DeepCore: A Comprehensive Library for Coreset Selection in Deep Learning

Chencheng Guo[1†], Bo Zhao[2†], and Yanbing Bai[1⋆]

[1] Center for Applied Statistics, School of Statistics, Renmin University of China
[2] School of Informatics, The University of Edinburgh
[†] Equal contribution
Correspondence to: Yanbing Bai <ybbai@ruc.edu.cn>

Abstract. Coreset selection, which aims to select a subset of the most informative training samples, is a long-standing learning problem that can benefit many downstream learning tasks such as data-efficient learning, continual learning, neural architecture search, active learning, etc. However, most existing coreset selection methods are not designed for deep learning, which may have high complexity and poor generalization ability to unseen representations. In addition, the recently proposed methods are evaluated on models, datasets and settings of different complexities. To advance the research of coreset selection in deep learning, we provide an empirical study on popular coreset selection methods on CIFAR10 and ImageNet datasets and contribute a comprehensive code library, namely, *DeepCore*. We find although state-of-the-art methods perform well in their own experiment settings, random selection is still a strong baseline and few coreset selection methods can outperform it, when they are fairly competed in the same experiment settings of image classification. The code has been released in `https://github.com/PatrickZH/DeepCore`.

Keywords: Coreset selection · Data-efficient learning · Deep learning · CIFAR10 · ImageNet

1 Introduction

Deep learning has shown unprecedented success in many research areas such as computer vision, etc. As it evolves, not only the neural networks but also the training datasets are becoming increasingly larger, which requires massive memory and computation to achieve the state-of-the-art. One promising technique to reduce the computational cost is coreset selection [29,21,34,20] that aims to select a small subset of the most informative training samples $\mathcal{S}$ from a given large training dataset $\mathcal{T}$ in order to achieve data-efficient learning. The models trained on the coreset are supposed to have close generalization performance to those trained on the original training dataset.

⋆ This research was supported by Public Computing Cloud, Renmin University of China.

Coreset, is an important tool in many aspects of machine learning, and has been widely studied since the era of traditional machine learning, whose research generally focus on how to approximate the distribution of the whole dataset with a subset, for example, they assume that data are from a mixture of Gaussians in a given metric space [47,7,14,4,3]. Recently, a few new researches on deep learning emerge and have become of research interest [45,34,21].

However, for those classic coreset selection methods proposed in conventional machine learning, their effectiveness in deep learning is doubtful, due to the high computational complexity and fixed data representations. In addition, the newly developed coreset selection methods are evaluated on different settings on deep learning in terms of models, datasets, tasks or hyperparameters, resulting in their performance hardly being compared fairly.

To address the above problems, in this paper, we for the first time provide an exhaustive empirical study on popular coreset selection methods in the same settings. We also contribute a comprehensive code library, namely *DeepCore*, for advancing the research of coreset selection in deep learning. Specifically, we re-implement 12 popular coreset selection methods in a unified framework based on PyTorch [33]. These methods are compared in settings of various fractions from 0.1% to 90% on CIFAR10 [25] and ImageNet-1K [35] datasets. Besides the reported results in the paper, our library supports most popular deep neural architectures, image classification datasets and coreset selection settings. The results show that submodular functions maximization generally has the best performance, and in some scenarios, random selection is enough to provide outstanding results.

2 Review of Coreset Selection Methods

In this section, we first introduce the problem of coreset selection and the notations. Then, the surveys of methods and applications of coreset selection are provided respectively.

2.1 Problem Statement

In a learning problem, we are given a large training set $\mathcal{T} = \{(\mathbf{x}_i,\ y_i)\}_{i=1}^{|\mathcal{T}|}$, where $\mathbf{x}_i \in \mathcal{X}$ is the input, $y_i \in \mathcal{Y}$ is the ground-truth label of $\mathbf{x}_i$, where $\mathcal{X}$ and $\mathcal{Y}$ denote the input and output spaces, respectively. Coreset selection aims to find the most informative subset $\mathcal{S} \subset \mathcal{T}$ with the constraint $|\mathcal{S}| < |\mathcal{T}|$, so that the model $\theta^{\mathcal{S}}$ trained on $\mathcal{S}$ have comparable generalization performance to the model $\theta^{\mathcal{T}}$ trained on the whole training set $\mathcal{T}$.

2.2 Survey: Methodologies

Geometry Based Methods It is assumed that data points that are close to each other in the feature space tend to have similar properties. Therefore, geometry based methods [8,37,42,1] try to remove those data points providing

redundant information then the left data points form a coreset $\mathcal{S}$ where $|\mathcal{S}| \ll |\mathcal{T}|$.

HERDING. HERDING selects data points based on the distance between the coreset center and class center in the feature space. The algorithm incrementally and greedily adds one sample each time into the coreset that can minimize distance between two centers [47,8].

K-CENTER GREEDY. This method greedily solves the *minimax facility location* problem [13], i.e. selecting k samples as $\mathcal{S}$ from the full dataset $\mathcal{T}$ such that the largest distance between a data point in $\mathcal{T}\backslash\mathcal{S}$ and its closest data point in $\mathcal{S}$ is minimized:

$$\min_{\mathcal{S}\subset\mathcal{T}} \max_{\mathbf{x}_i\in\mathcal{T}\backslash\mathcal{S}} \min_{\mathbf{x}_j\in\mathcal{S}} \mathcal{D}(\mathbf{x}_i, \mathbf{x}_j), \tag{1}$$

where $\mathcal{D}(\cdot,\cdot)$ is the distance function. The problem is NP-hard, and a greedy approximation known as K-CENTER GREEDY has been proposed in [37]. K-CENTER GREEDY has been successfully extended to a wide range of applications, for instance, active learning [37], Small-GAN [42], and Contextual Diversity [1].

