# DATA VALUATION WITHOUT TRAINING OF A MODEL

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Many recent works on understanding deep learning try to quantify how much individual data instances influence the optimization and generalization of a model, either by analyzing the behavior of the model during training or by measuring the performance gap of the model when the instance is removed from the dataset. Such approaches reveal characteristics and importance of individual instances, which may provide useful information in diagnosing and improving deep learning. However, most of the existing works on data valuation require actual training of a model, which often demands high-computational cost. In this paper, we provide a training-free data valuation score, called *complexity-gap score*, which is a data-centric score to quantify the influence of individual instances in generalization of two-layer overparameterized neural networks. The proposed score can quantify irregularity of the instances and measure how much each data instance contributes in the total movement of the network parameters during training. We theoretically analyze and empirically demonstrate the effectiveness of the complexity-gap score in finding 'irregular or mislabeled' data instances, and also provide applications of the score in analyzing datasets and diagnosing training dynamics.

## 1 INTRODUCTION

Creation of large datasets has driven recent development of deep learning in diverse applications including computer vision (Krizhevsky et al., 2012; Dosovitskiy et al., 2021), natural language processing (Vaswani et al., 2017; Brown et al., 2020) and reinforcement learning (Mnih et al., 2015; Silver et al., 2016). To utilize the dataset in a more efficient and effective manner, many recent works have attempted to understand the role of *individual* data instances in training and generalization of neural networks. As an example, in (Ghorbani & Zou, 2019), a metric to quantify the contribution of each training instance in achieving a high test accuracy was analyzed under the assumption that not only the training data but also the test data is available. Jiang et al. (2021) defined a score to identify irregular examples that need to be memorized during training, in order for the model to accurately predict the class of the example. Swayamdipta et al. (2020), on the other hand, analyzed the characteristics of data instances with respect to their role in out-of-distribution generalizations.

All these previous methods for data valuation require *actual training* of a model to quantify the role of individual instances at the model. Thus, the valuation itself often requires high-computational cost, which may contradict some motivations of data valuation. For example, in (Ghorbani & Zou, 2019; Jiang et al., 2021), to examine the effect of individual data instances in training, one needs to train a model repeatedly while eliminating each instance or subsets of instances, which requires training of a model at least the number of times proportional to the number of training instances. In (Swayamdipta et al., 2020; Toneva et al., 2019), on the other hand, training dynamics–the behavior of a model on each instance throughout the training–is analyzed to categorize data instances, which also requires the training of a model with the full dataset. When the motivation for data valuation lies in pruning less important examples to save computational cost for training, the previous valuation methods might not be suitable, since they require the training of a model with the full dataset before one can figure out 'important' instances and possibly prune the rest of the examples.

In this paper, our main contribution is on defining a *training-free* data valuation score, which can be directly computed from data and can effectively quantify the impact of individual instances in optimization and generalization of neural networks. The proposed score, called *complexity-gap score*, measures the gap in *data complexity* where a certain data instance is removed from the full dataset. The data complexity measure was originally introduced in Arora et al. (2019) to quantify the com-

plexity of the full dataset, which was used in bounding the generalization error of overparameterized two-layer neural networks trained by gradient descent. Different from that work, where the complexity of the full dataset was main concern, our focus is on decomposing the effect of individual data instances in the training, and thus we newly introduce a complexity gap score (CG-score). We theoretically analyze and empirically demonstrate that the CG-score can quantify 'irregularity' of instances within each class, and thus can be used in identifying atypical examples, either due to the inherent irregularity of the instance or mislabeled classification. We also demonstrate that the proposed score has a close relation to 'learning difficult' of the instances by analyzing the training dynamics of data instances. Our key contributions are as below:

- *Training-free data valuation:* Different from previous methods for data valuation, most of which leverage the information from training itself, we provide a training-free data valuation score, CG-score, which is the data-centric score to quantify the effect of individual data instances in optimization and generalization of neural networks.
- *Geometric interpretation:* We provide theoretical analysis that the CG-score can measure *irregularity* of instances within each class, i.e., how much each instance represents the instances of the same class and how much it is different from those of other classes.
- *Effectiveness of the score*: We empirically demonstrate the effectiveness of the CG-score in data valuation. We show that pruning data instances with small CG-score do not significantly degrade the generalization capability of a model, e.g., for CIFAR-10 we can prune 40% of the data with only 1% of drop in test accuracy. Our scoring method is especially useful in data pruning, since different from other scores, which require the training with the full dataset, our method does not require any training of a model.
- *Application of the score:* We provide potential applications of the CG-score in analyzing datasets and training dynamics. We analyze the histograms of the CG-score for various datasets to demonstrate that the CG-score can measure irregularity of the instances. We also demonstrate that the instances with higher CG-score are 'difficult' examples, which are learned slowly by the models, by comparing the loss and test accuracy curves and the evolution of Neural Tangent Kernel (NTK) submatrices of lowest/highest-scoring groups.

**Related Works** Different from many existing works on deep learning analysis, which compute the impact of the training data as a whole, some recent works have focused on understanding the impact of individual data instances in training of neural networks. Ghorbani & Zou (2019) defined a method for data valuation, called Data Shapley, to evaluate the value of each data instance, by measuring the average gap in performances when an instance is held-out from any subsets of a given training data. Jiang et al. (2021) defined consistency score (C-score) of each instance by estimating the prediction accuracy of the instance attained by the model trained with the full dataset except the instance. Both Data Shapley and C-score require multiple trainings of a model to compute the scores. Another main stream of methods for data valuation uses the information from training dynamics to identify 'difficult' instances for classification, either due to irregularity or mislabeling, by measuring different forms of *confidence or stability* in the decision of the networks throughout the training (Toneva et al., 2019; Swayamdipta et al., 2020). Some recent work also uses the information from the network during training, by defining the computational difficulty of an instance as the number of hidden layers after which the networks' prediction coincides with the prediction at the output (Baldock et al., 2021). With application of robust learning, there also exist some works that quantify the difficulty of each instance by a 'margin' from the decision boundary, which can be measured by the amount of noise sufficient to alter the decision of the network when the noise is added to the instance (Zhang et al., 2021). All these previous methods are demonstrated to be effective in at least one or more applications of data valuation, including data pruning (Paul et al., 2021; Swayamdipta et al., 2020; Agarwal et al., 2022; Feldman & Zhang, 2020), importance-based weighted sampling (Chang et al., 2017; Koh & Liang, 2017), noise filtering (Li et al., 2020; Lee et al., 2019b; Kim et al., 2021), robust learning (Ren et al., 2018; Pleiss et al., 2020), or out-of-distribution generalizations (Swayamdipta et al., 2020). However, all these methods require the training of a model with the full dataset (at least for a few optimization steps) or possibly for as many times as the number of instances in the training dataset. Our data valuation method, on the other hand, is a data-centric method, which does not require training of a model. We demonstrate that with a simple scoring method, which can be efficiently calculated from data only, we can effectively analyze the impact of individual data instance in optimization and generalization of neural networks.

## 2 COMPLEXITY-GAP SCORE: DATA VALUATION WITHOUT TRAINING

In this section, we introduce a new data valuation score, called *complexity-gap score*, based on the analysis of overparameterized two-layer neural networks from Arora et al. (2019).

### 2.1 PRELIMINARIES: DATA COMPLEXITY MEASURE IN TWO-LAYER NEURAL NETWORKS

We first review the result from Arora et al. (2019), where a two-layer neural network trained by randomly initialized gradient descent is analyzed. Following the notations from Arora et al. (2019), we consider a two-layer ReLU activated neural network having $m$ neurons in the hidden layer, of which the output is $f_{\mathbf{W},\mathbf{a}}(\mathbf{x}) = \frac{1}{\sqrt{m}}\sum_{r=1}^{m} a_r \sigma(\mathbf{w}_r^\top \mathbf{x})$ where $\mathbf{x} \in \mathbb{R}^d$ is the input, $\mathbf{w}_1, \dots, \mathbf{w}_m \in \mathbb{R}^d$ are weight vectors in the first layer, $a_1, \dots, a_m \in \mathbb{R}$ are weights in the second layer, and $\sigma(x) = \max(0, x)$ is the ReLU activation function. We denote $\mathbf{W} = (\mathbf{w}_1, \dots, \mathbf{w}_m) \in \mathbb{R}^{d\times m}$ and $\mathbf{a} = (a_1, \dots, a_m)^\top \in \mathbb{R}^m$. It is assumed that the network parameters are randomly initialized as $\mathbf{w}_r(0) \sim \mathcal{N}(0, \kappa^2 I_{d\times d})$ and $a_r \sim \text{unif}(\{-1, 1\})$, $\forall r \in [m]$, where $\kappa \in (0, 1]$ is the size of random initialization. The second layer $\mathbf{a}$ is then fixed and only the first layer $\mathbf{W}$ is optimized through gradient descent (GD) to minimize the quadratic loss, $\Phi(\mathbf{W}) = \frac{1}{2}\sum_{i=1}^{n}(y_i - u_i)^2$ where $u_i = f_{\mathbf{W},\mathbf{a}}(\mathbf{x}_i)$ and $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ is the dataset drawn i.i.d. from an underlying distribution $\mathcal{D}$. The GD update rule can be written as $\mathbf{w}_r(k+1) - \mathbf{w}_r(k) = -\eta \frac{\partial \Phi(\mathbf{W}(k))}{\partial \mathbf{w}_r}$ where $\eta > 0$ is the learning rate. The output of the network for the input $\mathbf{x}_i$ at the $k$-th iteration is denoted by $u_i(k) = f_{\mathbf{W}(k),\mathbf{a}}(\mathbf{x}_i)$. For simplicity, it is assumed that $\|\mathbf{x}\|_2 = 1$ and $|y| \le 1$.

