The CLEAR Benchmark: Continual LEArning on Real-World Imagery

Zhiqiu Lin **Jia Shi** **Deepak Pathak*** **Deva Ramanan****
Robotics Institute
Carnegie Mellon University

Abstract

Continual learning (CL) is widely regarded as crucial challenge for lifelong AI. However, existing CL benchmarks, e.g. Permuted-MNIST and Split-CIFAR, make use of artificial temporal variation and do not align with or generalize to the real-world. In this paper, we introduce CLEAR, the first continual image classification benchmark dataset with a natural *temporal evolution of visual concepts* in the real world that spans a *decade* (2004-2014). We build CLEAR from existing large-scale image collections (YFCC100M) through a novel and scalable low-cost approach to *visio-linguistic dataset curation*. Our pipeline makes use of pretrained vision-language models (e.g. CLIP) to interactively build labeled datasets, which are further validated with crowd-sourcing to remove errors and even inappropriate images (hidden in original YFCC100M). The major strength of CLEAR over prior CL benchmarks is the smooth temporal evolution of visual concepts with real-world imagery, including both high-quality labeled data along with abundant unlabeled samples per time period for continual semi-supervised learning. We find that a simple unsupervised pre-training step can already dramatically boost state-of-the-arts CL algorithms that only utilize fully-supervised data. Our analysis also reveals that mainstream CL evaluation protocols that train and test on iid data artificially inflate performance of CL system. To address this, we propose a novel "streaming" protocol for CL that always test on the (near) future. Interestingly, streaming protocols (a) can simplify dataset curation since today's testset can be repurposed for tomorrow's trainset and (b) can produce more generalizable models with more accurate estimates of performance since *all* labeled data from each time-period is used for *both* training and testing (unlike classic train-test splits).

1 Introduction

Web-scale image recognition datasets such as ImageNet [50] and MS-COCO [35] revolutionized the field of machine learning and computer vision by becoming touchstones for the modern algorithms [13, 21, 24]. These benchmarks are designed to solve a *stationary* task where the distribution of underlying visual concepts is assumed to be same during train and test. However, in reality, most ML models have to cope with a *dynamic* environment as the world is changing over time. Figure 1 shows web-scale visual concepts that have naturally evolved over time in the last couple of decades. Although such dynamic behaviors are readily prevalent in web image collections [27], recent learning benchmarks as well as algorithms fail to recognize the temporally dynamic nature of real-world data.

That said, there exists a tremendous body of work on continual/lifelong learning, with the aim of developing ML models that can adapt to *dynamic* environments, e.g., non-iid data streams. Many algorithms [18, 28, 30, 34, 41, 44, 59] have been purposed to combat the well-known failure mode of catastrophic forgetting [17, 42, 55]. More recently, new algorithms and metrics [12, 38] have been introduced to explore other aspects of CL beyond learning-without-forgetting, such as, the ability to

*Equal advising. CLEAR benchmark page: https://clear-benchmark.github.io

35th Conference on Neural Information Processing Systems (NeurIPS 2021) Track on Datasets and Benchmarks.

Figure 1: **Temporal evolution of visual concepts in Internet images.** We show the evolution of three concepts (`computer`, `camera`, and `cosplay`) from the Flickr YFCC100M with timestamps spanning 2004 to 2014. The industry advanced rapidly over this decade from Canon EOS 30D (2006) to Canon EOS 6D (2013), and from Apple Powerbook G4 (2004) to Macbook Pro (2011) with substantial design changes. The definition of visual concepts also expanded, e.g., the common usage of the term `camera` evolved from standalone ones to ones in smartphones (iphone 5 in last image of second row). We see evolution in other visual concepts such as `cosplay` as well: Cosplayers often dress as topical characters of the day. From 2004-2007, popular characters reflect anime such as `Rayman Rabbit (2006)`, `Rebuild of Evangelion (2007)`, etc. while 2011-2014 includes characters such as `Assassin's Creed II (2010)`, `Steins;Gate (2011)` and so on.

transfer knowledge across tasks [12, 38] as well as to minimize model size and sample storage [12]. However, all of the above works were evaluated on datasets and benchmarks with synthetic temporal variation because it is easier to artificially introduce new tasks at arbitrary timestamps. Such continual datasets tend to contain *abrupt* changes:

- *Pixel permutation* (e.g., Permuted MNIST [18]) fully randomizes the positions of pixels at discrete intervals to form new tasks.
- *Incremental scenarios on static datasets* (e.g., Split-MNIST [59] and Split-CIFAR [29, 47]) split existing datasets designed for "static" evaluation to multiple tasks, each with a random subset of classes.
- *New Instances (NI) learning* (e.g., CORe50 [36]) adds brand new training patterns to existing classes. Yet these new patterns are all artificially designed, e.g., whoever collect the images manually change the illumination and background of the captured objects.
- *Two-task transfer* (e.g., Li and Hoiem [34]) setup trains a single model on a pair of datasets consecutively, such as ImageNet [50] followed by Pascal VOC [15], with the aim of preserving the performance on the first dataset while training on the second one.

Why are abrupt changes undesirable? Simulated continual data with abrupt changes is not only unnatural but may make the problem harder than need be. The real-world tends to exhibit *smooth* evolution, which may enable out-of-distribution generalization in biological entities. Indeed, synthetic benchmarks have been criticized [5, 16, 36] for their artificial abruptness because (a) knowledge accumulated on past tasks cannot be generalized to future tasks and (b) degenerate solutions that train "from scratch" on new tasks (such as GDumb [44]) still perform quite well. Aware of these criticisms, we propose this new CL benchmark to promote natural and smooth distribution shifts, and experiments in this paper confirm that GDumb [44] falls short compared with other baselines.

The CLEAR Benchmark. In this work, we propose the CLEAR Benchmark for studying Continual LEArning on Real-World Imagery. To our knowledge, CLEAR is the first continual image recognition benchmark based on the natural temporal evolution of visual concepts of Internet images. We curate CLEAR from the largest available public image collection (YFCC100M [53]) and use the timestamps of images to sort them into a temporal stream spanning from 2004 to 2014. We split an iid subset of the stream (around 7.8M images) into 11 equal-sized "buckets" with 700K images each. The labeled portion of CLEAR is designed to be similar in scale to popular ML benchmarks such as CIFAR [29]. For each of the bucket 1^{st} to 10^{th}, we curate a small labeled subset consisting of 11 temporally dynamic classes (10 illustrative classes such as `computer`, `cosplay`, etc. plus an 11^{th} `background` class) with 300 labeled images per class. Besides high-quality labeled data, the rest of the images per bucket in CLEAR can be viewed as large-scale free unlabeled data, which we hope will spur future research on continual semi-supervised learning [20, 45, 52, 58].

