Coresets via Bilevel Optimization for Continual Learning and Streaming

Zalán Borsos
Dept. of Computer Science
ETH Zurich
zalan.borsos@inf.ethz.ch

Mojmír Mutný
Dept. of Computer Science
ETH Zurich
mojmir.mutny@inf.ethz.ch

Andreas Krause
Dept. of Computer Science
ETH Zurich
krausea@ethz.ch

Abstract

Coresets are small data summaries that are sufficient for model training. They can be maintained online, enabling efficient handling of large data streams under resource constraints. However, existing constructions are *limited* to *simple models* such as k-means and logistic regression. In this work, we propose a novel coreset construction via *cardinality-constrained bilevel optimization*. We show how our framework can efficiently generate coresets for deep neural networks, and demonstrate its empirical benefits in continual learning and in streaming settings.

1 Introduction

More and more applications rely on predictive models that are learnt online. A crucial, and in general open problem is to reliably maintain accurate models as data arrives over time. *Continual learning*, for example, refers to the setting where a learning algorithm is applied to a sequence of tasks, without the possibility of revisiting old tasks. In the *streaming* setting, the data arrives sequentially and the notion of task is not defined. For such practically important settings where data arrives in a non-iid manner, the performance of models can degrade arbitrarily. This is especially problematic in the non-convex setting of deep learning, where this phenomenon is referred to as *catastrophic forgetting* [42, 23].

One of the oldest and most efficient ways to combat catastrophic forgetting is the *replay memory*-based approach, where a small subset of past data is maintained and revisited during training. In this work, we investigate how to effectively generate and maintain such summaries via *coresets*, which are small, weighted subsets of the data. They have the property that a model trained on the coreset performs almost as well as when trained on the full dataset. Moreover, coresets can be effectively maintained over data streams, thus yielding an efficient way of handling massive datasets and streams.

Contributions. We present a novel and general coreset construction framework,[1] where we formulate the coreset selection as a *cardinality-constrained bilevel optimization* problem that we solve by greedy forward selection via matching pursuit. Using a reformulation via a proxy model, we show that our method is especially well suited for replay memory-based continual learning and streaming with neural networks. We demonstrate the effectiveness of our approach in an extensive empirical study. A demo of our method for streaming can be seen in Figure 1.

2 Related Work

Continual Learning and Streaming. Continual learning with neural networks has received an increasing interest recently. The approaches for alleviating catastrophic forgetting fall into three main categories: using weight regularization to restrict deviation from parameters learned on old tasks

[1] Our coreset library is available at `https://github.com/zalanborsos/bilevel_coresets`.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

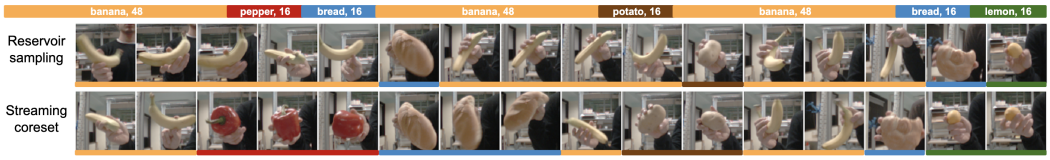


Figure 1: Data summarization on an imbalanced stream of images created from the iCub World 1.0 dataset [16]. First row: the stream's composition containing 5 object classes. Second row: selection by reservoir (uniform) sampling. Third row: selection by our method. Reservoir sampling misses classes (pepper) due to imbalance and does not choose diverse samples, in contrast to our method.

[32, 43]; architectural adaptations for the tasks [48]; and replay-based approaches, where samples from old tasks are either reproduced via a replay memory [39] or via generative models [50]. In this work, we focus on the replay-based approach, which provides strong empirical performance [9], despite its simplicity. In contrast, the more challenging setting of streaming using neural networks has received little attention. To the best of our knowledge, the replay-based approach to streaming has been tackled by [2, 27, 11], which we compare against experimentally.

Coresets. Several coreset definitions exist, each resulting in a different construction method. A popular approach is to require uniform approximation guarantees on the loss function, e.g., coresets for k-means [19], Gaussian mixture model [41] and logistic regression [29]. However, this approach produces a large coreset for more complex hypothesis classes, prohibiting its successful application to models such as neural networks. Several approaches relax the uniform approximation guarantees: the Hilbert coreset [7] which poses the subset selection problem as a vector approximation in a normed vector space, and data selection techniques that were successfully employed in active learning with convolutional neural networks [56, 49].[2] The term "coreset" in the continual learning literature is used more loosely and in most cases it is a synonym for the samples kept in the replay memory [43, 17], with the most common construction methods being uniform sampling and clustering in feature or embedding space. While in [27] the authors compare several data selection methods on streaming, to the best of our knowledge, no thorough analysis has been conducted on the effect of the data summarization strategy in replay memory-based continual learning, which we address with an extensive experimental study.

Bilevel optimization. Modeling hierarchical decision making processes [54], bilevel optimization has witnessed an increasing popularity in machine learning recently, with applications ranging from meta-learning [20], to hyperparameter optimization [45, 21] and neural architecture search [37]. We use the framework of bilevel optimization to generate coresets. Closest to our work, bilevel optimization was used in [47] to reweigh samples to overcome training set imbalance or corruption, and sensor selection was approached via bilevel optimization in [51]. We note that, while the latter work uses a similar strategy for sensor subset selection to ours, we investigate the different setting of weighted data summarization for continual learning and streaming with neural networks.

3 Coresets via Bilevel Optimization

We first present our coreset construction given a *fixed* dataset $\mathcal{X} = \{(x_i, y_i)\}_{i=1}^n$.[3] We consider a weighted variant of empirical risk minimization (ERM) where our goal is to minimize $L(\theta, w) = \sum_{i=1}^n w_i \ell_i(\theta)$, where $\ell_i(\theta) = \ell(x_i, y_i; \theta)$, and $w = \{w_1, \ldots, w_n\} \in \mathbb{R}_+^n$ is a set of positive weights. Standard ERM is recovered when $w_i = 1$, $\forall i \in [n]$, in which case we simply write $L(\theta)$. A *coreset* is a weighted subset of $\mathcal{X}$, equivalently represented by a sparse vector $\hat{w}$, where points having zero weights are not considered to be part of the coreset. A good coreset ensures that $L(\theta, \hat{w})$ is a good proxy for $L(\theta)$, i.e., optimizing on $L(\theta, \hat{w})$ over θ yields good solutions when evaluated on $L(\theta)$. In this spirit, a natural goal thus would be to determine a set of weights $\hat{w}$, such that we minimize

$$\hat{w} \in \underset{w \in \mathbb{R}_+^n, ||w||_0 \leq m}{\arg\min} \; L(\theta^*(w)) \text{ s.t. } \theta^*(w) \in \underset{\theta}{\arg\min}\, L(\theta, w), \tag{1}$$

[2] Although these methods were not designed for replay-based continual learning or streaming, they can be employed in these settings; we thus compare them with our method empirically.

[3] Our construction also applies in the unsupervised setting.

where $m \leq n$ is a constraint on the coreset size. This problem is an instance of *bilevel optimization*, where we minimize an *outer* objective, here $L(\theta^*(w))$, which in turn depends on the solution $\theta^*(w)$ to an *inner* optimization problem $\arg\min_\theta L(\theta, w)$. Before presenting our algorithm for solving (1), we present some background on bilevel optimization.