The data complexity measure governing the training of the two-layer ReLU activated neural network is defined in terms of the following *Gram matrix* $\mathbf{H}^\infty \in \mathbb{R}^{n\times n}$ associated with ReLU activation:

$$\mathbf{H}_{ij}^\infty = \mathbb{E}_{\boldsymbol{w}\sim\mathcal{N}(0, I_{d\times d})}\left[\mathbf{x}_i^\top \mathbf{x}_j \mathbb{1}\{\mathbf{w}^\top \mathbf{x}_i \ge 0, \mathbf{w}^\top \mathbf{x}_j \ge 0\}\right] = \frac{\mathbf{x}_i^\top \mathbf{x}_j (\pi - \arccos(\mathbf{x}_i^\top \mathbf{x}_j))}{2\pi}. \tag{1}$$

In Arora et al. (2019), a complexity measure of data was defined as $\mathbf{y}^\top (\mathbf{H}^\infty)^{-1}\mathbf{y}$ where $\mathbf{y} = (y_1, \dots, y_n)$, and it was shown that this measure bounds the total movement of all neurons in $\mathbf{W}$ from their random initialization. Moreover, the data complexity measure bounds the generalization error by restricting the Rademacher complexity of the resulting function class. In the following, we write the eigen-decomposition of $\mathbf{H}^\infty$ as $\mathbf{H}^\infty = \sum_{i=1}^n \lambda_i \mathbf{v}_i \mathbf{v}_i^\top$ where $\lambda_i$'s are ordered such that $\lambda_1 \ge \lambda_2 \ge \dots \lambda_n$. Further, assuming $\lambda_n \ge \lambda_0 > 0$, we can write $(\mathbf{H}^\infty)^{-1} = \sum_{i=1}^n (\lambda_i)^{-1} \mathbf{v}_i \mathbf{v}_i^\top$.

**Theorem 1** (Informal version of (Arora et al., 2019))**.** *Assume that $\lambda_{\min}(\mathbf{H}^\infty) = \lambda_n \ge \lambda_0 > 0$. For sufficiently large width $m$, sufficiently small learning rate $\eta > 0$ and sufficiently small random initialization $\kappa > 0$, with probability at least $1 - \delta$ over the random initialization, we have*

*a) Bound in loss* : $\|\mathbf{y} - \mathbf{u}(k)\|_2 = \sqrt{\sum_{i=1}^{n}(1 - \eta\lambda_i)^{2k}(\mathbf{v}_i^\top \mathbf{y})^2} +$ *small constant*,

*b) Bound in total movement of neurons:* $\|\mathbf{W}(k) - \mathbf{W}(0)\|_F \le \sqrt{\mathbf{y}^\top (\mathbf{H}^\infty)^{-1}\mathbf{y}} +$ *small constant*,

*c) Bound in population loss:* $\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[l(f_{\mathbf{W}(k),\mathbf{a}}(\mathbf{x}), y)] \le \sqrt{\frac{\mathbf{y}^\top (\mathbf{H}^\infty)^{-1}\mathbf{y}}{n}} + O\left(\sqrt{\frac{\log \frac{n}{\lambda_0 \delta}}{n}}\right)$,

*where a) and b) hold for all iteration $k \ge 0$ of GD, and c) holds for $k \ge \Omega\left(1/(\eta\lambda_0)\log(n/\delta)\right)$.*

The above theorem shows that if the label vector $\mathbf{y}$ is aligned with top eigenvectors of $\mathbf{H}^\infty$, i.e., $(\mathbf{v}_i^\top \mathbf{y})$ is large for large $\lambda_i$, then the loss decreases quickly and the total movement of neurons from their random initialization is small, which implies a small generalization error. Thus, the data complexity measure $\mathbf{y}^\top (\mathbf{H}^\infty)^{-1}\mathbf{y}$ captures the complexity of data governing both the optimization and generalization of the overparameterized two-layer neural networks. However, this quantity captures the complexity of the overall data. To decompose the individual data instances, in the next section, we newly define a *complexity-gap score*.

### 2.2 COMPLEXITY-GAP SCORE AND TRAINING DYNAMICS

We define the *complexity-gap score* (CG-score) of $(\mathbf{x}_i, y_i)$ as the difference between the data complexity measure when $(\mathbf{x}_i, y_i)$ is removed from a given dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$:

$$\mathrm{CG}(i) = \mathbf{y}^\top (\mathbf{H}^\infty)^{-1}\mathbf{y} - \mathbf{y}_{-i}^\top (\mathbf{H}_{-i}^\infty)^{-1}\mathbf{y}_{-i} \tag{2}$$

where $\mathbf{y}_{-i}$ is the label vector except the $i$-th sample point and $\mathbf{H}_{-i}^\infty$ is the $(n-1)\times(n-1)$ matrix obtained by removing the $i$-th row and column of $\mathbf{H}^\infty$.

We first emphasize that the proposed score can be easily calculated from given data without the need of training neural networks, as opposed to other data valuation scores requiring either a trained neural network or statistics calculated from training dynamics. Yet, the proposed score captures two important properties on the training and generalization of data instance, implied by Theorem 1:

1. An instance $(\mathbf{x}_i, y_i)$ with a large CG-score is a 'difficult' example, in the sense that removing it from the dataset reduces the generalization error bound by a large amount, which implies that the dataset without $(\mathbf{x}_i, y_i)$ is much easier to be learned and generalized.
2. An instance $(\mathbf{x}_i, y_i)$ with a larger CG-score contributes more on the optimization and drives the total movement of neurons by a larger amount, measured by $\|\mathbf{W}(k) - \mathbf{W}(0)\|_F$.

We next discuss the computational complexity of calculating the CG-score. To calculate $\{\mathrm{CG}(i)\}_{i=1}^n$, we need to have the inversion of matrices $\mathbf{H}^\infty$ and $\mathbf{H}_{-i}^\infty$ for all $i \in [n]$, which requires $O(n^4)$ complexity when we use general $O(n^3)$-complexity algorithm for the matrix inversion. By using Schur complement, however, we can reduce this complexity to $O(n^3)$. Without loss of generality, we can assume $i = n$. Denote $\mathbf{H}^\infty$ and $(\mathbf{H}^\infty)^{-1}$ by

$$\mathbf{H}^\infty = \begin{pmatrix} \mathbf{H}_{n-1}^\infty & \mathbf{g}_i \\ \mathbf{g}_i^\top & c_i \end{pmatrix}, \quad (\mathbf{H}^\infty)^{-1} = \begin{pmatrix} (\mathbf{H}^\infty)_{n-1}^{-1} & \mathbf{h}_i \\ \mathbf{h}_i^\top & d_i \end{pmatrix}, \tag{3}$$

where $\mathbf{H}_{n-1}^\infty, (\mathbf{H}^\infty)_{n-1}^{-1} \in \mathbb{R}^{(n-1)\times(n-1)}$, $\mathbf{g}_i, \mathbf{h}_i \in \mathbb{R}^{n-1}$ and $c_i, d_i \in \mathbb{R}$. From $\mathbf{H}^\infty(\mathbf{H}^\infty)^{-1} = \boldsymbol{I}_n$, we have $\mathbf{g}_i^\top (\mathbf{H}^\infty)_{n-1}^{-1} + c_i\mathbf{h}_i^\top = 0$, i.e., $\mathbf{h}_i^\top = -c_i^{-1}\mathbf{g}_i^\top (\mathbf{H}^\infty)_{n-1}^{-1}$.

By Shur complement, $(\mathbf{H}_{-i}^\infty)^{-1}$, which is equal to $(\mathbf{H}_{n-1}^\infty)^{-1}$ for $i = n$, can be calculated as

$$(\mathbf{H}_{-i}^\infty)^{-1} = (\mathbf{H}^\infty)_{n-1}^{-1} - d_i^{-1}\mathbf{h}_i\mathbf{h}_i^\top. \tag{4}$$

Since we have

$$\begin{aligned} \mathbf{y}^\top (\mathbf{H}^\infty)^{-1}\mathbf{y} &= \mathbf{y}_{-i}^\top (\mathbf{H}^\infty)_{n-1}^{-1}\mathbf{y}_{-i} + y_i\mathbf{h}_i^\top\mathbf{y}_{-i} + y_i\mathbf{y}_{-i}^\top\mathbf{h}_i + y_i^2 d_i, \\ \mathbf{y}_{-i}^\top (\mathbf{H}_{-i}^\infty)^{-1}\mathbf{y}_{-i} &= \mathbf{y}_{-i}^\top (\mathbf{H}^\infty)_{n-1}^{-1}\mathbf{y}_{-i} - d_i^{-1}(\mathbf{y}_{-i}^\top\mathbf{h}_i)^2, \end{aligned} \tag{5}$$

the CG-score, $\mathrm{CG}(i)$ in equation 2, can be calculated by

$$\mathrm{CG}(i) = d_i^{-1}(\mathbf{y}_{-i}^\top\mathbf{h}_i)^2 + 2y_i(\mathbf{y}_{-i}^\top\mathbf{h}_i) + y_i^2 d_i = \left((\mathbf{y}_{-i}^\top\mathbf{h}_i)/\sqrt{d_i} + y_i\sqrt{d_i}\right)^2. \tag{6}$$

Thus, $\mathrm{CG}(i)$ can be calculated by the $n$-th column $(\mathbf{h}_i, d_i)$ of $(\mathbf{H}^\infty)^{-1}$, without the need of calculating $(\mathbf{H}_{-i}^\infty)^{-1}$ when $i = n$. The case for general $i \neq n$ can also be readily solved by permuting the $i$-th rwo and column of $\mathbf{H}^\infty$ into the last positions.

