Influence Estimation in Self-Supervised Learning

Reza Akbarian Bafghi[1*]
reza.akbarianbafghi@colorado.edu

Nidhin Harilal[1*]
nidhin.harilal@colorado.edu

Amit Kiran Rege[1*]
amit.kiranrege@colorado.edu

Maziar Raissi[3]
maziar.raissi@ucr.edu

Claire Monteleoni[1,2]
cmontel@colorado.edu

[1]University of Colorado Boulder, [2]INRIA Paris, [3]University of California, Riverside

Abstract

Self-supervised learning (SSL) has emerged as a key method for training powerful encoders on large-scale unlabeled data. However, recent research indicates that SSL encoders may over-rely on or even memorize many data points from their training set. While supervised learning benefits from tools like influence functions to identify such memorable data points, these methods do not effectively apply to SSL due to their reliance on labels. In this work, we introduce a new label-free definition of influence functions for SSL. Our implementation utilizes Eigenvalue-corrected Kronecker-Factored Approximate Curvature (EK-FAC) to efficiently estimate influence functions without requiring any retraining, making it applicable to any pre-trained SSL model to assess the effect of training examples on its model behavior. Our results suggest that the proposed method is informative about memorization that can be detrimental to SSL pre-training[2].

1 Introduction

Self-supervised learning (SSL) is a promising approach to extracting meaningful representations on vast amounts of unlabeled data by solving a surrogate objective, also known as a pretext task. In many recent SSL methods [8, 17, 13, 7, 5], such pretext tasks rely on joint-embedding architectures where multiple network branches aim to learn representations by maximizing agreement between differently augmented views of the same data example in the embedding space. However, recent research shows that SSL encoders may over-rely, or in the worst case, memorize aspects of their training data [23, 24]. This behavior not only raises privacy concerns, particularly when models are trained on datasets containing personal information, but it also demands defining a notion of the influence of data points on model behavior, indicating memorization.

In supervised learning, the concept of *Influence functions* [20] has proven valuable in assessing the impact of individual training examples on a model's behavior. These functions quantify how a model's predictions would change if a specific example were removed from the training data [11, 4, 14, 28]. Another work [11] discusses the link between memorization and the influence functions (See Sec 2.1). Extending this notion of influence to the SSL setting is crucial, as it could provide insights into the learning dynamics of pretrained encoders. However, this extension to SSL remains unexplored due to

[*]Equal contribution
[2]The implementation is available at https://github.com/cryptonymous9/Influence-SSL

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

its unique challenges, as existing definitions of influence functions often rely on labels [20] and, in some cases [11, 16], require supervised retraining of the model multiple times.

In this paper, we address these challenges by proposing a new definition of influence function in a label-free setting for SSL. We apply our method to popular SSL methods like SimCLR [8], VicReg [5] and DINO [7], to demonstrate our method's effectiveness in quantifying influence. Interestingly, when we retrain the network after removing the highest-influence examples identified by our method, we observe improved SSL pretraining. This suggests that our proposed definition effectively identifies instances of memorization that may be detrimental to SSL pre-training, thus offering a valuable tool for enhancing the performance and understanding of SSL models.

Related Work: The fundamental idea of quantifying influence based on the impact of individual training data points on model predictions has been extensively studied in supervised learning [2, 26, 6]. Feldman et al. [11] discusses link between this influence and data memorization. However, this exploration has been limited in self-supervised learning (SSL). Only two works, Deja Vu [23] and SSLMem [24], have addressed memorization in SSL, but both have limitations. Deja Vu assumes access to labeled data from the encoder's training distribution, and both methods require training multiple encoders on subsets of training data. Our work differs significantly by not requiring training of multiple networks, making it more efficient. We propose extending the widely accepted influence function definition of memorization to SSL, marking the first attempt to do so in this context.

