Online Continual Learning Without the Storage Constraint

Ameya Prabhu *ameya@robots.ox.ac.uk*
University of Oxford

Zhipeng Cai *zhipeng.cai@intel.com*
Intel Labs

Puneet Dokania *puneet@robots.ox.ac.uk*
University of Oxford

Philip Torr *phst@robots.ox.ac.uk*
University of Oxford

Vladlen Koltun *vkoltun@apple.com*
Apple

Ozan Sener *ozansener@apple.com*
Apple

Abstract

Online continual learning (OCL) research has primarily focused on mitigating catastrophic forgetting with fixed and limited storage allocation throughout the agent's lifetime. However, the growing affordability of data storage highlights a broad range of applications that do not adhere to these assumptions. In these cases, the primary concern lies in managing computational expenditures rather than storage. In this paper, we target such settings, investigating the online continual learning problem by relaxing storage constraints and emphasizing fixed, limited economical budget. We provide a simple algorithm that can compactly store and utilize the entirety of the incoming data stream under tiny computational budgets using a kNN classifier and universal pre-trained feature extractors. Our algorithm provides a consistency property attractive to continual learning: It will never forget past seen data. We set a new state of the art on two large-scale OCL datasets: Continual LOCalization (CLOC), which has 39M images over 712 classes, and Continual Google Landmarks V2 (CGLM), which has 580K images over 10,788 classes – beating methods under far higher computational budgets than ours in terms of both reducing catastrophic forgetting of past data and quickly adapting to rapidly changing data streams. We provide code to reproduce our results at `https://github.com/drimpossible/ACM`.

1 Introduction

In online continual learning, a learner processes a continuous stream of data originating from a non-stationary distribution. The learner is required to solve a number of problems: it needs to successfully learn the main task (accuracy), adapt to changes in the distribution (rapid adaptation), and retain information from the past (backward transfer). A key motif in recent work on online continual learning is the search for algorithms that control the trade-off between these possibly competing objectives under resource constraints.

To establish the resource constraints for typical commercial settings, we assess what is required of continual learners. A continual learner must deliver accurate predictions, scale to large datasets encountered during its operational lifetime, and operate within the system's total cost budget (in dollars). The economics of data storage have been studied since 1987 (Gray & Putzolu, 1987; Gray & Graefe, 1997; Graefe, 2009; Appuswamy et al., 2017). Table 1 summarizes the trends, show a rapid decline in storage costs over time ($\sim$ \$100 to store CLOC, the largest dataset for OCL (Cai et al., 2021), in 2017). In contrast, running ER (Cai et al., 2021), the state-of-the-art OCL method on the subset of YFCC-100M currently costs over \$2000 on a GCP

Table 1: The cost of storing data has decreased rapidly, allowing the storage of a large dataset for a negligible cost compared to the cost of computation.

	1987	1997	2007	2017
Storage Cost				
Unit price ($)	30K	2K	80	49
Unit capacity	180MB	9GB	250GB	2TB
$/MB	83.33	0.22	0.0003	0.00002
Cost of storing YFCC ($)	350M	920K	1250	83
Compute Cost				
Training ER on YFCC		>2000$		

server. Consequently, computational costs are the primary budgetary concern, with storage costs being relatively insignificant. Therefore, as long as computational costs are controlled, economically storing the entire incoming data stream is feasible.

However, online continual learning has primarily been studied under limited storage constraints (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019a; Aljundi et al., 2019b), with learners only allowed to store a subset of incoming data. This constraint has led to many algorithms focusing on identifying a representative data subset (Aljundi et al., 2019b; Yoon et al., 2022; Chrysakis & Moens, 2020; Bang et al., 2021; Sun et al., 2022; Koh et al., 2022). Although limited storage is a practical constraint for biological learning agents and offline embodied artificial agents, deep learning-based systems are predominantly compute-constrained with a high throughput requirement. They must process incoming data points per second faster than the incoming stream speed to successfully keep up with the data stream. Cai et al. (2021) shows that even with unlimited storage, the online continual learning problem is hard as the constraint of limited computation implicitly restricts the effective samples used for training.

In this paper, we argue that storing the entirety of the data stream while meeting these requirements is possible. We propose a system based on approximate k-nearest neighbor (kNN) algorithms (Malkov & Yashunin, 2018), which are well-known for their scalability and inherent incremental nature (using only insert and lookup operations). The computational cost of this system has a graceful logarithmic scaling with data size even though it stores the entirety of past data. The further rationales for kNN are threefold. i) With the right representation, the nearest neighbour rule is an effective predictor at scale. ii) It does not forget past data. In other words, if a data point from history is queried again, the query yields the same label. We refer to this as the consistency property. iii) All past data can be compactly stored in low-dimensional feature representations.

A critical aspect of the aforementioned system is obtaining an effective feature representation. While feature learning is the standard approach, it is not viable in our setting due to the need to recompute the representation of stored data, implying a quadratic computational cost. Instead, we propose to use existing pretrained features that are based on extensive and diverse datasets and yield robust representations. Remarkably, we find that even pretrained representations trained on rather small-scale ImageNet1K (Caron et al., 2021) can provide effective features on datasets like Continual YFCC-100M (CLOC) (Cai et al., 2021) which are comparatively more complex and far larger in size. Additionally, our approach overcomes a significant limitation of existing gradient-descent-based methods: the ability to learn from a single example. While using one gradient per example (i.e., batch size 1) is computationally infeasible for deep networks on large-scale datasets, our method efficiently stores a single feature extracted from the pretrained model in memory. The kNN mechanism immediately utilizes this data point, enabling rapid adaptation. We argue that the capacity to adapt to a single example is essential for truly online operation, allowing our simple method to outperform existing continual learning baselines.

Problem formulation. We formally define the online continual learning (OCL) problem following Cai et al. (2021). In classification settings, we aim to continually learn a function $f: \mathcal{X} \to \mathcal{Y}$, parameterized by θ_t at time t. OCL is an iterative process where each step consists of a learner receiving information and updating its model. Specifically, at each step t of the interaction,

1. *One* data point $x_t \sim \pi_t$ sampled from a non-stationary distribution π_t is revealed.
2. The learner makes the scalar prediction $\hat{y}_t = f(x_t; \theta_t)$ using a compute budget, B_t^{pred}.