3.1 Background: Bilevel Optimization

Suppose $g : \Theta \times \mathcal{D} \to \mathbb{R}$ and $f : \Theta \times \mathcal{D} \to \mathbb{R}$, then the general form of bilevel optimization is

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

Our data summarization problem is recovered with $g(\theta^*(w), w) = L(\theta^*(w))$ and $f(\theta, w) = L(\theta, w)$. The general optimization problem above is known to be NP-hard even if both sets Θ and $\mathcal{D}$ are polytopes and both g and f are linear. Provably correct solvers rely on branch and bound techniques [5]. For differentiable g and twice differentiable f, the above problem can however be heuristically solved via first-order methods. Namely, the constraint $\theta^*(w) \in \arg\min_{\theta \in \Theta} f(\theta, w)$ can be relaxed to $\frac{\partial f(\theta^*, w)}{\partial \theta^*} = 0$. This relaxation is tight if f is strictly convex. A crucial result that allows us apply first-order methods by enabling the calculation of the gradient of G with respect to w is the *implicit function theorem* applied to $\frac{\partial f(\theta^*, w)}{\partial \theta^*} = 0$. Combined with the the total derivative and the chain rule,

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

where the terms are evaluated at $\theta = \theta^*(w)$. At each step of the gradient descent on the outer objective, the inner objective needs to be solved to optimality. While this heuristic may be converging only to a stationary point [24], such solutions are successful in many applications [45, 51].

3.2 Warm-up: Least Squares Regression, and Connections to Experimental Design

We first instantiate our approach to the problem of weighted and regularized least squares regression. In this case, the inner optimization problem, $\hat{\theta}(w) = \arg\min_\theta \sum_{i=1}^n w_i (x_i^\top \theta - y_i)^2 + \lambda ||\theta||_2^2$, admits a closed-form solution. For this special case there are natural connections to the literature on *optimal experimental design*, a well-studied topic in statistics [8].

The data summarization problem with outer objective $g(\hat{\theta}) = \sum_{i=1}^n (x_i^\top \hat{\theta}(w) - y_i)^2$ is closely related to *Bayesian V-optimal design*. Namely, under standard Bayesian linear regression assumptions $y = X\theta + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$ and $\theta \sim \mathcal{N}(0, \lambda^{-1})$, the summarization objective $\mathbb{E}_{\epsilon,\theta}[g(\hat{\theta})]$ and the Bayesian V-experimental design outer objective $g_V(\hat{\theta}) = \mathbb{E}_{\epsilon,\theta}[\|X^\top(\hat{\theta} - \theta)\|_2^2]$ differ by $\sigma^2/2$ in the infinite data limit, as shown in Proposition 6 in Appendix A. 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. We defer the detailed discussion to Appendix A.

Using g_V as our outer objective, solving the inner objective in closed form, we identify the Bayesian V-experimental design objective,

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

where $D(w) := \text{diag}(w)$. In Lemma 8 in Appendix A we show that $G(w)$ is smooth and convex in w when the integrality is relaxed. This, together with the relationship between g and g_V, suggests that first order methods for solving the bilevel data summarization objective can be expected to succeed, at least in the large data limit and the special case of least squares regression.

3.3 Incremental Subset Selection

One challenge that we have not addressed yet is how to deal with the cardinality constraint $||w||_0 \leq m$. A natural attempt is, in the spirit of the Lasso [52], to transform the $\|w\|_0$ constraint into the $\|w\|_1$ constraint. However, the solution of the inner optimization is unchanged if the weights

and the regularizer are rescaled by a common factor, rendering norm-based sparsity regularization meaningless. A similar observation was made in [51]. On the other hand, a greedy combinatorial approach would treat G as a set function and increment the set of selected points by inspecting marginal gains. It turns out that Bayesian V-experimental design is approximately *submodular*, as shown in Proposition 7 in Appendix A. As a consequence, at least for the linear regression case, such greedy algorithms produce provably good solutions, if we restrict the weights to be binary [15, 26]. Unfortunately, for general losses the greedy approach comes at a significant cost: at each step, the bilevel optimization problem of finding the optimal weights must be solved for each point which may be added to the coreset. This makes greedy selection impractical.

Selection using Matching Pursuit. In this work, we propose an efficient solution summarized in Algorithm 1 based on *cone constrained generalized matching pursuit* [38]. With the atom set $\mathcal{A}$ corresponding to the standard basis of $\mathbb{R}^n$, generalized matching pursuit proceeds by incrementally increasing the active set of atoms that represents the solution by selecting the atom that minimizes the linearization of the objective at the current iterate. The benefit of this approach is that incrementally growing the atom set can be stopped when the desired size m is reached and thus the $\|w\|_0 \leq m$ constraint is active. We use the fully-corrective variant of the algorithm, where, once a new atom is added, the weights are fully reoptimized by gradient descent using the implicit gradient (Eq. (3)) with projection to positive weights.

Algorithm 1 Coresets via Bilevel Optimization

Input: Data $\mathcal{X} = \{(x_i, y_i)\}_{i=1}^n$, coreset size m
Output: weights w encoding the coreset
$w = [0, \ldots, 0]$
Choose $i \in [n]$ randomly, set $w_i = 1$, $S_1 = \{i\}$.
for $t \in [2, ..., m]$ **do**
  Find $w^*_{S_{t-1}}$ local min of $G(w)$ by projected GD
  s.t. supp$(w^*_{S_{t-1}}) = S_{t-1}$.
  $k^* = \arg\min_{k \in [n]} \nabla_{w_k} G(w^*_{S_{t-1}})$
  $S_t = S_{t-1} \cup \{k^*\}$, $w_{k^*} = 1$
end for

Suppose a set of atoms $S_t \subset \mathcal{A}$ of size t 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. Then we optimize to find the weights $w^*_{S_t}$ with domain support restricted to S_t that represent a local minimum of $G(w)$ defined in Eq. 2 with $g(\theta^*(w), w) = L(\theta^*(w))$ and $f(\theta, w) = L(\theta, w)$. Once these weights are found, the algorithm increments S_t with the atom that minimizes the first order Taylor expansion of the outer objective around $w^*_{S_t}$,

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

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 largest negative implicit gradient (Eq. (3)).

We can gain an insights into the selection rule in Eq. (4) by expanding $\nabla_w G$ using Eq. (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 $\frac{\partial^2 f(\theta, w)}{\partial w_k \partial \theta^\top} = \nabla_\theta \ell_k(\theta)$ we can expand Eq. (4) to get,

$$k^* = \operatorname*{argmax}_{k \in [n]} \nabla_\theta \ell_k(\theta)^\top \left(\frac{\partial^2 f(\theta, w^*_{S_t})}{\partial \theta \partial \theta^\top} \right)^{-1} \nabla_\theta g(\theta), \tag{5}$$

where the gradients and partial derivatives are evaluated at $w^*_{S_t}$ and $\theta^*(w^*_{S_t})$. Thus with the choice $g(\theta) = \sum_{i=1}^n \ell_i(\theta)$, the selected point's 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.

Theoretical Guarantees. If the overall optimization problem is convex (as in the case of Bayesian V-experimental design, Lemma 8 in Appendix A), one can show that cone constrained generalized matching pursuit provably converges. Following [38], the convergence of the algorithm depends on properties of the atom set $\mathcal{A}$, which we do not review here in full.

Theorem 1 (cf. Theorem 2 of [38])**.** *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 $t = 1$.

