Journal of Machine Learning Research 25 (2024) 1-53 Submitted 9/21; Revised 1/24; Published 3/24

Data Summarization via Bilevel Optimization

Zalán Borsos* ZALAN.BORSOS@GMAIL.COM
Department of Computer Science
ETH Zurich

Mojmír Mutný MOJMIR.MUTNY@INF.ETHZ.CH
Department of Computer Science
ETH Zurich

Marco Tagliasacchi* MTAGLIASACCHI@GOOGLE.COM
Google Research

Andreas Krause KRAUSEA@ETHZ.CH
Department of Computer Science
ETH Zurich

Editor: Moritz Hardt

Abstract

The increasing availability of massive data sets poses various challenges for machine learning. Prominent among these is learning models under hardware or human resource constraints. In such resource-constrained settings, a simple yet powerful approach is operating on small subsets of the data. Coresets are weighted subsets of the data that provide approximation guarantees for the optimization objective. However, existing coreset constructions are highly model-specific and are limited to simple models such as linear regression, logistic regression, and k-means. In this work, we propose a generic coreset construction framework that formulates the coreset selection as a cardinality-constrained bilevel optimization problem. In contrast to existing approaches, our framework does not require model-specific adaptations and applies to any twice differentiable model, including neural networks. We show the effectiveness of our framework for a wide range of models in various settings, including training non-convex models online and batch active learning.

Keywords: data summarization, coresets, bilevel optimization, continual learning, streaming, batch active learning

1. Introduction

Learning models on massive data sets face several challenges. From a computational perspective, specific hardware resource constraints must be met: the data is loaded into the system's main memory with limited capacity, it is processed by algorithms with usually superlinear space or time complexity. Moreover, commonly used specialized hardware for accelerating data processing, such as GPUs, introduces another layer of constraints due to their limited memory. A simple yet powerful approach for tackling these computational challenges is to operate on small subsets of the data sampled *uniformly at random*—this idea is crucial to the success of stochastic optimization. However, real-world settings often

∗. Now at Google DeepMind

involve rare but essential events with a significant impact on the optimization objective that are unlikely to be represented in a small uniform summary—a drawback that can be remedied by constructing a more *representative* summary.

Some settings face human resource constraints. A prominent example of such a setting is batch active learning, where the human expert provides labels for a selected batch of points in each round of active learning. The cost and the limited availability of human attention impose explicit constraints on the number of points that can be labeled, accentuating the importance of compact and informative summaries.

Coresets are weighted subsets of the data that provide approximation guarantees for the optimization objective. The strongest guarantees are *uniform multiplicative* guarantees. Coresets with uniform approximation guarantees are a versatile tool for obtaining provably good solutions for optimization problems on massive data sets in batch, distributed, and streaming settings, but such strong approximation guarantees are hard or even impossible to obtain for practical coreset sizes for many relevant problems. Consequently, coresets with uniform approximation guarantees are *limited to simple models* such as linear regression, logistic regression, k-means, and Gaussian mixture models.

Similarly, existing coreset construction strategies are either *model-specific* or require model-specific derivations for instantiation. For example, the popular framework of Feldman and Langberg (2011) for constructing coresets with uniform approximation guarantees relies on importance sampling based on upper bounds on the sensitivity of data points; the derivation of non-vacuous sensitivity upper bounds is a difficult model-specific task. While several alternative coreset definitions have been proposed, no generic coreset construction approach has been shown to succeed for a wide range of models that also include neural networks. Moreover, coresets remain underexplored in practically relevant settings using data summarization, such as batch active learning, compression, and the training of non-convex models online, despite coresets being natural candidates for these settings.

In this work, we propose a *generic coreset construction framework* for twice differentiable models. We show that our method is effective for various models in various resource-constrained settings. In particular:

- We formulate the coreset selection as a decision problem mathematically equivalent to a cardinality-constrained *bilevel optimization* problem that we solve by greedy forward selection and first-order methods. In contrast to existing coreset constructions, our framework applies to *any* twice differentiable model and does not require model-specific modifications.

- We point out connections to *robust statistics* and *experimental design*, discuss theoretical guarantees for our framework, and offer several variants for improved scalability. We present various extensions, including generating joint coresets for multiple models.

- We demonstrate the advantage of our framework over other data summarization techniques in extensive experimental studies, over a wide range of models and resource-constrained settings, such as *continual learning, streaming* and *batch active learning* and *dictionary selection* for compressed sensing.

This work is a significant extension of our original conference publication (Borsos et al., 2020) that demonstrated the effectiveness of the framework for building coresets of size up to

a few hundred points for neural networks with proxy models only. We extend the framework to constructing coresets directly for the target models, without a proxy, and provide several ways to speed up the construction while maintaining its empirical effectiveness. To establish connections to prior work, we introduce several variants of the main algorithm. We demonstrate the effectiveness of our approach for models with millions of parameters, including wide residual networks, for which we show that we can compress CIFAR-10 by a factor of 2 and SVHN by a factor of 3 with less than 0.05% loss of test accuracy. Furthermore, we offer several extensions to our framework, including constructing joint coresets for multiple models and dictionary selection for compressed sensing. The batch active learning application presented in this work is based on Borsos et al. (2021) with performance improvements.

2. Background and Related Work

In this section, we present relevant works on coresets and other data summarization techniques. The list of presented approaches is by no means exhaustive—we refer to Har-peled (2011); Phillips (2016); Bachem et al. (2017); Feldman (2020) for surveys about the topic.

Coresets are commonly defined as weighted subsets of the data. The types of theoretical guarantees provided by coresets, however, vary significantly. Consequently, to match these guarantees, a wide range of coreset-construction algorithms have been proposed, most of which apply only to a specific model. The most common type of guarantees for coresets are uniform multiplicative approximation guarantees: for a given family of nonnegative real functions $\mathcal{F}$ on the space $\mathcal{X} \subseteq \mathbb{R}^d$, we want to find the coreset C as the subset of the data $X = \{x_i\}_{i=1}^n$ and the associated weights w such that for $\delta, \epsilon \in (0, 1)$, $\left|\sum_{x \in C} f(x)w(x) - \sum_{x \in X} f(x)\right| \leq \epsilon \sum_{x \in X} f(x)$ holds with probability $1 - \delta$ *uniformly* for all $f \in \mathcal{F}$—we refer to coresets with such guarantees as ϵ-coresets.

Earliest approaches for constructing ϵ-coresets appear in computational geometry (Agarwal et al., 2005) and rely on exponential grids (Har-Peled and Mazumdar, 2004). Feldman and Langberg (2011) provide a unified ϵ-coreset-construction framework based on importance sampling via the data points' sensitivity (Langberg and Schulman, 2010). The framework has been applied to several models such as k-median (Feldman and Langberg, 2011), logistic regression (Huggins et al., 2016), and Gaussian mixture models (Lucic et al., 2017), but its theoretical guarantees are restricted to low-complexity models since the required coreset size also depends on the pseudo-dimension of the function class $\mathcal{F}$. More importantly, the framework relies on bounding the sensitivity, a nontrivial model-specific task. The absence of these bounds renders these methods inexecutable for many relevant problems. One step towards providing a recipe for calculating the sensitivity for a larger class of models is provided by Tukan et al. (2020) for near-convex functions that include SVMs, logistic regression, and ℓ_z-regression.

Closely related to our work are coresets that require guarantees only with respect to the *optimal solution* on coreset instead of uniform guarantees. The majority of constructions providing such guarantees are deterministic, in contrast to the sampling-based sensitivity framework: Badoiu and Clarkson (2003) provide a greedy forward selection of coresets for the minimum enclosing ball problem of size independent of both n and d; based on the latter, Tsang et al. (2005) propose the Core Vector Machine, which selects a superset of support vectors for SVM in linear time, in a forward greedy manner. Clarkson (2010) points out that

these approaches are instances of the Frank-Wolfe algorithm (Frank and Wolfe, 1956). When evaluating the optimal solution found on the coreset (in the unweighted case) on the full data, Wei et al. (2015) show that, in the case of nearest neighbors and naive Bayes, the resulting set function is submodular, hence greedy selection based on marginal gains guarantees a $1 - 1/e$ approximation factor to the cost of the solution on the best coreset of a given size. Like ϵ-coresets, existing deterministic coreset construction algorithms are also highly model-specific, and no algorithm has been shown to succeed over a wide range of models.

Other notable approaches to data summarization include Hilbert coresets (Campbell and Broderick, 2019), which formulates coreset selection as a sparse vector sum approximation in a Hilbert space with a chosen inner product. The key challenge is choosing an appropriate Hilbert space such that the resulting coreset is a good summary for the original problem. Instead of selecting subsets of the data, data set distillation (Wang et al., 2018; Lorraine et al., 2020; Zhao et al., 2021) synthesizes data points by optimizing the features to obtain the summary. Whereas data set distillation creates small summaries effectively, the optimization process is computationally intensive due to the large number of parameters (number of pixels). Consequently, widely used image classification data sets (e.g., CIFAR-10, SVHN, ImageNet) cannot be compressed using existing data distillation techniques without a significant loss in test accuracy compared to training on the full data.

Subset selection has also been formulated as a bilevel optimization problem. This is implicit in the work of Wei et al. (2015), where the optimization problem can be collapsed into a single-level problem due to closed-form solutions. Explicit bilevel formulations have been explored in the context of dictionary selection (Krause and Cevher, 2010). Furthermore, Tapia et al. (2020) analyze sensor subset selection as a bilevel optimization problem. While they use a similar strategy to the one developed here, we investigate considerably different settings of weighted data summarization for a wide range of models and applications.

Techniques related to coresets have also been explored in stochastic optimization. These works aim to improve convergence by selecting better summaries for minibatches than uniform sampling. The challenge here is to design an effective but lightweight selection strategy with negligible computational cost compared to the optimization. With these considerations, Mirzasoleiman et al. (2020) propose greedy subset selection based on the submodular facility location problem. Concurrently to our work, Killamsetty et al. (2021) formulate the selection of points based on a bilevel optimization problem, the unweighted equivalent of our proposed method (with the outer objective defined on the validation). We note that their solution based on Taylor expansion is covered by our framework and is equivalent to our greedy forward selection with Hessians approximated by the identity matrix in the implicit gradient (Equation (3)).

3. Coresets via Bilevel Optimization

In this section, we present a *generic coreset construction framework* that does not rely on model-specific derivations while—in contrast to other coreset constructions—being effective for advanced models such as deep neural networks. In the design of our generic framework, we focus on approximation guarantees related to the *solution* on the coreset: informally, we define a "good" coreset for a model to be a weighted subset of the data with fixed cardinality, such that when the model is trained on the coreset, it will achieve a low loss on the full

data set. Mathematically, this formulation translates naturally into a *bilevel optimization* problem with cardinality constraints.

This section provides an overview of the generic coreset construction framework based on Borsos et al. (2020) (Sections 3.1-3.4). We then propose several principled relaxations (Section 3.5) for tackling the resulting combinatorial optimization problem directly for the target models—in contrast to Borsos et al. (2020), which relied on proxy models—and we show the empirical effectiveness of the proposed approaches in Section 5 in a variety of novel coreset settings.

3.1 Problem Setup

Let us consider a supervised setting with data set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ and let us introduce the nonnegative weight vector $w = [w_1, \ldots, w_n] \in \mathbb{R}_+^n$ for representing the coreset C_w. Here, $(x_i, y_i, w_i) \in C_w$ iff $w_i > 0$, i.e., points with 0 weights are not part of the coreset. Let us denote the model by h, its parameters by θ, and let ℓ be the loss function. Furthermore, for brevity, let $\ell_i(\theta) := \ell(h_\theta(x_i), y_i)$.

We can formalize our coreset requirement stated in the introduction: we want to find a coreset $C_{\hat{w}}$ of size m such that if we solve the weighted empirical risk minimization (ERM) problem on the coreset, $\theta^*_{\hat{w}} \in \arg\min_\theta \sum_{i=1}^n \hat{w}_i \ell_i(\theta)$, then $\theta^*_{\hat{w}}$ is a good solution for the ERM problem on the full data set $\min_\theta \sum_{i=1}^n \ell_i(\theta)$. We can write this optimization problem as

$$\begin{aligned} \hat{w} \in \underset{w \in \mathbb{R}_+^n,\, ||w||_0 \le m}{\arg\min} & \sum_{i=1}^n \ell_i(\theta^*(w)) \\ \text{s.t.} \quad & \theta^*(w) \in \arg\min_\theta \sum_{i=1}^n w_i \ell_i(\theta), \end{aligned} \tag{1}$$

where the m-sparse vector $\hat{w}$ indicates the selected points' indices and weights at nonzero positions. Although we formulated the problem in the supervised setting, the construction can be easily adapted to semi-supervised and unsupervised settings, as we will demonstrate in Section 5. Problem (1) is an instance of *bilevel optimization*, where we minimize an *outer* objective, here $\sum_{i=1}^n \ell_i(\theta^*(w))$, which in turn depends on the solution $\theta^*(w)$ to an *inner* optimization problem, here $\arg\min_\theta \sum_{i=1}^n w_i \ell_i(\theta)$. Before presenting our proposed algorithm for solving (1), we discuss some basic background on bilevel optimization.

3.2 Background on Bilevel Optimization

Modeling hierarchical decision-making processes (von Stackelberg and Peacock, 1952; Vicente and Calamai, 1994), bilevel optimization has witnessed increasing popularity in machine learning. Recently, bilevel optimization has found applications ranging from meta-learning (Finn et al., 2017; Li et al., 2017), to hyperparameter optimization (Pedregosa, 2016; Franceschi et al., 2018) and neural architecture search (Liu et al., 2019).

Suppose $g : \Theta \times \Omega \to \mathbb{R}$ and $f : \Theta \times \Omega \to \mathbb{R}$ are continuous functions, then we call

$$\begin{aligned} \min_{w \in \Omega} \quad & G(w) := g(\theta^*(w), w) \\ \text{s.t.} \quad & \theta^*(w) \in \arg\min_{\theta \in \Theta} f(\theta, w) \end{aligned} \tag{2}$$

a bilevel optimization problem with the outer (upper level) objective $\min_w g(\theta^*(w), w)$ and the inner (lower level) objective $\theta^*(w) \in \arg\min_\theta f(\theta, w)$.

Bilevel programming is generally NP-hard even in the linear case (Vicente et al., 1994). Despite the challenge of non-convexity, first-order methods for solving bilevel optimization problems are successful in many applications (Finn et al., 2017; Pedregosa, 2016; Liu et al., 2019). A common simplifying assumption for achieving asymptotic convergence guarantees is that the inner problem's solution set in Equation (2) is a *singleton* for every $w \in \Omega$ (Franceschi et al., 2018), fulfilled if f is strongly convex in θ.

First-order bilevel optimization solvers can be further categorized based on how they evaluate or approximate the implicit gradient, $\nabla_w G(w)$ for which it is necessary to consider the change of the best response θ^* as a function of w. The first class of approaches defines a recurrence relation $\theta_t = \varphi(\theta_{t-1}, w)$: the recurrence is unrolled, truncated to T steps, and $\nabla_w G(w)$ is approximated by differentiation through the unrolled iterations either by forward- or reverse-mode automatic differentiation (Franceschi et al., 2017). Whereas choosing a small number of unrolling steps T can potentially introduce bias in the gradient estimation (Wu et al., 2018), a large T can incur a high computational cost (either in time or space complexity) when Θ and Ω are high-dimensional.

The second class of first-order bilevel optimization solvers obtains the gradient implicitly: under the assumption that f is twice differentiable, the constraint $\theta^*(w) \in \arg\min_{\theta \in \Theta} f(\theta, w)$ can be relaxed to $\frac{\partial f(\theta, w)}{\partial \theta}\big|_{\theta=\theta^*} = 0$, which is tight when f is strictly convex. A key result for obtaining $\nabla_w G(w)$ is the *implicit function theorem* applied to $\frac{\partial f(\theta, w)}{\partial \theta}\big|_{\theta=\theta^*} = 0$. Combined with the total derivative and the chain rule, we get

$$\frac{\partial G(w)}{\partial w} = \frac{\partial g}{\partial w} - \frac{\partial g}{\partial \theta} \left(\frac{\partial^2 f}{\partial \theta \partial \theta^\top} \right)^{-1} \frac{\partial^2 f}{\partial \theta \partial w^\top}, \tag{3}$$

where the partial derivatives with respect to θ are evaluated at $\theta^*(w)$.

In this work, we use the framework of bilevel optimization to generate coresets. We assume that f is twice differentiable and that the inner solution set is a singleton. Due to their flexibility and scalability, we use first-order methods based on implicit gradients to solve our proposed bilevel optimization problems.

3.3 Constructing Coresets via Incremental Subset Selection (BiCo)

In the previous section, we presented different approaches for solving bilevel optimization problems. However, an additional challenge in our coreset formulation (1) is the cardinality constraint $||w||_0 \leq m$. One approach for this combinatorial problem would be to treat G as a set function and increment the set of selected points *greedily* by inspecting marginal gains. Unfortunately, for general losses, this approach comes at a high cost. At each step, we must solve the bilevel optimization problem of finding the optimal weights for each candidate point in the data set we consider adding to the coreset. This makes greedy selection based on marginal gains impractical for large models and large data sets.

Hence, our problem belongs to a rare class of problems where even the greedy algorithm is too expensive to run. In order to improve the computational complexity, an efficient solution summarized in Algorithm 1 is based on *cone constrained generalized matching pursuit* (Locatello et al., 2017). The algorithm uses a continuous relaxation and constructs the

Algorithm 1 Bilevel Coreset (BiCo)

1: **Input:** Data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, coreset size m
2: **Output:** weights w encoding the coreset
3: $w = [0, \ldots, 0]$
4: Choose $i \in [n]$ randomly, set $w_i = 1$, $S_1 = \{i\}$
5: **for** $t \in [2, ..., m]$ **do**
6: Find $w^*_{S_{t-1}} \in \mathbb{R}^n_+$ local min of $G(w)$ s.t. $\text{supp}(w^*_{S_{t-1}}) = S_{t-1}$
7: $k^* = \arg\min_{k \in [n]} \nabla_{w_k} G(w^*_{S_{t-1}})$
8: $S_t = S_{t-1} \cup \{k^*\}$, $w_{k^*} = 1$
9: **end for**
10: Find $w^*_{S_m}$ local min of $G(w)$ s.t. $\text{supp}(w^*_{S_m}) = S_m$

coresets incrementally. Consider the atom set $\mathcal{A}$ corresponding to the n data points to each we associate one standard basis vector of $\mathbb{R}^n$. Similarly to the Frank-Wolfe algorithm, generalized matching pursuit proceeds by incrementally increasing the *active set of atoms* – set of already included points in the coreset. It performs the increment by selecting the atom that minimizes the linearization of the objective at the current iteration with the current active set. Growing the atom set incrementally can be stopped when the desired size m is reached, and thus the $\|w\|_0 \leq m$ constraint is active.