3. Learner receives the true label y_t.

4. Learner updates the model θ_{t+1} using a compute budget, B_t^{learn}

We evaluate the performance of the algorithm using the metrics used forward transfer (adaptability) and backward transfer (information retention) as given in Cai et al. (2021). A critical aspect of OCL is the budget in the second and forth steps, which limits the computation that the learner can expend. A common choice in past work is to impose a fixed limit on storage and computation per operation (Cai et al., 2021). We remove the storage constraint and argue that storing the entirety of the data is cost-effective as long as impact on computation is controlled. We relax the fixed computation constraint to a logarithmic constraint. In other words, we require that the computation time per operation fit within $B_t^{pred}, B_t^{learn} \sim \mathcal{O}(\log t)$. This construction results in total cost scaling $\mathcal{O}(n \log n)$ with the amount of data [1]

2 Related Work

Formulations. Parisi et al. (2019) and De Lange et al. (2020) have argued for improving the realism of online continual learning benchmarks. Earliest formulations (Lopez-Paz & Ranzato, 2017) worked in a task-incremental setup, assuming access to which subset of classes a test sample is from. Subsequent mainstream formulation (Aljundi et al., 2019b;a) required models to predict across all seen classes at test time, with progress in the train-time sample ordering (Bang et al., 2021; Koh et al., 2022). However, Prabhu et al. (2020) highlighted the limitations of current formulations by achieving good performance despite not using any unstored training data. Latest works (Hu et al., 2022; Cai et al., 2021; Lin et al., 2021) overcome this limitation by testing the capability for rapid adaptation to next incoming sample and eliminate data-ordering requirements by simply using timestamps of real-world data streams. Our work builds on the latest generation of formulation Cai et al. (2021). Unlike Cai et al. (2021), we perform one-sample learning; in other words, we entirely remove the concept of task by processing the incoming stream one sample at a time, in a truly online manner. Some parallel works stress on computation as the key limited resource (Harun et al., 2023). However, we further remove the storage constraint which is the key to eliminating degenerate solutions like GDumb (Prabhu et al., 2020).

Methods. Traditional methods of adapting to concept drift (Gama et al., 2014) include a variety of approaches based on SVMs (Laskov et al., 2006; Zheng et al., 2013), random forests (Gomes et al., 2017; Ristin et al., 2015; Mourtada et al., 2019), and other models (Oza & Russell, 2001; Mensink et al., 2013). They are the most similar to our proposed method and offer natural incremental additional and querying properties, but have not been leveraged in the deep learning-based continual learning literature (Ostapenko et al., 2022). Note that this direction is explored in works like Streaming LDA (Hayes & Kanan, 2020) and ExStream (Hayes et al., 2019). However, these approaches perform worse than partial training of a deep network (Hayes et al., 2020; Ostapenko et al., 2022). In contrast, we outperform fully finetuned deep networks with unrestricted access to past samples and far larger computational budgets.

The (online) continual learning methods designed for deep networks are typically based on experience replay (Chaudhry et al., 2019b) and change a subset of the three aspects summarized in Table 2: (i) the loss function used for learning, (ii) the algorithm to sample points into the replay buffer, and (iii) the algorithm to sample a batch from the replay buffer. Methods to sample points into the replay buffer include GSS (Aljundi et al., 2019b), RingBuffer (Chaudhry et al., 2019b), class-balanced reservoir (Chrysakis & Moens, 2020), greedy balancing (Prabhu et al., 2020), rainbow memory (Bang et al., 2021), herding (Rebuffi et al., 2017), coreset selection (Yoon et al., 2022), information-theoretic reservoir (Sun et al., 2022), and samplewise importance (Koh et al., 2022). These approaches do not apply to our setting because we simply remove the storage constraint. Approaches to sampling batches from the replay buffer include MIR (Aljundi et al., 2019a), ASER (Shim et al., 2021), and AML (Caccia et al., 2022). These require mining hard negatives or performing additional updates for importance sampling over the stored data, which are simply unscalable to large-scale storage as in our work. In our experiments, we compare with several of these approaches including ER as proposed in Cai et al. (2021) scaled appropriately to our setting. Additionally, our kNN

[1] Although we believe $\mathcal{O}(n \log n)$ complexity is not prohibitive for practical applications, a further reduction (i.e $\mathcal{O}(n \log \log n)$) can be obtained by carefully introducing additional levels of hierarchy for astronomically large n.

Table 2: Recent online continual learning approaches, with key contributions in red. Most methods focus on better techniques for sampling into storage, while in our framework there is no storage constraint.

Works	MemSamp	BatchSamp	Loss	Other Cont.
ER (Base)	Random	Random	CEnt	-
GSS (Aljundi et al., 2019b)	GSS	Random	CEnt	-
MIR (Aljundi et al., 2019a)	Reservoir	MIR	CEnt	-
ER-Ring (Chaudhry et al., 2019b)	RingBuf	Random	CEnt	-
GDumb (Prabhu et al., 2020)	GreedyBal	Random	CEnt	MR
HAL (Chaudhry et al., 2021)	RingBuf	Random	CEnt	HAL
CBRS (Chrysakis & Moens, 2020)	CBRS	Weighting	CEnt	-
CLIB (Koh et al., 2022)	ImpSamp	Random	CEnt	MR, AdO
CoPE (De Lange & Tuytelaars, 2021)	CBRS	Random	PPPLoss	-
CLOC (Cai et al., 2021)	FIFO	Random	CEnt	AdO
InfoRS (Sun et al., 2022)	InfoRS	Random	CEnt	-
OCS (Yoon et al., 2022)	OCS	Random	CEnt	-
AML (Caccia et al., 2022)	Reservoir	PosNeg	AML/ACE	-

based method offers attractive properties for OCL like never forgetting previously seen samples, not possible in most previous parametric approaches including the deep OCL methods presented above.