Thus, by iterating $m-1$ times, we reach the cardinality constraint. Note that by imposing a bound on the weights as commonly done in experimental design [18], our algorithm would be equivalent to Frank-Wolfe [22]. While in general the function G might not be convex for more complex models, we nevertheless demonstrate the effectiveness of our method empirically in such scenarios in Section 6.

3.4 Relation to Influence Functions

It turns out our approach is also related to incremental subset selection via *influence functions*. The empirical influence function, known from robust statistics [13], denotes the effect of a single sample on the estimator. Influence functions have been recently used in [34] to understand the dependence of neural network predictions on a single training point and to generate adversarial training examples. To uncover the relationship of our method to influence functions, let us consider the influence of the k-th point on the outer objective. Suppose that 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

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

Following [34] and using the result of [14], under twice differentiability and strict convexity of the inner loss, the empirical influence function at k is $\frac{\partial \theta^*}{\partial \varepsilon}\big|_{\varepsilon=0} = -\left(\frac{\partial^2 \sum_{i=1}^{n} w_{S,i}^* \ell_i(\theta^*)}{\partial \theta \partial \theta^\top}\right)^{-1} \nabla_\theta \ell_k(\theta^*)$. Now, applying the chain rule to $\mathcal{I}(k)$, as shown in Proposition 9 in Appendix B, we can see that $\operatorname{argmax}_k \mathcal{I}(k)$ and the selection rule in Equation (5) are the same.

4 Coresets for Neural Networks

While our proposed coreset framework is generally applicable to any twice differentiable model, we showcase it for the challenging case of deep neural networks. In applying Algorithm 1 to a neural network with a large number of parameters, inverting the Hessian in each outer iteration (Eq. (3)) is an impeding factor. Several works propose Hessian-vector product approximations, for example, through the conjugate gradient algorithm [45] or Neumann series [40]. While applicable in our setting, these methods become costly depending on the number of coreset points and outer iterations.

We propose an alternative approach that results in a large speedup for small coreset size (max. 500 points). The key idea is to use a *proxy model* in the bilevel optimization that provides a good data summary for the original model. In this work, our choice for proxies are functions in a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ with associated kernel k, possibly adjusted to the neural network architecture. We assume k to be positive definite for simplicity and we use the same convex loss as for the neural network. Now, the coreset generation problem with regularized inner objective transforms into

$$\min_{w \in \mathbb{R}_+^n,\, \|w\|_0 \leq m} g(h^*) \text{ s.t. } h^* = \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{n} w_i \ell_i(h) + \lambda \|h\|_{\mathcal{H}}^2 .$$

The regularizer λ could also be optimized jointly with w, but we use fixed values in our applications. Now suppose we have selected a subset of atoms and denote their index set as S. From the *representer theorem* we know that the inner problem admits the representation $h^*(\cdot) = \alpha^\top K_{S,\cdot}$, where $\alpha \in \mathbb{R}^{|S|}$ and K is the Gram matrix associated with the data. Now the bilevel optimization problem for optimizing the weights takes the form

$$\min_{w \in \mathbb{R}_+^n,\ \text{supp}(w)=S} \quad g(\alpha^{*\top} K_{S,\cdot}), \quad \text{s.t. } \alpha^* = \arg\min_{\alpha \in \mathbb{R}^{|S|}} \sum_{i \in S} w_i \ell_i(\alpha^\top K_{S,i}) + \lambda \alpha^\top K_{S,S} \alpha. \tag{6}$$

That is, with the help of the representer theorem, we can reduce the size of the inner level parameters to at most the size m of the coreset. This allows us to use fast solvers for the inner problem, and to use the aforementioned approximate inverse Hessian-vector product methods when calculating the implicit gradient. In this work, we use the conjugate gradient method [45].

The choice of $\mathcal{H}$ is crucial for the success of the proxy reformulation. For neural networks, we propose to use as proxies their corresponding *Neural Tangent Kernels (NTK)* [30], which are fixed kernels characterizing the network's training with gradient descent in the infinite-width limit. While other proxy choices are possible (e.g., fixed last-layer embedding with linear kernel on top, or even RBF kernel, see Appendix D) we leave their investigation for future work.

Computational Cost. In the standard formulation, each inner iteration is performed using a small number of SGD steps to find an approximate minimizer of the inner optimization problem. The bottleneck is introduced by the outer descent step: the implicit gradient approximation via 30 conjugate gradient steps requires one minute for a ResNet-18 on a GPU, which makes the approach impractical for weighted coreset selection. The proxy reformulation reduces the number of parameters to $\mathcal{O}(m)$ and its time complexity depends cubically on the coreset size m. As a result, an outer descent step is $200\times$ faster in the proxy compared to the standard formulation. On the other hand, the proxy reformulation introduces the overhead of calculating the proxy kernel — for calculating the NTK efficiently, we rely on the library of [44]. We measure the runtime of coreset generation in the proxy formulation in Section 6.3. Further speedups are discussed in Appendix E.

Limitations. While our proposed framework excels at small coreset sizes, generating summaries larger than 500 incurs significant computational overhead. A possible remedy is to perform the greedy selection step in batches or to perform greedy elimination instead of forward selection. In terms of theoretical guarantees, due to the hardness of the cardinality-constrained bilevel optimization problem, our method is a heuristic for coreset selection for neural networks.

5 Applications in Continual Learning and Streaming Deep Learning

We now demonstrate how our coreset construction can achieve significant performance gains in continual learning and in the more challenging streaming settings with neural networks. We build on approaches that alleviate catastrophic forgetting by keeping representative past samples in a replay memory. Our goal is to compare our method to other data summarization strategies for managing the replay memory. We keep the network structure fixed during training. This is termed 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 [17].

For *continual learning with replay memory* we employ the following protocol. The learning algorithm receives data $\mathcal{X}_1, \ldots, \mathcal{X}_T$ arriving in order from T different tasks. At time t, the learner receives $\mathcal{X}_t$ 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, with weights equal to 1. Thus, the optimization objective at time t is

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

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, $\mathcal{X}_t$ 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.

The *streaming setting* is more challenging: we assume the learner is faced with small data batches $\mathcal{X}_1, ..., \mathcal{X}_T$ arriving in order, but the batches do not contain information about the task boundaries. In fact, even the notion of tasks might not be defined. 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

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

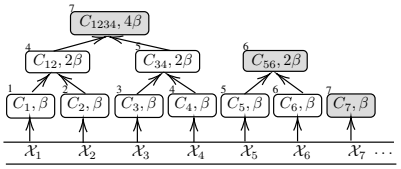
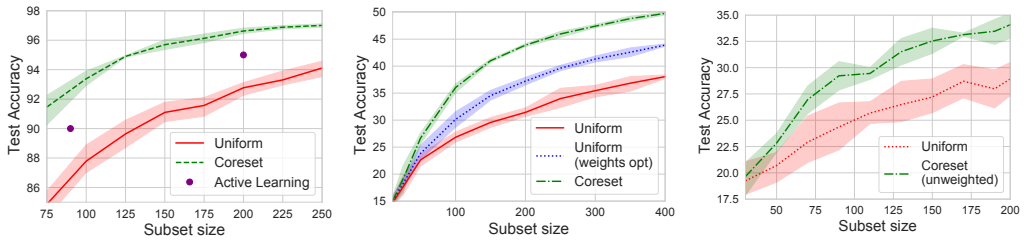


Figure 2: Merge-reduce on 7 steps with a buffer with 3 slots. The grey nodes are in the buffer after the 7 steps, the numbers in the upper left corners represent the construction time of the corresponding coresets.

