⏲ TiC-CLIP: Continual Training of CLIP Models

Saurabh Garg[‡*] **Mehrdad Farajtabar**[†] **Hadi Pouransari**[†] **Raviteja Vemulapalli**[†]

Sachin Mehta[†] **Oncel Tuzel**[†] **Vaishaal Shankar**[†] **Fartash Faghri**[†]

[†]Apple [‡]Carnegie Mellon University
sgarg2@andrew.cmu.edu, fartash@apple.com

Abstract

Keeping large foundation models up to date on latest data is inherently expensive. To avoid the prohibitive costs of constantly retraining, it is imperative to *continually* train these models. This problem is exacerbated by the lack of any large scale continual learning benchmarks or baselines. We introduce the first set of web-scale Time-Continual (TiC) benchmarks for training vision-language models: TIC-DataComp, TIC-YFCC, and TIC-RedCaps. TIC-DataComp, our largest dataset, contains over 12.7B timestamped image-text pairs spanning 9 years (2014–2022). We first use our benchmarks to curate various *dynamic* evaluations to measure temporal robustness of existing models. We show OpenAI's CLIP (trained on data up to 2020) loses $\approx 8\%$ zero-shot accuracy on our curated retrieval task from 2021–2022 compared with more recently trained models in OpenCLIP repository. We then study how to efficiently train models on time-continuous data. We demonstrate that a simple rehearsal-based approach that continues training from the last checkpoint and replays old data reduces compute by $2.5\times$ when compared to the standard practice of retraining from scratch. A longer version of the paper is available at https://arxiv.org/abs/2310.16226.

1 Introduction

Large multimodal foundation models [9] have offered unprecedented advancements in image-generation and zero-shot generalization, and have led to a paradigm shift in multimodal learning, e.g., CLIP [76], Flamingo [2], and Stable Diffusion [83]. These foundation models are typically trained on large web-scale datasets which are fixed and *static* in nature. For example, CLIP's training data contains 400 million image-text pairs, and Stable Diffusion was trained on LAION-2B dataset [85]. In reality, however, these models must operate in a *dynamic* environment, where the world is in a state of constant change. For instance, the internet continually evolves, with petabytes of new data being added daily [104, 105]. It remains unclear how legacy models, e.g., OpenAI's CLIP models which were trained on internet-scale data up until 2020, work on future data and whether they even require any re-training to adapt to time-evolving data.

We begin by comparing robustness of OpenAI's CLIP models to others in OpenCLIP repository that are trained on more recently curated web-datasets (e.g., LAION-5B, DataComp) containing data up until 2022 [44]. Since there is no existing benchmark to understand robustness to time-evolving vision-language data, we curate *dynamic* classification and retrieval tasks for years 2014–2022 and evaluate different CLIP models (see Sec. A.2 for our evaluation tasks). We make an intriguing observation that OpenAI models exhibit a significant gap in retrieval performance on data from 2021–2022 compared with 2014–2016 whereas OpenCLIP models retain their performance. In contrast, standard evaluations such as accuracy on ImageNet distribution shifts paint an incomplete picture that OpenAI's CLIP models are slightly more robust than OpenCLIP models (Fig. 1). Our findings not

*Work done during an internship at Apple.

Workshop on Distribution Shifts, 37th Conference on Neural Information Processing Systems (NeurIPS 2023).

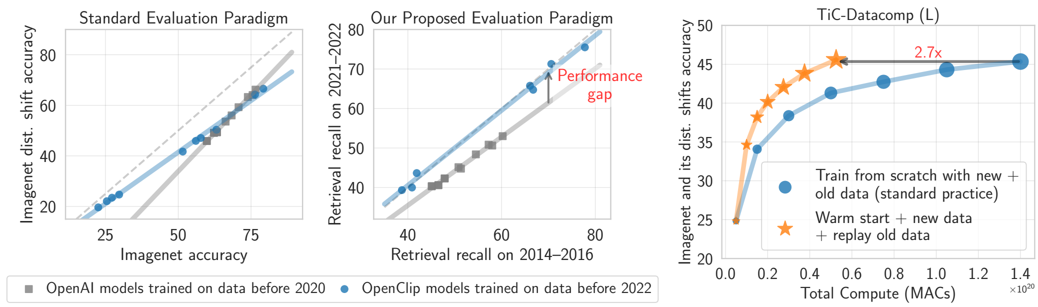


Figure 1: *(Left, Middle)* **OpenAI models show less zero-shot robustness on retrieval task from 2021–2022.** OpenCLIP models and OpenAI models have similar robustness on standard benchmarks. However, OpenAI models show less robustness on our retrieval task when compared with recent models in OpenCLIP repository, highlighting susceptibility to a time-evolving data distribution *(Right)* **Simple continual training baseline is computationally efficient and competitive to retraining from scratch.** Different points denote models trained sequentially on our TiC-DataComp (L) as data arrives over time. Warm start training with previous checkpoint and replaying all old data, performs similar to Oracle which trains from scratch every time new data arrives, by using 2.7× less compute.

only demonstrate the critical need for models to adapt and evolve alongside dynamic data distributions, but also underscores the limitations of relying solely on static benchmarks (e.g. ImageNet).