Pretrained Representations. Pretrained representations (Yuan et al., 2021; Caron et al., 2021; Chen et al., 2021; Ali et al., 2021) have been utilized as initializations for continual learning, but in settings with harsh constraints on memory (Wu et al., 2022; Ostapenko et al., 2022). Inspired by Ostapenko et al. (2022), we additionally explore effects of different pretrained representations along with comparison among traditional classifiers like logistic regression and online SVMs and discuss interesting findings. Another emerging direction for using pretrained models in continual learning has been prompt-tuning as it produces accurate classifiers while being computationally efficient (Wang et al., 2022b;a; Chen et al., 2023). However, Janson et al. (2022) show that simple traditional classification models outperform these complex prompt tuning strategies by significant margins.

Lastly, the direction most similar to ours is methods which use kNN classifiers alongside deep networks for classification (Nakata et al., 2022; Iscen et al., 2022). We operate in a very different setting with no storage constraints, online learning, illustrate the effectiveness of weaker pretrained classifiers trained on ImageNet1K when testing on large-scale datasets, and show that approximate kNN can achieve a high accuracy-performance tradeoff at scale. Additionally, Nakata et al. (2022) imposes restrictions on stored samples in compared methods but uses all past data, allowing comparatively higher performance using kNN.

3 Approach

We utilize pre-trained features as representations and k-nearest neighbor rule as a learning algorithm. Hence, our algorithm is rather simple. The key to operationalizing our algorithm is utilizing an efficient memory structure that satisfies cost constraints. We refer to our algorithm as Adaptive Continual Memory (ACM) and refer to the kNN index as Memory.

At each time step, our learner performs the following steps:

1. *One* data point $x_t \sim \pi_t$ sampled from a non-stationary distribution π_t is revealed.
2. Learner extracts features $z_t = f(x_t; \theta_t)$
3. Learner retrieves nearest neighbors $\mathcal{N}_t = \texttt{Memory.Retrieve}(z_t, k)$.
4. Learner makes the prediction $\hat{y}_t = \texttt{majority-vote}(\mathcal{N}_t)$.[2]
5. Learner receives the true label y_t.
6. Learner inserts new data: $\texttt{Memory.Insert}(z_t, y_t)$.

We summarize this approach in Figure 1. Before presenting further implementation details, we discuss two advantages of this method.

[2]We choose $k = 1$ but a larger k can be chosen.

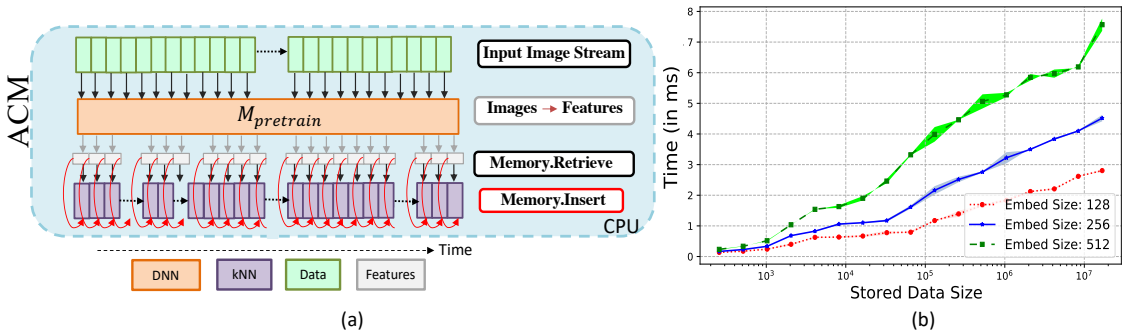

Figure 1: (a) Adaptive Continual Memory (ACM) performs `Memory.Retrieve` and `Memory.Insert` operations on features of new incoming samples, extracted by a static, pretrained deep network. (b) Wall clock time overhead of ACM Memory after feature extraction (x-axis is log-scaled) on a 16-core i7 CPU server. The longest observed overhead time using 256 dim embeddings is 5ms on 40 million samples in memory.

Fast adaptation. Suppose the learner makes a mistake in a given time step. If the same data point is received in the next time step, the learner will produce the correct answer. By leveraging nearest neighbors, we enable the system to incorporate new data immediately and locally modify its answers in response to as little as a single datapoint. Such fast adaptation, a core desideratum in online continual learning, is difficult with pure gradient descent and is not presently a characteristic of deep continual learning systems.

Consistency. Consider a hypothetical scenario in which a data point is queried at multiple time instances. Our learner will never forget the correct label for this data point and will consistently produce it when queried, even after long time spans. While learning and memory are much more general than rote memorization, producing the correct answer on previously seen data is an informative sanity check. For comparison, existing continual learning systems forget a large fraction of previously seen datapoints even with a minimal delay (Toneva et al., 2019).

3.1 Efficient Memory Implementation

In the presented algorithm above for our method, feature extraction (step 2) and prediction (step 4) have a fixed overhead cost. However, nearest-neighbour retrieval (step 3) and inserting new data (step 6) can have high computational costs if done naively. However, literature in approximate k-nearest neighbours (Shakhnarovich et al., 2006) has demonstrated a high performance while scaling down computational complexity from linear $\mathcal{O}(n)$ to logarithmic $\mathcal{O}(\log n)$, where n is the number of data points in memory. We leverage approximate kNN to allow our proposed approach to operate on the entirety of the data received so far under the constraint of logarithmic computational complexity, with approximate guarantees on preserving the above two discussed properties. We use the HNSW algorithm to give high accuracy in minimal time based on the benchmarking results of Aumüller et al. (2020).

Furthermore, Figure 1 presents the wall-clock time of the overhead cost imposed by ACM on datasets of 40 million samples to practically ground the logarithmic computational complexity. We observe that the computational overhead while using ACM scales logarithmically with the maximum time of ∼5ms. In comparison, the time required for classification of one sample for deep models like ResNet50 is ∼10ms on an Intel 16-core CPU. Note that while using ACM, the total inference cost of ACM inference would be 15ms considering the constant cost of feature extraction when using 40 million samples in storage.

