Bilevel Coreset Selection in Continual Learning: A New Formulation and Algorithm

Jie Hao
Department of Computer Science
George Mason University
jha06@gmu.edu

Kaiyi Ji
Department of CSE
University at Buffalo
kaiyiji@buffalo.edu

Mingrui Liu∗
Department of Computer Science
George Mason University
mingruil@gmu.edu

Abstract

Coreset is a small set that provides a data summary for a large dataset, such that training solely on the small set achieves competitive performance compared with a large dataset. In rehearsal-based continual learning, the coreset is typically used in the memory replay buffer to stand for representative samples in previous tasks, and the coreset selection procedure is typically formulated as a bilevel problem. However, the typical bilevel formulation for coreset selection explicitly performs optimization over discrete decision variables with greedy search, which is computationally expensive. Several works consider other formulations to address this issue, but they ignore the nested nature of bilevel optimization problems and may not solve the bilevel coreset selection problem accurately. To address these issues, we propose a new bilevel formulation, where the inner problem tries to find a model which minimizes the expected training error sampled from a given probability distribution, and the outer problem aims to learn the probability distribution with approximately $K$ (coreset size) nonzero entries such that learned model in the inner problem minimizes the training error over the whole data. To ensure the learned probability has approximately $K$ nonzero entries, we introduce a novel regularizer based on the smoothed top-$K$ loss in the upper problem. We design a new optimization algorithm that provably converges to the $\epsilon$-stationary point with $O(1/\epsilon^4)$ computational complexity. We conduct extensive experiments in various settings in continual learning, including balanced data, imbalanced data, and label noise, to show that our proposed formulation and new algorithm significantly outperform competitive baselines. From bilevel optimization point of view, our algorithm significantly improves the vanilla greedy coreset selection method in terms of running time on continual learning benchmark datasets. The code is available at https://github.com/MingruiLiu-ML-Lab/Bilevel-Coreset-Selection-via-Regularization.

1 Introduction

Deep Neural Networks (DNNs) have achieved tremendous successes in various domains, including computer vision [41, 30], natural language processing [72, 18], generative modeling [26] and

∗Corresponding Author.

Figure 1: Illustration of our algorithm. There are two neural network models, one is $M_{tr}$ for model training, and the other is $M_{cs}$ for coreset selection. A coreset is selected from the current data stream $B_t$ by conducting six steps. ① Feed a stream mini-batch $B_t$ and sampled buffer data to $M_{tr}$. ② Copy model parameters from $M_{tr}$ to $M_{cs}$: $\theta_{cs} \leftarrow \theta_{tr}$. ③ Feed a mini-batch $B_t$ into model $M_{cs}$. ④ Conduct bilevel optimization to update the model parameter $\theta_{cs}$ in $M_{cs}$ and output a probability distribution $w$. ⑤ Sample a coreset from $B_t$ based on the distribution of $w$ and add the sampled data into buffer. ⑥ Calculate stochastic gradient based on $B_t$ and sampled data in the buffer in Step 1, and update $\theta_{tr}$ based on gradient information. Repeat the above steps for each stream mini-batch.

games [68]. However, in continual learning, where DNNs are trained on a sequence of tasks with possibly non-i.i.d. data, the performance will be degraded on the previously trained tasks. This is referred to as catastrophic forgetting [53, 52, 60]. To alleviate catastrophic forgetting, one of the effective ways is rehearsal-based continual learning, where a small replay buffer is maintained and revisited during the continuous learning process. There is a line of works studying how to efficiently maintain the replay buffer using the coreset selection approach [6, 78, 83], in which a small set of data is selected as representative samples to be used in continual learning.

The coreset selection in continual learning is formulated as a cardinality-constrained bilevel optimization problem which is solved by incremental subset selection [6]. This greedy approach is computationally expensive and hence is not scalable when the coreset size is large. To address this issue, Zhou et al. [83] propose a relaxation of the bilevel formulation in [6], which drops the nested nature of bilevel formulation and actually becomes two sequential optimization problems. Tiwari et al. proposed a gradient approximation method in [71], which selects a coreset that approximates the gradient of model parameters over the entirety of the data seen so far. Yoon et al. [78] proposes an online coreset selection method by maximizing several similarity metrics based on data pairs within each minibatch, and sample pairs between each minibatch and coreset. These approaches do not directly address the algorithmic challenges caused by the nested nature of the bilevel optimization problem, and may not solve the original bilevel coreset selection problem efficiently.

The key challenges in the bilevel coreset selection problems are two folds. First, the bilevel formulation in [6] needs to directly perform optimization over cardinality constraint, which is a nonconvex set and greedy approaches are expensive when the coreset size is large. Second, the bilevel formulation in [6] has a nested structure: one problem is embedded within another, and the outer and inner functions both have dependencies over the same set of decision variables. It remains unclear how to design efficient algorithms to solve constrained bilevel optimization algorithms for the coreset selection problem with provable theoretical guarantees.

Our proposed solution addresses these challenges with a novel bilevel formulation and provably efficient optimization algorithms. The proposed new bilevel formulation is referred to as Bilevel Coreset Selection via Regularization (BCSR). The main differences between our approach and the standard bilevel approach in [6] are: (i) unlike the standard bilevel formulation which requires performing optimization based on a cardinality constraint, we propose to solve a bilevel optimization on a probability simplex over training examples; (ii) to make sure the probability distribution lies in a low dimensional manifold, we propose to add a smoothed top-$K$ loss as a regularizer to the upper-level problem; (iii) due to our new formulation, we are able to design a simple and effective first-order method to solve this new bilevel problem with provable non-asymptotic convergence guarantees. The first-order method is easy to implement and much faster than the greedy approach as in [6]. Our main contribution is listed as follows.
We propose a new bilevel formulation, namely BCSR, for the coreset selection in rehearsal-based continual learning. Instead of directly learning the binary masks for each sample, the new formulation tries to learn a probability distribution in a low-dimensional manifold by adding a smoothed top-K loss as a regularizer in the upper problem. This formulation is designed to satisfy two important features in continual learning with DNNs: (i) keeping the nested structure in the coreset selection; (ii) being amenable to first-order algorithms, which makes it easy to implement in modern deep learning frameworks such as PyTorch and TensorFlow. Based on the new formulation, we propose an efficient first-order algorithm for solving it. The main workflow of our algorithm is illustrated in Figure 1 and the corresponding Pytorch-style pseudocode is presented in Algorithm 1.

We have conducted extensive experiments among various scenarios to verify the effectiveness of our proposed algorithm, including balanced, imbalanced, and label-noise data. Our algorithm outperforms all baselines for all settings in terms of average accuracy, and it is much better than all other coreset selection algorithms. For example, on imbalanced data of Multiple Datasets, BCSR is better than the best coreset selection algorithm by 4.65% in average accuracy. From bilevel optimization point of view, our algorithm significantly improves the vanilla greedy coreset selection method [6] in terms of running time on continual learning benchmark datasets.

Under the standard smoothness assumptions of the loss function, we show that our algorithm requires at most $O(1/\epsilon^4)$ complexity for finding an $\epsilon$-stationary point in the constrained case. Notably, the $O(1/\epsilon^4)$ complexity consists of $O(1/\epsilon^2)$ backpropagations and $O(1/\epsilon^4)$ samplings from Gaussian distribution, where the latter cost is computationally cheap.

2 Related Work

Continual Learning There are different classes of continual learning methods, including regularization-based approaches [39, 81, 10, 1, 59, 64, 17], dynamic architecture methods [61].
Coreset Selection The coreset selection methods were used frequently in supervised and unsupervised learning, such as $k$-means [19], Gaussian mixture model [49], logistic regression [33] and Bayesian inference [9]. They were also used frequently in the active learning literature [74][63]. The coreset selection in continual learning is related to the sample selection [34][2][3]. Nguyen et al. [55] introduce variational continual learning which is combined with coreset summarization [5]. Borsos et al. [6] proposed the first bilevel formulation for the coreset selection in continual learning, which is later improved by [83][78]. Compared with these works, our work focuses on improved bilevel coreset selection: we provide a better bilevel formulation than [6] and design a provably efficient optimization algorithm.

Bilevel optimization Bilevel optimization is used to model nested structure in the decision-making process [73]. Recently, gradient-based bilevel optimization methods have broad applications in machine learning, including meta-learning [20], hyperparameter optimization [56][22], neural architecture search [49], and reinforcement learning [40][32]. These methods can be generally categorized into implicit differentiation [16][50][45][4] and iterative differentiation [50][21][20][65][27] based approaches. Recently, various stochastic bilevel algorithms have been also proposed and analyzed by [12][77][25][32][29][4][14]. A comprehensive introduction can be found in the survey [47]. In this work, we propose a novel stochastic bilevel optimizer with very flexible parameter selection, which shows great promise in the coreset selection for continual learning.

3 New Bilevel Formulation for Coreset Selection in Continual Learning

In this section, we first introduce our new bilevel formulation, namely Bilevel Coreset Selection via Regularization (BCSR). The key idea of this approach is to learn a probability distribution over the whole dataset such that the best model parameter obtained by minimizing the loss on the sampled dataset (i.e., the minimizer for the lower-level problem) is also the best for the whole dataset (i.e., the minimizer for the upper-level problem), and then a coreset can be sampled based on the learned probability distribution. In addition, the learned probability distribution is expected to lie in a low dimensional manifold (i.e., with $K$ nonzero entries where $K$ is the coreset size). To achieve this, we added a smooth top-$K$ loss as a regularizer to promote the probability distribution to have $K$ nonzero entries. Specifically, the objective function of BCSR is:

$$\min_{\theta, w} \left[ \phi(w) = \sum_{i=1}^{n} \ell_i(\theta(w)) - \lambda \sum_{i=1}^{K} E_{\delta z}(w + \delta z)_{[i]} \right]$$

s.t., $\theta^*(w) = \arg\min_\theta \left[ L(\theta, w) = \sum_{i=1}^{n} w(i) \ell_i(\theta) \right]$  \hspace{1cm} (1)

where $n$ is the sample size, $\theta$ is the model parameter, $w$ is the sample weights, $\ell_i(\theta)$ denote the loss function calculated based on $i$-th sample with model parameter $\theta$, $w(i)$ is the $i$-th coordinate of $w$, $w_{[i]}$ is the $i$-th largest component of $w$, $\lambda > 0$ is the regularization parameter, $w + \delta z$ denote to adding $\delta z$ on each coordinate of $w$ where $z \sim \mathcal{N}(0, 1)$. Note that $R(w, \delta) := -\lambda \sum_{i=1}^{K} E_{\delta z}(w + \delta z)_{[i]}$ denote the smoothed top-$K$ regularization. We add this regularization to make sure the summation of the top-$K$ entries of the learned probability vector is large, such that we can confidently choose a coreset with size $K$. The goal of employing Gaussian noise to the regularizer is for the ease of algorithm design: this Gaussian smoothing technique can make the regularizer to be smooth such that it is easier to design efficient first-order bilevel optimization solvers. Otherwise, the upper-level problem would become nonconvex and nonsmooth, and it would be difficult for algorithm design under this case.