A low-cost and scalable dataset curation pipeline. However, constructing such a benchmark a natural continuity is non-trivial at a web-scale, e.g., it takes weeks to even download all images of

YFCC100M [53]. To annotate CLEAR in an efficient manner, we propose a novel visio-linguisitic dataset curation method. The key is to make use of recent vision-language models (CLIP [46]) with text-prompt engineering followed by crowdsourced quality assurance (i.e. MTurk). Our semi-automatic pipeline makes it possible for researchers to efficiently curate future datasets out of massive image collections such as YFCC100M. With this pipeline, we curate CLEAR with merely one day of engineering effort, and believe it can be easily scaled up to orders-of-magnitude more data.

A realistic "streaming" evaluation protocol for CL. Realistic temporal evolution encourages smooth transitions while suggestive of a more natural *streaming* evaluation protocol for CL, inspired by evaluation protocols for online learning [4, 14]. In essence, one deploys a model trained with present data at some point in the future. Interestingly, traditional CL evaluation protocols train and test on iid data buckets, failing to model this domain gap. We conduct extensive baseline experiments on CLEAR by simply fixing the label space to be the same 11 classes across time (i.e., the incremental domain learning setup [22]), and preliminary results confirm that mainstream (train-test) "iid" evaluation protocols artificially inflate performance of CL algorithms. Our streaming protocol can (a) simplify dataset curation since today's testset can be repurposed for tomorrow's trainset and (b) produce more generalizable models with more accurate estimates of performance since all labeled data from each time-period is used for both training and testing.

Large-scale unlabeled data boosts CL performances. Moreover, we find that unsupervised pre-training (MoCo V2) on the first bucket 0^{th} (around 700K images) of CLEAR can boost the performance of all state-of-the-arts CL techniques only making use of labeled data. In particular, finetuning a linear layer on top of those MoCo pre-trained features with naive rehearsal strategy (reservoir sampling) already surpasses all popular CL methods by a large margin. This suggests that future works on real-world CL should embrace large-scale unlabeled data to maximize performances.

2 Background and Related Work

2.1 Existing CL Datasets and Benchmarks

Most established works on CL focus on overcoming catastrophic forgetting [7, 18, 28, 30, 31, 33, 41, 44, 56, 59] through replay-based, regularization-based, distillation-based, and architecture-based methods, and we refer readers to [10, 43, 44] for surveys and overviews. However, these algorithms are all evaluated on CL benchmarks with synthetic temporal evolution like Permuted-MNIST [18], CoRE50 [36], and other incremental learning scenarios [10, 29, 40, 59]. We echo the sentiment from Sambasivan et. al [51] that solid datasets are vital for progress on new problems; especially, the growing field of continual and lifelong learning is in dire need of more practical benchmarks. One notable exception to contrived incremental scenarios is the recent work of Hu et. al [23], who assemble a CL dataset of Tweet messages that naturally evolve over time. A concurrently developed CL dataset [4] also utilize the timestamps of YFCC100M [53] images along with geostamps to construct a benchmark on image localization, whereas we focus on classification of temporally dynamic visual concept.

2.2 Continual Learning Settings

As CL is a board field of learning with an over-arching goal to develop algorithms that can adapt to non-iid data streams, many different CL settings have been proposed and tackled in prior works. In this section we explain some of the major CL settings that CLEAR adopts and refer readers to [1, 22, 44, 54, 60] for more thorough discussion.

Task-based sequential learning: In most CL works, *task-based sequential learning* [44] (or called "boundary-aware" CL [33]) is assumed, in which case a sequence of distinct tasks with clear task boundary is given. The tasks are iterated in a sequential fashion, and each time only the current task data is available. This setting is easy to set up (e.g., Split-MNIST) and has spurred many classic CL algorithms such as EWC [28] and SI [59] that heavily rely on task boundaries in order to know when to perform core actions such as knowledge consolidation, usually at the end of each task. In this paper, we also adopt task-based sequential learning with a sequence of (same) 11-way classification tasks by splitting the temporal stream into 11 buckets, each consisting of a labeled subset for training and evaluation. However, it could be argued that in real-world, the model will not be informed about the task boundary (also called boundary-agnostic [33], task-free[1], or task-agnostic CL [60]).

Such boundary-agnostic settings have been explored in recent works [1, 4, 23, 60], in which a data stream continuously spits out new samples without a notion of task switch. In this paper, we assume a task-based sequential learning setting to ease benchmark design, but future works could adapt CLEAR to boundary-agnostic or task-free CL by processing data in an online streaming fashion using timestamps of CLEAR images.

Locally-iid assumption: Following the task-based sequential learning setup, almost all prior works adopt a "locally-iid" assumption that each task's data is from an iid distribution [38]. In fact, mainstream CL evaluation protocols that sample training and testing data from this same iid distribution cannot be justified without this rather simplified assumption. In this work, we propose a novel "streaming" evaluation protocol that tests the current model on the data of next task, which does not implicitly assume that each task has its own iid distribution. Still, we can adopt the locally-iid assumption for CLEAR and evaluate with the mainstream "iid" protocol; therefore, we report comprehensive results under both "iid" and "streaming" protocols in this work.

Incremental task/domain/class learning: [22, 54] categorize existing CL setups for task-based sequential learning into three incremental learning scenarios. In incremental task learning, the task identity of each test sample is *known*; such a-prior knowledge could be exploited to ease algorithmic design, such as training a separate classification head for each task and switching the head based on the task identity. Incremental domain and class learning are more challenging since the task identity is *unknown* during test time. We adopt incremental domain learning in this work by fixing the label space for all tasks (same 11-way classification) and only the input distribution is changing. CLEAR can be adapted for incremental class learning in future by assigning each class to different tasks, i.e., making a growing label space.

Online v.s. offline continual learning: To encourage quick model adaption in CL, some prior works on task-based sequential learning require each sample to be used just once for model update. This setting is called online CL [40, 44] since it mimics an online stream of data, in which samples are spitted out one by one and cannot be revisited except when it is stored in a memory buffer (usually of limited size). On the other hand, offline CL assumes all samples in current task (plus the ones in buffer) can be revisited without constraint. However, some well-known online CL methods, for example GEM [38], use each sample only once but solve an expensive quadratic program per model update. Therefore, online CL does not directly imply quick model adaption unless one carefully studies the total resource consumption [44]. Instead, offline CL is still adopted in this work, but we allocate roughly the same resources (e.g., buffer size and training time) for each baseline algorithm for fair comparison. Note that [4] also name their own learning setup as *online continual learning*, though it is more similar to our "steaming" protocol. In both our setup and theirs, new incoming data is first used for evaluation and then added to the training set. In this paper, we name our protocol "streaming" in order to avoid notation clash since "online" CL has been well defined in prior works [40, 44].

