COVERAGE-CENTRIC CORESET SELECTION FOR HIGH PRUNING RATES

Anonymous authors
Paper under double-blind review

ABSTRACT

One-shot coreset selection aims to select a subset of the training data, given a pruning rate, that can achieve high accuracy for models that are subsequently trained only with that subset. State-of-the-art coreset selection methods typically assign an importance score to each example and select the most important examples to form a coreset. These methods perform well at low pruning rates; but at high pruning rates, they have been found to suffer a *catastrophic accuracy drop*, performing worse than even random coreset selection. In this paper, we explore the reasons for this accuracy drop both theoretically and empirically. We extend previous theoretical results on the bound for model loss in terms of coverage provided by the coreset. Inspired by theoretical results, we propose a novel coverage-based metric and, based on the metric, find that coresets selected by importance-based coreset methods at high pruning rates can be expected to perform poorly compared to random coresets because of worse data coverage. We then propose a new coreset selection method, Coreset-centric Coreset Selection (CCS), where we jointly consider overall data coverage based on the proposed metric as well as importance of each example. We evaluate CCS on four datasets and show that they achieve significantly better accuracy than state-of-the-art coreset selection methods as well as random sampling under high pruning rates, and comparable performance at low pruning rates. For example, CCS achieves 7.04% better accuracy than random sampling and at least 20.16% better than popular importance-based selection methods on CIFAR10 with a 90% pruning rate.

1 INTRODUCTION

Massive training datasets help deep neural networks achieve impressive performance in many real-world applications such as image classification (Deng et al., 2009), speech recognition (Chung et al., 2018), and machine translation (Vaswani et al., 2017). But, such training is computationally expensive. How to select a small subset of the training data that still achieves high model accuracy is attracting more and more attention in the machine learning community (Liang et al., 2022) and is characterized as the *coreset selection problem*. In this paper, we consider *one-shot coreset selection* that aims to select a subset of the training data that can later be used to train models while retaining high accuracy (Coleman et al., 2019; Toneva et al., 2018) [1] A favored way to select such coresets is to assign an importance score to each example and select more important examples to form the coreset (Paul et al., 2021; Sorscher et al., 2022). Recent importance-based coreset methods (Toneva et al., 2018; Coleman et al., 2019; Paul et al., 2021) propose various different importance metrics. Pruning unimportant examples with these metrics achieves better accuracy than random sampling with low pruning rates. However, surprisingly, these methods suffer *a catastrophic accuracy drop* under high pruning rates (Guo et al., 2022; Paul et al., 2021), performing significantly worse than models trained with randomly sampled subsets. This accuracy drop is a significant limitation as it limits the extent to which coresets can be reduced in size with these selection methods.

In this paper, we explore the reasons behind this catastrophic accuracy drop, both theoretically and empirically, and show that current coreset selection methods ignore an essential factor, data

[1] A related coreset research direction is *dynamic coreset selection*. Dynamic coreset selection allows the coreset to be changed during training between epochs (Killamsetty et al., 2021b; Mirzasoleiman et al., 2020). Dynamic coreset selection is out of scope for our paper. We discussed more details in Section 5.

coverage, which is important for model accuracy. We propose a novel coreset selection method, called Coverage-centric Coreset Selection (CCS), that combines data coverage with importance-based sampling. We find that CCS significantly outperforms existing methods at high pruning rates. More specifically, we make the following contributions:

1. We start by generalizing the result in (Sener & Savarese, 2017), which provided a probabilistic bound on model loss for a r-cover coreset, where r is the smallest radius around the coreset points that covers 100% of the training set, to a density-based characterization of the coreset in which the radius r is allowed to cover a percentage p of a density space. This density-based partial cover allows us to use a p-r curve to describe the trade-off between radius r and cover percentage p. We theoretically prove a bound on model loss as a function of r and p for a given coreset (Section 3.1).
2. Inspired by the theoretical results, we propose a novel metric, the area under the p-r curve (AUC_{pr}) (Section 3.1). Lower values of AUC_{pr} are more desirable and suggest higher data coverage. Based on the proposed metric, we show that coresets selected by current coreset methods at high pruning rates can be expected to perform worse than random pruning (Section 3.2).
3. We then investigate if factoring in coverage in importance-based sampling can help prevent catastrophic accuracy drop at high pruning rates. We propose the CCS method that adopts a classical idea from statistics: stratified sampling. Different from current methods that prune unimportant (easy) examples first, CCS, based on stratified sampling, ensures sampling budgets for data with different importance scores (Section 3.3).
4. We evaluate CCS on four datasets (CIFAR10, CIFAR100 (Krizhevsky et al., 2009), SVHN (Netzer et al., 2011), and CINIC10 (Darlow et al., 2018)) and compare it with five other coreset selection methods: random, entropy (Coleman et al., 2019), forgetting (Toneva et al., 2018), EL2N (Paul et al., 2021), and AUM (Pleiss et al., 2020). Compared to these counterparts, we find that CCS achieves better accuracy with high pruning rates on all four datasets and comparable performance with low pruning rates. For example, with a 90% pruning rate, on CIFAR 10 (CIFAR100), CCS achieves 7.04% (10.83%) better accuracy than random sampling and at least 20.16% (26.63%) better than other baseline coreset methods [2] (Section 4).

To the best of our knowledge, our work is the first to study the data coverage problem of importance-based one-shot coreset selection and propose a one-shot coreset selection method that works well at both low and high pruning rates.

2 PRELIMINARIES

2.1 ONE-SHOT CORESET SELECTION

Consider a classification task with a training dataset containing N examples drawn i.i.d. from an underlying distribution P. We denote the training dataset as $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $\mathbf{x}_i$ is the data, and y_i is the ground-truth label. The goal of one-shot coreset selection is to select a fixed b budget training subset *before training* to maximize the accuracy of models trained on this subset, which can be formulated as the following optimization problem (Sener & Savarese, 2017):

$$\min_{\mathcal{S}' \subset \mathcal{S}: |\mathcal{S}'| \leq b} \mathbb{E}_{\boldsymbol{x}, y \sim P}[l(\boldsymbol{x}, y; h_{\mathcal{S}'})], \tag{1}$$

where l is the loss function, and $h_{\mathcal{S}}$ is the model trained with a labelled dataset $\mathcal{S}$.

State-of-the-art methods typically assign an importance score (also called a difficulty score) to each example and preferably select more important (difficulty) examples to form the coreset. Forgetting score (Toneva et al., 2018) is defined as the number of times an example is incorrectly classified after having been correctly classified earlier during model training. The area under the margin (AUM) (Pleiss et al., 2020) is a data difficulty metric proposed to identify mislabeled data, but also achieves good performance on coreset selection. EL2N score (Paul et al., 2021) estimates data

[2] Values in parenthesis are for CIFAR100.

difficulty by the L2 norm of error vectors and shows that the difficulty of data can be determined at the early stage of training. Algorithms based on these metrics all prune examples with low scores (i.e., easy or low-importance examples) first (Sorscher et al., 2022). As discussed in (Sorscher et al., 2022), harder examples usually contains more information to learn and are treated as more important examples. We collectively refer to these score-based strategies that prune easy or low-importance examples first as SOTA (State-of-the-art).