To give more details, suppose a set of atoms $S_{t-1} \subset \mathcal{A}$ of size $t-1$ has already been selected. Our method proceeds in two steps. First, the bilevel optimization problem (1) is restricted to weights w having support S_{t-1}, i.e., w can only have nonzero entries at indices in S_{t-1}. Then we optimize to find the weights $w^*_{S_{t-1}}$ with support restricted to S_{t-1} that represents a local minimum of $G(w)$ defined in Eq. (2) with $g(\theta^*(w), w) = \sum_{i=1}^n \ell_i(\theta^*(w))$ and $f(\theta, w) = \sum_{i=1}^n w_i \ell_i(\theta)$—i.e., we use the algorithm's corrective variant, where, once a new atom is added, the weights are reoptimized by gradient descent using the implicit gradient with projection to nonnegative weights (line 6 in Algorithm 1). Once these weights are found, the algorithm increments S_{t-1} with the atom that minimizes the linearization of the outer objective at $w^*_{S_{t-1}}$,

$$k^* = \arg\min_{k \in [n]} e_k^\top \nabla_w G(w^*_{S_{t-1}}), \tag{4}$$

where e_k denotes the k-th standard basis vector of $\mathbb{R}^n$. In other words, the chosen point is the one with the minimum implicit gradient.

We can gain insight into the selection rule in Equation (4) by expanding $\nabla_w G$ using Equation (3). For this, we use the inner objective $f(\theta, w) = \sum_{i=1}^n w_i \ell_i(\theta)$ without regularization for simplicity. Noting that $\frac{\partial^2 \sum_{i=1}^n w_i \ell_i(\theta)}{\partial w_k \partial \theta^\top} = \nabla_\theta \ell_k(\theta)$, we can expand Equation (4) to get

$$k^* = \operatorname*{argmax}_{k \in [n]} \nabla_\theta \ell_k(\theta^*)^\top \left(\frac{\partial^2 \sum_{i=1}^n w^*_{S_{t-1},i} \ell_i(\theta^*)}{\partial \theta \partial \theta^\top} \right)^{-1} \nabla_\theta \sum_{i=1}^n \ell_i(\theta^*). \tag{5}$$

Thus, with the choice $g(\theta) = \sum_{i=1}^{n} \ell_i(\theta)$, the gradient of the selected point gradient has the largest bilinear similarity with $\nabla_\theta \sum_{i=1}^{n} \ell_i(\theta)$, where the similarity is parameterized by the inverse Hessian of the inner problem.

We note that if the measure w would be normalized, the optimization would be over the convex hull of the atoms. Furthermore, in the rare settings where $G(w)$ is convex, this approach would be equivalent to the Frank-Wolfe algorithm, already applied in the coreset literature (Clarkson, 2010).

Computational complexity. The computational cost of Algorithm 1 makes it impractical for many settings. We illustrate this with an example, where we assume that the inner optimization in line 6 of Algorithm 1 is performed by gradient descent with t_g iterations, and the weights are updated using implicit gradients t_w times. Let us denote the number of model parameters by d and the complexity of calculating the gradient on a single point by $\mathcal{O}(g)$. Furthermore, we assume additive losses to that the gradient calculation on a batch of n points takes $\mathcal{O}(ng)$.

Suppose we have already selected i coreset points. Solving the inner problem with gradient descent for given weights w is $\mathcal{O}(t_g ig)$. For obtaining the implicit gradient, calculating the Hessian can be done using efficient Hessian-vector products (Pearlmutter, 1994) in $\mathcal{O}(igd)$, while inverting it is $\mathcal{O}\left(d^3\right)$, and calculating the last term in Eq. 5 is $\mathcal{O}(ng)$; hence the implicit gradient calculation takes $\mathcal{O}\left(igd + d^3 + ng\right)$. Consequently, the complexity of building a coreset of size m with Algorithm 1 is $\mathcal{O}\left(\sum_{i=1}^{m} \left(t_w(t_g ig + igd + d^3 + ng)\right)\right) = \mathcal{O}\left(mt_w(t_g mg + mgd + d^3 + ng)\right)$. As this computational complexity is clearly prohibitive for many practical purposes, we propose several practical variants in Sec. 3.5. These practical variants rely on further approximations. First, by using *binary weights* for the coreset, the dependence on t_w is eliminated altogether (i.e., $t_w = 1$ in that case). Second, by using efficient inverse-Hessian-vector products, the dependence on d is reduced. Third, by selecting multiple coreset points to be added at once (forward selection in batches), the dependence on m is improved.

3.4 Connections and Guarantees

Before delving into the practical variants of the algorithms, we present a connection of our approach to other decision theory-based problems arising in statistics such as influence functions and experimental design. These connections demonstrate that our framework generalizes certain well-established approaches on simple linear models that study best-subset selection in classical statistics. On these simple problems, we can derive theoretical guarantees which carry over to our framework.

3.4.1 Connection to Influence Functions

Our approach is closely related to incremental subset selection via influence functions. The *empirical influence function*, known from robust statistics (Cook and Weisberg, 1980), denotes the effect of a single sample on the estimator. Influence functions have been used by Koh and Liang (2017) to understand the dependence of neural network predictions on a single training point and to generate adversarial training examples. To uncover the relationship between our method and influence functions, consider the influence of the k-th point on the outer objective. Suppose we have already selected the subset S and found the

corresponding weights w_S^*. Then, the influence of point k on the outer objective is

$$\mathcal{I}(k) := -\frac{\partial \sum_{i=1}^n \ell_i(\theta^*)}{\partial \varepsilon}\bigg|_{\varepsilon=0}, \quad \text{s.t.} \quad \theta^* = \arg\min_\theta \sum_{i=1}^n w_{S,i}^* \, \ell_i(\theta) + \varepsilon \ell_k(\theta).$$

Proposition 1 *Under twice differentiability and strict convexity of the inner loss, the choice* $\operatorname{argmax}_k \mathcal{I}(k)$ *corresponds to the selection rule in Equation* (5)*.*

According to Proposition 1, Algorithm 1 can be interpreted as incremental identification of influential points. However, as discussed in the previous section, this algorithm is computationally prohibitive for practically relevant models—we address this issue in Section 3.5.

3.4.2 Connection to Experimental Design

Let us instantiate our approach for the problem of weighted and regularized least squares regression. In this case, the inner optimization problem $\hat{\theta}(w) = \arg\min_\theta \sum_{i=1}^n w_i (x_i^\top \theta - y_i)^2 + \lambda\sigma^2 ||\theta||_2^2$, where weights are assumed to be non-zero. The problem admits a closed-form solution

$$\hat{\theta} = (X^\top D(w) X + \lambda\sigma^2 I)^{-1} X^\top D(w) y, \tag{6}$$

where $D(w) := \operatorname{diag}(w)$. If the weights are assumed to be binary, we can identify this problem to be equivalent to *optimal experimental design* (Chaloner and Verdinelli, 1995), itself being motivated by bilevel optimization problem with closed form inner optimization problem. In particular, the data summarization with the outer objective defined as $g(\hat{\theta}) = \frac{1}{2n}\mathbb{E}_{\epsilon,\theta}\left[\sum_{i=1}^n (x_i^\top \hat{\theta} - y_i)^2\right]$ is closely related to *Bayesian V-optimal design*, as the following proposition shows.

Proposition 2 *Under the Bayesian linear regression assumptions* $y = X\theta + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$ *and* $\theta \sim \mathcal{N}(0, \lambda^{-1})$*, let* $g_V(\hat{\theta}) = \frac{1}{2n}\mathbb{E}_{\epsilon,\theta}\left[\left\|X\theta - X\hat{\theta}\right\|_2^2\right]$ *be the Bayesian V-experimental design outer objective. For all* $\hat{\theta}_S$ *in Eq.* (6)*, we have*

$$\lim_{n\to\infty} g(\hat{\theta}_S) - g_V(\hat{\theta}_S) = \frac{\sigma^2}{2}.$$

Consequently, it can be argued that, in the large data limit, the optimal coreset with binary weights corresponds to the solution of Bayesian V-experimental design. Further discussion can be found in Appendix A. By using g_V as our outer objective, solving the inner objective in closed form, we identify the Bayesian V-experimental design objective,

$$G(w) = \frac{1}{2n}\operatorname{Tr}\left(X\left(\frac{1}{\sigma^2}X^\top D(w) X + \lambda I\right)^{-1} X^\top\right).$$

The literature on Bayesian experimental design suggests relaxing this objective and solving it using convex optimization methods. In Lemma 10 in Appendix A we show that $G(w)$ is smooth and convex in w when the integrality is relaxed. As our methodology is equivalent in the large data limit, our framework enjoys additive approximation guarantees for large n on models like this as we show next.

3.4.3 Theoretical Guarantees

Let $G(w)$ be smooth and convex and dependent on the w only, i.e., restricted to lie on the optimality manifold of the inner problem. If the inner objective has a closed form, as in the case of experiment design, one can obtain $G(w)$ by substituting the closed form, otherwise, this objective is only defined implicitly. It can be shown that Algorithm 1, being an instance of cone-constrained generalized matching pursuit (Locatello et al., 2017), provably converges at a rate of $\mathcal{O}(1/t)$.

Theorem 3 (cf. Theorem 2 of Locatello et al. (2017)) *Let G be L-smooth and convex. After t iterations in Algorithm 1 we have,*

$$G(w^*_{S_t}) - G^* \leq \frac{8L + 4\epsilon_1}{t+3}$$

where $t \leq m$ (number of atoms) and $\epsilon_1 = G(w^*_{S_1}) - G^*$ is the suboptimality gap at $t = 1$.

Corollary 4 *Under the conditions of Theorem 3, Algorithm 1 produces a coreset of size $m \in \mathcal{O}((L + \epsilon_1)\epsilon^{-1})$, where ϵ is the target approximation error.*

In the absence of knowledge of L, which could be large, possibly scaling with the number of model parameters, we can nevertheless show that the suboptimality decreases as $1/t$. Even though, in general, the function G might not be convex for more complex models, variants of the proposed coreset selection algorithm can be executed nevertheless, and we demonstrate their effectiveness empirically in such settings in Section 5.

3.5 Practical Bilevel Coreset Construction: Variants

In the previous sections, we presented and analyzed Algorithm 1, our basic algorithm for coreset selection. However, we have also seen its prohibitive computational complexity. In this section, we provide approximations that improve the complexity significantly and turn the algorithm into a practical one, while still being empirically effective.

3.5.1 Binary Weights, IHVP Approximations and Selection in Batches

The three approximations we consider in this section are restricting the coreset weights to binary, using the Neumann series for efficient inverse-Hessian-vector products, and selection in batches. These approximations enable us to scale our method even to neural networks with millions of parameters on large data sets, as demonstrated in Section 5.2.3.

Binary coreset weights. In Algorithm 1, the coreset weights need to be determined for every selection step, which is itself an iterative procedure. We propose to restrict the coreset weights to *binary* (unweighted coreset), which eliminates the coreset weight optimization step and reduces the number of implicit gradient calculations to the number of forward selection steps.

Inverse-Hessian-vector product (IHVP) approximations. For obtaining the implicit gradient, the Hessian calculation and inversion in Equation (3) is computationally intractable for models with a large number of parameters. Efficient inverse-Hessian-vector

product approximations can be obtained by approximately solving the linear system $\frac{\partial^2 f}{\partial\theta\partial\theta^\top}x = \left(\frac{\partial g}{\partial\theta}\right)^\top$ with conjugate gradients (Pedregosa, 2016) or by approximating the inverse Hessian with the *Neumann series* $\left(\frac{\partial^2 f}{\partial\theta\partial\theta^\top}\right)^{-1} = \lim_{T\to\infty}\sum_{i=0}^{T}\left(I - \frac{\partial^2 f}{\partial\theta\partial\theta^\top}\right)^i$ (Lorraine et al., 2020). Since Hessian-vector products can be calculated with the complexity of the backward pass (Pearlmutter, 1994), this approximation enables the scalability of first-order optimization methods based on implicit gradients for models with a large number of parameters.

Selection in batches. Since each selection step requires evaluating the implicit gradient, adding points one by one might be too costly for generating the coreset. We propose *forward selection in batches*: start with a small random subset and increase the chosen subset by a batch of b points with the smallest implicit gradient in each step.

We note that other approaches are also possible, such as exchange in batches, where we start with a random subset of the desired coreset size, remove b of the chosen points having the largest implicit gradient, and add b new points having the smallest implicit gradient in each step; and elimination in batches, where we start with the full data set, remove b of the chosen points having the largest implicit gradient in each step. We compare the three approaches (forward selection, exchange, elimination) empirically in Section 5.1.

Similar ideas are prevalent in experimental design, e.g., the "excursion" version of Fedorov's exchange algorithm (Fedorov, 1972; Mitchell, 1974). The significant difference to our approach is that, for the experimental design objectives available in closed-form, the selection algorithm can easily evaluate the exact effect of adding or removing points—in our case, this is prohibitively expensive, thus we must resort to our proposed heuristic based on first-order Taylor expansions. Furthermore, we note that we perform the selection of batches by choosing greedily the b points that have the smallest (largest, in case of elimination) implicit gradient. Exploring approaches that enforce the diversity of the points in the selected batch is a promising future direction.

Improved computational complexity. We revisit the computational complexity of $\mathcal{O}\left(mt_w(t_g mg + mgd + d^3 + ng)\right)$ of Algorithm 1 from Section 3.3 after applying the approximations proposed in this section. By using binary weights, the weight update step is eliminated, reducing the complexity by t_w. Replacing the Hessian calculation and inversion by the Neumann series approximation reduces the factor $mgd + d^3$ to $t_h mg$, where t_h is the number of terms used in the series. Furthermore, forward selection in batches of size b reduces the complexity by a factor of b, resulting in a final complexity of $\mathcal{O}\left(mb^{-1}(t_g mg + t_h mg + ng)\right)$.

3.5.2 Selection via Proxy

A simple way to generate very small coresets fast is to perform the coreset selection on a *proxy* instead of the original model. Our proposed proxy allows for efficient, provably certifiable convex solvers to be used for the inner problem.

In particular, we study the special case of bilevel coreset construction (Eq. 1) when the proxy hypothesis class is a reproducing kernel Hilbert space (RKHS). This proxy class is relevant for a wide range of models, including neural networks due to the connection between the Neural Tangent Kernel (Jacot et al., 2018) and the training of infinitely wide neural networks with batch gradient descent.

Let $\kappa(\cdot,\cdot)$ be a positive definite kernel function, and let K denote the Gram matrix associated with the data. The Nyström method provides a low-rank approximation $\hat{K}$ to K by selecting a data-dependent basis as a subset of the training data $Q \subseteq [n]$, $|Q| = q$ such that $\hat{K} = K_{[n],Q} K_{Q,Q}^{+} K_{Q,[n]}$. Given the eigendecomposition of $K_{Q,Q} = UDU^\top$, the q-dimensional Nyström feature map is given by $z(\cdot) = D^{-1/2}U^\top[\kappa(\cdot, x_i),\, i \in Q]$, such that $\hat{K}_{i,j} = z(x_i)^\top z(x_j)$. The problem of selecting the subset Q has attracted significant interest, where the prominent tool is nonuniform sampling based on leverage scores and its variants (Mahoney and Drineas, 2009). We use the simplest and computationally most efficient method of uniform sampling for selecting Q. With the Nyström approximation, Equation (1) can be rewritten as

$$\begin{aligned} \min_{w \in \mathbb{R}^n_+,\, \|w\|_0 \leq m} \quad & \sum_{i=1}^{n} \ell(\theta^{*\top} z(x_i), y_i) \\ \text{s.t.} \quad & \theta^* = \arg\min_{\theta} \sum_{i=1}^{n} w_i \ell(\theta^\top z(x_i), y_i). \end{aligned} \tag{7}$$

For common loss functions ℓ (such as cross-entropy or squared loss) the inner optimization problem is convex and smooth, which allows us to track the suboptimality gap and use a plethora of fast and provably accurate optimization algorithms suited for these well-behaved problems. However, there is a trade-off. While using the simple form of the proxy model allows us to solve the bilevel optimization problem efficiently, the discrepancy between the proxy model and the original model can result in coresets that perform poorly for the original model when the coreset is allowed to be larger than a few tens of points.

3.5.3 Bilevel Coresets via Regularization

When not restricting the coreset weights to binary, one approach for solving the cardinality-constrained bilevel optimization for coreset selection (Equation (1)) is to transform the $\|w\|_0$ constraint into a sparsity-inducing regularizer, for example into an L_1-penalty in the spirit of Lasso (Tibshirani, 1996). However, this approach fails for Equation (1), since the solution of the inner optimization problem is a minimizer also when the weights are rescaled by a common factor.

We propose the following regularized version of the problem. First, we restrict the weights to the n-dimensional simplex Δ_n, such that $\sum_{i=1}^n w_i = 1$. Now, since $\|w\|_1 = 1$, we should use another sparsity-inducing penalty in the outer loss: any $L_q = \sum_{i=1}^n w_i^q$ with $q \in (0,1)$, where we choose $q = 1/2$ in this work—Figure 1 shows $L_{1/2}$ in three dimensions restricted to the simplex. Thus, our proposed regularized bilevel coreset selection problem (optional inner regularization is also supported, here exemplified with L_2 penalty) is

$$\begin{aligned} \min_{w \in \Delta_n} \quad & \sum_{i=1}^{n} \ell_i(\theta^*) + \beta \sum_{i=1}^{n} \sqrt{w_i} \\ \text{s.t.} \quad & \theta^* = \arg\min_{\theta} \sum_{i=1}^{n} w_i \ell_i(\theta) + \lambda \|\theta\|_2^2 . \end{aligned} \tag{8}$$

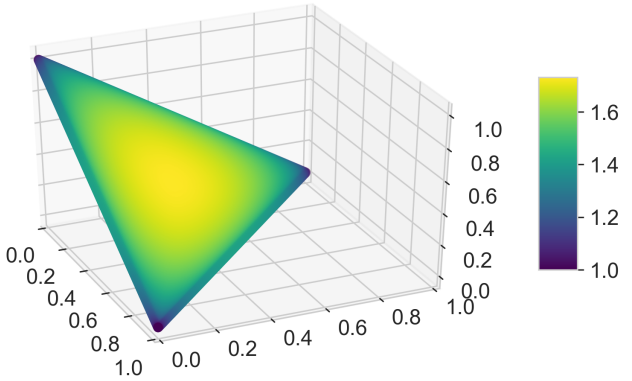


Figure 1: $L_{1/2}$ penalty in three dimensions restricted to the simplex. The value of the penalty decreases towards the edges of the simplex, inducing sparsity.

