Computationally Budgeted Continual Learning: What Does Matter?

Ameya Prabhu[1*] Hasan Abed Al Kader Hammoud[2*] Puneet Dokania[1] Philip H.S. Torr[1]
Ser-Nam Lim[3] Bernard Ghanem[2] Adel Bibi[1]
[1]University of Oxford [2]King Abdullah University of Science and Technology (KAUST) [3]Meta AI

Abstract

Continual Learning (CL) aims to sequentially train models on streams of incoming data that vary in distribution by preserving previous knowledge while adapting to new data. Current CL literature focuses on restricted access to previously seen data, while imposing no constraints on the computational budget for training. This is unreasonable for applications in-the-wild, where systems are primarily constrained by computational and time budgets, not storage. We revisit this problem with a large-scale benchmark and analyze the performance of traditional CL approaches in a compute-constrained setting, where effective memory samples used in training can be implicitly restricted as a consequence of limited computation. We conduct experiments evaluating various CL sampling strategies, distillation losses, and partial fine-tuning on two large-scale datasets, namely ImageNet2K and Continual Google Landmarks V2 in data incremental, class incremental, and time incremental settings. Through extensive experiments amounting to a total of over 1500 GPU-hours, we find that, under compute-constrained setting, traditional CL approaches, with no exception, fail to outperform a simple minimal baseline that samples uniformly from memory. Our conclusions are consistent in a different number of stream time steps, e.g.*, 20 to 200, and under several computational budgets. This suggests that most existing CL methods are particularly too computationally expensive for realistic budgeted deployment.*

1. Introduction

Deep learning has excelled in various computer vision tasks [8,24,28,47] by performing hundreds of shuffled passes through well-curated offline static labeled datasets. However, modern real-world systems, *e.g.*, Instagram, TikTok, and Flickr, experience high throughput of a constantly changing stream of data, which poses a challenge for deep learning to cope with such a setting. Continual learning (CL) aims to go beyond static datasets and develop learning strategies that can adapt and learn from streams where data is presented incrementally over time, often referred to as time steps. However, the current CL literature overlooks a key necessity for practical real deployment of such algorithms. In particular, most prior art is focused on *offline continual learning* [25,26,44] where, despite limited access to previous stream data, training algorithms do not have restrictions on the computational training budget per time step.

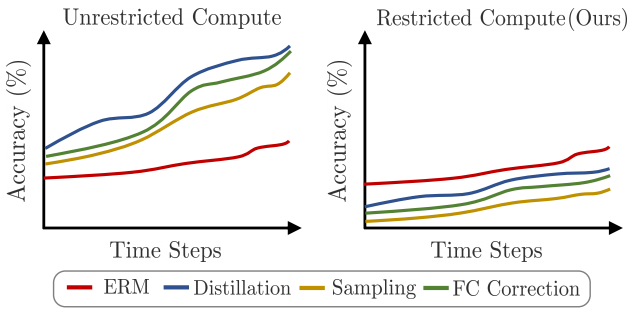

Figure 1. **Main Findings.** Under per time step computationally budgeted continual learning, classical continual learning methods, *e.g.*, sampling strategies, distillation losses, and fully connected (FC) layer correction based methods such as calibration, struggle to cope with such a setting. Most proposed continual algorithms are particularly useful only when large computation is available, where, otherwise, minimalistic algorithms (ERM) are superior.

High-throughput streams, *e.g.*, Instagram, where every stream sample at every time step needs to be classified for, say, misinformation or hate speech, are time-sensitive in which long training times before deployment are simply not an option. Otherwise, new stream data will accumulate until training is completed, causing server delays and worsening user experience.

Moreover, limiting the computational budget is necessary towards reducing the overall cost. This is because computational costs are higher compared to any storage associated costs. For example, on Google Cloud Standard Storage (2¢ per GB per month), it costs no more than 6¢ to store the entire CLEAR benchmark [29], a recent large-scale CL dataset. On the contrary, one run of a CL algorithm on CLEAR performing $\sim$ 300K iterations costs around 100$ on an A100 Google instance (3$ per hour for 1 GPU). Therefore, it is prudent to have computationally budgeted methods where the memory size, as a consequence, is implicitly restricted. This is because, under a computational budget, it is no longer possible to revisit all previous data even if they

*authors contributed equally; order decided by a coin flip.

were all stored in memory (given their low memory costs).

This raises the question: "*Do existing continual learning algorithms perform well under per step restricted computation?*" To address this question, we exhaustively study continual learning systems, analyzing the effect of the primary directions of progress proposed in the literature in the setting where algorithms are permitted fixed computational budget per stream time step. We evaluate and benchmark at scale various classical CL sampling strategies (Uniform, Class-Balanced [40], Recency-Biased [29], FIFO [11, 15], Max Loss, Uncertainity Loss [6], and KMeans [15]), CL distillation strategies (BCE [44], MSE [9], CrossEntropy [53], and Cosine [25]) and FC layer corrections (ACE [10, 34, 58], BiC [53], CosFC [25], and WA [60]) that are common in the literature. Evaluation is carried on two large-scale datasets, amounting to a total of 1500 GPU-hours, namely ImageNet [18] and Continual Google Landmarks V2 [39] (CGLM) under various stream settings, namely, data incremental, class incremental, and time incremental settings. We compare against Naive; a simple baseline that, utilizing all the per step computational budget, trains while sampling from previous memory samples.

Conclusions. We summarize our empirical conclusions in three folds. **(1)** None of the proposed CL algorithms, see Table 1 for considered methods, can outperform our simple baseline when computation is restricted. **(2)** The gap between existing CL algorithms and our baseline becomes larger with harsher compute restrictions. **(3)** We find that training a minimal subset of the model can close the performance gap compared to our baseline in our setting, but only when supported by strong pretrained models.

Surprisingly, we find that these observations hold even when the number of time steps is increased to 200, a large increase compared to current benchmarks, while normalizing the effective total computation accordingly. This suggests that existing CL literature is particularly suited for settings where memory is limited, and less practical in scenarios having limited computational budgets.

2. Continual Learning with Limited Compute

2.1. Problem Formulation

We start by first defining our proposed setting of *computationally budgeted* continual learning. Let $\mathcal{S}$ be a stream revealing data sequentially over time steps. At each time step $t \in \{1, 2, \dots, \infty\}$, the stream $\mathcal{S}$ reveals n_t image-label pairs $\{(x_i^t, y_i^t)\}_{i=1}^{n_t} \sim \mathcal{D}_j$ from distribution $\mathcal{D}_j$ where $j \in \{1, \dots, t\}$. In this setting, we seek to learn a function $f_{\theta_t} : \mathcal{X} \rightarrow \mathcal{Y}_t$ parameterized by θ_t that maps images $x \in \mathcal{X}$ to class labels $y \in \mathcal{Y}_t$, where $\mathcal{Y}_t = \bigcup_{i=1}^{t} \mathcal{Y}_i$, which aims to correctly classify samples from any of the previous distributions $\mathcal{D}_{j \leq t}$. In general, there are no constraints on the incoming distribution $\mathcal{D}_j$, *e.g.*, the distribution might change after every time step or it may stay unchanged for all time steps. The size of the revealed stream data n_t can generally change per step, *e.g.*, the rate at which users upload data to a server. The unique aspect about our setting is that at *every time step t, a computational budget* $\mathcal{C}_t$ is available for the CL method to update the parameters from θ_{t-1} to θ_t in light of the new revealed data. Due to the inexpensive costs associated with memory storage, in our setting, we assume that CL methods in our setting can have full access to all previous samples $\mathcal{T}_t = \cup_{r=1}^{t} \{(x_i^r, y_i^r)\}_{i=1}^{n_r}$.[1] However, as will be discussed later, while all samples can be stored, they cannot all be used for training due to the constrained computation imposing an implicit memory restriction.

2.2. Key Differences with Prior Art

(1) *Tasks*: In most prior work, CL is simplified to the problem of learning a set of non-overlapping tasks, *i.e.*, distributions, *with known boundaries* between them [13, 32, 44]. In particular, the data of a given distribution $\mathcal{D}_j$ is given all at once for the model to train. This is as opposed to our setup, where there is no knowledge about the distribution boundaries, since they are often gradually changing and not known a priori. As such, continual learning methods cannot train only just before the distribution changes.

(2) *Computational Budget*: A key feature of our work is that, per time step, CL methods are given a fixed computational budget $\mathcal{C}_t$ to train on $\{(x_i^t, y_i^t)\}_{i=1}^{n_t}$. For ease, we assume throughout that $\mathcal{C}_t = \mathcal{C}\ \forall t$, and that $n_t = n \forall t$. Although $\mathcal{C}$ can be represented in terms of wall clock training time, for a given f_θ and stream $\mathcal{S}$, and comparability between GPUs, we state $\mathcal{C}$ in terms of the number of training iterations instead. This avoids hardware dependency or suboptimal implementations when comparing methods. This is unlike prior work, which do not put hard constraints on compute per step [9, 25, 44] giving rise to degenerate but well-performing algorithms such as GDumb [40]. Concurrent works [20, 39] restrict the computational budget, however they operate in an online continual learning scenario where the focus is on rapid adaptation.

(3) *Memory Constraints*: Prior work focuses on a fixed, small memory buffer for learning and thereof proposing various memory update strategies to select samples from the stream. We assume that all the samples seen so far can be stored at little cost. However, given the restricted imposed computation $\mathcal{C}$, CL methods cannot revisit or learn from all stored samples. For example, as shown in Figure 2, consider performing continual learning on ImageNet2K, composed of 1.2M samples from ImageNet1K and 1.2M samples from ImageNet21K forming 2K classes, which will be detailed later, over 20 time steps, where the stream reveals sequentially $n = 60$K images per step. Then, under a computation budget of 8000 iterations, the model cannot revisit more than 50% of all seen data at any given time step, *i.e.* 600K samples. Our proposed setting is closer to realistic scenarios that cope with high-throughput streams, where *computational*

[1] We discuss in the Appendix the privacy arguments often used towards restricting the memory.

Dir.	Reference	Applicability (our setup)	Components: Distillation	MemUpdate	MemRetrieve	FC Correction	Others
	Naive	✓	-	Random	Random	-	-
Distillation	iCARL [44]	✓	**BCE**	**Herding**	Random	-	**NCM**
	LUCIR [25]	✓	**Cosine**	Herding	MargRank	**CosFC**	NCM
	PODNet [19]	✓	**POD**	Herding	Random	**LSC**	Imprint,NCM
	DER [9]	✓	**MSE**	Reservoir	Random	-	-
	CO^2L [12]	×	**IRD**	Random	Random	**Asym.SupCon**	-
	SCR [35]	✓	-	Reservoir	Random	**SupCon**	NCM
Sampling	TinyER [15]	✓	-	**FIFO,KMeans,Reservoir**	-	-	-
	GSS [5]	×	-	**GSS**	Random	-	-
	MIR [3]	×	-	Reservoir	**MIR**	-	-
	GDumb [40]	✓	-	**Balanced**	Random	-	**MemOnly**
	Mnemonics [31]	×	-	**Mnemonics**	-	-	BalFineTune
	OCS [57]	×	-	**OCS**	Random	-	-
	InfoRS [49]	×	MSE	**InfoRS**	Random	-	-
	RMM [30]	×	-	**RMM**	-	-	-
	ASER [48]	×	-	**SV**	**ASV**	-	-
	RM [6]	✓	-	**Uncertainty**	Random	-	AutoDA
	CLIB [27]	×	-	**Max Loss**	Random	-	MemOnly,AdaLR
FC Layer	BiC [53]	×	CrossEnt	Random	Random	**BiC**	-
	WA [60]	×	CrossEnt	Random	Random	**WA**	-
	SS-IL [2]	×	TKD	Random	Balanced	**SS**	-
	CoPE [17]	✓	-	Balanced	Random	**PPPLoss**	-
	ACE [10]	✓	-	Reservoir	Random	**ACE**	-

Table 1. **Primary Directions of Progress in CL.** Analysis of recent replay-based systems, with **bold** highlighting the primary contribution. We observe that there are three primary directions of improvement. "App." denotes the applicability to our setting based on whether they are scalable to large datasets and applicable beyond the class-incremental stream.

bottlenecks impose implicit constraints on learning from past samples that can be too many to be revisited during training.