2.3 Building Blocks for CLEAR

YFCC100M: We build CLEAR from YFCC100M [53], consisting of media artifacts uploaded to `www.flickr.com` from 2004 to 2014. YFCC100M's massive scale (over 100 million images and videos) and the wealth of metadata (timestamps for capture and upload date, GPS, user tag, image description, camera specs, *etc.*) makes it one of the largest and richest publicly-available image datasets. However, it is cumbersome to work with as downloading can take months and user-uploaded hashtags and descriptions can be extremely noisy and useless [3, 25, 31, 39], arguably limiting its impact compared to relatively smaller, curated data like ImageNet [50] and MS-COCO [35]. Nonetheless, YFCC100M consists of real-world and more complex imagery unlike other popular datasets favoring only centered objects or visual elements, e.g., MNIST [11], CIFAR [29], and ImageNet [50]. We believe it is more practical to develop algorithms and models on real-world data such as YFCC100M.

CLIP: CLIP [46] is a vision-language model that learns to associate texts and images by training on a massive dataset of over 400M image-text pairs. We use CLIP to automatically retrieve subsets of YFCC100M most relevant to particular visual concepts (Figure 2).

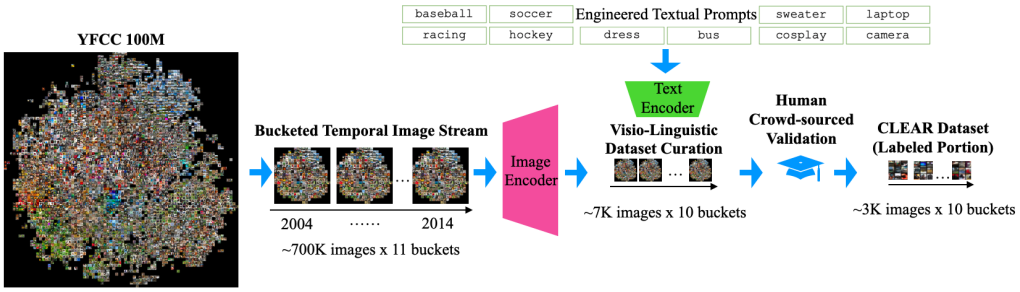

Figure 2: **Visio-Linguistic Dataset Curation.** We download a random subset of 7.8M images and their associated metadata from YFCC100M. We use upload timestamps to reconstruct a temporal stream, splitting it into 11 time-indexed buckets of roughly 700K images each. Given a list of text queries (found by text-prompt engineering in order to effectively retrieve the visual concepts of interest), we use CLIP [46] to extract their respective L2-normalized query features, ranking each image by the cosine similarity of its image feature to each query. We assign the top-ranked (0.7K out of 700K) images of each bucket to the query, removing ambiguous images that rank high across multiple queries. We also include a *background* class with images that rank low across the queries (details in Sec 1. of supplement). We then use human crowd-sourcing (MTurk) to remove misclassified and inappropriate images from the CLIP-retrieved images. As a result, CLEAR contains 3.3K high-quality labeled images for 10 buckets (excluding bucket 0^{th} for unsupervised pre-training only). Our dataset curation pipeline reduces the annotation cost by 99%.

3 CLEAR: Dataset Design and Curation

We describe how we curate CLEAR from YFCC100M [53] and CLIP [46]. Because YFCC100M is too large (the metadata text files already exceed 40 GBs) to gather and annotate, we believe our dataset curation procedure can benefit future benchmark creation, as this pipeline can be managed without massive infrastructure or engineering efforts from big organizations. We summarize the entire pipeline in Fig. 2.

Concept selection: We select temporally dynamic visual concepts from following super-categories:

- **Trends and Fashion**: People's aesthetics and interests shift over time. A fashionable dress in 2004 may be deemed outdated in 2014. Similarly, the clothing style for popular music performers were also changing, e.g. punk style in 1970s and Kpop idols in 2010s.
- **Consumer products**: Industry is constantly producing new commercial products to meet shifting consumer needs, e.g., models of vehicles, cameras, laptops, and cellphones change routinely.
- **Social Events**: Social/multimedia events are often dated and evolving, e.g., the FIFA world cup features different themes every 4 years. Cosplayers tend to dress as topical characters of-the-day.

We choose 10 dynamic visual concepts that span the above super-categories: `computer`, `camera`, `bus`, `sweater`, `dress`, `racing`, `hockey`, `cosplay`, `baseball`, and `soccer`. Please refer to supplement for a detailed discussion of these visual concepts and see examples in Fig. 1.

Stream recreation: To recover the temporal evolution of visual concepts from YFCC100M, we sort images of YFCC100M by their uploaded date to recreate the temporal stream of Flickr images from 2004 to 2014. In the interest of time, we only downloaded the first 7.8M images offered by their metadata file which is a random subset of YFCC100 images. We then chunk the uploading stream to 11 buckets of 700K images each, indexed from 0^{th} to 10^{th}. We gather a small set of 3.3K labeled data for buckets 1^{st} to 10^{th} with the same set of 10 dynamic visual concepts plus a 11^{st} `background` class. The 0^{th} bucket does not have a labeled set because we only use it to pretrain an unsupervised MoCo V2 model [9] for feature extraction in subsequent buckets (Sec. 5).

Visio-linguistic curation: We work on each of the 10 buckets of the temporal image stream independently to gather 10 labeled subsets consisting of the above dynamic visual concepts. Since it would be too costly to manually label all 700K images in each bucket, we use CLIP [46] to facilitate the annotation process. Given a pair of (image, text query), CLIP first encodes both image and query to two normalized features of same dimension (1024), and then performs a dot product between the two features to calculate a cosine similarity score. It has been shown [46] that the higher the score is, the more aligned the image content is to the query content. We then retrieve images with top 0.1% cosine similarity scores with respect to each given query in order to filter out most irrelevant images. As suggested in [46], one can refine textual queries to better capture a visual concept. In particular, we found it useful to enumerate *sub*category queries to assemble final datasets (e.g., use `laptop`