Algorithm 2 Bilevel Coreset via Regularization

Input: Data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, T, regularizers λ, β
Output: weights w encoding the coreset
$w = [1/n, \ldots, 1/n]$, $\epsilon = 10^{-8}$
for $it \in [1, \ldots, T]$ **do**
    Find $\theta^* = \arg\min_\theta \sum_{i=1}^n w_i \ell_i(\theta) + \lambda \|\theta\|_2^2$
    Update w by gradient descent using Equation (9)
    $\tilde{w} = \arg\min_{w' \in \Delta_n} \|w' - w\|_2$ ▷ Duchi et al. (2008)
    $w = (1 - \epsilon)\tilde{w} + \epsilon \mathbb{1}_n$
end for
$w[w < 10^{-4}] = 0$

For optimizing the bilevel problem in Equation (8), we apply first-order based on the implicit gradient

$$\beta \frac{\partial \sum_{i=1}^n \sqrt{w_i}}{\partial w} - \frac{\partial \sum_{i=1}^n \ell_i(\theta^*)}{\partial \theta} \left(\frac{\partial^2 \sum_{i=1}^n w_i \ell_i(\theta^*)}{\partial \theta \partial \theta^\top} \right)^{-1} \frac{\partial^2 \sum_{i=1}^n w_i \ell_i(\theta^*)}{\partial \theta \partial w^\top}. \tag{9}$$

Additionally, since the weights are constrained to Δ_n, we project the weights after each gradient descent step to Δ_n using the efficient Euclidean projection step proposed by Duchi et al. (2008). Furthermore, to ensure numerical stability of derivatives in Equation (9) due to the $L_{1/2}$-penalty, we found it useful to mix the projected weight vector with the identity vector to avoid exactly 0 weights. This extra component is smaller than our final truncation threshold. Our proposed method is summarized in Algorithm 2.

In practice, to reach a desired coreset size m, we tune our hyperparameters λ and β as follows. We first tune λ based on the validation performance by solving the inner optimization problem with $w = [1/n, \ldots, 1/n]$; after the tuning, λ will be fixed. We set β to small value, e.g., $\beta = 10^{-7}$, start the loop in Algorithm 2, and we monitor the number of selected coreset points: if the number of the selected coreset points was plateauing in recent iterations, then we increase the sparsity penalty by doubling β—we use the doubling until the desired coreset

size m is reached. Compared to the selection strategies presented in the previous sections operating with binary coreset weights, this approach has the advantage of generating weighted coresets, that are significantly more compact for some models (Section 5.1). On the other hand, its computational complexity is increased due to the fact that the implicit gradient must be calculated at every step—hence this method is only practical for simple models.

4. Extensions and Applications of Bilevel Coresets

Our framework has the advantage of flexibility in handling extensions that can be incorporated into the outer and inner objectives (Equation (1)) and is thus applicable in a wide range of settings. We present the framework's applications in continual learning and streaming based on Borsos et al. (2020), in batch active learning based on Borsos et al. (2021); we propose new applications for joint coresets construction for multiple models and dictionary selection in compressed sensing.

4.1 Continual Learning

In contrast to the standard supervised setting, where the learning algorithm has access to an i.i.d. data set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, continual learning assumes that $\mathcal{D}$ is the union of T disjoint subsets $\mathcal{D}_1, \ldots, \mathcal{D}_T$ such that each $\mathcal{D}_i$ contains data drawn from a different i.i.d. distribution. The goal is to learn a model based on the data that arrives sequentially from different tasks, such that the model achieves good performance on all tasks. An additional constraint in the setting is that the model cannot revisit all data from the previous tasks $1, \ldots, t-1$ when learning on task t. The challenge is to avoid *catastrophic forgetting* (McCloskey and Cohen, 1989; French, 1999), which occurs when the optimization on $\mathcal{D}_t$ degrades the model's performance significantly on some of $\mathcal{D}_1, \ldots, \mathcal{D}_{t-1}$.

Continual learning with neural networks has received increasing interest recently. The approaches for alleviating catastrophic forgetting fall into three main categories: weight regularization to restrict deviations from parameters learned on old tasks (Kirkpatrick et al., 2017; Nguyen et al., 2018); architectural adaptations for different tasks (Rusu et al., 2016); and replay-based approaches, where samples from old tasks are either reproduced via replay memory (Lopez-Paz and Ranzato, 2017) or generative models (Shin et al., 2017).

In this work, we focus on the replay-based approach, which provides strong empirical performance despite its simplicity (Chaudhry et al., 2019). In this setting, coresets are natural candidates for the summaries of the tasks to be stored in the replay memory, and we can readily use our coreset construction for the selection.

For continual learning with replay memory, we employ the following protocol. The learning algorithm receives data $\mathcal{D}_1, \ldots, \mathcal{D}_T$ arriving in order from T different tasks. At time t, the learner receives $\mathcal{D}_t$ but can only access past data through a small number of samples from the replay memory of size m. We assume that equal memory is allocated for each task in the buffer and that the summaries $C_1, \ldots, C_T$ are created per task. Thus, the optimization objective at time t is

$$\min_\theta \frac{1}{|\mathcal{D}_t|} \sum_{(x,y)\in\mathcal{D}_t} \ell(h_\theta(x), y) + \beta \sum_{\tau=1}^{t-1} \frac{1}{|\mathcal{C}_\tau|} \sum_{(x,y)\in\mathcal{C}_\tau} \ell(h_\theta(x), y),$$

where $\sum_{\tau=1}^{t-1} |C_\tau| = m$ and β is a hyperparameter controlling the regularization strength of the loss on the samples from the replay memory. After performing the optimization, $\mathcal{D}_t$ is summarized into $\mathcal{C}_t$ and added to the buffer, while previous summaries $\mathcal{C}_1, \ldots, \mathcal{C}_{t-1}$ are shrunk such that $|\mathcal{C}_\tau| = \lfloor m/t \rfloor$. The shrinkage is performed by running the summarization algorithms on each $\mathcal{C}_1, \ldots, \mathcal{C}_{t-1}$ again, which for greedy strategies is equivalent to retaining the first $\lfloor m/t \rfloor$ samples from each summary.

4.2 Streaming

We can also apply our coreset construction in the more challenging setting of streaming. In contrast to continual learning, the streaming setting does not define tasks and does not assume i.i.d. data in any portion of the stream. Concretely, in this work, we assume that the learner observes small data batches $\mathcal{D}_1, ..., \mathcal{D}_T$ arriving in order, where no i.i.d. and task boundary assumptions are made.

As in the case of continual learning, one approach for alleviating catastrophic forgetting in the streaming setting is the retraining on data from the memory replay buffer. Denoting by $\mathcal{M}_t$ the replay memory at time t, the optimization objective at time t for learning under streaming with replay memory is

$$\min_\theta \frac{1}{|\mathcal{D}_t|} \sum_{(x,y)\in\mathcal{D}_t} \ell(h_\theta(x), y) + \frac{\beta}{|\mathcal{M}_{t-1}|} \sum_{(x,y)\in\mathcal{M}_{t-1}} \ell(h_\theta(x), y).$$

Maintaining a coreset of constant size over data streams is a cornerstone of training nonconvex models in a streaming setting. We offer a principled way to achieve this, naturally supported by our framework, using the following idea: two coresets can be summarized into a single one by applying our bilevel construction with the outer objective as the loss on the union of the two coresets.

Based on this idea, we use a variant of the merge-reduce framework of Chazelle and Matoušek (1996). For this, we assume that we can store at most m coreset points in a buffer; we split the buffer into s slots of equal size $m_s := m/s$. We associate values β_i with each of the slots, which will be *proportional to the number of points* they represent. A new batch is compressed into a new slot with associated default β, and it is appended to the buffer, which now might contain an extra slot. The reduction to size m happens as follows: select the two consecutive slots i and $i+1$ with smallest i for which $\beta_i = \beta_{i+1}$ or, if this does not exist, choose the last two slots; then join the content of the slots (*merge*) and create the coreset of the merged data (*reduce*). The new coreset replaces the two original slots with $\beta_i + \beta_{i+1}$ associated with it. The pseudocode of the construction is shown in Algorithm 3 in Appendix D together with the illustration of the merge-reduce coreset construction for a buffer with 3 slots and 7 steps in Figure 13. The coreset produced by our construction for a two-layer fully connected neural network on the imbalanced video stream created from the iCub World 1.0 data set (Fanello et al., 2013) can be seen in Figure 2.

4.3 Batch Active Learning

The prominent use cases of our proposed method are scenarios with explicit budget constraints for subset selection. These constraints can be due to computational resource constraints, as in the case of continual learning and streaming with replay memory, or can relate

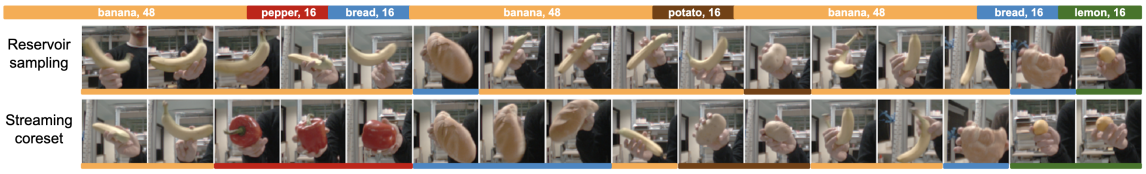


Figure 2: Data summarization on an imbalanced stream of images created from the iCub World 1.0 data set (Fanello et al., 2013). Row 1: the stream's composition containing 5 object classes. Row 2: selection by reservoir (uniform) sampling. Row 3: selection by our method. Reservoir sampling misses classes (pepper) due to imbalance and does not choose diverse samples, in contrast to our method.

to the cost of involving human interaction. *Active learning* falls into the latter category and aims to improve the data efficiency of learning algorithms by interleaving training rounds with selective query of the labels for informative unlabeled points from human experts.

The active learning setting assumes that, while unlabeled data is available abundantly, acquiring labels involves the cost of relying on human expertise. In the *pool-based* setup, each active learning round consists of training the model using the already labeled data and choosing points from the unlabeled pool for label acquisition. The challenge in this setting is to select the most informative samples, i.e., the samples with the highest potential of reducing the model's generalization error. When the cost of performing a new training round after every single acquired label is considered, active learning becomes computationally unattractive. *Batch active learning* tackles this issue by acquiring labels for a batch of points in a single round but faces the additional challenge of ensuring diversity between the chosen points.

Active learning and its batch variant have received significant attention (MacKay, 1992; Lewis and Gale, 1994; Balcan et al., 2007; Hoi et al., 2006; Guo and Schuurmans, 2008; Kirsch et al., 2019). Although vastly available unlabeled data is assumed in this setting, most active learning approaches ignore the unlabeled pool while training the model in a supervised manner on the labeled pool only. On the other hand, recent advances in semi-supervised learning (SSL) have shown significant performance improvements of models trained with only a small number of labeled samples. Prominent SSL methods in the image domain include Mean Teacher (Tarvainen and Valpola, 2017), MixMatch (Berthelot et al., 2019) and its improvement, FixMatch (Sohn et al., 2020). These methods achieve the following CIFAR-10 test accuracies: Mean teacher - 78.5% with 1000 labeled samples; MixMatch - 88.2% with 250 labeled samples; FixMatch - 95% with 250 labeled samples.

The success of semi-supervised methods suggests that using the unlabeled data pool in active learning for acquisition only is suboptimal. Based on this observation, early approaches propose to combine active learning with SSL for Gaussian fields (Zhu et al., 2003) and SVMs (Hoi and Lyu, 2005; Leng et al., 2013). In the context of semi-supervised active learning with neural networks, Sener and Savarese (2018) investigate SSL training and selecting points for label acquisition that represent the k-centers of the embeddings in the last layer. Song et al. (2019) show that training in a semi-supervised manner with MixMatch and selecting the candidates for label query with standard acquisition functions improves the active learner's generalization performance compared to uniform sampling. Gao et al. (2019) propose to train in each acquisition round with MixMatch and query the points that

produce the most inconsistent predictions when undergoing random data augmentations, as measured by the sum of per-class variances in the predicted class probabilities. We compare our proposed acquisition strategy with these methods empirically.

We propose a simple yet highly effective label acquisition strategy based on bilevel coreset construction that works in the semi-supervised batch active learning setup. The basic idea is the following: in each round of active learning, we train the semi-supervised learner and use its predictions to provide labels for the samples in the unlabeled pool; then, using these "pseudo-labels", we construct the coreset of the unlabeled pool and query the true labels for the selected points. This strategy naturally accommodates the selection of batches and prohibits redundancy in the selected batch by the design of the objective.

Let us formalize our approach in a single round of batch active learning. Denote the labeled pool by $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{n_{\text{labeled}}}$ and the unlabeled pool by $\mathcal{D}_{\text{u}} = \{x'_i\}_{i=1}^{n_{\text{unlabeled}}}$. Let h denote the model and θ^*_{SSL} denote the parameters that minimize the semi-supervised loss—our strategy is oblivious to the choice of the SSL algorithm, it only assumes that the semi-supervised training outperforms supervised training of the model in terms of the generalization error. Lastly, let $\widehat{\mathcal{D}}_{\text{u}} = \{(x, h_{\theta^*_{\text{SSL}}}(x)),\ x \in \mathcal{D}_{\text{u}}\}$ denote the data set of points from $\mathcal{D}_{\text{u}}$ together with their soft pseudo-labels provided by the semi-supervised learner.

The goal of batch active learning is to select and query the labels of the most informative subset of the unlabeled data pool $\mathcal{M} \subseteq \widehat{\mathcal{D}}_{\text{u}}$ of size $m = |\mathcal{M}|$ that would result in a maximal reduction of the model's generalization error. We propose to select $\mathcal{M}$ as follows: $\mathcal{D}_{\text{train}} \cup \mathcal{M}$ should be the coreset of $\mathcal{D}_{\text{train}} \cup \widehat{\mathcal{D}}_{\text{u}}$ for training h in a *supervised* manner. Formally,

$$\begin{aligned} \mathcal{M} := \underset{\mathcal{M} \subseteq \widehat{\mathcal{D}}_{\text{u}},\, |\mathcal{M}| = m}{\arg\min} & \sum_{(x,y) \in \mathcal{D}_{\text{train}}} \ell(h_{\theta^*}(x), y) + \sum_{(x,\widehat{y}) \in \widehat{\mathcal{D}}_{\text{u}}} \ell(h_{\theta^*}(x), \widehat{y}) \\ \text{s.t. } \theta^* = \underset{\theta}{\arg\min} & \sum_{(x,y) \in \mathcal{D}_{\text{train}}} \ell(h_\theta(x), y) + \sum_{(x,\widehat{y}) \in \mathcal{M}} \ell(h_\theta(x), \widehat{y}), \end{aligned} \tag{10}$$

where $\widehat{y}$ denote the pseudo-labels. The motivation for the formulation in Equation (10) is twofold. Firstly, as the coreset of $\widehat{\mathcal{D}}_{\text{u}}$, $\mathcal{M}$ will contain the most essential points of the unlabeled data pool for supervised training. In case some of these points have been wrongly pseudo-labeled, we expect that querying the correct labels induces a large model change. In the other case, acquiring hard labels benefits the semi-supervised learner in label propagation. Secondly, the coreset selection in Equation (10) naturally supports batch selection while avoiding redundancy among the selected points. We provide empirical support for this hypothesis in the experiments.

4.4 Joint Coresets

One application of our framework is speeding up model selection and hyperparameter tuning by performing these on the coreset instead of the full data. For this, we expect the coreset to be transferable to multiple models, whereas our formulation (Equation (1)) is tied to a model and a loss function. A simple idea to construct a coreset with better transferability is to ensure that it is a suitable coreset for multiple models. This is straightforward to achieve within our framework—for brevity, we present the idea for two models: consider models

f and g, and denote their parameters by θ_f and θ_g. The problem of generating the joint coreset can be formulated as

$$\begin{aligned} &\min_{w\in\mathbb{R}^n_+,\,||w||_0\leq m} \sum_{i=1}^{n} \Big(\ell(f_{\theta_f^*}(x_i), y_i) + \lambda \ell(g_{\theta_g^*}(x_i), y_i)\Big) \\ &\text{s.t.} \quad (\theta_f^*, \theta_g^*) \in \arg\min_{(\theta_f,\theta_g)} \sum_{i=1}^{n} w_i \left(\ell(f_{\theta_f}(x_i), y_i) + \lambda \ell(g_{\theta_g}(x_i), y_i)\right). \end{aligned} \tag{11}$$

For solving this bilevel problem, we can rely on the previously presented techniques. In practice, if the loss magnitudes are of the same order, we can set $\lambda = 1$; an additional heuristic for solving the problem with (batch) forward selection is to perform the selection step alternatingly for each model. We verify the validity of this approach in the next section, where we demonstrate the improvement in the transferability of the coreset to deep convolutional networks.

4.5 Dictionary Selection for Compressed Sensing

In signal compression, a collection of signals (data points) needs to be summarized by a small set of measurements ensuring high-fidelity reconstruction. In this section, we are concerned with selecting low-dimensional projections of the data instead of selecting data points directly. Despite this difference, due to the generality of bilevel framework, we can demonstrate our proposed approach for selecting measurements from a set of dictionary elements in order to improve the compression performance. This problem closely resembles dictionary learning and can be seen as a special case of it, without the individual sparsity constraints (Krause and Cevher, 2010). The classical greedy method, which can obey cardinality constraints on the measurement set, and thus control the compression ratio, is computationally very expensive: for each element of the dictionary and at each enlargement, the whole data set needs to be reconstructed. This increases the computational burden by the size of the dictionary, which can be prohibitively large.

Classically, the compression is addressed by transforming the data (signal) to a basis with a known redundancy such as a Fourier transform, and subsequently applying a set of linear measurements. These are then recovered by imposing a regularization strategy such as the smallest squared norm (L_2). Alternatively, Chen (2005) proposed to use absolute norm regularization (L_1) instead, referred to as *basis pursuit* or *compressed sensing*. In fact, compressed sensing can provably recover s-sparse signals with a much smaller set of linear measurements than L_2 regularization, scaling as $\mathcal{O}(s\log(d))$ (Donoho, 2006). The measurement vectors, however, have to satisfy specific conditions such as restricted isometry property (RIP) (Candes et al., 2006) for this to be guaranteed. Certifying that a measurement matrix is RIP is known to be NP-hard (Bandeira et al., 2013). Constructing these matrices randomly is easy albeit the procedure generates them only with a certain probability (Candes et al., 2006).

Given a representative curated data set sufficiently covering all reasonable signals, a natural question is whether one can design a tailored measurement set that improves the compression ratio beyond the randomly generated RIP matrices or other classical measure-

ments. In fact, when the recovery procedure is formulated as an optimization problem, it can be captured in the familiar bilevel form:

$$\begin{aligned} \min_{\|w\|_0 \leq k} \quad & \frac{1}{n} \sum_{i=1}^{n} \|x_i - \hat{x}_i(w)\|_2^2 \\ \text{s.t.} \quad & \hat{x}_i(w) = \arg\min_{y} \sum_{j=1}^{m} w_j \left(a_j^\top (x_i - y)\right)^2 + \lambda R(y), \quad \forall i \in \{1, \ldots, n\}, \end{aligned} \tag{12}$$

where n is the size of representative data set, $a_j \in \mathbb{R}^d$ is one of the m elements of the dictionary we select from, and $R(y)$ is the regularization term corresponding to $R(y) = \|y\|_2^2$ or $R(y) = \|y\|_1$. The values of w_i are restricted to binary in this application. While the optimization problems might not always be differentiable, in practice, however, using an element of the sub-differential proves to be a viable strategy. We demonstrate the versatility of our framework by applying it to solve Equation (12) in Section 5.6.