Uncertainty Methods Samples with poorer confidence have greater impact on model optimization than those with higher confidence, and should therefore be included in the coreset. The following are commonly used metrics of sample uncertainty under the current classifier at same training epoch, where C is the number of classes, and we select samples in descending order of the scores [10]:

$$\begin{aligned} s_{least\ confidence}(\mathbf{x}) &= 1 - \max_{i=1,\dots,C} P(\hat{y}=i|\mathbf{x}) \\ s_{entropy}(\mathbf{x}) &= -\sum_{i=1}^{C} P(\hat{y}=i|\mathbf{x}) \log P(\hat{y}=i|\mathbf{x}) \\ s_{margin}(\mathbf{x}) &= 1 - \min_{y\neq\hat{y}}(P(\hat{y}|x) - P(y|x)). \end{aligned} \tag{2}$$

Importance based Methods In a dataset, training samples may have different importance for training neural networks. Importance can be measures by the loss and gradient of each sample or its influence on other samples during model training. Those samples with largest importance are selected as the coreset.

Forgetting Events. Toneva et al. [45] count how many times the *forgetting* happens during the training, i.e. the misclassification of a sample in the current epoch after having been correctly classified in the previous epoch, formally $acc_i^t > acc_i^{t+1}$, where acc_i^t indicating the correctness (True or False) of the prediction of sample i at epoch t. The number of forgetting reveals intrinsic properties of the data, allowing for the removal of unforgettable example with minimal performance drop.

GRAND *and* EL2N *Scores.* The GRAND score [34] of sample i at epoch t is defined as

$$\chi_t(\mathbf{x}_i, y_i) \triangleq \mathbb{E}_{\theta_t} ||\nabla_{\theta_t} \ell(\mathbf{x}, y;\ \theta_t)||_2. \tag{3}$$

It measures the average contribution from each sample to the decline of the training loss at early epoch t across several different independent runs. The score calculated at early training stages, e.g. after a few epochs, works well, thus this method requires less computational cost. An approximation of the GraNd score is also provided, named EL2N score, which measures the norm of error vector:

$$\chi_t^*(\mathbf{x}_i, y_i) \triangleq \mathbb{E}_{\theta_t} ||p(\theta_t, \mathbf{x}_i) - y_i||_2. \tag{4}$$

Importance Sampling. In importance sampling (or adaptive sampling), we define $s(\mathbf{x}_i,\ y_i)$ is the upper-bounded (worst-case) contribution to the total loss function from the i-th data point, aka sensitivity score. It can be formulated as:

$$s(\mathbf{x},\ y) = \max_{\theta \in \Theta} \frac{\ell(\mathbf{x},\ y;\ \theta)}{\sum_{(\mathbf{x}',\ y') \in \mathcal{T}} \ell(\mathbf{x}',\ y';\ \theta)}, \tag{5}$$

where $\ell(\mathbf{x},\ y)$ is a non-negative cost function with parameter $\theta \in \Theta$. For each data point in $\mathcal{T}$, the probability of being selected is set as $p(\mathbf{x},\ y) = \frac{s(\mathbf{x},\ y)}{\sum_{(\mathbf{x},\ y) \in \mathcal{T}}^{n} s(\mathbf{x},\ y)}$. The coreset $\mathcal{S}$ is constructed based on the probabilities [3,31]. Similar ideas are proposed in *Black box learners* [11] and Jtt [26], where wrongly classified samples will be upweighted or their sampling probability will be increased.

Decision Boundary Based Methods Since data points distributed near the decision boundary are hard to separate, those data points closest to the decision boundary can also be used as the coreset.

Adversarial DeepFool. While exact distance to the decision boundary is inaccessible, Ducoffe and Precioso [12] seek the approximation of these distance in the input space $\mathcal{X}$. By giving perturbations to samples until the predictive labels of samples are changed, those data points with smallest adversarial perturbation are closest to the decision boundary.

Contrastive Active Learning. To find data points near the decision boundary, Cal [27] selects samples whose predictive likelihoods diverge the most from their neighbors to construct the coreset.

Gradient Matching Classification models are usually trained using (stochastic) gradient descent algorithm. Therefore, we expect that the gradients produced by the full training dataset $\sum_{(\mathbf{x},\ y) \in \mathcal{T}} \nabla_\theta \ell(\mathbf{x},\ y;\ \theta)$ can be replaced by the (weighted) gradients produced by a subset $\sum_{(\mathbf{x},\ y) \in \mathcal{S}} w_{\mathbf{x}} \nabla_\theta \ell(\mathbf{x},\ y;\ \theta)$ with minimal difference:

$$\begin{aligned} &\min_{\mathbf{w}, \mathcal{S}} \mathcal{D}\Big(\frac{1}{|\mathcal{T}|} \sum_{(\mathbf{x},\ y) \in \mathcal{T}} \nabla_\theta \ell(\mathbf{x},\ y;\ \theta), \frac{1}{|\mathbf{w}|_1} \sum_{(\mathbf{x},\ y) \in \mathcal{S}} w_{\mathbf{x}} \nabla_\theta \ell(\mathbf{x},\ y;\ \theta)\Big) \\ &\quad s.t. \quad \mathcal{S} \subset \mathcal{T}, w_{\mathbf{x}} \geq 0, \end{aligned} \tag{6}$$

where $\mathbf{w}$ is the subset weight vector, $|\mathbf{w}|_1$ is the sum of the absolute vector value and $\mathcal{D}(\cdot, \cdot)$ is a measure of distance between two gradients.

Craig. Mirzasoleiman et al. [29] try to find an optimal coreset that approximates the full dataset gradient under a maximum error ε by converting gradient

matching problem to the maximization of a monotone submodular function F and then use greedy approach to optimize F to get the coreset.