2 Influence in a Label-Free Setting

2.1 Preliminaries and Problem Setup

In supervised learning, we consider a training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ and a model f_θ parameterized by θ, trained to minimize a loss function $\mathcal{L}(f_\theta(x), y)$. Feldman et al. [11] defines the concept of label memorization in terms of influence - which measures how much a model's prediction changes when a particular example is removed from the training set:

$$\mathcal{I}(x_i, y_i) = \mathbb{E}_{z \sim \mathcal{D} \setminus \{(x_i, y_i)\}}[f_{\theta(\mathcal{D})}(x_i) - f_{\theta(\mathcal{D} \setminus \{(x_i, y_i)\} \cup \{(x_i, z)\})}(x_i)] \tag{1}$$

Here, $\theta(\mathcal{D})$ represents the model parameters learned from dataset $\mathcal{D}$, and z is a random label drawn from the marginal label distribution of $\mathcal{D}$. In practice, directly computing this would involve retraining multiple networks on various data subsets, which is computationally expensive. For instance, Feldman [11] retrained 2,500 ResNet models to achieve this. However, on the bright side, we can use influence functions as an approximation to simplify the computation.

Influence function: Influence functions, introduced by Cook and Weisberg [3] and popularized in machine learning by Koh and Liang [20], provide a way to estimate the effect of individual training points on a model's predictions without actually retraining different networks. For a model minimizing empirical risk $R(\theta, \mathcal{D}) = \frac{1}{n} \sum_{i=1}^n \mathcal{L}(f_\theta(x_i), y_i)$, the influence of removing a training point (x_z, y_z) on the loss at a test point (x_t, y_t) is defined as:

$$\tilde{\mathcal{I}}(f_\theta, t, z) = -\nabla_\theta \mathcal{L}(f_\theta(x_t), y_t)^\top H_\theta^{-1} \nabla_\theta \mathcal{L}(f_\theta(x_z), y_z) \tag{2}$$

where $H_\theta = \frac{1}{n} \sum_{i=1}^n \nabla_\theta^2 \mathcal{L}(f_\theta(x_i), y_i)$ is the Hessian of the empirical risk (See Koh and Liang [20] for a proof). Feldman [11] describes memorization corresponds to the influence of example i on the accuracy on itself (or self-influence), i.e $\text{mem}(f_\theta, i) = \tilde{\mathcal{I}}(f_\theta, i, i)$.

2.2 Formalizing influence for SSL

We aim to generalize equation 2, which depends on the label y, to a fully label-free framework. Naturally, this requires modifying the function L, but the challenge is determining the appropriate modification. We propose to adapt influence functions to SSL based on the invariance-distinctiveness trade-off inherent in SSL objectives. Our approach quantifies a training point's influence by measuring how its removal affects the model's ability to match augmented views. This provides a principled method for understanding individual training examples' roles in SSL, bridging the gap between influence functions and self-supervised learning paradigms.

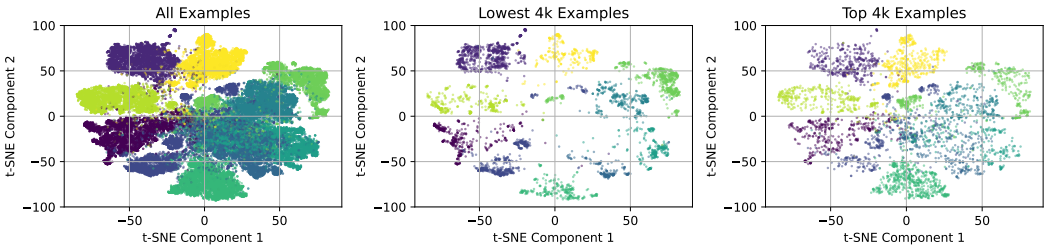


Figure 1: t-SNE projection of CIFAR-10 training images. The left plot shows all examples, while the middle and right plots display the lowest 4,000 and top 4,000 examples by influence score, respectively. Low-influence images are tightly clustered, while high-influence ones are more spread out. This clustering explains why removing low-influence images affects accuracy more in SSL.