2.3. Constructing the Stream

We explore three stream settings in our proposed benchmark, which we now describe in detail.

(**1**) *Data Incremental Stream*: In this setting, there is no restriction on the incoming distribution $\mathcal{D}_j$ over time that has not been well-explored in prior works. We randomly shuffle all data and then reveal it sequentially over steps, which could lead to a varying distribution $\mathcal{D}_j$ over steps in which there are no clear distribution boundaries.

(**2**) *Time Incremental Stream*: In this setting, the stream data is ordered by the upload timestamp to a server, reflecting a natural distribution change $\mathcal{D}_j$ across the stream as it would in real scenarios. There is a recent shift toward studying this ordering as apparent in recent CL benchmarks, *e.g.*, CLEAR [29], Yearbook [55] and FMoW [55], Continual YFCC100M [11] and Continual Google Landmarks V2 [39].

(**3**) *Class Incremental Stream*: For completeness, we consider this classical setting in the CL literature. Each of the distributions $\mathcal{D}_j$ represents images belonging to a set of classes different from the classes of images in any other distribution $\mathcal{D}_{i\neq j}$. We benchmark these three settings using a large-scale dataset that will be detailed in the Experiments.

3. Dissecting Continual Learning Systems

Continual learning methods typically propose a system of multiple components that jointly help improve learning performance. For example, LUCIR [25] is composed of a cosine linear layer, a cosine distillation loss function, and a hard-negative mining memory-based selector. In this section, we analyze continual learning systems and dissect them into their underlying components. This helps to analyze and isolate the role of different components under our budgeted computation setting and helps us to understand the most relevant components. In Table 1, we present the breakdown of novel contributions that have been the focus of recent progress in CL. The columns indicate the major directions of change in the CL literature. Overall, there have been three major components on which advances have focused, namely distillation, sampling, and FC layer correction. These three components are considered additions to a naive baseline that simply performs uniform sampling from memory. We refer to this baseline as Naive in Table 1.

(**1**) *Distillation*: One popular approach towards preserving model performance on previous distributions has been through distillation. It enables student models, *i.e.*, current time step model, to learn from a teacher model, *i.e.*, one that has been training for many time steps, through the logits providing a rich signal. In this paper, we consider four widely adopted distillation losses, namely, Binary CrossEntropy (BCE) [44], CrossEntropy [31, 53, 60], Cosine Similarity (Cosine) [25], and Mean Square Error (MSE) [9, 49] Loss.

(**2**) *Sampling*: Rehearsing samples from previous distributions is another popular approach in CL. However, sampling strategies have been used for two objectives. Particularly when access to previous samples is restricted to a small memory, they are used to select which samples from the stream will update the memory (MemUpdate) or to decide on which memory samples are retrieved for rehearsal (MemRetrieve). In our unconstrained memory setup, simply sampling uniformly over the joint data of past and current time step data (as in Naive) exposes a particular shortcoming. When training for a large number of time steps, uniform sampling reduces the probability of selecting samples from the current time step. For that, we consider various sampling strategies, *e.g.*, recency sampling [29] that biases toward sampling current time step data, and FIFO [11, 15] that exclusively samples from the current step. We do not consider Reservoir, since it approximates uniform sampling in our setup with no memory restrictions. In addition to bench-

marking the sampling strategies mentioned above, we also consider approaches that evaluate the contribution of each memory sample to learning [50]. For example, Herding [44], K-Means [15], OCS [57], InfoRS [49], RM [6], and GSS [5] aim to maximize diversity among samples selected for training with different metrics. MIR [3], ASER [48], and CLIB [27] rank the samples according to their informativeness and select the top-k. Lastly, balanced sampling [16, 17, 40] select samples such that an equal distribution of classes is selected for training. In our experiments, we only consider previous sampling strategies that are applicable to our setup and compare them against Naive.

(**3**) *FC Layer Correction*: It has been hypothesized that the large difference in the magnitudes of the weights associated with different classes in the last fully connected (FC) layer is among the key reasons behind catastrophic forgetting [53]. There has been a family of different methods addressing this problem.

These include methods that improve the design of FC layers, such as CosFC [25], LSC [19], and PPP [17], by making the predictions independent of their magnitude. Other approaches such as SS-IL [2] and ACE [10, 34, 58] mask out unaffected classes to reduce their interference during training. In addition, calibrating the FC layer in post-training, *e.g.*, BiC [53], WA [60], and IL2M [7] is widely used. Note that the calibration techniques are only applicable to the class-incremental setup. We benchmark existing methods applicable to our setting against the Naive approach that does not implement any FC layer correction.

(**4**) *Model Expansion Methods*: Several works attempt to adapt the model architecture according to the data. This is done by only training part of the model [1, 4, 36, 37, 41] or by directly expanding the model when data is presented [43, 46, 51, 54, 56, 59]. However, most of the previous techniques in this area do not apply to our setup. Most of this line of work [36, 37, 43] assumes a task-incremental setting, where at every time step, new samples are known to what set of classes they belong, *i.e.*, the distribution boundaries are known, even at test time. To overcome these limitations, newer methods [1, 42] use a bilevel prediction structure, predicting the task at one level and the label within the task at the second level. They are restricted to the class-incremental setting as they assume each task corresponds to a set of non-overlapping classes. We seek to understand the limitation of partial retraining in a network; hence, instead, we compare Naive against a setting where only the FC layer is being trained, thus minimally training the network per time step. In addition, we examine the role of pretraining which has recently become a widely popular direction for exploration in continual learning [52].

4. Experiments

We first start by detailing the experimental setup, datasets, computational budget $\mathcal{C}$, and evaluation metrics for our large-scale benchmark. We then present the main results evaluating various CL components, followed by extensive analysis.

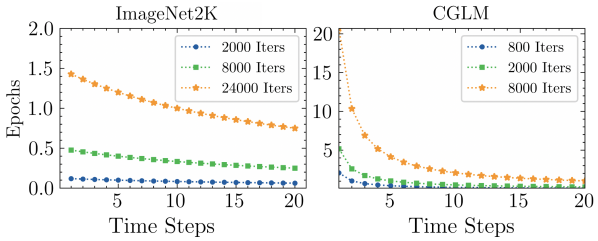

Figure 2. **Effective Training Epochs Per Time Step.** Our default setting sets a total training budget over all 20 time steps of 8000 and 2000 iterations for ImageNet2K and CGLM ,respectively, with a per iteration batch size of $\mathcal{B} = 1500$. Effectively, this reflects to training on 25-50% of the stored data, except in the first few time steps on CGLM. Note that for ImageNet2K, we assume that ImageNet1K of 1.2M samples is available in memory.

4.1. Experimental Setup and Details

Model. We use a standard ResNet50 following prior work on continual learning [11]. The model is ImageNet1K pre-trained used as a backbone throughout all experiments.

Datasets. We conduct experiments using two large-scale datasets, namely ImageNet2K and Continual Google Landmarks V2 (CGLM). We construct ImageNet2K by augmenting ImageNet1K with 1.2M images from ImageNet21K [18], thus, adding 1K new non-overlapping classes with ImageNet1K amounting to a total of 2K classes.

(**1**) *Data Incremental ImageNet2K*: The stream is constructed by randomly shuffling the set of images from the 1K classes of ImageNet21K, by doing so, there is no knowledge of the distribution boundaries. The model continually learns on this set of images, while ImageNet1K is available in memory. CL methods are expected to learn both the new classes from the stream while maintaining the performance on ImageNet1K. We refer to this setting as *DI-ImageNet2K*.

(**2**) *Class Incremental ImageNet2K*: Similar to the above defined DI-ImageNet2K, ImageNet1K is available in memory and the 1K classes of ImageNet21K are presented sequentially by the stream but in a class incremental setting. We refer to this setting as *CI-ImageNet2K*.

(**3**) *Time Incremental Google Landmarks V2 (CGLM)*: In this setting, the stream consists of data from the CGLM dataset ordered according to the timestamps of the images mimicking a natural distribution shift. Note that ImageNet1K is not considered as part of the evaluation. We refer to this setting simply as *CGLM*.

Throughout, unless stated otherwise, the stream reveals data incrementally over 20 time steps. This amounts to a per step stream size of $n = 60$K for the CI-ImageNet2K and DI-ImageNet2K settings, and $n = 29$K for the CGLM setting. More details on the construction of datasets is given in the Appendix along with the sample orders.

Computational Budget. We set the computational budget $\mathcal{C}$ to 400 training iterations per time step ($8000 = 20$ time steps $\times$ 400) for ImageNet2K, *i.e.*, DI-ImageNet2K

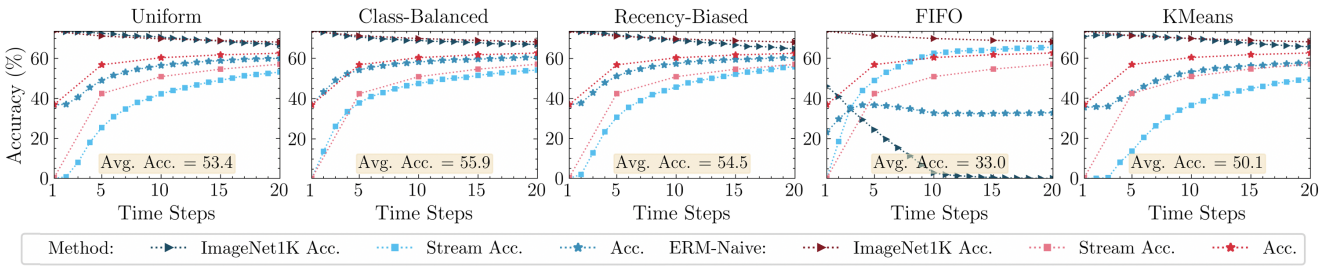

Figure 3. **DI-ImageNet2K** (400 **Iterations).** ERM-Naive, a non-continual learning algorithm, is compared against inexpensive sampling strategies (first four plots) with 400 training iterations and the costly KMeans (fifth plot) with 200 iterations. All CL methods perform similarly but worse than ERM-Naive. This is the case for FIFO that suffers from forgetting the KMeans due to its expensive nature. ImageNet2K experiments performance can be decomposed into (i) accuracy on classes seen during pre-training on ImageNet1K and (ii) accuracy on newly seen classes in ImageNet2K, allowing analysis of forgetting old classes and learning newly introduced classes.

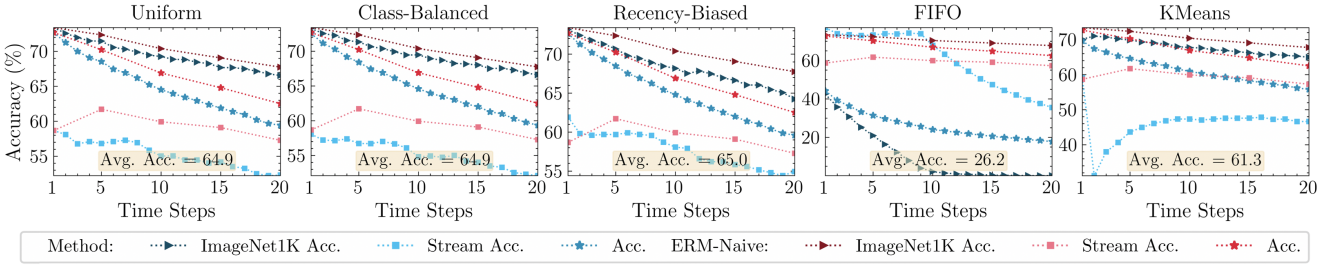

Figure 4. **CI-ImageNet2K** (400 **Iterations).** Similarly, ERM-Naive is compared on CI-ImageNet2K. Both FIFO, which suffers from forgetting, and KMeans due to its expensive nature, struggle to compete against simpler inexpensive methods as Class-Balanced. However, overall, all other methods perform very similarly with no clear advantage. ImageNet2K experiments performance can be decomposed into (i) accuracy on classes seen during pre-training on ImageNet1K and (ii) accuracy on newly seen classes in ImageNet2K, allowing analysis of forgetting old classes and learning newly introduced classes.