One naive but common practice for adapting to time-evolving data is to train a new CLIP model from *scratch* every time we obtain a new pool of image-text data. This practice has its rationale: initiating training from a pre-existing model can make it difficult to change the model's behavior in light of new data [3, 1, 61]. However, training foundation models from scratch demands significant computational resources and is often infeasible to repeat frequently. For example, ViT-g-14 in Schuhmann et al. [85], Cherti et al. [17] was trained for 240K A100 GPU hours which is approximately one month on 400 GPUs. The prevailing training guidelines centered around scaling laws for CLIP training have only looked at training from scratch [18]. This leads to a pivotal question: *How can we continuously update models as the data distribution evolves over time given computational constraints?*

There exists a vast literature on continual learning, with a focus on adapting models to dynamic environments [72, 37, 23]. Traditionally, this field concentrated on synthetic incremental benchmarks that lack natural evolution between tasks, and hence, continual learning methods are seldom used in real-world scenarios [22, 59]. In contrast, recent works focusing on continual learning methods for CLIP models, primarily target improving performance on a single or a sequence of disjoint downstream tasks [27, 112, 111, 43]. While some recent works have started to address these problems, existing benchmarks are comparatively much smaller in scale, or lack paired image-text data [70, 59]. Simply put, there is a scarcity of work focusing on continual training of CLIP models on naturally evolving data with time at web-scale.

We take the first step towards **Time-Continual (TiC)** training of CLIP models where data distribution evolves naturally over time (overview in Fig. 2). We introduce TiC-DataComp, a new benchmark for Time-Continual training of CLIP models, which we create by appending "crawl time" information to existing CommonPool dataset [34]. We also repurpose other web-scale datasets gathered from diverse sources, such as Reddit and Flickr. Specifically, we curate TiC-YFCC and TiC-RedCaps by leveraging time information available in YFCC [94] and Redcaps [25] respectively. The primary objective of our study on this benchmark is to develop continual learning methods that operate within a constrained computational budget (say C) each time a fresh batch of data becomes available. These methods compete with an Oracle, which starts training from scratch every time new data arrives, utilizing a cumulative computational budget.

To assess models trained in our TiC-CLIP framework, we evaluate models on our proposed dynamic evaluation tasks that evolve with time along with 28 standard classification and retrieval tasks including ImageNet [55], ImageNet distributions shifts, and Flickr [74], in a zero-shot manner following the work of Gadre et al. [34], Radford et al. [76].

Finally, we develop continual learning methods on our benchmarks and perform over two hundred experiments with different baselines that utilize previous checkpoints (e.g., warm start, patching,

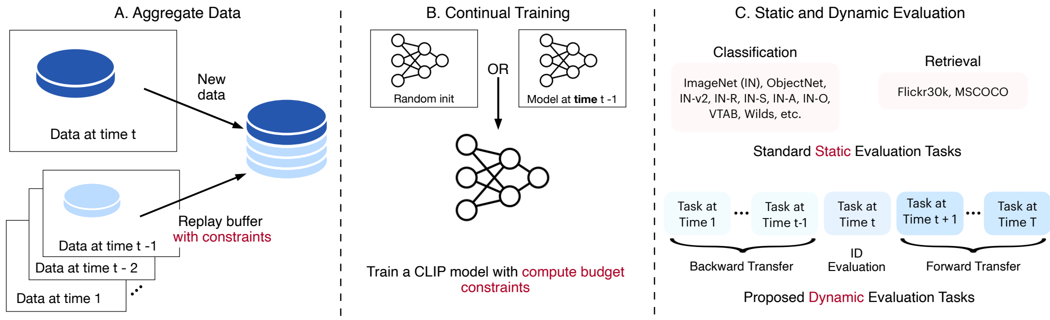


Figure 2: **Experimental protocol on our benchmarks.** *(A)* Combine new and old data given buffer constraints. *(B)* Continually train a model with a compute budget (say C) either by starting with the previous checkpoint or from scratch. *(C)* Evaluate on standard tasks and our proposed dynamic tasks.

and distillation), replay buffers, and learning rate schedules. Our findings highlight a key takeaway: Cumulative method that warm starts training with the latest checkpoint and replays all old data, achieves performance competitive to an Oracle while being 2.7$\times$ computationally more efficient. Additionally, our experiments demonstrate interesting trade-offs between buffer sizes for static and dynamic performance and provide valuable insights into learning rate schedules for sequential training. Our results span over various dataset scales (from 11M samples to 3B) and highlight trends with different methods that are largely consistent across scales.

To make our benchmarks accessible, we are commited to publicly releasing the time information we collect on top of existing datasets. Our work is just an initial step towards continual training of foundation models, and we believe our research would spur more attention to this understudied area.

2 TiC-CLIP: Benchmarks, Experimental Protocol and Methods

We train on image-text data that arrives sequentially unlike the conventional image-text datasets which are static (e.g. WiT in CLIP, DataComp in Gadre et al. [34]). The goal of a learner is to train a *deployable* model at each step as new data becomes available with a fixed compute budget.