**Correlation to other scores** We show relation between our score and other data valuation scores that require the training of neural networks. Toneva et al. (2019) define 'the forgetting score' for each training example as the number of times during training the decision of that sample switches from a correct one to incorrect one. Paul et al. (2021), on the other hand, suggest the GraNd score, which is the expected loss gradient norm $\mathbb{E}[\|\nabla_{\mathbf{W}(k)} l(u(k), y)\|]$, to bound the contribution of each training example to the decrease of loss on any other example over a single gradient step. The GraNd score is further approximated (under some assumptions) by the EL2N score, defined to be $\mathbb{E}[|y - u(k)|]$ where $u(k)$ is the output of the neural network for the sample $(x, y)$ at the $k$-th step. Since $|y - u(k)|$, if rescaled, is an upper bound on 0–1 loss, $\sum_k |y - u(k)|$ upper bounds forgetting score after rescaling. Thus, an example with a high forgetting score will also have a high GraND score and high EL2N score averaged over multiple time steps. We next relate our complexity-gap score to $\sum_k (y - u(k))$, and thus to all the three previous scores defined using training dynamics.

Table 1: Spearman's rank correlation between CG-score (CG'-score) and other data valuation scores.

| Datasets | Correlation btwn. CG-score and | | | Correlation btwn. CG'-score and | | |
|---|---|---|---|---|---|---|
| | C-score | Forgetting | EL2N | C-score | Forgetting | EL2N |
| CIFAR-10 | 0.557 | 0.432 | 0.365 | 0.115 | 0.110 | 0.136 |
| CIFAR-100 | 0.529 | 0.289 | 0.356 | 0.243 | 0.090 | 0.177 |

Arora et al. (2019) show that for the overparameterized two-layer networks trained by GD, the gap between the label vector and the network output at the step $k$ can be approximated as $\mathbf{y} - \mathbf{u}(k) \approx (I - \eta \mathbf{H}^{\infty})^k \mathbf{y}$. By summing both sides over $k$, we get $\sum_{k=0}^{\infty} \mathbf{y} - \mathbf{u}(k) \approx \sum_{k=0}^{\infty} (I - \eta \mathbf{H}^{\infty})^k \mathbf{y} = \frac{1}{\eta}(\mathbf{H}^{\infty})^{-1}\mathbf{y}$. Without loss of generality, consider $i = n$. Then, the accumulated difference between $y_i$ and $u_i(k)$ over $k$ can be approximated as

$$\sum_{k=0}^{\infty}(y_i - u_i(k)) \approx \left(\mathbf{h}_i^\top \mathbf{y}_{-i} + y_i d_i\right)/\eta = \left(\mathbf{y}_{-i}^\top \mathbf{h}_i/\sqrt{d_i} + y_i\sqrt{d_i}\right)\sqrt{d_i}/\eta. \tag{7}$$

Note that both the right-hand side of equation 7 and $\mathrm{CG}(i)$ in equation 6 depend on the term $\left(\mathbf{y}_{-i}^\top \mathbf{h}_i/\sqrt{d_i} + y_i\sqrt{d_i}\right)$. Thus, we can expect that $\mathrm{CG}(i)$ will be correlated with the scores related to training dynamics, including the forgetting score, GraND score and EL2N score. Different from those scores, our score can be directly calculated from the data without training of a model.

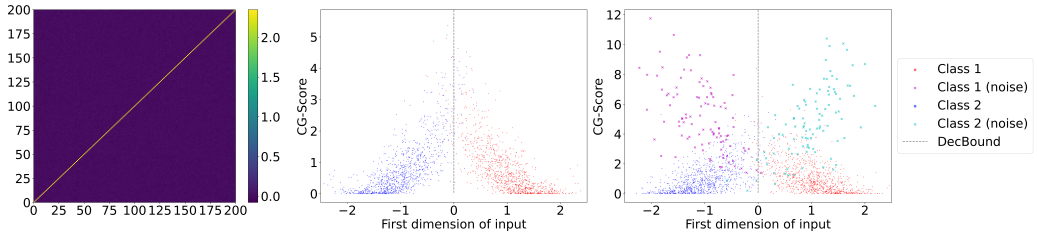
**Inversion of $\mathbf{H}^{\infty}$ is an effective step** In the definition of $\mathrm{CG}(i)$ in equation 2, we use the inverse of $\mathbf{H}^{\infty}$ to measure the alignment of the eigenvectors of $\mathbf{H}^{\infty}$ with the label vector $\mathbf{y}$. One could suggest another score that directly uses $\mathbf{H}^{\infty}$ instead of $(\mathbf{H}^{\infty})^{-1}$, which can save the computation for inversion. However, we argue that the score calculated by using $(\mathbf{H}^{\infty})^{-1}$ includes more information than the score calculated by $\mathbf{H}^{\infty}$ due to the reason described below. Let us define $\mathrm{CG}'(i) := \mathbf{y}^\top \mathbf{H}^{\infty}\mathbf{y} - \mathbf{y}_{-i}^\top \mathbf{H}_{-i}^{\infty}\mathbf{y}_{-i}$. Without loss of generality, assume $i = n$. Then, $\mathrm{CG}'(i) = 2y_i(\mathbf{g}_i^\top \mathbf{y}_{-i}) + y_i^2 c_i$ where $\mathbf{g}_i = (\mathbf{H}^{\infty})_{1:(n-1),n}$ and $c_i = H_{n,n}^{\infty}$ as defined in equation 3. Since $c_i = 1/2$ for all $i \in [n]$ from the definition of $\mathbf{H}^{\infty}$ in equation 1 and $y_i^2 = 1$ for $y_i = \pm 1$, we have $\mathrm{CG}'(i) = 2(y_i(\mathbf{H}^{\infty}\mathbf{y})_i - 1/2) + 1/2$. By using the approximation $\mathbf{y} - \mathbf{u}(k) \approx (I - \eta\mathbf{H}^{\infty})^k\mathbf{y}$ from Arora et al. (2019), we have $\mathbf{y} - \mathbf{u}(1) \approx \mathbf{y} - \eta\mathbf{H}^{\infty}\mathbf{y}$ when $k = 1$, which implies $\mathbf{u}(1) \approx \eta\mathbf{H}^{\infty}\mathbf{y}$. Thus, $\mathrm{CG}'(i) = \frac{2}{\eta}y_i u_i(1) + 1/2$. Note that an instance $(\mathbf{x}_i, y_i)$ has a large $\mathrm{CG}'(i)$ if $y_i u_i(1)$ is large, i.e., the network output $u_i(1)$ after 1-time step has the same sign as the targeted label $y_i \in \{-1, 1\}$ and has a large magnitude. Thus, $\mathrm{CG}'(i)$ measures how fast the training instance can be learned by the neural network. Different from $\mathrm{CG}'(i)$, our original score $\mathrm{CG}(i)$ is correlated with the accumulated error between $y_i$ and $u_i(k)$ averaged over the training as shown in equation 7. Thus, our original score $\mathrm{CG}(i)$, using the inverse of $\mathbf{H}^{\infty}$, reflects the statistics averaged over the whole training steps.

In Table 1, we compare the Spearman'a rank correlation between CG-score/CG'-score and other data valuation scores including C-score (Jiang et al., 2021), forgetting score (Toneva et al., 2019) and EL2N score (Paul et al., 2021) for CIFAR-10/100 datasets[1]. We can observe that CG-score has higher correlations with the previous scores compared to those of CG'-score. In the rest of this paper, we focus on the CG-score for our data valuation score.