GradMatch. Compared to Craig, the GradMatch [20] method is able to achieve the same error ε of the gradient matching but with a smaller subset. GradMatch introduces a squared l2 regularization term over the weight vector $\mathbf{x}$ with coefficient λ to discourage assigning large weights to individual samples. To solve the optimization problem, it presents a greedy algorithm – *Orthogonal Matching Pursuit*, which can guarantee $1-exp(\frac{-\lambda}{\lambda+k\nabla^2_{max}})$ error with the constraint $|\mathcal{S}| \leq k$, k is a preset constant.

Bilevel Optimization based Methods Coreset selection can be posed as a bilevel optimization problem. Existing studies usually consider the selection of subset (optimization of samples $\mathcal{S}$ or weights $\mathbf{w}$) as the outer objective and the optimization of model parameters θ as the inner objective. Representative methods include cardinality-constrained BO [5] for continual learning, Retrieve for semi-supervised learning [22], and Glister [21] for supervised learning and active learning.

Glister. To guarantee the robustness, Glister [21] introduces a validation set $\mathcal{V}$ on the outer optimization and the log-likelihood $\ell\ell$ in the bilevel optimization:

$$\mathcal{S}^* = \arg\max_{\mathcal{S}\subset\mathcal{T}} \sum_{(\mathbf{x},\ y)\in\mathcal{V}} \ell\ell(\mathbf{x},\ y;\ \arg\max_{\theta} \sum_{(\mathbf{x},\ y)\in\mathcal{S}} \ell\ell(\mathbf{x},\ y;\ \theta)). \tag{7}$$

Submodularity based Methods Submodular functions [17] are set functions $f: 2^{\mathcal{V}} \rightarrow \mathbb{R}$, which return a real value for any $\mathcal{U} \subset \mathcal{V}$. f is a submodular function, if for $\mathcal{A} \subset \mathcal{B} \subset \mathcal{V}$ and $\forall x \in \mathcal{V}\backslash\mathcal{B}$:

$$f(\mathcal{A}\cup\{x\}) - f(\mathcal{A}) \geq f(\mathcal{B}\cup\{x\}) - f(\mathcal{B}). \tag{8}$$

Submodular functions naturally measure the diversity and information, thus can be a powerful tool for coreset selection by maximizing them. Many functions obey the above definition, e.g. Graph Cut, Facility Location, Log Determinant [16], etc. For maximizing submodular functions under cardinality constraint, greedy algorithms have been proved to have a bounded approximation factor of $1-\frac{1}{e}$ [32].

Wei et al. [46] introduce submodularity into Naive Bayes and Nearest Neighbor, and propose a novel framework for Active Learning (AL) namely Fass. Kaushal et al. [19] develop Prism, an algorithm for *targeted subset selection*, which is a learning scenario similar to AL. The subset $\mathcal{S}$ will be selected from a large unlabeled set $\mathcal{P}$ to label, but with additional requirement that $\mathcal{S}$ has to be aligned with the specific targeted set $\mathcal{T}$. Kothawade et al. [24] introduce Similar that extends submodularity to broader settings which may involve rare classes, redundancy, out-of-distribution data, etc.

Proxy based Methods Many coreset selection methods require to train models on the whole dataset for calculating features or some metrics for one or many times. To reduce this training cost, Selection via Proxy [10,36] is proposed, which trains a lighter or shallower version of the target models. Specifically, they create proxy models by reducing hidden layers, narrowing dimensions, or cutting down training epochs. Then, coresets are selected more efficiently on these proxy models.

2.3 Survey: Applications

Data-efficient Learning. The basic application of coreset selection is to enable efficient machine learning [45,29,34,20]. Training models on coreset can reduce the training cost while preserving testing performance. Especially, in Neural Architecture Search (NAS) [40], thousands to millions deep models have to be trained and then evaluated on the same dataset. Coreset can be used as a proxy dataset to efficiently train and evaluate candidates [10,36], which significantly reduces computational cost.

Continual Learning. Coreset selection is also a key technique to construct memory for continual learning or incremental learning [2,5,48]. In the popular continual learning setting, a memory buffer is maintained to store informative training samples from previous tasks for rehearsal in future tasks. It has been proven that continual learning performance heavily rely on the quality of memory, i.e. coreset [23].

Active Learning. Active learning [38,39] aims to achieve greater performance with the minimal query cost by selecting informative samples from the unlabeled pool to label. Thus, it can be posed as a coreset selection problem [46,37,12,24,27].

Besides the above, coreset selection been studied and applied in many other machine learning problems, such as robust learning against noise [30,22,24], clustering [4,3,43], semi-supervised learning [6,22], unsupervised learning [18], efficient GAN training [42], regression [31,9] etc.

3 DeepCore Library

In the literature, methods of coresets have been proposed and tested in different experiment settings, including dataset, model architecture, coreset size, augmentations, training strategy, etc. This may lead to direct comparison between testing accuracy of different algorithms unfair and conclusions untenable. For instance, some methods may have only been evaluated on MNIST, but others on massive complicated dataset ImageNet. Even though tested on the same dataset, they are likely to use different data augmentations, which significantly affect the performance. Further, it causes inconvenience for future researches to test and compare their own new algorithms to existing results, struggling with reproduction issues of existing methods.

Therefore we develop *DeepCore*, an extendable library for coreset selection on deep learning, integrating a dozen of data-efficient algorithms proposed in recent years, enabling a fair comparison of different methods in equal experimental settings to evaluate coreset algorithms in various scenarios. *DeepCore* is highly modular, allowing to add new methods easily for comparison, and other datasets or networks as well.