Benchmark Design: How we Create Time-Continual Datasets? To instantiate continual training of CLIP, we extend existing image-text datasets with time information collected from the original source of the datasets. Our largest dataset is TIC-DataComp which contains 12.7 billion image-text pairs with "crawl-time" metadata created on top of the existing DataComp benchmark [34]. The source of DataComp is Common Crawl, which periodically releases web-crawled data snapshots, typically on a monthly basis since 2014 with new and updated webpages. To construct TIC-DataComp, we augment each image-text pair in DataComp with the timestamp of the *first* snapshot containing that pair. We also create TIC-YFCC and TIC-RedCaps on top of existing YFCC15M [94, 76] and Redcaps [25] datasets to highlight the broad applicability of our findings to diverse datasets. While time-related metadata is absent in the DataComp, it is available in the original releases of YFCC and Redcaps. Nevertheless, to the best of our knowledge, no prior work utilizes such time information for continual training of CLIP models. Although our benchmark contains time information at the granularity of months, we limit our experiments to the granularity of years. See App. A.1 for details.

Evaluation Testbed We leverage the temporal information in our benchmarks to create *dynamic evaluation* tasks. For TIC-DataComp, we create dynamic tasks for both retrieval and classification (examples in Figure 3). We sample a batch of IID image-text pairs from different timestamps to create TIC-DataComp-Retrieval and evaluate text retrieval performance given the corresponding image and vice-versa. We also create a classification dataset TIC-DataComp-Net with ImageNet classes from CommonPool and augmented with timestamps. Our construction is inspired by LAIONNet [88] with one difference. Unlike LAIONNet, we do not filter the image-text pairs with CLIP similarity scores to avoid biasing the selection process. We describe the construction process in detail in App. A.2. Evaluations are done in a zero-shot manner. We also evaluate models on 28 standard *static* classification and retrieval tasks as in Gadre et al. [34]. We list all the datasets in App. G.2.

Evaluation metrics For static datasets (e.g., ImageNet), we report performance of T-th model. When dealing with dynamic evaluation datasets, we assess the performance of models trained at all time steps

Table 2: **Zero shot performance on our time-continual benchmarks.** * and ** denote methods that violate the compute budget. For static tasks, we tabulate accuracy of the models obtained on the final timestamp. For dynamic tasks, we tabulate forward/backward transfer and ID performance on retrieval tasks (Sec. A.3). Results with all datasets are in Table 4 (see App. C). For TiC-DataComp (XL), we include results with Bestpool filtering (basic filtering in Table 7). For all metrics, higher is better.

Benchmark	**Method**	**Compute** (MACs)	Static Tasks				Dynamic Retrieval Tasks		
			ImageNet	ImageNet dist. shift	Flickr30k	Average over 28 datasets	Backward Transfer	ID Performance	Forward Transfer
TiC-YFCC	Restart	3.4×10^{18}	5.2	3.6	3.0	4.0	13.2	41.4	18.6
	Sequential	3.4×10^{18}	17.3	10.5	15.9	11.6	42.2	48.4	23.7
	Patching	3.4×10^{18}	18.9	11.3	18.5	12.1	44.7	53.4	24.5
	Cumulative-Exp	3.4×10^{18}	24.1	14.3	20.4	15.4	60.4	60.1	27.1
	Cumulative-Equal	3.4×10^{18}	23.9	13.8	20.5	14.7	60.4	60.4	27.1
	Cumulative-All	3.4×10^{18}	**29.3**	**17.6**	**26.8**	**18.0**	**66.4**	**60.2**	**27.6**
	LwF*	4.1×10^{18}	16.9	9.8	14.7	10.5	36.6	56.0	23.2
	Cumulative-All*	3.6×10^{18}	**29.2**	**17.5**	**27.4**	**18.1**	**66.8**	**60.3**	**27.6**
	Oracle**	8.5×10^{18}	**29.2**	**17.0**	**25.9**	**18.0**	**66.1**	**61.8**	**26.9**
TiC-RedCaps	Restart	3.4×10^{18}	11.7	8.5	3.7	7.6	21.3	25.4	22.4
	Sequential	3.4×10^{18}	19.3	13.7	6.2	11.9	33.0	33.6	27.5
	Patching	3.4×10^{18}	21.3	15.2	7.7	14.0	34.8	34.8	27.8
	Cumulative-Exp	3.4×10^{18}	27.3	19.1	10.5	16.3	44.5	42.0	32.6
	Cumulative-Equal	3.4×10^{18}	27.8	19.4	10.0	16.7	44.4	42.0	32.6
	Cumulative-All	3.4×10^{18}	**32.2**	18.7	14.5	**19.7**	**48.9**	**43.2**	**33.4**
	LwF*	4.1×10^{18}	21.6	14.8	8.2	13.5	35.4	36.0	28.4
	Cumulative-All*	3.6×10^{18}	**32.9**	**23.7**	**14.1**	**20.1**	**49.0**	**43.4**	**33.4**
	Oracle**	8.5×10^{18}	**32.7**	**22.7**	**14.3**	**20.6**	**48.5**	**43.1**	**33.4**
TiC-DataComp (L)	Sequential	2.7×10^{19}	44.7	37.4	48.4	32.7	52.6	**58.4**	41.1
	Patching	2.7×10^{19}	45.8	38.9	49.7	33.6	55.2	57.5	40.9
	Cumulative-Exp	2.7×10^{19}	47.3	39.6	50.8	35.0	60.4	**58.4**	**41.4**
	Cumulative-Equal	2.7×10^{19}	47.7	40.3	51.8	36.0	60.9	**58.2**	**41.4**
	Cumulative-All	2.7×10^{19}	48.9	41.3	50.9	36.3	62.1	57.3	41.2
	Cumulative-All*	4.1×10^{19}	53.0	**44.3**	**54.4**	**39.0**	63.0	57.8	41.2
	Oracle**	1.1×10^{20}	**53.6**	44.0	53.9	38.0	**64.3**	**58.6**	**41.8**
TiC-DataComp (XL)	Sequential	2.7×10^{20}	66.5	54.2	61.2	51.7	63.1	68.9	56.8
	Cumulative-All	2.7×10^{20}	71.6	58.8	65.1	55.7	**70.7**	**68.5**	**57.1**
	Cumulative-All*	3.5×10^{20}	**72.8**	60.4	66.5	**57.7**	**71.0**	**68.6**	**57.1**
	Oracle**	1.1×10^{21}	**73.3**	**61.3**	**68.0**	**58.1**	-	-	-