Streaming Coresets via Merge-Reduce. Managing the replay memory is crucial for the success of our method in streaming. 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. Relying on this idea, we use a variant of the merge-reduce framework of [10]. For this,



(a) CNN on MNIST (b) KRR with CNTK on CIFAR-10 (c) ResNet-18 on CIFAR-10

Figure 3: a) Performance of a CNN trained on subsets of MNIST. b) Kernel ridge regression (KRR) on subsets of CIFAR-10. Our method obtains almost 50% test accuracy on CIFAR-10 when trained on only 400 points. c) ResNet-18 trained on subsets of CIFAR-10. Coresets have binary weights.

we divide the buffer into s equally-sized slots. We associate regularizers 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 β and it is appended to the buffer, which now might contain an extra slot. The reduction to size m happens as follows: select consecutive slots i and $i+1$ based on Algorithm 3 in Appendix C, then join the contents 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 (Algorithm 2 in Appendix C). We illustrate the merge-reduce coreset construction for a buffer with 3 slots and 7 steps in Figure 2.

6 Experiments

We demonstrate the empirical effectiveness of our method in various settings. In all our experiments we generate the coreset via our proxy formulation. For each neural network architecture we calculate the corresponding (convolutional) neural tangent kernel without pooling using the library of [44].

6.1 Dataset summarization

We showcase our method by training a convolutional neural network (CNN) on a small subset of MNIST selected by our coreset construction. The CNN consists of two blocks of convolution, dropout, max-pooling and ReLU activation, where the number of filters are 32 and 64 and have size 5x5, followed by two fully connected layers of size 128 and 10 with dropout. The dropout probability is 0.5. The CNN is trained on the data summary using Adam with learning rate $5 \cdot 10^{-4}$. The results for summarizing MNIST are shown in Figure 3a, where we plot the test accuracy against the summary size over 5 random seeds for uniform sampling, coreset generation, and for the state-of-the-art in active learning for our chosen CNN architecture [33].

Our next experiment follows [3] and solves classification on CIFAR-10 [35] via kernelized ridge regression (KRR) applied to the one-hot-encoded labels Y. KRR can be solved on a subset S in closed-form $\alpha^* = (D(w_S)K_{S,S} + \lambda\mathbb{I})^{-1}D(w_S)Y_S$, which replaces the inner optimization problem. We solve the coreset selection for KRR using the CNTK proposed in [3] with 6 layers and global average pooling, with normalization. We also experiment with uniform sampling of points with weights optimized through bilevel optimization. The results are shown in Figure 3b. We observe that with CNTK we can obtain a test accuracy of almost 50% with only 400 samples. We further validate our proxy formulation by selecting subsets of CIFAR-10 and training a variant of ResNet-18 [28] without batch normalization on the chosen subsets. We restrict the coreset weights to binary in order to show that choosing representative *unweighted* points can alone improve the performance, as illustrated in Figure 3c.

6.2 Continual Learning

We next validate our method in the replay memory-based approach to continual learning. We use the following 10-class classification datasets:

- **PermMNIST** [25]: consist of 10 tasks, where in each task all images' pixels undergo the same fixed random permutation.

Table 1: 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 over 5 runs with different random seeds. Our coreset construction performs among the best on all datasets.

Method	PermMNIST	SplitMNIST	SplitCIFAR-10
Uniform sampling	78.46 ± 0.40	92.80 ± 0.79	$\mathbf{36.20 \pm 3.19}$
k-means of features	78.34 ± 0.49	93.40 ± 0.56	33.41 ± 2.48
k-center of embeddings	78.57 ± 0.58	93.84 ± 0.78	$\mathbf{36.91 \pm 2.42}$
Hardest samples	76.79 ± 0.55	89.62 ± 1.23	28.10 ± 1.79
iCaRL's selection	$\mathbf{79.68 \pm 0.41}$	93.99 ± 0.39	34.52 ± 1.62
Coreset	$\mathbf{79.26 \pm 0.43}$	$\mathbf{95.87 \pm 0.20}$	$\mathbf{37.60 \pm 2.41}$

Table 2: Upper: VCL with 20 summary points / task. VCL can benefit from our coreset. Lower: Streaming with buffer size 100. Coreset methods use the merge-reduce buffer.

	Method	PermMNIST	SplitMNIST
VCL	k-center	85.33 ± 0.67	65.71 ± 3.17
	Uniform	84.96 ± 0.17	80.06 ± 2.19
	Coreset	$\mathbf{86.18 \pm 0.21}$	$\mathbf{84.66 \pm 0.66}$
Stream	Train on coreset only	45.03 ± 1.31	89.99 ± 0.76
	Reservoir sampling	73.21 ± 0.59	90.72 ± 0.97
	Streaming coreset	$\mathbf{74.44 \pm 0.52}$	$\mathbf{92.59 \pm 1.20}$

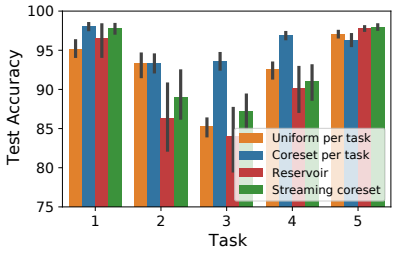


Figure 4: Per-task test accuracy on SplitMNIST. Out method represents most of the tasks better than uniform / reservoir sampling.

- **SplitMNIST** [57]: MNIST is split into 5 tasks, where each task consists of distinguishing between consecutive image classes.
- **SplitCIFAR-10**: similar to SplitMNIST on CIFAR-10.

We keep a subsample of 1000 points for each task for all datasets, while we retain the full test sets. For PermMNIST we use a fully connected net with two hidden layers with 100 units, ReLU activations, and dropout with probability 0.2 on the hidden layers. For SplitMNIST we use the CNN described in Section 6.1. We fix the replay memory size $m = 100$ for these task. For SplitCIFAR-10 we the ResNet presented in Section 6.1 and 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 exhaustive comparison of our method to other data selection methods proposed in the continual learning or the coreset literature, under the protocol described in Section 5. These include, among others, k-center clustering in last layer embedding [49] and feature space [43], iCaRL's selection [46] and retaining the hardest-to-classify points [1]. 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 method in all of the continual learning and streaming experiments to using *binary coreset weights* only. A selection of results is presented in Table 1, while the full list is available in Appendix D. Our coreset construction consistently performs among the best on all datasets. In Appendix D we present a study of the effect of the replay memory size.

Our method can also be combined with different approaches to continual learning, such as VCL [43]. While VCL also relies on coresets, it was proposed with uniform and k-center summaries. We replace these with our coreset construction, and, following [43], we conduct an experiment using a single-headed two-layer network with 256 units per layer and ReLU activations, where the coreset size is set to 20 points per task. The results in Table 2 corroborate the advantage of our method over simple selection rules, and suggest that VCL can benefit from representative coresets.

6.3 Streaming

We now turn to the more challenging streaming setting, which is oblivious to the existence of tasks. For this experiment, we modify PermMNIST and SplitMNIST by first concatenating all tasks for each dataset and then streaming them in batches of size 125. We fix the replay memory size to $m = 100$

and set the number of slots $s = 10$. We train our networks 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 previous experiments.