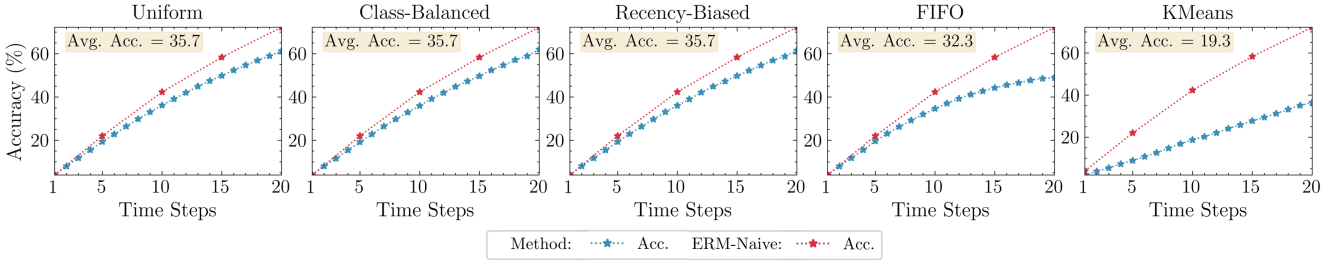

Figure 5. **CGLM** (100 **Iterations).** All inexpensive methods perform overall similarly with the exception for KMeans due to its expensive nature. This highlights that simplicity is key under a budgeted continual learning setting. CGLM is not an extension of ImageNet1K and involves a different task: landmark classification. Hence, we measure only the stream accuracy resulting in two lines instead of six.

and CI-ImageNet2K, and set $\mathcal{C}$ to 100 training iterations for CGLM. In each iteration, a batch of images is used to update the model in training where we set the training batch size $\mathcal{B}$ to 1500. The choice of $\mathcal{C}$ is made such that it corresponds to training on at most $25-50\%$ of all observed data at any given step. For example, as highlighted in Figure 2 for ImageNet2K, time step $t=5$. corresponds to training only on about 40% of the complete observed data at this step, *i.e.*, ${}^{400\times1500}/_{1.2M+5\times60K} \approx 0.4$ of an epoch where 1.2M denotes the ImageNet1K samples. Furthermore, we set $\mathcal{C}$ to 100 iterations for CGLM, since the dataset contains ¼ of the total data in ImageNet2K. Note that after 20 time steps on CGLM, the data that would have been seen is $20\times29K$ images, as opposed to $1.2M+20\times60K$ images for ImageNet2K experiments.

Metrics. We report the accuracy (Acc) on a separate test set after training at each time step. This test set simply comprises the joint test set for all classes seen up to the current time step. Moreover, for ImageNet2K, we decompose the test accuracy into the accuracy on ImageNet1K (ImageNet1K Acc), which measures forgetting, and the accuracy on the stream (Stream Acc), which measures adaptation. For GCLM, we only report stream accuracy.

Training Details. We use SGD as an optimizer with a linear learning rate schedule and a weight decay of 0. We follow standard augmentation techniques. All experiments were run on the same A100 GPU. For a fair comparison, we fix the order of the samples revealed by the stream $\mathcal{S}$ in all experiments on a given dataset and comparisons.

We summarize all the settings with all the benchmark parameters in the first part of Table 2.

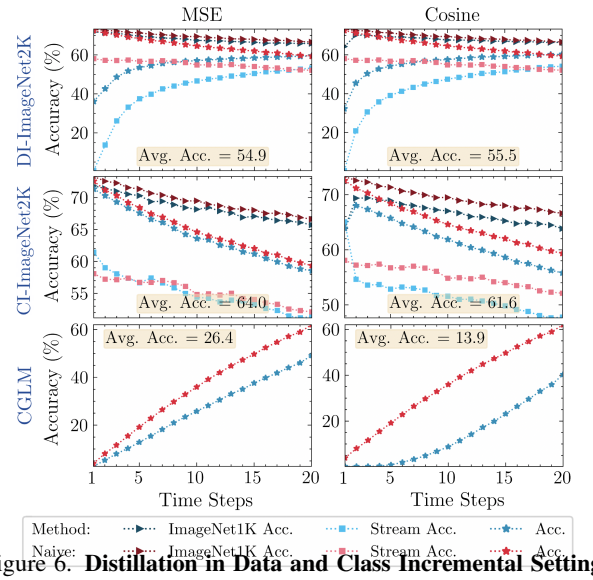

Figure 6. **Distillation in Data and Class Incremental Settings.** Naive, which does not employ any distillation loss, outperforms all distillation methods (MSE, and Cosine) across all three settings.

4.2. Budgeted Continual Learning

In this section, we investigate the effectiveness of the three main directions studied in the CL literature, namely sampling strategies, distillation, and FC layer correction.

1. Do Sampling Strategies Matter? We evaluate seven sampling strategies that govern the construction of the training batch from memory. These strategies are grouped into two categories based on their computational cost. Inexpensive sampling methods include Uniform, Class-Balanced, Recency-Biased and FIFO sampling. On the other hand, costly sampling strategies include KMeans, Max Loss, and Uncertainty loss sampling.

To normalize for the effective $\mathcal{C}$ due to the overhead of associated extra forward passes to decide on the sampling, costly sampling strategies are allowed $\mathcal{C}/2$ training iterations, where the exact calculation is left for the Appendix. That is to say, costly sampling strategies perform 200 training iterations for ImageNet2K and 50 training iterations for CGLM as the rest of the budget is for the extra forward passes. We report the performance of the five sampling strategies consisting of the inexpensive and the best performing costly sampling strategy (KMeans), presented in shades of blue, in Figures 3, 4, and 5 for DI-ImageNet2K, CI-ImageNet2K, and CGLM, respectively. Other methods are listed in the Appendix due to lack of space. We compare against a non-continual learning oracle that performs classical empirical risk minimization at every step on $\mathcal{T}_t = \cup_{r=1}^{t}\{(x_i^r, y_i^r)\}_{i=1}^{n_r}$ with a computational budget of $\mathcal{C} \times t$, which we refer to as ERM-Naive; this is as opposed to the previously mentioned continual learning methods that have only $\mathcal{C}$ per step t spent equally over all steps. ERM-Naive acts as a training method with hindsight, spending the complete computational budget at once after collecting the full dataset. This acts as a very strong baseline against all continual learning methods. We report it in shades of red in the same figures. We also report the average accuracy, averaged over all time steps, for each sampling method in the yellow box in each figure.

Attributes	ImageNet2K	CGLM
Initial memory	ImageNet1K	{}
Initial memory size	1.2M	0
Per step stream size n	60K	29K
Time steps	20	20
Stream size	1.2M	58K
Size of data by the last time step	2.4M	58K
Stream	Class incremental Data incremental	Time incremental
# iterations per time step $\mathcal{C}$	400	100
Training batch size $\mathcal{B}$	1500	1500
Metrics	Acc on ImageNet1K Acc on Stream	Acc on Stream
Eq. Distillation Iters	267	67
Eq. Sampling Iters	200	100
Eq. FC Correction Iters	400	100
Iters per t (Sensitivity)	100, 1200	40, 400
Time Steps (Sensitivity)	50, 200	50, 200

Table 2. **Experimental Details.** The first block shows the various considered settings in the experiments section. The second block denotes the effective training iterations $\mathcal{C}$ for each class of methods due to their over head extra computation. The last block details the setup for our sensitivity analysis.

Conclusion. First, we observe that the top inexpensive sampling strategies perform very similarly to each other. This is consistent across settings, CI-ImageNet2K, and CGLM, on both ImageNet1K accuracy and Stream Accuracy. There are some advantages for Class-Balanced over other sampling strategies, *e.g.*, gaining an average accuracy of 2.5% over Uniform in DI-ImageNet2K. However, sampling strategies such as FIFO completely forget ImageNet1K (dark blue line), leading to poor performance over all three settings. Interestingly, costly sampling strategies perform significantly worse in CL performance over the simple Uniform sampling when subjected to an effectively similar computational budget. This observation is different from previous settings [38], as the additional computational overhead of costly sampling does not seem worthwhile to improve performance.

2. Does Distillation Matter? We evaluate four well-known distillation losses in our benchmark, namely, Cosine, CrossEntropy, BCE, and MSE losses. Given that Class-Balanced is a simple inexpensive sampling procedure that performed slightly favorably, as highlighted in the previous section, we use it as a default sampling strategy from now onward, where the number of samples used per training step is equal over all classes. We refer to this basic approach with a cross entropy loss as Naive. To fairly factor in the overhead of an additional forward pass, distillation approaches are allowed $2\mathcal{C}/3$ iterations compared to Naive with $\mathcal{C}$ training iterations. That is, the distillation losses perform 267 iterations for ImageNet2k and 67 iterations for CGLM compared to 400 and 100 iterations for Naive. We report the results for Cosine and MSE on DI-ImageNet2K, CI-ImageNet2K, and CGLM datasets in the first, second, and third rows of Figure 6, respectively. Other methods are left for the Appendix due to lack of space. Distillation methods are shown in shades of blue, whereas Naive is shown in shades of red. We report

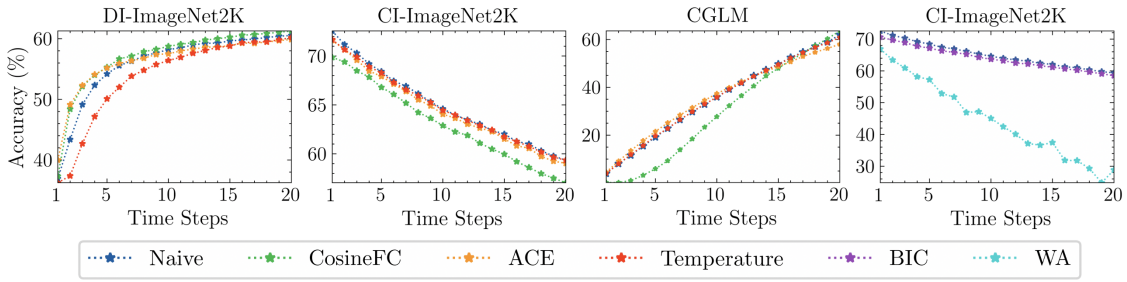

Figure 7. **FC Layer Correction.** Even though loss functions (CosineFC and ACE) might outperform Naive in the first few time steps, eventually Naive catches up. Overall, Naive consistently outperforms all considered calibration methods too, namely, BIC and WA.

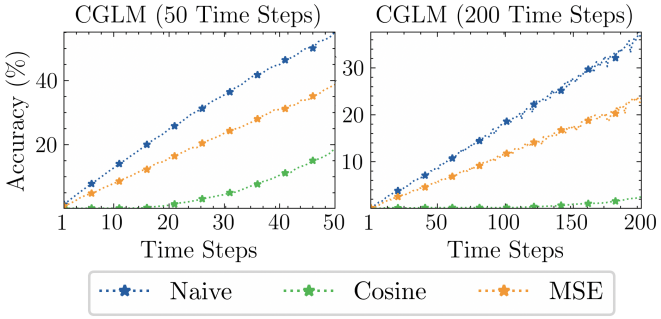

Figure 8. **CGLM Distillation with Different Number of Time Steps.** Under larger number of time steps, where total number of iterations is normalized accordingly, Naive outperforms distillation in both settings, namely, 50 and 200 time steps.