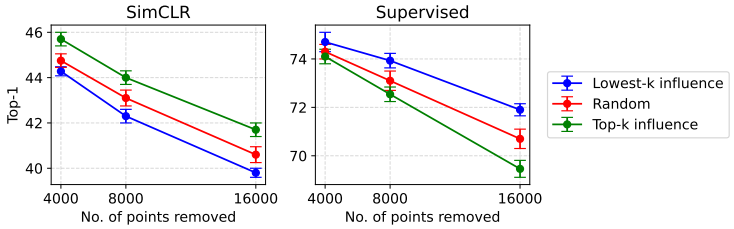


Figure 2: Accuracy on CIFAR-100 (y-axis) vs. number of removed samples (x-axis). Data points were removed using influence from pre-trained SimCLR, and supervised ResNet-18. The trend in performance is observed to be reversed in the supervised setting compared to SSL.

To this end, we introduce a new definition for the influence score $\mathcal{I}$ using a pre-trained self-supervised learning (SSL) model f_θ, applied to an unlabelled image x_i, as follows:

$$\mathcal{I}(f, i) = -\nabla_\theta \mathcal{L}(f_\theta(x_i), f_\theta(\hat{x}_i))^\top H_\theta^{-1} \nabla_\theta \mathcal{L}(f_\theta(x_i), f_\theta(\hat{x}_i)) \tag{3}$$

where L is defined as the cosine distance between an image x_i and its augmented counterpart $\hat{x}_i$, i.e., $L(t_i, t_j) = 1 - \frac{t_i \cdot t_j}{||t_i|| ||t_j||}$ and we use EK-FAC (See Appendix A) to efficiently compute H_θ^{-1}.

One might question how the choice of $\mathcal{L}$ as the cosine distance is justified, given that different SSL methods optimize empirical risk using varying loss functions $\tilde{\mathcal{L}}$. We argue that cosine distance is a valid and consistent measure of influence across most SSL methods. While distillation-based approaches like DINO [7] and BYOL [13] explicitly maximize cosine similarity, recent theoretical studies [19, 25] show that contrastive methods implicitly maximize cosine similarity. Moreover, recent work [27] demonstrates that Masked AutoEncoders (MAE) also implicitly align mask-induced positive pairs in a similar manner.

3 Experiments

We leverage various SSL frameworks, including the contrastive-based SimCLR [8], distillation-based DINO [7], and regularization-based VICReg [5], to ensure a broad evaluation of our method. While our ultimate goal is to scale the method to ImageNet-1k [10], due to its large size, we begin by demonstrating the effectiveness of our influence estimation on smaller, more manageable datasets like CIFAR-10 [21] and CIFAR-100 [21]. For additional implementation details, please see Appendix B.1.

In the subsequent sections, we describe the results of the experiments based on the proposed new definition of influence function adapted for SSL as described in section 2.2.

3.1 Marginal utility with low vs. high-influence examples

Given a pretrained SSL model, we first calculate the influence scores. In order to evaluate the marginal utility of influential examples, we removed the images with both the highest and lowest influence scores, then trained the model on the remaining dataset.

Is there a case of detrimental memorization in SSL? We explored the impact of removing high-influence data points in both self-supervised learning (SSL) and supervised learning using CIFAR-100. In the case of SimCLR, removing the top 16,000 most influential points led to a 2.3% improvement in linear evaluation accuracy compared to random removal, as shown in Figure 2, indicating the presence of detrimental memorization in SSL. A similar pattern was observed in VICReg, where performance improved by approximately 4% on the test set. These findings suggest that in SimCLR, the high-influence points have been leading to a form of *detrimental memorization* - perhaps the model was relying too heavily on specific image features that were easy to match across augmentations, rather than learning more generalizable representations.

An interesting contrast emerges when we use the highest influence scores obtained from SimCLR to remove examples in supervised training. Unlike in the self-supervised setting, where removing high-influence examples improves performance, removing these influential points in a supervised setting results in a performance drop compared to removing random examples (Figure 2). This divergence likely stems from the differing objectives between supervised and self-supervised learning. In self-supervised learning, learn representations that are invariant to augmentations but distinct between different images, and we hypothesize that removing highly influential examples may prevent the model from focusing too heavily on the overrepresented instances, leading to better generalization. However, in supervised learning, these high-influence examples may correspond to critical points that guide the model in learning discriminative features tied to the labels. Consequently, their removal disrupts the model's ability to accurately map inputs to labels, causing a decline in performance. This indicates that in supervised learning, influential examples are more essential for the task-specific learning process, whereas in self-supervised learning, their removal can help balance the representation learning. We also tested this on CIFAR-10, as detailed in Appendix B.4.