Discussion and Comparison with Prior Works In this part, we illustrate how this new formulation addresses the drawbacks of the previous approaches. The work of [6] does not use any regularizer and regards the weight of each sample as a binary mask. This formulation needs to solve a combinatorial
where $\theta$ within this minibatch by invoking Algorithm 3. Algorithm 3 is a first-order algorithm for solving Algorithm 3 as a subroutine. Each time the algorithm encounters a minibatch $B$, a coreset is selected within this minibatch by invoking Algorithm 3. Algorithm 3 is a first-order algorithm for solving the bilevel formulation (1). In Algorithm 3, the model parameter $\theta$ and the weight distribution $w$ are updated alternatively. We first perform $N$ steps of gradient descent steps to find a sufficiently good $\theta$ for the lower-level problem (lines 5-7) by the update rule:

$$\theta^k_j = \theta^{k-1}_j - \alpha \nabla_{\theta} L(\theta^{k-1}_j, w_j),$$

where $\theta^k_j$ denotes the model parameters at the $j$-th outer loop and the $k$-th inner loop. To update the outer variable $w$, BCSR approximates the true gradient $\nabla \phi(w)$ of the outer function $\phi$ w.r.t $w$, which is called hypergradient \[50\]. BCSR constructs a hypergradient estimator:

$$\varphi_{\delta} = \frac{1}{|B|} \sum_{\bar{z} \in \tilde{B}} \nabla_w R(w, \delta; \bar{z}) - \nabla \phi \nabla_{\theta} L(\theta^N_j, w_j) \left[ (\nabla_{\theta}^2 L(\theta^N_j, w_j))^{-1} \left( \sum_{i=1}^n \nabla_{\theta} \ell_i(\theta^N_j) \right) \right],$$

\[50\]The coreset weight $w$ in our formulation is equivalent to sample mask $s$ in \[83\]

4 \hspace{1em} Algorithm Design

Equipped with the new formulation, the entire algorithm is presented in Algorithm 2, which calls Algorithm 3 as a subroutine. Each time the algorithm encounters a minibatch $B$, a coreset is selected within this minibatch by invoking Algorithm 3. Algorithm 3 is a first-order algorithm for solving the bilevel formulation (1). In Algorithm 3, the model parameter $\theta$ and the weight distribution $w$ are updated alternatively. We first perform $N$ steps of gradient descent steps to find a sufficiently good $\theta$ for the lower-level problem (lines 5-7) by the update rule:

$$\theta^k_j = \theta^{k-1}_j - \alpha \nabla_{\theta} L(\theta^{k-1}_j, w_j),$$

where $\theta^k_j$ denotes the model parameters at the $j$-th outer loop and the $k$-th inner loop. To update the outer variable $w$, BCSR approximates the true gradient $\nabla \phi(w)$ of the outer function $\phi$ w.r.t $w$, which is called hypergradient [50]. BCSR constructs a hypergradient estimator:

$$\varphi_{\delta} = \frac{1}{|B|} \sum_{\bar{z} \in \tilde{B}} \nabla_w R(w, \delta; \bar{z}) - \nabla \phi \nabla_{\theta} L(\theta^N_j, w_j) \left[ (\nabla_{\theta}^2 L(\theta^N_j, w_j))^{-1} \left( \sum_{i=1}^n \nabla_{\theta} \ell_i(\theta^N_j) \right) \right],$$

\[50\]The coreset weight $w$ in our formulation is equivalent to sample mask $s$ in [83]

5
where $R(w, \delta; \bar{z}) := -\lambda \sum_{i=1}^{K} (w + \delta \bar{z})_{[i]}$ and $\bar{z} \sim N(0, 1)$. Solving the Hessian-inverse-vector product in eq. (3) is computationally intractable. We denote $v^* := (\nabla^2_{\theta} L(\theta^N_j, w_j))^{-1}(\sum_{i=1}^{n} \nabla_\theta \ell_i(\theta^N_j))$ in eq. (3), where $v^*$ can be approximated by solving the following quadratic programming problem efficiently by $Q$ steps of gradient descent (line 9):

$$
\min_v \frac{1}{2} v^T \nabla^2_{\theta} L(\theta^N_j, w_j) v - v^T \sum_{i=1}^{n} \nabla_\theta \ell_i(\theta^N_j).
$$

Next, the hypergradient estimate (line 10) is computed based on the output of approximated quadratic programming. Note both model parameters and sample weights need to use warm start initialization (line 4 and line 8). Then the weight is updated and projected onto simplex (line 11):

$$
\hat{w}_{j+1} = w_j - \beta \phi_j, \quad w_{j+1} = \mathcal{P}_{\Delta^n}(\hat{w}_{j+1}),
$$

where $\Delta^n := \{ w \in \mathbb{R}^n : 0 \leq w_{[i]} \leq 1, ||w||_1 = 1 \}$. In terms of other experimental hyperparameters, we allow very flexible choices of hyperparameters (e.g., $N$, $Q$) as shown in our theory to achieve polynomial time complexity for finding a $c$-stationary point.

The selected coresets for each task are stored in a memory buffer with a fixed size $m$. There is a separated memory slot for each task with size $[m/i]$ when the task $i$ comes. After each task $i$, the memory slots before $i$ will randomly remove some samples to adjust all the memory slots to $[m/i]$. That means the memory size for each task decreases as the task ID increase to maintain the total buffer size $m$ unchanged. The same memory strategy is also used in the greedy coreset approach [6].

## 5 Experiments

We conduct extensive experiments under various settings, including balanced data, imbalanced data, and label-noise data. The empirical results demonstrate the effectiveness of our method in rehearsal-based continual learning.

### 5.1 Experimental Setup

**Datasets** We use commonly-used datasets in the field of continual learning, including Split CIFAR-100, Permuted MNIST, Multiple Datasets, Tiny-ImageNet, and Split Food-101. We follow the experimental settings as that in prior work [78] and [83]. Each dataset is processed with three approaches: balanced, imbalanced, and label-noise. Please refer to Appendix L for more details about data processing and settings.

**Baselines** We compare our algorithm BCSR with other continual learning methods based on coreset strategy, including $k$-means features [55], $k$-means embedding [63], Uniform Sampling, iCaRL [57], Grad Matching [9], Greedy Coreset [6], MetaSP [70], GCR [71], and OCS [78]. We also compare with non-coreset reply method, SPR [38], MetaSP [70]. All algorithms are built upon episodic memory, which stores coreset selected from stream data. Then a model, such as ResNet-18 [30], is trained over the data from the current stream and the episodic memory.

**Metrics** Average accuracy and forgetting measure [10] are two primary evaluation metrics that are used in continual learning literature. AVG ACC ($A_T$) is the average accuracy tested on all tasks after finishing the task $T$: $A_T = \frac{1}{T} \sum_{i=1}^{T} \alpha_{T,i}$, where $\alpha_{T,i}$ is the test accuracy of task $i$ after training.

### Table 1: Experiment results on Split CIFAR-100

<table>
<thead>
<tr>
<th>Methods</th>
<th>Balanced</th>
<th>Imbalanced</th>
<th>Label Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$A_T$</td>
<td>$F_{GT_T}$</td>
<td>$A_T$</td>
</tr>
<tr>
<td>K-means Features</td>
<td>57.82±0.69</td>
<td>45.44±0.76</td>
<td>57.38±0.26</td>
</tr>
<tr>
<td>K-means Embedding</td>
<td>59.79±0.24</td>
<td>43.91±0.15</td>
<td>57.92±0.25</td>
</tr>
<tr>
<td>Uniform</td>
<td>58.99±0.54</td>
<td>44.73±0.11</td>
<td>58.76±0.10</td>
</tr>
<tr>
<td>iCaRL</td>
<td>60.74±0.09</td>
<td>44.25±0.04</td>
<td>59.70±0.70</td>
</tr>
<tr>
<td>Grad Matching</td>
<td>59.17±0.38</td>
<td>45.44±0.64</td>
<td>59.58±0.28</td>
</tr>
<tr>
<td>SPR</td>
<td>59.56±0.73</td>
<td>44.45±0.55</td>
<td>58.74±0.63</td>
</tr>
<tr>
<td>MetaSP</td>
<td>60.34±0.25</td>
<td>43.74±0.36</td>
<td>57.43±0.54</td>
</tr>
<tr>
<td>Greedy Coreset</td>
<td>59.39±0.16</td>
<td>43.80±0.01</td>
<td>58.22±0.16</td>
</tr>
<tr>
<td>GCR</td>
<td>58.73±0.43</td>
<td>44.8±0.05</td>
<td>58.72±0.63</td>
</tr>
<tr>
<td>PICS</td>
<td>55.64±0.26</td>
<td>39.8±1.12</td>
<td>56.93±0.14</td>
</tr>
<tr>
<td>OCS</td>
<td>52.57±0.37</td>
<td>46.54±0.34</td>
<td>51.77±0.81</td>
</tr>
<tr>
<td>BCSR</td>
<td>61.6±0.14</td>
<td>47.30±0.57</td>
<td>60.7±0.08</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>Balanced</th>
<th>Imbalanced</th>
<th>Label Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$A_T$</td>
<td>$F_{GT_T}$</td>
<td>$A_T$</td>
</tr>
<tr>
<td>K-means Features</td>
<td>57.82±0.69</td>
<td>45.44±0.76</td>
<td>57.38±0.26</td>
</tr>
<tr>
<td>K-means Embedding</td>
<td>59.79±0.24</td>
<td>43.91±0.15</td>
<td>57.92±0.25</td>
</tr>
<tr>
<td>Uniform</td>
<td>58.99±0.54</td>
<td>44.73±0.11</td>
<td>58.76±0.10</td>
</tr>
<tr>
<td>iCaRL</td>
<td>60.74±0.09</td>
<td>44.25±0.04</td>
<td>59.70±0.70</td>
</tr>
<tr>
<td>Grad Matching</td>
<td>59.17±0.38</td>
<td>45.44±0.64</td>
<td>59.58±0.28</td>
</tr>
<tr>
<td>SPR</td>
<td>59.56±0.73</td>
<td>44.45±0.55</td>
<td>58.74±0.63</td>
</tr>
<tr>
<td>MetaSP</td>
<td>60.34±0.25</td>
<td>43.74±0.36</td>
<td>57.43±0.54</td>
</tr>
<tr>
<td>Greedy Coreset</td>
<td>59.39±0.16</td>
<td>43.80±0.01</td>
<td>58.22±0.16</td>
</tr>
<tr>
<td>GCR</td>
<td>58.73±0.43</td>
<td>44.8±0.05</td>
<td>58.72±0.63</td>
</tr>
<tr>
<td>PICS</td>
<td>55.64±0.26</td>
<td>39.8±1.12</td>
<td>56.93±0.14</td>
</tr>
<tr>
<td>OCS</td>
<td>52.57±0.37</td>
<td>46.54±0.34</td>
<td>51.77±0.81</td>
</tr>
<tr>
<td>BCSR</td>
<td>61.6±0.14</td>
<td>47.30±0.57</td>
<td>60.7±0.08</td>
</tr>
</tbody>
</table>
Table 2: Experiment results on Multiple Datasets