Coreset Methods. We list the methods in DeepCore according to the categories in 2.2, they are geometry based methods Contextual Diversity (CD) [1], herding [45] and k-Center Greedy [37], uncertainty scores (see Eq. 2), importance based methods forgetting [45] and GraNd score [34], decision boundary based methods Cal [28] and DeepFool [12] , gradient matching methods Craig [29] and GradMatch [20], bilevel optimization methods Glister [21], and submodular maximization on both Graph Cut (GC) and Facility Location (FL) functions.

Datasets. The experiments in this paper were conducted on CIFAR10 and ImageNet. In addition, it contains a series of other popular computer vision datasets, namely MNIST, QMNIST,FashionMNIST, SVHN, CIFAR100 and Tiny-ImageNet.

Network Architectures. They are two-layer fully connected MLP, LeNet, AlexNet, VGG, InceptionV3, ResNet, WideResNet and MobileNetV3.

4 Experiment Results

In this section, we conduct extensive experiments on *DeepCore* to evaluate different coreset methods on fair experimental settings to verify performance of each method.

Fraction	0.1%	0.5%	1.0%	5.0%	10.0%	20.0%	30.0%	40.0%	50.0%	60.0%	90.0%	full
random	21.03±0.3	30.82±0.6	36.74±1.7	64.50±1.1	75.72±2.0	87.10±0.5	90.23±0.3	92.15±0.1	93.28±0.2	94.00±0.2	95.19±0.1	95.65±0.2
CD	15.78±1.2	20.54±0.7	23.60±1.9	38.07±2.2	58.80±2.0	81.32±2.5	90.82±0.5	93.27±0.4	94.29±0.2	94.56±0.6	95.40±0.1	95.65±0.2
Herding	20.19±2.3	27.30±1.5	34.75±3.3	50.99±3.1	63.53±3.4	74.11±2.5	80.11±2.2	85.22±0.9	87.99±1.1	89.80±0.9	94.64±0.4	95.65±0.2
kCenterGreedy	18.45±0.3	26.76±1.2	31.11±1.2	51.37±2.1	75.83±2.4	87.05±0.3	90.86±0.4	92.83±0.1	93.91±0.2	94.07±0.1	95.39±0.1	95.65±0.2
Least Confidence	14.15±0.9	17.24±1.8	19.78±2.3	36.21±1.9	57.60±3.1	81.94±2.2	90.33±0.4	93.07±0.5	94.46±0.1	94.73±0.1	95.46±0.1	95.65±0.2
Entropy	14.64±2.2	17.54±1.3	21.09±1.3	35.32±3.0	57.58±2.8	81.93±0.4	89.83±1.6	93.22±0.2	94.44±0.3	94.97±0.1	95.38±0.1	95.65±0.2
Margin	17.17±1.1	21.73±1.6	28.22±1.0	43.37±3.3	59.88±2.9	81.70±3.2	90.87±0.4	93.03±0.2	94.34±0.3	94.82±0.3	95.49±0.1	95.65±0.2
Forgetting	20.49±1.1	30.59±1.0	37.76±1.0	62.52±2.3	75.96±1.4	**87.22±0.3**	90.40±0.3	92.37±0.2	93.45±0.2	93.42±1.0	95.45±0.1	95.65±0.2
GraNd	17.72±1.0	23.96±1.1	26.69±1.3	39.80±2.3	52.73±1.9	78.16±3.0	91.19±0.7	**93.70±0.3**	**94.60±0.1**	**95.02±0.3**	95.52±0.2	95.65±0.2
Cal	22.73±2.7	33.07±2.3	37.79±2.0	59.95±1.4	71.76±1.0	80.85±1.1	86.01±1.9	87.46±0.8	89.37±0.6	91.63±0.9	94.71±0.3	95.65±0.2
DeepFool	17.55±0.4	22.36±0.8	27.57±2.2	42.63±3.5	60.84±2.5	83.03±2.3	90.01±0.7	93.08±0.2	94.15±0.1	94.81±0.2	95.49±0.1	95.65±0.2
Craig	22.52±1.2	27.04±0.7	31.75±1.1	45.18±2.9	60.15±4.4	79.63±3.1	88.42±0.5	90.83±1.4	93.28±0.6	94.24±0.2	95.54±0.1	95.65±0.2
GradMatch	17.44±1.7	25.56±2.6	30.77±1.0	47.16±0.7	61.52±2.4	79.85±2.6	87.41±2.0	90.44±1.5	92.90±0.6	93.15±1.0	93.73±0.5	95.65±0.2
Glister	19.51±2.1	27.48±1.4	32.93±2.4	50.68±1.5	66.26±3.5	84.84±0.9	90.86±0.3	92.97±0.2	94.04±0.3	94.82±0.2	**95.59±0.2**	95.65±0.2
FL	22.29±2.0	31.65±0.6	38.87±1.4	60.79±2.5	74.74±1.3	85.63±1.9	**91.39±0.4**	93.16±0.3	93.93±0.2	94.46±0.3	95.49±0.2	95.65±0.2
GC	**24.25±1.5**	**34.90±2.3**	**42.78±1.3**	**65.70±1.2**	**76.65±1.5**	84.01±0.5	87.75±0.4	90.62±0.3	93.21±0.3	94.43±0.3	95.35±0.1	95.65±0.2

Table 1. Testing accuracy (%) results of DeepCore on ResNet-18 and CIFAR10 with 11 different coreset fractions.