### 2.3 Geometric interpretation of the complexity-gap score

We next provide geometric interpretation for the CG-score. For the sake of simplicity, we consider binary dataset with $\frac{n}{2}$ samples having $y_i = 1$ and $\frac{n}{2}$ samples having $y_i = -1$. We further assume that $\mathbb{E}[H_{ij}^{\infty}] = p$ if $y_i = y_j$ and $\mathbb{E}[H_{ij}^{\infty}] = q$ if $y_i \neq y_j$ for some $|p| > |q|$. Note that the diagonal entires $H_{ii}^{\infty} = 1/2$ for all $i \in [n]$. Thus, $\mathbb{E}[\mathbf{H}^{\infty}]$ can be decomposed as $\mathbb{E}[\mathbf{H}^{\infty}] = \left(\frac{1}{2} - p\right)\boldsymbol{I}_n + \boldsymbol{S}$ where $\boldsymbol{S}$ is a block matrix with $\boldsymbol{S} = \begin{pmatrix} p & q \\ q & p \end{pmatrix} \otimes \boldsymbol{I}_{n/2}$, and the resulting eigenvalues of $\mathbb{E}[\mathbf{H}^{\infty}]$ are $\frac{(p+q)n}{2} + \left(\frac{1}{2} - p\right)$, $\frac{(p-q)n}{2} + \left(\frac{1}{2} - p\right)$ and $\left(\frac{1}{2} - p\right)$ with multiplicity $(n-2)$. When $p = \Theta(q)$ and $p = o(1/n)$, the matrix $\mathbb{E}[\mathbf{H}^{\infty}] \approx (1/2)\boldsymbol{I}_n$. From the representations of $\mathbf{H}^{\infty}$ and $(\mathbf{H}^{\infty})^{-1}$ in

[1]The way we calculate the CG-score for multi-label public datasets is explained in Appendix §A.




(a) Heat map of $(\mathbf{H}^{\infty})^{-1}_{n-1}$ (b) CG-score (clean) (c) CG-score (10% label noise)

Figure 1: (a) Heat map of $(\mathbf{H}^{\infty})^{-1}_{n-1}$ for 200 indices. We plot 200 indices only for clear visualization. (b) Scatter graph of CG-Score for two groups of samples from 3000-D Gaussian distributions having the same mean except the first dimension, where class 1 has mean +1 (red) and class 2 has mean -1 (blue). Samples near the boundary ($x_1 = 0$) tend to have higher CG-score. (c) Same plot as (b) with 10% label noise. Samples with label noise (marked by plus symbol) tend to have higher CG-score.

equation 3 and the implied relation $\mathbf{h}_i^\top = -c_i^{-1}\mathbf{g}_i^\top(\mathbf{H}^\infty)^{-1}_{n-1}$, assuming $\mathbf{H}^\infty \approx (1/2)\boldsymbol{I}_n$ and using $c_i = 1/2$, we can write $\mathbf{h}_i^\top = -4\mathbf{g}_i^\top$. By using this approximation, our CG-score in equation 6 can be approximated as

$$\mathrm{CG}(i) = d_i^{-1}(\mathbf{y}_{-i}^\top \mathbf{h}_i)^2 + 2y_i(\mathbf{y}_{-i}^\top \mathbf{h}_i) + y_i^2 d_i \approx 8(y_i(\mathbf{y}_{-i}^\top \mathbf{g}_i))^2 - 8y_i(\mathbf{y}_{-i}^\top \mathbf{g}_i) + 2 \tag{8}$$

since $d_i \approx 2$ and $y_i^2 = 1$. From this approximation, we can first notice that $-8y_i(\mathbf{y}_{-i}^\top \mathbf{g}_i)$ is the main term that determines the order of $\{\mathrm{CG}(i)\}_{i=1}^n$, since $y_i(\mathbf{y}_{-i}^\top \mathbf{g}_i) = \sum_{j\in\{[n]:y_i=y_j\}} H_{ij}^\infty - \sum_{j\in\{[n]:y_i\neq y_j\}} H_{ij}^\infty$, and thus $\mathbb{E}[y_i(\mathbf{y}_{-i}^\top \mathbf{g}_i)] = \frac{(p-q)n}{2} = o(1)$, assuming $p = \Theta(q)$ and $p = o(1/n)$, which is much larger than the first term, proportional to $(y_i(\mathbf{y}_{-i}^\top \mathbf{g}_i))^2$. Note that $y_i(\mathbf{y}_{-i}^\top \mathbf{g}_i)$ measures the gap between the summation of $H_{ij}^\infty$ over the samples of the same class as $(\mathbf{x}_i, y_i)$ and that over the samples of the different class from $(\mathbf{x}_i, y_i)$. Since $H_{ij}^\infty = \frac{\mathbf{x}_i^\top \mathbf{x}_j(\pi - \arccos(\mathbf{x}_i^\top \mathbf{x}_j))}{2\pi}$, $y_i(\mathbf{y}_{-i}^\top \mathbf{g}_i)$ measures the gap between the average similarities of $\mathbf{x}_i$ with other samples $\{\mathbf{x}_j\}$ of the same class $y_j = y_i$ and that with samples of the different class $y_j \neq y_i$, where the similarity is measured by the cosine between the two samples $(\mathbf{x}_i, \mathbf{x}_j)$ multiplied by the chance that ReLU is activated for both the samples. Thus, we can expect two important trends regarding the CG-score:

1. Regular vs. irregular samples: If a sample $(\mathbf{x}_i, y_i)$ is a 'regular' sample representing the class $y_i$ in the sense that it has a large similarity with the other samples of the same class, much larger than the similarity with the samples from the different class, then the sample will have a large $y_i(\mathbf{y}_{-i}^\top \mathbf{g}_i)$, resulting in a lower CG-score due to the minus sign. On the other hand, if a sample $(\mathbf{x}_i, y_i)$ is 'irregular' in the sense that it does not represent the class and has a small similarity with the samples of the same class, then $y_i(\mathbf{y}_{-i}^\top \mathbf{g}_i)$ will be small, resulting in a higher CG-score.
2. Clean label vs. noisy label: A sample with label noise tend to have a large CG-score since $y_i(\mathbf{y}_{-i}^\top \mathbf{g}_i)$ is negative for a sample with label noise, while it is positive for a sample with clean label.

We empirically demonstrate the above two trends by a controlled experiment. Consider two 3000-dimensional Gaussian distributions with a fixed covariance matrix $0.25I_{3000}$ and different means $(1, 0, \ldots 0)$ and $(-1, 0, \ldots, 0)$, representing class +1 and -1, respectively. We generate 1000 samples from each class. We first check whether the structural assumptions on $\mathbf{H}^\infty$ hold with this synthetic dataset. In Fig. 1a, we observe that $(\mathbf{H}^\infty)^{-1}_{n-1}$ can be well approximated by $2I_{n-1}$, and thus the approximation for $\mathrm{CG}(i)$ in equation 8 may hold well. We then examine the correlation between the CG-score and the sample value at the first dimension. In Fig. 1b, we can observe that instances near the decision boundary $x_1 = 0$ have higher CG-scores compared to those located further from the boundary. This shows that irregular samples, e.g., samples near the boundary of the two classes, indeed have higher CG-scores. We further modify the generated samples by randomly flipping the labels of 10% samples and we re-calculate the CG-scores of all the samples. As shown in Fig. 1c, the samples with label noise tend to have higher CG-scores, agreeing with our intuition.

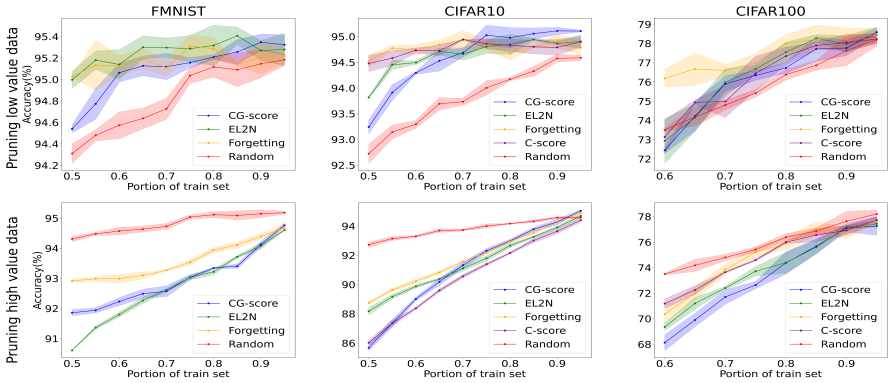



Figure 2: Pruning experiments with three datasets FMNIST (left), CIFAR-10 (middle) and CIFAR-100 (right). The test accuracy of the model trained with different portions ($x$-axis) of subset data when the low-scoring examples are removed first (top row) / the high-scoring examples are removed first (bottom row). Only CG-Score does not require any training of the model to calculate the score for examples, but it achieves competitive performances compared to other state-of-the-art data valuation scores and significantly outperforms the performance of random baseline. When low-scoring examples are removed first, CG-score maintains the test accuracy up to significant removing portions; when high-scoring examples are removed first, the test accuracy drops most rapidly for CG-score since the examples with high CG-score are essential in generalization of the model.

## 3 Data Valuation through Complexity Gap Score

In this section, we show diverse applications of the CG-score in data valuation for real datasets.