Bora et al. (2017) and Jalal et al. (2021) have demonstrated that the sparsity-inducing regularizers can be substituted by the constraint that the data belongs to the support of a generative model $G(z)$, where $z \in \mathbb{R}^p$ is the latent space. In this case, the regularization term becomes an indicator function $R(y) = \mathbf{1}_{\exists z \mid y = G(z)}$, which can be reformulated to get a simplified inner problem $\hat{x_i} = G(z_i)$ and $z_i = \arg\min_z \sum_{j=1}^{m} w_j (a_j^\top (x_i - G(z)))^2$. The measurements are assumed to be linear as in classical compressed sensing, and the recovery guarantees satisfy similar conditions on measurement vectors as with sparse signals in previous works (Jalal et al., 2021). Naturally, a more informed measurement selection can further reduce the compression ratio.

5. Experiments

In this section, we demonstrate the flexibility and effectiveness of our framework for a wide range of models and various settings. We start by evaluating the practical variants of Algorithm 1 proposed Section 3.5, and we compare our method to model-specific coreset constructions and other data summarization strategies in Section 5.2. We then study our approach in the memory-constrained settings of continual learning and streaming in Sections 5.3, 5.4, of dictionary selection in Section 5.6, and the human-resource constrained setting of batch active learning in Section 5.5. Sections 5.3, 5.4 are based on Borsos et al. (2020), Section 5.5 is based on Borsos et al. (2021).

5.1 Practical Variants of Algorithm 1

The basic algorithm (Algorithm 1) for bilevel coresets is impractical due to its computational complexity. Hence, we focus on the variants proposed in Section 3.5. Our target model is multiclass logistic regression, where the feature space is the $q = 2048$-dimensional Nyström feature space of the Convolutional Neural Tangent Kernel (CNTK) proposed by Arora et al. (2019) with six layers and global average pooling on CIFAR-10. In this case, $\theta \in \mathbb{R}^{q \times c}$, and $\ell(\theta^\top z(x), y) = -\sum_{j=1}^{c} y_j \log \frac{\exp(\theta_{\cdot,j}^\top z(x))}{\sum_{j'=1}^{c} \exp(\theta_{\cdot,j'}^\top z(x))}$, where y is the one-hot encoded label vector, ℓ is the cross-entropy loss with softmax, and $z(\cdot)$ is the Nyström feature mapping. In each step,

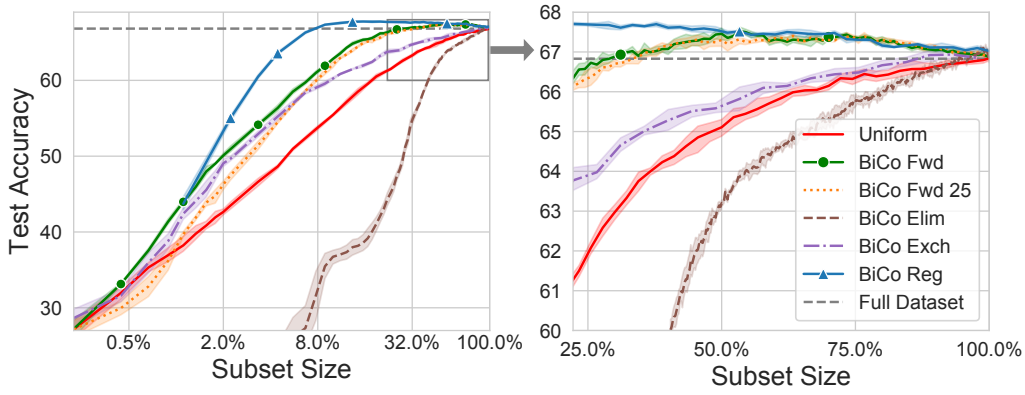


Figure 3: Bilevel coresets for logistic regression on the Nyström feature space of CIFAR-10 CNTK. Building unweighted coresets by forward selection in batches achieves the same performance as one-by-one forward selection after 25% of the points have been selected. Training on a weighted coreset ("BiCo Reg") of size 8% of the full data set produced by our method achieves the same performance as training on the full data set.

we solve the inner optimization problem iteratively up to a tolerance with gradient descent and approximate the implicit gradient with 100 conjugate gradient steps (Pedregosa, 2016). We split CIFAR-10 into a train and validation set, where the validation set is a randomly chosen 10% of the original training set. We instantiate the outer loss as the sum of training and validation losses, whereas the inner optimization problem is defined on the training set. Further details about the experimental setup can be found in Appendix C.

We study coresets with binary weights built using one-by-one forward selection, forward selection in batches of 25, elimination in batches of 200, and exchange with 200 steps (each step exchanges 1% of the selected points; we found that more steps did not increase the performance). For constructing weighted coresets, we solve the regularized version of the bilevel optimization proposed in Section 3.5.3. The results are shown in Figure 3. We can observe that forward selection in batches initially incurs a performance penalty but performs similarly to one-by-one forward selection after 25% of the points have been selected. Both forward selection methods produce coresets of sizes between 33% and 90% on which logistic regression achieves lower test error compared to when trained on the full data set. We observe that elimination increases the test accuracy in initial iterations; however, it significantly underperforms compared to uniform sampling for generating coresets smaller than 90%. Bilevel coresets via regularization (weighted) of size 8% achieve the same performance as training on the full data set. We note that the higher test performance for the weighted coreset with size 20% compared to 90% is due to the higher number of total outer gradient steps performed.

5.2 Comparison to other Summaries

We compare bilevel coresets to coresets designed for specific models, as well as to other data summarization methods. In all experiments, we observe that other methods do not consistently outperform uniform sampling over all subset sizes in contrast to our method.

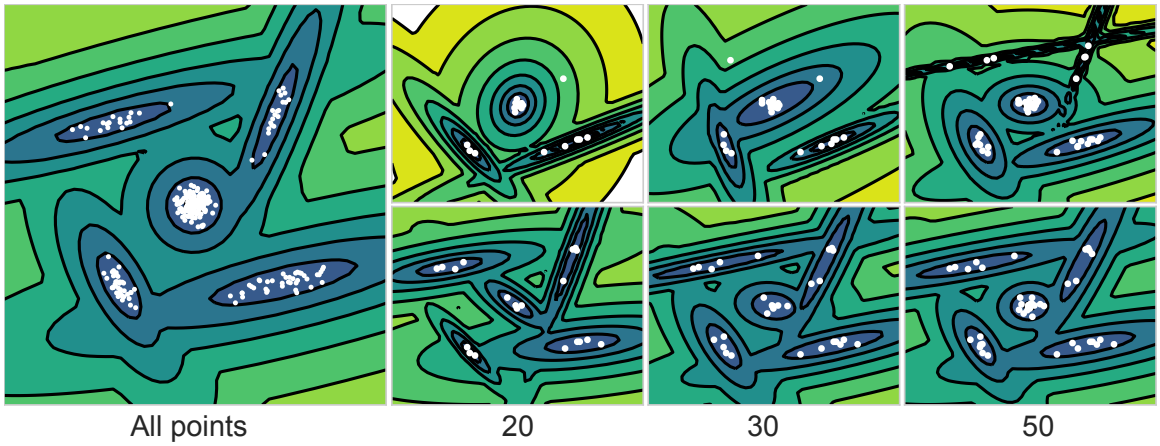


Figure 4: Contours of the log-marginal probability $\log p(x)$ of the Gaussian mixture models (GMM) with $k = 5$ components fitted to different subsets of the data. Left: GMM fitted to the full data set; right: GMM fitted to uniform sample (upper row) and to the bilevel coreset (lower row) of sizes indicated by the subscripts. The bilevel coreset provides good approximate density already with 30 samples.

5.2.1 Gaussian Mixture Models

The first experiment serves as a toy example and proves the versatility of our approach in its broad applicability. In this experiment, we illustrate coreset construction in the *unsupervised* setting of mixture models. We build coresets for Gaussian mixture models with the log marginal probability

$$\log p(x) = \log \left(\sum_{i=1}^{k} \pi_i \mathcal{N}(x|\mu_i, \Sigma_i) \right),$$

where $\{\pi_i\}_{i=1}^k$, $\sum_{i=1}^k \pi_i = 1$ are the mixture weights and $\{\mu_i\}_{i=1}^k$ and $\{\Sigma_i\}_{i=1}^k$ are the component means and covariances. The loss function is thus the negative marginal log-likelihood (NLL) $-\sum_{i=1}^n w_i \log p(x_i)$ minimized over the model parameters $\theta := \{\pi_i, \mu_i, \Sigma_i\}_{i=1}^k$ for the data set $\mathcal{D} = \{x_i\}_{i=1}^n$, where $w \in \mathbb{R}_+^n$ are the data weights. We generate a synthetic two-dimensional data set so that we can visualize and inspect the choices of the coreset selection. We fit a $k = 5$-component Gaussian mixture model to the data by minimizing the loss using the EM algorithm. To generate the coreset, we use the one-by-one forward selection with binary coreset weights, starting from a random sample of 10, and approximate the inverse Hessian-vector product via conjugate gradients.

In Figure 4, we plot the contours of the log-marginal probabilities of the mixtures obtained from fitting the GMM to uniform subsamples and coreset summaries. A coreset of size 30 already provides accurate mean and covariance estimates, with density contours closely resembling the contours of the model fitted to the full data set. We can observe the following progression of the coreset selection: first, points are picked to represent the modes, after which the component covariance and weight estimates are improved.

To quantify the improvement obtained by coresets, we measure the relative errors of the negative log-likelihood (NLL) obtained for subsets of different sizes compared to the negative log-likelihood obtained by fitting on the full data set. Furthermore, we also compare to

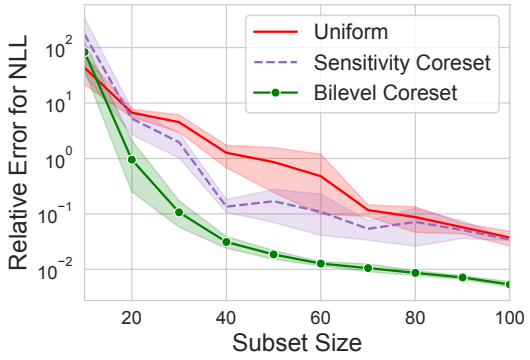


Figure 5: Relative error for the negative log-likelihood of a $k = 5$ component GMM obtained by different methods. Our coreset construction outperforms other methods by an order of magnitude, even those designed specifically for GMMs.

coresets for GMM generated via the sensitivity framework (Lucic et al., 2017). The results in Figure 5 show an improvement of an order of magnitude by our method.

5.2.2 Logistic Regression

The target model in this experiment is logistic regression. For binary classification with logistic regression, several coreset constructions have been proposed that serve as our baselines. We choose four standard binary classification data sets (Dua and Graff, 2017; Uzilov et al., 2006) from the LIBSVM library[1] for this experiment, of size between 9000 and 600000 samples and feature dimensions between 8 and 123. We standardize the features and solve the logistic regression on the subsets selected by different methods to compare their test performance with the one achieved by training on the full data set.

Since the model has low capacity, our framework needs only a small coreset for perfect approximation. Hence, we evaluate the one-by-one forward selection version of our algorithm with weights (Algorithm 1, with 150 outer iterations) and its unweighted (binary weights) version ("BiCo w/ Weights" and "BiCo" in the figures). As for the baselines, we compare to k-means++ (Arthur and Vassilvitskii, 2007) and coresets via sensitivity (Huggins et al., 2016). We also experimented with Hilbert coresets (Campbell and Broderick, 2019). However, we were unable to tune this method to outperform uniform sampling on these data sets. Hence, we do not show its performance. We provide a detailed description of the baselines in Appendix C.

Figure 6 shows that our weighted coreset construction needs less than 2% of the data to obtain the same test accuracy as when the model is trained on the full data set. The unweighted variant needs twice as large a coreset on average to achieve the same performance, however, it is significantly faster to construct—concretely, it takes 12.2 seconds on average per data set to construct a coreset of size 100, which is a factor of 150 faster than weighted coreset generation (the speedup factor equals the number of outer iterations). Further details about the experimental setup can be found in Appendix C.

In the following experiment, we investigate coresets for multiclass logistic regression for MNIST and CIFAR-10. For MNIST, we use 500-dimensional Nyström features to approx-

1. https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html

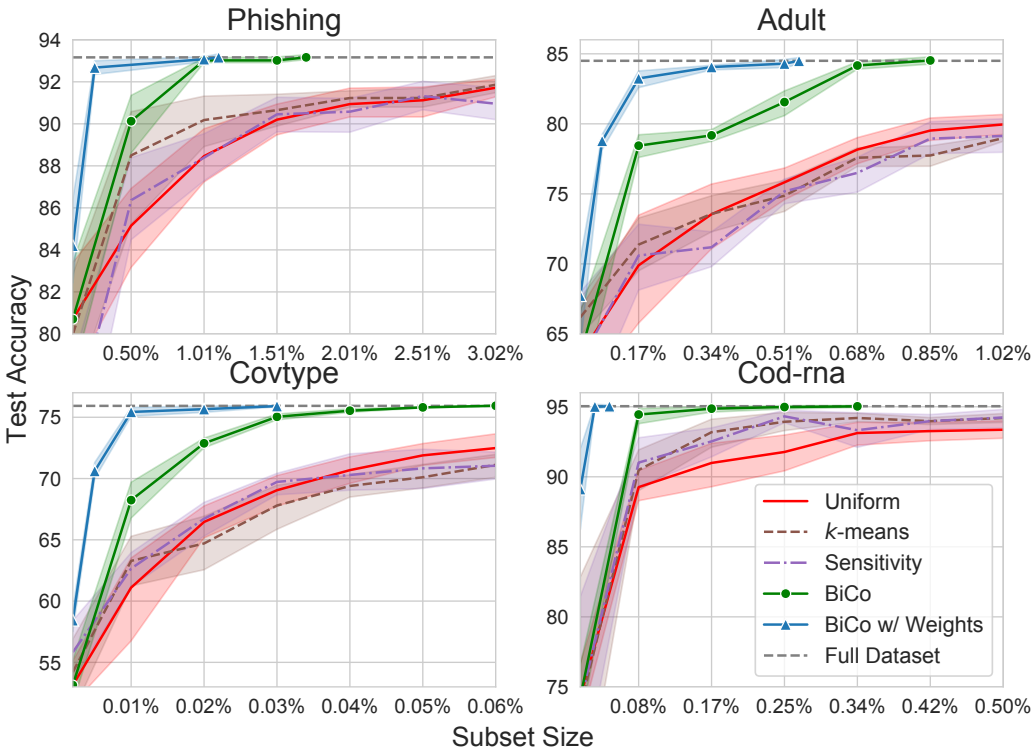


Figure 6: Coresets for binary logistic regression. Our coreset constructions consistently outperform other data summarization approaches, achieving the same performance with 2% of the data as training on the full data set.

imate the feature map of the RBF kernel $k(x, y) = \exp(-\gamma\|x - y\|^2)$ with $\gamma = 10^{-3}$. For CIFAR-10, we use the same setup as in Section 5.1. Figure 7 shows the comparison of one-by-one unweighted forward selection and weighted coreset generation via regularization (the algorithm is stopped when its test performance drops below the performance of the forward selection variant), to uniform sampling and k-means in the feature space (we also compared to k-center selection, which underperforms compared to k-means), and to the two-stage selection of samples that are most frequently "forgotten" during training (misclassified at some point in training after being classified correctly before; referred to as "forgetting" in the figures) (Toneva et al., 2019)—we have also experimented with this method in the binary logistic regression experiment, but it underperformed compared to uniform sampling under subset sizes of 300. Our proposed methods can achieve compression ratios of over 10 (data set size divided by smallest data set size required for obtaining the same test accuracy) on both data sets with weighted coresets generated via regularization.

5.2.3 Neural Networks

For computationally tractable implicit gradient evaluations for networks with millions of parameters, we construct coresets with binary weights using forward selection in batches, and inverse Hessian-vector products approximated using the Neumann series (Section 3.5.1). We truncate the series to $T = 100$ terms and we introduce a scaling hyperparameter α for

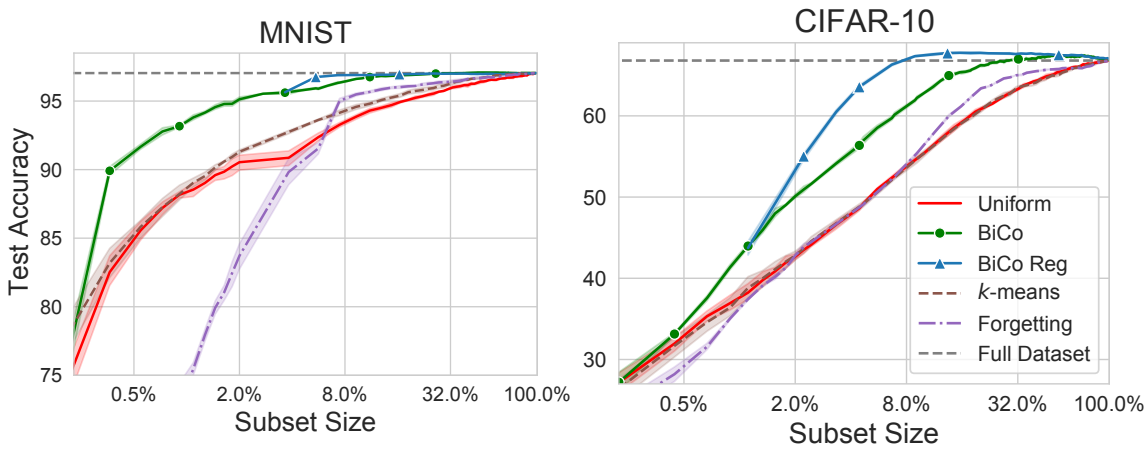


Figure 7: Coresets for multiclass logistic regression. Weighted coresets generated via regularization achieve compression ratios of over 10 on both data sets.