and report three aggregate metrics: In-domain performance, backward transfer and forward transfer (see App. A.2). While the static tasks capture performance on standard benchmarks, dynamic tasks capture problems due to distribution shift (for forward transfer) and forgetting (for backward transfer).

Experimental Protocol For Training We follow a streaming protocol, where data is progressively revealed to the learner in large batches with the objective of achieving a deployable model as early as possible after each batch arrives. We allow methods to use the last model checkpoint at each step as the cost of keeping one checkpoint per month is often negligible. In contrast, the cost of retaining old data can be high and might not be permitted due to data expiration policies. Thus, along with studying methods that retain all old data, we also explore strategies that restrict data persistence . To ensure a fair comparison between methods, we establish a consistent total compute budget, and allocate it evenly for training at every time step. Unless specified otherwise, for all methods except Oracle and LwF, we use the same compute budget.

How to Continually Train Models? We lay out different methods specifically focus on the following questions (Tab. 1): (i) How to utilize/replay data from previous time steps; (ii) How to leverage previously trained model checkpoints? (iii) What should be the training/optimization procedure? In our comparisons, Oracle methods Oracle represents a *prohibitively expensive* method that is the most common practice in training large-scale foundation models. The goal of other methods is to perform as close as possible to the Oracle within their limited budget. See App. B for details on these methods and discussion around learning rate schedules.

Table 1: Table summarizing our methods. D: data size in each step, T total time steps, t: current time step, C: compute budget (iterations).

Method	**Each Step**			**Total**
	Train Size	**Init.**	**Compute**	**Compute**
Cumulative-All	tD	Last	C	TC
Cumulative-Exp	$2D$	Last	C	TC
Cumulative-Equal	$2D$	Last	C	TC
Sequential	D	Last	C	TC
Restart	tD	Rand	C	TC
Patching	D	Last Patch	C	TC
LwF	D	Last	$1.2 \times C$	$1.2 \times TC$
Oracle**	tD	Rand	tC	$\frac{(T+1)T}{2}C$

3 Main Results

Here, we only summarize key takeaways due to space constraints (see App. C for detailed results).

Cumulative-All saves up to $4\times$ the cost. On dynamic evaluation tasks, we observe that Cumulative-All where we replay all the past data, achieves performance close to the Oracle (within 1%) using significantly less compute ($4\times$ less on TIC-DataComp and $2.5\times$ less on TIC-YFCC and TIC-RedCaps). On static tasks, the gap remains small at small scales but grows to 4.7% on `large`, 1.8% on `xlarge` Bestpool, and 4% on `xlarge` Basic (see Table 4 and Table 7). In these cases, training Cumulative models with slightly extra compute bridges the gap while remaining at least $2.7\times$ more computationally efficient (see rows with * in Table 4). This highlights that with unconstrained access to past data, we can simply train sequentially and save significant computational resources.

At scale, Sequential has strong forward transfer but lacks on static tasks. On TIC-YFCC and TIC-RedCaps, which are at the smallest scale, we observe a significant gap ($> 10\%$) between Sequential (with no data replay) and Oracle on all tasks. On the other hand, on all scales in TIC-DataComp, Sequential shows strong performance on forward transfer and ID dynamic evaluations. However, on static tasks and backward transfer evaluations, Sequential significantly underperforms the Oracle.

Patching and LwF improve over Sequential but lag behind Cumulative-All. On static tasks, LwF improves over Sequential by 2%, while on dynamic tasks, LwF improves backward transfer by 7% on TIC-DataComp (M). However, its computation cost is higher than even Cumulative-All* which outperforms LwF on all tasks. Patching improves over Sequential on backward transfer on all datasets (e.g., 5% boost on TIC-DataComp L) highlighting that Patching combines benefits of previously patched model and the new Sequential model without additional computation cost. However, such benefits do not show up on static tasks.

-Exp and -Equal significantly reduce replay buffer size and maintain static task performance and backward transfer. Recall, that -Exp and -Equal reduce the replay buffer size to a maximum $2D$ of old data. In particular, at the last time step, -Exp and -Equal reduce the buffer size by $3.5\times$ for TIC-DataComp datasets. While reducing the buffer sizes, these methods still achieve performance close to Cumulative-All (within 2%) on both static and dynamic tasks, with -Equal consistently better than -Exp strategy. As we go to large scale, e.g., from `medium` to `large`, the gap between these methods and Cumulative-All reduces. These findings demonstrate that even a small amount of replay data from old time steps stays competitive with replaying all data and significantly improves over no replay at all.