<table>
<thead>
<tr>
<th>Methods</th>
<th>Balanced</th>
<th>Imbalanced</th>
<th>Label Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$A_T$</td>
<td>$FGT_T$</td>
<td>$A_T$</td>
</tr>
<tr>
<td>K-means</td>
<td>54.63±0.88</td>
<td>0.134±0.007</td>
<td>33.63±2.66</td>
</tr>
<tr>
<td>K-means Embedding</td>
<td>56.34±1.65</td>
<td>0.106±0.019</td>
<td>35.93±1.60</td>
</tr>
<tr>
<td>Uniform</td>
<td>55.94±0.03</td>
<td>0.101±0.032</td>
<td>34.54±2.96</td>
</tr>
<tr>
<td>icRGL</td>
<td>56.19±0.32</td>
<td>0.130±0.102</td>
<td>42.18±1.59</td>
</tr>
<tr>
<td>Grad Matching</td>
<td>53.41±0.46</td>
<td>0.119±0.020</td>
<td>38.16±3.90</td>
</tr>
<tr>
<td>SPR</td>
<td>56.20±1.91</td>
<td>0.124±0.336</td>
<td>40.79±1.73</td>
</tr>
<tr>
<td>MetaSP</td>
<td>57.14±1.10</td>
<td>0.113±0.042</td>
<td>41.32±1.50</td>
</tr>
<tr>
<td>GCR</td>
<td>54.35±0.31</td>
<td>0.125±0.014</td>
<td>35.13±2.79</td>
</tr>
<tr>
<td>PBCS</td>
<td>52.93±0.28</td>
<td>0.152±0.014</td>
<td>37.25±2.93</td>
</tr>
<tr>
<td>BCSR</td>
<td>55.65±2.26</td>
<td>0.062±0.001</td>
<td>40.48±1.39</td>
</tr>
</tbody>
</table>

Table 3: Experiment results on Tiny-ImageNet

<table>
<thead>
<tr>
<th>Methods</th>
<th>Balanced</th>
<th>Imbalanced</th>
<th>Label Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$A_V$</td>
<td>$F_T$</td>
<td>$A_V$</td>
</tr>
<tr>
<td>K-means</td>
<td>41.20±0.75</td>
<td>0.131±0.004</td>
<td>35.27±0.30</td>
</tr>
<tr>
<td>K-means Embedding</td>
<td>41.18±1.21</td>
<td>0.129±0.002</td>
<td>36.29±0.23</td>
</tr>
<tr>
<td>Uniform</td>
<td>41.11±0.52</td>
<td>0.129±0.002</td>
<td>37.07±0.53</td>
</tr>
<tr>
<td>icRGL</td>
<td>43.84±0.09</td>
<td>0.114±0.004</td>
<td>37.65±0.84</td>
</tr>
<tr>
<td>Grad Matching</td>
<td>43.45±0.32</td>
<td>0.105±0.007</td>
<td>37.58±0.39</td>
</tr>
<tr>
<td>SPR</td>
<td>42.79±0.50</td>
<td>0.102±0.009</td>
<td>36.55±0.54</td>
</tr>
<tr>
<td>MetaSP</td>
<td>43.33±0.32</td>
<td>0.127±0.002</td>
<td>36.75±0.57</td>
</tr>
<tr>
<td>GCR</td>
<td>41.02±0.33</td>
<td>0.119±0.017</td>
<td>33.43±0.86</td>
</tr>
<tr>
<td>PBCS</td>
<td>36.99±0.15</td>
<td>0.177±0.002</td>
<td>35.88±0.16</td>
</tr>
<tr>
<td>BCSR</td>
<td>44.13±0.33</td>
<td>0.106±0.001</td>
<td>38.59±0.11</td>
</tr>
</tbody>
</table>

Table 4: Experiment results on Food-101

<table>
<thead>
<tr>
<th>Methods</th>
<th>Balanced</th>
<th>Imbalanced</th>
<th>Label Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$A_V$</td>
<td>$F_T$</td>
<td>$A_V$</td>
</tr>
<tr>
<td>K-means</td>
<td>46.35±1.41</td>
<td>0.057±0.004</td>
<td>33.35±0.93</td>
</tr>
<tr>
<td>K-means Embedding</td>
<td>49.47±0.46</td>
<td>0.040±0.004</td>
<td>34.07±0.89</td>
</tr>
<tr>
<td>Uniform</td>
<td>46.30±0.13</td>
<td>0.048±0.001</td>
<td>34.61±0.16</td>
</tr>
<tr>
<td>icRGL</td>
<td>48.12±0.78</td>
<td>0.035±0.001</td>
<td>36.83±1.06</td>
</tr>
<tr>
<td>Grad Matching</td>
<td>49.13±0.44</td>
<td>0.026±0.002</td>
<td>35.61±0.04</td>
</tr>
<tr>
<td>SPR</td>
<td>48.34±0.54</td>
<td>0.042±0.003</td>
<td>37.41±0.68</td>
</tr>
<tr>
<td>MetaSP</td>
<td>48.84±0.38</td>
<td>0.053±0.006</td>
<td>38.41±0.78</td>
</tr>
<tr>
<td>GCR</td>
<td>47.03±0.17</td>
<td>0.048±0.004</td>
<td>36.99±0.30</td>
</tr>
<tr>
<td>OCS</td>
<td>46.30±0.05</td>
<td>0.103±0.013</td>
<td>34.44±0.29</td>
</tr>
<tr>
<td>BCSR</td>
<td>51.05±0.62</td>
<td>0.039±0.008</td>
<td>39.05±0.09</td>
</tr>
</tbody>
</table>

Implementation Details

Following the implementation of the coreset-based algorithm [6], we use two models for the purposes of model training and coreset selection respectively, where the first model is denoted as $M_{tr}$ for model training, and the second model (also known as a proxy model) is denoted as $M_{cs}$ for coreset selection. In the section of the model training, we adopt a single-head MLP with two hidden layers for Permuted MNIST, and a ResNet-18 for Split CIFAR-100 and other datasets. In the section of the coreset selection, $M_{cs}$ adopts the same architecture as $M_{tr}$. This is the key difference between our work and the Greedy Coreset [6]: they use Neural Tangent Kernels (NTK) [35] while we use a specific deep neural network. Note that NTK performs learning based on fixed features, which is shown to be limited [66]. In contrast, our algorithm allows hierarchical feature learning during the training process. In addition, our algorithm does not rely on discrete decision variables as in [6]: we use first-order methods with gradient and Hessian-inverse-vector product information with convergence guarantee and hence is more efficient in practice. For non-coreset reply methods SPR and MetaSP, we set the size of the method buffer as that in coreset-based methods. Especially for SPR, there are two memory buffers: the delayed buffer $D$ temporarily stocks the incoming data stream, and the purified buffer $P$ maintains the cleansed data. To satisfy the task $T$. $FGT_T$ evaluates the performance drop on the past tasks after training on the task $T$: $FGT_T = \frac{1}{T} \sum_{j=1}^{T} [\max_{j \in 1,...,T-1} (a_{j,i} - a_{T,i})]$. 

Table 4: Experiment results on Food-101

<table>
<thead>
<tr>
<th>Methods</th>
<th>Balanced</th>
<th>Imbalanced</th>
<th>Label Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$A_V$</td>
<td>$F_T$</td>
<td>$A_V$</td>
</tr>
<tr>
<td>K-means</td>
<td>46.35±1.41</td>
<td>0.057±0.004</td>
<td>33.35±0.93</td>
</tr>
<tr>
<td>K-means Embedding</td>
<td>49.47±0.46</td>
<td>0.040±0.004</td>
<td>34.07±0.89</td>
</tr>
<tr>
<td>Uniform</td>
<td>46.30±0.13</td>
<td>0.048±0.001</td>
<td>34.61±0.16</td>
</tr>
<tr>
<td>icRGL</td>
<td>48.12±0.78</td>
<td>0.035±0.001</td>
<td>36.83±1.06</td>
</tr>
<tr>
<td>Grad Matching</td>
<td>49.13±0.44</td>
<td>0.026±0.002</td>
<td>35.61±0.04</td>
</tr>
<tr>
<td>SPR</td>
<td>48.34±0.54</td>
<td>0.042±0.003</td>
<td>37.41±0.68</td>
</tr>
<tr>
<td>MetaSP</td>
<td>48.84±0.38</td>
<td>0.053±0.006</td>
<td>38.41±0.78</td>
</tr>
<tr>
<td>GCR</td>
<td>47.03±0.17</td>
<td>0.048±0.004</td>
<td>36.99±0.30</td>
</tr>
<tr>
<td>OCS</td>
<td>46.30±0.05</td>
<td>0.103±0.013</td>
<td>34.44±0.29</td>
</tr>
<tr>
<td>BCSR</td>
<td>51.05±0.62</td>
<td>0.039±0.008</td>
<td>39.05±0.09</td>
</tr>
</tbody>
</table>
We conduct ablation studies to inspect the effectiveness of individual components in our proposed approach, including the effect of the bilevel optimizer, the smooth top-$K$ regularizer, and the coreset size $K$ (Appendix J) respectively.