We compare our coreset selection to reservoir sampling [55] and the sample selection methods of [2] and [27]. We were unable to tune the latter two to outperform reservoir sampling, except [2] on PermMNIST, achieving test accuracy of 74.43 ± 1.02. Table 2 confirms the dominance of our strategy over the competing methods and the validity of the merge-reduce framework. For inspecting the gains obtained by our method over uniform / reservoir sampling, we plot the final per task test accuracy on SplitMNIST in Figure 4. We notice that the advantage of the coreset method does not come from excelling at one particular task, but rather by representing the majority of tasks better than uniform sampling. 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, as we show in the following experiment.

Table 3: Imbalanced streaming on SplitMNIST and SplitCIFAR-10. Our proposed method is competitive with strategies designed for imbalanced streams.

Method	**SplitMNIST**	**SplitCIFAR-10**
Reservoir	80.60 ± 4.36	27.22 ± 1.24
CBRS	89.71 ± 1.31	$\mathbf{32.25 \pm 1.69}$
Coreset	$\mathbf{92.30 \pm 0.23}$	$\mathbf{33.98 \pm 1.44}$

Table 4: Runtimes for generating coresets out of 1000 points with CNTKs for the CNN and ResNet-18 described in Sec. 6.1.

Op / CNTK	**CNN**	**ResNet-18**
Kernel calc.	6.3 s	56.2 s
Coreset 100	21.2 s	25.4 s
Coreset 400	186.7 s	188.9 s

Imbalanced Streaming. The setup of the streaming experiment favors reservoir sampling, as the data in the stream from different tasks is balanced. We illustrate the benefit of our method in the more challenging scenario when the task representation is *imbalanced*. Similarly to [2], we create imbalanced streams from SplitMNIST and SplitCIFAR-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 SplitMNIST we set the replay buffer size to $m = 100$ while for SplitCIFAR-10 we use $m = 200$. We evaluate the test accuracy on the tasks individually, where we do not undersample the test set. We train on the two imbalanced streams the CNN and the ResNet-18 described in Section 6.1, and set the number of slots to $s = 1$. We compare our method to reservoir sampling and class-balancing reservoir sampling (CBRS) [11]. The results in Table 3 confirm the flexibility of our framework which is competitive with methods specifically designed for imbalanced streams.

Runtime. In Table 4 we measure the runtime of selecting 100 and 400 coreset points from a batch of 1000 points with our proxy formulation, with the CNTK corresponding to the two convolutional networks used in the experiments. The CNTKs are calculated on a GeForce GTX 1080 Ti GPU while the coreset selection is performed on a single CPU. Due to the non-linear increase of computation time in terms of the coreset size, our proxy reformulation is most practical for smaller coreset sizes.

7 Conclusion

We presented a novel framework for coreset generation based on bilevel optimization with cardinality constraints. We theoretically established connections to experimental design and empirical influence functions. We showed that our method yields representative data summaries for neural networks and illustrated its advantages in alleviating catastrophic forgetting in continual learning and streaming deep learning, where our coreset construction performs among the best summarization strategies.

Broader Impact

Coresets can efficiently handle large datasets under computational constraints, thus significantly reducing computational costs and possibly the energy consumption for applications. They also provide an avenue towards limiting the amount of data that needs to be stored, with possible privacy benefits. Explicitly optimizing representativeness of the retained samples beyond accuracy (e.g., for counteracting biases in the data) is a promising direction for future work.

Acknowledgments and Disclosure of Funding

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

References

[1] R. Aljundi, K. Kelchtermans, and T. Tuytelaars. Task-free continual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11254–11263, 2019.

[2] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio. Gradient based sample selection for online continual learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 11816–11825. Curran Associates, Inc., 2019.

[3] S. Arora, S. S. Du, W. Hu, Z. Li, R. R. Salakhutdinov, and R. Wang. On exact computation with an infinitely wide neural net. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8139–8148. Curran Associates, Inc., 2019.

[4] D. ARTHUR. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, 2007*, pages 1027–1035, 2007.

[5] J. F. Bard. *Practical Bilevel Optimization: Algorithms and Applications*. Springer, 1998.

[6] A. A. Bian, J. M. Buhmann, A. Krause, and S. Tschiatschek. Guarantees for greedy maximization of non-submodular functions with applications. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 498–507, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[7] T. Campbell and T. Broderick. Automated scalable bayesian inference via hilbert coresets. *The Journal of Machine Learning Research*, 20(1):551–588, 2019.

[8] K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Statist. Sci.*, 10(3):273–304, 08 1995.

[9] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. Torr, and M. Ranzato. Continual learning with tiny episodic memories. *arXiv preprint arXiv:1902.10486*, 2019.

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

[11] A. Chrysakis and M.-F. Moens. Online continual learning from imbalanced data. *Proceedings of Machine Learning Research*, 2020.

[12] C. Coleman, C. Yeh, S. Mussmann, B. Mirzasoleiman, P. Bailis, P. Liang, J. Leskovec, and M. Zaharia. Selection via proxy: Efficient data selection for deep learning. In *International Conference on Learning Representations*, 2020.

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

[14] R. D. Cook and S. Weisberg. *Residuals and influence in regression*. New York: Chapman and Hall, 1982.

[15] A. Das and D. Kempe. Submodular meets spectral: greedy algorithms for subset selection, sparse approximation and dictionary selection. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 1057–1064, 2011.

[16] S. Fanello, C. Ciliberto, M. Santoro, L. Natale, G. Metta, L. Rosasco, and F. Odone. icub world: Friendly robots help building good vision data-sets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 700–705, 2013.

[17] S. Farquhar and Y. Gal. Towards robust evaluations of continual learning. *arXiv preprint arXiv:1805.09733*, 2018.

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

[19] D. Feldman and M. Langberg. A unified framework for approximating and clustering data. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 569–578. ACM, 2011.

[20] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[21] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil. Bilevel programming for hyperparameter optimization and meta-learning. *arXiv preprint arXiv:1806.04910*, 2018.

[22] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

[23] R. M. French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[24] S. Ghadimi and W. Mengdi. Approximation methods for bilevel programming. *arXiv:1802.02246*, 2018.

[25] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks, 2014.

[26] C. Harshaw, M. Feldman, J. Ward, and A. Karbasi. Submodular maximization beyond non-negativity: Guarantees, fast algorithms, and applications. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2634–2643, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[27] T. L. Hayes, N. D. Cahill, and C. Kanan. Memory efficient experience replay for streaming learning. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[28] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[29] J. Huggins, T. Campbell, and T. Broderick. Coresets for scalable bayesian logistic regression. In *Advances in Neural Information Processing Systems*, pages 4080–4088, 2016.

[30] A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

[31] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[32] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[33] A. Kirsch, J. van Amersfoort, and Y. Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 7024–7035. Curran Associates, Inc., 2019.

[34] P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1885–1894. JMLR. org, 2017.

[35] A. Krizhevsky et al. Learning multiple layers of features from tiny images, 2009.

[36] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[37] H. Liu, K. Simonyan, and Y. Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

[38] F. Locatello, M. Tschannen, G. Rätsch, and M. Jaggi. Greedy algorithms for cone constrained optimization with convergence guarantees. In *Advances in Neural Information Processing Systems*, pages 773–784, 2017.

[39] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6467–6476, 2017.

[40] J. Lorraine, P. Vicol, and D. Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*, pages 1540–1552. PMLR, 2020.