Warm up helps training on data from first time step, but hurts on subsequent time steps. We investigate the effectiveness of warmup in first versus subsequent time steps. Surprisingly, we observe that not using warmup for subsequent training runs is *strictly* more beneficial than using warmup on both static and dynamic tasks. In particular, on TIC-DataComp (L), we observe about 2.5% increase in accuracy on ImageNet when not using warmup with Cumulative (see App. F.3). Moreover, we also ablate over not using warm up for the first training run and observe a drop of approximately 4.7% accuracy in the first time step on TIC-DataComp (L). Hence, we default to using warmup when training on the first time step and not using it on the subsequent time steps with all methods.

Same maximum LR works best across all runs when using cosine schedule. We ablate on TIC-DataComp (M) to investigate how to change LR after training on data from the first time step. Unlike conventional pretraining and finetuning settings where LR is typically decreased for subsequent training, we observe that decaying maximum LR for subsequent steps in our setup hurts on static and dynamic tasks and consequently, we use same maximum LR across our runs (see App. F.3).

Filtering strategy changes the ordering of performance on static and dynamic retrieval tasks. We observe that while bestpool filtering models outperform basic filterining models on TIC-DataComp (XL) by 6% on static tasks, they underperform by over 5% on dynamic retrieval task (see Fig. 9).

4 Conclusion and Future Work

In conclusion, we view TIC-DataComp as the initial stride toward the continual training of large-scale vision-language foundation models. We aspire to empower the research on large-scale continual-learning through our new benchmark and preliminary results obtained using simple baselines.

There are several pivotal directions for future work: (i) Reduce the replay buffer size while maintaining the performance on static evaluation tasks and backward-transfer; (ii) Compare our baselines on continually streaming data at finer granularity, e.g., streaming data at the monthly level; (iii) Investigate alternate learning rate schedules (e.g., Const-Cosine) that are forward looking, and are better suited to continual learning; (iv) Better data filtering techniques that are more inclusive of future data; (v) Expand our problem setup to encompass the training of other large-scale foundation models.

References

[1] Achille, A., Rovere, M., and Soatto, S. (2018). Critical learning periods in deep networks. In *International Conference on Learning Representations*.

[2] Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al. (2022). Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736.

[3] Ash, J. and Adams, R. P. (2020). On warm-starting neural network training. *Advances in neural information processing systems*, 33:3884–3894.

[4] Balaji, Y., Farajtabar, M., Yin, D., Mott, A., and Li, A. (2020). The effectiveness of memory replay in large scale continual learning. *arXiv preprint arXiv:2010.02418*.

[5] Bandi, P., Geessink, O., Manson, Q., Van Dijk, M., Balkenhol, M., Hermsen, M., Bejnordi, B. E., Lee, B., Paeng, K., Zhong, A., et al. (2018). From detection of individual metastases to classification of lymph node status at the patient level: the camelyon17 challenge. *IEEE Transactions on Medical Imaging*. `https://pubmed.ncbi.nlm.nih.gov/30716025/`.

[6] Barbu, A., Mayo, D., Alverio, J., Luo, W., Wang, C., Gutfreund, D., Tenenbaum, J., and Katz, B. (2019). Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32. Curran Associates, Inc. `https://proceedings.neurips.cc/paper/2019/file/97af07a14cacba681feacf3012730892-Paper.pdf`.

[7] Beery, S., Cole, E., and Gjoka, A. (2020). The iwildcam 2020 competition dataset. `https://arxiv.org/abs/2004.10340`.

[8] Bitton, Y., Guetta, N. B., Yosef, R., Elovici, Y., Bansal, M., Stanovsky, G., and Schwartz, R. (2022). WinoGAViL: Gamified association benchmark to challenge vision-and-language models. `https://arxiv.org/abs/2207.12576`.

[9] Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. (2021). On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

[10] Bornschein, J., Galashov, A., Hemsley, R., Rannen-Triki, A., Chen, Y., Chaudhry, A., He, X. O., Douillard, A., Caccia, M., Feng, Q., et al. (2022). Nevis'22: A stream of 100 tasks sampled from 30 years of computer vision research. *arXiv preprint arXiv:2211.11747*.

[11] Bossard, L., Guillaumin, M., and Van Gool, L. (2014). Food-101–mining discriminative components with random forests. In *European Conference on Computer Vision (ECCV)*. `https://link.springer.com/chapter/10.1007/978-3-319-10599-4_29`.

[12] Cai, Z., Sener, O., and Koltun, V. (2021). Online continual learning with natural distribution shifts: An empirical study with visual data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8281–8290.

[13] Cermelli, F., Mancini, M., Bulo, S. R., Ricci, E., and Caputo, B. (2020). Modeling the background for incremental learning in semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9233–9242.

[14] Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. (2018). Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*.

[15] Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H., and Ranzato, M. (2019). On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*.

[16] Cheng, G., Han, J., and Lu, X. (2017). Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the Institute of Electrical and Electronics Engineers (IEEE)*. `https://ieeexplore.ieee.org/abstract/document/7891544`.