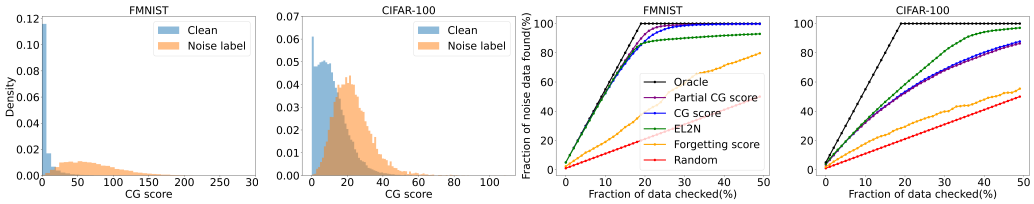
### 3.1 Data pruning experiments

To evaluate the ability of the CG-score in identifying important examples, we design data pruning experiments, similar to those in Ghorbani & Zou (2019); Paul et al. (2021). We evaluate our score on three public datasets, FMNIST, CIFAR-10/100 and train ResNet networks (He et al., 2016) of different depths, ResNet18 for FMNIST and CIFAR-10 and ResNet50 for CIFAR-100 dataset, respectively. As baseline methods for data valuation, we use three state-of-the-art scores, C-score (Jiang et al., 2021), EL2N (Paul et al., 2021), and Forgetting score (Toneva et al., 2019), all of which require training of models for computation and thus are dependent on the models at which the scores are calculated. On the contrary, our score is a data-centric measure and independent on the model. More details on the baselines and experiments are summarized in Appendix §B.

In Figure 2, the first row shows the test accuracy of the model trained with different subset sizes, when low-scoring (regular) examples are removed first, and the second row shows the similar result but when high-scoring (irregular) examples are removed first. We report the mean of the result after three independent runs, and the shaded regions indicate the standard deviation. The red-curve (random) is the result when randomly ordered examples are used. We can observe that when removing low-scoring examples first, networks maintain the test accuracy (with less than 1% drop) compared to the case of training with the full dataset up to significant removing portions, for example, 40% for CIFAR-10 and 20% for CIFAR-100 with the CG-score. Especially, our CG-score, which does not require any training of a model, can achieve competitive performances as the other baseline methods and it also significantly outperforms the random baseline. When we remove the high-scoring examples first, the test accuracy drops most rapidly for the CG-score curve, implying that examples with the high CG-score are essential part of the data governing the generalization of the model.

### 3.2 Detecting label noise

In Section 2.3, we analyzed the CG-score and showed that the examples with label noise tend to have higher CG-score. We next empirically investigate the ability of the CG-score in detecting label noise by artificially corrupting 20% of instances in FMNIST and CIFAR-100 with random label noise. We




(a) Density of CG-score for clean vs. label-noise (b) Fraction of detected label noise

Figure 3: (a) Density of CG-score for clean (80%) vs. label-noise (20%) examples. Examples with label noise tend to have higher CG-score. For FMNIST (left) dataset, the CG-score histograms betwen clean and label-noise groups are better separated than those for CIFAR-100 (right). (b) Fraction of label noise ($y$-axis) included in the examined portion ($x$-axis) for 20% label noise. CG-score (blue) achieves better noise detectability for FMNIST than for CIFAR-100.

first examine the distribution of the CG-scores for each dataset after the corruption in Figure 3a. We can observe that for both datasets, the examples with corrupted labels (orange) tend to have higher CG-scores than those with clean labels (blue). For a relatively simpler dataset, FMNIST (left), the CG-score histograms can be more clearly separable between the clean and noisy groups, compared to a more complex dataset, CIFAR-100 (right). With this observation, we can anticipate that the CG-score can have a better detectability of label noise for relatively simpler datasets.

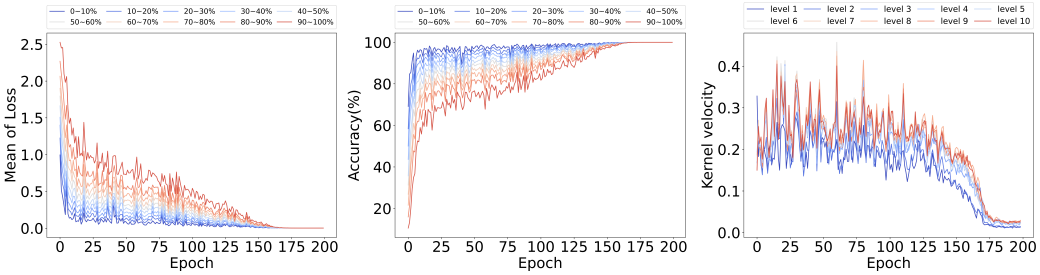
We next evaluate the noise detectability of the CG-score by checking a varying portion of the examples ordered by the CG-score (highest first) in Figure 3b. The curves indicate the fraction of noisy examples included in the examined subset. In this experiment, we compare our score with two other scores, Forgetting score and EL2N, as well as a random baseline and the oracle. We can observe that the CG-score achieves the best performance, near that of the oracle, for FMINST, and competitive performances for CIFAR-100. In the plot, we also compare the performance of the CG-score with that of 'Partial CG-score,' which is a new score defined by a sub-term of the CG-score. In the CG-score in equation 6, we have three terms, but only the second term $2y_i(\mathbf{y}_{-i}^\top \mathbf{h}_i)$ uses the label information of the data $(\mathbf{x}_i, y_i)$. Thus, we examined the ability of this term only, defined as the 'Partial CG-score', in detecting the label noise, and found that the CG-score and partial CG-score have similar performances in detecting label noise, which implies that the label-noise detectability of the CG-score mainly comes from the term $2y_i(\mathbf{y}_{-i}^\top \mathbf{h}_i)$.

## 4 Complexity-Gap Score and Deep Learning Phenomena

In this section, we empirically demonstrate the relation between the CG-score and the 'learning difficulty' of each data instance for CIFAR-10 dataset trained in ResNet18.

### 4.1 Complexity gap score captures learning difficulty

It has been widely observed that there exists an order of instances in which a model learns to classify the data correctly and this order is robust within model types (Wu et al., 2021). Many previous scoring functions use this type of observation and define the 'difficulty' of a given instance based on the speed at which the model's prediction converges (Toneva et al., 2019; Swayamdipta et al., 2020). Our data-centric CG-score is originally defined based on the gap in the generalization error bounds as in equation 2, but it also reflects the convergence speed of the instance, as analyzed in equation 7. We empirically demonstrate this relation by analyzing the training dynamics of CIFAR-10 dataset in ResNet18 network. We first sort the data instances in ascending order using the CG-score, and divide the data into 10 equal-sized subgroups. We then measure the mean of loss and training accuracy for the 10 subgroups as training progresses. Fig. 4a and Fig. 4b show the mean of loss and the training accuracy throughout the training for the 10 subgroups, respectively, where the blue color represents the low-scoring groups and the red color represents the high-scoring groups. We can observe that the mean loss and accuracy converge faster for low-scoring groups, while it takes longer to converge for high-scoring groups. This indicates that the CG-score is highly correlated with the 'difficulty' of an example measured by the learning speed at a model.




(a) Loss mean (b) Accuracy (c) Kernel velocity

Figure 4: Training dynamics measured for each subset of CIFAR-10 examples, grouped by the CG-score. The line color varies over groups, where the blue lines include low-scoring examples and the red lines include high-scoring groups. (a) and (b) Mean loss and accuracy for subgroups of CIFAR-10 data, sorted by the CG-score, trained on ResNet18. The mean loss and accuracy converge faster for low-scoring groups, while they converge slowly for high-scoring groups. (c) Kernel velocity of CIFAR-10 examples grouped by the CG-score. The Kernel evolves with a higher velocity for high-scoring groups throughout the training.

### 4.2 Data sample driving movement of neurons

We next investigate the relation between the CG-score and the evolution velocity of the data-dependent Neural Tangent Kernel (NTK) (Fort et al., 2020). NTK has been widely used to approximate the evolution of an infinite-width deep learning model via linearization around initial weights, when the network is trained by gradient descent with a sufficiently small learning rate (Jacot et al., 2018). The main idea is that in the limit of an infinite width, the network parameters do not move very far from its initialization throughout the training, so that the learning process can be approximated by a linear process along the tangent space to the manifold of the model's function class at the initialization (Lee et al., 2019a). However, as observed in Fort et al. (2020), for a finite-width network, the tangent kernel is not constant but it rather rapidly changes over time, especially at the beginning of the training. To quantify this change, Fort et al. (2020) addressed the data-dependent Kernel Gram matrix, which is the Gram matrix of the logit Jacobian, and defined the Kernel velocity as the cosine distance between two NTK Gram matrices before and after one epoch of training. In Paul et al. (2021), the Kernel velocity was used to evaluate the subgroups of data instances to figure out the subgroup driving the learning and the change in the NTK feature space. We conduct similar experiments for subgroups of data, divided according to our CG-score. Fig. 4c shows the Kernel velocities for 10 different subgroups of CIFAR-10 data trained in ResNet18. Each group is composed of 125 consecutive instances from each level, where the level is defined by dividing the full dataset, sorted in ascending order by the CG-score, into 10 groups. The higher level (red) is composed of instances having higher CG-scores, while the lower level (blue) is composed of instances having lower CG-scores. We can observe that the samples with high CG-score (red) maintains higher Kernel velocity throughout the training, which means that NTK Gram matrix evolves by a larger amount for the samples of high CG-score. Thus, we can hypothesize that the instances with high CG-score are 'difficult' examples the network may struggle to optimize and try to fit throughout the training.