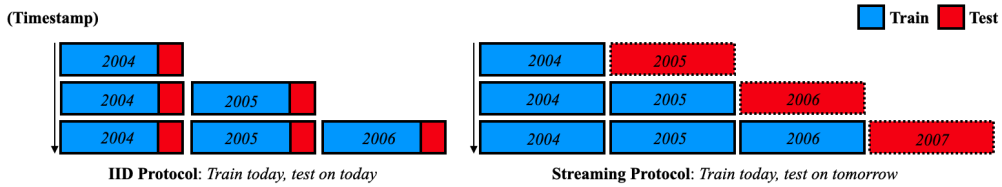

Figure 3: **IID vs Streaming Protocols for CL.** Traditional CL protocols (**left**) split incoming data buckets into a train/test split, typically 70/30%. However, this may overestimate performance since the train and test data are drawn from the same iid distribution, ignoring the train-test domain gap. We advocate a streaming protocol (**right**) where one must always evaluate on near-future data. This allows us to repurpose today's testset as tomorrow's trainset, increasing the total amount of recent data available for both training and testing. Note that the streaming protocol naturally allows for asynchronous training and testing; by the end of year 2006, one can train a model on data up to 2006, but needs additional data from 2007 to test it.

queries to retrieve `computer` images). Please refer to supplement (Sec. 1) for more details. Finally, to assemble `background` images, we construct a set of images that are consistently low-scoring across all queries. Details about how `background` class is constructed are in Sec. 1 of supplement.

Crowd-sourced validation (Mturk): We find that CLIP still produces a roughly 20% misclassification rate (though this varies by class). We use Amazon Mturk for crowd-sourced validation to filter out misclassified images. Each image in our dataset is verified by at least 3 workers. Additionally, we also use MTurk workers to mark any images with inappropriate content. Our pipeline successfully surfaced pornographic images contained in YCFF100M (that we have removed from CLEAR and subsequently reported to the original benchmark curators). Sec. 1 of supplement explains in details how we design the MTurk user interface and compose the worker results. Examples of images in our final dataset can be found in Sec. 1 of supplement.

4 Evaluation Protocols for Continual Learning

When presenting results on CLEAR, we make use of standard CL evaluation protocols to align with past works relying heavily on "locally-iid" assumption; many of them focus on an "iid" evaluation that examines performance on test samples from the same iid distribution of training samples. Instead, we advocate on a "streaming" perspective that evaluates on test data from future, motivated by real-world deployments that notoriously struggle with domain shifts between future test data and past training data. Moreover, such a streaming perspective *simplifies* dataset curation since *today's testset can be repurposed for tomorrow's trainset*. We will show that this translates to both improved models and more robust estimates of performance, since instead of making a classic 70/30% train-test split, we can use *all* labeled data of a bucket for both training and testing. Before we formalize our streaming evaluation protocol, we first review mainstream "iid" evaluation protocols for CL.

4.1 Review of IID Protocols

Following the "locally-iid" assumption, we have N timestamps with a sequence of unknown distributions $\mathcal{D} = \{D_1, D_2, \cdots, D_N\}$ with $D_i = \mathbf{X}_i \times \mathbf{Y}$, where $\mathbf{X}_i \subset \mathbf{X}$ is the input space at timestamp i and $\mathbf{Y}$ is the label space. For CLEAR, $\mathbf{X}_i$ is the image distribution from which bucket i is sampled and $\mathbf{Y}$ contains the 11 dynamic classes. The standard CL evaluation protocols then make a train-test **iid** assumption: Each task consists of a training set Tr_i and a test set Te_i sampled from the same distribution D_i at each timestamp. A learner then proceeds by sequentially fitting N predictor functions $\{h_1, h_2, \cdots, h_N\}$ on the N training sets. Each predictor function $h_i : \mathbf{X} \to \mathbf{Y}$ will be evaluated on all N test sets to generate an *accuracy matrix* $\mathcal{R} \in [0, 1]^{N \times N}$: $\mathcal{R}_{i,j}$ is defined to be the test accuracy of h_i on Te_j. The above formulation can be extended to a label space that grows over time to accommodate new classes (i.e., incremental class learning). However, for simplicity, we focus on "incremental domain learning" (all tasks share a fixed output space while only the input domain is changing), leaving "incremental class learning" (output space is changing as well) on CLEAR as future work. Note that these taxonomies are summarized in [22].

The standard iid evaluation protocols of CL algorithms mostly adopt the following 3 metrics, which can be readily calculated from the accuracy matrix $\mathcal{R}$:

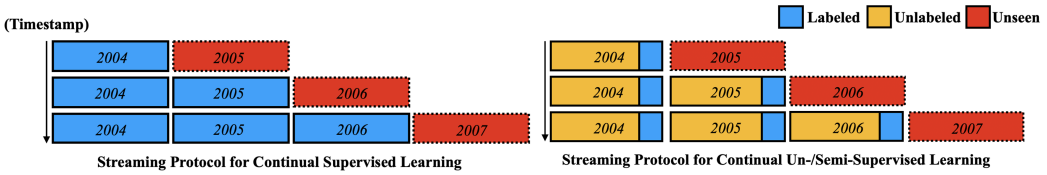

Figure 4: **Streaming Protocols for Continual Supervised vs. Un-/Semi-supervised Learning.** We compare streaming protocols for continual supervised and un-/semi-supervised learning. In real world, most incoming data will not be labeled due to the annotation cost; it is more natural to assume a small labeled subset along with large-scale unlabeled samples per time period. In this work, we achieve great performance boosts by only utilizing unlabeled samples in the first time period (bucket 0^{th}) for a self-supervised pre-training step. Therefore, we encourage future works to embrace unlabeled samples in later buckets for continual semi-supervised learning.

1. **In-domain Accuracy** (termed Average Accuracy in [12, 38]) measures the test accuracy on the current task immediately after training on it (averaged over all timestamps). This can calculated as the average of the diagonal entries of $\mathcal{R}$.
2. **Backward Transfer** measures the performance of previous tasks (i.e., learning without forgetting) by averaging lower triangular entries of $\mathcal{R}$.
3. **Forward Transfer** measures performance of future tasks (i.e., generalizing to future) by averaging upper triangular entries of $\mathcal{R}$.

We refer readers to Sec. 3 of supplement for the equations we used to calculate these metrics.

4.2 Our Streaming Protocol

We argue that the in-domain accuracy used in prior work is based on an unrealistic assumption that train and test data are iid. In real-world deployment scenarios, one must train on today's data and test on tomorrow's, introducing an undeniable domain shift, as demonstrated in Fig. 3.