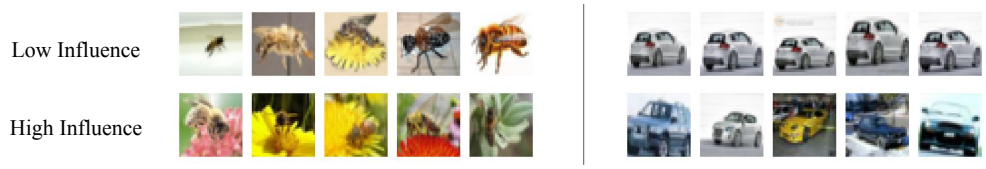
Embedding sparsity in low vs. high-influence examples: Figure 1 presents t-SNE visualizations of the model's embeddings, revealing distinct patterns for low- and high-influence examples. The 4,000 lowest-influence images cluster tightly, suggesting they occupy a homogeneous region in the feature space. In contrast, the 4,000 highest-influence images are more dispersed. This clustering difference offers insights into the model's utilization of these examples. Low-influence images likely contribute to well-represented, consistent regions in the feature space, supporting generalization without introducing significant variability. Their removal may affect model performance by eliminating a stable portion of the data. High-influence examples, being more diverse, may represent harder or unique samples. In SSL, removing these points seems to prevent overfitting to specific features, leading to better generalization. However, in supervised learning, these dispersed high-influence examples are likely crucial for capturing the variation needed to associate inputs with labels correctly. The spatial distribution of low- and high-influence examples in the embedding space thus highlights their distinct roles in model training and performance across different learning paradigms.

3.2 Visualization of influential examples

Figure 3 compares low and high-influence images in CIFAR-100 and CIFAR-10. High-influence examples are more diverse and complex, while low-influence examples are often similar, reflecting their tighter clusters in the t-SNE projections in Figure 1. Remarkably, removing high-influence examples enhances downstream classification performance, indicating that diverse examples may cause overfitting in self-supervised learning, while homogeneous low-influence examples support the learning of generalizable features. This highlights the effectiveness of our influence estimation method in optimizing training data selection for SSL, challenging traditional views on the role of diversity in representation learning.

4 Discussion and Conclusion

Our work extends influence functions to SSL, offering valuable insights into the dynamics of representation learning. Notably, we uncover a phenomenon where high-influence data points in SSL can contribute to detrimental memorization, demanding further exploration. Additionally, the divergence in trends between SSL and supervised learning calls for deeper investigation. We also examine sparsity patterns in influential examples, highlighting their connection to memorization. While our study is limited to smaller datasets and architectures, we believe future research can scale this approach to larger datasets and more complex models.



(a) CIFAR-100 (b) CIFAR-10

Figure 3: Images with the lowest (top) and highest (bottom) influence values for the "bee" (CIFAR-100) and "automobile" (CIFAR-10) classes. Harder examples (e.g., bees with flowers) appear in the bottom row, while duplicated cars appear in the top row.

References

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

[2] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International conference on machine learning*, pages 233–242. PMLR, 2017.

[3] A. C. Atkinson, R. D. Cook, and Sanford Weisberg. Residuals and influence in regression. *Biometrics*, 39(3):818–, 1983.

[4] Juhan Bae, Nathan Ng, Alston Lo, Marzyeh Ghassemi, and Roger B Grosse. If influence functions are the answer, then what is the question? *Advances in Neural Information Processing Systems*, 35:17953–17967, 2022.

[5] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

[6] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX security symposium (USENIX security 19)*, pages 267–284, 2019.

[7] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[9] Victor Guilherme Turrisi da Costa, Enrico Fini, Moin Nabi, Nicu Sebe, and Elisa Ricci. solo-learn: A library of self-supervised methods for visual representation learning. *Journal of Machine Learning Research*, 23(56):1–6, 2022.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, K. Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[11] Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959, 2020.