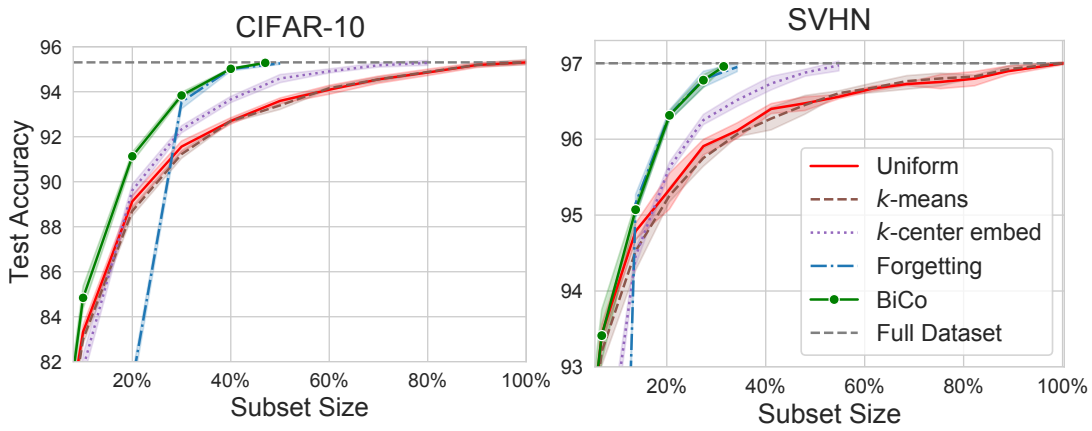


Figure 8: Coreset construction with forward batch selection for WideResNet-16-4; bilevel coresets achieve compression ratios of 2 and 3.

the inner loss f, such that the Neumann series approximation is now applied to $\left(\alpha \frac{\partial^2 f}{\partial \theta \partial \theta^\top}\right)^{-1}$. This is to ensure the convergence of the Neumann series, for which a necessary and sufficient condition is $\max_j \left|\lambda_j \left(I - \alpha \frac{\partial^2 f}{\partial \theta \partial \theta^\top}\right)\right| < 1$, where $\lambda_j(A)$ denote the j-th eigenvalue of A (Chen, 2005). In automatic differentiation frameworks, Hessian-vector products can be calculated efficiently without instantiating the Hessian. However, due to memory considerations, we can only afford to evaluate f on a single minibatch of data in the Hessian-vector products, which introduces another layer of approximation through the stochastic Hessian.

We demonstrate the effectiveness of bilevel coresets for wide residual networks (Zagoruyko and Komodakis, 2016) and search for the smallest coreset size such that the test performance matches the test performance of training on the full data set up to a 0.05% tolerance. For computational considerations, we showcase the unweighted coreset construction via forward

selection in batches of 250 points, starting from a random pool of 2500 points. We evaluate the method by constructing coresets for a WideResNet-16-4 (2.7 million parameters) on CIFAR-10 and SVHN (Netzer et al., 2011)—for SVHN we only use the train split, containing approximately 73000 images. We achieved the best results by retraining the network from scratch after every round of selection with SGD with momentum—further details about the training can be found in Appendix C.

We compare in Figure 8 our unweighted batch forward selection to the following subset selection methods for neural networks: uniform sampling, k-means/k-center in the pixel space (Nguyen et al., 2018), k-means/k-center in the last layer embedding of the trained network (Sener and Savarese, 2018), and selecting samples that are most frequently "forgotten" during training (Toneva et al., 2019). We plot each method's test performance until they first reach the test performance of training on the full data set.

We find that, for both CIFAR-10 and SVHN, k-means outperforms k-center in the feature space, while k-center is better for selection based on the last layer embeddings. The performance of k-means/k-center suggests that simple definitions of redundancy are suboptimal for constructing coresets. Figure 8 shows that our method achieves a compression ratio of 2 on CIFAR-10 and 3 on SVHN, i.e., it can find a representative subset of the training data of size 23500 (47%) for CIFAR-10 and 23000 (31%) for SVHN, such that the WideResNets trained on the chosen subsets achieve the test performance comparable to training on the full data set (within a 0.05 margin of 95.30 for CIFAR-10 and 97.01 for SVHN). Whereas retaining points that are frequently forgotten (Toneva et al., 2019) matches the performance of our method for coreset sizes above 15000 (30%) for CIFAR-10 and 10000 (14%) for SVHN, it underperforms uniform sampling in generating small coresets.

An important application of data summarization is speeding up *hyperparameter tuning*, since the evaluations can be performed on the summary instead of the full data set. In neural architecture search, highly model-specific summaries are undesirable, as they might favor specific architectural choices. To inspect whether the coresets for WideResNet-16-4 are transferable to other architectures, we measure the performance of VGG16 (Simonyan and Zisserman, 2015) and MobileNetV2 (Sandler et al., 2018) adapted to CIFAR-10 and SVHN (kernel strides and pooling kernel sizes reduced to accommodate 32×32 images) on coresets of size 23000; the training procedure is the same as for the WideResNet. Table 1 shows that, whereas the transferred coresets do not reach the full data set performance, they perform significantly better than uniform sampling and the transferred k-center summary and perform similarly to the "forgetting" summary.

We can improve the transferability of the coreset by building joint coresets for multiple models, as proposed in Section 4.4. In the following experiment, we generate a joint coreset for WideResNet-16-4 and VGG16 and evaluate the resulting coreset for transferability on MobileNetV2. For this, we use a simple heuristic for approximating the solution of Equation (11) with $\lambda = 1$: similarly to the previous experiment, we generate the coreset by forward greedy selection in batches of 250 by alternating the model in each step (i.e., we select a new batch of points for the WideResNet, then for VGG16). The results in Table 2 show that this simple heuristic improves the effectiveness of the joint coreset on VGG16 and the transferability to MobileNetV2 at the expense of small performance degradation on WideResNet.

	CIFAR-10		SVHN	
	VGG16	**MobileNetV2**	**VGG16**	**MobileNetV2**
Uniform	91.49 ± 0.16	91.71 ± 0.33	94.24 ± 0.26	94.22 ± 0.39
k-center emb.	92.72 ± 0.20	92.77 ± 0.29	94.59 ± 0.21	94.83 ± 0.34
Forgetting	$\mathbf{93.75 \pm 0.23}$	$\mathbf{93.80 \pm 0.08}$	$\mathbf{95.47 \pm 0.17}$	$\mathbf{95.30 \pm 0.16}$
BiCo	$\mathbf{93.66 \pm 0.15}$	$\mathbf{93.65 \pm 0.22}$	$\mathbf{95.43 \pm 0.15}$	$\mathbf{95.53 \pm 0.20}$
Full data set	94.23 ± 0.14	94.46 ± 0.15	95.93 ± 0.07	96.04 ± 0.09

Table 1: Coresets of size 23000 for WideResNet-16-4 transferred to VGG16 and MobileNetV2. Our method provides similar transfer performance to "forgetting" (Toneva et al., 2019), while both outperform other methods.

Data set	**Architecture**	**BiCo WRN**	**BiCo WRN + VGG**
CIFAR-10	**WideResNet-16-4**	95.24 ± 0.06	95.18 ± 0.07
	VGG16	93.66 ± 0.15	93.88 ± 0.16
	MobileNetV2	93.65 ± 0.22	93.79 ± 0.18
SVHN	**WideResNet-16-4**	96.97 ± 0.02	96.88 ± 0.09
	VGG16	95.43 ± 0.15	95.75 ± 0.14
	MobileNetV2	95.53 ± 0.20	95.76 ± 0.10

Table 2: Coresets of size 23000 for WideResNet-16-4 transferred to VGG16 and MobileNetV2 (Coreset WRN); coresets constructed jointly for WideResNet-16-4 and VGG16 transferred to MobileNetV2 (Coreset WRN + VGG).

5.3 Continual Learning

We compare our approach to existing replay memory management strategies by conducting an extensive experimental study. We focus on continual learning settings where the learning algorithm is a neural network, and we keep the network structure fixed during learning on different tasks. This is known as the "single-head" setup, which is more challenging than instantiating new top layers for different tasks ("multi-head" setup) and does not assume any knowledge of the task descriptor during training and test time. For validating our coreset construction in the continual learning setting, we use the following 10-class classification data sets:

- PMNIST (Goodfellow et al., 2014): consist of 10 tasks, where in each task all images' pixels undergo the same fixed random permutation.
- SMNIST (Zenke et al., 2017): MNIST is split into 5 tasks, where each task consists of distinguishing between consecutive image classes.
- SCIFAR-10: similar to SMNIST on CIFAR-10.

Following Aljundi et al. (2019b), we keep a subsample of 1000 points for each task for all data sets while we retain the full test sets. For PMNIST, we use a fully connected net with

Method	PMNIST	SMNIST	SCIFAR-10
Training w/o replay	73.82 ± 0.49	19.90 ± 0.03	19.95 ± 0.02
Uniform sampling	78.46 ± 0.40	92.80 ± 0.79	$\mathbf{43.22 \pm 0.62}$
k-means of features	78.34 ± 0.49	93.40 ± 0.56	$\mathbf{43.96 \pm 0.78}$
k-means of embeddings	$\mathbf{78.84 \pm 0.82}$	93.96 ± 0.48	$\mathbf{44.37 \pm 0.76}$
k-means of grads	76.71 ± 0.68	87.26 ± 4.08	36.99 ± 1.30
k-center of features	77.32 ± 0.47	93.16 ± 0.96	36.90 ± 1.09
k-center of embeddings	$\mathbf{78.57 \pm 0.58}$	93.84 ± 0.78	40.81 ± 0.53
k-center of grads	77.57 ± 1.12	88.76 ± 1.36	35.11 ± 1.66
Gradient matching	78.00 ± 0.57	92.36 ± 1.17	$\mathbf{43.69 \pm 0.73}$
Max entropy samples	77.13 ± 0.63	91.30 ± 2.77	35.31 ± 1.57
Hardest samples	76.79 ± 0.55	89.62 ± 1.23	32.31 ± 0.88
FRCL's selection	78.01 ± 0.44	91.96 ± 1.75	$\mathbf{43.35 \pm 1.15}$
iCaRL's selection	$\mathbf{79.68 \pm 0.41}$	93.99 ± 0.39	$\mathbf{44.22 \pm 1.31}$
BiCo	$\mathbf{79.33 \pm 0.51}$	$\mathbf{95.81 \pm 0.28}$	$\mathbf{44.51 \pm 1.41}$

Table 3: Continual learning with replay memory size of 100 for versions of MNIST and 200 for CIFAR-10. We report the average test accuracy over the tasks with one standard deviation over 5 runs with different random seeds. Our coreset construction performs consistently among the best.

two hidden layers with 100 units, ReLU activations, and dropout with probability 0.2 on the hidden layers. For SMNIST and SCIFAR-10, we use a CNN consisting of two blocks of convolution, dropout, max-pooling, and ReLU activation, where the number of filters are 32 and 64 and have size 5×5, followed by two fully connected layers of size 128 and 10 with dropout. The dropout probability is 0.5. We fix the replay memory size $m = 100$ for tasks derived from MNIST. For SCIFAR-10, we then set the memory size to $m = 200$. We train our networks for 400 epochs using Adam with step size $5 \cdot 10^{-4}$ after each task.

We perform an extensive comparison under the protocol described above of our method to other data selection methods proposed in the continual learning or the coreset literature—the detailed description of the baselines can be found in Appendix D. For each method, we report the test accuracy averaged over tasks on the best buffer regularization strength β. For a fair comparison to other methods in terms of summary generation time, we restrict our coreset selection method in all of the continual learning experiments to forward selection with binary weights, via the Nyström proxy method with $q = 512$ (Section 3.5.2)—we use the Neural Tangent Kernel (Jacot et al., 2018) corresponding to the chosen architecture, without dropout and max-pooling obtained with the library of Novak et al. (2020).

We report the results in Table 3. We note that while several methods outperform uniform sampling on some data sets, only our method is consistently outperforming it on all data sets. For inspecting the gains obtained by our method over uniform sampling, we plot the final per-task test accuracy on SMNIST in Figure 9. We notice that our method's advantage does not come from excelling at one task but rather by representing the majority of tasks better than uniform sampling. In Appendix D, we present a study of the effect of the replay memory size.

Method	PMNIST	SMNIST
k-center	85.33 ± 0.67	65.71 ± 3.17
Uniform	84.96 ± 0.17	80.06 ± 2.19
BiCo	$\mathbf{86.11 \pm 0.25}$	$\mathbf{84.62 \pm 0.89}$

Table 4: VCL with 20 points/task. VCL can benefit from our coreset construction.

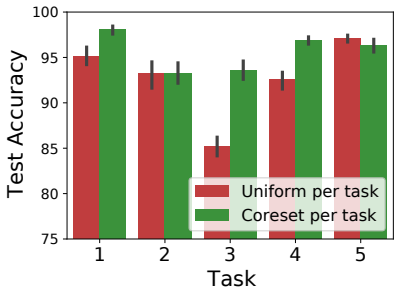


Figure 9: Per-task test accuracy on SMNIST.

Method	PMNIST	SMNIST
Train on coreset	45.03 ± 1.31	89.99 ± 0.76
Reservoir	73.21 ± 0.59	90.72 ± 0.97
BiCo	$\mathbf{74.49 \pm 0.59}$	$\mathbf{92.51 \pm 1.30}$

Table 5: Streaming with replay memory of size 100. BiCo uses the merge-reduce buffer.

Method	SMNIST	SCIFAR-10
Reservoir	80.60 ± 4.36	30.42 ± 0.93
CBRS	89.71 ± 1.31	$\mathbf{37.51 \pm 1.15}$
BiCo	$\mathbf{92.37 \pm 0.27}$	$\mathbf{37.09 \pm 0.65}$

Table 6: Imbalanced streaming on SMNIST and SCIFAR-10.

Our method can also be combined with different approaches to continual learning, such as variational continual learning (VCL) (Nguyen et al., 2018). Whereas VCL also relies on data summaries, it was proposed with uniform and k-center summaries. We replace these with our coreset construction, and, following Nguyen et al. (2018), we conduct an experiment using a single-headed two-layer network with 256 units per layer and ReLU activation, where the coreset size is set to 20 points per task. The results in Table 4 corroborate the advantage of our method over simple selection rules and suggest that VCL can benefit from representative coresets.

5.4 Streaming

Streaming using neural networks has received little attention. To the best of our knowledge, the replay-based approach to streaming has been tackled by Aljundi et al. (2019b), who propose to select points in the replay memory that maximize the angles between pairs of gradients corresponding to the selected points, Hayes et al. (2019), who propose storing cluster centers per class and merging closest clusters for reduction, and Chrysakis and Moens (2020), who propose to class-balance reservoir sampling for imbalanced streams. We compare our method with these methods experimentally.

For evaluating our proposed coreset construction method for training neural networks in the streaming setting, we modify PMNIST and SMNIST by first concatenating all tasks for each data set and then streaming them in batches of size 125. We fix the replay memory size to $m = 100$ and set the number of slots $s = 10$—the replay buffer is managed by the merge-reduce framework (Section 4.2). We train the models for 40 gradient descent steps using Adam with step size $5 \cdot 10^{-4}$ after each batch. We use the same architectures as in the continual learning experiments.

We compare our coreset selection to reservoir sampling (Vitter, 1985) and the sample selection methods of Aljundi et al. (2019b) and Hayes et al. (2019). We were unable to

tune the latter two to outperform reservoir sampling, except the gradient-based selection method of Aljundi et al. (2019b) on PMNIST, achieving a test accuracy of 74.43 ± 1.02. Table 5 shows the dominance of our strategy over reservoir sampling. The table also shows the performance on training only once in the end of the stream on the created coreset, which alone provides strong performance, confirming the merge-reduce framework's validity. We have also experimented with streaming on CIFAR-10 with buffer size $m = 200$, where our coreset construction did not outperform reservoir sampling. However, when the task representation in the stream is imbalanced, our method has significant advantages.

The setup of the streaming experiment favors reservoir sampling, as the data in the stream from different classes is balanced. We illustrate the benefit of our method in the more challenging scenario when the class representation is *imbalanced*. Similarly to Aljundi et al. (2019b), we create imbalanced streams from SMNIST and SCIFAR-10, by retaining 200 random samples from the first four tasks and 2000 from the last task. In this setup, reservoir sampling will underrepresent the first tasks. For SMNIST, we set the replay buffer size to $m = 100$, while for SCIFAR-10, we use $m = 200$. We evaluate the test accuracy on the tasks individually, where we do not undersample the test set. We train the same CNN as in the continual learning experiments on the two imbalanced streams and set the number of slots to $s = 1$. We compare our method to reservoir sampling and class-balancing reservoir sampling (CBRS) (Chrysakis and Moens, 2020). The results in Table 6 confirm the flexibility of our framework and show that it is competitive with CBRS, which is specifically designed for imbalanced streams.

5.5 Batch Active Learning

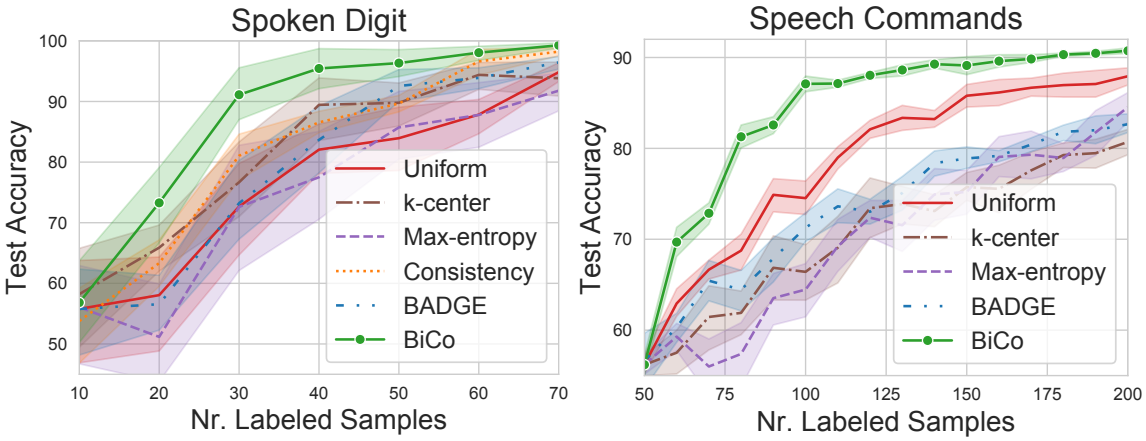
We evaluate our proposed method focusing on the audio domain, where semi-supervised batch active learning has not yet been studied to the best of our knowledge. Our first contribution in this section is showing that semi-supervised strategies proposed in the image domain are also highly effective in audio keyword recognition tasks. Then, we show our batch selection strategy significantly outperforms other acquisition strategies on these tasks. Whereas our strategy is oblivious to the SSL algorithm, we choose MixMatch (Berthelot et al., 2019) as the semi-supervised learning algorithm due to its simplicity, ease of adaptation to the audio domain, and strong empirical performance.

For demonstrating the effectiveness of SSL and its combination with active learning in the audio domain, we focus on the Spoken Digit data set (Jackson, 2016) (2700 utterances, 10 classes) and Speech Commands V2 (Warden, 2018) (85000 utterances, 35 classes) data sets, both containing utterances of the length of one second or shorter. With the goal of applying deep neural network architectures from the image domain with minimal adaptations, we map the utterances to 32×32 mel spectrograms by first resampling them to 16kHz and applying the mel feature extraction with of window length of 128 ms, hop length of 32 ms and 32 bins.