## 5 Discussion

We proposed the CG-score, a data-centric valuation score, to quantify the effect of individual data instances in optimization and generalization of overparameterized two-layer neural networks trained by gradient descent. We theoretically and empirically demonstrated that the CG-score can identify 'irregular' instances within each class, and can be used as a score to select instances essential for generalization of a model or in filtering instances with label noise. We also showed the close relation between the CG-score and learning difficulty of instances by analyzing training dynamics. Interesting open problems related to the CG-score include 1) providing theoretical justification of the score for more general deep neural networks and 2) improving the score by modifying the definition in terms of 'features' of data instances.

## REFERENCES

Chirag Agarwal, Daniel D'souza, and Sara Hooker. Estimating example difficulty using variance of gradients. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, 2019.

Robert Baldock, Hartmut Maennel, and Behnam Neyshabur. Deep learning through the lens of example difficulty. *Advances in Neural Information Processing Systems*, 2021.

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. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. Active bias: Training more accurate neural networks by emphasizing high variance samples. In *Advances in Neural Information Processing Systems*, 2017.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.

Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M. Roy, and Surya Ganguli. Deep learning versus kernel learning: An empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 2018.

Ziheng Jiang, Chiyuan Zhang, Kunal Talwar, and Michael C Mozer. Characterizing structural regularities of labeled data in overparameterized models. In *International Conference on Machine Learning*, 2021.

Taehyeon Kim, Jongwoo Ko, Sangwook Cho, JinHwan Choi, and Se-Young Yun. FINE samples for learning with noisy labels. In *Advances in Neural Information Processing Systems*, 2021.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 2019a.

Kimin Lee, Sukmin Yun, Kibok Lee, Honglak Lee, Bo Li, and Jinwoo Shin. Robust inference via generative classifiers for handling noisy labels. In *Proceedings of the 36th International Conference on Machine Learning*, 2019b.

Junnan Li, Richard Socher, and Steven C.H. Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. In *International Conference on Learning Representations*, 2020.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 2019.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. *Advances in Neural Information Processing Systems*, 2021.

Geoff Pleiss, Tianyi Zhang, Ethan Elenberg, and Kilian Q Weinberger. Identifying mislabeled data using the area under the margin ranking. In *Advances in Neural Information Processing Systems*, 2020.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. An empirical study of example forgetting during deep neural network learning. In *7th International Conference on Learning Representations*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Xiaoxia Wu, Ethan Dyer, and Behnam Neyshabur. When do curricula work? In *9th International Conference on Learning Representations*, 2021.

Jingfeng Zhang, Jianing Zhu, Gang Niu, Bo Han, Masashi Sugiyama, and Mohan S. Kankanhalli. Geometry-aware instance-reweighted adversarial training. In *9th International Conference on Learning Representations*, 2021.

# A EXTENSIONS TO MULTI-CLASS COMPLEXITY GAP SCORE

In this section, we explain the details of how we calculated the CG-scores for multi-label public datasets.

## A.1 CALCULATION OF CG-SCORE FOR MULTI-LABEL DATASETS

In defining the CG-score, we assumed the binary datasets $y \in \{\pm 1\}$ with inputs having a fixed norm $\|\mathbf{x}\|_2 = 1$. To calculate the CG-score for multi-label ($k$-class) public datasets, we normalize all the inputs to have $\|\mathbf{x}\|_2 = 1$. We then calculate the CG-score for examples of each class $j \in [k]$, assuming that all the examples from class $j$ have label $+1$ and the rest of the examples from any other classes have label $-1$. In detail, let $\mathbf{y} = (y_1, y_2, ..., y_n) \in \{1, 2, \dots k\}^n$ be the label vector for $n$ data instances and, without loss of generality, assume that $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_l$ belong to class 1, i.e. $y_1, y_2, ..., y_l = 1$. Then, to calculate the CG-score for $\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_l$, we generate the gram matrix $\mathbf{H}^\infty$ with $(\mathbf{x}_1, 1), (\mathbf{x}_2, 1), \dots, (\mathbf{x}_l, 1), (\mathbf{x}_{l+1}, -1), (\mathbf{x}_{l+2}, -1), \dots, (\mathbf{x}_n, -1)$ and calculate the CG-scores of $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_l$ as the two-label case.

## A.2 STOCHASTIC METHOD TO CALCULATE CG-SCORES

Since the calculation of the CG-score requires the inversion of $n$-dimensional matrix $\mathbf{H}^\infty$ where $n$ is the number of total samples, it would demand expensive memory and computational cost. To lower the complexity, we sub-sample the examples with class $-1$ so that the ratio between class +1 and class -1 is reduced from $1 : (k-1)$ to $1 : 3$ for MNIST and FMINST, $1 : 4$ for CIFAR-10 and CIFAR-100. We repeat this process ten times by randomly sampling examples of label $-1$ and then average out the calculated CG-score of each example from class $+1$.

## A.3 JUSTIFICATION OF STOCHASTIC METHOD TO CALCULATE CG-SCORES

To justify the stochastic method in calculating the CG-scores, we conduct an experiment to check whether the CG-scores calculated by the stochastic method converges well to the true CG-scores utilizing the full dataset. We created a subset of the CIFAR-10 dataset, the Small-CIFAR-10 dataset, which consists of 1000 instances for each label (a total of 10000 instances). Then, we compared the CG-score calculated by 10000x10000 Gram matrix $\mathbf{H}^\infty$ of the full dataset (true CG-score) with the CG-score calculated by the stochastic method, where the stochastic CG-score for the instances from a class (class 1) are calculated by subsampling the samples from any other classes (class -1) with the ratio between class 1 and -1 equal to $1 : 1$ (size 2000x2000), $1 : 2$ (size 3000x3000), $1 : 3$ (size 4000x4000) and $1 : 4$ (size 5000x5000) instead of $1 : 9$. We repeat this process multiple times by randomly sampling examples of label $-1$ and then average out the calculated CG-score of each example from class $+1$. In Figure 5, we plot the rank correlation between the true CG-score and the stochastic CG-score for each ratio (different colors) as the number of random sampling increases. We can observe that the rank correlation converges to a certain number as the number of random sampling increases, and the amount of correlation increases as the size of the matrix (the number of instances included in defining $\mathbf{H}^\infty$) increases.

# B IMPLEMENTATION DETAILS

## B.1 TRAINING DETAILS

In Section 3 and 4, we evaluate our score on three public datasets, FMNIST, CIFAR-10/100, by training ResNet networks (He et al., 2016) of different depths. ResNet18 is used for FMNIST and CIFAR-10 dataset and ResNet50 is used for CIFAR-100 dataset. Implementation of the ResNet is based on the ResNet network in torchvision (Paszke et al., 2019). Since FMNIST and CIFAR images are smaller than Imagenet (Deng et al., 2009) images, we replace the front parts of the ResNet (convolution layer with 7x7 kernel and 2x2 stride, max pooling layer with 3x3 kernel and 2x2 stride) with a single convolution layer with 3x3 kernel and 1x1 stride for small size image. The details on hyperparameters and optimization methods used in training are summarized in the Table 2.

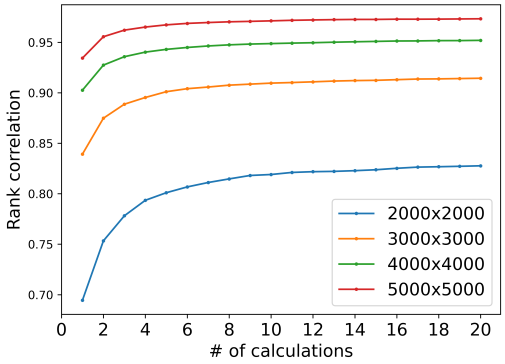



Figure 5: Rank correlation between the CG-score with full dataset and CG-score calculated by subsampling the dataset (stochastic CG-score) as the number of calculations (random sampling) increases. Stochastic CG-scores are calculated with 2000 (cyan), 3000 (orange), 4000 (green), and 5000 (red) instances, when the full data includes 10000 instances (1000 for each class).