2.2 Catastrophic accuracy drop with high pruning rates

Although SOTA coreset methods (Toneva et al., 2018; Coleman et al., 2019; Paul et al., 2021) report encouraging performance, there is observed *a catastrophic accuracy drop* when the pruning rate increases (Guo et al., 2022; Paul et al., 2021): models trained with selected coresets have even worse accuracy than models trained with randomly sampled subset. As shown in Figure 1, forgetting number and AUM outperform random sampling when the pruning rate is less than 70%, but only achieve 34.03% and 28.06% accuracy, respectively, with a 90% pruning rate, which is much worse than 79.04% achieved by random sampling. A potential reason behind this accuracy drop is the bad data coverage caused by the biased pruning of importance-based methods. In this paper, we conduct an in-depth study on this catastrophic accuracy drop and propose a novel coreset method to mitigate the accuracy drop for the high pruning rate coreset selection.

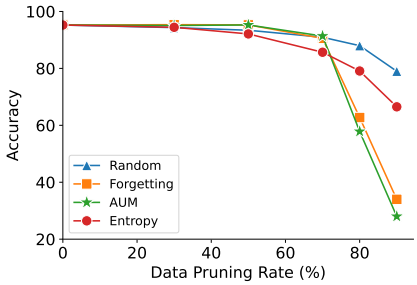


Figure 1: Existing coreset solutions have better accuracy than random sampling with low pruning rates, but perform worse with high pruning rates.

3 The coverage of coresets

In this section, we start by studying data coverage provided by different coresets. We first extend the classical geometric set cover to a density-based partial cover problem and propose a novel metric to compare the data coverage of different coresets (Section 3.1). We then show that biased pruning on easy data leads to poor data coverage of coresets with high pruning rates (Section 3.2). Finally, we design a novel coverage-centric coreset selection method based on stratified sampling (Section 3.3).

3.1 Density-based partial coverage

To compare the data coverage of different coreset methods, we need to quantitatively measure how well a dataset $\mathcal{S}$ covers a distribution P. In classical geometric set cover setting (Fowler et al., 1981; Sener & Savarese, 2017), we say a set $\mathcal{S}'$ is a r-cover of another set $\mathcal{S}$, when a set of r-radius balls centered at each element in $\mathcal{S}'$ covers the entire $\mathcal{S}$. The radius r can be used as a metric to measure coverage of $\mathcal{S}'$ on $\mathcal{S}$.

To measure how well a set covers a distribution, we extend the classical geometric set cover to the *density-based partial cover*. Instead of covering a set, we study the covering on a distribution P_μ and consider probability density in different areas of the input space. Instead of taking a traditional complete cover, we introduce the cover percentage p to describe how a set covers different percentages of a distribution to better understand the trade-off between cover radius r and cover percentage p:

Definition 1 (p-partial r-cover)**.** *Given a metric space (X, d), a set $\mathcal{S} \subset X$ is a p-partial r-cover for a distribution P_μ on the space X if:*

$$\int_X \mathbf{1}_{\cup_{\mathrm{x}\in\mathcal{S}}\mathrm{B_d(x,r)}}(\boldsymbol{x}) d\mu(\boldsymbol{x}) = p,$$

where $B_d(\boldsymbol{x}, r) = \{\boldsymbol{x}' \in X : d(x, x') \leq r\}$ is a r-radius ball whose center is $\boldsymbol{x}$, p measures coverage (which we will usually express as a percentage and refer to as percentage coverage*), and μ is the probability measure*[3] *corresponding to probability density function P_μ.*

[3]By Radon–Nikodym theorem, there is a probability measure μ: $P_\mu(x)dx = d\mu(x)$.

In Definition 1, the probability measure μ represents the density information in the input space.

For the model trained on a r-cover coreset, Sener & Savarese (2017) provides a bounded risk on the entire training dataset. Here, we extend their result, but for a p-partial r-cover stated as Theorem 1 below. The proof of Theorem 1 relies on the same assumptions as in (Sener & Savarese, 2017), in particular Lipschitz-continuity loss function and zero training error on the coreset. The zero training error assumption may not always be realistic, Sener & Savarese (2017) found that the resulting bounds still hold and we rely on the same assumptions. The proof can be found in the Appendix C.

Theorem 1. *Given n i.i.d. samples drawn from P_μ as $\mathcal{S} = \{\boldsymbol{x}_i, y_i\}_{i\in[n]}$ where $y_i \in [C]$ is the class label for example x_i, a coreset $\mathcal{S}'$ which is a p-partial r-cover for P_μ on the input space X, and an $\epsilon > 1 - p$, if the loss function $l(\cdot, y, \boldsymbol{w})$ is λ_l-Lipschitz continuous for all y, $\boldsymbol{w}$ and bounded by L, regression function is λ_η-Lipschitz, and $l(\boldsymbol{x}, y; h_{\mathcal{S}'}) = 0$, $\forall(\boldsymbol{x}, y) \in \mathcal{S}'$, then with probability at least $1 - \epsilon$:*

$$\left| \frac{1}{n} \sum_{\boldsymbol{x},y\in\mathcal{S}} l(\boldsymbol{x}, y; h_{\mathcal{S}'}) \right| \leq r(\lambda_l + \lambda_\eta LC) + L\sqrt{\frac{\log \frac{p}{p+\epsilon-1}}{2n}} \tag{2}$$

Theorem 1 shows that the model trained on a p-partial r-cover coreset $\mathcal{S}'$ has a bounded risk on the entire training dataset $\mathcal{S}$. There are several implications of Theorem 1: 1) A model trained on a p-partial r-cover coreset has a training risk bounded with the covering radius r and an additional term. This term goes to zero as n increases to infinity. Also, we want a small ϵ, a large p and a small radius r to get a high probability small bound on the loss. 2) The probability that Equation 2 holds is bounded by cover percentage p (since $1 - \epsilon < p$). This indicates that the bound in Theorem 1 becomes meaningless (low probability) when the cover percentage p is too small. 3) Theorem 1 is a more general form of Theorem 1 in (Sener & Savarese, 2017); if we set $p = 1$ (i.e., complete cover), these two theorems are the same.

Definition 1 shows that, for a given set $\mathcal{S}$ and a distribution P_μ, the cover percentage p increases with the cover radius r. Conversely, given a cover percentage p and a set S, we can get a minimum cover radius r for the $\mathcal{S}$ to achieve p percent coverage on distribution P_μ. For a given set $\mathcal{S}$ and a distribution P_μ, we can plot a p-r curve to represent this relationship. In practice, the distribution P_μ is unknown. So, we estimate the cover percentage p for a given subset S of the training dataset and a radius r by measuring the fraction of the test dataset that is covered [4]. To assess coverage for a radius r, we use the L2 distance between activations of the final convolutional layer of a model trained with the entire dataset as the distance metric d, as suggested in (Babenko & Lempitsky, 2015; Sharif Razavian et al., 2014; Hsieh et al., 2018).

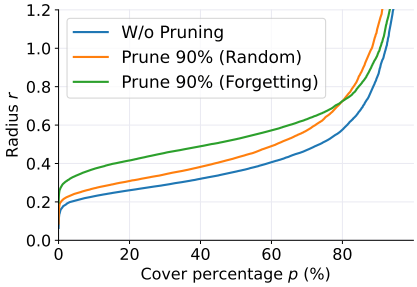


Figure 2: The p-r curves of different subsets of CIFAR10 training dataset. The dataset without any pruning has the lowest curve (blue). Forgetting curve (green) and random curve (orange) have a crossover around $p = 80\%$.