4.1 CIFAR10 Results

We have trained convolutional neural networks of ResNet-18 on CIFAR10 and ImageNet with various coreset methods and various coreset sizes to comprehensively evaluate each methods in fair scenarios. For CIFAR10, we use SGD as the

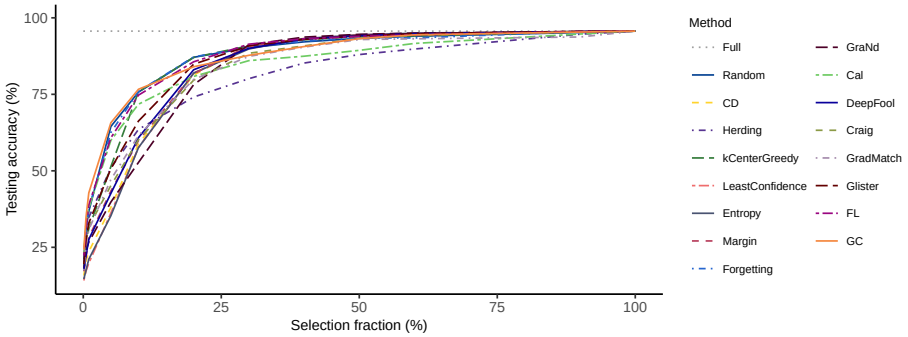

Fig. 1. Coreset selection performances. We train randomly initialized ResNet-18 on the selected coresets of CIFAR10 and then test on the real testing set.

optimizer with batch size 128, initial learning rate 0.1 with cosine decay scheduler, momentum 0.9 and weight decay 5×10^{-4} training for 200 epochs, at subset fractions of 0.1%, 0.5%, 1%, 5%, 10%, 20%, 30%, 40%, 50%, 60%, 90%. To serve as benchmarks, we also run the training results with the full data set (without pruning), as well as the results with random sampling at each fraction. When training on subsets, all images were padded by 4 pixels on all sides, randomly cropped to 32×32, and left-right flipped with probability half.

For most of the algorithms, information such as the gradient, predicted probability, or feature vector of each sample may be required for extraction, so images may need to be fed into a pretrained model to obtain this information. When required, to ensure a fair comparison, we obtain pretrained ResNet-18 models all by training 10 epochs on the full dataset following the same training strategy except without data augmentation. If gradient vector $\nabla_\theta \ell(\mathbf{x}, y; \theta)$ is required, we use the gradient of the parameters of the last fully connected layer $\nabla_{\theta_L} \ell(\mathbf{x}, y; \theta)$ to capture, as many past studies [29,20,21] suggested. This allows gradient vectors easily obtained without any back propagation. While DeepCore support both balance and imbalance selection, experiments in this paper all adopt balance selection, where fraction of selection is assigned equally per class.

Tab. 1 shows the final testing accuracy results on CIFAR10. All results are average value from 5 different random seeds. The best experimental results come from the submodular function maximization group, with both functions performing well under both large and small blocks, especially in small fractions of 0.1%-1%, significantly better than any other method. Graph Cut is more prominent among both functions, and provides outstanding results between 0.1% and 10%. In particular, Graph Cut outperforms the other methods by more than 5% when 50 samples are selected per class. Cal shows superiority in small fractions such as 0.1%-5%, with performance equivalent to Facility Location, yet it gets poorer as the fraction increases, especially at fractions greater than 30%. Except for the above, all other methods fails to compete random sampling at small coreset fractions. Between 10% and 30%, forgetting stands out among all methods, but is

also on a par with random sampling. Between 40% and 60%, we see improvement of some methods, most notably in *GraNd* and uncertainty scores. Meanwhile, FL starts outperform GC, also a prominent result at this interval. Between 60%-90%, GLISTER, based on bilevel optimization, and GRAND performs the best. Other good results at this interval can be found at DEEPFOOL, CRAIG and three uncertainty scores. In this interval, except for CAL, all other methods have surpassed random sampling, meaning CAL is less capable of selecting representative samples with large subsets. In all fraction settings, GRADMATCH and HERDING barely compete random sampling. For GRADMATCH, the experiment setting in the original paper was adaptive sampling, where subsets iteratively updated as training proceeded. Here, to make a fair comparison, coresets are fixed in all experiments.

4.2 ImageNet Results

	0.001	0.005	0.01	0.05	0.1	0.3	full
random	0.76±0.01	3.78±0.14	8.85±0.46	40.09±0.21	**52.10±0.22**	**64.11±0.05**	69.78±0.03
CD	-	1.18±0.06	2.16±0.18	25.82±2.02	43.84±0.12	62.13±0.45	69.78±0.03
Herding	0.34±0.01	1.70±0.13	4.17±0.26	17.41±0.34	28.06±0.05	48.58±0.49	69.78±0.03
kCenterGreedy	-	1.55±0.12	2.96±0.24	27.36±0.08	44.84±1.03	62.120.46±	69.78±0.03
Least Confidence	0.29±0.04	1.03±0.25	2.05±0.38	27.05±3.25	44.47±1.42	61.80±0.33	69.78±0.03
Entropy	0.31±0.02	1.01±0.17	2.26±0.30	28.21±2.83	44.68±1.54	61.82±0.31	69.78±0.03
Margin	0.47±0.02	1.99±0.29	4.73±0.64	35.99±1.67	50.29±0.92	63.62±0.15	69.78±0.03
Forgetting	0.66±0.03	3.83±0.10	9.45±0.59	40.45±1.13	51.71±0.35	63.16±0.32	69.78±0.03
GradNd	0.49±0.01	1.76±0.08	3.73±0.18	17.58±0.30	31.90±0.84	58.77±1.15	69.78±0.03
Cal	**1.29±0.09**	7.50±0.26	15.94±1.30	38.32±0.78	46.49±0.29	58.31±0.32	69.78±0.03
Craig	1.13±0.08	5.44±0.52	9.40±1.69	32.30±1.24	38.77±0.56	45.74±4.83	69.78±0.03
GradMatch	0.93±0.04	5.20±0.22	12.28±0.49	40.16±2.28	45.91±1.73	52.69±2.16	69.78±0.03
Glister	0.98±0.06	5.91±0.42	14.87±0.14	**44.95±0.28**	52.04±1.18	60.12±0.16	69.78±0.03
FL	1.23±0.03	5.78±0.08	12.72±0.21	40.85±1.25	51.05±0.59	63.14±0.03	69.78±0.03
GC	1.17±0.07	**7.66±0.43**	**16.43±0.53**	42.23±0.60	50.53±0.42	60.90±0.19	69.78±0.03

Table 2. Testing accuracy (%) results of *DeepCore* on ResNet-18 and ImageNet with 6 different coreset fractions.