Table 2: Details for the experiments used in the training of the dataset.

| | FMNIST | CIFAR10 | CIFAR100 |
|---|---|---|---|
| Architecture | ResNet18 | ResNet18 | ResNet50 |
| Batch size | 128 | 128 | 128 |
| Epochs | 100 | 200 | 200 |
| Initial Learning Rate | 0.02 | 0.05 | 0.1 |
| Weight decay | 5e-4 | 5e-4 | 5e-4 |
| Optimizer | SGD with momentum 0.9 | | |
| Learning Rate Scheduler | Cosine annealing schedule (Loshchilov & Hutter, 2017) | | |
| Data Augmentation | Normalize by dataset's mean, variance<br>Random Zero Padded Cropping (4 pixels on all sides)<br>Random left-right filpping (probability 0.5) | | |

### B.2 Experimental details

**Baseline scores for data valuation** We use three state-of-the-art scores, C-score (Jiang et al., 2021), EL2N (Paul et al., 2021), and Forgetting score (Toneva et al., 2019) as baselines with which our CG-score is compared. We use pre-calculated C-score for CIFAR-10 and CIFAR-100 from Jiang et al. (2021) and calculate EL2N and Forgetting score by averaging the score across five independent training using the full dataset. We use the same network architectures to calculate EL2N and Forgetting score: ResNet18 for FMNIST and CIFAR-10 dataset and ResNet50 for CIFAR-100 dataset. Detailed definitions of the scores are as follows:

- Consistency score (C-score): C-score of each instance is calculated by estimating the prediction accuracy of the instance attained by the model trained with the full dataset except the instance.

$$\text{C-score}(\mathbf{x}_i, y_i) = \mathbb{E}_n\left[\hat{\mathbb{E}}^r_{S\sim\{(\mathbf{x}_j,y_j)\}_{j=1}^n}\left[\mathbb{P}(f(\mathbf{x}_i; S\backslash\{(\mathbf{x}_i, y_i)\}) = y_i)\right]\right], \tag{9}$$

  where $f(\mathbf{x}_i; S)$ is trained network using subset $S$, and $\hat{\mathbb{E}}^r$ denotes empirical averaging with $r$ i.i.d. samples of such subsets.
- Error L2-Norm (EL2N): The EL2N score of a training sample $(\mathbf{x}_i, y_i)$ is defined to be $\mathbb{E}[\|f(\mathbf{W}(t), \mathbf{x}_i) - y_i\|_2]$ where $f(\mathbf{W}(t), \mathbf{x})$ is the output of the neural network for the sample $(\mathbf{x}, y)$ at the $t$-th step.
- Forgetting score: Forgetting score is defined as the number of times during training (until time step $T$) the decision of that sample switches from a correct one to an incorrect one:

Forgetting$(\mathbf{x}_i, y_i)$ is defined as

$$\sum_{k=2}^{T} \mathbb{1}\{\arg\max f(\mathbf{W}(t-1), \mathbf{x}_i) = y_i\}(1 - \mathbb{1}\{\arg\max f(\mathbf{W}(t), \mathbf{x}_i) = y_i\}). \tag{10}$$

**Data pruning experiment** We report the mean of the results after three independent runs. The shaded regions indicate the standard deviation. We acquire each result by training a network using a dataset pruned by specified portions, where the training instances are ordered by each score. We compute the number of iterations at which all data can be used in one epoch, and use the same number of iterations in all pruning experiments for a fair comparison. When we prune the dataset, we remove the instances from each class by the same amount, so as to preserve the original proportion between classes. As will be shown in Section E, the distribution of scores is different among classes, so an imbalance problem between classes may occur if the data pruning is performed without considering the portion of classes within the dataset.

## C Details of kernel velocity

We calculate the kernel velocity of 10 groups of instances, where each group is composed of 125 samples. The following is how we construct each group: Sort CIFAR-10 examples in ascending order by the CG-score, divide the examples into 10 groups, and select 125 consecutive samples from the beginning of each group.

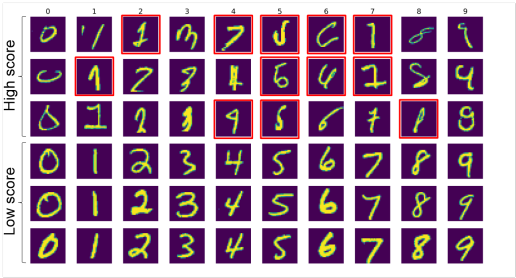
We calculate the NTK kernel velocity as described in Paul et al. (2021): Let C be the number of classes. Let $f_t^{(c)}(\mathbf{x}_i)$ be the $c$-th logit value for input $\mathbf{x}_i$ at the $t$-epoch. When $\mathbf{W}(t)$ is the parameters of the model, the $c$-th logit gradient at input $\mathbf{x}_i$ is $\psi_t^{(c)}(\mathbf{x}_i) = \nabla_{\mathbf{W}(t)} f_t^{(c)}(x_i) \in \mathbb{R}^N$. Then, the data-dependent NTK submatrix of a group of $m$ samples $S := \{\mathbf{x}_{a_1}, \ldots, \mathbf{x}_{a_m}\}$ is defined as $K_t(S) = \Psi_t(S)\Psi_t(S)^\top$, where $\Psi_t(S) \in \mathbb{R}^{mC \times N}$ is constructed by placing $\{\psi_t^{(c)}(\mathbf{x}_{a_i})\}_{c\in[C], i\in[m]}$ in rows of $\Psi_t(S)$. The kernel velocity is defined as

$$v_t(S) = 1 - \frac{\langle K_t(S), K_{t+1}(S)\rangle}{\|K_t(S)\| \, \|K_{t+1}(S)\|}. \tag{11}$$

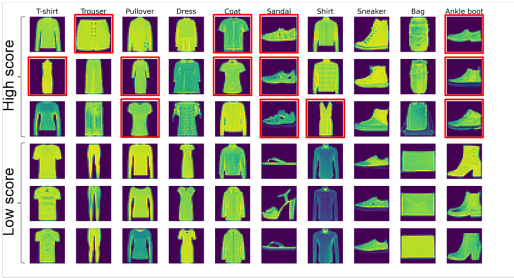
Lastly, we provide some explanations of what the kernel velocity measures if we define it for finite-width ReLU activated 2-layer neural network. Remind that our analysis uses the Gram matrix $\mathbf{H}^\infty$ defined in equation 1, which is derived for an overparameterized ReLU activated 2-layer neural network. We can generalize the definition of the Gram matrix assuming a finite-width network, similar to $K_t(S)$, as follows. The output of the network is $f_{\mathbf{W},\mathbf{a}}(\mathbf{x}) = \frac{1}{\sqrt{m}} \sum_{r=1}^m a_r \sigma(\mathbf{w}_r^\top \mathbf{x})$, and the gradient with respect to $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_m) \in \mathbb{R}^{d\times m}$ is $\nabla_{\mathbf{W}} f(\mathbf{x}_i) = [\nabla_{\mathbf{w}_1} f(\mathbf{x}_i), \nabla_{\mathbf{w}_2} f(\mathbf{x}_i), \ldots, \nabla_{\mathbf{w}_m} f(\mathbf{x}_i)] = \frac{\mathbf{x}_i^\top}{\sqrt{m}}[a_1 \mathbb{1}\{\mathbf{w}_1^\top \mathbf{x}_i \geq 0\}, a_2 \mathbb{1}\{\mathbf{w}_2^\top \mathbf{x}_i \geq 0\}, \ldots, a_m \mathbb{1}\{\mathbf{w}_m^\top \mathbf{x}_i \geq 0\}]$. Denoting the network parameters at the $t$-th step as $\mathbf{W}(t)$, we can define the Gram matrix $\mathbf{H}_t$ as $(\mathbf{H}_t)_{ij} = \nabla_{\mathbf{W}(t)} f(\mathbf{x}_i) \nabla_{\mathbf{W}(t)} f(\mathbf{x}_j)^\top = \mathbf{x}_i^\top \mathbf{x}_j \frac{1}{m} \sum_{r=1}^m \mathbb{1}\{\mathbf{w}_r^\top \mathbf{x}_i \geq 0, \mathbf{w}_r^\top \mathbf{x}_j \geq 0\}$, which is proportion to the number of neurons in the hidden layer activated both for $\mathbf{x}_i$ and $\mathbf{x}_j$ at the epoch. We can define the kernel velocity similar to equation 11 by calculating $\mathbf{H}_t$ for a subset of data, and replacing $K_t(S)$ by $\mathbf{H}_t(S)$. For this case, the high kernel velocity implies that $\mathbf{H}_t$ differs much from $\mathbf{H}_{t+1}$, i.e., the portion of neurons activated for pairs of instances changes rapidly during training.

## D Visualization of examples sorted by CG-score

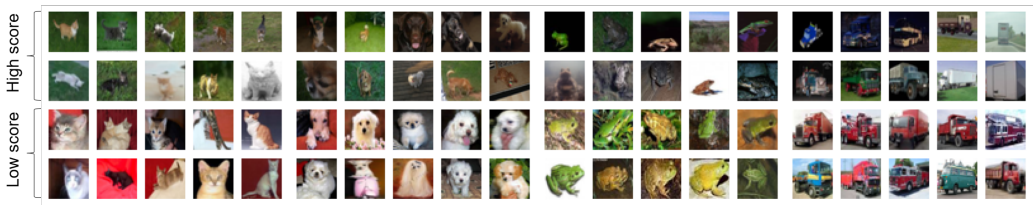
In Fig. 6, we show examples of MNIST (Fig. 6a) and FMNIST (Fig. 6b) images sorted by CG-score. The top/bottom three rows show the top-/bottom-ranked examples, respectively. We can observe that the examples with the lowest CG-score are regular examples representing each class and they look similar to each other, while the examples with the highest scores are irregular and they look different among themselves, among which we can identify (possibly) mislabeled instances, marked with red rectangles. Similarly, in CIFAR-10 images (Fig. 6c), we observe that low-scoring




(a) Examples of MNIST dataset




(b) Examples of FMNIST dataset




(c) Examples of CIFAR-10 dataset (Cat, dog, frog, and truck)

Figure 6: Examples sorted by CG-score. In (a) and (b), top-3 rows / bottom-3 rows display top-ranked / bottom-ranked examples, respectively. Among top-ranked examples, (possibly) mislabeled examples are marked by red rectangles. In (c), top-2 rows / bottom-2 rows display top-ranked / bottom-ranked examples, respectively.

examples (bottom two rows) are regular ones sharing similar features representing the class while high-scoring examples (top two rows) are irregular ones.