[41] M. Lucic, M. Faulkner, A. Krause, and D. Feldman. Training gaussian mixture models at scale via coresets. *The Journal of Machine Learning Research*, 18(1):5885–5909, 2017.

[42] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[43] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner. Variational continual learning. In *International Conference on Learning Representations*, 2018.

[44] R. Novak, L. Xiao, J. Hron, J. Lee, A. A. Alemi, J. Sohl-Dickstein, and S. S. Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. In *International Conference on Learning Representations*, 2020.

[45] F. Pedregosa. Hyperparameter optimization with approximate gradient. In *International Conference on Machine Learning*, pages 737–746, 2016.

[46] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[47] M. Ren, W. Zeng, B. Yang, and R. Urtasun. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning*, pages 4331–4340, 2018.

[48] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[49] O. Sener and S. Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018.

[50] H. Shin, J. K. Lee, J. Kim, and J. Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999, 2017.

[51] J. Tapia, E. Knoop, M. Mutnỳ, M. A. Otaduy, and M. Bächer. Makesense: Automated sensor design for proprioceptive soft robots. *Soft robotics*, 7(3):332–345, 2020.

[52] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[53] M. K. Titsias, J. Schwarz, A. G. de G. Matthews, R. Pascanu, and Y. W. Teh. Functional regularisation for continual learning with gaussian processes. In *International Conference on Learning Representations*, 2020.

[54] L. N. Vicente and P. H. Calamai. Bilevel and multilevel programming: A bibliography review. *Journal of Global optimization*, 5(3):291–306, 1994.

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

[56] K. Wei, R. Iyer, and J. Bilmes. Submodularity in data subset selection and active learning. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1954–1963, Lille, France, 07–09 Jul 2015. PMLR.

[57] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3987–3995, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Supplementary Material

A Connections 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 matrix where only rows of X whose indices are in $S \subset [n]$ are selected. This will be equivalent to selection done via 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,

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{n} w_i (x_i^\top \theta - y_i)^2 + \lambda \sigma^2 ||\theta||_2^2 \tag{7}$$

which has the following closed form,

$$\hat{\theta}_S = (X_S^\top X_S + \lambda \sigma^2 I)^{-1} X_S^\top y_S. \tag{8}$$

Frequentist Experimental Design. Under the assumption that the data follows a 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 (7) 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 2 (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* (7) *with* $\lambda = 0$*, the objective simplifies,*

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

Proof. Using the closed from in (8), and model assumptions, we see that $\hat{\theta} = \theta + (X_S^\top X_S)^{-1} X_S^\top \epsilon_S$. Plugging this in to the outer objective,

$$\begin{aligned}
g(\hat{\theta}) &= \frac{1}{2}\mathbb{E}_\epsilon \left[\left\| \theta - \hat{\theta} \right\|_2^2 \right] && (9)\\
&= \frac{1}{2}\mathbb{E}_\epsilon \left[\left\| (X_S^\top X_S)^{-1} X_S^\top \epsilon_S \right\|_2^2 \right] && (10)\\
&= \frac{1}{2}\mathbb{E}_\epsilon \left[\mathbf{Tr}\left(\epsilon_S^\top X_S (X_S^\top X_S)^{-2} X_S^\top \epsilon_S \right) \right] && (11)\\
&= \frac{\sigma^2}{2} \mathbf{Tr}\left((X_S^\top X_S)^{-1} \right) && (12)\\
&= \frac{\sigma^2}{2} \mathbf{Tr}\left((X^\top D(w) X)^{-1} \right) && (13)
\end{aligned}$$

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

Proposition 3 (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* (7) *with* $\lambda = 0$*, the objective simplifies,*

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

Proof. Using the closed from in (8), and model assumptions, we see that $\hat{\theta} = \theta + (X_S^\top X_S)^{-1} X_S^\top \epsilon_S$. Plugging this in to the outer objective $g(\hat{\theta})$,

$$G(w) = \frac{1}{2n}\mathbb{E}_\epsilon\left[\left\|X\theta - X\hat{\theta}\right\|_2^2\right] \tag{14}$$

$$= \frac{1}{2n}\mathbb{E}_\epsilon\left[\left\|X(X_S^\top X_S)^{-1} X_S^\top \epsilon_S\right\|_2^2\right] \tag{15}$$

$$= \frac{1}{2n}\mathbb{E}_\epsilon\left[\mathbf{Tr}\left(\epsilon_S^\top X_S (X_S^\top X_S)^{-1} X^\top X (X_S^\top X_S)^{-1} X_S^\top \epsilon_S\right)\right] \tag{16}$$

$$= \frac{\sigma^2}{2n}\mathbf{Tr}\left(X(X_S^\top X_S)^{-1} X^\top\right) \tag{17}$$

$$= \frac{\sigma^2}{2n}\mathbf{Tr}\left(X(X^\top D(w) X)^{-1} X^\top\right) \tag{18}$$

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 infinite data limit $n \to \infty$.

Proposition 4 (Infinite data limit)**.** *Under the linear regression assumptions, 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,

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

Let Θ be the set containing random variables $\hat{\theta}$ representing the inner problem's solution on a finite subset of the data. Thus each $\hat{\theta}$ depends on a finite set of $\epsilon_S := \cup_{i\in S}\{e_i\}$, where $S \subset [n]$. Then

$$\lim_{n\to\infty} g(\hat{\theta}) - g_V(\hat{\theta}) - \frac{\sigma^2}{2} = 0, \quad \forall \hat{\theta} \in \Theta.$$

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

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

Under the infinite data limit as $n \to \infty$, we have $\lim_{n\to\infty}\frac{1}{n}\mathbb{E}_\epsilon\left[\sum_{i\in S}\epsilon_i(x_i^\top\hat{\theta} - x_i^\top\theta)\right] = 0$ since S is a finite set. Since $\hat{\theta}$ only depends on ϵ_S, the independence of $\hat{\theta}$ and ϵ_i, $i \in [n]\setminus S$ can be established. Thus,

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

As a consequence, as $\lim_{n\to\infty} g(\hat{\theta}) - g_V(\hat{\theta}) - \frac{\sigma^2}{2} = 0$ for all $\hat{\theta} \in \Theta$.

□

Note that the Proposition 4 does not imply that our algorithm performs the same steps when used with g_V instead of g. It only means that the optimal solutions of the problems are converging to selections with the same quality in the infinite data limit.

Bayesian V-Experimental Design. Bayesian experimental designs [8] can be incorporated as well into our framework. In Bayesian modelling, 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 an appropriate inner and outer objectives.

Proposition 5. *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. Further, let the inner objective be equal to (7) *with the same value of λ, then the overall objective simplifies to*

$$G(w) = \frac{1}{2n}\mathbf{Tr}\left(X\left(\frac{1}{\sigma^2}X^\top D(w)X + \lambda I\right)^{-1}X^\top\right). \tag{19}$$

Proof. Using the closed from in (8), and model assumptions, we see that $\hat{\theta} = (X_S^\top X_S + \lambda\sigma^2 I)^{-1}X_S^\top(X_S\theta + \epsilon_S)$. Plugging this in to the outer objective $g_V(\hat{\theta})$,

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

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

□

Similarly to the case of 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 over it using the prior.