Figure 2 shows the p-r curves for several subsets of CIFAR10 training dataset: (1) entire dataset (no pruning); (2) random coreset at 90% pruning rate (i.e., random 10% of the training data); and (3) coreset selected based on forgetting score as the importance metric, as proposed by Toneva et al. (2018). Notice that, as expected, the required radius r increases to get the desired cover percentage p for all coresets. Also coreset with no pruning has the lowest curve, as expected. In general, lower curves are better and we expect models based on the corresponding datasets to have lower loss. But, we find that there is crossover between forgetting curve (green) and random curve (orange) around $p = 80\%$, making a comparison less obvious. We propose to use area under

[4] We notice that some misclassified examples have a much larger minimum cover radius than other data. To eliminate the influence of outliers, when we plot p-r curves, we ignore the data misclassified by the model trained with the entire CIFAR10 dataset (with a 95.25% accuracy).

the p-r curve, AUC_{pr}, as a proxy metric to assess the quality of a coreset selection strategy. Lower values of AUC_{pr} suggest better coverage provided by the coreset.

We note that AUC_{pr} is the expected minimum distance between examples following the underlying distribution P_μ to those in the coreset, as stated in the following proposition (proved in Appendix C):

Proposition 1. *Given a metric space (X, d), a distribution P_μ, cover percentage p, a set $\mathcal{S}$, and $f_r(p) : [0, 1] \to [0, +\infty]$ representing the mapping between p and r, if f_r is Riemann-integrable, the AUC of the p-r curve $AUC_{pr}(\mathcal{S}) = \int_0^1 f_r(p)dp = \mathbb{E}_{x\sim P_\mu}[d(S, \boldsymbol{x})]$, where $d(S, \boldsymbol{x}) = \min_{\boldsymbol{x}'\in S} d(\boldsymbol{x}', \boldsymbol{x})$.*

In practice, we can assess AUC_{pr} with test set D_{test} : $AUC_{pr}(S) = \mathbb{E}_{\boldsymbol{x}\in D_{test}}[\min_{\boldsymbol{x}'\in S} d(\boldsymbol{x}', \boldsymbol{x})]$. We also note that this expected minimum distance (and thus AUC) is related to r in Theorem 1; lower values of AUC_{pr} are preferred for lower loss and thus higher model accuracy.

3.2 Coverage analysis on coresets

Table 1: AUC_{pr} of different methods with different pruning rates. With low pruning rates (30%, 50%), forgetting and AUM have similar AUC_{pr} to random. With high pruning rates (80%, 90%), forgetting and AUM have larger AUC_{pr} than random, which means worse data coverage.

Pruning Rate	30%	50%	70%	80%	90%
Random	0.496	0.509	0.532	0.552	0.589
Forgetting	0.492	0.511	0.551	0.580	0.631
AUM	0.496	0.518	0.558	0.586	0.646

Table 1 shows the AUC_{pr} for different CIFAR10 coresets with different pruning rates. With a 30% or 50% pruning rate, forgetting and AUM have similar AUC_{pr} to random sampling. However, with a pruning rate larger than 70%, forgetting and AUM have larger AUC_{pr} than random pruning, suggesting that forgetting and AUM can be expected to have a worse accuracy than random pruning. The reason behind higher AUC_{pr} for forgetting and AUM is that they prune easy examples (Toneva et al., 2018; Sorscher et al., 2022), which tend to be located in the high-density area of distribution; more common examples often tend to be well-separated easier examples as well for classifiers. Low pruning rate coresets still contain plenty of easy examples for all methods; thus coverage is not a significant factor at low pruning rates. However, with high pruning rates, coresets can completely exclude examples from the high-density areas, reducing coverage and raising AUC_{pr}, and thus potentially raising loss and reducing accuracy.

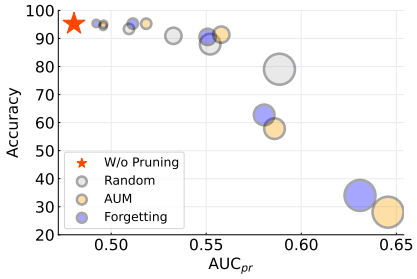


Figure 3: The relationship between AUC_{pr} and accuracy. Different colors stands for different coreset selected methods. A larger circle size indicates a higher pruning ratio on the dataset. Smaller AUC_{pr} often ends up with better accuracy. With high pruning rates (larger balls), forgetting and AUM have a higher AUC_{pr} and worse accuracy.

For the data points in Table 1, Figure 3 plots model accuracy and AUC_{pr}. Size of the ball at a point represents pruning rate for that point – larger ball size indicates a larger pruning rate. The plot generally confirms our intuitions from Section 3.1 that higher AUC_{pr} values tend to lead to lower model accuracy.

3.3 Methodology: Coverage-centric coreset selection

Based on the observations in Section 3.2 that current SOTA methods (forgetting and AUM) prune low-importance (easy) data from the high-density area, which contributes to larger values of AUC_{pr}, and thus possibly lower model accuracy, we propose a novel coreset selection method called Coverage-centric Coreset Selection (CCS), presented in Algorithm 1. CCS is still based on existing

Algorithm 1: Coverage-centric Coreset Selection (CCS)

Inputs:
$\mathbb{S} = \{(x_i, y_i, s_i)\}_{i=1}^{n}$: dataset with normalized importance score for each example;
α: dataset pruning rate; β: hard cutoff rate; k: the number of strata.

$\mathbb{S}' \leftarrow \mathbb{S} \setminus \{\lfloor n * \beta \rfloor$ hardest examples $\}$; `/* Prune hard examples first */`
$\mathcal{B} \leftarrow \{\mathbb{B}_i$: split $\mathbb{S}'$ into k strata evenly by difficulty score.$\}$;
$m \leftarrow n \times (1 - \alpha)$;
$\mathbb{S}_c \leftarrow \varnothing$;
while $\mathcal{B} \neq \varnothing$ **do**
  $\mathbb{B}_{min} \leftarrow \arg\min_{\mathbb{B} \in \mathcal{B}} |\mathbb{B}|$; `/* Select the stratum with the fewest data */`
  $m_B \leftarrow \min\{|\mathbb{B}_{min}|, \lfloor \frac{m}{|\mathcal{B}|} \rfloor\}$;
  $\mathbb{S}_B \leftarrow$ randomly sample m_B examples from $\mathbb{B}_{min}$;
  $\mathbb{S}_c \leftarrow \mathbb{S}_c \cup \mathbb{S}_B$;
  $\mathcal{B} \leftarrow \mathcal{B} \setminus \{\mathbb{B}_{min}\}$;
  $m \leftarrow m - m_B$;
return $\mathbb{S}_c$

importance scores (e.g., entropy, forgetting score, AUM, and EL2N). However, unlike SOTA methods that simply select examples with highest importance score, CCS uses stratified sampling (Neyman, 1992) on top of importance scores to improve coverage and lower the value of AUC_{pr}.

Stratified sampling first divides a population into different exclusive strata and then applies random sampling in each stratum based on the desired pruning rate α. The advantage of stratified sampling is to guarantee that data in all strata can be sampled. In our algorithm, we divide all training data into k even strata by importance scores. If a stratum has fewer examples than the budget, remaining budget is evenly assigned to other strata. With a low pruning rate, stratified sampling tends to first prune examples in high-density strata – similar to SOTA methods. But, at high pruning rates, stratified sampling diverges from SOTA methods in that high-density strata still get some budget to guarantee data coverage.

Before stratified sampling, CCS prunes β percent "hard" examples based on importance scores. This is based on two insights: (1) Mislabeled examples often also have higher importance scores (Swayamdipta et al., 2020) but do not benefit accuracy. As the pruning rate increases, the percentage of mislabeled data in the coreset also rises, which hurts model accuracy. (2) When data is really scarce, we need more data from higher-density areas to get better coverage. Pruning hard examples help allocate more budget to high-density strata. In practice, we use grid search to find the optimal β. We discuss more details on selecting β in Appendix B and how β impacts training in Section 4.2.