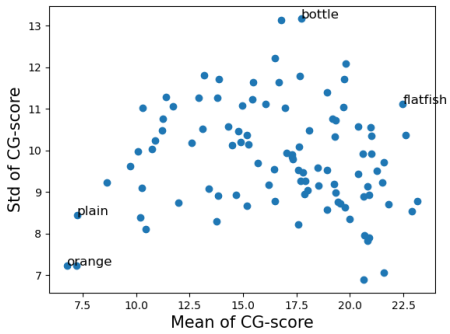
Fig. 7 shows the examples of CIFAR-100 dataset. Fig. 7a shows the means and standard variations of 100 classes in CIFAR-100 dataset. Among 100 classes, 'orange' and 'plain' classes have small CG-score means and variances, while 'bottle' and 'flatfish' classes have larger CG-score means and variances. We display ten top-/bottom-ranked examples of these four classes, with the histograms of the scores in Fig. 7b.

## E SCORE DISTRIBUTION FOR PUBLIC DATASETS

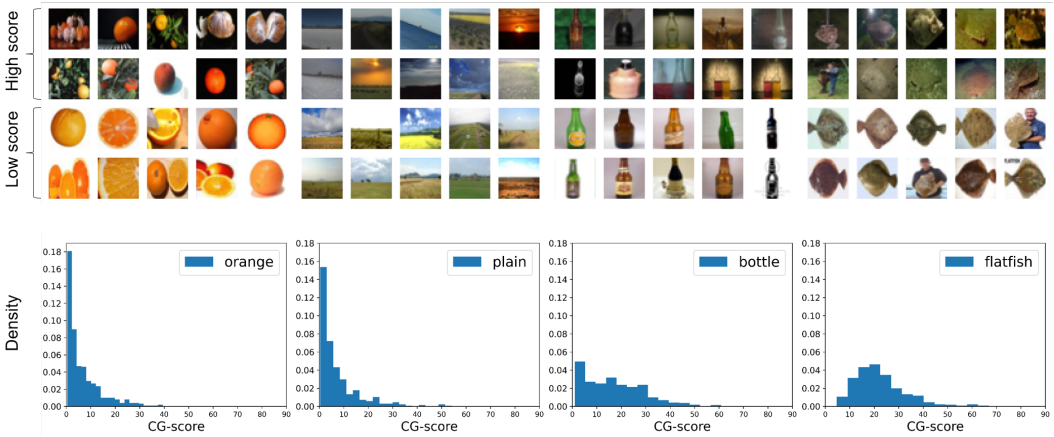
In Fig. 8, we compare the CG-score distributions of three public datasets, FMNIST, CIFAR-10 and CIFAR-100 before and after we artificially corrupt 20% of instances with random label noise. The top row shows the distributions before the corruption, and the bottom row shows the distributions after the corruption, where the orange color displays the distribution of the CG-score for 20% instances with label noise and the blue displays that for 80% instances with clean label. The gap between scores of clean data and noisy data, which enables us to detect the noise by the CG-score, is larger for relatively simpler dataset, FMINST, than those for CIFAR-10/100.

## F CORRELATIONS BETWEEN CG-SCORE AND PARTIAL CG-SCORES

From the CG-score, which is expressed as $d_i^{-1}(\mathbf{y}_{-i}^\top \mathbf{h}_i)^2 + 2y_i(\mathbf{y}_{-i}^\top \mathbf{h}_i) + y_i^2 d_i$ at the equation 6, we can define three partial CG-scores, $d_i^{-1}(\mathbf{y}_{-i}^\top \mathbf{h}_i)^2$, $2y_i(\mathbf{y}_{-i}^\top \mathbf{h}_i)$, and $y_i^2 d_i$, respectively. In this section, we analyze which term dominates the order of CG-scores for two public datasets, CIFAR-10/100, by analyzing the correlation between the CG-score and the three partial CG-scores. We also examine whether the correlations vary when we change the size of the Gram matrix $\mathbf{H}^\infty$, whose dimension changes according to the level of sub-sampling, explained in Sec. A.2. We get $\mathbf{H}^\infty$ with different subsampling ratios, 1:1, 1:2, 1:3, and 1:4, and then calculate the CG-score and partial CG-scores, defined by the three partial terms. Table 3 shows the Spearman rank correlations between




(a) Distribution of CG-scores at CIFAR-100




(b) Examples and histogram of four classes in CIFAR-100

Figure 7: Sample analysis for CIFAR-100 dataset in terms of the CG-score. (a) shows the CG-score means and standard deviations of 100 classes in CIFAR-100 dataset. (b) shows ten top-ranked examples (top two rows) and ten bottom ranked examples (bottom two rows) for orange, plain, bottle, and flatfish classes of CIFAR-100 dataset. Histograms show distributions of the CG-scores for each class.

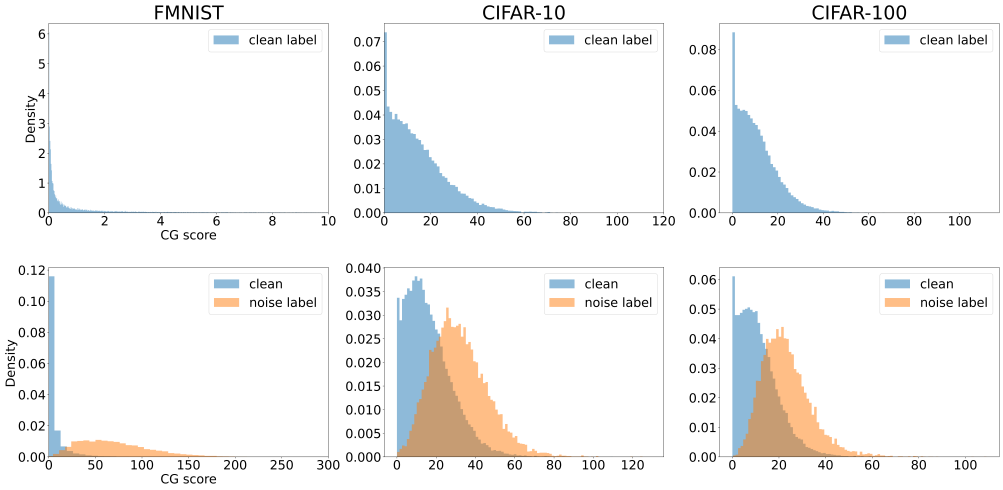



Figure 8: Density of CG-scores at clean dataset and label noisy dataset.

Table 3: Spearman rank correlations between the CG-score and the partial CG-scores

| Dataset | CIFAR-10 | | | | CIFAR-100 | | | |
|---|---|---|---|---|---|---|---|---|
| Subsampling Ratio | 1:1 | 1:2 | 1:3 | 1:4 | 1:1 | 1:2 | 1:3 | 1:4 |
| $d_i^{-1}(\mathbf{y}_{-i}^\top \mathbf{h}_i)^2$ | -0.759 | -0.461 | -0.136 | 0.132 | -0.671 | -0.296 | 0.035 | 0.264 |
| $2y_i(\mathbf{h}_i^\top \mathbf{h}_i)$ | 0.912 | 0.948 | 0.962 | 0.970 | 0.902 | 0.939 | 0.955 | 0.964 |
| $y_i^2 d_i$ | -0.015 | 0.066 | 0.121 | 0.163 | -0.069 | 0.018 | 0.087 | 0.136 |

Table 4: Correlation between feature-space CG-score, CG-score and other training-based scores

| Correlation btwn. Feature-space CG-score and | | | |
|---|---|---|---|
| CG-score | C-score | Forgetting | EL2N |
| 0.251 | 0.414 | 0.437 | 0.338 |

the CG-scores and the partial CG-scores. We can see that $2y_i(\mathbf{y}_{-i}^\top \mathbf{h}_i)$, which was utilized at noise detection experiments, is the partial score having the largest correlation with the CG-score across all subsampling ratios. On the other hand, correlations to other partial-scores are relatively low, and the correlations change much as the subsampling ratio changes.

## G FEATURE-SPACE CG-SCORE

Our original CG-score can be calculated by data only without any trained network. In this section we check the validity of a new CG-score, which is defined in terms of feature of data. We train ResNet18 using the full CIFAR-10 dataset to obtain each data's feature, which is defined at the output of the penultimate layer value (512 dimensions). Then we calculate the feature-space CG score, which is calculated by feature instead of the data itself, and check the Spearman rank correlation between the feature-space CG-score and the previous training-based scores including C-score, Forgetting score, and EL2N. In Table 4, we can see that there exist correlations between the feature-space CG-score and previous training-based scores. The correlation between the original (data-centric) CG-score and feature-space CG-score is relatively low. This result implies that the feature-space CG score includes meaningful information for data valuation, correlated with other training-based scores, but the context of information might be different from that of the original CG-score. The feature-space CG score can also be easily calculated if a pretrained network exists. Understanding and evaluating the feature-space CG score can be an interesting future research direction.