Proposition 6 (Infinite data limit)**.** *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,

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

Under the same assumptions as in Proposition 4 and $\theta \sim \mathcal{N}(0, \lambda^{-1} I)$

$$\lim_{n\to\infty} g(\hat{\theta}) - g_V(\hat{\theta}) - \frac{\sigma^2}{2} = 0, \quad \forall \hat{\theta} \in \Theta$$

.

Proof. The proof follows similarly as in Proposition 4. □

Weak-submodularity of Bayesian V-experimental design. The greedy algorithm is known to perform well on A-experimental design due to the function $G(w)$ being weakly-submodular and monotone [6]. Cardinality-constrained maximization of such problems with greedy algorithm is known to find a solution which $1 - e^{-\gamma}$ [15] approximation to the optimal subset. The parameter γ is known as weak submodularity ratio.

Following the analysis of [26], which derives the weak submodularity ratio of A-experimental design, we show that V-experimental design surprisingly has the same submodularity ratio as well, and hence we expect greedy strategy to perform well. We follow slightly different nomenclature than [26].

Proposition 7. *The function* $R(w) = G(0) - G(w)$ *as in* (19) *is non-negative, monotone and* γ*-weakly submodular function with*

$$\gamma = \left(1 + s^2 \frac{1}{\sigma^2 \lambda}\right)^{-1}$$

where $s = \max_{i\in\mathcal{D}} \|x_i\|_2$.

Proof. We employ exactly the same proof technique as [26] relying on Sherman-Morison identity.

Let A and B be disjoint sets without loss of generality. Also, let $M_A := I\lambda + \frac{1}{\sigma^2} X_A X_A^\top$, which positive definite by definition. Following the line of proof in [26], it can be shown that a marginal gain of an element $e \in B$ is equal to

$$R(e|A) = \frac{x_e^\top M_A^{-1} X X^\top M_A^{-1} x_e}{\sigma^2 + x_e^\top M_A^{-1} x_e} \tag{20}$$

In order to derive the weak-submodularity ratio we need to lower bound,

$$\frac{\sum_{e\in B} R(e|A)}{R(B \cup A) - R(A)}. \tag{21}$$

Note the observation made by [26] that $\sigma^2 + x_e^\top M_A^{-1} x_e \leq \sigma^2 + s^2 \lambda^{-1}$ in their Equation 13.

$$\begin{aligned}
\sum_{e\in B} R(e|A) &= \sum_{e\in B} \frac{x_e^\top M_A^{-1} X X^\top M_A^{-1} x_e}{\sigma^2 + x_e^\top M_A^{-1} x_e} && (22)\\
&= \sum_{e\in B} \frac{\mathbf{Tr}(x_e^\top M_A^{-1} X X^\top M_A^{-1} x_e)}{\sigma^2 + x_e^\top M_A^{-1} x_e} && (23)\\
&\overset{\text{as above}}{\geq} \sum_{e\in B} \frac{\mathbf{Tr}(x_e^\top M_A^{-1} X X^\top M_A^{-1} x_e)}{\sigma^2 + s^2 \lambda^{-1}} && (24)\\
&= \frac{\mathbf{Tr}(X_B^\top M_A^{-1} X X^\top M_A^{-1} X_B)}{\sigma^2 + s^2 \lambda^{-1}} && (25)
\end{aligned}$$

Now the denominator which is a more general version of (20)

$$\begin{aligned}
R(B \cup A) - R(A) &= \mathbf{Tr}\left((\sigma^2 I + X_B^\top M_A^{-1} X_B)^{-1} X_B^\top M_A^{-1} X X^\top M_A^{-1} X_B\right) && (26)\\
&\leq \frac{1}{\sigma^2} \mathbf{Tr}(X_B^\top M_A^{-1} X X^\top M_A^{-1} X_B) && (27)
\end{aligned}$$

where we used the fact that the $(\sigma^2 I + X_B^\top M_A^{-1} X_B) \succeq \sigma^2 I$. Note that the result follows by plugging (27) and (25) to the expression (21) and observing that the fraction can be simplified. The simplification finishes the proof. □

Lemma 8. *Assume $\|x_i\|_2 < L < \infty$ for all $i \in [n]$ and $w \in \mathbb{R}^n_+$ s.t. $\|w\|_2 < \infty$. The function*

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

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 imply the convexity and smoothness of $G(w)$.

For the sake of brevity, we will work with the function $\hat{G}(w) = \mathbf{Tr}\left(X\left(X^\top D(w)X + \lambda\sigma^2 I\right)^{-1}X^\top\right)$ where $\frac{\sigma^2}{2n}\hat{G}(w) = G(w)$. Also, let us 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 make use of directional derivatives:

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

In order 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:

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

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

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

Now, since,

$$F(w+hv)F^+(w) = (F(w)+hX^\top D(v)X)F^+(w) = I + hX^\top D(v)XF^+(w)$$
$$F^+(w+hv)F(w) = F^+(w+hv)(F(w+hv)-hX^\top D(v)X) = I - hF^+(w+hv)X^\top D(v)X$$

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:

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

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:

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

Thus G is $\frac{nL^6}{\lambda^3\sigma^4}$-smooth.

□

B Connections to Influence Functions.

We show that our approach is also related to incremental subset selection via influence functions. Let us consider the influence of the k-th point on the outer objective. Suppose that we have already selected the subset S and found the corresponding optimal weights w_S^*. Then, the influence of point k on the outer objective is

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

Proposition 9. *Under twice differentiability and strict convexity of the inner loss,* $\operatorname{argmax}_k \mathcal{I}(k)$ *corresponds to the selection rule in Equation* (5)*.*

Proof. Following [34] and using the result of [14], the empirical influence function at k is

$$\left.\frac{\partial \theta^*}{\partial \varepsilon}\right|_{\varepsilon=0} = -\left(\frac{\partial^2 \sum_{i=1}^n w^*_{S,i} \ell_i(\theta^*)}{\partial \theta \partial \theta^\top}\right)^{-1} \nabla_\theta \ell_k(\theta^*). \tag{28}$$

Now, we use the chain rule for $\mathcal{I}(k)$:

$$\begin{aligned}
\mathcal{I}(k) &= -\left.\frac{\partial \sum_{i=1}^n \ell_i(\theta^*)}{\partial \varepsilon}\right|_{\varepsilon=0} \\
&= -\left(\nabla_\theta \sum_{i=1}^n \ell_i(\theta^*)\right)^\top \left.\frac{\partial \theta^*}{\partial \varepsilon}\right|_{\varepsilon=0} \\
&\overset{\text{Eq. 28}}{=} \nabla_\theta \ell_k(\theta^*)^\top \left(\frac{\partial^2 \sum_{i=1}^n w^*_{S,i} \ell_i(\theta^*)}{\partial \theta \partial \theta^\top}\right)^{-1} \nabla_\theta \sum_{i=1}^n \ell_i(\theta^*).
\end{aligned}$$

□

C Merge-reduce Streaming Coreset Construction

Algorithm 2 streaming_coreset

Input: stream S, number of slots s, β
buffer = [ ]
for $\mathcal{X}_t$ in stream S **do**
  $\mathcal{C}_t$ = construct_coreset($\mathcal{X}_t$)
  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

Algorithm 3 select_index

Input: buffer = $[(C_1, \beta_1), \ldots, (C_{s+1}, \beta_{s+1})]$ containing an extra slot
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

D Continual Learning and Streaming Experiments

We conduct an extensive study of several selection methods for the replay memory in the continual learning setup. We compare the following methods:

- Training w/o replay: train after each task without replay memory. Demonstrates how catastrophic forgetting occurs.
- Uniform sampling / per task coreset: the network is only trained on the points in the replay memory with the different selection methods.