[17] Cherti, M., Beaumont, R., Wightman, R., Wortsman, M., Ilharco, G., Gordon, C., Schuhmann, C., Schmidt, L., and Jitsev, J. (2022). Reproducible scaling laws for contrastive language-image learning. `https://arxiv.org/abs/2212.07143`.

[18] Cherti, M., Beaumont, R., Wightman, R., Wortsman, M., Ilharco, G., Gordon, C., Schuhmann, C., Schmidt, L., and Jitsev, J. (2023). Reproducible scaling laws for contrastive language-image learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2818–2829.

[19] Christie, G., Fendley, N., Wilson, J., and Mukherjee, R. (2018). Functional map of the world. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. `https://arxiv.org/abs/1711.07846`.

[20] Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., and Vedaldi, A. (2014). Describing textures in the wild. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. `https://openaccess.thecvf.com/content_cvpr_2014/html/Cimpoi_Describing_Textures_in_2014_CVPR_paper.html`.

[21] Coates, A., Ng, A., and Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. `https://proceedings.mlr.press/v15/coates11a.html`.

[22] Cossu, A., Graffieti, G., Pellegrini, L., Maltoni, D., Bacciu, D., Carta, A., and Lomonaco, V. (2022). Is class-incremental enough for continual learning? *Frontiers in Artificial Intelligence*, 5:829842.

[23] De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., and Tuytelaars, T. (2021). A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385.

[24] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. `https://ieeexplore.ieee.org/abstract/document/5206848`.

[25] Desai, K., Kaul, G., Aysola, Z., and Johnson, J. (2021). Redcaps: Web-curated image-text data created by the people, for the people. *arXiv preprint arXiv:2111.11431*.

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

[27] Ding, Y., Liu, L., Tian, C., Yang, J., and Ding, H. (2022). Don't stop learning: Towards continual learning for the clip model. *arXiv preprint arXiv:2207.09248*.

[28] Dohare, S., Hernandez-Garcia, J., Rahman, P., Sutton, R., and Mahmood, A. R. (2023). Loss of plasticity in deep continual learning.

[29] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

[30] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*. `https://openreview.net/forum?id=YicbFdNTTy`.

[31] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2007). The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. `http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html`.

[32] Farajtabar, M., Azizan, N., Mott, A., and Li, A. (2020). Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3762–3773. PMLR.

[33] Fei-Fei, L., Fergus, R., and Perona, P. (2004). Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories. *Conference on Computer Vision and Pattern Recognition (CVPR) Workshop*. `https://ieeexplore.ieee.org/document/1384978`.

[34] Gadre, S. Y., Ilharco, G., Fang, A., Hayase, J., Smyrnis, G., Nguyen, T., Marten, R., Wortsman, M., Ghosh, D., Zhang, J., et al. (2023). Datacomp: In search of the next generation of multimodal datasets. *arXiv preprint arXiv:2304.14108*.

[35] Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. `https://ieeexplore.ieee.org/abstract/document/6248074`.

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

[37] Hadsell, R., Rao, D., Rusu, A. A., and Pascanu, R. (2020). Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences*, 24(12):1028–1040.

[38] Hayes, T. L., Cahill, N. D., and Kanan, C. (2019). Memory efficient experience replay for streaming learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9769–9776. IEEE.

[39] Helber, P., Bischke, B., Dengel, A., and Borth, D. (2019). Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *Journal of Selected Topics in Applied Earth Observations and Remote Sensing*. https://arxiv.org/abs/1709.00029.

[40] Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., Song, D., Steinhardt, J., and Gilmer, J. (2021a). The many faces of robustness: A critical analysis of out-of-distribution generalization. *ICCV*. https://arxiv.org/abs/2006.16241.

[41] Hendrycks, D., Zhao, K., Basart, S., Steinhardt, J., and Song, D. (2021b). Natural adversarial examples. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. https://arxiv.org/abs/1907.07174.

[42] Hsu, Y.-C., Liu, Y.-C., Ramasamy, A., and Kira, Z. (2018). Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *arXiv preprint arXiv:1810.12488*.

[43] Ilharco, G., Wortsman, M., Gadre, S. Y., Song, S., Hajishirzi, H., Kornblith, S., Farhadi, A., and Schmidt, L. (2022). Patching open-vocabulary models by interpolating weights. *Advances in Neural Information Processing Systems*, 35:29262–29277.

[44] Ilharco, G., Wortsman, M., Wightman, R., Gordon, C., Carlini, N., Taori, R., Dave, A., Shankar, V., Namkoong, H., Miller, J., Hajishirzi, H., Farhadi, A., and Schmidt, L. (2021). Openclip. If you use this software, please cite it as below.

[45] Jang, J., Ye, S., Lee, C., Yang, S., Shin, J., Han, J., Kim, G., and Seo, M. (2022). Temporalwiki: A lifelong benchmark for training and evaluating ever-evolving language models. *arXiv preprint arXiv:2204.14211*.

[46] Jang, J., Ye, S., Yang, S., Shin, J., Han, J., Kim, G., Choi, S. J., and Seo, M. (2021). Towards continual knowledge learning of language models. *arXiv preprint arXiv:2110.03215*.