A more realistic CL scenario is therefore to evaluate on the immediately *next* time period. Formally, we define a "streaming" evaluation protocol for CL: Given N timestamps, we have a stream of data $S = \{S_1, S_2, \cdots, S_N\}$; S_i could be a single or a bucket of samples, sampled from a non-stationary distribution $D_i = \mathbf{X_i} \times \mathbf{Y}$. In CLEAR, the stream S is the 10 buckets of labeled data (bucket 1^{st} to 10^{th}). Then a learner sequentially fits N predictor functions $\{h_1, h_2, \cdots, h_N\}$: At i_{th} timestamp, h_i is trained with the entire S_i. We call this protocol "streaming" because after training on S_i, we evaluate h_i on S_{i+1}, which are samples of the next timestamp. Once evaluation is done, S_{i+1} can be repurposed as the new trainset for fitting the next predictor h_{i+1}. This is similar to online learning protocols [4, 14], because the current predictor h_i will first make prediction on S_{i+1}, and the environment then reveals ground-truth labels of S_{i+1} for evaluation and model update.

Under streaming protocol, a similar accuracy matrix $\mathcal{R}$ can be defined: $\mathcal{R}_{i,j}$ is the accuracy of h_i on S_j. We then define *"next-domain"* accuracy to be the average accuracy of current predictor evaluated on the data of next timestamp (i.e., accuracy of h_i on S_{i+1}). This translates to the average of *superdiagonal* of $\mathcal{R}$:

$$\textbf{Next-domain Accuracy} = \frac{\sum_{i=1}^{N-1} \mathcal{R}_{i,i+1}}{N-1} \tag{1}$$

Our streaming protocol makes use of much more data for both testing and training: instead of slicing up data into a 70-30% train-test split, we can use 100% of data from a time period for testing (when first encountered) and 100% of that data for training (after time moves forward). This may result in better models and more accurate (e.g., lower variance) estimates of performance [55]. The price we pay is we can no longer evaluate on historical tasks for "learning-without-forgetting" since there is no longer any held-out test data. But from a truly streaming perspective, evaluating an "outdated" task may not be as relevant as more accurate models and more robust performance estimates of the task-at-hand.

We stress that CLEAR can be evaluated under both iid and streaming protocols (of which we are aware). We perform exhaustive experiments on both protocols, but highlight the latter because it has been historically underexplored. Especially, the domain shift between future test data and past training data is usually the bottleneck in real-world continual deployments.

Network	Unsup Repr.	Sampling Strategy	Method	IID Protocol					Streaming Protocol	
				In-domain Acc	Next-domain Acc	Acc	BwT	FwT	Next-domain Acc	FwT
ResNet18	-	N/A	EWC [28]	76.6% ± .2%	74.3% ± .6%	76.7% ± .3%	76.5% ± .4%	71.1% ± .6%	77.1% ± .6%	74.4% ± .6%
ResNet18	-	N/A	SI [59]	76.0% ± .2%	73.6% ± .2%	76.0% ± .5%	76.0% ± .6%	71.0% ± .4%	76.9% ± .2%	74.3% ± .2%
ResNet18	-	N/A	LwF [31]	77.8% ± .3%	75.7% ± .3%	79.2% ± .3%	79.6% ± .3%	72.5% ± .3%	78.8% ± .2%	76.1% ± .3%
ResNet18	-	N/A	CWR [36]	69.5% ± .2%	67.8% ± .3%	68.9% ± .3%	68.8% ± .3%	66.6% ± .3%	71.1% ± .4%	69.9% ± .3%
ResNet18	-	GDumb [44]	GDumb [44]	66.0% ± .4%	64.3% ± .5%	68.4% ± .4%	68.9% ± .4%	61.4% ± .5%	67.4% ± .1%	64.9% ± .1%
ResNet18	-	ER [49]	ER [49]	77.3% ± .1%	75.6% ± .3%	79.0% ± .1%	79.3% ± .1%	72.4% ± .2%	78.1% ± .2%	75.8% ± .2%
ResNet18	-	Reservoir	AGEM [6]	76.2% ± .3%	73.6% ± .2%	75.9% ± .2%	75.9% ± .3%	70.7% ± .2%	77.4% ± .2%	74.5% ± .2%
ResNet18	-	Reservoir	Finetuning	69.5% ± .3%	67.7% ± .2%	70.0% ± .2%	70.0% ± .1%	66.5% ± .2%	71.6% ± .2%	70.6% ± .2%
ResNet18	-	Biased Reservoir	Finetuning	75.5% ± .2%	72.7% ± .3%	75.7% ± .2%	75.8% ± .2%	70.2% ± .2%	77.2% ± .3%	74.4% ± .2%
Linear	YFCC-B0	N/A	EWC [28]	90.4% ± .3%	89.4% ± .4%	90.6% ± .3%	90.6% ± .3%	87.2% ± .2%	90.1% ± .2%	88.1% ± .0%
Linear	YFCC-B0	N/A	SI [59]	90.4% ± .3%	89.3% ± .3%	90.5% ± .3%	90.6% ± .3%	87.3% ± .2%	90.0% ± .0%	88.0% ± .0%
Linear	YFCC-B0	N/A	LwF [31]	90.3% ± .2%	89.2% ± .3%	90.9% ± .3%	91.0% ± .3%	87.3% ± .2%	90.0% ± .0%	88.1% ± .0%
Linear	YFCC-B0	N/A	CWR [36]	89.4% ± .3%	88.4% ± .3%	89.8% ± .3%	89.9% ± .3%	86.6% ± .2%	89.1% ± .0%	87.5% ± .0%
Linear	YFCC-B0	GDumb [44]	GDumb [44]	82.6% ± .4%	82.2% ± .5%	82.4% ± .8%	82.4% ± .8%	82.1% ± .3%	83.3% ± .1%	82.8% ± .1%
Linear	YFCC-B0	ER [49]	ER [49]	90.5% ± .3%	89.8% ± .4%	91.0% ± .3%	91.0% ± .3%	87.9% ± .2%	90.4% ± .1%	88.5% ± .1%
Linear	YFCC-B0	Reservoir	AGEM [6]	90.6% ± .2%	89.5% ± .2%	90.9% ± .3%	91.0% ± .3%	87.4% ± .2%	90.3% ± .0%	88.3% ± .0%
Linear	YFCC-B0	Reservoir	Finetuning	88.0% ± .2%	87.3% ± .3%	88.9% ± .3%	89.1% ± .3%	85.9% ± .3%	88.2% ± .2%	86.8% ± .1%
Linear	YFCC-B0	Biased Reservoir	Finetuning	90.2% ± .2%	89.1% ± .2%	90.5% ± .3%	90.6% ± .3%	87.2% ± .2%	89.8% ± .0%	88.0% ± .0%