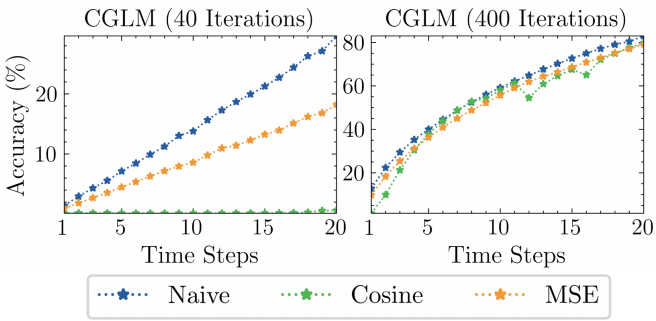

Figure 9. **CGLM Distillation with Different Computational Budgets.** Naive outperforms distillation methods under the restricted 40 and the larger 400 iterations (originally 100). Distillation methods become competitive when enough budget is available.

the average accuracy, averaged over all time steps, for each distillation method in the yellow box in each figure.

Conclusion. Overall, we notice that all distillation methods clearly underperform when compared to Naive. This is consistent over all three settings. Surprisingly, although distillation is designed to reduce forgetting, this is not the case in budgeted continual learning. The results in Figure 6 show that Naive performs similarly, or slightly better, even in ImageNet1K Acc (measures a notion of forgetting) than all distillation methods in DI-ImageNet2K and CI-ImageNet2K streams. Interestingly, we find that the top distillation methods, *e.g.* MSE, perform only slightly worse compared to Naive (54.9 vs 55.9 on DI-ImageNet2K) and (64 vs 64.9 on CI-ImageNet2K). However, in CGLM they perform significantly worse 26.4 compared to 35.7 due to the limited iterations. We attribute this to the fact that distillation methods often require a larger number of training samples, and thereof a large enough computational budget per time step.

3. Does FC Layer Correction Matter? We evaluate five FC layer correction approaches from two different families. A family of methods that modifies the FC layer directly, including CosineFC [25] and ACE [10, 34, 58]. The other family of methods applies post-training calibration including BiC [53], WA [60], along with temperature scaling [22]. All methods employ Class-Balanced as a sampling strategy and compare against Naive (Class-balanced with cross entropy loss) with no corrections in FC layer. The first three subplots of Figure 7 correspond to comparisons of direct FC layer modification methods against Naive on DI-ImageNet2K, CI-ImageNet2K, and CGLM. Since calibration methods tailored for Class Incremental settings, in the rightmost plot of Figure 7, we report comparisons with Naive on CI-ImageNet2K. Since all FC layer corrections are with virtually no extra cost, the number of training iterations per time step is set to $\mathcal{C}$, *i.e.*, 400 for ImageNet2K and 100 for CGLM.

Conclusion. Overall, there is no method that consistently outperforms Naive. While the first family of methods that help in DI-ImageNet2K, especially in the initial steps, due to the class imbalance between ImageNet1K and newer samples. However, no method outperforms Naive in the CI-ImageNet2K set-up. Similarly, for calibration based methods, we find that while BIC is somewhat competitive to Naive, where but WA fails. Surprisingly, even under various FC correction approaches, all methods fail to outperform Naive under computationally budgeted continual learning.

4.3. Sensitivity Analysis

We have analyzed the performance of various CL methods under budgeted computation. We have consistently observed over a variety of settings on large-scale datasets that a simple method, *i.e.*, Naive, simply sampling with a Class-Balanced strategy and a cross entropy loss outperforms all existing methods. However, all reported results were for 20 time steps with $\mathcal{C} = 400$ or $\mathcal{C} = 100$ training iterations for ImageNet2K and CGLM, respectively, in which expensive methods were normalized accordingly. Now, we analyze the sensitivity of our conclusions over different time steps and iterations $\mathcal{C}$.

Does the Number of Time Steps Matter? Prior art, such as GDumb [40], found that the relative performance of CL

methods changes drastically when the number of time steps is varied. Subsequently, we increased the number of time steps to 50 and 200 from 20, a more extreme setting than explored in recent works, while maintaining the same overall computational budget $\mathcal{C}$ eliminating any source of performance variation due to a different total computational budget. This is since per time step, the stream reveals fewer number of samples n with an increased number of time steps. We report experiments in the CGLM setting where Naive will receive only 40 and 10 iterations for the 50 and 200 time steps, respectively. We consider distillation approaches where they are permitted $^2/_3\mathcal{C}$, which is 27 and 7 iterations, respectively, on the 50 and 200 time steps, respectively. Note that, in these settings, per time step, methods observe $^2/_3 \times 11.6$K and $^2/_3 \times 2.9$K samples, respectively. We leave the experiments on ImageNet2K for the Appendix due to space constraints. We compare two distillation methods against Naive in Figure 8. Other methods are presented in the Appendix.

Conclusion. We still consistently observe that Naive outperforms all distillation methods on both the 50 and the 200 time steps. Moreover, the relative performance across distillation methods is preserved similarly to the 20 time steps setup. That is, our conclusions are largely robust under different number of time steps. This is contrary to the observation of the prior art [40], this is because unlike our setting, [40] does not scale the compute with increased number of time steps.

Does the Compute Budget Matter? Finally, we explore the impact of changing the computational budget on the performance of different distillation methods on CGLM under 20 time steps. We study two scenarios, one where the budget is increased to $\mathcal{C} = 400$ and where it is reduced to $\mathcal{C} = 40$, originally $\mathcal{C} = 100$ for CGLM. Hence, distillation would be allocated 267 and 27 iterations in this setting, respectively. As shown in Figure 2, the higher budget setting allows approximately a full pass per time step over all stored data. We leave the experiments on ImageNet2K for the Appendix. We compare two distillation methods with Naive in Figure 9. The remaining methods are presented in the Appendix.

Conclusion. Again, we observe that Naive outperforms all distillation methods in both increased and decreased compute budget settings. The final gap between MSE distillation and Naive is 11.41% for $\mathcal{C} = 40$, this gap is reduced to 3.85% for $\mathcal{C} = 400$. Surprisingly, even with increased compute budget, distillation methods still fall behind Naive. However, the reduced gap in performance compared to that of Naive is a strong indication that the reason behind the failure of distillation methods is indeed the limited computation.

4.4. Exploring Partial Training

Now, we explore reallocating the computational budget through partial training of the model. This falls into the category discussed earlier of "Model Expansion Methods" where we simply pre-select the subnetwork to be trained. This approach is more computationally efficient compared to training the full model, especially on large-scale datasets.

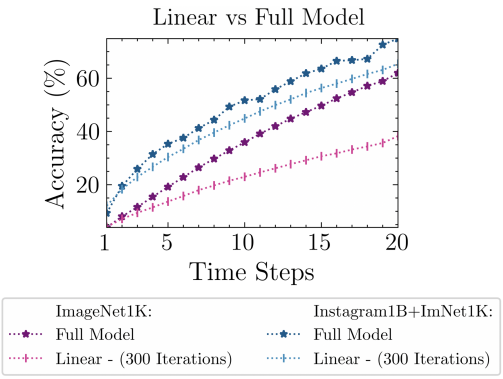

Figure 10. **Linear vs Full Model Training.** Performing Linear fine tuning allows to leveraging the computational budget efficiently improving the gap compared to full model training particularly for better pretrained models, *e.g.*, Instagram1B+ImNet1K.

The minimal part of a network that can be retrained is the top (FC) layer. We compare partial training of the network, *i.e.*, FC layer only, against training the full model (Naive) for two different model initializations, ImageNet1K pretraining [24] and Instagram1B+ImageNet1K pretraining [33]. Note that Instagram1B+ImageNet1K is a stronger pretrained model, with better feature representations. Since training the fully connected layer is computationally cheaper than training the full model, we normalize the computation for the FC partial training, permiting $3\mathcal{C}$ training iterations compared to full model training (Naive) with $\mathcal{C}$ training iterations. Therefore, CGLM FC partial training performs 300 iterations compared to 100 iterations for training Naive. We present our results in Figure 10, where the shades of purple represents models trained starting from a pretrained ImageNet1K model, while the shades of blue represent models trained starting with a pretrained Instagram1B+ImageNet1K model.

Conclusion. There exists a gap between full model training and partial FC layer training (denoted as Linear). However, this gap is greatly reduced when a stronger pretrained model is adopted as an initialization. More specifically, the final gap drops from 23.73% for ImageNet1K initialization to 9.45% for Instagram1B+ImageNet1K initialization. Partial training of the FC layer for Instagram1B+ImageNet1K model initialization outperforms ImageNet1K full model training on average, over time steps, by 8.08%, which verifies that partially training a strong backbone could be more beneficial than fully training a weaker one.

5. Conclusion

Existing CL algorithms, such as sampling strategies, distillation, and FC layer corrections, fail in a budgeted computational setup. Simple Naive methods based on experience replay outperform all the considered CL methods. This conclusion was persistent even under various computational budgets and an increased number of time steps. We find that most CL approaches perform worse when a lower computational budget is allowed per time step.

6. Acknowledgements

This work was supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research (OSR) under Award No. OSR-CRG2021, SDAIA-KAUST Center of Excellence in Data Science, Artificial Intelligence (SDAIA-KAUST AI), and UKRI grant: Turing AI Fellowship EP/W002981/1. We thank the Royal Academy of Engineering and FiveAI for their support. Ser-Nam Lim from Meta AI is neither supported nor has relationships with the mentioned grants. Ameya Prabhu is funded by Meta AI Grant No DFR05540.

References

[1] Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 4

[2] Hongjoon Ahn, Jihwan Kwak, Subin Lim, Hyeonsu Bang, Hyojun Kim, and Taesup Moon. Ss-il: Separated softmax for incremental learning. In *International Conference on Compute Vision (ICCV)*, 2021. 3, 4

[3] Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Laurent Charlin, and Tinne Tuytelaars. Online continual learning with maximally interfered retrieval. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019. 3, 4

[4] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 4

[5] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019. 3, 4, 12

[6] Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 3, 4

[7] Eden Belouadah and Adrian Popescu. Il2m: Class incremental learning with dual memory. In *International Conference on Computer Vision (ICCV)*, 2019. 4

[8] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021. 1

[9] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 2, 3, 12

[10] Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New insights on reducing abrupt representation change in online continual learning. In *International Conference on Learning Representations (ICLR)*, 2022. 2, 3, 4, 7

[11] Zhipeng Cai, Ozan Sener, and Vladlen Koltun. Online continual learning with natural distribution shifts: An empirical study with visual data. In *International Conference on Computer Vision (ICCV)*, 2021. 2, 3, 4

[12] Hyuntak Cha, Jaeho Lee, and Jinwoo Shin. Co2l: Contrastive continual learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3

[13] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *European Conference on Computer Vision (ECCV)*, 2018. 2

[14] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *International Conference on Learning Representations (ICLR)*, 2019. 12

[15] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. Continual learning with tiny episodic memories. In *International Conference on Machine Learning Workshop (ICML-W)*, 2019. 2, 3, 4

[16] Aristotelis Chrysakis and Marie-Francine Moens. Online continual learning from imbalanced data. In *ICML*, 2020. 4

[17] Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. In *International Conference on Computer Vision (ICCV)*, 2021. 3, 4

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 2, 4, 12

[19] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *European Conference on Computer Vision (ECCV)*, 2020. 3, 4, 12