**Random initialization with and without bilevel optimizers.** Our new bilevel optimization algorithm (Algorithm 3) is of vital importance for finding a good distribution $w$ for selecting the coreset. To demonstrate this, we design comparative experiments: one experiment initializes $w$ randomly without updating it, and the other one adopts the same initialization strategy but updates $w$ by the proposed bilevel optimizer. Then we sample coresets based on $w$ from two methods respectively. As you can see, the random initialization method is equivalent to Uniform Sampling. We report the experimental results of average accuracy and forgetting on three balanced datasets in Table 5. It can be observed that the sampling strategy based on bilevel optimization significantly outperforms random sampling. In particular, the strategy with bilevel optimization shows $3.23\%$, $2.89\%$, and $4.39\%$ AVG ACC improvement on three benchmarks. Meanwhile, it also obtains $2.20\%$, $1.00\%$, and $3.20\%$ FGT reduction on three benchmarks.

### 5.2 Results

We report the results of average accuracy (AVG ACC) and forgetting (FGT) for all the algorithms on the balanced, imbalanced, and label-noise benchmarks, and the results are presented in Table 1, Table 2, Table 3, Table 4, and Table 9 (Appendix E) respectively. We have the following observations. (i) In the balanced setting, our method outperforms other baselines significantly in terms of AVG ACC. For example, compared with the best coreset selection methods, our BCSR shows $2.21\%$, $4.24\%$, $2.68\%$, $2.11\%$, and $1.7\%$ improvements in AVG ACC on five benchmarks, respectively. (ii) In the imbalanced and label-noise setting, our method also demonstrates relatively higher performance. For example, on imbalanced data of Multiple Datasets, BCSR is better than the best coreset selection algorithm by $4.65\%$ in average accuracy. An interesting observation is that Greedy Coreset [3] does not perform very well, especially in imbalanced and label-noise settings. The reason is that the inner optimization in Greedy Coreset is conducted by an NTK, where only a fixed feature is used but not the learned feature. (iii) From bilevel optimization point of view, our algorithm significantly reduces the running time of the vanilla greedy coreset selection method [3] by at least $58\%$ on continual learning benchmark datasets. For detailed comparison, please check Table 6 in Appendix J. (iv) In addition, the test AVG ACC (Figure 2 and Figure 4 in Appendix F) of the BCSR during the training process shows that our algorithm alleviates catastrophic forgetting and it is comparable to other best baselines. For example, BCSR enjoys the lowest forgetting for almost all datasets under the label-noise setting (except for Multiple Datasets in Table 2). For most experiments on other settings, BCSR achieves a forgetting performance that is comparable to the best methods with at most $1\%$ gap.

### 5.3 Ablation Studies

We conduct ablation studies to inspect the effectiveness of individual components in our proposed approach, including the effect of the bilevel optimizer, the smooth top-$K$ regularizer, and the coreset size $K$ (Appendix J) respectively.
Table 6: The impact of different values of $\lambda$

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>Split CIFAR-100</th>
<th>Permuted MNIST</th>
<th>Multiple Datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$A_T$</td>
<td>$FGT_T$</td>
<td>$A_T$</td>
</tr>
<tr>
<td>0.00</td>
<td>60.43±0.15</td>
<td>0.064±0.005</td>
<td>54.20±1.53</td>
</tr>
<tr>
<td>0.01</td>
<td>60.56±2.05</td>
<td>0.074±0.016</td>
<td>55.09±2.94</td>
</tr>
<tr>
<td>0.10</td>
<td>61.04±0.53</td>
<td>0.063±0.007</td>
<td>57.20±0.58</td>
</tr>
<tr>
<td>1.00</td>
<td>58.80±1.58</td>
<td>0.084±0.017</td>
<td>53.13±2.74</td>
</tr>
<tr>
<td>5.00</td>
<td>59.53±0.42</td>
<td>0.075±0.012</td>
<td>53.54±2.02</td>
</tr>
</tbody>
</table>

To verify the effectiveness of our bilevel optimizer, we compare the loss curve with Greedy Coreset that uses NTK. The result is presented in Figure 3. There are a number of stream mini-batches in each task. The bilevel optimizer trains over each mini-batch, where the upper loss is plotted in the figure. In the experiment, we plot the loss value for every 5 mini-batches. Within each task, the loss from BCSR gradually decreases with slight fluctuations, and it increases only when encountering a new task. In contrast, the loss value of the Greedy Coreset approach always stays large. It indicates that our bilevel optimizer is more effective than the Greedy Coreset approach.

**Effectiveness of the regularizer.** In our algorithm, the bilevel formulation has a smooth top-$K$ loss as a regularizer in the objective function to promote the probability distribution to have $K$ nonzero entries. The goal is to make sure that the summation of top-$K$ entries of the learned probability vector is large, which increases confidence in the coreset selection. The hyperparameter $\lambda$ is used to balance the cross-entropy loss and the regularization term. We explore the effects on the performance with different $\lambda$, and list the results in Table 6.

This ablation experiment is performed on our framework BCSR, and average accuracy and forgetting on three balanced benchmarks are reported. When $\lambda = 0.10$, BCSR can reach the best performance (The highest AVG ACC and lowest FGT) on Split CIFAR-100 and Permuted MNIST. While on Multiple Datasets, BCSR performs the best when $\lambda = 1.0$. This dataset contains more categories, and hence it is more challenging to select the appropriate representative samples. In this case, larger $\lambda$ would emphasize more on maximizing the top-$K$ probability and help select better representative samples. In addition, we observe that $\lambda$ set as a too large or too small value will damage the performance, which is in line with the observations in standard regularized empirical risk minimization problems such as overfitting and underfitting. Please refer to Appendix H for further analysis.

6 Theoretical Analysis

In this section, we provide a theoretical analysis for our proposed method. In particular, we establish convergence rate for our algorithm BCSR.

**Theorem 1.** Suppose standard assumptions hold in bilevel optimization (see Assumptions 1, 2 and 3 in Appendix M). Choose parameters $\lambda$, $\alpha$, $\eta$ and $N$ such that $(1 + \lambda)(1 - \alpha \mu)^N(1 + \frac{8rL^2}{\eta \mu}) \leq 1 - \eta \mu$, where $r = \frac{C_Q^2}{(\frac{C_Q}{L^2} + L^2)^2}$ and $C_Q = \frac{Q \alpha M}{\mu} + \eta^2 Q^2 \rho M + \eta Q L$. Furthermore, choose the stepsize $\beta$ such that $6\omega \beta^2 L^2 < \frac{1}{4\eta \mu}$ and $\beta \leq \frac{1}{4L_\phi}$, where the constant $\omega$ is given by Equation (16) in Appendix A.
Then, we have the following convergence result.

\[
\frac{1}{J} \sum_{j=0}^{J-1} \mathbb{E}\|G_j\|^2 \leq O\left(\frac{D_\phi \epsilon}{\beta J} + \frac{1}{|\mathcal{B}|} + \frac{D_0}{\eta \mu J}\right).
\]

where \(G_j := \frac{1}{\beta}(w_j - \mathcal{P}_{\Delta^M}(w_j - \beta \nabla \phi(w_j)))\) denote the generalized projected gradient, \(D_\phi := \phi(w_0) - \min_w \phi(w) > 0, D_0 = \|\theta_0^0 - \theta_0^0\|^2 + \|v_0^0 - v_0^0\|^2\), \(L_\phi\) is the smoothness parameter of the total objective \(\phi(w)\) whose form is proved in Appendix A.

Theorem 1 provides a general convergence result for the proposed bilevel algorithm, which allows for a very flexible selection of subloop lengths \(N\) and \(Q\) as long as the inequality \((1 + \lambda)(1 - \alpha \mu)^N(1 + \frac{3\mu L^2}{\eta}) \leq 1 - \eta \mu\) holds given proper stepsizes \(\lambda, \eta, \alpha\). For example, in most of our experiments, the choice of \(N = 1\) and \(Q = 3\) works the best. Then, in this case, we further specify the parameters in Theorem 1 and provide the following corollary.

**Corollary 1.** Under the same setting as in Theorem 1, choose \(N = 1, Q = 3\) and set \(\lambda = \frac{2\mu}{\mu L^2} \leq \frac{1}{\eta}\). Then, to make sure an \(\epsilon\)-accurate stationary point, i.e., \(\frac{1}{3} \sum_{j=0}^{J-1} \mathbb{E}\|G_j\|^2 \leq \epsilon^2\), the number of iterations is \(O(\epsilon^{-2})\), each using \(|\mathcal{B}| = O(\epsilon^{-2})\) of samples from the standard Gaussian distribution \(\mathcal{N}(0, 1)\).

Corollary 1 shows that the proposed bilevel algorithm converges to an \(\epsilon\)-accurate stationary point using only \(O(\epsilon^{-2})\) iterations and \(O(\epsilon^{-2})\) samples drawn from \(\mathcal{N}(0, 1)\) per iteration. Note the the large batch size \(|\mathcal{B}| = O(\epsilon^{-2})\) is necessary here to guarantee a convergence rate of \(O(1/\epsilon^4)\), which matches the results in solving single-level constrained nonconvex problems [24]. In addition, it is computationally tractable because sampling from a known Gaussian distribution is easy and cheap.

7 Conclusion

In this paper, we advance the state-of-the-art bilevel coreset selection in continual learning. We first introduce a new bilevel formulation with smoothed top-\(K\) regularization and then design an efficient bilevel optimizer as a solver. We conduct extensive experiments in continual learning benchmark datasets to demonstrate the effectiveness of our proposed approach. We also show that the bilevel optimizer can efficiently find \(\epsilon\)-stationary point with \(O(1/\epsilon^4)\) computational complexity, which matches the best complexity of projected SGD for a single-level problem.

Acknowledgments and Disclosure of Funding

We would like to thank the anonymous reviewers for their helpful comments. Jie Hao and Mingrui Liu are both supported by a grant from George Mason University. Computations were run on ARGO, a research computing cluster provided by the Office of Research Computing at George Mason University (URL: https://orc.gmu.edu).

References


A Proof of Theorem 1

Let \( \Delta^{(n)} := \{ w : 0 \leq w(i) \leq 1, \|w\|_1 = 1 \} \) denote the constraint set of the upper-level problem. We first provide some important inequalities.