[12] Thomas George, César Laurent, Xavier Bouthillier, Nicolas Ballas, and Pascal Vincent. Fast approximate natural gradient descent in a kronecker factored eigenbasis. *Advances in Neural Information Processing Systems*, 31, 2018.

[13] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

[14] Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.

[15] Roger Baker Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, Evan Hubinger, Kamil.e Lukovsiut.e, Karina Nguyen, Nicholas Joseph, Sam McCandlish, Jared Kaplan, and Sam Bowman. Studying large language model generalization with influence functions. *ArXiv*, abs/2308.03296, 2023.

[16] Kelvin Guu, Albert Webson, Ellie Pavlick, Lucas Dixon, Ian Tenney, and Tolga Bolukbasi. Simfluence: Modeling the influence of individual training examples by simulating training runs. *arXiv preprint arXiv:2303.08114*, 2023.

[17] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[18] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015.

[19] Weiran Huang, Mingyang Yi, Xuyang Zhao, and Zihao Jiang. Towards the generalization of contrastive self-supervised learning. *arXiv preprint arXiv:2111.00743*, 2021.

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

[21] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[22] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 2408–2417. JMLR.org, 2015.

[23] Casey Meehan, Florian Bordes, Pascal Vincent, Kamalika Chaudhuri, and Chuan Guo. Do ssl models have déjà vu? a case of unintended memorization in self-supervised learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[24] Wenhao Wang, Muhammad Ahmad Kaleem, Adam Dziedzic, Michael Backes, Nicolas Papernot, and Franziska Boenisch. Memorization in self-supervised learning improves downstream generalization. *arXiv preprint arXiv:2401.12233*, 2024.

[25] Yifei Wang, Qi Zhang, Yisen Wang, Jiansheng Yang, and Zhouchen Lin. Chaos is a ladder: A new theoretical understanding of contrastive learning via augmentation overlap. *arXiv preprint arXiv:2203.13457*, 2022.

[26] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[27] Qi Zhang, Yifei Wang, and Yisen Wang. How mask matters: Towards theoretical understandings of masked autoencoders. *Advances in Neural Information Processing Systems*, 35:27127–27139, 2022.

[28] Rui Zhang and Shihua Zhang. Rethinking influence functions of neural networks in the over-parameterized regime. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 9082–9090, 2022.

Supplementary material

This document presents the materials that were excluded or summarized due to space limitations in the main text. It is organized as follows:

Appendix A provides additional information related to the method, especially covering more details on K-FAC approximation for computing the inverse Hessian H^{-1} additional results.

Appendix B covers the experimental configurations, the impact of different settings, and the distribution of influence scores in SSL.

A Computational Challenges with Influence Estimation

One of the main computational bottlenecks with estimating influence function is the estimation of Inverse Hessian Vector Product (IHVP). While most of the existing work uses iterative approximations [1], we instead use Eigenvalue-corrected Kronecker-Factored Approximate Curvature (EK-FAC) [12] to approximate the IHVP. The following section describes EK-FAC in more detail.

A.1 EK-FAC Approximation for Estimating IHVP

Computing the exact inverse Hessian H^{-1} for large neural networks is intractable, requiring $O(d^2)$ memory and $O(d^3)$ time for d parameters. The Kronecker-Factored Approximate Curvature (K-FAC) method [22] addresses this by approximating the Fisher information matrix F as:

$$F \approx \text{diag}(F_1, F_2, ..., F_L), \quad F_l \approx A_l \otimes G_l \tag{4}$$

where L is the number of layers, A_l is the second moment of activations, and G_l is the second moment of gradients for layer l, $\otimes$ is the Kronecker product. This allows efficient inversion:

$$F_l^{-1} \approx A_l^{-1} \otimes G_l^{-1} \tag{5}$$

For a layer with dimensions m and n, this reduces inversion cost from $O((mn)^3)$ to $O(m^3 + n^3)$. EK-FAC approximates the inverse-Hessian-vector product as:

$$H^{-1}v \approx (F + \lambda I)^{-1}v \tag{6}$$

where λ is a damping term.