For ImageNet, we train ResNet-18 network with batch size 256 for 200 epochs. While training, all images were randomly resized to 224×224 and then left-right flipped with probability of 0.5. Other experimental settings not mentioned are consistent with CIFAR10 scenarios. Due to the long execution time of DEEPFOOL on ImageNet, e.g. for a dataset of 1000 classes, each sample needs to be back-propagated 1000 times separately, no experimental data is available for the time being. For K-CENTER GREEDY and CONTEXTUAL DIVERSITY, here we do not provide the results when only 1 sample is selected from each class (i.e. fraction of 0.1%), as the first sample from each class is drawn randomly from that class. All results were repeated on 3 different random seeds.

Experiment results are shown in Tab. 2, where we witness a similar pattern of experiment results from CIFAR10. Graph Cut maximization generally outperforms all other methods with fractions between 0.1% and 5%. The next best is CAL, slightly lower than GC, which is consistent to CIFAR10 results, then followed by Facility Location maximization. Slightly different from CIFAR10, GLISTER provides relatively good results when fractions are between 0.1% and 5%.

Also, gradient matching methods CRAIG outperforms random sampling when fractions below 5%.

4.3 Cross-architecture Experiments

To examine whether methods with good performance are model-agnostic, i.e., whether coresets perform equivalently well when selections are done on various model architectures, we conduct cross-architecture experiments. Four representative methods, four neural network architectures (VGG-16 [41], Inception-v3 [44], ResNet-18 [15] and WideResNet-16-8 [49]) and two fractions (1% and 10%) are chosen for the experiment. All other unspecified settings are aligned with 4.1. In Tab. 3, the rows represent models used to obtain coresets, and the columns indicate models on which coresets are trained. We can see submodular selection with Graph Cut provides stably good testing results, regardless of what model architecture is used to perform the selection. Also, *forgetting events* are stable among different models, illustrating that forgetting events of samples are intrinsic characteristic of samples themselves. But GRADND show preferences of models on which gradient norms are computed. Coresets initialized on Inception-v3 generally have the best performance, and ResNet-18 the least. GLISTER is also strongly influenced by the model architecture for the selection. In our experiment, coresets from Inception-v3 significantly outperform coresets from other architectures.

C\T	VGG-16	Inception-v3	ResNet-18	WRN-16-8	VGG-16	Inception-v3	ResNet-18	WRN-16-8
Random			**1%**				**10%**	
Random Coreset	15.36±2.03	32.98±1.20	36.74±1.69	45.77±1.17	78.03±0.92	76.01±0.82	75.72±2.02	82.72±0.54
Forgetting			**1%**				**10%**	
VGG-16	**19.91±7.67**	**33.14±2.73**	37.14±0.97	**47.3±0.98**	77.35±1.64	**75.78±0.90**	**76.86±0.90**	**82.62±0.34**
Inception-v3	19.24±3.88	32.08±1.30	36.12±1.09	46.32±1.03	**78.28±0.50**	75.34±2.02	75.54±0.90	82.49±0.26
ResNet-18	17.14±3.00	32.55±2.22	37.76±1.01	46.70±0.95	77.73±1.09	75.24±1.17	75.96±1.39	82.41±0.25
WRN-16-8	15.58±2.78	32.83±0.91	**38.03±1.86**	46.89±1.28	77.97±0.36	75.42±1.75	76.64±0.87	82.45±0.27
GraNd			**1%**				**10%**	
VGG-16	**18.61±3.84**	29.78±0.90	33.77±0.87	38.07±1.75	69.74±1.48	65.9±1.88	65.45±1.33	76.63±0.74
Inception-v3	15.94±2.50	**31.46±0.98**	**34.73±1.04**	**40.16±1.83**	**73.51±0.75**	**70.52±3.15**	**70.07±2.91**	**79.62±1.27**
ResNet18	14.42±3.10	25.91±1.59	26.69±1.30	30.40±0.75	61.05±1.91	58.48±3.95	52.73±1.86	70.96±1.14
WideResNet-16-8	14.59±4.03	28.68±1.43	32.30±1.87	35.88±3.18	61.49±1.81	57.19±2.42	57.82±2.27	69.19±1.92
Glister			**1%**				**10%**	
VGG-16	14.5±3.86	31.08±2.30	34.10±1.71	39.45±2.55	71.71±1.83	70.23±1.78	69.31±2.19	77.74±0.68
Inception-v3	**19.74±4.01**	**32.05±1.12**	**35.52±2.09**	**41.24±1.39**	**73.15±1.94**	**71.32±1.77**	**71.03±1.39**	**78.57±1.45**
ResNet-18	15.16±4.47	30.41±2.08	32.93±2.36	37.64±1.83	67.37±2.48	66.34±2.18	66.26±3.47	75.36±1.52
WRN-16-8	14.16±4.15	28.39±2.50	32.83±0.98	37.05±2.72	70.70±2.40	64.25±2.53	66.88±2.97	75.07±2.96
Graph Cut			**1%**				**10%**	
VGG-16	27.47±4.00	37.38±2.09	**43.02±1.3**	51.80±0.82	**77.91±0.71**	**76.64±1.25**	**78.66±0.55**	**81.06±0.78**
Inception-v3	25.00±3.91	37.26±1.23	42.06±0.69	51.67±1.20	75.15±1.09	73.69±1.42	75.49±0.91	78.33±0.40
ResNet-18	**29.01±3.63**	37.54±0.62	42.78±1.30	51.50±1.37	75.29±1.05	73.94±1.11	76.65±1.48	79.13±0.75
WRN-16-8	22.64±3.82	**37.71±1.73**	40.78±1.79	**53.02±1.80**	76.64±0.92	75.84±0.84	77.19±1.14	80.77±0.30

Table 3. Cross-architecture performance in testing accuracy (%) for four representative methods (forgetting, GRAND, GLISTER and submodular maximization with Graph Cut) and random coresets.