4 Experiments

Datasets. We benchmark using a subset of Google Landmarks V2 and YFCC-100M datasets. Both ordered by timestamps of upload date-time, with the task being online image classification, predicting the label of the incoming image.

i) Continual YFCC100M (CLOC): The subset of YFCC100M, which has date and time annotations (Cai et al., 2021). We follow their dataset splits. We order the images by timestep and iterate over 39 million

online timesteps, one image at a time, with evaluation on the next image in the stream. In contrast, CLOC uses a more restricted protocol assuming 256 images per timestep and evaluates on images uploaded by a different user in the next batch of samples.

ii) Continual Google Landmarks V2 (CGLM) : We use a subset of Google Landmarks V2 dataset (Weyand et al., 2020) as our second benchmark. We use the train-clean subset, filtering it further based on the availability of upload timestamps on Flickr. We filter out the classes that have less than 25 samples. We uniformly in random sample 10% of data for testing and then use the first 20% of the remaining data as a hyperparameter tuning set, similar to CLOC. We get $430K$ images for continual learning with 10788 classes. We use the same hyperparameters as obtained on CLOC as Continual Learning algorithms should work with different data distributions.

Metrics. We follow Cai et al. (2021), using their average online accuracy until the current timestep k (a_k) as a metric for measuring rapid adaptation (forward transfer), given by $a_k = {}^1/_{N_t} \sum_{t=1}^{k} \mathbb{1}_{y_t=\hat{y}_t}$ where $\mathbb{1}_{(\cdot)}$ is the indicator function. Similarly, we measure information retention (preventing catastrophic forgetting) after finishing training by computing the average accuracy historically. Formally, information retention for i timesteps (IR_i) at time T is defined as $IR_i = {}^1/_i \sum_{s=T-i}^{T} \mathbb{1}_{y_t=\hat{y}_t}$

Approaches. We compare with a diverse range of methods, all of which are computationally capped to have the computational budget of one training pass over the CLOC dataset, termed as *fast stream* in the parallel work (Ghunaim et al., 2023). We take their top two performing methods and compare their performance on the CGLM dataset. We use the same hyperparameters as Ghunaim et al. (2023) for all methods unless specified otherwise, please refer to it for details about hyperparameters. Note that unlike Ghunaim et al. (2023) setup, methods can use the full set of stored samples with no storage restrictions. We use a batch incoming samples with a size of 64 for CGLM and 128 for CLOC for computational restrictions. The training batch size is double the size of the incoming samples, with the rest half of the batch being uniformly selected from all past stored data. Each resultant model is used for predicting the next 64/128 samples in CGLM/CLOC datasets respectively. We describe each method below:

i) ER (Cai et al., 2021): ER performs online continual learning using an learning rate of 0.0005. We use the vanilla version as reduction in the batch size is responsible for nearly all of the performance gain amongst components tested (PoLRS and ADRep).
iii) MIR (Aljundi et al., 2019a): This adds MIR as the selection mechanism for choosing samples instead of uniform, to train on for training the base ER model. However, it is used in a task-free fashion as there are no task boundaries in tested datasets.
iv) ACE (Caccia et al., 2022): This replaces the loss function in the base ER model from Crossentropy to ACE loss for reducing the interference of classes not present in the current batch. It is done in a task-free fashion as there are no task boundaries in tested datasets.
v) RWalk (Chaudhry et al., 2018): This adds a regularization term based on a combination of Fisher information matrix and the optimization-path based importance scores. We consider each incoming batch a new task as there are no specified task boundaries.

Alongside these existing methods, we evaluate two approaches that are far computationally cheaper as they involve no training.

vi) Blind Clf-k (Cai et al., 2021): This is a baseline classifier with no access to the images, predicting the label of the current datapoint as mode of the recent k datapoints with a memory requirement of k (k=1 for CGLM and k=25 for CLOC).
vii) ACM (Ours): ACM uses an XCIT DINO model pre-trained on ImageNet1K (Caron et al., 2021) with similar performance on ImageNet as the ResNet50-V2 model used in the above methods for fairness. We replace the FC layer with a two layer MLP, first projecting the features to a 256-dimensions and second layer performing classification. We train this two layer MLP on the hyperparameter tuning set for a few epochs. We extract the features from the 256-dimensional embedding space to avoid the curse of dimensionality in kNN. We choose HNSW-kNN based on the benchmarking results of Aumüller et al. (2020). We use NMSlib (Malkov & Yashunin, 2018), with default hyperparameters of k=1 (nearest neighbour), ef=200 and m=25 for rapid adaptation evaluation and use FAISS-based kNN for backward transfer evaluation.