EK-FAC [12] further improves this approximation by learning individual eigenvalues for the full Kronecker product, rather than just using the eigenvalues of the factors. It decomposes each F_l as:

$$F_l \approx (U_A \otimes U_G)\text{diag}(\lambda)(U_A \otimes U_G)^T \tag{7}$$

where U_A and U_G are eigenvectors of A_l and G_l respectively, and λ are learned eigenvalues. This allows for a more accurate approximation while maintaining computational efficiency.

The inverse-Hessian-vector product is then approximated as:

$$H^{-1}v \approx (U_A \otimes U_G)(\Lambda + \lambda I)^{-1}(U_A \otimes U_G)^T v \tag{8}$$

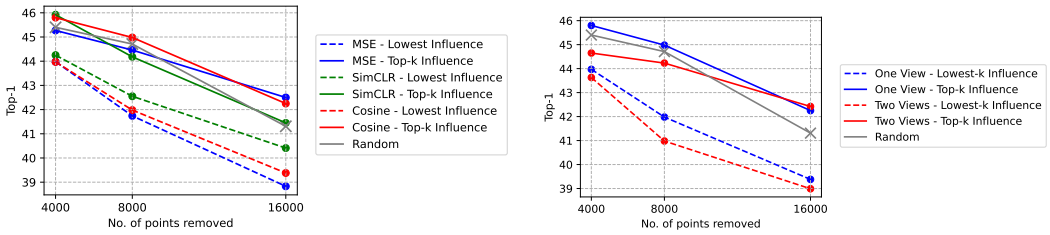
where Λ is the diagonal matrix of learned eigenvalues.

EK-FAC provides a better trade-off between approximation accuracy and computational efficiency compared to K-FAC, enabling more accurate influence function computations for large neural networks. It's worth noting that while EK-FAC provides improved accuracy over K-FAC, it introduces additional memory overhead for storing the eigendecompositions. This trade-off between accuracy and memory usage should be considered when applying EK-FAC to very large models.

B Experiments

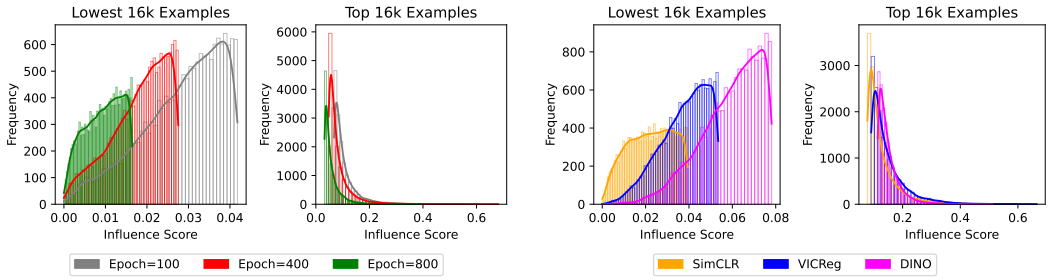
B.1 Configurations

We utilized the solo-learn library [9] for our training and adhered to the default augmentation settings for each joint embedding SSL algorithm. The SimCLR model was trained for 1000 epochs, with



(a) Different Methods (b) Different Augmentations

Figure 4: The x-axis shows the number of points removed from the training dataset, and y-axis shows the top-1 accuracy on CIFAR-100. The left plot compares different methods for calculating influence functions, while the right plot shows the effect of different data augmentation settings. Both plots demonstrate that removing low-influence points results in a sharper drop in accuracy compared to randomly removing points.



(a) Different Epochs (b) Different Models

Figure 5: Distribution of influence scores across training epochs for SimCLR on CIFAR-10. The x-axis represents the influence score, and the y-axis represents the frequency of examples. The leftward shift indicates a decrease in the number of high-influence examples.

checkpoints saved at epochs 100, 400, and 800. We used the exact hyperparameters from solo-learn for training SimCLR and VICReg, including the LARS optimizer and learning rates of 0.4 for SimCLR and 0.3 for VICReg. Influence scores were calculated using the kronfluence library [15]. We use ResNet-18 [18] as the backbone for all models.