Recall that \( v_j^q \) be the \( q^{th} \) GD iterate in solving the linear system \( \nabla_\theta^2 L(\theta_j^N, w_j)v = \sum_{i=1}^{n} \nabla \ell_i(\theta_j^N) \) at iteration \( j \) via the following process:

\[
v_j^{q+1} = (I - \eta \nabla_\theta^2 L(\theta_j^N, w_j))v_j^q + \eta \sum_{i=1}^{n} \nabla \ell_i(\theta_j^N) \tag{6}
\]

which, by telescoping Equation (6) over \( q \) from 0 to \( Q - 1 \), yields

\[
v_j^Q = (I - \eta \nabla_\theta^2 L(\theta_j^N, w_j))^Q v_j^0 + \eta \sum_{q=0}^{Q-1} (I - \eta \nabla_\theta^2 L(\theta_j^N, w_j))^q \sum_{i=1}^{n} \nabla \ell_i(\theta_j^N) \tag{7}
\]

Let \( v_j^* \) be the solution of the linear system \( \nabla_\theta^2 L(\theta_j^*, w_j)v = \sum_{i=1}^{n} \nabla \ell_i(\theta_j^*) \), and then we have

\[
v_j^* = (I - \eta \nabla_\theta^2 L(\theta_j^*, w_j))^Q v_j^* + \eta \sum_{q=0}^{Q-1} (I - \eta \nabla_\theta^2 L(\theta_j^*, w_j))^q \sum_{i=1}^{n} \nabla \ell_i(\theta_j^*) \tag{8}
\]

Combining Equation (6) and Equation (7), noting that \( v_j^0 = v_j^{Q-1} \) and using Assumption 2 that \( \|v_j^*\| \leq \|(\nabla_\theta^2 L(\theta_j^*, w_j))^{-1}\| \sum_{i=1}^{n} \|\nabla \ell_i(\theta_j^*)\| \leq \frac{M}{\mu} \), the difference between \( v_j^Q \) and \( v_j^* \) can be bounded as

\[
\|v_j^Q - v_j^*\| \leq \|(I - \eta \nabla_\theta^2 L(\theta_j^N, w_j))^Q - (I - \eta \nabla_\theta^2 L(\theta_j^*, w_j))^Q\| \frac{M}{\mu} + (1 - \eta \mu)^Q \|v_j^{Q-1} - v_j^*\|
\]

\[
+ \eta M \sum_{q=0}^{Q-1} (I - \eta \nabla_\theta^2 L(\theta_j^N, w_j))^q \sum_{q=0}^{Q-1} (I - \eta \nabla_\theta^2 L(\theta_j^*, w_j))^q \|
\]

\[
+ (1 - (1 - \eta \mu)^Q) \frac{L}{\mu} \|\theta_j^* - \theta_j^N\|. \tag{9}
\]

We next bound \( \Delta_q := \|(I - \eta \nabla_\theta^2 L(\theta_j^N, w_j)) - (I - \eta \nabla_\theta^2 L(\theta_j^*, w_j))\| \) in Equation (9) as:

\[
\Delta_q \leq (1 - \eta \mu) \Delta_{q-1} + (1 - \eta \mu)^{q-1} \eta \rho \|\theta_j^N - \theta_j^*\|. \tag{10}
\]

which, by telescoping Equation (10) and in conjunction with Equation (9), yields

\[
\|v_j^Q - v_j^*\| \leq Q(1 - \eta \mu)^{Q-1} \eta \rho \frac{M}{\mu} \|\theta_j^N - \theta_j^*\| + (1 - \eta \mu)^Q \|v_j^{Q-1} - v_j^*\|
\]

\[
+ \eta M \sum_{q=0}^{Q-1} q(1 - \eta \mu)^{q-1} \eta \rho \|\theta_j^N - \theta_j^*\| + (1 - (1 - \eta \mu)^Q) \frac{L}{\mu} \|\theta_j^* - \theta_j^N\|
\]

\[
\leq Q(1 - \eta \mu)^{Q-1} \rho M \frac{\eta}{\mu} \|\theta_j^N - \theta_j^*\| + (1 - \eta \mu)^Q \|v_j^{Q-1} - v_j^*\|
\]

\[
+ (1 - \eta \mu)^Q \|v_j^{Q-1} - v_j^*\| + \frac{1 - (1 - \eta \mu)^Q(1 + \eta Q \mu)}{\mu^2} \rho M \|\theta_j^N - \theta_j^*\|
\]

\[
+ (1 - (1 - \eta \mu)^Q) \frac{L}{\mu} \|\theta_j^* - \theta_j^N\|
\]

which, combined with \( \|v_j^* - v_j^{Q-1}\| \leq \frac{(1 - \mu)}{\mu} + \frac{M \rho}{\mu^2} \) and using the fact that \( (1 - x)^Q \geq 1 - xQ \) for \( 0 \leq x \leq 1 \), yields

\[
\mathbb{E}\|v_j^Q - v_j^*\|^2 \leq (1 - \eta \mu) \mathbb{E}\|v_j^{Q-1} - v_j^{Q-1}\|^2 + \frac{4}{\eta \mu} C_0^2 \|\theta_j^* - \theta_j^N\|^2
\]

\[
+ \frac{4}{\eta \mu} \left( \frac{L}{\mu} + \frac{M \rho}{\mu^2} \right)^2 \left( \frac{L}{\mu} + 1 \right) \mathbb{E}\|w_j - w_{j-1}\|^2 \tag{12}
\]
where the constant \( C_Q := \frac{QaM_n}{\mu} + \eta^2Q^2 \rho M + \eta Q L \).

The next step is to characterize the error induced by the lower-level updates on \( \theta \).

Note that \( \theta_j = \arg \min_\theta L(\theta, w_j) \). Using Assumptions 1 and 2 we have

\[
\|\theta_j^N - \theta_j^*\|^2 \leq (1 - \alpha \mu)^N \|\theta_j^0 - \theta_j^*\|^2,
\]

(13)

which, in conjunction with \( \theta_j^0 = \theta_j^{N-1} \) and Lemma 2.2 in (25), yields

\[
\mathbb{E}\|\theta_j^N - \theta_j^*\|^2 \leq (1 - \alpha \mu)^N (1 + \lambda) \mathbb{E}\|\theta_j^{N-1} - \theta_j^{N-1}\|^2
\]

+ (1 - \alpha \mu)^N \left(1 + \frac{1}{\lambda} \right) \frac{L^2}{\mu^2} \mathbb{E}\|w_j - w_j^{N-1}\|^2.
\]

(14)

Recall the definition that \( r = \frac{C_Q^2}{(\frac{\alpha}{\eta} + L)^2} \). Then, combining Equation (12) and Equation (14) yields we can obtain

\[
\left(1 + \frac{\rho^2M^2}{L^2\mu^2}\right) \mathbb{E}\|\theta_j^N - \theta_j^*\|^2 + \mathbb{E}\|v_j^Q - v_j^*\|^2
\]

\[
\leq (1 + \lambda)(1 - \alpha \mu)^N \left(1 + \frac{\rho^2M^2}{L^2\mu^2}\right) \left(1 + \frac{8rL^2}{\eta \mu}\right) \mathbb{E}\|\theta_j^{N-1} - \theta_j^{N-1}\|^2
\]

+ (1 - \eta \mu)\mathbb{E}\|v_j^Q - v_j^{N-1}\|^2 + \omega \mathbb{E}\|w_j - w_j^{N-1}\|^2;
\]

(15)

where the constant \( \omega \) is given by

\[
\omega := \left(1 + \frac{1}{\lambda}\right)(1 - \alpha \mu)^N \left(1 + \frac{\rho^2M^2}{L^2\mu^2}\right) \frac{L^2}{\mu^2}
\]

\[
+ \frac{8L^4}{\eta \mu \mu^2} \left(1 + \frac{\rho^2M^2}{L^2\mu^2}\right) \left(\frac{4}{\mu^2} + r(1 - \alpha \mu)^N \left(1 + \frac{1}{\lambda}\right)\right).
\]

(16)

Recall that we choose \( (1 + \lambda)(1 - \alpha \mu)^N \left(1 + \frac{8rL^2}{\eta \mu}\right) \leq 1 - \eta \mu \). Then, we obtain from Equation (15) that

\[
\left(1 + \frac{\rho^2M^2}{L^2\mu^2}\right) \mathbb{E}\|\theta_j^N - \theta_j^*\|^2 + \mathbb{E}\|v_j^Q - v_j^*\|^2
\]

\[
\leq (1 - \eta \mu) \left(1 + \frac{\rho^2M^2}{L^2\mu^2}\right) \mathbb{E}\|\theta_j^{N-1} - \theta_j^{N-1}\|^2
\]

\[
+ (1 - \eta \mu)\mathbb{E}\|v_j^Q - v_j^{N-1}\|^2 + \omega \mathbb{E}\|w_j - w_j^{N-1}\|^2.
\]

(17)

Let \( \delta_j := (1 + \frac{\rho^2M^2}{L^2\mu^2}) \mathbb{E}\|\theta_j^N - \theta_j^*\|^2 + \mathbb{E}\|v_j^Q - v_j^*\|^2 \). Then, incorporating the update that \( w_j = \mathcal{P}_{\Delta_\ast}(w_j^{N-1} - \beta \varphi_j) \) into Equation (17) yields

\[
\delta_j \leq (1 - \eta \mu)\delta_{j-1} + 2\omega \beta^2 \mathbb{E}\left|\frac{1}{\beta}(w_j^{N-1} - \mathcal{P}_{\Delta_\ast}(w_j^{N-1} - \beta \nabla \phi(w_j^{N-1})))\right|^2
\]

\[
+ 2\omega \beta^2 \|\varphi_j - \nabla \phi(w_j^{N-1})\|^2.
\]

(18)

We next bound the last term in the above Equation (18). Based on the definition of the hypergradient estimate \( \varphi_j = \nabla_w R(w, \delta; B) - \nabla_w \nabla_{\theta} L(\theta_j^N, w_j) v_j^Q \), we have

\[
\mathbb{E}\|\varphi_j - \nabla \phi(w_j)^2\|
\]

\[
\leq 3\mathbb{E}\|\nabla R(w_j, \delta; B) - \nabla R(w_j, \delta)\|^2 + \frac{3\rho^2M^2}{\mu^2} \mathbb{E}\|\theta_j^N - \theta_j^*\|^2 + 3L^2 \mathbb{E}\|v_j^Q - v_j^*\|^2
\]

\[
\leq \frac{3}{|B|^2} \sum_{z \in B} \mathbb{E}\|\nabla R(w_j, \delta; z) - \nabla R(w_j, \delta)\|^2 + 3L^2 \left(1 + \frac{\rho^2M^2}{\mu^2} L^2\right) \mathbb{E}\|\theta_j^N - \theta_j^*\|^2 + 3L^2 \mathbb{E}\|v_j^Q - v_j^*\|^2
\]

\[
\leq 6K \frac{1}{|B|} + 3L^2 \left(1 + \frac{\rho^2M^2}{\mu^2} L^2\right) \mathbb{E}\|\theta_j^N - \theta_j^*\|^2 + 3L^2 \mathbb{E}\|v_j^Q - v_j^*\|^2 = \frac{6K}{|B|} + 3L^2 \delta_j,
\]