We first investigate the advantages of data augmentation and semi-supervised learning. Our model is a Wide ResNet-28-10 (Zagoruyko and Komodakis, 2016) with weight decay of 10^{-4} and without dropout, whereas the SSL algorithm is MixMatch with two augmentations for label guessing and unlabeled cost weight $\lambda_u = 10$, with other hyperparameters are set to their defaults (Berthelot et al., 2019). As for data augmentation, we apply the following transformations in order with 0.5 probability: i) amplitude change by $a \sim U(0.8, 1.2)$,

Method	Spoken Digit Nr. of Labeled Samples 10	30	60
Supervised w/o augm.	17.33 ± 3.03	35.17 ± 9.74	54.56 ± 5.67
Supervised w augm.	$\mathbf{44.06 \pm 6.98}$	$\mathbf{63.33 \pm 5.25}$	79.17 ± 3.17
MixMatch	$\mathbf{55.78 \pm 11.88}$	$\mathbf{72.67 \pm 10.46}$	$\mathbf{87.83 \pm 3.91}$
	Speech Commands Nr. of Labeled Samples 50	**100**	**200**
Supervised w/o augm.	6.30 ± 0.66	12.03 ± 2.01	34.20 ± 0.91
Supervised w augm.	23.26 ± 2.27	36.94 ± 1.87	54.34 ± 1.46
MixMatch	$\mathbf{56.19 \pm 3.02}$	$\mathbf{74.52 \pm 5.24}$	$\mathbf{87.94 \pm 2.70}$

Table 7: Supervised and semi-supervised learning with uniformly chosen labeled subsets of Spoken Digit (Jackson, 2016) and Speech Commands data set (Warden, 2018).



Method	Nr. of Labeled Samples Spoken Digit 40	70	Speech Commands 100	200
Uniform	82.06 ± 5.31	94.83 ± 2.09	74.52 ± 5.24	87.94 ± 2.70
Max-ent.	77.50 ± 10.23	91.78 ± 4.40	64.46 ± 8.44	84.53 ± 4.34
k-center	$\mathbf{89.44 \pm 7.50}$	93.83 ± 3.92	66.41 ± 8.76	80.71 ± 3.86
Consist.	86.56 ± 2.33	$\mathbf{98.22 \pm 0.96}$	-	-
BADGE	83.67 ± 7.72	96.44 ± 1.46	71.28 ± 4.66	82.66 ± 2.90
BiCo	$\mathbf{95.44 \pm 4.67}$	$\mathbf{99.27 \pm 0.60}$	$\mathbf{87.10 \pm 2.39}$	$\mathbf{90.74 \pm 0.85}$

Figure 10: Batch active learning with batch size $m = 10$ under semi-supervised training with MixMatch (Berthelot et al., 2019). Results averaged over six random seeds, shaded areas represent one standard deviation. Our method provides a significant advantage with a small labeled pool.

ii) audio speed change by $s \sim U(0.8, 1.2)$, iii) random time shifts by t ms, where $t \sim U(-250, 250)$, iv) mixing in background noise with SNR r dB, where $r \sim U(0, 40)$; we use the noise segments from the Speech Commands data set.

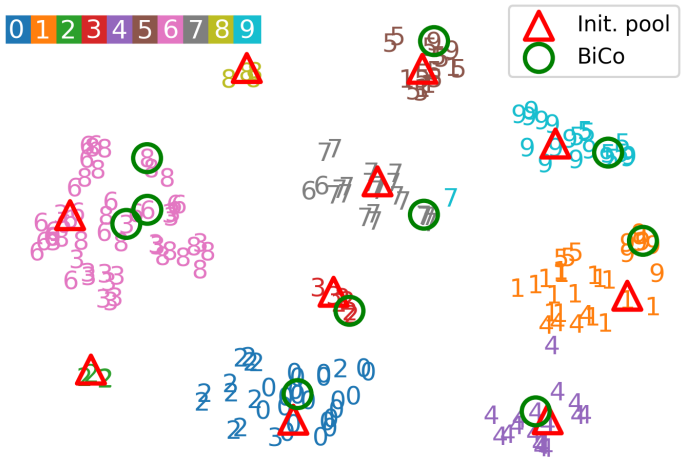


Figure 11: The selected batch of samples for label query by our method (green circles) in the first round of active learning on Spoken Digit data set, when the model is trained on the initial pool (red triangles). The digit values denote the true classes, whereas colors denote predicted classes. The points chosen by our method represent a diverse batch where 6 out of 10 points are misclassified.

We train the models with Adam with an initial learning rate 10^{-3} cosine annealed to 0 over 30 epochs. The results in Table 7 demonstrate the superiority of semi-supervised learning via MixMatch on the chosen keyword recognition tasks. These results are also strong indicators of the necessity of evaluating batch active learning in the semi-supervised setting. For this, we compare our proposed method with batch selection strategies compatible with semi-supervised learning. We note that some batch selection strategies are not applicable to SSL: prominent examples are Bayesian techniques since the common semi-supervised losses do not have Bayesian interpretations. To this end, we implement uniform subsampling, max-entropy selection (predictions averaged over two augmentations), selection based on the k-center algorithm in the last layer of the trained network (Sener and Savarese, 2018), the consistency-based batch selection of Gao et al. (2019) (with five augmentations for calculating the variance), and BADGE (Ash et al., 2020), that selects the batch based on the k-means centers of the last layer gradient embeddings of the hard pseudo-labeled $\mathcal{D}_{\mathrm{u}}$.

For our proposed method, we solve the coreset selection problem in Equation (10) with the CNTK proxy with cross-entropy loss (Section 3.5.2) with 2048-dimensional features and we add 10^{-4} L_2 penalty to the inner objective, turning it into strongly convex multiclass logistic regression problem. Furthermore, we use data augmentations for the inner problem: for each labeled point, we presample 100 augmentations (choosing each randomly from the four types of MixMatch augmentations) and concatenate them for batch gradient descent. We perform one-by-one forward selection, with approximate implicit gradients obtained using 100 steps of conjugate gradients to generate the unweighted coreset.

We compare the methods in the challenging setting of small labeled pools ($n_{\mathrm{labeled}} \leq 200$) and perform the acquisition in batches of size $m = 10$ starting with 10 and 50 labeled samples for Spoken Digit and Speech Commands, respectively. The starting labeled pools are guaranteed to contain at least one sample from each class. In every round of active learning,

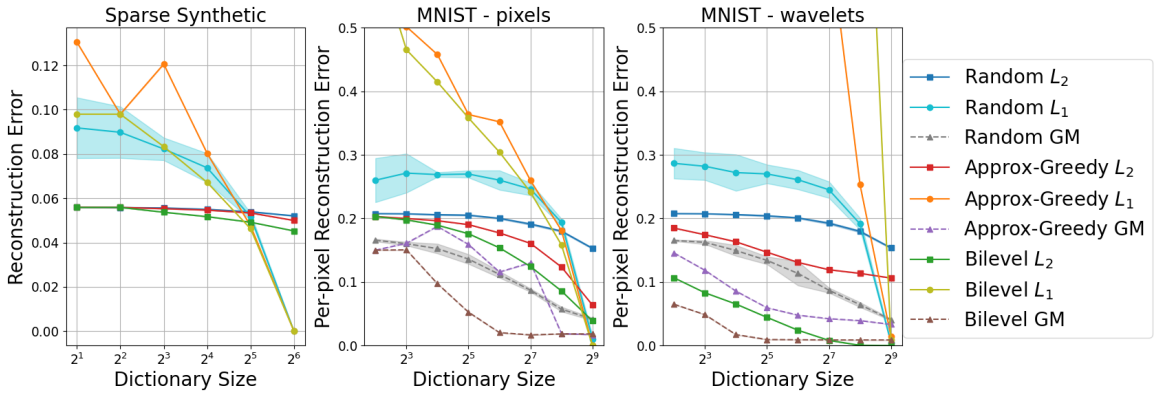


Figure 12: Reconstruction error (mean squared error in Eq. 12) over the dictionary of signals plotted against the subset size of measurements. The recovery methods L_2, L_1, and generative model (GM) are compared with different algorithms to generate the coreset of measurements for recovery: random, approx-greedy, and bilevel. The size of the dictionary is 16384, 786, and 786 from left to right.

we retrain the models from scratch until convergence using MixMatch with the four types of augmentations (amplitude, speedup/slowdown, random shift, and background noise augmentations). We found that retraining from scratch outperformed the warm-started training (with the model from the previous round) for all acquisition strategies. Consequently, running the acquisition strategies (including BiCo) have negligible computational costs compared to the training.

The results in Figure 10 show a significant advantage of our method over other acquisition strategies with only a small number of labeled samples. Especially for Speech Commands, some acquisition strategies suffer from redundancy in the selected batch and, consequently, underperform compared to uniform sampling. We were unable to achieve good performance with the consistency-based acquisition (Gao et al., 2019) on Speech Commands—this phenomenon was also observed by the authors when starting the method with only a few labeled samples, who refer to it as "cold start failure". We also evaluated starting the consistency-based acquisition after a larger number of labeled samples have been acquired by uniform sampling, but the method did not outperform uniform sampling.

We can gain insight into our proposed method by inspecting the chosen batches of points in the active learning round. For this, we plot the acquisitions in the first round of active learning on the Spoken Digit data set in Figure 11, which represents points by their last layer embeddings (after training the network on the initial pool with 60.33% test accuracy) mapped to two dimensions by t-SNE (van der Maaten and Hinton, 2011). The points chosen by our method represent a diverse batch where 6 out of 10 points are misclassified.

5.6 Dictionary Selection for Compressed Sensing

In this section, we showcase our framework for selecting dictionary measurement adaptively and incrementally on two examples in Figure 12: a synthetic data set containing a set of random sparse vectors, and the recovery of MNIST digits using a variational autoencoder (VAE) as the generative model for the images coupled with the reconstruction method of

Bora et al. (2017). The synthetic data set contains 1024 vectors in 128-dimensional Euclidean space where the sparsity level is set to 10%, meaning only approximately 12 values are non-zero per vector. For this data set, the dictionary elements (measurements) are vectors in 128 dimensional space that have normally distributed entries with mean zero and variance 1/128. The MNIST dataset has dimensionality 28^2 since we consider the images as flattened vectors of pixels and we select 250 at random for computational efficiency. Note that in these examples, we create a coreset of the measurements, not of the data points, hence the measurement set is what we choose to be large. The dimensionality of the measurements is either in the size of the data or in the dimensionality in the latent space of the variational autoencoder (VAE). In both data sets, we used $\lambda = 0.01$ as in Eq. (12). When constructing the coreset, we assume that the measurement $a_j^\top x_i$ for each element $i \in \mathcal{D}$ is noiseless since we are constructing the measurements ourselves and have full control over the process.

The baseline algorithms are randomly sampled measurements with normally distributed entries where the variance is proportional to the inverse of the dimension, which satisfy the RIP property with high probability, and approximate-greedy, which is inspired by the heuristics of Krause and Cevher (2010) to speed up the greedy algorithm by picking the measurements with the largest average inner product between the signal and the measurements. The classical greedy algorithm is too expensive given the dictionary sizes used here. The dictionary of linear measurements is chosen as a set of random matrices with entries distributed according to the unit normal distribution, or a wavelets basis (db1 wavelet) for MNIST as done by Bora et al. (2017), which is a more challenging baseline since not necessarily all elements are equally sparse. We use the architecture and loss function to train the VAE as in Kingma and Welling (2014) where we chose the latent vectors to be 20-dimensional. Overall, the compression ratio significantly improves when using our bilevel method.

5.7 Computational Cost

In this section, we measure the runtime of our method. Recall from Section 3.5.1 that the computational complexity of our algorithm with binary weights, forward selection in batches, and Neumann series approximation is $\mathcal{O}\left(mb^{-1}(t_g mg + t_h mg + ng)\right)$.

A single implicit gradient calculation (Equation 3) incurs the cost of calculating $\partial g/\partial\theta$ and $\partial^2 f/\partial\theta\partial w^\top$ and the cost of the Neumann series approximation. For large models, each of these operations has to be performed in minibatches, requiring multiple backpropagation steps. We measure the cost of these operations for WideResNet-16-4 in Table 8, totaling to two minutes per implicit gradient calculation—for reference, we need 84 implicit gradient calculations for generating the coreset of size 23500 for CIFAR-10 in Figure 8.

With the proxy reformulation, the number of parameters is reduced to the order of the dimension of the Nyström features $\mathcal{O}(q)$. On the other hand, the proxy reformulation introduces the overhead of calculating the proxy kernel, which might be a significant overhead for deep neural networks. We measure the time for generating coresets with the CNTK Nyström proxy with $q = 512$ from a data set of 1000 points for the small CNN (SCNN) described in Section 5.3 and for the WideResNet-16-4 from Section 5.2. We calculate the corresponding NTKs without batch normalization and pooling with the library of Novak et al. (2020) on a single GeForce GTX 1080 Ti GPU, whereas the coreset selection is performed on a single CPU. The results are shown in Table 8.

Op	Time
$\partial g / \partial \theta$	29.4 s
Neumann s.	18.9 s
$\partial^2 f / \partial \theta \partial w^\top$	70.8 s

Op	SCNN	WideResNet-16-4
NTK calc.	5.1 s	40.2 s
Coreset 100	29.8 s	31.0 s
Coreset 400	150.6 s	154.4 s

Table 8: Left: a single implicit gradient calculation step for a WideResNet-16-4 with Neumann series approximation with 100 terms on CIFAR-10. Right: runtimes for generating coresets out of 1000 points with CNTKs.

Another important consideration is how to solve the inner optimization problem after an implicit gradient step. In all our applications except for deep neural networks Section 5.2, we resume the inner optimization after the gradient update with the optimal parameters found before the update and perform a small number of inner update steps. For deep neural networks trained with learning rate schedules, we find it beneficial to retrain our models from scratch after each forward batch selection step. A promising future direction is to speed up the selection process without a proxy for neural networks by eliminating the need for retraining from scratch.

6. Discussion and Conclusion

We presented a generic coreset construction framework applicable to any twice differentiable model without requiring model-specific adaptations. We proposed several variants for scaling the basic algorithm to large models and data sets. We showed that our method is effective for various models in various settings, outperforming specialized coreset constructions and other data summarization methods.

6.1 Limitations

We provide guarantees only for the case where the overall optimization objective $G(w)$ is convex, showing the nonvacuousness of the bound on the coreset size is an open problem. Due to the hardness of the cardinality-constrained bilevel optimization problem, our method is only a heuristic for the non-convex settings. In this paper, we provide empirical evidence for the effectiveness of the proposed framework.

Except for the binary logistic regression experiments or kernelized linear regression, the cost of our proposed coreset construction is higher than the cost of training the model on the full data set. This contrasts with some of the previous coreset constructions' goals to speed up the training process. Our method is thus suited for settings with memory or human resource constraints, as well as when the summary is reused (e.g., in hyperparameter tuning)—settings for which we demonstrated the effectiveness of our approach empirically in Section 5.

6.2 Future Work

The flexibility of our framework in accommodating different upper and lower-level objectives allows for various extensions and applications. While we discussed some in this work, there are several promising directions, e.g., the framework could be extended to Bayesian inference

by using objectives from variational inference. Furthermore, the idea of formulating subset selection as a cardinality-constrained bilevel optimization problem is very general and can be applied to problems besides coreset construction. Some notable examples include basis selection for the Nyström approximation, feature selection, and neural network pruning.

Acknowledgments

This research was supported by the SNSF grant 407540_167212 through the NRP 75 Big Data program, by the European Research Council (ERC) under the European Union's Horizon 2020 research, innovation programme grant agreement No 815943, and by the Swiss National Science Foundation through the NCCR Catalysis.

Appendix A. Connection to Experimental Design

In this section, the weights are assumed to be binary, i.e., $w \in \{0,1\}^n$. We will use a shorthand X_S for the matrix where only rows of X whose indices are in $S \subset [n]$ are selected. This will be equivalent to selection done via the diagonal matrix $D(w)$, where $i \in S$ corresponds to $w_i = 1$ and zero otherwise. Additionally, let $\hat{\theta}$ be a minimizer of the following loss,

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{n} w_i (x_i^\top \theta - y_i)^2 + \lambda \sigma^2 ||\theta||_2^2 \tag{13}$$

which has the following closed form,

$$\hat{\theta}_S = (X_S^\top X_S + \lambda \sigma^2 I)^{-1} X_S^\top y_S. \tag{14}$$

Frequentist Experimental Design. Under the assumption that the data follows the linear model $y = X\theta + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$, we can show that the bilevel coreset framework instantiated with the inner objective (13) and $\lambda = 0$, with various choices of outer objectives is related to frequentist optimal experimental design problems. The following propositions show how different outer objectives give rise to different experimental design objectives.

Proposition 5 (A-experimental design) *Under the linear regression assumptions and when* $g(\hat{\theta}) = \frac{1}{2}\mathbb{E}_\epsilon \left[\left\| \theta - \hat{\theta} \right\|_2^2 \right]$, *with the inner objective is equal to* (13) *with* $\lambda = 0$, *the objective simplifies,*

$$G(w) = \frac{\sigma^2}{2} \mathrm{Tr}((X^\top D(w) X)^{-1}).$$

Proof Using the closed form in (14), and model assumptions, we see that $\hat{\theta}_S = \theta + (X_S^\top X_S)^{-1} X_S^\top \epsilon_S$. Plugging this into the outer objective,

$$\begin{aligned}
g(\hat{\theta}) &= \frac{1}{2}\mathbb{E}_\epsilon \left[\left\| \theta - \hat{\theta}_S \right\|_2^2 \right] \\
&= \frac{1}{2}\mathbb{E}_\epsilon \left[\left\| (X_S^\top X_S)^{-1} X_S^\top \epsilon_S \right\|_2^2 \right] \\
&= \frac{1}{2}\mathbb{E}_\epsilon \left[\mathrm{Tr}\left(\epsilon_S^\top X_S (X_S^\top X_S)^{-2} X_S^\top \epsilon_S \right) \right] \\
&= \frac{\sigma^2}{2} \mathrm{Tr}\left((X_S^\top X_S)^{-1} \right) \\
&= \frac{\sigma^2}{2} \mathrm{Tr}\left((X^\top D(w) X)^{-1} \right)
\end{aligned}$$

where in the third line, we used the cyclic property of trace and, subsequently, the normality of ϵ. ■

Proposition 6 (V-experimental design) *Under the linear regression assumptions and when* $g(\hat{\theta}) = \frac{1}{2n}\mathbb{E}_\epsilon \left[\left\| X\theta - X\hat{\theta} \right\|_2^2 \right]$ *and the inner objective is equal to* (13) *with* $\lambda = 0$, *the objective simplifies,*

$$G(w) = \frac{\sigma^2}{2n} \mathrm{Tr}(X (X^\top D(w) X)^{-1} X^\top).$$

Proof Using the closed form in (14), and model assumptions, we see that $\hat{\theta}_S = \theta + (X_S^\top X_S)^{-1} X_S^\top \epsilon_S$. Plugging this in to the outer objective $g(\hat{\theta})$,

$$\begin{aligned} G(w) &= \frac{1}{2n}\mathbb{E}_\epsilon\left[\left\|X\theta - X\hat{\theta}_S\right\|_2^2\right] \\ &= \frac{1}{2n}\mathbb{E}_\epsilon\left[\left\|X(X_S^\top X_S)^{-1}X_S^\top \epsilon_S\right\|_2^2\right] \\ &= \frac{1}{2n}\mathbb{E}_\epsilon\left[\mathrm{Tr}\left(\epsilon_S^\top X_S (X_S^\top X_S)^{-1} X^\top X (X_S^\top X_S)^{-1} X_S^\top \epsilon_S\right)\right] \\ &= \frac{\sigma^2}{2n}\mathrm{Tr}\left(X(X_S^\top X_S)^{-1}X^\top\right) \\ &= \frac{\sigma^2}{2n}\mathrm{Tr}\left(X(X^\top D(w) X)^{-1}X^\top\right) \end{aligned}$$

where in the third line, we used the cyclic property of trace and, subsequently, the normality of ϵ. ■

Infinite data limit. The following proposition links the data summarization objective and V-experimental design in the infinite data limit $n \to \infty$.