[20] Yasir Ghunaim, Adel Bibi, Kumail Alhamoud, Motasem Alfarra, Hasan Abed Al Kader Hammoud, Ameya Prabhu, Philip HS Torr, and Bernard Ghanem. Real-time evaluation in online continual learning: A new paradigm. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[21] Shashwat Goel, Ameya Prabhu, and Ponnurangam Kumaraguru. Evaluating inexact unlearning requires revisiting forgetting. *arXiv preprint arXiv:2201.06640*, 2022. 12

[22] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning (ICML)*, 2017. 7

[23] Niv Haim, Gal Vardi, Gilad Yehudai, Ohad Shamir, and Michal Irani. Reconstructing training data from trained neural networks. *arXiv preprint arXiv:2206.07758*, 2022. 12

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 8, 12

[25] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 3, 4, 7, 12

[26] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska,

et al. Overcoming catastrophic forgetting in neural networks. In *PNAS*, 2017. 1

[27] Hyunseo Koh, Dahyun Kim, Jung-Woo Ha, and Jonghyun Choi. Online continual learning on class incremental blurry task configuration with anytime inference. *International Conference on Learning Representations (ICLR)*, 2022. 3, 4

[28] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 2015. 1

[29] Zhiqiu Lin, Jia Shi, Deepak Pathak, and Deva Ramanan. The clear benchmark: Continual learning on real-world imagery. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021. 1, 2, 3, 12

[30] Yaoyao Liu, Bernt Schiele, and Qianru Sun. Rmm: Reinforced memory management for class-incremental learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 3

[31] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3

[32] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2017. 2, 12

[33] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. Exploring the limits of weakly supervised pretraining. In *European conference on computer vision (ECCV)*, 2018. 8

[34] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 2022. 2, 4, 7

[35] Zheda Mai, Ruiwen Li, Hyunwoo Kim, and Scott Sanner. Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshop (CVPR-W)*, 2021. 3

[36] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *European Conference on Computer Vision (ECCV)*, 2018. 4

[37] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 4

[38] Marc Masana, Xialei Liu, Bartlomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost van de Weijer. Class-incremental learning: survey and performance evaluation on image classification. *arXiv preprint arXiv:2010.15277*, 2020. 6

[39] Ameya Prabhu, Zhipeng Cai, Puneet Dokania, Philip HS Torr, Vladlen Koltun, and Ozan Sener. Online continual learning without the storage constraint. *Link*, 2023. 2, 3

[40] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *European Conference on Computer Vision (ECCV)*, 2020. 2, 3, 4, 7, 8

[41] Jathushan Rajasegaran, Munawar Hayat, Salman Khan, Fahad Shahbaz Khan, and Ling Shao. Random path selection for incremental learning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019. 4

[42] Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. itaml: An incremental task-agnostic meta-learning approach. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 4

[43] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. *Advances in neural information processing systems*, 30, 2017. 4

[44] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2, 3, 4, 12

[45] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning (ICML)*, 2019. 12

[46] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. 4

[47] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 2015. 1

[48] Dongsub Shim, Zheda Mai, Jihwan Jeong, Scott Sanner, Hyunwoo Kim, and Jongseong Jang. Online class-incremental continual learning with adversarial shapley value. In *AAAI Conference on Artificial Intelligence*, 2021. 3, 4

[49] Shengyang Sun, Daniele Calandriello, Huiyi Hu, Ang Li, and Michalis Titsias. Information-theoretic online memory selection for continual learning. In *International Conference on Learning Representations (ICLR)*, 2022. 3, 4

[50] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations (ICLR)*, 2019. 4

[51] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Growing a brain: Fine-tuning by increasing model capacity. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 4

[52] Tz-Ying Wu, Gurumurthy Swaminathan, Zhizhong Li, Avinash Ravichandran, Nuno Vasconcelos, Rahul Bhotika, and Stefano Soatto. Class-incremental learning with strong pre-trained models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 4

[53] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 3, 4, 7

[54] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. *arXiv preprint arXiv:2103.16788*, 2021. 4

[55] Huaxiu Yao, Caroline Choi, Yoonho Lee, Pang Wei Koh, and Chelsea Finn. Wild-time: A benchmark of in-the-wild distri-

bution shift over time. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022. 3

[56] JaeHong Yoon, Jeongtae Lee, Eunho Yang, and Sung Ju Hwang. Lifelong learning with dynamically expandable network. In *International Conference on Learning Representations (ICLR)*, 2018. 4

[57] Jaehong Yoon, Divyam Madaan, Eunho Yang, and Sung Ju Hwang. Online coreset selection for rehearsal-based continual learning. In *International Conference on Learning Representations (ICLR)*, 2022. 3, 4

[58] Chen Zeno, Itay Golan, Elad Hoffer, and Daniel Soudry. Task agnostic continual learning using online variational bayes. *Neural Computation*, 2021. 2, 4, 7

[59] Jeffrey O Zhang, Alexander Sax, Amir Zamir, Leonidas Guibas, and Jitendra Malik. Side-tuning: a baseline for network adaptation via additive side networks. In *European Conference on Computer Vision (ECCV)*, 2020. 4

[60] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 3, 4, 7

A. Motivation: Privacy for Restricting Memory

Ref.	Dataset	Ordering	Memory	Cost	Iters	Cost
[9]	CIFAR10	Cls Inc	1-25MB	0.05¢	250K-375K	20$
[44]	CIFAR100	Cls Inc	10 MB	0.02¢	50K	8$
[19, 25]					125K	15$
[9]	TinyImageNet	Cls Inc	5-20 MB	0.04¢	350K-500K	25$
[19, 25]	ImageNet100	Cls Inc	0.3-1 GB	2¢	100K	50$
[19, 25]	ImageNet1K	Cls Inc	33GB	66¢	1M	500$
[29]	CLEAR	Dist Shift	0.4-1.2GB	2¢	300K	100$
[24]	ResNet50 (bs=256)		22GB			
Ours	GLDv2-CL	Dist Shift	**90GB**	2$	2K	10$
	ImageNet21K-CL	ClsInc, DataInc	**400GB**	10$	8K	35$

Table 3. **Cost of Memory vs Computation.** Google Cloud Standard Storage (2¢per GB per month for 1 month) and Compute Cost measured as running cost of an A2 instance (3$ per hour for 1 GPU). Number of iterations (forward+backward passes) for training a CL model on that dataset listed for comparison invariant to input image and model sizes. We observe that computational costs for running an experiment far outweigh the costs for storing replay samples.

Prior art on continual learning [5, 9, 14, 29, 32, 44] motivate the problem from the aspect of prohibited access to previously received data; except for a small portion that is allowed to store in memory. The two principal motivations behind restricting the access to past samples in the literature are two folds. (**i**) Storage space is expensive. (**ii**) Access to previous data is prohibitive due to privacy and GDPR constraints.

As for the first argument used as a motivation for limiting the memory size, as we have elaborated in Section 1 of the main paper and similarly further detail in Table 3, the cost of storing data is insignificant. This is particularly the case when considering the associated computational costs of training deep models. To that end, the argument of to restricting the memory size is a an enough justifiable reason. For example, as per Table 3, it costs 2 cents to store the entirety of the CLEAR dataset, among the largest datasets for continual learning, while it costs about 100$ to train a model continually on the same dataset. If reducing costs are the key issue, limited computational budget and not memory, as argued earlier, is the way forward.

As for privacy considerations, this is too a not well-motivated reason behind limiting memory. A classical argument is that due to GDPR requirements, data needs to be removed or company privacy policies, it can no longer accessible after x number of months. First, any previous benchmark with memory constraints already violates this privacy consideration. This is since, which data is to be made private and shall be deleted should not be up to the learning algorithm to decide. To that end, restricting memory samples does not help solving the privacy considerations. Even if the samples stored in memory were selected such that they do not violate any privacy constraints, which none of the prior art address, it remains a question on whether the trained models preserve any sensitive information after training on private data. Without imposing such restriction on the learning algorithm and the underlying model, restricting the memory for the argument of privacy does not meet its stated objectives. This is particularly the case, as Haim *et al*. [23] have found that models retain a lot of information of the training samples. This is to an extent that large number of training samples can be reconstructed only given the trained model. Goel *et al*. [21] presents a catastrophic forgetting baseline, indicating forgetting might be the very objective of sustaining privacy which is antithetical to the objective of continual learning.

In conclusion, restricting access to previous samples by restricting memory do not contribute to solving the privacy problem. Instead, we consider the more realistic setting where only limited amount of computation is given due to cost restraint or the need to predict every sample in a high throughput stream. This in any how imposes an implicit restraint on access of past memory samples.

B. Dataset Construction

B.1. Constructing Imagenet2K

ImageNet2K train set is constructed using all training images in ImageNet1K dataset [18] for 1K classes as an initialization, with selecting an additional 1K non-overlapping classes from ImageNet21K dataset [18] to form the ImageNet2K dataset. We illustrate the creation of the test, validation and train sets below:

Test Set: We use the ImageNet1K val set as the test set to be consistent with test sets used in previous literature using ImageNet1K. We separate 50 images per class from the sample set of the new 1K classes. We combine these two sets to create the overall test set for experiments. The test set for every timestep consists of classes from this test set which have been seen so far.

Validation Set: We use ImagenetV2 dataset [45] as the validation set from Imagenet1K data. We seperate 50 images per class, not used in the test set to create the val set for the new 1K classes. We combine these two sets to create the overall

validation set for experiments. The validation set for every timestep consists of classes from this validation set which have been seen so far.

Train Set: We order all the samples from the new 1K classes not used for creating the test and val sets for training. We order them by classes to form the CI-ImageNet2K stream and randomly shuffle all these images to form the DI-ImageNet2K stream. Note that the stream order is provided samplewise, allowing the stream size N to be adjusted. In the standard experiments, data equivalent to 50 classes is sampled every timestep, for 20 timesteps.

B.2. Constructing Continual Google Landmarks V2

Continual Google Landmarks V2 (CGLM) consists of 580K samples. To obtain this subset, we start with the train-clean subset of the Google Landmarks V2 available from the Google Landmarks V2 dataset website[2]. We apply the following preprocessing steps in order:

1. Filter out images which do not have timestamp metadata available.
2. Remove images of classes that have less than 25 samples in total
3. Order data by timestamp.
4. Randomly sample 10% of data from across time as the test set

We get the rest $580K$ images as the train set for continual learning over 10788 classe, with rapid temporal distribution shifts. We do not have a validation set here as we benchmark transfer of hyperparameters used from ImageNet to this dataset.

C. Estimation of Equivalent Iterations

In this section, we elaborate on the details of selecting the computational budget for distillation and expensive sampling approaches. The key point for these calculations is the fact that the computational (and time) cost for a forward pass is $1/2$ the cost of a backward pass [3].

Distillation: When distillation approaches have a budget of $2/3^{rd}$ iterations of the naive baseline, the computational cost is as follows: $2/3\mathcal{C}$ for training of the student model, and $1/2 \times 2/3 = 1/3\mathcal{C}$ for the teacher model which only has a forward pass, which sums up to $\mathcal{C}$. Hence, distillation methods have an equivalent computational budget as the naive baseline with $2/3^{rd}$ training iterations.

Sampling: We train the expensive models for $1/2$ the number of iterations as a naive model. To select that subset of training data, we randomly sample $3\times$ the required number of training samples from the stored set and forward pass them through the latest trained model to obtain the features/probabilities. And then we select the best $1/3^{rd}$ of the $3\times$ set for training using different selection functions. We assume the cost of selecting samples given the features/probabilities is negligible.

The computational cost of training for expensive sampling methods is $1/2\mathcal{C}$, as the selected sample set is half the size compared to the naive baseline. The computational cost of selecting the samples is $1/2 \times 3 \times 1/3\mathcal{C} = 1/2\mathcal{C}$ (forward pass requires $1/3^{rd}$ of the total cost, on $3\times$ the required data, the required data size being $1/2$ when compared to naive). The combined cost is the sum of selection and training cost, which is $\mathcal{C}$. Hence, expensive sampling methods have an equivalent budget with $1/2$ training iterations.