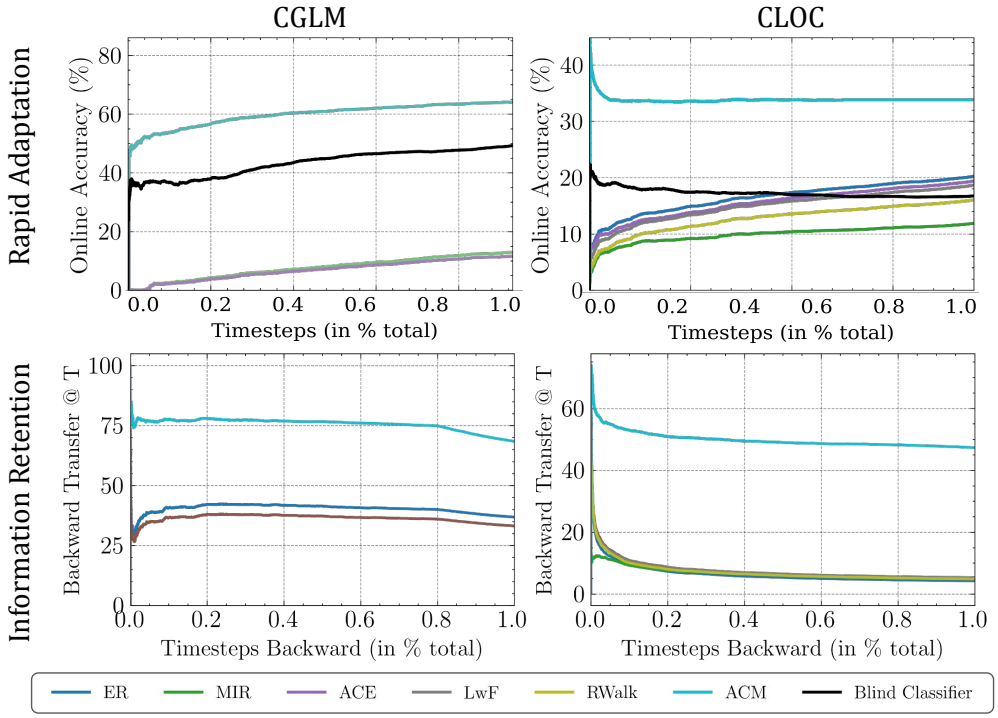

Figure 2: Performance on Rapid Adaptation (top) and Information Retention (Bottom) on CGLM (left) and CLOC (right) datasets: We observe that ACM outperforms existing methods by a large margin despite being far cheaper computationally. Traditional methods perform poorly given unrestricted access to past seen data indicating continual learning is a hard problem even without storage constraints.

4.1 Main Result: Evaluating ACM

Online adaptation. We compare the average online accuracy over time of ACM to current state-of-the-art approaches on CGLM and CLOC in Figure 2. We observe that ACM significantly outperformed past approaches, by enabling efficient learning with a memory based approach. Moreover, the pre-trained features are universal enough to enable high performance. Note that ACM has nearly no training compute cost compared to other methods resulting not only better accuracy but also a significant cost effectiveness.

Information retention. We compare the cumulative average performance of ACM to current state-of-the-art approaches on CGLM and CLOC in Figure 2. We observe that ACM outperforms existing methods on both datasets. Interestingly, ACM shows an nearly flat cumulative accuracy even over CLOC dataset with 39 million samples illustrating the benefits of utilizing past samples instead of encoding it in DNN parameters in backward transfer into history.

We notice comparing to Ghunaim et al. (2023), removing the memory restriction from 40000 samples did not significantly change performance in the methods listed below, indicating that online continual learning with limited computation is hard even without storage constraints.

Take-away messages. We achieve a significantly better tradeoff between rapid adaptation and information retention, illustrating the overwhelming benefit of storing information across time with a good initialization instead of trying to modify weights across time which causes catastrophic forgetting. Notably, it is surprising that ImageNet1K pretrained representations scale well for data streams like subsets of YFCC100M which are significantly bigger and more challenging than ImageNet1K (Goyal et al., 2019), while enabling rapid adaptation (in terms of sample complexity) to the distribution shifts over time.

4.2 Studying the ACM Model

Ablating the contribution of features from kNN. The main contribution of ACM is the adaptive memory formed using the kNN classifier. Here, we try to answer the question: *"Is the rapid adaptation property due to the proposed memory or due to the quality of the feature representations?"*.

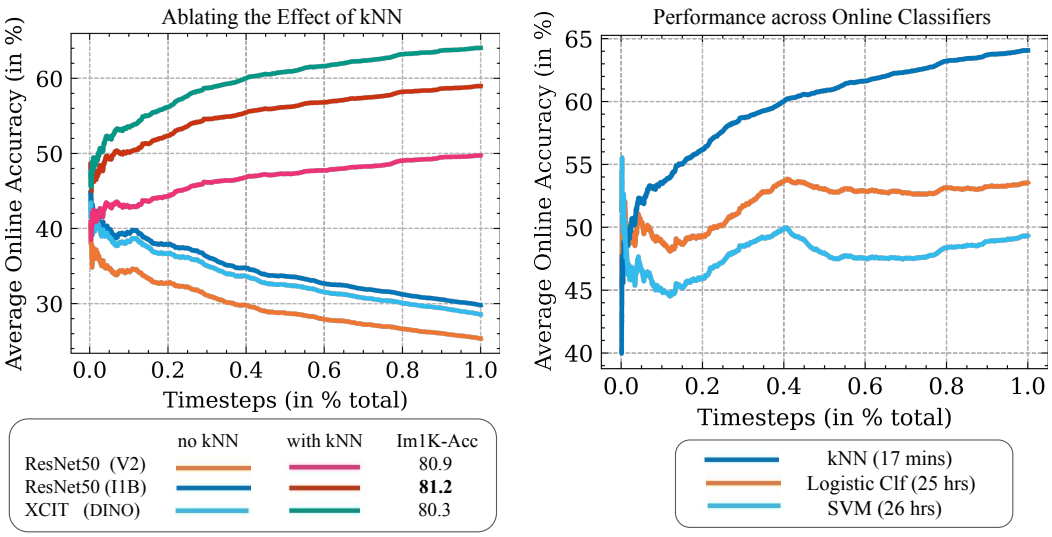

Figure 3: **Left:** contribution of kNN beyond the information encoded in feature representations by predicting using the linear classifier in the MLP instead of the kNN. The MLP classifier is frozen and only used for prediction. **Right:** comparison of kNN with other online classifiers in performance and speed. Online classifiers learn weights given the 256-dimensional features.

In order to test this, we use the classifier in the MLP as an alternative to classify images. This eliminates kNN from proposed ACM system with minimal changes. If kNN representations are the primary reason for the high performance then we should observe a significant decrease in online accuracy. We test this across three pretrained models with similar accuracy on ImageNet1K dataset: DINO (Caron et al., 2021), ResNet50 (V2) (Vryniotis et al., 2020) trained on ImageNet1K and a ResNet50 (I1B) trained on Instagram1B and finetuned on ImageNet1K (Mahajan et al., 2018) on the CGLM dataset.

We present our results in Figure 3 (left) which shows that using a linear classifier instead of a kNN for classification results in far lower performance. Interestingly, the linear classifier shows a downward drift, losing upward of 10% accuracy, attributable to to distribution shift in the dataset. On the contrary, the kNN performance improves over time consistently across various architectures. We see that XCIT performs significantly better when used with a kNN compared to ResNet50 (I1B) models despite being significantly worse than ResNet50 (I1B) when using a linear classifier in CGLM dataset and on ImageNet1K classification performance.