(19)
where (i) follows because $\nabla R(w_j, \delta; z)$ is an unbiased estimate of $\nabla R(w, \delta)$ and (ii) follows from Proposition[1]. Substituting Equation (19) into Equation (18) yields

$$\delta_j \leq (1 - \eta \mu + 6\omega \beta^2 L^2)\delta_{j-1} + \frac{12\omega K \beta^2}{|B|} + 2\omega \beta^2 \|G_{j-1}\|^2. \quad (20)$$

Let $\tau := 1 - \eta \mu + 6\omega \beta^2 L^2$. Then, telescoping Equation (20) yields

$$\delta_j \leq \tau^j \delta_0 + \frac{12\omega K \beta^2}{(1 - \tau)|B|} + 2\omega \beta^2 \sum_{i=0}^{j-1} \tau^i \|G_{j-1-i}\|^2. \quad (21)$$

Based on Equation (21), we are ready to provide the final convergence result. First, based on the Lipschitz continuity in Assumption[1] Assumption[2] and Assumption[3] we have

$$\|\nabla \phi(w_1) - \nabla \phi(w_2)\| \leq L_{\phi}\|w_1 - w_2\|,$$

where the constant $L_{\phi} = \frac{\sqrt{K \mu}}{s} + \frac{L^2 + \rho M^2}{\mu^2} + \frac{2\omega L M + L^3}{\mu^3} + \frac{\rho L^2 M}{\mu^3}$ is the smoothness parameter. Then, this inequality further implies

$$\phi(w_{j+1}) \leq \phi(w_j) + \langle \nabla \phi(w_j), w_{j+1} - w_j \rangle + \frac{L_{\phi}}{2}\|w_{j+1} - w_j\|^2$$

$$\leq \phi(w_j) + \frac{1}{\beta}(\beta \phi_j, P_{\Delta^c}(w_j - \beta \phi_j) - w_j) + \langle \nabla \phi(w_j) - \phi_j, P_{\Delta^c}(w_j - \beta \phi_j) - w_j \rangle$$

$$+ \frac{L_{\phi}}{2}\|w_{j+1} - w_j\|^2. \quad (22)$$

To analyze the second term at the right hand side of the above Equation (22), we note that

$$-(\beta \phi_j, P_{\Delta^c}(w_j - \beta \phi_j) - w_j) = \langle (w_j - \beta \phi_j) - P_{\Delta^c}(w_j - \beta \phi_j), P_{\Delta^c}(w_j - \beta \phi_j) - w_j \rangle + \|P_{\Delta^c}(w_j - \beta \phi_j) - w_j\|^2,$$

which, in conjunction with the property of projection on convex set that $\langle x - P_{\Delta^c}(x), y - P_{\Delta^c}(x) \rangle \leq 0$ for any $y \in S$ and the fact that $w_j = P_{\Delta^c}(w_{j-1} - \beta \phi_{j-1})$ yields

$$-(\beta \phi_j, P_{\Delta^c}(w_j - \beta \phi_j) - w_j) \geq \|P_{\Delta^c}(w_j - \beta \phi_j) - w_j\|^2 \geq 0. \quad (23)$$

Then, substituting Equation (23) into Equation (22) yields

$$\phi(w_{j+1}) \leq \phi(w_j) + \langle \nabla \phi(w_j) - \phi_j, P_{\Delta^c}(w_j - \beta \phi_j) - w_j \rangle + \frac{L_{\phi}}{2}\|w_{j+1} - w_j\|^2$$

$$\leq \phi(w_j) - \frac{\beta}{2}\|\phi_j\|^2 + \frac{\beta}{2}\|\phi_j - \nabla \phi(w_j)\|^2 + \frac{L_{\phi} \beta^2}{2}\|\phi_j\|^2$$

$$\leq \phi(w_j) - \left(\frac{\beta}{4} - \frac{L_{\phi} \beta^2}{4}\right)\|G_j\|^2 + \left(\beta - \frac{L_{\phi} \beta^2}{2}\right)\|\phi_j - \nabla \phi(w_j)\|^2, \quad (24)$$

where we use the notation that $\phi_j = \frac{1}{n}(w_j - P_{\Delta^c}(w_j - \beta \phi_j))$, and the non-expansive property of projection. Then, taking the expectation and incorporating Equation (19) into Equation (24), we have

$$E\phi(w_{j+1}) \leq E\phi(w_j) - \left(\frac{\beta}{4} - \frac{L_{\phi} \beta^2}{4}\right)E\|G_j\|^2 + \left(\beta - \frac{L_{\phi} \beta^2}{2}\right)E\|\phi_j - \nabla \phi(w_j)\|^2$$

$$\leq E\phi(w_j) - \left(\frac{\beta}{4} - \frac{L_{\phi} \beta^2}{4}\right)E\|G_j\|^2 + \left(\beta - \frac{L_{\phi} \beta^2}{2}\right)\left(\frac{6K}{|B|} + 3L^2 \delta_j\right). \quad (25)$$

Then, substituting Equation (21) into the above Equation (25) yields

$$E\phi(w_{j+1}) \leq E\phi(w_j) - \left(\frac{\beta}{4} - \frac{L_{\phi} \beta^2}{4}\right)E\|G_j\|^2 + \left(\beta - \frac{L_{\phi} \beta^2}{2}\right)\left(\frac{6K}{|B|}\right.$$

$$+ 3L^2 \left(\beta - \frac{L_{\phi} \beta^2}{2}\right)\sum_{i=0}^{j-1} \frac{12\omega K \beta^2}{(1 - \tau)|B|} + 2\omega \beta^2 \sum_{i=0}^{j-1} \tau^i E\|G_{j-1-i}\|^2\right)$$

$$\leq E\phi(w_j) - \left(\frac{\beta}{4} - \frac{L_{\phi} \beta^2}{4}\right)E\|G_j\|^2 + \left(1 + 6\omega \beta^2 L^2 \frac{1}{1 - \tau}\right)\left(\beta - \frac{L_{\phi} \beta^2}{2}\right)\frac{6K}{|B|}$$

$$+ 3L^2 \left(\beta - \frac{L_{\phi} \beta^2}{2}\right)\tau^j \delta_0 + 6\omega \beta^2 L^2 \left(\beta - \frac{L_{\phi} \beta^2}{2}\right) \sum_{i=0}^{j-1} \tau^i E\|G_{j-1-i}\|^2,$$
which, by taking the telescoping over $j$ from 0 to $J - 1$, yields

\[
\frac{1}{J} \sum_{j=0}^{J-1} \left( \frac{1}{4} - \frac{L_\phi \beta}{4} \right) E\|G_j\|^2 \\
\leq \frac{\phi(w_0) - \min_w \phi(w)}{\beta J} + \left( 1 + 6\omega \beta^2 L^2 \right) \left( \frac{1 - L_\phi \beta}{2} \right) \frac{6K}{|B|} \\
+ 3L^2 \left( \frac{1 - L_\phi \beta}{2} \right) \frac{\delta_0}{(1 - \tau)J} + 6\omega \beta^2 L^2 \left( \frac{1 - L_\phi \beta}{2} \right) \frac{1}{J} \sum_{j=0}^{J-1} \sum_{\tau=t=0}^{j-1} \tau^t E\|G_{j-t}\|^2 \\
\leq \frac{\phi(w_0) - \min_w \phi(w)}{\beta J} + \left( 1 + 6\omega \beta^2 L^2 \right) \left( \frac{1 - L_\phi \beta}{2} \right) \frac{6K}{|B|} \\
+ 3L^2 \left( \frac{1 - L_\phi \beta}{2} \right) \frac{\delta_0}{(1 - \tau)J} + 6\omega \beta^2 L^2 \left( \frac{1 - L_\phi \beta}{2} \right) \frac{1}{(1 - \tau)J} \sum_{j=0}^{J-1} E\|G_j\|^2.
\]

Rearranging the above inequality, we have

\[
\frac{1}{J} \sum_{j=0}^{J-1} \left( \frac{1}{4} - \frac{L_\phi \beta}{4} - \frac{6\omega \beta^2 L^2}{1 - \tau} \left( \frac{1 - L_\phi \beta}{2} \right) \right) E\|G_j\|^2 \\
\leq \frac{\phi(w_0) - \min_w \phi(w)}{\beta J} + \left( 1 + 6\omega \beta^2 L^2 \right) \left( \frac{1 - L_\phi \beta}{2} \right) \frac{6K}{|B|} + \frac{3L^2 \left( 1 - \frac{L_\phi \beta}{2} \right) \delta_0}{(1 - \tau)J}.
\]

Recalling the definition that $\tau = 1 - \eta \mu + 6\omega \beta^2 L^2$, and noting that we choose the stepsize $\beta$ such that $\frac{6\omega \beta^2 L^2}{\beta} < \frac{1}{2} \eta \mu$ and $\beta \leq \frac{1}{4\lambda}$, we can simplify Equation (26) as

\[
\frac{1}{16J} \sum_{j=0}^{J-1} E\|G_j\|^2 \leq \frac{\phi(w_0) - \min_w \phi(w)}{\beta J} + \frac{27K}{4|B|} + \frac{27L^2 \delta_0}{8\eta \mu J}.
\]

From the gradient descent based updates, we have $\delta_0 = \left( 1 + \frac{\log^2 M}{\eta^2 \mu^2} \right) E\|\theta_0^N - \theta_0^*\|^2 + E\|v_0^N - v_0^*\|^2 \leq \left( 1 + \frac{\log^2 M}{\eta^2 \mu^2} \right) \|\theta_0^0 - \theta_0^*\|^2 + \|v_0^0 - v_0^*\|^2 < +\infty$. Then, the proof is complete.