Table 1: **Results of baseline CL algorithms on CLEAR.** We evaluated a variety of SOTA algorithms under both **IID** and **Streaming Protocols**. Under the classic **IID Protocol**, **In-domain Acc** (avg of diagonal entries of $\mathcal{R}$) is consistently larger than **Next-domain Acc** (avg of superdiagonal entries of $\mathcal{R}$), indicating that classic (70%-30%) iid train-test construction overestimates performance of real-world CL systems, which must be deployed on future data. Crucially, this drop can be addressed by our **Streaming Protocol**, which trains on all data of the previous bucket (by repurposing yesterday's testset as today's trainset). Moreover, while most prior CL algorithms make use of supervised learning, we find that unsupervised pre-trained representations (YFCC-B0) can boost performances of all baseline algorithms; especially, linear models are far more effective than ResNet18 trained from scratch, even when using naive **Finetuning** strategy with buffer populated with simple reservoir sampling. Note that for replay-based methods, we keep the same buffer size of one bucket of images (2310 for IID and 3300 for streaming protocol).

5 Approaches

Fully-supervised baselines: We evaluate state-of-the-arts CL algorithms on CLEAR using implementation from an open-sourced CL library Avalanche [37]. In specific, we test replay-based methods such as **ER** [49], **AGEM** [6], and **GDumb** [44], while allowing them to maintain a sufficiently large buffer of one bucket of training images (2310 for iid protocol, 3300 for streaming protocol). We also test an architecture-based method **CWR** [36], a distillation-based method **LwF** [31], and regularization-based methods **SI** [41, 59] and **EWC** [28]. We also include a naive replay-based **Finetuning** strategy, which is to finetune the model solely on the replay buffer (inspired by GDumb [44]), while exploring variants of reservoir sampling strategy to populate the buffer. For all these baseline methods, we use SGD with momentum to train ResNet18 [21] initialized from scratch and report all hyperparameters in Sec. 6 of supplement. Note that some of these approaches such as AGEM can be used for online CL [40] by visiting each sample just once, but in this work we assume the most relaxed offline CL setup so that all samples in the current bucket can be revisited without constraint.

Unsupervised pre-training: Since CLEAR comes with abundant unlabeled samples per time period (700K images per bucket), one may also explore continual semi-supervised or unsupervised learning, as suggested in Fig. 4. As a preliminary experiment, we perform a simple unsupervised pre-training step by self-supervised learning on YFCC100M data collected prior to bucket 1^{st}. Specifically, we train an unsupervised feature model (Moco V2 [9]) with ResNet50 backbone on bucket 0^{th} of 700K images. We release both our MoCo V2 model pre-trained on bucket 0^{th} and self-supervised features (termed as **YFCC-B0**) associated with each image. After pre-training the MoCo model, we follow the popular linear evaluation protocol in self-supervised learning literature [9, 19] to train a linear layer upon the extracted YFCC-B0 features. Surprisingly, we observe significant performance boost across all baseline methods, even with naive **Finetuning** with simple reservoir sampling strategies. Additionally, in Sec. 5 of supplement, we present linear and nonlinear (2 layer MLP) classification results using a variety of other pre-trained feature representations (including ImageNet, CLIP, and others).

Reservoir Sampling: For naive **Finetuning**, we adopt reservoir sampling [32, 57] that uniformly samples from all data encountered in the stream to populate the replay buffer. Specifically, given a buffer of size k, it repeats the following at each timestamp i:

1. If the buffer has not reached its maximal capacity, keep adding new sample.
2. If the buffer is full, when a new sample comes, replace a random sample in the buffer with this new sample with probability $\frac{k}{i}$.

α for Biased Reservoir	IID Protocol					Streaming Protocol	
	In-domain Acc	Next-domain Acc	Acc	BwT	FwT	Next-domain Acc	FwT
α = 0.5	87.4% ± .4%	87.0% ± .4%	88.5% ± .5%	88.7% ± .5%	85.6% ± .4%	87.7% ± .1%	86.4% ± .1%
α = 1.0	88.0% ± .2%	87.3% ± .3%	88.9% ± .3%	89.1% ± .3%	85.9% ± .3%	88.2% ± .2%	86.8% ± .1%
α = 2.0	89.2% ± .2%	88.5% ± .2%	89.7% ± .3%	89.8% ± .3%	86.9% ± .2%	89.3% ± .1%	87.6% ± .1%
α = 5.0	90.2% ± .2%	89.1% ± .2%	90.5% ± .3%	90.6% ± .3%	87.2% ± .2%	89.8% ± .0%	88.0% ± .0%
α = 0.25 * i/k	88.3% ± .4%	87.2% ± .4%	89.1% ± .4%	89.3% ± .4%	85.4% ± .3%	88.3% ± .2%	86.6% ± .1%
α = 0.50 * i/k	89.2% ± .2%	88.4% ± .3%	89.9% ± .3%	90.1% ± .3%	86.4% ± .4%	89.1% ± .1%	87.3% ± .1%
α = 0.75 * i/k	90.1% ± .2%	89.0% ± .2%	90.2% ± .7%	90.2% ± .4%	87.0% ± .2%	89.8% ± .1%	87.8% ± .0%
α = 1.00 * i/k	90.3% ± .2%	89.3% ± .2%	90.6% ± .3%	87.3% ± .3%	90.6% ± .2%	90.1% ± .0%	88.1% ± .0%

Table 2: **Analysis of biased reservoir sampling.** We consider the case where we pretrain MoCo on 0^{th} bucket via unsupervised learning and finetune the last linear layer. Table shows different alpha values for biased reservoir sampling algorithms with replay buffer size of one bucket. Note that the higher the alpha values are, more recent samples will be added to the replay buffer (when $\alpha = 1.0$, it is equivalent to naive reservoir sampling). The key takeaway is that when there is a limited memory buffer, we should bias towards storing more recent samples, e.g., when $\alpha = 5.0$ or $\alpha = 0.75 * i/k$, all metrics enjoy a 1% boost from naive reservoir sampling.

Following this procedure, we can ensure a uniform probability that a visited sample is included in the buffer, i.e., at time i, any sample in stream has probability $\frac{k}{i}$ to be stored in buffer. Note that this sampling procedure is performed once per incoming bucket in our implementation, and we make a simplified assumption that all samples in one bucket own the same timestamp i.

This uniform sampling procedure has shown to work well in prior works [8, 48] as well as some variants that tackle class-imbalanced data streams [2, 26]. However, in CLEAR, we will show that it is beneficial to simply *bias* the probability by an alpha value α larger than 1.0, i.e., $\alpha * \frac{k}{i}$ such that the sampling procedure favors more recent samples.