Conclusion. kNN is the primary reason for rapid adaptation to distribution shifts and is primarily responsible for the online learning performance, consistent across architectures. Simply having good feature representation is not enough to tackle online continual learning. Lastly, ResNet50 architecture is a poor fit for ACM.

Choice of kNN vis-a-vis other online classifiers. Now that we know that the online classifier is important for rapid adaptation, we study the choice of the online classifier. The motivation behind using kNN is that it avoids the failure modes in optimization that exacerbate catastrophic forgetting such as lack of consistency property. On the other hand, there are parametric alternatives. In this study, we ask the question: *how effective is kNN compared to parametric online classifiers like logistic regression or SVM?*

We compare the performance and time efficiency of online learning on the CGLM dataset using XCIT DINO features and compare kNN with widely used traditional models like logistic classifier and SVM, implemented using an efficient online learning library VowPalWabbit. We present results in Figure 3 (right). We see that kNN achieves significantly superior classification performance compared to popular online learning algorithms using the same features while being nearly two orders of magnitude faster, enabling use in large datasets like CLOC.

Conclusion. kNN is a powerful non-linear classifier which can act like a knowledge base, learning from relevant memories ranging from fresh mistakes to those indexed long ago. Moreover, it is incredibly fast, possibly due to efficient implementations.

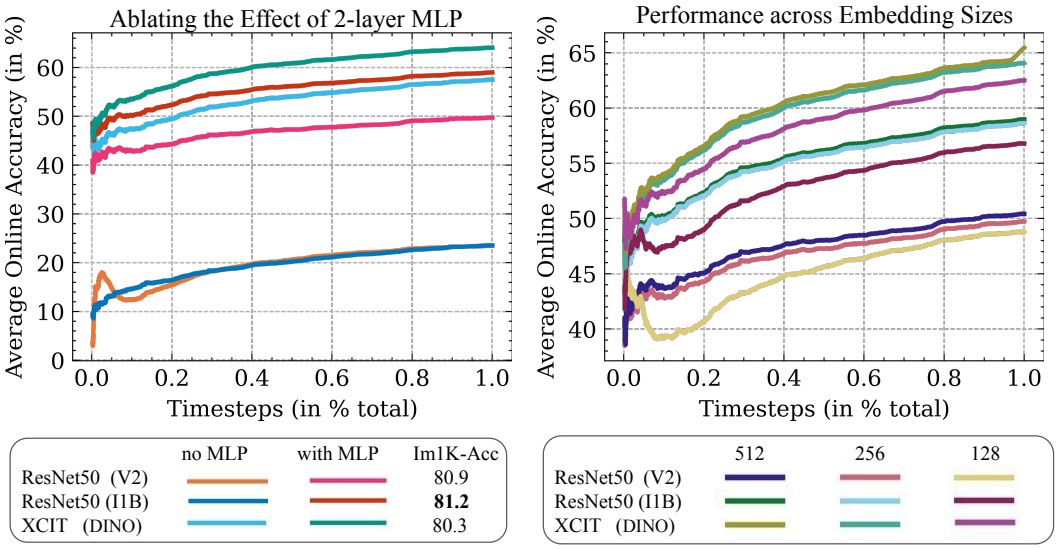

Figure 4: **Left:** ablating the effect of using representations from a learned MLP with off-the-shelf features. **Right:** comparing across small embedding dimensions to gain higher efficiency without losing accuracy.

Ablating the contribution of the 2-layer MLP. Finally, we ablate the contribution of the 2-layer MLP that was trained on the hyperparameter tuning set. We compare using the features before the FC layer from the network as-is instead of using the 256-dimensional embedding layer in the trained MLP. We present the results across the same three models in Figure 4 (left). Comparing the performance with and without the MLP, we observe that the performance in XCIT DINO model has a small drop of 5% illustrating that tuning on the hyperparameter set causes minor improvements in performance. However, both ResNet50 models face large drops of over 30% in online accuracy. We show that this is attributable to the curse of dimensionality, the large feature dimension of 2048 in ResNet50 architecture compared to 512 dimensions in XCIT causing a major drop in performance. Comparing models with MLP, we surprisingly observe that ResNet50-I1B performs worse than XCIT-DINO model despite ResNet50-I1B arguably has robust and generalizable features. We conclude that ResNet50 architecture is a poor fit for the ACM method.

Conclusion. Models with high-dimensional embeddings perform much more poorly in combination with a kNN despite better representational power due to the curse of dimensionality. We use the XCIT-DINO model instead of ResNet50-V2 despite poorer performance on ImageNet1K as ResNet50 models seem to be poorer fit with kNN despite measures to equalize performance.

Ablating the effect of the embedding size. Lastly, since embedding size is critical, we explore to what degree can we decrease the embedding dimensions without impacting performance significantly. Since XCIT features are 512 dimensional, we explore embedding sizes of 512, 256 and 128 and benchmark using the above three models for robust conclusions.

We present the results in Figure 4 (right). First, we observe that decreasing the embedding dimension to 256 results in minimal drop in accuracy across all the three models but reduces the computational costs by half as shown in Figure 1(b). Further reduction in embedding dimension leads to significant loss of performance, hence 256 dimensions achieves the best tradeoff.

5 Discussion

We would like to start our discussion with the limitations of our approach. It is important to state that our method is not applicable to many interesting application scenarios where pretrained features are not available or storage is truly limited, such as end-devices and embodied agents. Although we believe this is a strong limitation, we do not think it invalidates the importance and impact of the settings our method operates in. Our setting, along with its constraints, are applicable in a broad set of conditions, including but not limited to cloud-based systems using natural language and images, and we believe that studying online continual learning in these settings is important.

While memory restriction can also be motivated by privacy considerations (Farquhar & Gal, 2018), recent progress in machine unlearning suggests that simply preventing access to old data is inadequate to fulfill any reasonable privacy requirements (Cao & Yang, 2015). We believe that privacy-constrained continual learning should be studied with a dedicated problem definition using appropriate application domains and benchmarks.