**B Proof of Corollary [1]**

Based on the definition of $C_Q = \frac{Q \eta M \mu}{\mu} + \eta Q^2 \rho M + \eta Q L$ and noting that $\eta \leq \frac{1}{L} \leq \frac{1}{\mu}$ and $Q = 3$, we have $C_Q \leq 12\eta (\frac{\rho M}{\mu} + L)$, which, combined with the definition that $r = \frac{C_Q}{Q \eta M \mu (\mu + L)}$, yields $r \leq 144\eta^2$. Then, based on the choice of $N = 1$ and $\lambda = \frac{\alpha \mu}{2}$, we have

\[
(1 + \lambda)(1 - \alpha \mu)^N \left( 1 + \frac{8r L^2}{\eta \mu} \right) \leq \left( 1 - \frac{\alpha \mu}{2} \right) \left( 1 + \frac{1152\eta L^2}{\mu} \right) \leq 1 - \frac{\alpha \mu}{2} + \frac{1152\eta L^2}{\mu},
\]

which, in conjunction with $\eta \leq \frac{\mu^2}{3608L^2 \alpha}$ and $\alpha \leq \frac{1}{L}$, yields

\[
(1 + \lambda)(1 - \alpha \mu)^N \left( 1 + \frac{8r L^2}{\eta \mu} \right) \leq 1 - \frac{1}{4} \alpha \mu \leq 1 - \eta \mu.
\]

This implies that the inequality $(1 + \lambda)(1 - \alpha \mu)^N \left( 1 + \frac{8r L^2}{\eta \mu} \right) \leq 1 - \eta \mu$ required by Theorem [1] is satisfied. Then, treating $\mu, \eta, L, \rho, M, \alpha, \beta, K, \|\theta_0^0 - \theta_0^*\|^2$ and $\|v_0^0 - v_0^*\|^2$ as constants independent of the total number $J$ of iterations, we have

\[
\frac{1}{J} \sum_{j=0}^{J-1} E\|G_j\|^2 \leq O \left( \frac{1}{J} + \frac{1}{|B|} \right).
\]

Then, to ensure an $\epsilon$-accurate stationary point, the number of iterations is $\epsilon^{-2}$ with a batch size $|B| = O(\epsilon^{-2})$.  

19
C Experiment Hyperparameters

The experimental hyperparameters are listed in Table 7 including the memory size: $|\mathcal{M}|$, coreset size: $|\mathcal{S}|$, stream batch size: $|\mathcal{B}|$, learning rate for $M_{tr}$: $lr_{t}$, learning rate for $M_{cs}$: $lr_{p}$, learning rate for sample weights: $lr_{w}$, regularization efficient: $\lambda$, epochs for training model: $E$, outer loops in bilevel optimization: $J$, inner loops in bilevel optimization: $N$, loops for estimating the Hessian-inverse-vector product: $Q$, a factor of Gaussian noise: $\delta$.

### Table 7: Hyperparameters settings in experiments.

<table>
<thead>
<tr>
<th>Hyperparameters</th>
<th>Permutation Split MNIST</th>
<th>Split CIFAR-100</th>
<th>Split Tiny-ImageNet</th>
<th>Split Multiple Datasets</th>
<th>Split Food-101</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>\mathcal{M}</td>
<td>$</td>
<td>200</td>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>$</td>
<td>\mathcal{S}</td>
<td>$</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>$</td>
<td>\mathcal{B}</td>
<td>$</td>
<td>50</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>$lr_{t}$</td>
<td>0.005</td>
<td>0.15</td>
<td>0.20</td>
<td>0.1</td>
<td>0.15</td>
</tr>
<tr>
<td>$lr_{p}$</td>
<td>5.0</td>
<td>5.0</td>
<td>10</td>
<td>5.0</td>
<td>10</td>
</tr>
<tr>
<td>$lr_{w}$</td>
<td>5.0</td>
<td>5.0</td>
<td>10</td>
<td>5.0</td>
<td>10</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>$E$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$J$</td>
<td>5</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>$N$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$Q$</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$\delta$</td>
<td>$1e-3$</td>
<td>$1e-3$</td>
<td>$1e-3$</td>
<td>$1e-3$</td>
<td>$1e-3$</td>
</tr>
</tbody>
</table>

D Running Time Comparison

We evaluate the running time (wall-clock time) of baseline methods on Split Tiny-ImageNet and Split Food-101 in Table 8. All the training sections keep the same among these algorithms. Since there is no advanced coreset selection procedures in K-means Features, K-means Embedding, Uniform sampling, iCaRL, and Grad Matching, their running costs are pretty low but with worse performance (i.e., accuracy and forgetting) compared with our approach BCSR. PBCS and GCR, SPR and MetaSP don’t involve bilevel formulations, they cannot guarantee accurate coreset sampling though with lower time cost against BCSR. Compared with OCS, BCSR takes 23% and 35% running time reduction on Tiny-ImageNet and Split Food-101, respectively. Because similarity computing based on data pairs is computationally expensive for OCS. Greedy Coreset takes much more time than BCSR due to the usage of NTK.

### Table 8: Running time.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Time (hours) Tiny-ImageNet</th>
<th>Time (hours) Split Food-101</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-means Features</td>
<td>0.15</td>
<td>0.02</td>
</tr>
<tr>
<td>K-means Embedding</td>
<td>0.41</td>
<td>0.04</td>
</tr>
<tr>
<td>Uniform</td>
<td>0.13</td>
<td>0.03</td>
</tr>
<tr>
<td>iCaRL</td>
<td>0.41</td>
<td>0.04</td>
</tr>
<tr>
<td>Grad Matching</td>
<td>0.57</td>
<td>0.06</td>
</tr>
<tr>
<td>SPR</td>
<td>0.77</td>
<td>0.11</td>
</tr>
<tr>
<td>MetaSP</td>
<td>0.86</td>
<td>0.13</td>
</tr>
<tr>
<td>PBCS</td>
<td>1.13</td>
<td>0.15</td>
</tr>
<tr>
<td>GCR</td>
<td>0.91</td>
<td>0.13</td>
</tr>
<tr>
<td>Greedy Coreset</td>
<td>6.26</td>
<td>0.83</td>
</tr>
<tr>
<td>OCS</td>
<td>3.45</td>
<td>0.40</td>
</tr>
<tr>
<td>BCSR</td>
<td>2.65</td>
<td>0.26</td>
</tr>
</tbody>
</table>

E Experiments on Permuted-MNIST

The experiment results on Permuted-MNIST is presented in this section. BCSR outperforms other baselines significantly in AVG ACC on all the data settings, while showing relatively low forgetting.
Table 9: Experiment results on Permuted MNIST

<table>
<thead>
<tr>
<th>Methods</th>
<th>Balanced ( A_T )</th>
<th>Imbalanced ( A_T )</th>
<th>Label Noise ( A_T )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \pm 0.012 )</td>
<td>( \pm 0.009 )</td>
<td>( \pm 0.003 )</td>
</tr>
<tr>
<td>K-means Features</td>
<td>54.30±0.93</td>
<td>39.46±0.11</td>
<td>46.71±0.64</td>
</tr>
<tr>
<td>K-means Embedding</td>
<td>54.78±1.83</td>
<td>41.97±2.02</td>
<td>47.36±1.58</td>
</tr>
<tr>
<td>Uniform</td>
<td>53.74±0.35</td>
<td>38.49±1.89</td>
<td>46.35±1.72</td>
</tr>
<tr>
<td>iCaRL</td>
<td>52.62±0.01</td>
<td>48.64±0.58</td>
<td>48.14±0.83</td>
</tr>
<tr>
<td>Grad Matching</td>
<td>54.76±1.61</td>
<td>33.67±1.24</td>
<td>48.24±0.42</td>
</tr>
<tr>
<td>SPR</td>
<td>54.24±0.45</td>
<td>40.79±0.83</td>
<td>48.23±0.75</td>
</tr>
<tr>
<td>MetaSP</td>
<td>54.63±0.31</td>
<td>41.32±0.94</td>
<td>48.84±0.77</td>
</tr>
<tr>
<td>Greedy Coreset</td>
<td>54.10±0.81</td>
<td>26.68±13.39</td>
<td>49.45±1.73</td>
</tr>
<tr>
<td>GCR</td>
<td>54.53±0.64</td>
<td>40.63±0.50</td>
<td>48.64±0.75</td>
</tr>
<tr>
<td>BCSR</td>
<td>56.23±0.29</td>
<td>50.19±0.47</td>
<td>48.08±1.44</td>
</tr>
<tr>
<td>OCS</td>
<td>54.37±0.34</td>
<td>50.82±3.03</td>
<td>48.04±0.003</td>
</tr>
</tbody>
</table>

F Evolution of Average Accuracy

We present the learning process of different methods on Multiple Datasets in Figure 4. There are five tasks, where the average accuracy is tested after training each task. BCSR shows better performance than other baselines in different settings, including Balanced, Imbalanced, and label-noise.

![Figure 4: Evolution of average accuracy during the continual learning process for Multiple Datasets.](image)

G Possibility Distribution of Candidate Coreset

To explore the effect of top-\( K \) regularizer, we observe the possibility distribution of sample weights after each bilevel optimization, where sample weights of candidate coresets are initialized uniformly. The novel top-\( K \) regularizer makes sure that the summation of the top-\( K \) entries of the learned probability vector is large, such that we can confidently choose a coreset with size \( K \). We show the weight distribution after optimization in Figure 5, where you can find that top-\( K \) weights are much higher than others (with a margin 2%), which easily distinguishes the top-\( K \) core samples from candidates.

![Figure 5: Possibility distribution of candidate coreset for one mini-batch stream data.](image)

![Figure 6: The hypergradients in an outer-loop.](image)
The Effect of Top-$K$ Regularizer

To further analyze the effects of top-$K$ regularizer, we conduct the ablation study with different values of regularizer coefficient $\lambda$ on balanced Split CIFAR-100. The performance results with different $\lambda$ are shown in Table 10 and the corresponding average top-$K$ summations of coreset weights are in Table 11. In our experiment, there are 50 candidate samples in each mini-batch data, and the summation of 50 coreset weights is equal to 1.00. Top-$K$ summation of weights increases as $\lambda$ increases, which imposes higher probabilities on top-$K$ entries and lower probabilities on the rest candidates. The best performance is achieved when $\lambda = 0.1$, which means $\lambda$ balances the trade-off between the loss function and regularizer strength: if $\lambda$ is too large, the algorithm primarily focuses on choosing the important samples instead of updating the model parameter, and vice versa.