Biased Reservoir Sampling: In particular, we experiment with two types of alpha:

- **Fixed** α. We can select alpha as a constant value, e.g., $\alpha \in \{0.5, 1.0, 2.0, 5.0\}$. Note that $\alpha = 1.0$ is equivalent to unbiased reservoir sampling. Higher α biases the memory towards storing more recent samples (and vice-versa).
- **Dynamic** α. The alpha value could also change according to the timestamp i. For example, we can have $\alpha \in \{\frac{0.25i}{k}, \frac{0.5i}{k}, \frac{0.75i}{k}, \frac{i}{k}\}$. In this scenario, the probability of replacing an old sample in buffer with a new sample is always a constant, e.g. when $\alpha = \frac{i}{k}$, we always store the new sample with probability $1 = \alpha * \frac{k}{i}$. The latter is equivalent to a first-in first-out (FIFO) priority queue that ensures the k most recent examples remain in memory. Interestingly, many recent works on online CL also adopt FIFO queues [4, 23].

In Sec. 4 of supplement, we provide formal pseudocode of this biased reservoir sampling algorithm.

6 Results and Discussion

In this section, we include a summary of the salient conclusions. We run 5 random seeds for each experiments and report both mean and std over the 5 runs. For the **IID Protocol**, we use 5 random 70-30 train-test splits, reporting both **In-domain Acc** and **Next-domain Acc**, plus three other metrics inspired by prior work [12] including Accuracy (**Acc**), Backward Transfer (**BwT**), and Forward Transfer (**FwT**). For the **Streaming Protocol** (that repurposes previous testsets as trainsets), we report **Next-domain Acc** (Eq.1) and Forward Transfer (**FwT**). Table 1 presents all baseline results.

In-domain Acc inflates performance: We first demonstrate that **In-domain Acc** can falsely inflate the performance of CL algorithms in real world that must be deployed on data from the immediate future. Figure 5 includes an accuracy matrix with `Linear, YFCC-B0, Reservoir Sampling` strategy. Clearly, accuracies on the main diagonal (where train and testsets are drawn from the same iid distribution) are larger than those on the superdiagonal. The accuracy drops on the superdiagonal (train today, test on tomorrow), and continues to drop as we evaluate further into the future (towards the right of the matrix), suggesting CLEAR contains smooth temporal evolution of data. Crucially, Table 1 shows that this drop can be partially addressed by our **Streaming Protocol** that trains on all 100% of prior data (rather than 70%, as dictated by classic iid protocols).

Unsupervised pre-training significantly boosts performance: Without unsupervised pre-training, training ResNet18 on raw RGB images with state-of-the-art fully-supervised CL techniques achieves at best 77.8% **In-domain acc** (under iid protocol) and 78.8% **Next-domain acc** (under streaming

protocol) using LwF [31]. However, **Finetuning** a linear layer on top of unsupervised feature representations (YFCC-B0) pre-trained on bucket 0^{th} of CLEAR improves performance to 88.0% **In-domain acc** (under iid protocol) and 88.2% **Next-domain acc** (under streaming protocol), even with the most basic reservoir sampling strategy. This suggests that CLEAR is still challenging without unsupervised pre-training even in the simplest incremental domain learning setup, and future works should embrace unlabeled data for continual semi-supervised learning to maximize performances.

GDumb falls short compared to other baselines: GDumb [44] as a degenerate solution is far less competitive on CLEAR, most likely because it trains a network from scratch for each bucket while giving up previously trained models. This suggests that CLEAR has smooth temporal variation, in which case continuous representation learning becomes beneficial. To verify this hypothesis, in Sec. 5 of supplement, we show that it is always better to finetune than to train from scratch per timestamp.

Biased reservoir towards more recent samples is beneficial: Biased reservoir sampling that assigns higher sampling probability towards more recent samples combined with naive **Finetuning** improves upon unbiased reservoir and achieves competitive results on CLEAR under both iid and streaming evaluation protocols as in Table 1. We show analysis for different alpha values in Table 2.

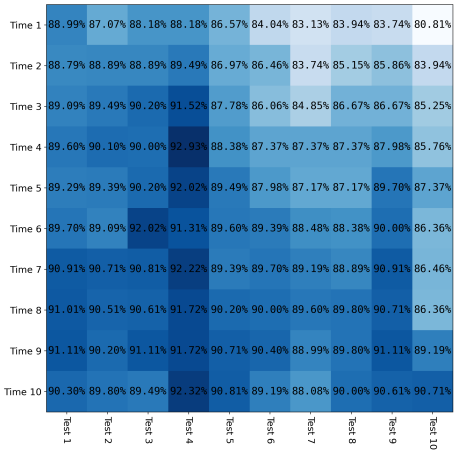

Figure 5: **Accuracy matrix under iid protocol with `Linear, YFCC-B0, Biased Reservoir, Finetuning` strategy**. We show the accuracy matrix under iid protocol which performs training and testing on the same bucket. The x-axis is the test performance on Te_1 to Te_{10} (from left to right), and the y-axis is the training timestamp 1 to 10 (from top to bottom). The main diagonal (**In-domain Acc**) tends to have better performance than the superdiagonal (**Next-domain Acc**) because the former ensures train and test distributions are iid. The further the test bucket is from the current timestamp, the worse the performance, e.g., if we use timestamp 1^{st} model to evaluate on Te_{10}, the accuracy drops from 89% to 81%.

7 Conclusion

We present CLEAR, the first benchmark for naturally-evolving continual image classification. We describe a scalable and semi-automatic visio-linguistic approach for (continual) benchmark construction and present a suite of baseline algorithms and analysis. Salient conclusions are as follows (1) Embrace train-test domain shift! Though a widely-held sentiment, it is surprising to see that traditional evaluation protocols of CL still rely on locally iid assumptions. (2) Unsupervised pre-training with simple sampling strategies surpasses state-of-the-arts full-supervised CL techniques, and thus future CL benchmarks or algorithms should take unlabeled data into accounts. We plan to host CLEAR benchmark on a public platform (link: `https://clear-benchmark.github.io`) and maintain leaderboard of different competing approaches to further ensure reproducibility.

Broader Impacts: Continual Learning has been heavily studied algorithmically but the evaluation has been plagued with datasets with synthetic changes in the distribution over time. We believe CLEAR is the first step filling the gap between CL benchmarks and real-world deployment. Even though, our dataset is a subset of already existing YFCC100M, we still performed due diligence to ensure that the labeled portion of the dataset is free of inappropriate images. We hope that the real world nature of our benchmark will allow the community to identify biases arising from continual distributions, an under-explored but relevant problem for real-world ML deployments.