[47] Jia, C., Yang, Y., Xia, Y., Chen, Y.-T., Parekh, Z., Pham, H., Le, Q., Sung, Y.-H., Li, Z., and Duerig, T. (2021). Scaling up visual and vision-language representation learning with noisy text supervision. In *International conference on machine learning*, pages 4904–4916. PMLR.

[48] Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L., and Girshick, R. B. (2017). CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *Conference on Computer Vision and Pattern Recognition (CVPR)*. https://arxiv.org/abs/1612.06890.

[49] Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2017). Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

[50] Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

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

[52] Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., Gao, I., Lee, T., David, E., Stavness, I., Guo, W., Earnshaw, B. A., Haque, I. S., Beery, S., Leskovec, J., Kundaje, A., Pierson, E., Levine, S., Finn, C., and Liang, P. (2021). WILDS: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning (ICML)*. https://arxiv.org/abs/2012.07421.

[53] Krause, J., Stark, M., Deng, J., and Fei-Fei, L. (2013). 3d object representations for fine-grained categorization. In *International Conference on Computer Vision Workshops (ICML)*. https://www.cv-foundation.org/openaccess/content_iccv_workshops_2013/W19/html/Krause_3D_Object_Representations_2013_ICCV_paper.html.

[54] Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[55] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

[56] LeCun, Y. (1998). The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/.

[57] Li, J., Li, D., Savarese, S., and Hoi, S. (2023). Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*.

[58] Li, Z. and Hoiem, D. (2017). Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947.

[59] Lin, Z., Shi, J., Pathak, D., and Ramanan, D. (2021). The clear benchmark: Continual learning on real-world imagery. In *Thirty-fifth conference on neural information processing systems datasets and benchmarks track (round 2)*.

[60] Liška, A., Kočiský, T., Gribovskaya, E., Terzi, T., Sezener, E., Agrawal, D., de Masson d'Autume, C., Scholtes, T., Zaheer, M., Young, S., Austin, E. G.-M. S., Blunsom, P., and Lazaridou, A. (2022). Streamingqa: A benchmark for adaptation to new knowledge over time in question answering models. *arXiv preprint arXiv:2205.11388*.

[61] Liu, X., Leonardi, A., Yu, L., Gilmer-Hill, C., Leavitt, M., and Frankle, J. (2023). Knowledge distillation for efficient sequences of training runs. *arXiv preprint arXiv:2303.06480*.

[62] Lomonaco, V. and Maltoni, D. (2017). Core50: a new dataset and benchmark for continuous object recognition. In *Conference on robot learning*, pages 17–26. PMLR.

[63] Lomonaco, V., Pellegrini, L., Rodriguez, P., Caccia, M., She, Q., Chen, Y., Jodelet, Q., Wang, R., Mai, Z., Vazquez, D., et al. (2022). Cvpr 2020 continual learning in computer vision competition: Approaches, results, current challenges and future directions. *Artificial Intelligence*, 303:103635.

[64] Lopez-Paz, D. and Ranzato, M. (2017). Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30.

[65] Maji, S., Kannala, J., Rahtu, E., Blaschko, M., and Vedaldi, A. (2013). Fine-grained visual classification of aircraft. https://arxiv.org/abs/1306.5151.

[66] Michieli, U. and Zanuttigh, P. (2019). Incremental learning techniques for semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision workshops*, pages 0–0.

[67] Mirzadeh, S. I., Farajtabar, M., and Ghasemzadeh, H. (2020a). Dropout as an implicit gating mechanism for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 232–233.

[68] Mirzadeh, S. I., Farajtabar, M., Pascanu, R., and Ghasemzadeh, H. (2020b). Understanding the role of training regimes in continual learning. *Advances in Neural Information Processing Systems*, 33:7308–7320.

[69] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *Advances in Neural Information Processing Systems (NeurIPS) Workshops*. https://storage.googleapis.com/pub-tools-public-publication-data/pdf/37648.pdf.

[70] Ni, Z., Wei, L., Tang, S., Zhuang, Y., and Tian, Q. (2023). Continual vision-language representaion learning with off-diagonal information. *arXiv preprint arXiv:2305.07437*.

[71] Nilsback, M.-E. and Zisserman, A. (2008). Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*. https://ieeexplore.ieee.org/document/4756141.

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

[73] Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. V. (2012). Cats and dogs. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. https://ieeexplore.ieee.org/document/6248092.

[74] Plummer, B. A., Wang, L., Cervantes, C. M., Caicedo, J. C., Hockenmaier, J., and Lazebnik, S. (2015). Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Proceedings of the IEEE international conference on computer vision*, pages 2641–2649.

[75] Prabhu, A., Torr, P. H., and Dokania, P. K. (2020). Gdumb: A simple approach that questions our progress in continual learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 524–540. Springer.

[76] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

[77] Ramaswamy, V. V., Lin, S. Y., Zhao, D., Adcock, A. B., van der Maaten, L., Ghadiyaram, D., and Russakovsky, O. (2023). Beyond web-scraping: Crowd-sourcing a geodiverse datase. `https://arxiv.org/abs/2301.02560`.

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

[79] Recht, B., Roelofs, R., Schmidt, L., and Shankar, V. (2019). Do ImageNet classifiers generalize to ImageNet? In *International Conference on Machine Learning (ICML)*. `http://proceedings.mlr.press/v97/recht19a.html`.