In Figure 4, we show an example on CIFAR10 to compare the fraction of data selected between a SOTA method (AUM) and CCS (also based on AUM) for 70% pruning rate. The data selected by the SOTA method Pleiss et al. (2020), shown with red shading, prunes all the easy examples (high AUM value). In contrast, CCS selects the data in the green region. In Section D.2, we showed that the coresets selected by CCS have a lower AUC_{pr} (i.e., better coverage) than the SOTA method, which prunes easy examples first.

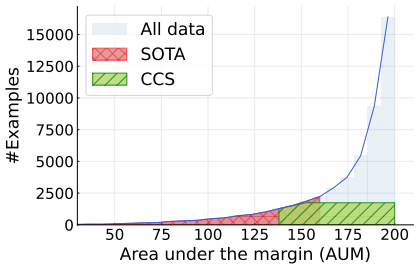


Figure 4: CIFAR10 data distribution on AUM. We use different colors to show data selected by a SOTA method (red), which prunes low-importance (easy) examples first, and our method (green) for a 70% pruning rate. The coreset selected by the SOTA method lack "easy" examples on the right in the high-density region, since they are considered to be of low importance. However, our method provides coverage on high-density region.

Table 2: Accuracy results on CIFAR10 comparing CCS against other baseline methods. Pruning rate is the fraction of examples that are removed from the original dataset. We report the testing accuracy of the model trained on the pruned dataset and highlight The model trained with the entire dataset has 95.23% accuracy. When we prune out 30% or 40% fraction of data, CCS achieves comparable accuracy. With a 70%, 80%, or 90% pruning rate, CCS outperforms baseline methods by a large margin.

Pruning Rate	30%	50%	70%	80%	90%
Random	$94.33_{\pm 0.17}$	$93.4_{\pm 0.17}$	$90.94_{\pm 0.38}$	$87.98_{\pm 0.39}$	$79.04_{\pm 1.53}$
Entropy	$94.44_{\pm 0.2}$	$92.11_{\pm 0.47}$	$85.67_{\pm 0.71}$	$79.08_{\pm 0.36}$	$66.52_{\pm 1.08}$
Forgetting	$95.36_{\pm 0.13}$	$\mathbf{95.29_{\pm 0.18}}$	$90.56_{\pm 1.8}$	$62.74_{\pm 2.42}$	$34.03_{\pm 1.05}$
EL2N	$\mathbf{95.44_{\pm 0.15}}$	$94.61_{\pm 0.20}$	$87.48_{\pm 1.33}$	$70.32_{\pm 2.11}$	$22.33_{\pm 0.54}$
AUM	$95.07_{\pm 0.24}$	$95.26_{\pm 0.15}$	$91.36_{\pm 1.4}$	$57.84_{\pm 4.1}$	$28.06_{\pm 1.09}$
Forgetting (CCS)	$95.40_{\pm 0.12}$	$95.04_{\pm 0.37}$	$92.97_{\pm 0.25}$	$90.93_{\pm 0.22}$	$85.70_{\pm 0.36}$
EL2N (CCS)	$94.84_{\pm 0.15}$	$94.03_{\pm 0.24}$	$92.25_{\pm 0.24}$	$89.81_{\pm 0.30}$	$\mathbf{86.68_{\pm 1.25}}$
AUM (CCS)	$95.27_{\pm 0.06}$	$94.93_{\pm 0.18}$	$\mathbf{93.00_{\pm 0.16}}$	$\mathbf{90.91_{\pm 0.27}}$	$86.08_{\pm 0.61}$

4 EVALUATION

This section presents experiments to validate the effectiveness of CCS. We compare CCS with five baseline methods (random, entropy, forgetting, EL2N, and AUM) on four datasets (CIFAR10, CIFAR100, SVHN, and CINIC10). In total, we trained 800 models for performance comparison. The evaluation results show that CCS outperforms other baseline methods with high pruning rates and achieves comparable accuracy with low pruning rates. We also conduct an ablation study (Section 4.2) to better understand how hyperparameters and different components influence the accuracy. Due to space limits, we include detailed experimental setting in Appendix. B. All training is repeated 5 times with different random seeds to calculate mean accuracy with standard deviation.

Baselines. We compare CCS with the following five baselines, with latter four being SOTA methods: 1) **Random**: Uniformly sampling the examples to form coresets. 2) **Entropy** (Coleman et al., 2019): The entropy of outputs reflects the uncertainty of training examples, and a high entropy indicates an example containing more information and is more important. 3) **Forgetting score** (Toneva et al., 2018): Forgetting score is the number of times an example is incorrectly classified after being correctly classified earlier during model training. A higher forgetting score suggests that the example is more important. 4) **EL2N** (Paul et al., 2021): EL2N scores estimate data difficulty by the L2 norm of error vectors. Examples with large difficulty score are more important. 5) **Area under the margin (AUM)** (Pleiss et al., 2020): AUM represents the probability gap between the target class and the largest other class across all training epochs. A larger AUM means higher difficulty and importance. We apply our coverage-centric method on forgetting score, EL2N, and AUM.

4.1 CORESET PERFORMANCE COMPARISON

We compare CCS against above baseline methods for five different pruning rates. Table 2 shows the evaluation results on the CIFAR10 dataset. We observe that the four SOTA methods tend to have similar or higher accuracy than random for 50% or lower pruning rates. At high pruning rates (e.g., 80% and 90%), they consistently have a worse performance than random. In contrast, CCS-based schemes achieve comparable accuracy under low pruning rates, while trouncing their counterparts under high pruning rates. At 90% pruning rate, best CCS-scheme achieve 7.64% better accuracy than random sampling and 20.16% better than the best-performing SOTA method.

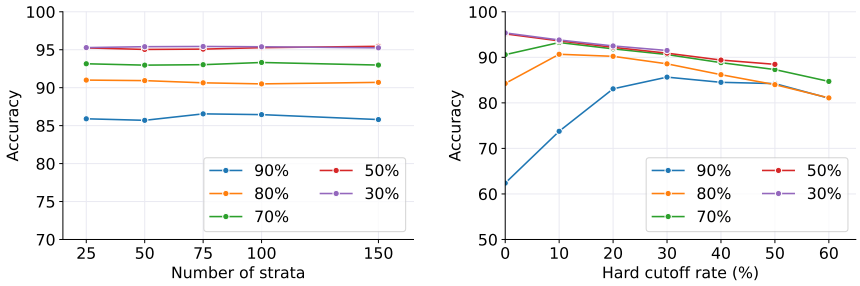
Table 3 shows results on CIFAR100. CIFAR100 is a more challenging dataset than CIFAR10, since CIFAR100 only contains 500 images per class. This can cause fewer redundant easy images in the dataset. We find that SOTA methods have worse performance than random sampling even at 50% pruning rate and significantly underperform random sampling at higher pruning rates. In contrast, CCS achieves comparable accuracy at low pruning rates (up to 50%) and much higher accuracy as pruning rate increases. When we prune 90% fraction of data, the accuracy of best CCS method is 10.83% higher than random sampling and 26.63% higher than best-performing SOTA method.

Table 3: Accuracy results on CIFAR100 comparing CCS against other baseline methods. Pruning rate is the fraction of examples that are removed from the original dataset. The model trained with the entire dataset has 78.21% accuracy. When we prune out 30% or 40% fraction of data, CCS achieves comparable accuracy. With a 70%, 80%, or 90% pruning rate, CCS outperforms baseline methods by a large margin.