Table 5: 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 among the best.

Method	PermMNIST	SplitMNIST	SplitCIFAR-10
Training w/o replay	73.82 ± 0.49	19.90 ± 0.03	19.95 ± 0.02
Uniform sampling, train at the end	34.31 ± 1.37	86.09 ± 1.84	28.31 ± 0.94
Per task coreset, train at the end	46.22 ± 0.45	92.53 ± 0.53	31.01 ± 1.16
Uniform sampling	78.46 ± 0.40	92.80 ± 0.79	36.20 ± 3.19
k-means of features	78.34 ± 0.49	93.40 ± 0.56	33.41 ± 2.48
k-means of embeddings	78.84 ± 0.82	93.96 ± 0.48	36.91 ± 2.42
k-means of grads	76.71 ± 0.68	87.26 ± 4.08	27.92 ± 3.66
k-center of features	77.32 ± 0.47	93.16 ± 0.96	32.77 ± 1.62
k-center of embeddings	78.57 ± 0.58	93.84 ± 0.78	36.91 ± 2.42
k-center of grads	77.57 ± 1.12	88.76 ± 1.36	27.01 ± 0.96
Gradient matching	78.00 ± 0.57	92.36 ± 1.17	35.69 ± 2.61
Max entropy samples	77.13 ± 0.63	91.30 ± 2.77	27.00 ± 1.83
Hardest samples	76.79 ± 0.55	89.62 ± 1.23	28.10 ± 1.79
FRCL's selection	78.01 ± 0.44	91.96 ± 1.75	34.50 ± 1.29
iCaRL's selection	79.68 ± 0.41	93.99 ± 0.39	34.52 ± 1.62
Coreset	79.26 ± 0.43	95.87 ± 0.20	37.60 ± 2.41

- Training per task with uniform sampling / coreset: the network is trained after each task and regularized with the loss on the samples in the replay memory chosen as a uniform sample / coreset per task.
- k-means / k-center in feature / embedding / gradient space: the per-task selection retains points in the replay memory that are generated by the k-means++ [4] / 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 the different spaces are the ones chosen to be saved in the memory. Note that the k-center summarization in the last layer embedding space is the coreset method proposed for active learning by [49].
- 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 [12] and [1].
- Training per task with FRCL's / iCaRL's selection: the points per task are selected by FRCL's inducing point selection [53], where the kernel is chosen as the linear kernel over the last layer embeddings / iCaRL's selection (Algorithm 4 in [46]) performed in the normalized embedding space.
- Gradient matching per task: same as iCaRL's selection, but in the space of the gradient with respect to the last layer and jointly over all classes. This is a variant of Hilbert coreset [7] 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.

We report the results in Table 5. We note that while several methods outperform uniform sampling on some datasets, only our method is consistently outperforming it on all datasets. We also conduct a study on the effect of the replay memory size shown in Table 6.

Table 6: Replay memory size study on SplitMNIST. Our method offers bigger improvements with smaller memory sizes.

Method / Memory size	50	100	200
CL uniform sampling	85.23 ± 1.84	92.80 ± 0.79	95.08 ± 0.30
CL coreset	91.79 ± 0.71	95.87 ± 0.20	97.06 ± 0.30
Streaming reservoir sampling	83.90 ± 3.18	90.72 ± 0.97	94.12 ± 0.61
Streaming coreset	85.68 ± 2.05	92.59 ± 1.20	95.64 ± 0.68

RBF kernel as proxy. In our experiments we used the Neural Tangent Kernel as proxy. It turns out that on the datasets 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 7. 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 datasets, it fails on harder datasets such as CIFAR-10.

Table 7: RBF vs CNTK proxies.

	Method	PermMNIST	SplitMNIST
CL	Coreset CNTK	79.26 ± 0.43	95.87 ± 0.20
	Coreset RBF	79.89 ± 0.87	96.18 ± 0.33
VCL	Coreset CNTK	86.18 ± 0.21	84.66 ± 0.63
	Coreset RBF	86.13 ± 0.31	82.37 ± 1.40
Str.	Coreset CNTK	74.44 ± 0.52	92.59 ± 1.20
	Coreset RBF	75.93 ± 0.59	92.61 ± 0.83

Inspecting the coreset. We provide further insights for the large gains obtained by our method by inspecting the resulting coreset when summarizing MNIST in Figure 5. We can notice that in each step, the method selects the sample that has the potential to increase the accuracy by the largest amount: the first 10 samples (first row) are picked from different classes, after which the method diversifies within the classes. Even though the dataset is balanced, our method chooses more samples from more diverse classes, e.g., it picks twice as many 8s than 1s.

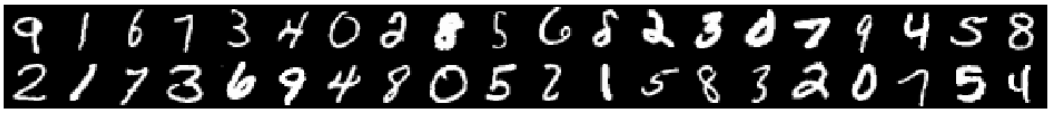


Figure 5: Coreset of size 40 of MNIST produced by our method: samples are diverse within class, and harder classes are represented with more samples.

E Speedups, Data Preprocessing, Architectures and Hyperparameters

Speedups. We now discuss several speedups. Since the number of parameters in the inner optimization problem in Eq. 6 is small, we can use quasi-Newton methods such as L-BFGS [36] for faster convergence. For the outer level optimization we use Adam [31]. After a step on the outer level, we can reuse the result from the last inner iteration to restart the optimization of the inner variable. For additional speedup in the streaming and continual learning setting, we use binary weights and consequently do not perform weight optimization.

Bilevel optimization. When the inner optimization problem cannot be solved in closed form, we solve it using L-BFGS using step size 0.25, 200 iterations and 10^{-5} tolerance on first order optimality. For the inner optimization problem we report the best result obtained for $\lambda \in \{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$. The number of conjugate gradient steps per approximate inverse Hessian-vector product calculation is set to 50.

For the outer level we use Adam, with learning rate 0.05 and 10 iterations for the CNN on MNIST experiment, and with learning rate 0.02 and 200 iterations for the KRR with CNTK on CIFAR-10 experiment — here we can afford more outer iterations as the inner problem can be solved in closed form. When applying the gradient on weights, we ensure the positivity of weights by projection. In continual learning and streaming we use unweighted samples, so we do not perform outer iterations. When applying our incremental selection of points, we select the best candidate out of a random sample of 200.

Data preprocessing. For all datasets, we standardize the pixel values per channel for each dataset separately. For the CIFAR-10 streaming experiment we perform the standard data augmentations of random cropping and horizontal flipping.

Training details. 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 [2], we use a streaming batch size of 10 for better performance, as indicated in Section 2.4 of the supplementary materials of [2].

Hyperparameter selection in continual learning and streaming. The replay memory regularization strength β was tuned separately for each method from the set $\{0.01, 0.1, 1, 10, 100, 1000\}$ and the best result on the test set was reported.

Computational resources. The neural networks are trained on a single GeForce GTX 1080 Ti. (C)NTK kernels are calculated on the same GPU. The coreset generation is performed with a single CPU thread.