Table 10: Ablation study for the regularize coefficient $\lambda$.

<table>
<thead>
<tr>
<th>Measure</th>
<th>$\lambda=0.01$</th>
<th>$\lambda=0.05$</th>
<th>$\lambda=0.10$</th>
<th>$\lambda=0.50$</th>
<th>$\lambda=1.00$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_T$</td>
<td>59.37±0.35</td>
<td>60.23±0.43</td>
<td>61.60±0.14</td>
<td>59.42±1.45</td>
<td>58.89±1.64</td>
</tr>
<tr>
<td>$F GT_T$</td>
<td>0.095±0.098</td>
<td>0.074±0.054</td>
<td>0.051±0.015</td>
<td>0.138±0.075</td>
<td>0.128±0.076</td>
</tr>
</tbody>
</table>

Table 11: Top-$K$ summation/Total summation of coreset weights ($K = 10$).

<table>
<thead>
<tr>
<th>Measure</th>
<th>$\lambda=0.01$</th>
<th>$\lambda=0.05$</th>
<th>$\lambda=0.10$</th>
<th>$\lambda=0.50$</th>
<th>$\lambda=1.00$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top-$K$ Sum/Total Sum</td>
<td>0.41/1.00</td>
<td>0.56/1.00</td>
<td>0.63/1.00</td>
<td>0.73/1.00</td>
<td>0.84/1.00</td>
</tr>
</tbody>
</table>

I Hypergradient Evolution

To demonstrate the efficiency of bilevel optimization, we illustrate the evolution of hypergradients for a bilevel optimization on Split CIFAR-100 in Figure 6. We observe the average norm of hypergradient reduces from $10^{-1}$ to less than $10^{-2}$ in each round of coreset selection with loops equal to 10. The hypergradient curves show that our designed bilevel optimization provides both theoretical and practical convergence guarantees.

Table 12: The effect of coreset size

<table>
<thead>
<tr>
<th>Methods</th>
<th>$K=10$</th>
<th>$K=20$</th>
<th>$K=40$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_T$</td>
<td>58.99±0.54</td>
<td>53.57±2.93</td>
<td>53.03±1.97</td>
</tr>
<tr>
<td>Greedy Coreset</td>
<td>59.39±0.16</td>
<td>56.81±3.32</td>
<td>56.09±0.42</td>
</tr>
<tr>
<td>PBCS</td>
<td>55.64±2.26</td>
<td>49.84±1.76</td>
<td>40.95±0.32</td>
</tr>
<tr>
<td>OCS</td>
<td>52.57±0.37</td>
<td>54.87±0.58</td>
<td>56.46±0.07</td>
</tr>
<tr>
<td>BCSR</td>
<td>61.60±0.14</td>
<td>59.06±1.15</td>
<td>56.58±0.21</td>
</tr>
</tbody>
</table>

J The Effect of Coreset Size $K$

Coreset size $K$ also plays an important role in the experiment. We compare our BCSR with other coreset-based algorithms on different $K$. Note that coreset is selected from the current stream mini-batch $B_t$, so the coreset size $K$ satisfies $K \leq |B_t|$. In the main experiment results, $K = 10$ is fixed in all the algorithms for a fair comparison. Here, we set $K = 10, 20, 40$ to conduct the continual learning experiments, respectively. The result is represented in Table 12. Note that all the results presented here are based on balanced Split CIFAR-100.

We mainly compare with the other four methods, including Uniform Sampling and three coreset-based methods, Greedy Coreset, PBCS, and OCS. We can observe that the performance of almost all methods becomes worse when coreset size is large. The reason is that if coreset size is larger and closer to $B_t$, more redundant or noisy data are selected from the current stream mini-batch, and the coreset would not be representative anymore. In contrast, the smaller coreset could reduce the probability that redundant data are selected. Compared with other methods, BCSR shows the best performance (both accuracy and forgetting) and better robustness on different $K$. 

22
K The Effect of Inner Loops $N$ and Hessian-inverse-vector Product Loops $Q$

We conduct ablation studies to explore the sensitivity of hyperparameters ($N$ and $Q$) on Split CIFAR-100. The results are presented in Tables 13 ($N$) and Table 14 ($Q$). The model performance remains relatively stable when increasing inner loops ($N$) while fixing $Q$. But too large $N$ ($N \geq 15$) leads to performance degradation due to overfitting. The $Q$ loops show similar properties that a few $Q$ loops (e.g., $Q = 3$) are enough to approximate the Hessian-inverse-vector product. Too small $Q$ and too large $Q$ will hurt the performance due to possible underfitting (e.g., $Q = 1$) and overfitting (e.g., $Q = 20$).

Table 13: Ablation study for the inner loops ($N$) with fixed $Q = 3$.

<table>
<thead>
<tr>
<th>Measure</th>
<th>$N=1$</th>
<th>$N=5$</th>
<th>$N=10$</th>
<th>$N=15$</th>
<th>$N=20$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_T$</td>
<td>61.60±0.14</td>
<td>61.75±0.11</td>
<td>61.64±0.15</td>
<td>60.77±0.32</td>
<td>59.20±0.41</td>
</tr>
<tr>
<td>$FGT_T$</td>
<td>0.051±0.015</td>
<td><strong>0.047±0.013</strong></td>
<td>0.063±0.017</td>
<td>0.074±0.021</td>
<td>0.079±0.035</td>
</tr>
</tbody>
</table>

Table 14: Ablation study for the loops $Q$ with fixed $N = 1$.

<table>
<thead>
<tr>
<th>Measure</th>
<th>$Q=1$</th>
<th>$Q=3$</th>
<th>$Q=5$</th>
<th>$Q=10$</th>
<th>$Q=20$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_T$</td>
<td>52.14±1.53</td>
<td><strong>61.60±0.14</strong></td>
<td>61.57±0.15</td>
<td>58.42±0.53</td>
<td>57.80±1.31</td>
</tr>
<tr>
<td>$FGT_T$</td>
<td>0.123±0.038</td>
<td><strong>0.051±0.015</strong></td>
<td>0.064±0.012</td>
<td>0.173±0.045</td>
<td>0.162±0.041</td>
</tr>
</tbody>
</table>

L Datasets

- **Split CIFAR-100.** Balanced split CIFAR-100 is based on the original CIFAR-100 and is split into 20 tasks, each consisting of 5 disjoint classes. The imbalanced setting and label noise are also applied to this dataset to make the task more challenging. We follow [13] to transform the original dataset to imbalanced long-tailed CIFAR-100. In the label-noise setting, we randomly select 20% data in each task and randomly change their labels to an arbitrary label of 10 classes.

- **Permuted MNIST.** Balanced MNIST is a handwritten digits dataset [43] containing 20 tasks, where each task applies a fixed random permutation to the image pixels. For the imbalanced setting, we randomly select 8 classes over 10 and sample 10% of the selected classes for training. We also conduct the experiments on the label noise scenario, where symmetric label noise with 20% noise rate is imposed on the data. In particular, to make the problem setting more challenging, each task only retains 3000 training data randomly sampled from the original data.

- **Multiple Datasets.** This dataset [78] contains a couple of totally different datasets, including MNIST [43], fashion-MNIST [75], NotMNIST [8], Traffic Sign [69] and SVHN [54]. There are 5 tasks, and each task is constructed by randomly selecting 1000 training samples from a different dataset. The procedure of creating the dataset in the imbalanced and label-noise settings is the same as that in Split CIFAR-100.

- **Split Tiny-ImageNet.** This dataset [42] contains 100000 images of 200 classes (500 for each class) downsized to $64 \times 64$ colored images. Each class has 500 training images, 50 validation images, and 50 test images. We construct the task sequences by splitting data into 20 tasks, where each task consists of 10 disjoint classes.

- **Split Food-101.** This dataset [7] is a challenging data set of 101 food categories, with 101'000 images. All the images are resized to $64 \times 64$ pixels to be fed into models. To build continual learning tasks easily, We discard the last category and split data into 20 tasks, with 5 categories within each task. Other data settings, including imbalanced and lable-noise, are the same as Split CIFAR-100.

M Assumptions and Properties

We first provide standard definitions and assumptions for the convergence rate analysis of bilevel optimization [25][56]. For notational convenience, we define $\ell(\theta) := \sum_{i=1}^{n} \ell_i(\theta)$. 

23
**Definition 1.** A mapping $f$ is $L_f$-Lipschitz continuous if for all $z, z'$, $\| f(z) - f(z') \| \leq L_f \| z - z' \|$. 

**Assumption 1.** The lower-level function $L(\theta, w)$ is $\mu$-strongly-convex w.r.t. $\theta$. 

Assumption 1 is a necessary geometric assumption in analyzing the convergence rate of bilevel optimization algorithms, as also widely adopted existing theories in [25][37][32]. We also note that this condition is satisfied for overparameterized neural networks [82]. The following assumption imposes some Lipschitz continuity conditions on the upper- and lower-level objective functions. 

**Assumption 2.** The gradients $\nabla_{\theta} L(\theta, w)$, $\nabla_{w} L(\theta, w)$ and $\ell(\theta)$ are $L$-Lipschitz continuous w.r.t. $\theta$ and $w$. In addition, the gradient norm $\| \nabla \ell(\theta^*(w)) \| \leq M$. 

Note that we do not impose any conditions on the regularization function $R(w, \delta)$. The following assumption imposes the Lipschitz continuity on the second-order derivatives of the lower-level functions. 

**Assumption 3.** The second-order derivatives $\nabla_{w} \nabla_{\theta} L(\theta, w)$ and $\nabla_{\theta}^{2} L(w, \theta)$ are $\rho$-Lipschitz continuous. 

Then, we use the following proposition to characterize the properties of the smoothed top-$K$ regularizer $R(w, \delta)$, based on the results in [23].

**Proposition 1.** The smoothed regularizer $R(w, \delta)$ and its sampled version $R(w, \delta; \bar{z})$ satisfy the following two important properties: (i) The gradient $\nabla R(w, \delta)$ exists and is $\sqrt{K} / \delta$-Lipschitz continuous; (ii) The gradient norm $\| R(w, \delta; \bar{z}) \|$ is bounded by $\sqrt{K}$ for any sample $\bar{z}$. 

Proposition 1 shows that the regularizer $R(w, \delta)$ is smooth and its stochastic version $R(w, \delta; \bar{z})$ is bounded. These two properties are important to guarantee the non-asymptotic convergence of our proposed bilevel method.