B.2 The effect of different settings

Figure 4 illustrates the effect of removing images based on influence scores and its impact on model performance. The left subplot (Figure 4a) compares different methods for calculating influence functions, showing how removing the lowest-influence images leads to a significant drop in accuracy. We select the loss function and measurement methods, including Mean Squared Error, the SimCLR loss, and cosine distance, and compare them against a random baseline. The right subplot (Figure 4b) highlights the role of data augmentations, demonstrating that different augmentation strategies consistently exhibit similar trends: removing low-influence examples causes a more severe decline in top-1 accuracy than removing randomly selected images. One-view augmentation means that one input remains unchanged, while in two-view augmentation, both inputs are augmented. These results suggest that low-influence images, despite their subtle contribution to individual predictions, collectively play a critical role in maintaining overall model accuracy, particularly in SSL settings.

B.3 Influence distribution across SSL

We visualize the distribution of influence estimates over training epochs in Figure 5a and across different SSL frameworks in Figure 5b. Consistent with existing studies [11, 24], we observe that our influence estimates follow long-tailed distribution. We compare the distribution of examples with the both the highes and the lowest influence scores for SimCLR across epochs 100, 400, and 800 in Figure 5a. We observe a noticeable leftward shift as training progresses. This suggests that as the

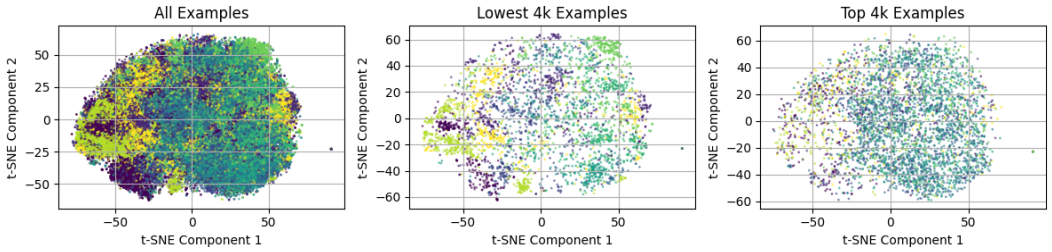


Figure 6: t-SNE projection of CIFAR-10 training images calculated with supervised loss, with clusters created through supervised training. The left plot shows all examples, while the middle and right plots display the lowest 4,000 and top 4,000 examples by influence score, respectively. Low-influence images form tightly clustered groups, whereas high-influence images are more dispersed."

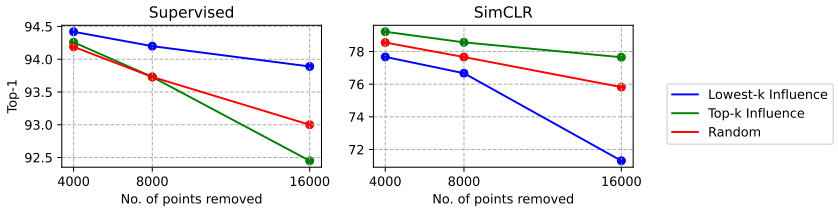


Figure 7: Accuracy on CIFAR-10 (y-axis) vs. number of removed samples (x-axis). Data points were removed using pre-trained SimCLR. Models were retrained for 100 epochs. The trend is also reversed in the supervised setting compared to SSL.

model trains for more epochs, the examples with high influence become less impactful. Over time, the most influential examples early in training may no longer be as critical, possibly due to the model aligning the features across different views of the data better. When comparing influence distributions across SSL frameworks, we observe SimCLR to have more low influential examples compared to VICReg and DINO, with DINO having the most high-influential examples.

B.4 Marginal utility of influence estimates on CIFAR-10

We repeat the experiment in 3.1 on CIFAR-10. The results in Figure 7 show that the trend differs between supervised and self-supervised methods, similar to our observations with CIFAR-100. In this experiment, we include only SimCLR.