Pruning Rate	30%	50%	70%	80%	90%
Random	$74.59_{\pm 0.27}$	$71.07_{\pm 0.4}$	$65.3_{\pm 0.21}$	$57.36_{\pm 0.64}$	$44.76_{\pm 1.58}$
Entropy	$72.26_{\pm 0.08}$	$63.26_{\pm 0.29}$	$50.49_{\pm 0.88}$	$41.83_{\pm 0.33}$	$28.96_{\pm 0.78}$
Forgetting	$76.91_{\pm 0.32}$	$68.6_{\pm 1.02}$	$38.06_{\pm 1.14}$	$24.23_{\pm 0.59}$	$15.93_{\pm 0.24}$
EL2N	$76.25_{\pm 0.24}$	$65.90_{\pm 1.06}$	$34.42_{\pm 1.50}$	$15.51_{\pm 1.20}$	$8.36_{\pm 0.19}$
AUM	$76.93_{\pm 0.32}$	$67.42_{\pm 0.49}$	$30.64_{\pm 0.58}$	$16.38_{\pm 0.4}$	$8.77_{\pm 0.35}$
Forgetting (CCS)	$\mathbf{77.14_{\pm 0.31}}$	$\mathbf{74.45_{\pm 0.16}}$	$\mathbf{68.92_{\pm 0.12}}$	$\mathbf{63.99_{\pm 0.37}}$	$\mathbf{55.59_{\pm 0.7}}$
EL2N (CCS)	$75.02_{\pm 0.12}$	$72.09_{\pm 0.53}$	$67.13_{\pm 0.22}$	$61.83_{\pm 0.93}$	$52.55_{\pm 1.63}$
AUM (CCS)	$76.84_{\pm 0.25}$	$73.77_{\pm 0.21}$	$68.85_{\pm 0.08}$	$63.2_{\pm 0.54}$	$55.03_{\pm 0.72}$

We report additional evaluation results on SVHN and CINIC10 datasets in Appendix D. We draw the same conclusions on those datasets.

4.2 Ablation Study & Analysis



(a) Relationship between k and accuracy. (b) Relationship between β and accuracy.

Figure 5: Ablation study on the number of strata (k) and hard cutoff rate (β). We apply CCS with the forgetting score on CIFAR10. Each curve is corresponding to a pruning rate. Different k lead to similar accuracy. The optimal β goes up as the pruning rate increases.

We study the effect of hyperparameters (k and β) and different components of CCS on CIFAR10 dataset sorted by the forgetting score. We first discuss the effect of the number of strata k in stratified sampling. As shown in Figure 5(a), model accuracy only changes slightly as the number of strata changes, which suggests that the number of strata is not a sensitive hyperparameter. We then study the hard cutoff rate β. From Figure 5(b), we observe that the optimal hard cutoff rate β goes up as the pruning rate increases. With a larger pruning rate, we need to prune more hard examples to get the best accuracy. As discussed in Section 3.3, a larger hard cutoff rate helps allocate more budget to high-density strata. When the total data budget is limited, better coverage on high-density strata helps improve model performance. However, further increasing β causes a drop in accuracy for all pruning rates. We also observe that the optimal β is 0 when we need to prune at lower levels of 30% and 50%. A possible reason is that the percentage of mislabeled data is low in CIFAR10 (Pleiss et al., 2020), and deep neural network is robust to a small portion of noisy data (Rolnick et al., 2017).

Grid search for β can introduce extra costs in selecting effective coresets. However, our ablation study shows that our solution consistently performs well across different values of β. Moreover, we note that our paper shows the possibility of achieving better accuracy than random sampling even with a high pruning rate. We leave how to more efficiently find the optimal β for future work.

Table 4: Ablation study on different components of coverage-centric coreset select method. We compare the accuracy of different selection methods on the CIFAR10 dataset sorted by forgetting score. We use only stratified sampling and pruning hard examples to select the coreset, respectively. We highlight the best accuracy in bold in each column.

Pruning Rate	30%	50%	70%	80%	90%
Random	94.33	93.4	90.94	87.98	79.04
Forgetting	95.36	**95.29**	90.56	62.74	34.03
Stratified only	95.23	95.13	90.78	81.82	59.35
Prune hard only	91.50	88.46	85.62	80.98	80.62
CCS (Stratified + Prune hard)	**95.40**	95.04	**92.97**	**90.93**	**85.7**

Next, we study the impact of stratified sampling and pruning hard examples on model accuracy. We measure the forgetting score for CIFAR10 and then apply only stratified sampling and pruning hard examples to select the coreset, respectively, in Table 4. By comparing stratified sampling only with the vanilla forgetting method (pruning easy first), we observe that applying stratified sampling improves accuracy compared to the vanilla forgetting method, but it is still inferior to random sampling when we prune 80%, or 90% fraction of examples. Sorscher et al. (2022) discussed that pruning hard data is a better strategy than pruning easy data when the original entire data is scarce. In our evaluation, we observe that pruning hard data outperforms pruning easy data with high pruning rates, but still has worse accuracy than CCS.

5 Related work

One-shot coreset selection. One-shot coreset selection methods aim to select a fixed b budget training subset before training to maximize the accuracy of models trained on this subset. Besides the coreset methods like entropy Coleman et al. (2019), forgetting score Toneva et al. (2018), AUM Pleiss et al. (2020), EL2N scores Paul et al. (2021) discussed in Section 2.1, Sorscher et al. (2022) use k-means to estimate the example difficulty score, and Sener & Savarese (2017) apply greedy k-center to choose the coreset with good data coverage. However, k-means and k-center algorithms need to calculate the distance matrix between each example, which leads to an $O(n^2)$ time complexity and are expensive to scale to large datasets. As far as we know, we propose the first coverage-centric method to enhance the difficulty-based one-shot coreset selection. Sener & Savarese (2017) also discuss the coverage of the coreset problem with the classical geometric set covering setting, which studies the coverage problem between two sets. In this paper, we further study how a set covers a distribution, showing that coverage is another important factor to consider in importance-based coreset selection.

Dynamic coreset selection. Unlike in one-shot coreset selection, dynamic coreset selection allows the coreset to be changed for every specific number of training iterations. An example is gradient-matching methods (Killamsetty et al., 2021a; Mirzasoleiman et al., 2020; Pooladzandi et al., 2022) which aims to use a subset to estimate the gradient of the full dataset. Since the gradients vary as model parameters change during training, the method updates the coreset to match the gradients for the new parameters (Mirzasoleiman et al., 2020). Unlike in one-shot coreset selection, the much larger dataset must be available during any training to update the coreset.

6 Conclusion

We conduct an in-depth study to understand the *catastrophic accuracy drop* issue of coreset selection. By extending classical geometric set cover to density-based partial cover, we propose a novel data coverage metric and show that existing importance-based coreset selections lead to poor data coverage, which in turn degrades model performance. Based on this observation, we design a novel Coverage-centric Coreset Selection (CCS) method based on stratified sampling to improve the data coverage. Our evaluation results on four datasets show that our method outperforms other methods under high pruning rates and achieves comparable performance for low pruning rates.

7 REPRODUCIBILITY STATEMENT

In the supplemental, we include a zip file containing our code and corresponding instructions. We also include detailed experimental setting in the appendix.

REFERENCES

Artem Babenko and Victor Lempitsky. Aggregating deep convolutional features for image retrieval. *arXiv preprint arXiv:1510.07493*, 2015.

Christopher Berlind and Ruth Urner. Active nearest neighbors in changing environments. In *International conference on machine learning*, pp. 1870–1879. PMLR, 2015.

Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. Voxceleb2: Deep speaker recognition. *Proc. Interspeech 2018*, pp. 1086–1090, 2018.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. In *International Conference on Learning Representations*, 2019.

Luke N Darlow, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

Robert J Fowler, Michael S Paterson, and Steven L Tanimoto. Optimal packing and covering in the plane are np-complete. *Information processing letters*, 12(3):133–137, 1981.

Chengcheng Guo, Bo Zhao, and Yanbing Bai. Deepcore: A comprehensive library for coreset selection in deep learning. *arXiv preprint arXiv:2204.08499*, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. In *The collected works of Wassily Hoeffding*, pp. 409–426. Springer, 1994.

Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodik, Shivaram Venkataraman, Paramvir Bahl, Matthai Philipose, Phillip B Gibbons, and Onur Mutlu. Focus: Querying large video datasets with low latency and low cost. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pp. 269–286, 2018.

Krishnateja Killamsetty, S Durga, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *International Conference on Machine Learning*, pp. 5464–5474. PMLR, 2021a.

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. Glister: Generalization based data subset selection for efficient and robust learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 8110–8118, 2021b.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Weixin Liang, Girmaw Abebe Tadesse, Daniel Ho, Fei-Fei Li, Matei Zaharia, Ce Zhang, and James Zou. Advances, challenges and opportunities in creating data for trustworthy ai. *Nature Machine Intelligence*, pp. 1–9, 2022.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning*, pp. 6950–6960. PMLR, 2020.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. URL `http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf`.

Jerzy Neyman. On the two different aspects of the representative method: the method of stratified sampling and the method of purposive selection. In *Breakthroughs in statistics*, pp. 123–150. Springer, 1992.

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*, 34, 2021.

Geoff Pleiss, Tianyi Zhang, Ethan Elenberg, and Kilian Q Weinberger. Identifying mislabeled data using the area under the margin ranking. *Advances in Neural Information Processing Systems*, 33:17044–17056, 2020.

Omead Pooladzandi, David Davini, and Baharan Mirzasoleiman. Adaptive second order coresets for data-efficient machine learning. In *International Conference on Machine Learning*, pp. 17848–17869. PMLR, 2022.

David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*, 2017.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 806–813, 2014.

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari S Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. *arXiv preprint arXiv:2206.14486*, 2022.

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 (EMNLP)*, pp. 9275–9293, 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 *International Conference on Learning Representations*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

A OVERVIEW

This appendix provides details on our experimental setting, theoretical analysis, and additional evaluation results. In Section B, we introduce the detailed setup of our experiment to improve the reproducibility of our paper. In Section C, we provide proof for Theorem 1 and Proposition 1. At last, in Section D, we present evaluation results on SVHN and CINIC10. We also conduct a study on the transferability of the coreset selected by CCS.

B DETAILED EXPERIMENTAL SETTING

CIFAR10 and CIFAR100 (Krizhevsky et al., 2009). We use ResNet18 (He et al., 2016) as the network architecture for CIFAR10/CIFAR100. For all coresets with different pruning rates, we train models for $40,000$ iterations with a 256 batch size (about 200 epochs over the entire dataset). We use the SGD optimizer (0.9 momentum and 0.0002 weight decay) with a 0.1 initial learning rate. The learning rate scheduler is the cosine annealing learning rate scheduler (Loshchilov & Hutter, 2017) with a 0.0001 minimum learning rate. We use a 4-pixel padding crop and a randomly horizontal flip as data augmentation.

SVHN (Netzer et al., 2011). We use ResNet18 (He et al., 2016) as the network architecture. For all coresets with different pruning rates, we train models for $30,000$ iterations with a 256 batch size (about 100 epochs over the entire dataset). We use the SGD optimizer (0.9 momentum and 0.0002 weight decay) with a 0.1 initial learning rate. The learning rate scheduler is the cosine annealing learning rate scheduler (Loshchilov & Hutter, 2017) with a 0.0001 minimum learning rate.

CINIC10 (Darlow et al., 2018). The CINIC10 dataset contains $270,000$ images in total and is evenly split to three subsets: training, valid, and test. Guided by (Darlow et al., 2018), we combine the training and valid set to form a large training dataset containing $180,000$ images and measure the test accuracy with the test set (containing $90,000$ examples). We use ResNet18 (He et al., 2016) as the network architecture. For all coresets with different pruning rates, we train models for $70,000$ iterations with a 256 batch size (about 100 epochs over the entire dataset). We use the SGD optimizer (0.9 momentum and 0.0002 weight decay) with a 0.1 initial learning rate. The learning rate scheduler is the cosine annealing learning rate scheduler (Loshchilov & Hutter, 2017) with a 0.0001 minimum learning rate. We use a 4-pixel padding crop and a randomly horizontal flip as data augmentation.

Difficulty score calculation. For CIFAR10 and CIFAR100, we train a model with the entire dataset for 200 epochs to calculate the forgetting score and AUM and use the last checkpoint to calculate entropy for each training example. We use the first 10 epoch's training information to calculate the EL2N score. For SVHN and CINIC10, we train a model with the entire dataset for 100 epochs to calculate the forgetting score and AUM and use the last checkpoint to calculate entropy for each training example. We use the first 10 epoch's training information to calculate the EL2N score.

Coverage-centric methods setting. We set the number of strata $k = 50$ for all datasets and pruning rates. We use the grid search with 0.1 step size to find an optimal hard cutoff rate β for different datasets and pruning rates. For each dataset, we list the optimal β value for every α as follows in the format of tuple (α, β). For CIFAR10, the optimal setting is $(30\%, 0)$, $(50\%, 0)$, $(70\%, 10\%)$, $(80\%, 10\%)$, $(90\%, 30\%)$. For CIFAR100, the optimal setting is $(30\%, 10\%)$, $(50\%, 20\%)$, $(70\%, 20\%)$, $(80\%, 40\%)$, $(90\%, 50\%)$. For SVHN, the optimal setting is $(30\%, 0)$, $(50\%, 0)$, $(70\%, 10\%)$, $(90\%, 10\%)$, $(95\%, 20\%)$. For CINIC10, the optimal setting is $(30\%, 0)$, $(50\%, 0)$, $(70\%, 10\%)$, $(80\%, 10\%)$, $(90\%, 20\%)$. We conduct a study in Section 4.2 to discuss how k and β influence the performance of coresets and how to search β more efficiently.

Each model is trained on a NVIDIA 2080TI GPU. We also include the implementation in the supplementary material.

C THEORETICAL ANALYSIS

We include full proofs of Theorem 1 and Proposition 1 in this section.

C.1 PROOF OF THEOREM 1

Our proof flow is similar to the proof of Theorem in Sener & Savarese (2017).