D. Additional Results

Due to limited space, some of the experiments in the manuscripts were deferred to the Appendix. In this section we present results for all mentioned costly sampling methods and distillation methods. Additional results for ***Section 4.2: 1 "Do Sampling Strategies Matter?"*** are presented in Figure 11 where all three costly sampling methods are presented, namely Max Loss, Uncertainty Loss, and KMeans. Similarly, additional results for ***Section 4.2: 2 "Does Distillation Matter?"*** are presented in Figure 12 where all four distillation methods are presented, namely BCE, MSE, Cosine, and CrossEntropy. We observe that all previous conclusions consistently hold, *i.e.* the Naive baseline is still leading in comparison to all previous approaches.

We also extend the time steps and the number of iterations sensitivity experiments to all four considered distillation methods in all three setups, DI-ImageNet2K, CI-ImageNet2K and CGLM. We present additional results for ***Section 4.3: 1. Does the Number of Time Steps Matter?***: with results for DI-ImageNet2K presented in Figures 13 and 14 with 50 and 200 time steps respectively, CI-ImageNet2K in Figure 15 and 16 with 50 and 200 time steps respectively and CGLM in Figures 17 and 18 with 50 and 200 time steps respectively. We observe that all previous conclusions consistently hold, *i.e.* the conclusions are robust to changing time steps for a given cost $\mathcal{C}$.

[2] https://github.com/cvdfoundation/google-landmark

[3] https://www.lesswrong.com/posts/jJApGWG95495pYM7C/how-to-measure-flop-s-for-neural-networks-empirically

We present additional results for ***Section 4.3: 2. Does the Compute Budget Matter?***: on DI-ImageNet2K in Figures 19 and 20 for 100 and 1200 iterations respectively, and CI-ImageNet2K in Figures 21 and 22 for 100 and 1200 iterations respectively and on CGLM in Figures 23 and 24 for 40 and 400 iterations respectively. We observe that all previous conclusions consistently hold, *i.e*. the conclusions are robust to changing computational cost, towards both harsher and laxer computational constraint regimes.

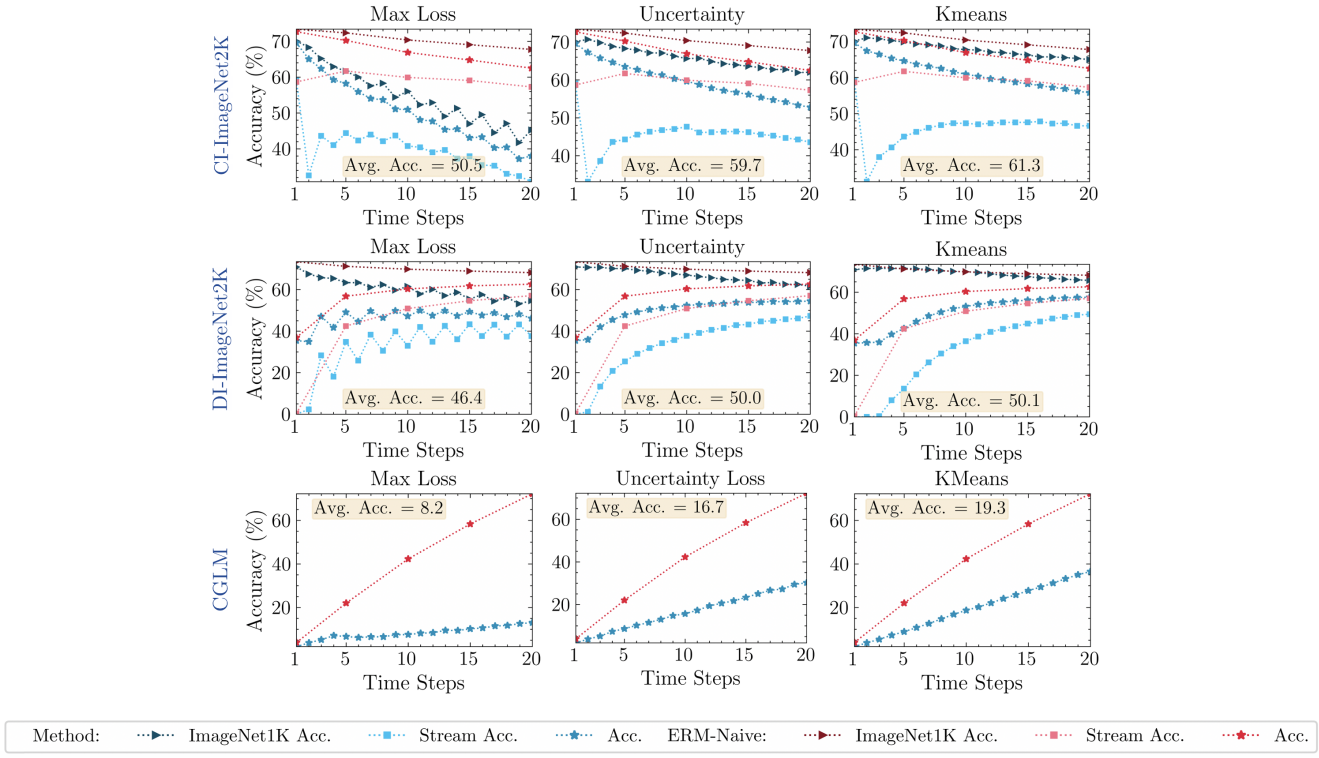

Figure 11. **Expensive Sampling.** As mentioned in the manuscript, KMeans performs the best among expensive sampling techniques such as Max Loss and Uncertainty Loss. Nevertheless, the performance of the expensive sampling methods is worse than simple Naive.

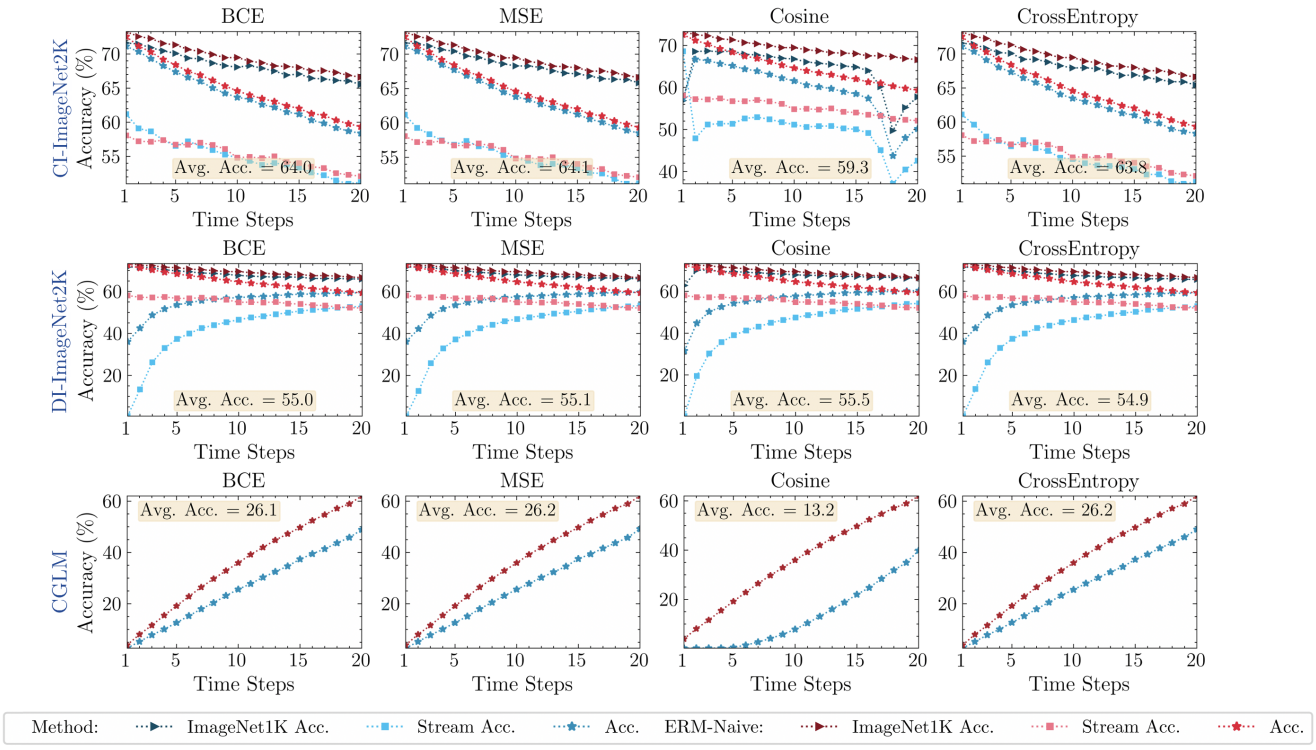

Figure 12. **Distillation Methods.** All four studied distillation methods under perform compared to the simple Naive baseline in all three studied settings. ImageNet experiments are allowed 400 iterations whereas CGLM is allowed 100 iterations.

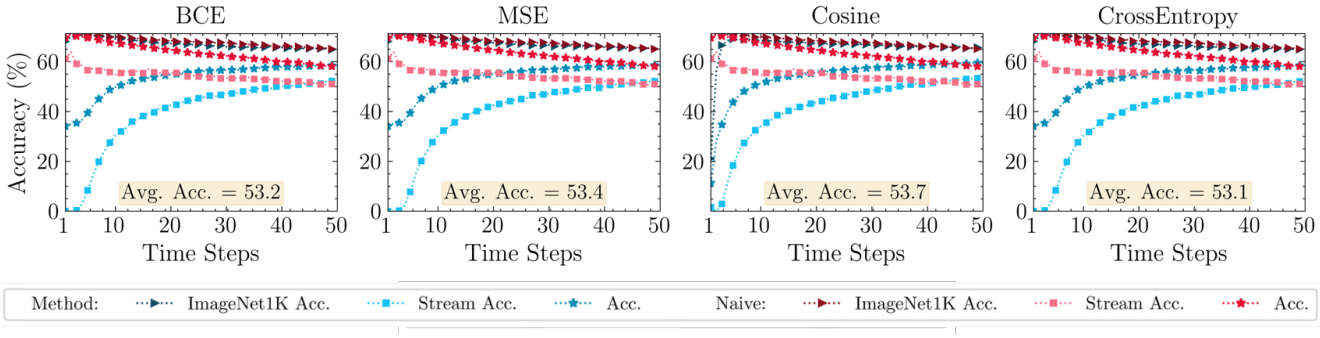

Figure 13. **DI-ImageNet2K 50 Time Steps.** As observed in the manuscript, when the number of time steps, distillation methods still under perform compared to the Naive baseline.

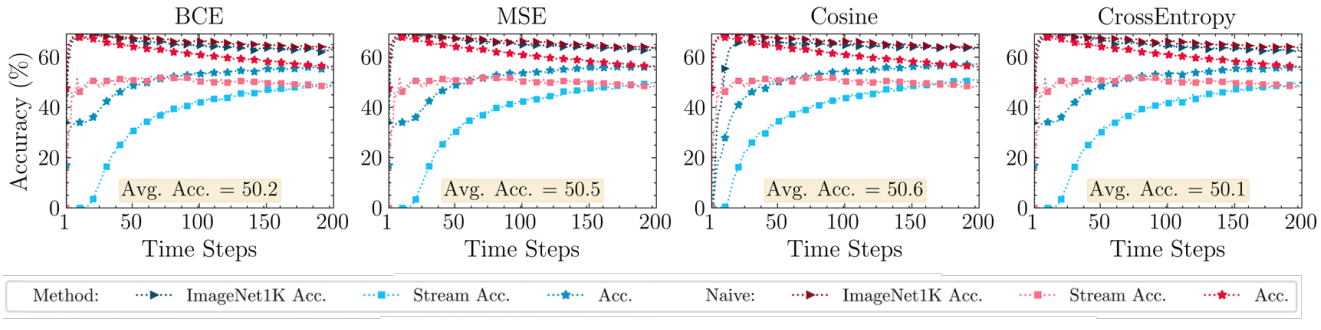

Figure 14. **DI-ImageNet2K 200 Time Steps.** As observed in the manuscript, when the number of time steps, distillation methods still under perform compared to the Naive baseline.

