90.82 ± 0.06	71.70 ± 0.15	68.12 ± 0.23	55.20 ± 0.06
$\mathcal{I}_{\text{Arnoldi}}$	91.10 ± 0.09	72.50 ± 0.08	68.67 ± 0.02	55.37 ± 0.14
$\mathcal{I}_{\text{GEX-lin}}$	91.04 ± 0.16	70.08 ± 0.12	68.44 ± 0.08	55.51 ± 0.21
$\mathcal{I}_{\text{GEX}}$	**93.54 ± 0.05**	**75.04 ± 0.10**	**73.94 ± 0.24**	**57.13 ± 0.10**

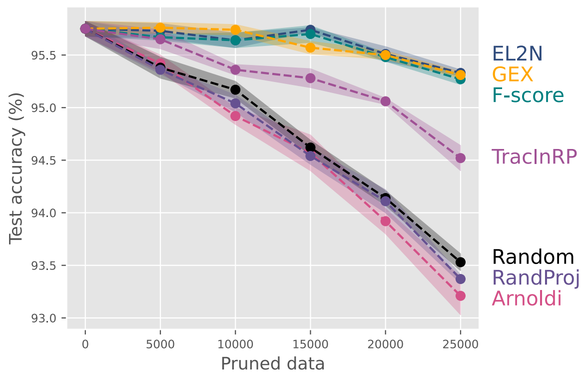
Results on noisy label relabeling

Empirical evaluations

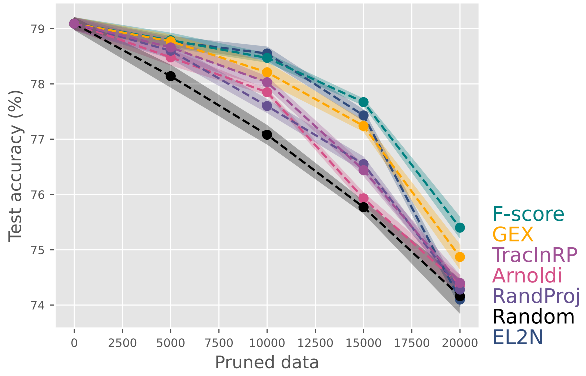
- **Dataset pruning**
  - GEX can detect low self-influence samples that can be pruned.
  - Note that among the influence-based methods, only GEX is comparable to the SOTA methods, F-score and EL2N.
  - Metrics
    - Test accuracy after pruning the fixed number of samples.



MNIST



CIFAR-10



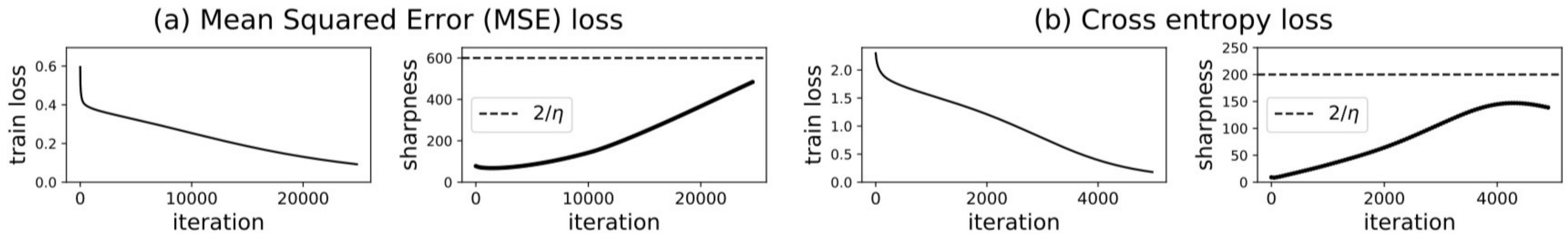
CIFAR-100

4. Discussion points

Conclusion and Future directions

- **Conclusion**
  - Bilinear form of influence approximation severely restrict the expressivity of influence distribution.
  - Removing this restriction can enhance the separability of influence and improve the performance of downstream tasks.
  - Furthermore, the proposed method can be implemented in purely post-hoc manner and does not require forward-mode AD.

- **Future directions: Understanding the evolution of Sharpness & Influence Function**
  - As shown in Theorem 4.1, IF is closely related to the Laplace Approximation.
  - On the other hand, [23] formalized the relationship between sharpness and Laplace Approximation through PAC-Bayes.
  - Although there is criticism that sharpness is not directly related to generalization[24], an accurate understanding of the evolution (dynamics) of sharpness will help clarify the relationship between sharpness and generalization.
  - Recent works on Edge-of-Stability (EOS)[25, 26, 27] would be a good starting point.



Loss & Sharpness in NNs[25]

Thank you !

Any Questions ?