Lemma 1. *Berlind & Urner (2015) Fix some $p, p' \in [0,1]$ and $y' \in \{0,1\}$. Then,*

$$p_{y\sim p}(y = y') \leq p_{y\sim p'}(y = y') + |p - p'| \tag{3}$$

Theorem 1. *Given n i.i.d. samples drawn from P_μ as $\mathcal{S} = \{\boldsymbol{x}_i, y_i\}_{i\in[n]}$ where $y_i \in [C]$ is the class label for example x_i, a coreset $\mathcal{S}'$ which is a p-partial r-cover for P_μ on the input space X, and an $\epsilon > 1 - p$, if the loss function $l(\cdot, y, \boldsymbol{w})$ is λ_l-Lipschitz continuous for all y, $\boldsymbol{w}$ and bounded by L, regression function is λ_η-Lipschitz, and $l(\boldsymbol{x}, y; h_{\mathcal{S}'}) = 0$, $\forall(\boldsymbol{x}, y) \in \mathcal{S}'$, then with probability at least $1 - \epsilon$:*

$$\left|\frac{1}{n}\sum_{\boldsymbol{x},y\in\mathcal{S}} l(\boldsymbol{x}, y; h_{\mathcal{S}'})\right| \leq r(\lambda_l + \lambda_\eta LC) + L\sqrt{\frac{\log\frac{p}{p+\epsilon-1}}{2n}}$$

Proof. We first bound $\mathbb{E}_{y_i\sim\eta(\mathbf{x}_i)}[l(\mathbf{x}_i, y_i; h_{\mathcal{S}'})]$, where $\eta(\boldsymbol{x})$ is the output of the model, and we represent $\{y_i = k\} \sim \eta_k(\boldsymbol{x}_i)$ with $y_i \sim \eta_k(\boldsymbol{x}_i)$. We have a condition which states that there exists an $\mathbf{x}_j$ in δ ball around $\mathbf{x}_i$ such that $\mathbf{x}_j$ has 0 loss. Since $\mathcal{S}'$ is a p-partial r-cover for the distribution, for $\boldsymbol{x}_i \sim \mu$, with the probability p, there is $\boldsymbol{x}_j \in \mathcal{S}'$ such that $|\boldsymbol{x}_i - \boldsymbol{x}_j| \leq r$ and $\boldsymbol{x}_j$ has 0 loss.

The following inequality holds with the probability p:

$$\begin{aligned}
&E_{y_i\sim\eta(\mathbf{x}_i)}[l(\mathbf{x}_i, y_i; h_{\mathcal{S}'})] \\
&= \sum_{k\in[C]} p_{y_i\sim\eta_k(\mathbf{x}_i)}(y_i = k) l(\mathbf{x}_i, k; h_{\mathcal{S}'})
\end{aligned}$$

By applying Lemma 1 and 0 training loss condition on $\boldsymbol{x}_j$, we have:

$$\begin{aligned}
&\leq \sum_{k\in[C]} p_{y_i\sim\eta_k(\mathbf{x}_j)}(y_i = k) l(\mathbf{x}_i, k; h_{\mathcal{S}'}) + \sum_{k\in[C]} |\eta_k(\mathbf{x}_i) - \eta_k(\mathbf{x}_j)| l(\mathbf{x}_i, k; h_{\mathcal{S}'}) \\
&= \sum_{k\in[C]} p_{y_i\sim\eta_k(\mathbf{x}_j)}(y_i = k)[l(\mathbf{x}_i, k; h_{\mathcal{S}'}) - l(\mathbf{x}_j, k; h_{\mathcal{S}'})] + \sum_{k\in[C]} |\eta_k(\mathbf{x}_i) - \eta_k(\mathbf{x}_j)| l(\mathbf{x}_i, k; h_{\mathcal{S}'}).
\end{aligned}$$

By applying the Lipschitz property of regression function and loss function, we have:

$$\begin{aligned}
&\leq \sum_{k\in[C]} p_{y_i\sim\eta_k(\mathbf{x}_j)}(y_i = k)\lambda_l|\boldsymbol{x}_i - \boldsymbol{x}_j| + \sum_{k\in[C]} l(\mathbf{x}_i, k; h_{\mathcal{S}'})\lambda_\eta|\boldsymbol{x}_i - \boldsymbol{x}_j| \\
&\leq \lambda_l|\boldsymbol{x}_i - \boldsymbol{x}_j| + \sum_{k\in[C]} l(\mathbf{x}_i, k; h_{\mathcal{S}'})\lambda_\eta|\boldsymbol{x}_i - \boldsymbol{x}_j|
\end{aligned}$$

By applying the loss bound and the p-partial r-cover constraint, we have:

$$\begin{aligned}
&\leq \lambda_l|\boldsymbol{x}_i - \boldsymbol{x}_j| + LC\lambda_\eta|\boldsymbol{x}_i - \boldsymbol{x}_j| = (\lambda_l + \lambda_\eta LC)|\boldsymbol{x}_i - \boldsymbol{x}_j| \\
&\leq r(\lambda_l + \lambda_\eta LC).
\end{aligned}$$

At last, we have the following inequality with probability p:

$$E_{y_i\sim\eta(\mathbf{x}_i)}[l(\mathbf{x}_i, y_i; h_{\mathcal{S}'})] \leq r(\lambda_l + \lambda_\eta LC). \tag{4}$$

Also, by the Hoeffding's Bound (Hoeffding, 1994), with probability at least $1 - \gamma$, we have:

$$\left|\frac{1}{n}\sum_{\boldsymbol{x},y\in\mathcal{S}} l(\boldsymbol{x}, y; h_{\mathcal{S}'}) - E_{y_i\sim\eta(\mathbf{x}_i)}[l(\mathbf{x}_i, y_i; h_{\mathcal{S}'})]\right| \leq L\sqrt{\frac{\log(1/\gamma)}{2n}}. \tag{5}$$

By combining Equation 4 and Equation 5, we have that with probability at least $1 - \epsilon$:

$$\left|\frac{1}{n}\sum_{\boldsymbol{x},y\in\mathcal{S}} l(\boldsymbol{x}, y; h_{\mathcal{S}'})\right| \leq r(\lambda_l + \lambda_\eta LC) + L\sqrt{\frac{\log\frac{p}{p+\epsilon-1}}{2n}}$$

with $p > 1 - \epsilon$. □

C.2 Proof of Proposition 1

Proposition 1. *Given a metric space (X, d), a distribution P_μ, cover percentage p, a set $\mathcal{S}$, and $f_r(p) : [0, 1] \to [0, +\infty]$ representing the mapping between p and r, if f_r is Riemann-integrable, the AUC of the p-r curve $AUC_{pr}(\mathcal{S}) = \int_0^1 f_r(p)dp = \mathbb{E}_{x\sim P_\mu}[d(S, \boldsymbol{x})]$, where $d(S, \boldsymbol{x}) = \min_{\boldsymbol{x}'\in S} d(\boldsymbol{x}', \boldsymbol{x})$.*

Proof. Since f_r is Riemann-integrable, we can calculate $\int_0^1 f_r(p)dp$, which is AUC_{pr}, with Riemann sum. Specifically, by evenly dividing $[0, 1]$ to $0 = p_0 < p_1 < p_2 < ... < p_n = 1$, we have:

$$\int_0^1 f_r(p)dp = \lim_{n\to\infty} \sum_{i=1}^{n} f_r(p_i)(p_i - p_{i-1}).$$

Since $p_i = P_\mu(d(\mathcal{S}, x) \leq f_r(p_i))$, we have

$$\int_0^1 f_r(p)dp = \lim_{n\to\infty} \sum_{i=1}^{n} f_r(p_i)P_\mu(f_r(p_{i-1}) < d(\mathcal{S}, x) \leq f_r(p_i))$$

We know f_r is an increasing function and $f_r(0) = 0$. Let $\Delta_i = f_r(p_i) - f_r(p_{i-1})$, we have:

$$= \lim_{n\to\infty} \sum_{i=1}^{n} (\sum_{j=1}^{i} \Delta_j) P_\mu(f_r(p_{i-1}) < d(\mathcal{S}, x) \leq f_r(p_{i-1}) + \Delta_i)$$

As $n \to \infty$, we have $\Delta_i \to 0$, so we have:

$$= \lim_{\Delta\to 0} \int_0^\infty rP_\mu(r \leq d(\mathcal{S}, \boldsymbol{x}) < r + \Delta)dr,$$

which is the expectation of the minimum distance to data in $\mathcal{S}$.