Proposition 7 (Infinite data limit) *Under the linear regression assumptions $y = X\theta + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$, let g_V be*

$$g_V(\hat{\theta}) = \frac{1}{2n}\mathbb{E}_\epsilon\left[\left\|X\theta - X\hat{\theta}\right\|_2^2\right]$$

the V-experimental design outer objective, and let the summarization objective be,

$$g(\hat{\theta}) = \frac{1}{2n}\mathbb{E}_\epsilon\left[\sum_{i=1}^n (x_i^\top \hat{\theta} - y_i)^2\right].$$

For all $\hat{\theta}_S$ in Eq. (14), we have

$$\lim_{n\to\infty} g(\hat{\theta}_S) - g_V(\hat{\theta}_S) = \frac{\sigma^2}{2}.$$

Proof Since $y_i = x_i^\top \theta + \epsilon_i$, we have,

$$\begin{aligned} g(\hat{\theta}_S) &= \frac{1}{2n}\mathbb{E}_\epsilon\left[\sum_{i=1}^n (x_i^\top \hat{\theta}_S - x_i^\top\theta - \epsilon_i)^2\right] \\ &= \frac{1}{2n}\mathbb{E}_\epsilon\left[\left\|X\theta - X\hat{\theta}_S\right\|_2^2\right] - \frac{1}{n}\mathbb{E}_\epsilon\left[\epsilon^\top(X\hat{\theta}_S - X\theta)\right] + \frac{1}{2n}\mathbb{E}_\epsilon\left[\|\epsilon\|_2^2\right] \\ &= g_V(\hat{\theta}_S) - \frac{1}{n}\mathbb{E}_\epsilon\left[\epsilon^\top(X\hat{\theta}_S - X\theta)\right] + \frac{\sigma^2}{2} \\ &= g_V(\hat{\theta}_S) - \frac{1}{n}\mathbb{E}_\epsilon\left[\sum_{i\in S}\epsilon_i(x_i^\top\hat{\theta}_S - x_i^\top\theta)\right] - \frac{1}{n}\mathbb{E}_\epsilon\left[\sum_{i\in[n]\setminus S}\epsilon_i(x_i^\top\hat{\theta}_S - x_i^\top\theta)\right] + \frac{\sigma^2}{2} \end{aligned}$$

We have $\lim_{n\to\infty} \frac{1}{n}\mathbb{E}_\epsilon\left[\sum_{i\in S}\epsilon_i(x_i^\top\hat{\theta}_S - x_i^\top\theta)\right] = 0$ since S is a finite set. Since $\hat{\theta}_S$ is independent of ϵ_i, $i \in [n]\setminus S$,

$$\mathbb{E}_\epsilon\left[\sum_{i\in[n]\setminus S}\epsilon_i(x_i^\top\hat{\theta}_S - x_i^\top\theta)\right] = \sum_{i\in[n]\setminus S}\mathbb{E}_\epsilon\left[\epsilon_i\right]\mathbb{E}_\epsilon\left[x_i^\top\hat{\theta}_S - x_i^\top\theta\right] = 0.$$

As a consequence, as $\lim_{n\to\infty} g(\hat{\theta}_S) - g_V(\hat{\theta}_S) = \frac{\sigma^2}{2}$.

■

Proposition 7 does not imply that our algorithm performs the same steps with g_V instead of g. It only means that the optimal solutions to the problems converge to selections with the same quality in the infinite data limit.

Bayesian V-Experimental Design. Bayesian experimental design (Chaloner and Verdinelli, 1995) can be incorporated as well into our framework. In Bayesian modeling, the "true" parameter θ is not a fixed value, but instead a sample from a prior distribution $p(\theta)$ and hence a random variable. Consequently, upon taking into account the random nature of the coefficient vector, we can find appropriate inner and outer objectives.

Proposition 8 *Under Bayesian linear regression assumptions and where $\theta \sim \mathcal{N}(0, \lambda^{-1}I)$, let the outer objective*

$$g_V(\hat{\theta}) = \frac{1}{2n}\mathbb{E}_{\epsilon,\theta}\left[\left\|X\theta - X\hat{\theta}\right\|_2^2\right],$$

where expectation is over the prior as well. Furthermore, let the inner objective be Eq. (13) *with the same value of λ, then the overall objective simplifies to*

$$G(w) = \frac{1}{2n}\mathrm{Tr}\left(X\left(\frac{1}{\sigma^2}X^\top D(w)X + \lambda I\right)^{-1}X^\top\right). \tag{15}$$

Proof Using the closed form in (14), and model assumptions, we see that $\hat{\theta}_S = (X_S^\top X_S + \lambda\sigma^2 I)^{-1}X_S^\top(X_S\theta + \epsilon_S)$. Plugging this in to the outer objective $g_V(\hat{\theta})$,

$$\begin{aligned}
G(w) &= \frac{1}{2n}\mathbb{E}_{\epsilon,\theta}\left[\left\|X\theta - X\hat{\theta}_S\right\|_2^2\right] \\
&= \frac{1}{2n}\mathbb{E}_{\epsilon,\theta}\left[\left\|X((X_S^\top X_S + \lambda\sigma^2 I)^{-1}X_S^\top(X_S\theta + \epsilon_S) - \theta)\right\|_2^2\right] \\
&= \frac{1}{2n}\mathbb{E}_{\epsilon,\theta}\left[\left\|X(X_S^\top X_S + \lambda\sigma^2 I)^{-1}X_S^\top\epsilon_S - \sigma^2\lambda X(X_S^\top X_S + \lambda\sigma^2 I)^{-1}\theta\right\|_2^2\right] \\
&= \frac{1}{2n}\mathbb{E}_{\theta}\left[\left\|\lambda\sigma^2 X(X_S^\top X_S + \lambda\sigma^2 I)^{-1}\theta\right\|_2^2\right] \\
&\quad + \frac{1}{2n}\mathbb{E}_{\epsilon}\left[\left\|X(X_S^\top X_S + \lambda\sigma^2 I)^{-1}X_S^\top\epsilon_S\right\|_2^2\right] \\
&= \frac{\sigma^2}{2n}\mathrm{Tr}\left(\lambda\sigma^2(X_S^\top X_S + \lambda\sigma^2 I)^{-1}X^\top X(X_S^\top X_S + \lambda\sigma^2 I)^{-1}\right)
\end{aligned}$$

$$
\begin{aligned}
&+\frac{\sigma^2}{2n}\mathrm{Tr}(X_S(X_S^\top X_S+\lambda\sigma^2 I)^{-1}X^\top X(X_S^\top X_S+\lambda\sigma^2 I)^{-1}X_S^\top)\\
=\;&\frac{\sigma^2}{2n}\mathrm{Tr}\left((X_S^\top X_S+\lambda\sigma^2 I)^{-1}X^\top X(X_S^\top X_S+\lambda\sigma^2 I)^{-1}\left(\lambda\sigma^2 I+X_S^\top X_S\right)\right)\\
=\;&\frac{\sigma^2}{2n}\mathrm{Tr}\left(X(X_S^\top X_S+\lambda\sigma^2 I)^{-1}X^\top\right)\\
=\;&\frac{\sigma^2}{2n}\mathrm{Tr}\left(X(X^\top D(w)X+\lambda\sigma^2 I)^{-1}X^\top\right)
\end{aligned}
$$

where we used that $\mathbb{E}_\epsilon[\epsilon]=0$, and the cyclic property of the trace, and the final results follow by rearranging.

■

Similarly to the unregularized frequentist experimental design, in the infinite data limit, even the Bayesian objectives share the same optima. The difference here is that the true parameter is no longer a fixed value, and we need to integrate it using the prior.

Proposition 9 (identical to Proposition 2) *Under the Bayesian linear regression assumptions $y=X\theta+\epsilon$, $\epsilon\sim\mathcal{N}(0,\sigma^2 I)$ and $\theta\sim\mathcal{N}(0,\lambda^{-1})$, let g_V be*

$$g_V(\hat{\theta})=\frac{1}{2n}\mathbb{E}_{\epsilon,\theta}\left[\left\|X\theta-X\hat{\theta}\right\|_2^2\right]$$

the Bayesian V-experimental design outer objective, and let the summarization objective be,

$$g(\hat{\theta})=\frac{1}{2n}\mathbb{E}_{\epsilon,\theta}\left[\sum_{i=1}^{n}(x_i^\top\hat{\theta}-y_i)^2\right].$$

For all $\hat{\theta}_S$ in Eq. (14), we have

$$\lim_{n\to\infty} g(\hat{\theta}_S)-g_V(\hat{\theta}_S)=\frac{\sigma^2}{2}.$$

Proof The proof follows similarly as in Proposition 7. ■

Lemma 10 *Assume $\|x_i\|_2<L<\infty$ for all $i\in[n]$ and $w\in\mathbb{R}_+^n$ s.t. $\|w\|_2<\infty$. The function*

$$G(w)=\frac{1}{2n}\mathrm{Tr}\left(X\left(\frac{1}{\sigma^2}X^\top D(w)X+\lambda I\right)^{-1}X^\top\right)$$

is convex and smooth in w.

Proof We will show that the Hessian of $G(w)$ is positive semi-definite (PSD) and that the maximum eigenvalue of the Hessian is bounded, which implies the convexity and smoothness of $G(w)$.

For brevity, we work with $\hat{G}(w) = \mathrm{Tr}\left(X\left(X^\top D(w)X + \lambda\sigma^2 I\right)^{-1}X^\top\right)$ where $\frac{\sigma^2}{2n}\hat{G}(w) = G(w)$. In addition, denote $F(w) = X^\top D(w)X + \lambda\sigma^2 I$ and $F^+(w) = \left(X^\top D(w)X + \lambda\sigma^2 I\right)^{-1}$ s.t. $F(w)F^+(w) = I$. First, we would like to calculate $\frac{\partial\hat{G}(w)}{\partial w_i}$, for which we will use directional derivatives:

$$\begin{aligned}
D_v\hat{G}(w) &= \lim_{h\to 0}\frac{\hat{G}(w+hv)-\hat{G}(w)}{h}\\
&= \mathrm{Tr}\left(X\left(\lim_{h\to 0}\frac{F^+(w+hv)-F^+(w)}{h}\right)X^\top\right)\\
&= \mathrm{Tr}\left(X\left(\lim_{h\to 0}F^+(w+hv)\cdot\frac{F(w)-F(w+hv)}{h}\cdot F^+(w)\right)X^\top\right)\\
&\overset{\text{def. of } F}{=} -\mathrm{Tr}\left(X\left(\lim_{h\to 0}F^+(w+hv)\cdot\frac{\cancel{h}X^\top D(v)X}{\cancel{h}}\cdot F^+(w)\right)X^\top\right)\\
&= -\mathrm{Tr}\left(XF^+(w)X^\top D(v)XF^+(w)X^\top\right)
\end{aligned}$$

To get $\frac{\partial\hat{G}(w)}{\partial w_i}$, we should choose as direction $v_i := (0,\dots,0,1,0,\dots,0)^\top$ where 1 is on the i-th position. Since $X^\top D(v_i)X = x_i x_i^\top$, we have that:

$$\begin{aligned}
\frac{\partial\hat{G}(w)}{\partial w_i} = D_{v_i}\hat{G}(w) &= -\mathrm{Tr}\left(XF^+(w)x_i x_i^\top F^+(w)X^\top\right)\\
&\overset{\text{cyclic prop Tr}}{=} -x_i^\top F^+(w)X^\top XF^+(w)x_i
\end{aligned}$$

We will proceed similarly to get $\frac{\partial^2\hat{G}(w)}{\partial w_j\partial w_i}$.

$$\begin{aligned}
D_v\frac{\partial\hat{G}(w)}{\partial w_i} &= -x_i^\top\lim_{h\to 0}\frac{F^+(w+hv)X^\top XF^+(w+hv)-F^+(w)X^\top XF^+(w)}{h}x_i\\
&= x_i^\top\lim_{h\to 0}F^+(w+hv)\cdot\frac{F(w+hv)F^+(w)X^\top X}{h}\cdot F^+(w)x_i\\
&\quad -x_i^\top\lim_{h\to 0}F^+(w+hv)\cdot\frac{X^\top XF^+(w+hv)F(w)}{h}\cdot F^+(w)x_i
\end{aligned}$$

Now, since,

$$\begin{aligned}
F(w+hv)F^+(w) &= (F(w)+hX^\top D(v)X)F^+(w)\\
&= I + hX^\top D(v)XF^+(w)\\
F^+(w+hv)F(w) &= F^+(w+hv)(F(w+hv)-hX^\top D(v)X)\\
&= I - hF^+(w+hv)X^\top D(v)X
\end{aligned}$$

we have

$$\begin{aligned}
D_v\frac{\partial\hat{G}(w)}{\partial w_i} &= x_i^\top F^+(w)\left(X^\top D(v)XF^+(w)X^\top X + X^\top XF^+(w)X^\top D(v)X\right)F^+(w)x_i\\
&= 2x_i^\top F^+(w)X^\top D(v)XF^+(w)X^\top XF^+(w)x_i
\end{aligned}$$

Choosing v_j as our directional derivative, we have:

$$\begin{aligned}\frac{\partial^2 \hat{G}(w)}{\partial w_j \partial w_i} = D_{v_j} \frac{\partial \hat{G}(w)}{\partial w_i} &= 2\left(x_i^\top F^+(w) x_j\right)\left(x_j^\top F^+(w) X^\top X F^+(w) x_i\right) \\ &= 2\left(x_j^\top F^+(w) x_i\right)\left(x_j^\top F^+(w) X^\top X F^+(w) x_i\right)\end{aligned}$$

from which we can see that we can write the Hessian of $\hat{G}(w)$ in matrix form as:

$$\nabla_w^2 \hat{G}(w) = 2\left(XF^+(w)X^\top\right) \circ \left(XF^+(w)X^\top XF^+(w)X^\top\right)$$

where $\circ$ denotes the Hadamard product. Since $F^+(w)$ is PSD it immediately follows that $XF^+(w)X^\top$ and $XF^+(w)X^\top XF^+(w)X^\top$ are PSD. Since the Hadamard product of two PSD matrices is PSD due to the *Schur product theorem*, it follows that the Hessian $\nabla_w^2 \hat{G}(w)$ is PSD and thus $G(w)$ is convex.

As for smoothness, we need the largest eigenvalue of the Hessian to be bounded:

$$\begin{aligned}\lambda_{\max}(\nabla_w^2 \hat{G}(w)) &\leq \operatorname{Tr}(\nabla_w^2 \hat{G}(w)) \\ &= 2\sum_{i=1}^n \left(XF^+(w)X^\top\right)_{ii}\left(XF^+(w)X^\top XF^+(w)X^\top\right)_{ii} \\ &= 2\sum_{i=1}^n \left(x_i^\top F^+(w) x_i\right)\left(x_i^\top F^+(w) X^\top X F^+(w) x_i\right) \\ &= 2\sum_{i=1}^n \left(x_i^\top F^+(w) x_i\right)\left\|XF^+(w)x_i\right\|_2^2 \\ &\leq 2\sum_{i=1}^n \lambda_{\max}(F^+(w))\left\|x_i\right\|_2^2\left\|XF^+(w)x_i\right\|_2^2 \\ &\leq 2\sum_{i=1}^n \lambda_{\max}(F^+(w))\left\|x_i\right\|_2^2\left\|X\right\|_2^2\left\|F^+(w)\right\|_2^2\left\|x_i\right\|_2^2 \\ &= 2\lambda_{\max}^3(F^+(w))\left\|X\right\|_2^2\sum_{i=1}^n\left\|x_i\right\|_2^4 \\ &\leq \frac{2}{\lambda^3\sigma^6}\left\|X\right\|_2^2\sum_{i=1}^n\left\|x_i\right\|_2^4 \\ &\leq \frac{2}{\lambda^3\sigma^6}\left\|X\right\|_F^2 nL^4 \\ &\leq \frac{2n^2L^6}{\lambda^3\sigma^6},\end{aligned}$$

where in the fifth line, we have used the property of the Rayleigh quotient that for any nonzero vector x and self-adjoint matrix M, we have that $x^\top Mx \leq \lambda_{\max}(M)\left\|x\right\|_2^2$. Thus G is $\frac{nL^6}{\lambda^3\sigma^4}$-smooth.

■

Appendix B. Connection to Influence Functions

Proof of Proposition 1. Following Koh and Liang (2017) and using the result of Cook and Weisberg (1982), under twice differentiability and strict convexity of the inner loss, the empirical influence function at k is

$$\left.\frac{\partial \theta^*}{\partial \varepsilon^\top}\right|_{\varepsilon=0} = -\left(\frac{\partial^2 \sum_{i=1}^n w^*_{S,i} \ell_i(\theta^*)}{\partial \theta \partial \theta^\top}\right)^{-1} \nabla_\theta \ell_k(\theta^*). \tag{16}$$

Using the chain rule for $\mathcal{I}(k)$:

$$\begin{aligned}
\mathcal{I}(k) &= -\left.\frac{\partial \sum_{i=1}^n \ell_i(\theta^*)}{\partial \varepsilon}\right|_{\varepsilon=0} \\
&= -\left(\nabla_\theta \sum_{i=1}^n \ell_i(\theta^*)\right)^\top \left.\frac{\partial \theta^*}{\partial \varepsilon^\top}\right|_{\varepsilon=0} \\
&\overset{\text{Eq. (16)}}{=} \nabla_\theta \ell_k(\theta^*)^\top \left(\frac{\partial^2 \sum_{i=1}^n w^*_{S,i} \ell_i(\theta^*)}{\partial \theta \partial \theta^\top}\right)^{-1} \nabla_\theta \sum_{i=1}^n \ell_i(\theta^*).
\end{aligned}$$

Hence, $\operatorname{argmax}_k \mathcal{I}(k)$ and the selection rule in Equation (5) are the same. ■

Appendix C. Detailed Experimental Setup for Sections 3.5 and 5.2

Variants All variants in Section 3.5 use $\lambda = 10^{-7}$ regularizer in the inner problem. The inner optimization is performed with Adam using a step size of 0.01 as follows: all variants start with an optimization phase on the initial point set with $5 \cdot 10^4$ iterations; then, after each step, an additional 10^4 GD iterations are performed. We note that performing 10^4 GD iterations on the entire data set takes 2.3 seconds on a single GeForce GTX 1080 Ti.