Finally, we believe it is important to ground our system and its theoretical cost into a practical setting to guide practitioners and illustrate the applicability of our proposed system. Consider a real-time data stream over time. Given our representation size is 256 and type is float32, and CLOC has 39M datapoints, the total storage cost would be roughly, 40GB of storage ($256 \times 32 \times 39$M bytes) which would cost \$10 on a GCP cloud storage per year given 0.02\$ per GB per month rate. Moreover, the total computation time (inference and training cost) would support real-time operation at 30 frames per second without any additional optimization for 71 years, extrapolating with logarithmic scaling upto 20ms from Figure 1 (b).

6 Conclusion

This work considers online continual learning with no restrictions on storage. Our reformulation follows from a first-principles analysis of modern computing systems' economic and computational characteristics. We proposed an adaptive continual memory that stores the entirety of the data, performs per-sample adaptation at every timestep, and retains computational efficiency. When evaluated on large-scale OCL benchmarks, our system yields significant improvements over existing methods. Our approach is computationally cheap and scales gracefully to large-scale datasets.

7 Acknowledgements

This work is supportedin part by a UKRI grant: Turing AI Fellowship EP/W002981/1 and an EPSRC/MURI grant: EP/N019474/1. We would also like to thank the Royal Academy of Engineering and FiveAI. Ameya produced this work as part of his internship at Intel Labs. A special thanks to Hasan Abed Al Kader Hammoud for their help in experiments.

References

Alaaeldin Ali, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, et al. Xcit: Cross-covariance image transformers. *NeurIPS*, 2021.

Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Laurent Charlin, and Tinne Tuytelaars. Online continual learning with maximally interfered retrieval. In *NeurIPS*, 2019a.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *NeurIPS*, 2019b.

Raja Appuswamy, Renata Borovica-Gajic, Goetz Graefe, and Anastasia Ailamaki. The five-minute rule thirty years later and its impact on the storage hierarchy. In *ADMS@VLDB*, 2017.

Martin Aumüller, Erik Bernhardsson, and Alexander John Faithfull. Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Inf. Syst.*, 87, 2020.

Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. In *CVPR*, 2021.

Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New insights on reducing abrupt representation change in online continual learning. *ICLR*, 2022.

Zhipeng Cai, Ozan Sener, and Vladlen Koltun. Online continual learning with natural distribution shifts: An empirical study with visual data. In *ICCV*, 2021.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE symposium on security and privacy*, pp. 463–480. IEEE, 2015.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 532–547, 2018.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *ICLR*, 2019a.

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 *ICML-W*, 2019b.

Arslan Chaudhry, Albert Gordo, David Lopez-Paz, Puneet K. Dokania, and Philip Torr. Using hindsight to anchor past knowledge in continual learning, 2021.

Haoran Chen, Zuxuan Wu, Xintong Han, Menglin Jia, and Yu-Gang Jiang. Promptfusion: Decoupling stability and plasticity for continual learning. *arXiv preprint arXiv:2303.07223*, 2023.

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *ICCV*, 2021.

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

Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. In *ICCV*, 2021.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory G. Slabaugh, and Tinne Tuytelaars. Continual learning: A comparative study on how to defy forgetting in classification tasks. In *TPAMI*, 2020.

Sebastian Farquhar and Yarin Gal. Towards robust evaluations of continual learning. *arXiv preprint arXiv:1805.09733*, 2018.

João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys*, 2014.

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.

Heitor M Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfharinger, Geoff Holmes, and Talel Abdessalem. Adaptive random forests for evolving data stream classification. *Machine Learning*, 2017.

Priya Goyal, Dhruv Mahajan, Abhinav Gupta, and Ishan Misra. Scaling and benchmarking self-supervised visual representation learning. In *Proceedings of the ieee/cvf International Conference on computer vision*, pp. 6391–6400, 2019.

Goetz Graefe. The five-minute rule 20 years later (and how flash memory changes the rules). *Communications ACM*, 2009.

Jim Gray and Goetz Graefe. The five-minute rule ten years later, and other computer storage rules of thumb. *ACM SIGMOD*, 1997.

Jim Gray and Franco Putzolu. The 5 minute rule for trading memory for disc accesses and the 10 byte rule for trading memory for cpu time. In *ACM SIGMOD*, 1987.

Md Yousuf Harun, Jhair Gallardo, Tyler L Hayes, and Christopher Kanan. How efficient are today's continual learning algorithms? *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshop (CVPR-W)*, 2023.

Tyler L Hayes and Christopher Kanan. Lifelong machine learning with deep streaming linear discriminant analysis. In *CVPR-W*, 2020.

Tyler L Hayes, Nathan D Cahill, and Christopher Kanan. Memory efficient experience replay for streaming learning. In *ICRA*, 2019.

Tyler L Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting. In *ECCV*, 2020.

Hexiang Hu, Ozan Sener, Fei Sha, and Vladlen Koltun. Drinking from a firehose: Continual learning with web-scale natural language. *IEEE Transactions on Pattern Analysis and Machine Intelligence, to appear*, 2022.

Ahmet Iscen, Thomas Bird, Mathilde Caron, Alireza Fathi, and Cordelia Schmid. A memory transformer network for incremental learning. *arXiv preprint arXiv:2210.04485*, 2022.

Paul Janson, Wenxuan Zhang, Rahaf Aljundi, and Mohamed Elhoseiny. A simple baseline that questions the use of pretrained-models in continual learning. *arXiv preprint arXiv:2210.04428*, 2022.

Hyunseo Koh, Dahyun Kim, Jung-Woo Ha, and Jonghyun Choi. Online continual learning on class incremental blurry task configuration with anytime inference. *ICLR*, 2022.

Pavel Laskov, Christian Gehl, Stefan Krüger, Klaus-Robert Müller, Kristin P Bennett, and Emilio Parrado-Hernández. Incremental support vector learning: Analysis, implementation and applications. *JMLR*, 2006.

Zhiqiu Lin, Jia Shi, Deepak Pathak, and Deva Ramanan. The clear benchmark: Continual learning on real-world imagery. In *NeurIPS*, 2021.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *NeurIPS*, 2017.

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 *ECCV*, 2018.