So we conclude that $AUC_{pr}(\mathcal{S})$ is the expectation of the minimum distance to data in $\mathcal{S}$.

□

D Additional evaluation results

D.1 Evaluation results on SVHN and CINIC10

In this section, we present our comparison experiment on SVHN and CINIC10 in Table 5 and Table 6, respectively. Since SVHN classification is simpler than other classification tasks, we also study the performance of coreset with a 95% pruning rate. We find similar comparison results that we discussed in Section 4.1. For both datasets, we observe that CCS achieves comparable accuracy with low pruning rates, but higher accuracy than random and other baseline coreset selection methods.

D.2 Coreset coverage analysis

Figure 6 and Table 7 compare AUC_{pr} between CCS with forgetting score and the vanilla forgetting methods (pruning easy first) on the CIFAR10 dataset. We observe that CCS has a lower AUC_{pr} and better accuracy with the same pruning rate, which suggests that the proposed coverage-centric method does improve the data coverage of the coresets.

D.3 Transferability of coresets

Similar to Coleman et al. (2019), we also conduct a coreset transferability study. We train a ResNet18 model with the entire CIFAR10 dataset and use the forgetting score to select coresets with different pruning rates. Then we train the ResNet50 model with these coresets. The evaluation results are shown in Table 8. We find that coresets show good transferability across these two architectures.

Table 5: Accuracy results on SVHN comparing CCS against other baseline methods. Pruning rate is the fraction of examples that are removed from the original dataset. We report the testing accuracy of the model trained on the pruned dataset. The model trained with the entire dataset has 96.12% accuracy. When we prune out 30% or 40% fraction of data, CCS achieves comparable accuracy. With a 70%, 90%, or 95% pruning rate, CCS outperforms baseline methods by a large margin.

Pruning Rate	30%	50%	70%	90%	95%
Random	$95.38_{\pm 0.4}$	$94.81_{\pm 0.24}$	$93.39_{\pm 0.27}$	$89.58_{\pm 0.99}$	$85.34_{\pm 1.01}$
Entropy	$95.47_{\pm 0.45}$	$94.95_{\pm 0.14}$	$93.14_{\pm 0.55}$	$81.6_{\pm 1.65}$	$54.6_{\pm 3.88}$
Forgetting	$95.57_{\pm 0.27}$	$95.37_{\pm 0.29}$	$94.76_{\pm 0.09}$	$66.57_{\pm 2.83}$	$27.95_{\pm 1.21}$
EL2N	$\mathbf{96.05_{\pm 0.13}}$	$95.61_{\pm 0.25}$	$94.87_{\pm 0.28}$	$56.53_{\pm 1.68}$	$21.05_{\pm 1.85}$
AUM	$95.91_{\pm 0.38}$	$\mathbf{95.76_{\pm 0.10}}$	$\mathbf{94.93_{\pm 0.17}}$	$52.58_{\pm 5.98}$	$21.61_{\pm 1.68}$
Forgetting (CCS)	$95.96_{\pm 0.1}$	$95.5_{\pm 0.15}$	$94.56_{\pm 0.24}$	$91.49_{\pm 0.89}$	$87.78_{\pm 0.8}$
EL2N (CCS)	$95.42_{\pm 0.16}$	$95.59_{\pm 0.17}$	$95.05_{\pm 0.14}$	$92_{\pm 0.60}$	$88.55_{\pm 0.78}$
AUM (CCS)	$96.02_{\pm 0.19}$	$95.69_{\pm 0.18}$	$94.88_{\pm 0.41}$	$\mathbf{92.27_{\pm 0.67}}$	$\mathbf{88.86_{\pm 1.11}}$

Table 6: Accuracy results on CINIC10 comparing CCS against other baseline methods. Pruning rate is the fraction of examples that are removed from the original dataset. We report the testing accuracy of the model trained on the pruned dataset. The model trained with the entire dataset has 89.97% accuracy. When we prune out 30% or 40% fraction of data, CCS achieves comparable accuracy. With a 70%, 80%, or 90% pruning rate, CCS outperforms baseline methods by a large margin.

Pruning Rate	30%	50%	70%	80%	90%
Random	$88.88_{\pm 0.07}$	$87.64_{\pm 0.11}$	$85.13_{\pm 0.17}$	$82.63_{\pm 0.14}$	$77.31_{\pm 0.66}$
Entropy	$89.08_{\pm 0.11}$	$86.69_{\pm 0.21}$	$82.1_{\pm 0.15}$	$77.37_{\pm 0.51}$	$67.21_{\pm 0.89}$
Forgetting	$90.07_{\pm 0.11}$	$\mathbf{89.68_{\pm 0.16}}$	$81.96_{\pm 1.07}$	$61.26_{\pm 1.85}$	$45.06_{\pm 1.48}$
EL2N	$89.93_{\pm 0.11}$	$87.65_{\pm 0.25}$	$65.10_{\pm 0.71}$	$36.09_{\pm 1.37}$	$18.50_{\pm 0.25}$
AUM	$\mathbf{90.11_{\pm 0.04}}$	$89.5_{\pm 0.16}$	$66.23_{\pm 0.69}$	$33.89_{\pm 0.7}$	$18.61_{\pm 0.15}$
Forgetting (CCS)	$90.03_{\pm 0.14}$	$89.47_{\pm 0.08}$	$\mathbf{88.04_{\pm 0.04}}$	$85.88_{\pm 0.20}$	$81.04_{\pm 0.94}$
EL2N (CCS)	$89.52_{\pm 0.11}$	$88.41_{\pm 0.17}$	$87.76_{\pm 0.12}$	$\mathbf{86.20_{\pm 0.17}}$	$\mathbf{83.18_{\pm 0.56}}$
AUM (CCS)	$90.04_{\pm 0.09}$	$89.24_{\pm 0.11}$	$87.99_{\pm 0.04}$	$85.82_{\pm 0.27}$	$81.92_{\pm 0.44}$

Table 7: AUC_{pr} comparison between the vanilla forgetting method and CCS. Coresets obtained by CCS show a lower AUC_{pr}, which suggests better data coverage.

Pruning Rate	30%	50%	70%	80%	90%
Forgetting	0.492	0.511	0.551	0.580	0.631
Forgetting (CCS)	0.491	0.504	0.529	0.547	0.592

Table 8: We train ResNet50 with coresets of CIFAR10 selected by the forgetting score calculated on training information with ResNet18. The evaluation results shows that the coresets selected by CCS have good transferability across models.

Pruning Rate	30%	50%	70%	80%	90%
Random	94.53	92.87	89.03	88.48	78.94
Forgetting	95.22	94.67	91.95	71.10	35.21
Forgetting (CCS)	95.08	94.99	92.84	89.59	85.65

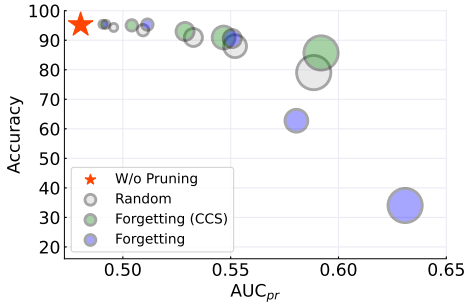


Figure 6: The relationship between AUC$_{pr}$ and accuracy. A larger circle size indicates a higher pruning ratio on the dataset. Compared to the vanilla forgetting method, CCS with the forgetting score achieves a lower AUC$_{pr}$ and better accuracy.