Binary Logistic Regression The features of the data sets are standardized to zero mean and unit variance. The logistic regression is solved using batch Adam with step size 0.01 and L_2-penalty of 0.01. For the bilevel coresets, the selection process is started from 10 randomly chosen points and the implicit gradients are calculated through 100 steps of conjugate gradients. For the unweighted version, 50 gradient descent steps are performed after each selection. For the weighted version, we use Adam with step size 0.01 to optimize the weights over 150 outer iterations in each step.

We consider the following baselines:

- k-means in the feature space, where the chosen subset is the set of centers selected by k-means++ (Arthur and Vassilvitskii, 2007); we also evaluated k-center, which performed worse than k-means on all data sets,
- coresets for binary logistic regression via sensitivity (Huggins et al., 2016), where, for each data set, we choose the best hyperparameter setting from a grid search over $k \in \{5, 10, 25\}$ and $R \in \{0.1, 1, 10, 100\}$ — we refer to Huggins et al. (2016) for the details about the hyperparameters k and R.

Algorithm 3 Streaming BiCo with Merge-reduce Buffer

Input: stream S, number of slots s, β

procedure SELECT_INDEX($[(C_1, \beta_1), \ldots, (C_{s+1}, \beta_{s+1})]$)
  if $s == 1$ or $\beta_{s-1} > \beta_s$ **then**
    return s
  else
    $k = \arg\min_{i \in [1,\ldots,s]} (\beta_i == \beta_{i+1})$
    return k
  end if
end procedure

buffer = [ ]

for $\mathcal{D}_t$ in stream S **do**
  $\mathcal{C}_t$ = construct_coreset($\mathcal{D}_t$)
  buffer.append($(\mathcal{C}_t, \beta)$)
  if buffer.size $> s$ **then**
    k = select_index(buffer)
    $\mathcal{C}'$ = construct_coreset($(\mathcal{C}_k, \beta_k), (\mathcal{C}_{k+1}, \beta_{k+1})$)
    $\beta' = \beta_k + \beta_{k+1}$
    delete buffer$[k+1]$
    buffer$[k] = (\mathcal{C}', \beta')$
  end if
end for

- Hilbert coresets (Campbell and Broderick, 2019) solved via Frank-Wolfe (Campbell and Broderick, 2019) and GIGA (Campbell and Broderick, 2018) with the norm chosen as the weighted Fisher information distance and with random features of 500 dimensions. However, we were unable to tune either of these methods to outperform uniform sampling on any of the data sets, hence we do not show their performance.

Neural Networks For training the networks, we use weight decay of $5 \cdot 10^{-4}$ and an initial learning rate of 0.1 cosine-annealed to 0 over $300 \cdot n/m$ epochs, where n is the full data set size and m is the subset size. Additionally, we use dropout with a rate of 0.4 for SVHN. For CIFAR-10, we use the standard data augmentation pipeline of random cropping and horizontal flipping, whereas we do not use data augmentation for SVHN.

Appendix D. Continual Learning and Streaming

For the continual learning experiments, we compare the following methods:

- Training w/o replay: train after each task without replay memory. Demonstrates how catastrophic forgetting occurs.

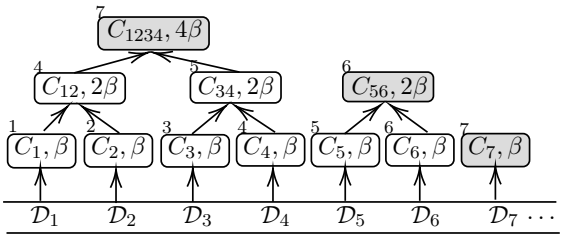


Figure 13: Merge-reduce on 7 steps with a buffer with 3 slots. The gray nodes are the ones in the buffer after the 7 steps, the numbers in the upper left corners represent the construction time of the corresponding coresets.

- Uniform sampling/per task coreset: the network is only trained on the points in the replay memory with different selection methods.
- k-means/k-center in feature/embedding/gradient space: the per-task selection retains points in the replay memory that are generated by k-means++ (Arthur and Vassilvitskii, 2007)/greedy k-center algorithm, where the clustering is done either in the original feature space, in the last layer embedding of the neural network, or in the space of the gradient with respect to the last layer (after training on the respective task). The points that are the cluster centers in different spaces are the ones chosen to be saved in the memory. We note that the k-center summarization in the last layer embedding space is the coreset method proposed for active learning by Sener and Savarese (2018).
- Hardest/max-entropy samples per task: the saved points have the highest loss after training on each task/have the highest uncertainty (as measured by the entropy of the prediction). Such selection strategies are used, among others, by Coleman et al. (2020) and Aljundi et al. (2019a).
- Training per task with FRCL's/iCaRL's selection: the points per task are selected by FRCL's inducing point selection (Titsias et al., 2020), where the kernel is chosen as the linear kernel over the last layer embeddings/iCaRL's selection (Algorithm 4 in Rebuffi et al. (2017)) performed in the normalized embedding space.
- Gradient matching per task: same as iCaRL's selection, but in the space of gradients with respect to the last layer and jointly over all classes. This is a variant of Hilbert coreset (Campbell and Broderick, 2019) with equal weights, where the Hilbert space norm is chosen to be the squared 2-norm difference of loss gradients with respect to the last layer at the maximum posterior value.

In the continual learning experiments, we train our networks for 400 epochs using Adam with step size $5 \cdot 10^{-4}$ after each task. The loss at each step consists of the loss on a minibatch of size 256 of the current tasks and loss on the replay memory scaled by β. For streaming, we train our networks for 40 gradient descent steps using Adam with step size $5 \cdot 10^{-4}$ after each batch. For Aljundi et al. (2019b), we use a streaming batch size of 10 for better performance, as indicated in Section 2.4 of the supplementary materials of Aljundi et al. (2019b). We tune the replay memory regularization strength β separately for each method from $\{0.01, 0.1, 1, 10, 100, 1000\}$ and report the best result on the test set.

Method/Memory size	50	100	200
CL uniform sampling	85.23 ± 1.84	92.80 ± 0.79	95.08 ± 0.30
CL BiCo	91.61 ± 0.78	95.81 ± 0.28	97.01 ± 0.41
Streaming reservoir sampling	83.90 ± 3.18	90.72 ± 0.97	94.12 ± 0.61
Streaming BiCo	85.32 ± 2.40	92.51 ± 1.30	95.50 ± 0.65

Table 9: Replay memory size study on SMNIST. Our method offers bigger improvements with smaller memory sizes.

In our experiments, we used the Neural Tangent Kernel as proxy. It turns out that on the data sets derived from MNIST, simpler kernels such as RBF are also good proxy choices. To illustrate this, we repeat the continual learning and streaming experiments and report the results in Table 10. For the RBF kernel $k(x, y) = \exp(-\gamma \|x - y\|_2^2)$ we set $\gamma = 5 \cdot 10^{-4}$. While the RBF kernel is a good proxy for these data sets, it fails on harder data sets such as CIFAR-10.

	Method	PMNIST	SMNIST
CL	BiCo CNTK	79.33 ± 0.51	95.81 ± 0.28
	BiCo RBF	79.95 ± 0.81	96.09 ± 0.32
VCL	BiCo CNTK	86.11 ± 0.25	84.62 ± 0.89
	BiCo RBF	86.16 ± 0.25	82.21 ± 1.35
Str.	BiCo CNTK	74.49 ± 0.69	92.57 ± 1.09
	BiCo RBF	75.85 ± 0.65	92.49 ± 0.71

Table 10: RBF vs CNTK proxies.

References

Pankaj K Agarwal, Sariel Har-Peled, Kasturi R Varadarajan, et al. Geometric approximation via coresets. *Combinatorial and Computational Geometry*, 52:1–30, 2005.

Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11254–11263. IEEE, 2019a.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 11816–11825, 2019b.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8139–8148, 2019.

David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. In *International Conference on Learning Representations (ICLR)*, 2020.

Olivier Bachem, Mario Lucic, and Andreas Krause. Practical coreset constructions for machine learning. *arXiv:1703.06476*, 2017.

Mihai Badoiu and Kenneth L. Clarkson. Smaller core-sets for balls. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 801–802. ACM/SIAM, 2003.

Maria-Florina Balcan, Andrei Broder, and Tong Zhang. Margin based active learning. In *International Conference on Computational Learning Theory (COLT)*, pages 35–50. Springer, 2007.

Afonso S Bandeira, Edgar Dobriban, Dustin G Mixon, and William F Sawin. Certifying the restricted isometry property is hard. *IEEE Transactions on Information Theory*, 59(6): 3448–3450, 2013.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5049–5059, 2019.

Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G Dimakis. Compressed sensing using generative models. In *International Conference on Machine Learning (ICLR)*, pages 537–546, 2017.

Zalán Borsos, Mojmir Mutný, and Andreas Krause. Coresets via bilevel optimization for continual learning and streaming. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 14879–14890, 2020.

Zalán Borsos, Marco Tagliasacchi, and Andreas Krause. Semi-supervised batch active learning via bilevel optimization. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3495–3499. IEEE, 2021.

Trevor Campbell and Tamara Broderick. Bayesian coreset construction via greedy iterative geodesic ascent. In *International Conference on Machine Learning, (ICML)*, pages 697–705, 2018.

Trevor Campbell and Tamara Broderick. Automated scalable Bayesian inference via Hilbert coresets. *The Journal of Machine Learning Research*, 20(1):551–588, 2019.

Emmanuel J Candes, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223, 2006.

Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 10(3):273–304, 1995.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. Continual learning with tiny episodic memories. *arXiv:1902.10486*, 2019.

Bernard Chazelle and Jiři Matoušek. On linear-time deterministic algorithms for optimization problems in fixed dimension. *Journal of Algorithms*, 21(3):579–597, 1996.

Ke Chen. *Matrix Preconditioning Techniques and Applications*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2005.

Aristotelis Chrysakis and Marie-Francine Moens. Online continual learning from imbalanced data. In *International Conference on Machine Learning (ICML)*, pages 1952–1961, 2020.

Kenneth L. Clarkson. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. *ACM Transactions on Algorithms*, 6(4):63:1–63:30, 2010.

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 (ICLR)*, 2020.

R. Dennis Cook and Sanford Weisberg. Characterizations of an empirical influence function for detecting influential cases in regression. *Technometrics*, 22(4):495–508, 1980.

R Dennis Cook and Sanford Weisberg. *Residuals and influence in regression*. New York: Chapman and Hall, 1982.

David L Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4): 1289–1306, 2006.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the L1-ball for learning in high dimensions. In *International Conference on Machine learning (ICML)*, pages 272–279, 2008.

Sean Ryan Fanello, Carlo Ciliberto, Matteo Santoro, Lorenzo Natale, Giorgio Metta, Lorenzo Rosasco, and Francesca Odone. icub world: Friendly robots help building good vision data-sets. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 700–705. IEEE, 2013.

Valerii V. Fedorov. *Theory of optimal experiments*. Probability and mathematical statistics. Academic Press, New York, NY, USA, 1972.

Dan Feldman. Introduction to core-sets: an updated survey. *arXiv:2011.09384*, 2020.

Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *ACM symposium on Theory of computing - STOC '11*, pages 569–578. ACM, ACM Press, 2011.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, pages 1126–1135, 2017.

Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. Forward and reverse gradient-based hyperparameter optimization. In *International Conference on Machine Learning (ICML)*, pages 1165–1173, 2017.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning (ICML)*, pages 1568–1577, 2018.

Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics*, 3(1-2):95–110, 1956. ISSN 0028-1441, 1931-9193.

R French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999. ISSN 1364-6613.

Mingfei Gao, Zizhao Zhang, Guo Yu, Sercan O Arik, Larry S Davis, and Tomas Pfister. Consistency-based semi-supervised active learning: Towards minimizing labeling cost. *arXiv:1910.07153*, 2019.

Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

Yuhong Guo and Dale Schuurmans. Discriminative batch mode active learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 593–600, 2008.

Sariel Har-peled. *Geometric Approximation Algorithms*. American Mathematical Society, USA, 2011.

Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *ACM symposium on Theory of Computing (STOC)*, pages 291–300. ACM, ACM Press, 2004.

Tyler L. Hayes, Nathan D. Cahill, and Christopher Kanan. Memory efficient experience replay for streaming learning. In *2019 International Conference on Robotics and Automation (ICRA)*, 2019.

Steven C.H. Hoi and M.R. Lyu. A semi-supervised active learning framework for image retrieval. In *Computer Vision and Pattern Recognition (CVPR)*, pages 302–309 vol. 2. IEEE, 2005.

Steven C.H. Hoi, Rong Jin, Jianke Zhu, and Michael R. Lyu. Batch mode active learning and its application to medical image classification. In *International Conference on Machine learning (ICML)*, pages 417–424, 2006.

Jonathan Huggins, Trevor Campbell, and Tamara Broderick. Coresets for scalable bayesian logistic regression. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4080–4088, 2016.

Zohar Jackson. Free spoken digit dataset, 2016. `https://github.com/Jakobovski/free-spoken-digit-dataset`.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems (NeurIPS)*, pages 8571–8580, 2018.

Ajil Jalal, Sushrut Karmalkar, Alexandros G Dimakis, and Eric Price. Instance-optimal compressed sensing via posterior sampling. In *International Conference on Machine Learning (ICML)*, 2021.

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. Glister: Generalization based data subset selection for efficient and robust learning. *AAAI Conference on Artificial Intelligence*, 35(9):8110–8118, 2021.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 7026–7037, 2019.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning (ICML)*, pages 1885–1894, 2017.

Andreas Krause and Volkan Cevher. Submodular dictionary selection for sparse representation. In *International Conference on Machine Learning (ICML)*, pages 567–574, 2010.

Michael Langberg and Leonard J. Schulman. Universal ϵ-approximators for integrals. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 598–607. SIAM, Society for Industrial and Applied Mathematics, 2010.

Yan Leng, Xinyan Xu, and Guanghui Qi. Combining active learning and semi-supervised learning to construct SVM classifier. *Knowledge-Based Systems*, 44:121–131, 2013.

David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer, 1994.

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-SGD: Learning to learn quickly for few-shot learning. *arXiv:1707.09835*, 2017.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations (ICLR)*, 2019.

Francesco Locatello, Michael Tschannen, Gunnar Rätsch, and Martin Jaggi. Greedy algorithms for cone constrained optimization with convergence guarantees. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 773–784, 2017.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6467–6476, 2017.

Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1540–1552, 2020.

Mario Lucic, Matthew Faulkner, Andreas Krause, and Dan Feldman. Training Gaussian mixture models at scale via coresets. *The Journal of Machine Learning Research*, 18(1): 5885–5909, 2017.

David J.C. MacKay. Information-based objective functions for active data selection. *Neural Comput.*, 4(4):590–604, 1992.

Michael W. Mahoney and Petros Drineas. Cur matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3):697–702, 2009.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pages 109–165. Elsevier, 1989.

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

Toby J. Mitchell. An algorithm for the construction of "D-Optimal" experimental designs. *Technometrics*, 16(2):203, 1974.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational continual learning. In *International Conference on Learning Representations (ICLR)*, 2018.

Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Neural Tangents: Fast and easy infinite neural networks in Python. In *International Conference on Learning Representations (ICLR)*, 2020.

Barak A Pearlmutter. Fast exact multiplication by the Hessian. *Neural computation*, 6(1): 147–160, 1994.

Fabian Pedregosa. Hyperparameter optimization with approximate gradient. In *International Conference on Machine Learning (ICML)*, pages 737–746, 2016.

Jeff M. Phillips. Coresets and sketches. *arXiv:1601.00617*, 2016.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2001–2010. IEEE, 2017.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv:1606.04671*, 2016.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520. IEEE, 2018.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations (ICLR)*, 2018.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2990–2999, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 596–608, 2020.

Shuang Song, David Berthelot, and Afshin Rostamizadeh. Combining mixmatch and active learning for better accuracy with fewer labels. *arXiv:1912.00594*, 2019.

Javier Tapia, Espen Knoop, Mojmir Mutný, Miguel A. Otaduy, and Moritz Bächer. Makesense: Automated sensor design for proprioceptive soft robots. *Soft Robotics*, 7(3):332–345, 2020.

Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Robert Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

Michalis K. Titsias, Jonathan Schwarz, Alexander G. de G. Matthews, Razvan Pascanu, and Yee Whye Teh. Functional regularisation for continual learning with Gaussian processes. In *International Conference on Learning Representations (ICLR)*, 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 (ICLR)*, 2019.

Ivor W. Tsang, James T. Kwok, and Pak-Ming Cheung. Core vector machines: Fast SVM training on very large data sets. *Journal of Machine Learning Research*, 6(13):363–392, 2005.

Murad Tukan, Alaa Maalouf, and Dan Feldman. Coresets for near-convex functions. *Advances in Neural Information Processing Systems (NeurIPS)*, pages 997–1009, 2020.

Andrew V Uzilov, Joshua M Keegan, and David H Mathews. Detection of non-coding RNAs on the basis of predicted secondary structure formation free energy change. *BMC Bioinf.*, 7(1):173, 2006.

Laurens van der Maaten and Geoffrey Hinton. Visualizing non-metric similarities in multiple maps. *Mach Learn*, 87(1):33–55, 2011.

L. Vicente, G. Savard, and J. Júdice. Descent approaches for quadratic bilevel programming. *Journal of Optimization Theory and Applications*, 81(2):379–399, 1994.

Luis N. Vicente and Paul H. Calamai. Bilevel and multilevel programming: A bibliography review. *Journal of Global Optimization*, 5(3):291–306, 1994.

Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1):37–57, 1985.

H. von Stackelberg and A. Peacock. *The Theory of the market economy*. Hodge, 1952.

Tianyang Wang, Jun Huan, and Bo Li. Data dropout: Optimizing training data for convolutional neural networks. In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 39–46. IEEE, IEEE, 2018.

P. Warden. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *arXiv:1804.03209*, 2018.

Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In Francis Bach and David Blei, editors, *International Conference on Machine Learning (ICML)*, pages 1954–1963, 2015.

Yuhuai Wu, Mengye Ren, Renjie Liao, and Roger Grosse. Understanding short-horizon bias in stochastic meta-optimization. In *International Conference on Learning Representations (ICLR)*, 2018.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv:1605.07146*, 2016.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning (ICML)*, pages 3987–3995, 2017.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *International Conference on Learning Representations (ICLR)*, 2021.

Xiaojin Zhu, Zoubin Ghahramani, and John D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *International Conference (ICML)*, pages 912–919, 2003.