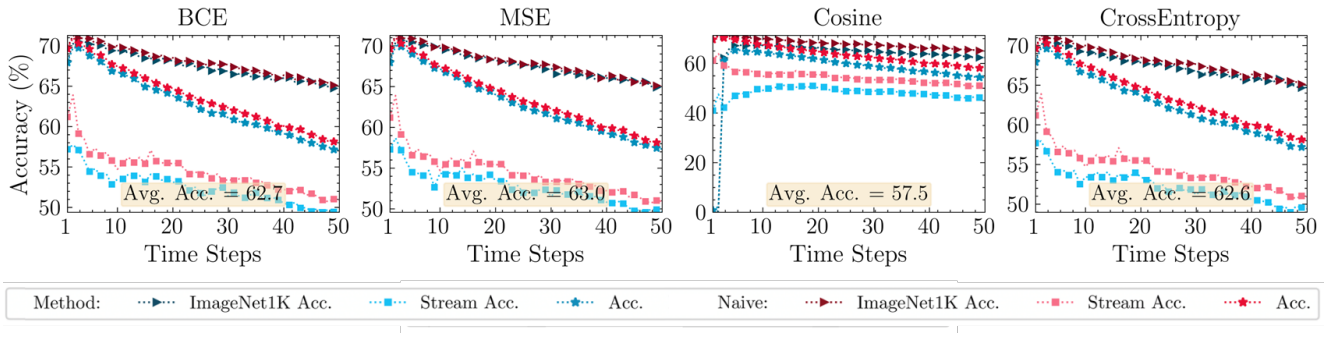

Figure 15. **CI-ImageNet2K 50 Time Steps.** As observed in the manuscript, when the number of time steps, distillation methods still under perform compared to the Naive baseline.

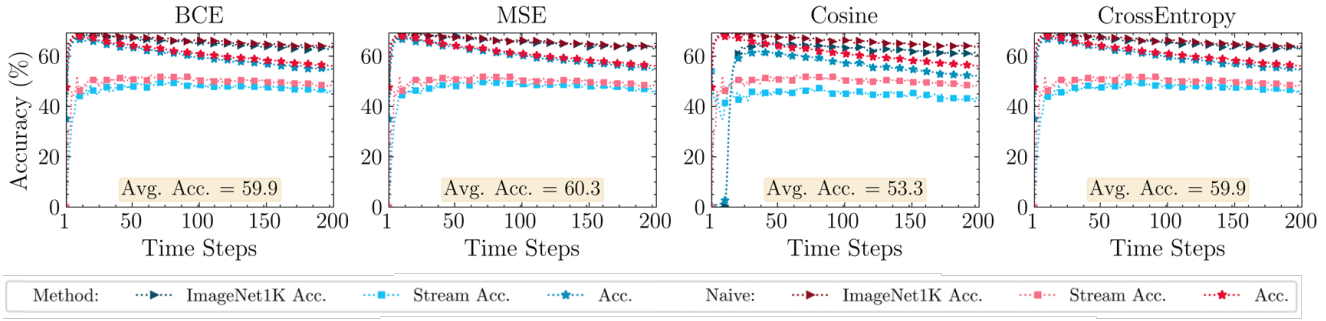

Figure 16. **CI-ImageNet2K 200 Time Steps.** As observed in the manuscript, when the number of time steps, distillation methods still under perform compared to the Naive baseline.

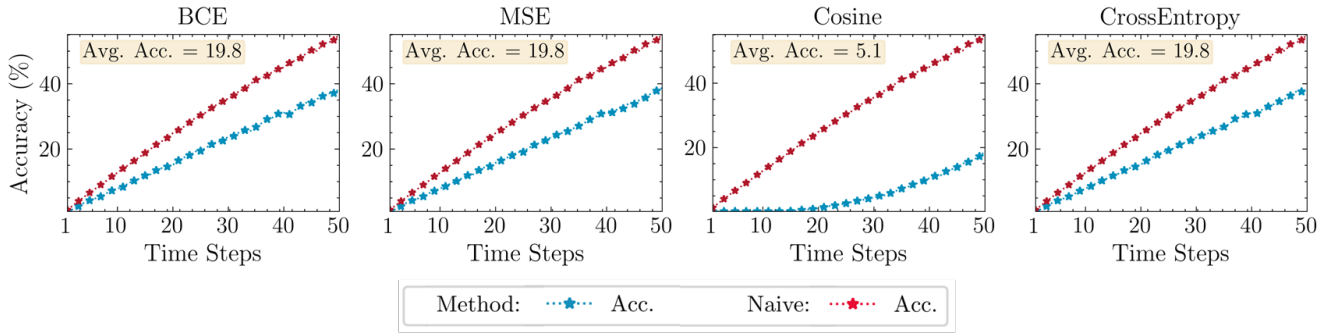

Figure 17. **CLGM 50 Time Steps.** As observed in the manuscript, when the number of time steps, distillation methods still under perform compared to the Naive baseline.

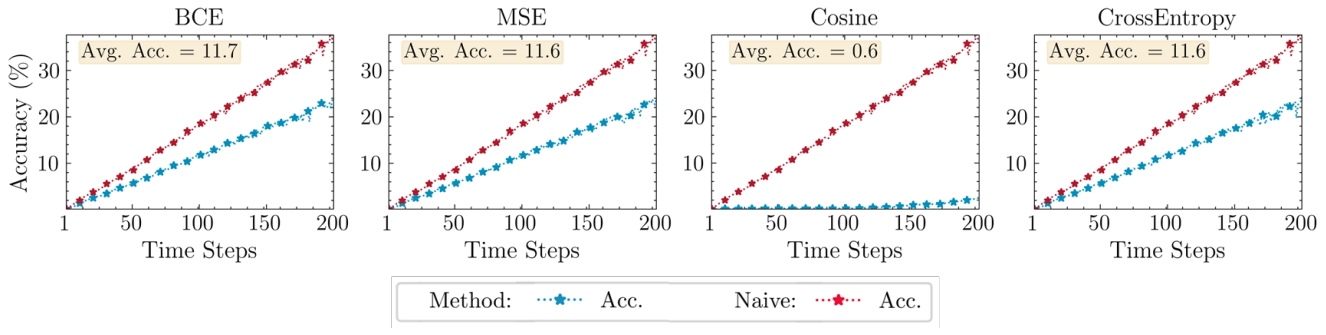

Figure 18. **CGLM 200 Time Steps.** As observed in the manuscript, when the number of time steps, distillation methods still under perform compared to the Naive baseline.

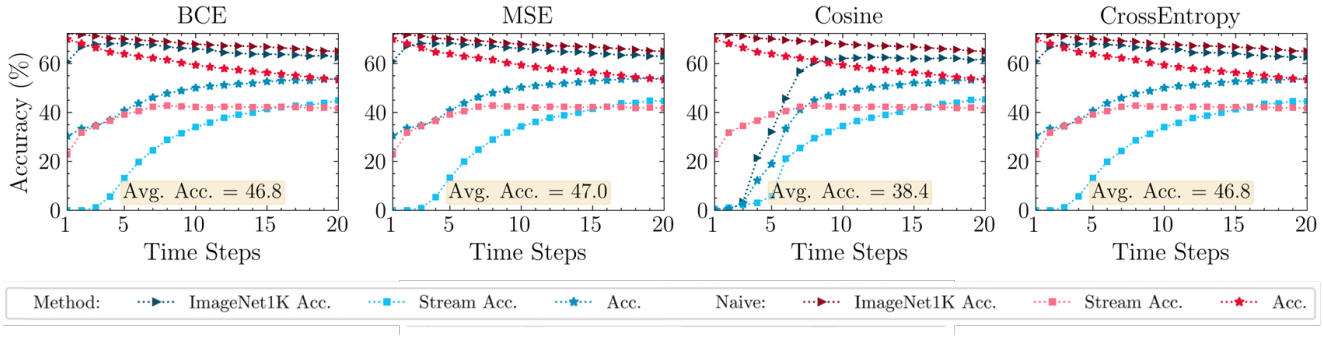

Figure 19. **DI-ImageNet2K - 100 Iterations.** As observed in the manuscript, with reduced compute, distillation methods still under perform compared to the Naive baseline. The compute budget of the Naive baseline, $\mathcal{C}$, is set to 100 iterations whereas that of the distillation methods is $^2/_3\,\mathcal{C} = 67$ iterations.

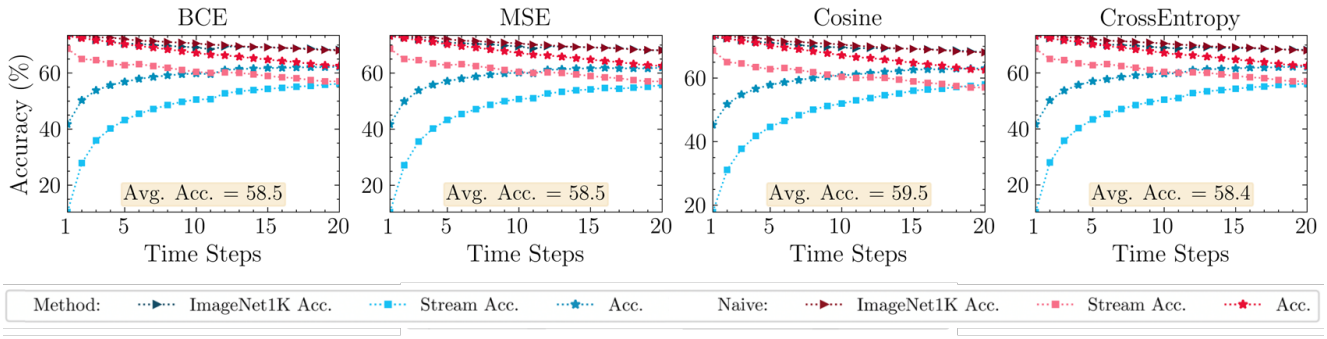

Figure 20. **DI-ImageNet2K - 1200 Iterations.** As observed in the manuscript, with increased compute, distillation methods still under perform compared to the Naive baseline. The compute budget of the Naive baseline, $\mathcal{C}$, is set to 1200 iterations whereas that of the distillation methods is $^2/_3\,\mathcal{C} = 800$ iterations.

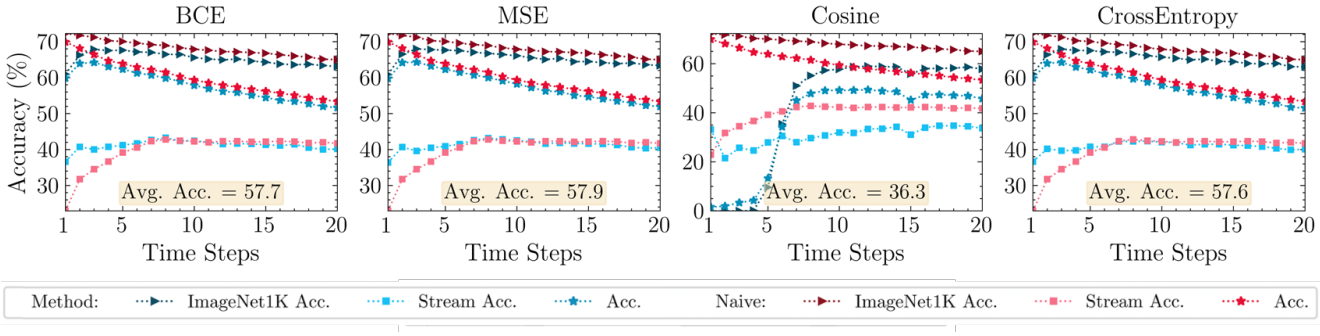

Figure 21. **CI-ImageNet2K - 100 Iterations.** As observed in the manuscript, with reduced compute, distillation methods still under perform compared to the Naive baseline. The compute budget of the Naive baseline, $\mathcal{C}$, is set to 100 iterations whereas that of the distillation methods is $^2/_3\,\mathcal{C} = 67$ iterations.