4.4 Pretrain Model Experiments

As previously mentioned, some coreset methods rely on a pretrained model to obtain crucial information for constructing the coreset, e.g. gradients, predictive likelihoods. This experiment explores the influence of numbers of epochs to obtain the pretrained models over the final performance. As with 4.3, four representative methods and two fractions (1% and 10%) were used for the experiment. Except for the training epochs of the pretrained model, all other settings are consistent with 4.1. We report our results in Tab. 4. For forgetting, the best results occur with 2 epochs, i.e. selection based on whether the first forgetting event occurs on each sample. GraNd also tops at the second epoch.

Pre-train Epo.	0	1	2	5	10	15	20	50	100	150	200
						1%					
Forgetting	36.74±1.69	38.31±1.15	38.35±1.96	36.99±2.02	37.76±1.01	37.85±1.97	38.14±1.19	38.81±1.73	39.48±1.41	36.83±1.81	38.28±1.72
GraNd	28.17±0.20	31.05±1.36	31.24±2.36	29.70±1.02	26.69±1.30	26.11±1.46	26.39±0.89	26.81±1.97	26.52±1.10	25.89±0.56	27.17±1.84
Glister	27.63±0.85	33.97±2.68	33.31±1.08	32.93±1.51	32.93±2.36	32.28±2.09	31.15±2.24	31.46±1.56	32.89±1.24	33.37±1.91	34.06±2.17
Graph Cut	33.61±1.40	43.15±1.31	43±0.76	44.33±1.55	42.78±1.30	41.33±2.01	41.3±2.80	42.23±1.72	40.46±0.93	41.74±1.46	40.53±2.27
						10%					
Forgetting	75.96±1.39	76.57±1.44	77.4±0.85	76.43±1.09	75.96±1.39	76.4±1.2	76.49±0.74	76.7±1.7	76.52±1.47	77.06±1.62	77.26±1.03
GraNd	62.54±2.15	63.15±1.99	71.34±1.82	67.97±1.86	52.73±1.86	64.76±1.83	65.20±1.21	66.33±2.29	57.21±1.75	58.36±1.49	65.34±0.55
Glister	59.35±2.31	60.83±3.18	68.79±1.15	68.81±2.75	66.26±3.47	61.99±3.05	68.03±1.72	65.05±1.66	66.26±2.92	68.16±2.78	68.16±3.03
Graph Cut	63.39±1.54	62.52±1.02	68.26±1.11	72.91±1.13	76.65±1.48	77.06±1.09	68.73±0.87	77.48±0.51	76.66±1.64	76.16±2.14	76.33±1.52

Table 4. Testing accuracy (%) when we change the numbers of epochs to obtain the pretrain model for the selection.

5 Conclusion

In this work, we for the first time give a comprehensive survey investigating existing coreset methods on deep learning and construct a taxonomy system based on the ideas of each algorithm, and their applications. We carried out experiments on our novel code library *DeepCore*, where we re-implemented a dozen of coreset algorithms with state-of-the-art datasets and network architectures supported, enabling a convenient and fair comparison of methods under a consistent experimental setup. Throughout experiments, we found submodular maximization methods generally perform well in coreset selection tasks, and good results can also be found on uncertainty scores, Glister, Cal, Craig, or even uniform sampling in their specific scenarios. We believe both our scalable and universal framework *DeepCore* and our empirical study findings will benefit future studies of coreset selection.

References

1. Agarwal, S., Arora, H., Anand, S., Arora, C.: Contextual diversity for active learning. In: ECCV. pp. 137–153. Springer (2020)
2. Aljundi, R., Lin, M., Goujaud, B., Bengio, Y.: Gradient based sample selection for online continual learning. Advances in Neural Information Processing Systems **32**, 11816–11825 (2019)