References

[1] R. Aljundi, K. Kelchtermans, and T. Tuytelaars. Task-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11254–11263, 2019. 3, 4

[2] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio. Gradient based sample selection for online continual learning. *NeurIPS*, 2019. 9

[3] K. Boakye, S. Farfade, H. Izadinia, Y. Kalantidis, and P. Garrigues. Tag prediction at flickr: A view from the darkroom. In *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*, pages 376–384, 2017. 4

[4] Z. Cai, O. Sener, and V. Koltun. Online continual learning with natural distribution shifts: An empirical study with visual data. In *ICCV*. 3, 4, 7, 9

[5] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *ECCV*. 2

[6] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny. Efficient lifelong learning with a-gem. *NeurIPS*, 2018. 8

[7] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. Torr, and M. Ranzato. Continual learning with tiny episodic memories. 2019. 3

[8] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. Torr, and M. Ranzato. On tiny episodic memories in continual learning. *NeurIPS*, 2019. 9

[9] X. Chen, H. Fan, R. Girshick, and K. He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 5, 8

[10] M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *PAMI*, 2021. 3

[11] L. Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. 4

[12] N. Díaz-Rodríguez, V. Lomonaco, D. Filliat, and D. Maltoni. Don't forget, there is more than forgetting: new metrics for continual learning. *arXiv preprint arXiv:1810.13166*, 2018. 1, 2, 7, 9

[13] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2020. 1

[14] L. eon Bottou. Online learning and stochastic approximations. 3, 7

[15] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 2

[16] S. Farquhar and Y. Gal. Towards robust evaluations of continual learning. 2018. 2

[17] R. M. French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999. 1

[18] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013. 1, 2, 3

[19] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. 8

[20] J. He and F. Zhu. Unsupervised continual learning via pseudo labels. *arXiv preprint arXiv:2104.07164*, 2021. 2

[21] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*. 1, 8

[22] Y.-C. Hsu, Y.-C. Liu, A. Ramasamy, and Z. Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. In *NeurIPS Continual learning Workshop*, 2018. URL https://arxiv.org/abs/1810.12488. 3, 4, 6

[23] H. Hu, O. Sener, F. Sha, and V. Koltun. Drinking from a firehose: Continual learning with web-scale natural language. *arXiv preprint arXiv:2007.09335*, 2020. 3, 4, 9

[24] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*. 1

[25] A. Joulin, L. Van Der Maaten, A. Jabri, and N. Vasilache. Learning visual features from large weakly supervised data. In *ECCV*. 4

[26] C. D. Kim, J. Jeong, and G. Kim. Imbalanced continual learning with partitioning reservoir sampling. In *ECCV*, . 9

[27] G. Kim, E. P. Xing, and A. Torralba. Modeling and analysis of dynamic behaviors of web image collections. In *ECCV*, . 1

[28] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 1, 3, 8

[29] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009. 2, 3, 4

[30] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang. Overcoming catastrophic forgetting by incremental moment matching. *NeurIPS*, 2017. 1, 3

[31] A. Li, A. Jabri, A. Joulin, and L. van der Maaten. Learning visual n-grams from web data. In *ICCV*. 3, 4, 8, 10

[32] K.-H. Li. Reservoir-sampling algorithms of time complexity o (n (1+ log (n/n))). *ACM Transactions on Mathematical Software (TOMS)*, 20(4):481–493, 1994. 8

[33] S. Li, Y. Du, G. M. van de Ven, A. Torralba, and I. Mordatch. Energy-based models for continual learning. *arXiv preprint arXiv:2011.12216*, 2020. 3

[34] Z. Li and D. Hoiem. Learning without forgetting. *PAMI*. 1, 2

[35] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*. 1, 4

[36] V. Lomonaco and D. Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *Conference on Robot Learning*, pages 17–26. PMLR, 2017. 2, 3, 8

[37] V. Lomonaco, L. Pellegrini, A. Cossu, and et al. Avalanche: an end-to-end library for continual learning. In *CVPR Workshop*. 8

[38] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. *NeurIPS*, 2017. 1, 2, 4, 7

[39] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. Van Der Maaten. Exploring the limits of weakly supervised pretraining. In *ECCV*. 4

[40] Z. Mai, R. Li, J. Jeong, D. Quispe, H. Kim, and S. Sanner. Online continual learning in image classification: An empirical survey. *arXiv preprint arXiv:2101.10423*, 2021. 3, 4, 8

[41] D. Maltoni and V. Lomonaco. Continuous learning in single-incremental-task scenarios. *Neural Networks*, 116:56–73, 2019. 1, 3, 8

[42] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989. 1

[43] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. 3

[44] A. Prabhu, P. H. Torr, and P. K. Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *ECCV*. 1, 2, 3, 4, 8, 10

[45] M. Pratama, A. Ashfahani, and E. Lughofer. Unsupervised continual learning via self-adaptive deep clustering approach. *1st CSSL Workshop @ IJCAI 2021*, 2021. 2

[46] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. *International Conference on Machine Learning*, 2021. 3, 4, 5

[47] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *CVPR*. 2

[48] M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *ICLR*, 2019. 9

[49] D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap, and G. Wayne. Experience replay for continual learning. *NeurIPS*, 2018. 8

[50] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*. 1, 2, 4

[51] N. Sambasivan, S. Kapania, H. Highfill, D. Akrong, P. Paritosh, and L. M. Aroyo. "everyone wants to do the model work, not the data work": Data cascades in high-stakes ai. In *CHI*. 3

[52] J. Smith, J. Balloch, Y.-C. Hsu, and Z. Kira. Memory-efficient semi-supervised continual learning: The world is its own replay buffer. *arXiv preprint arXiv:2101.09536*, 2021. 2

[53] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016. 2, 3, 4, 5

[54] G. M. Van de Ven and A. S. Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019. 3, 4

[55] V. N. Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999, 1999. 1, 7

[56] T. Veniat, L. Denoyer, and M. Ranzato. Efficient continual learning with modular networks and task-driven priors. In *ICLR*, 2021. 3

[57] J. S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985. 8

[58] L. Wang, K. Yang, C. Li, L. Hong, Z. Li, and J. Zhu. Ordisco: Effective and efficient usage of incremental unlabeled data for semi-supervised continual learning. In *CVPR*. 2

[59] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR, 2017. 1, 2, 3, 8

[60] C. Zeno, I. Golan, E. Hoffer, and D. Soudry. Task agnostic continual learning using online variational bayes. *arXiv preprint arXiv:1803.10123*, 2018. 3, 4