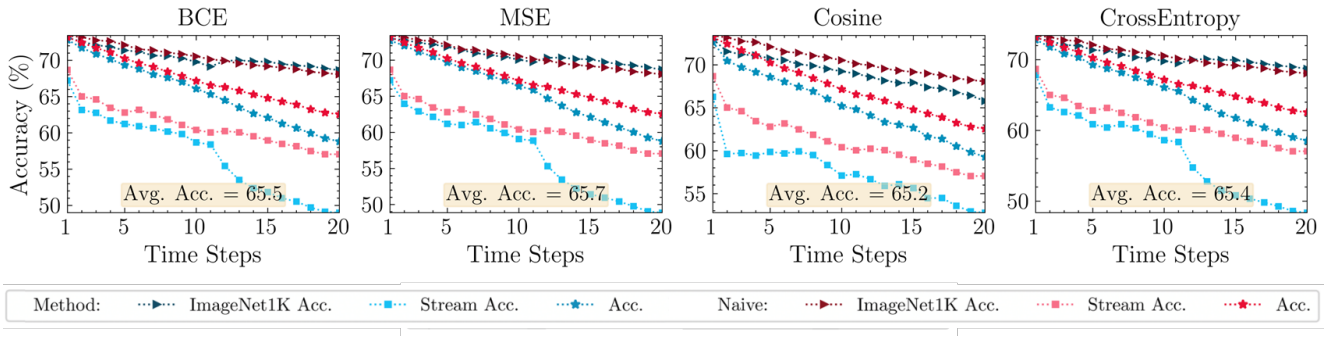

Figure 22. **CI-ImageNet2K - 1200 Iterations.** As observed in the manuscript, with increased compute, distillation methods still under perform compared to the Naive baseline. The compute budget of the Naive baseline, $\mathcal{C}$, is set to 1200 iterations whereas that of the distillation methods is $^2/_3\,\mathcal{C} = 800$ iterations.

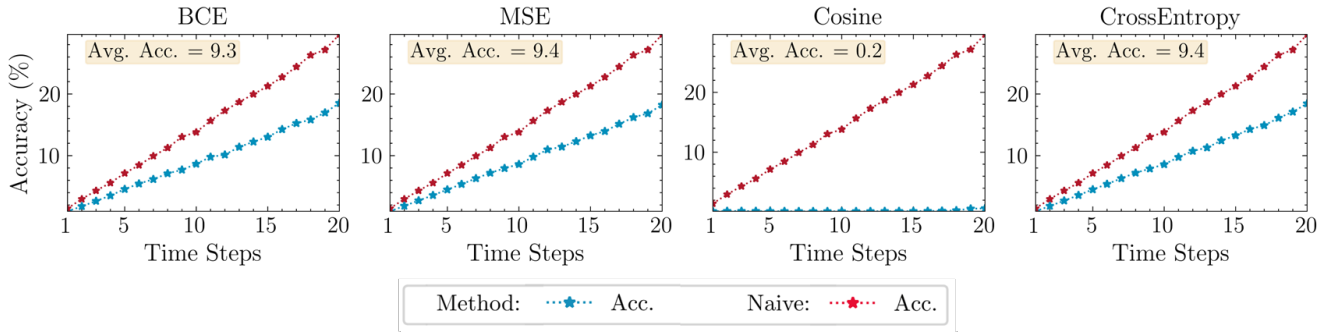

Figure 23. **CGLM 40 - Iterations.** As observed in the manuscript, with reduced compute, distillation methods still under perform compared to the Naive baseline. The compute budget of the Naive baseline, $\mathcal{C}$, is set to 40 iterations whereas that of the distillation methods is $^2/_3\,\mathcal{C} = 27$ iterations.

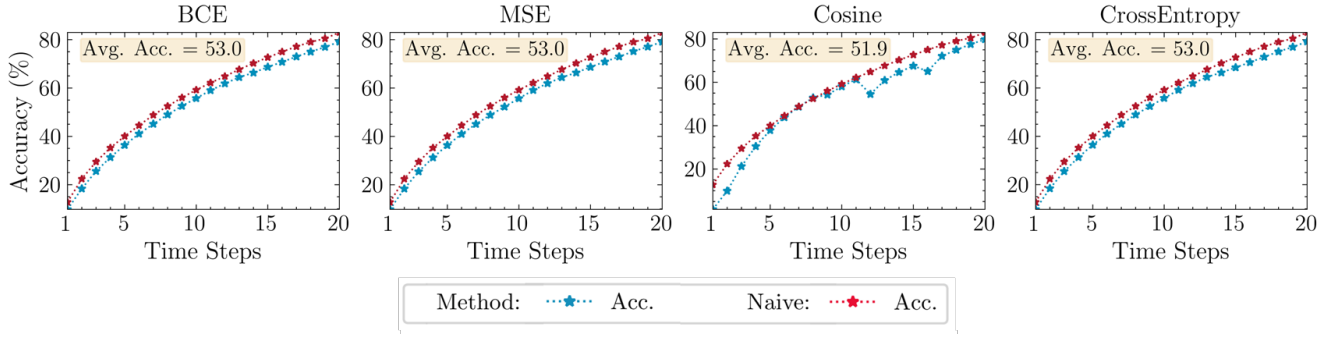

Figure 24. **CGLM 400 - Iterations.** As observed in the manuscript, with increased compute, distillation methods still under perform compared to the Naive baseline. The compute budget of the Naive baseline, $\mathcal{C}$, is set to 400 iterations whereas that of the distillation methods is $^2/_3\,\mathcal{C} = 267$ iterations.

D.1. Effect of Weight Decay

The choice of weight decay, wd $= 0$, in the manuscript was based on result of cross-validation from the set. More specifically, we try weight decays of $\{5 \times 10^{-5}, 1 \times 10^{-4}\}$. We observe a minor difference in performance between various weight decays, with a wd=0 consistently being slightly better. The results are shown in Figure 25

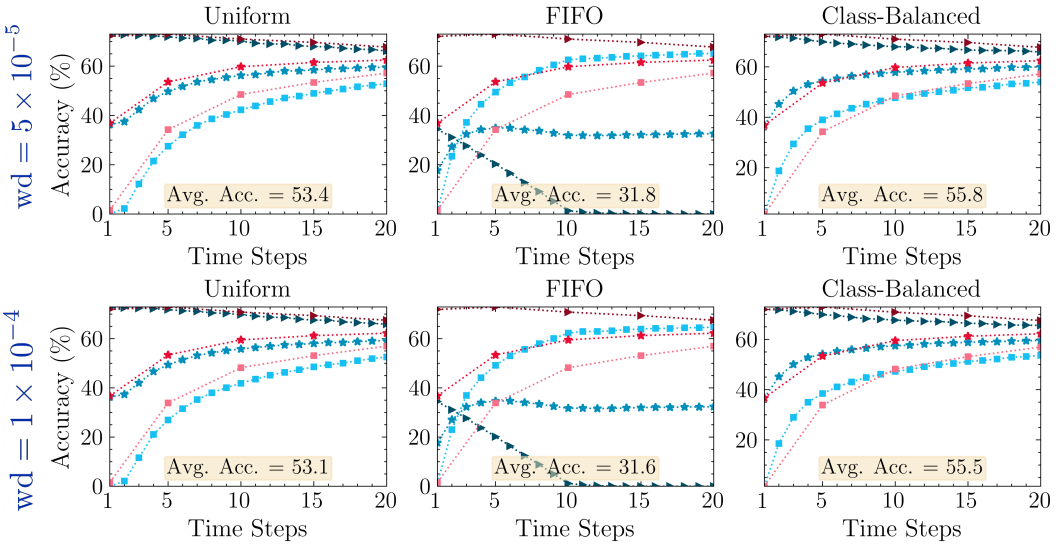

Figure 25. **Effect of Weight Decay.** Increasing the weight decay causes a slight drop in performance. Setting wd=0 gave the best results during our parameter cross-validation.

D.2. Effect of Batch Size

In all experiments, we fixed the batch size (BS) to 1500 to optimize the utilization of our hardware resources and minimize the training time. As shown in the literature, BS and learning rate (LR) are closely related. The selected LR was tuned to fit the selected BS. Regardless, we present varying BS experiments in Fig (1) where we study the latest FC correction method, ACE, and the distillation method, MSE, for BS of 250 and 500 with increased iterations of 600 and 300, respectively, and crossvalidated learning rates. We observe that the Naive baseline is still superior even when the batch size is adjusted. Our findings are summarized in Figure 26.

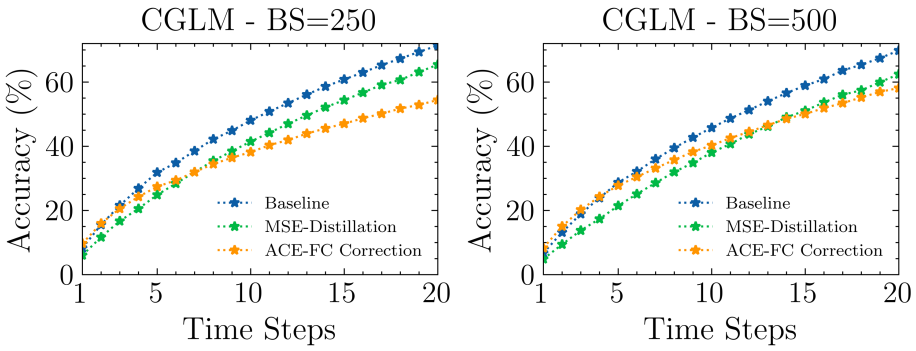

Figure 26. **Effect of Batch Size.** The conclusions presented in our work hold even when the batch size while is changed while the same overall computational budget.

D.3. Effect of Increasing Computational Budget on Distillation

We complement the results in Figure 9 with additional experiments using 800 and 1200 iterations. The observed results align with our previous observations; as long as the computation is normalized across methods, naive, the most simplest among

the considered methods, outperforms existing methods. This is as opposed to prior art comparison that does not normalize compute, which puts the Naive baseline in disadvantage. The results are shown in Figure 27.

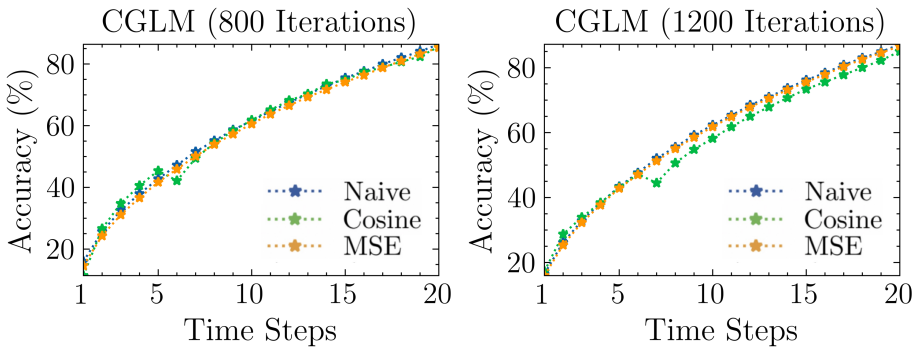

Figure 27. **Effect of Increasing Computation Budget on Distillation.** As the computation budget is increased while maintaining a normalized compute among different methods, Naive baseline still outperforms distillation based methods.