3. Bachem, O., Lucic, M., Krause, A.: Coresets for nonparametric estimation-the case of dp-means. In: ICML. pp. 209–217. PMLR (2015)
4. Bateni, M., Bhaskara, A., Lattanzi, S., Mirrokni, V.S.: Distributed balanced clustering via mapping coresets. In: NIPS. pp. 2591–2599 (2014)
5. Borsos, Z., Mutny, M., Krause, A.: Coresets via bilevel optimization for continual learning and streaming. Advances in Neural Information Processing Systems **33** (2020)
6. Borsos, Z., Tagliasacchi, M., Krause, A.: Semi-supervised batch active learning via bilevel optimization. In: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 3495–3499. IEEE (2021)
7. Chen, Y., Welling, M., Smola, A.: Super-samples from kernel herding. The Twenty-Sixth Conference Annual Conference on Uncertainty in Artificial Intelligence (2010)
8. Chen, Y., Welling, M., Smola, A.: Super-samples from kernel herding. arXiv preprint arXiv:1203.3472 (2012)
9. Chhaya, R., Dasgupta, A., Shit, S.: On coresets for regularized regression. In: International Conference on Machine Learning. pp. 1866–1876. PMLR (2020)
10. Coleman, C., Yeh, C., Mussmann, S., Mirzasoleiman, B., Bailis, P., Liang, P., Leskovec, J., Zaharia, M.: Selection via proxy: Efficient data selection for deep learning. In: ICLR (2019)
11. Dasgupta, S., Hsu, D., Poulis, S., Zhu, X.: Teaching a black-box learner. In: ICML. PMLR (2019)
12. Ducoffe, M., Precioso, F.: Adversarial active learning for deep networks: a margin based approach. arXiv preprint arXiv:1802.09841 (2018)
13. Farahani, R.Z., Hekmatfar, M.: Facility location: concepts, models, algorithms and case studies (2009)
14. Feldman, D., Faulkner, M., Krause, A.: Scalable training of mixture models via coresets. In: NIPS. pp. 2142–2150. Citeseer (2011)
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
16. Iyer, R., Khargoankar, N., Bilmes, J., Asanani, H.: Submodular combinatorial information measures with applications in machine learning. In: Algorithmic Learning Theory. pp. 722–754. PMLR (2021)
17. Iyer, R.K., Bilmes, J.A.: Submodular optimization with submodular cover and submodular knapsack constraints. Advances in neural information processing systems **26** (2013)
18. Ju, J., Jung, H., Oh, Y., Kim, J.: Extending contrastive learning to unsupervised coreset selection. arXiv preprint arXiv:2103.03574 (2021)
19. Kaushal, V., Kothawade, S., Ramakrishnan, G., Bilmes, J., Iyer, R.: Prism: A unified framework of parameterized submodular information measures for targeted data subset selection and summarization. arXiv preprint arXiv:2103.00128 (2021)
20. Killamsetty, K., Durga, S., Ramakrishnan, G., De, A., Iyer, R.: Grad-match: Gradient matching based data subset selection for efficient deep model training. In: ICML. pp. 5464–5474 (2021)
21. Killamsetty, K., Sivasubramanian, D., Ramakrishnan, G., Iyer, R.: Glister: Generalization based data subset selection for efficient and robust learning. In: Proceedings of the AAAI Conference on Artificial Intelligence (2021)
22. Killamsetty, K., Zhao, X., Chen, F., Iyer, R.: Retrieve: Coreset selection for efficient and robust semi-supervised learning. arXiv preprint arXiv:2106.07760 (2021)

23. Knoblauch, J., Husain, H., Diethe, T.: Optimal continual learning has perfect memory and is np-hard. In: International Conference on Machine Learning. pp. 5327–5337. PMLR (2020)
24. Kothawade, S., Beck, N., Killamsetty, K., Iyer, R.: Similar: Submodular information measures based active learning in realistic scenarios. arXiv preprint arXiv:2107.00717 (2021)
25. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
26. Liu, E.Z., Haghgoo, B., Chen, A.S., Raghunathan, A., Koh, P.W., Sagawa, S., Liang, P., Finn, C.: Just train twice: Improving group robustness without training group information. In: ICML. pp. 6781–6792 (2021)
27. Margatina, K., Vernikos, G., Barrault, L., Aletras, N.: Active learning by acquiring contrastive examples. arXiv preprint arXiv:2109.03764 (2021)
28. Margatina, K., Vernikos, G., Barrault, L., Aletras, N.: Active learning by acquiring contrastive examples. arXiv preprint arXiv:2109.03764 (2021)
29. Mirzasoleiman, B., Bilmes, J., Leskovec, J.: Coresets for data-efficient training of machine learning models. In: ICML. PMLR (2020)
30. Mirzasoleiman, B., Cao, K., Leskovec, J.: Coresets for robust training of deep neural networks against noisy labels (2020)
31. Munteanu, A., Schwiegelshohn, C., Sohler, C., Woodruff, D.P.: On coresets for logistic regression. In: NeurIPS (2018)
32. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions—i. Mathematical programming **14**(1), 265–294 (1978)
33. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)
34. Paul, M., Ganguli, S., Dziugaite, G.K.: Deep learning on a data diet: Finding important examples early in training. arXiv preprint arXiv:2107.07075 (2021)
35. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. IJCV (2015)
36. Sachdeva, N., Wu, C.J., McAuley, J.: Svp-cf: Selection via proxy for collaborative filtering data. arXiv preprint arXiv:2107.04984 (2021)
37. Sener, O., Savarese, S.: Active learning for convolutional neural networks: A core-set approach. In: ICLR (2018)
38. Settles, B.: Active learning literature survey (2009)
39. Settles, B.: From theories to queries: Active learning in practice. In: Active learning and experimental design workshop in conjunction with AISTATS 2010. pp. 1–18. JMLR Workshop and Conference Proceedings (2011)
40. Shim, J.h., Kong, K., Kang, S.J.: Core-set sampling for efficient neural architecture search. arXiv preprint arXiv:2107.06869 (2021)
41. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
42. Sinha, S., Zhang, H., Goyal, A., Bengio, Y., Larochelle, H., Odena, A.: Small-gan: Speeding up gan training using core-sets. In: ICML. PMLR (2020)
43. Sohler, C., Woodruff, D.P.: Strong coresets for k-median and subspace approximation: Goodbye dimension. In: 2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS). pp. 802–813. IEEE (2018)

44. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2818–2826 (2016)
45. Toneva, M., Sordoni, A., des Combes, R.T., Trischler, A., Bengio, Y., Gordon, G.J.: An empirical study of example forgetting during deep neural network learning. In: ICLR (2018)
46. Wei, K., Iyer, R., Bilmes, J.: Submodularity in data subset selection and active learning. In: International Conference on Machine Learning. PMLR (2015)
47. Welling, M.: Herding dynamical weights to learn. In: Proceedings of the 26th Annual International Conference on Machine Learning. pp. 1121–1128 (2009)
48. Yoon, J., Madaan, D., Yang, E., Hwang, S.J.: Online coreset selection for rehearsal-based continual learning. arXiv preprint arXiv:2106.01085 (2021)
49. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)