Yu A Malkov and Dmitry A Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *TPAMI*, 2018.

Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *TPAMI*, 2013.

Jaouad Mourtada, Stéphane Gaïffas, and Erwan Scornet. Amf: Aggregated mondrian forests for online learning. *Journal of the Royal Statistical Society*, 2019.

Kengo Nakata, Youyang Ng, Daisuke Miyashita, Asuka Maki, Yu-Chieh Lin, and Jun Deguchi. Revisiting a knn-based image classification system with high-capacity storage. In *European Conference on Computer Vision (ECCV)*, 2022.

Oleksiy Ostapenko, Timothee Lesort, Pau Rodríguez, Md Rifat Arefin, Arthur Douillard, Irina Rish, and Laurent Charlin. Continual learning with foundation models: An empirical study of latent replay. In *Conference on Lifelong Learning Agents (CoLLAs)*, 2022.

Nikunj C Oza and Stuart J Russell. Online bagging and boosting. In *International Workshop on Artificial Intelligence and Statistics*. PMLR, 2001.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.

Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *ECCV*, 2020.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 2017.

Marko Ristin, Matthieu Guillaumin, Juergen Gall, and Luc Van Gool. Incremental learning of random forests for large-scale image classification. *TPAMI*, 2015.

Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. The MIT Press, 2006.

Dongsub Shim, Zheda Mai, Jihwan Jeong, Scott Sanner, Hyunwoo Kim, and Jongseong Jang. Online class-incremental continual learning with adversarial shapley value. In *AAAI*, 2021.

Shengyang Sun, Daniele Calandriello, Huiyi Hu, Ang Li, and Michalis Titsias. Information-theoretic online memory selection for continual learning. *ICLR*, 2022.

Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 2016.

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 *ICLR*, 2019.

Vasilis Vryniotis et al. How to train state-of-the-art models using torchvision's latest primitives. In *https://pytorch.org/blog/how-to-train-state-of-the-art-models-using-torchvision-latest-primitives/*, 2020.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision (ECCV)*, 2022a.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022b.

T. Weyand, A. Araujo, B. Cao, and J. Sim. Google Landmarks Dataset v2 - A Large-Scale Benchmark for Instance-Level Recognition and Retrieval. In *CVPR*, 2020.

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.

Jaehong Yoon, Divyam Madaan, Eunho Yang, and Sung Ju Hwang. Online coreset selection for rehearsal-based continual learning. *ICLR*, 2022.

Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432*, 2021.

Jun Zheng, Furao Shen, Hongjun Fan, and Jinxi Zhao. An online incremental learning support vector machine for large-scale data. *Neural Computing and Applications*, 2013.

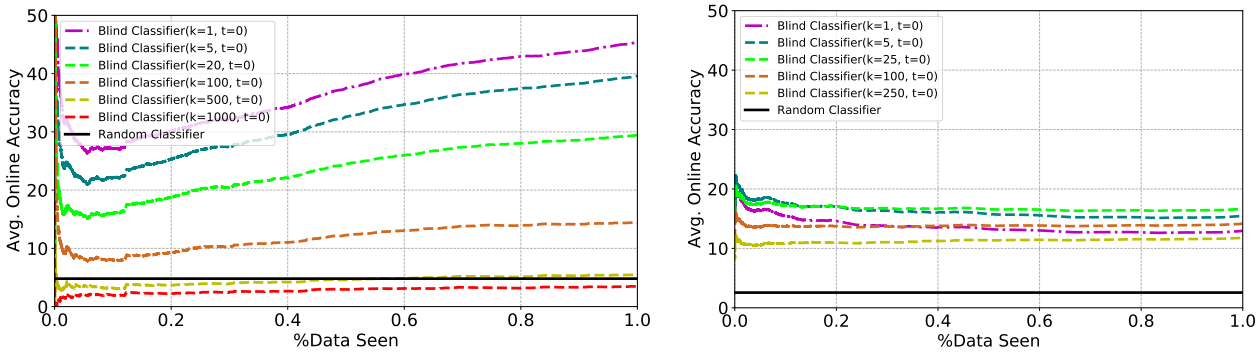

Figure 5: Variation of blind classifier through different k values on CGLM dataset (left) and CYFCC (right)

A Implementation Details

In this section we describe the experimental setup in detail: including dataset creation, methods and their training details.

A.1 Dataset Details

Continual Google Landmarks V2 (CGLM) (Weyand et al., 2020): We start with the train-clean subset of the GLDv2 available from the dataset website[3]. 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

We get the rest $580K$ images for continual learning over 10788 classes with large class-imbalance alongside the rapid distribution shift temporally. We allocate the first 20% of the dataset timestampwise for pretraining and randomly sample 10% of data from across time for testing.

We provide the scripts for cleaning the dataset alongside in the codebase.

Continual YFCC-100M (CYFCC) (Thomee et al., 2016): We download the images and the metadata as given by Cai et al. (2021) from their github repository. We provide a guide for downloading alongside in our codebase.

A.2 Model and Optimization Details

Training ACM: We use a 2-layer MLP with batchnorm and ReLU activations trained for 10 epochs on CGLM pretrain set and 2 epochs on the CLOC pretrain set. We do not do hyperparameter optimization and use the default parameters as hyperparameter optimization in online continual learning is an open problem. All ACM experiments were performed on one 48 GB RTX 6000 to extract features and the kNN computation was done on a 12th Gen Intel i7-12700 server.

Training other approaches: We use a ResNet50-V2 model from Pytorch for all other methods. We trained models starting from a pretrained ImageNet1K model with a batch size of 128 for CGLM and 256 for CLOC with a constant lr of 5e-3, SGD optimizer as specified in the original work. We used a 80GB A100 GPU server for the training.

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

A.3 Selection of k for Blind Classifiers

We selected the best k for evaluating performance of blind classifiers, for comparison with other approaches. We show the results in Figure 5, where we observe the best performing k values are $k = 1$ for CGLM dataset and $k = 25$ for CYFCC dataset. We observe that the variation of performance across k CGLM dataset is much sharper than in CYFCC dataset, implying the distribution changes much more quickly in the CGLM dataset.