[80] Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

[81] Robins, A. (1995). Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146.

[82] Rojas, W. A. G., Diamos, S., Kini, K. R., Kanter, D., Reddi, V. J., and Coleman, C. (2022). The dollar street dataset: Images representing the geographic and socioeconomic diversity of the world. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*. `https://openreview.net/forum?id=qnfYsave0U4`.

[83] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

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

[85] Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., et al. (2022). Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294.

[86] Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., and Hadsell, R. (2018). Progress & compress: A scalable framework for continual learning. In *International conference on machine learning*, pages 4528–4537. PMLR.

[87] Seitzer, M. (2020). pytorch-fid: FID Score for PyTorch. `https://github.com/mseitzer/pytorch-fid`. Version 0.3.0.

[88] Shirali, A. and Hardt, M. (2023). What makes imagenet look unlike laion. *arXiv preprint arXiv:2306.15769*.

[89] Srinivasan, T., Chang, T.-Y., Pinto Alva, L., Chochlakis, G., Rostami, M., and Thomason, J. (2022). Climb: A continual learning benchmark for vision-and-language tasks. *Advances in Neural Information Processing Systems*, 35:29440–29453.

[90] Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. (2011). The german traffic sign recognition benchmark: a multi-class classification competition. In *International Joint Conference on Neural Networks (IJCNN)*. `https://ieeexplore.ieee.org/document/6033395`.

[91] Stoica, G., Bolya, D., Bjorner, J., Hearn, T., and Hoffman, J. (2023). Zipit! merging models from different tasks without training. *arXiv preprint arXiv:2305.03053*.

[92] Sutton, R. S. (1986). Two problems with backpropagation and other steepest-descent learning procedures for networks. In *Proc. of Eightth Annual Conference of the Cognitive Science Society*, pages 823–831.

[93] Thengane, V., Khan, S., Hayat, M., and Khan, F. (2022). Clip model is an efficient continual learner. *arXiv preprint arXiv:2210.03114*.

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

[95] Thomee, B., Shamma, D. A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., and Li, L.-J. (2016b). YFCC100M: The new data in multimedia research. *Communications of the ACM*. `https://arxiv.org/abs/1503.01817`.

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

[97] Veeling, B. S., Linmans, J., Winkens, J., Cohen, T., and Welling, M. (2018). Rotation equivariant CNNs for digital pathology. `https://arxiv.org/abs/1806.03962`.

[98] Veniat, T., Denoyer, L., and Ranzato, M. (2020). Efficient continual learning with modular networks and task-driven priors. *arXiv preprint arXiv:2012.12631*.

[99] Verwimp, E., Yang, K., Parisot, S., Hong, L., McDonagh, S., Pérez-Pellitero, E., De Lange, M., and Tuytelaars, T. (2023). Clad: A realistic continual learning benchmark for autonomous driving. *Neural Networks*, 161:659–669.

[100] Wang, H., Ge, S., Lipton, Z., and Xing, E. P. (2019). Learning robust global representations by penalizing local predictive power. In *Advances in Neural Information Processing Systems (NeurIPS)*. `https://arxiv.org/abs/1905.13549`.

[101] Wang, J., Wang, X., Shang-Guan, Y., and Gupta, A. (2021). Wanderlust: Online continual object detection in the real world. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10829–10838.

[102] Wang, Z., Zhang, Z., Ebrahimi, S., Sun, R., Zhang, H., Lee, C.-Y., Ren, X., Su, G., Perot, V., Dy, J., et al. (2022). Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, pages 631–648. Springer.

[103] Wen, Y., Tran, D., and Ba, J. (2020). Batchensemble: an alternative approach to efficient ensemble and lifelong learning. *arXiv preprint arXiv:2002.06715*.

[104] Wenzek, G., Lachaux, M.-A., Conneau, A., Chaudhary, V., Guzmán, F., Joulin, A., and Grave, E. (2019). Ccnet: Extracting high quality monolingual datasets from web crawl data. *arXiv preprint arXiv:1911.00359*.

[105] Wiener, J. and Bronson, N. (2014). Facebook's top open data problems. `https://research.facebook.com/blog/2014/10/facebook-s-top-open-data-problems/`. Accessed: 2023-09-28.

[106] Wortsman, M., Ilharco, G., Kim, J. W., Li, M., Kornblith, S., Roelofs, R., Lopes, R. G., Hajishirzi, H., Farhadi, A., Namkoong, H., et al. (2022). Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7959–7971.

[107] Xiao, J., Ehinger, K. A., Hays, J., Torralba, A., and Oliva, A. (2016). Sun database: Exploring a large collection of scene categories. *International Journal of Computer Vision (IJCV)*. `https://link.springer.com/article/10.1007/s11263-014-0748-y`.

[108] Young, P., Lai, A., Hodosh, M., and Hockenmaier, J. (2014). From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*. `https://aclanthology.org/Q14-1006/`.

[109] Zenke, F., Poole, B., and Ganguli, S. (2017). Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. PMLR.

[110] Zhai, X., Puigcerver, J., Kolesnikov, A., Ruyssen, P., Riquelme, C., Lucic, M., Djolonga, J., Pinto, A. S., Neumann, M., Dosovitskiy, A., Beyer, L., Bachem, O., Tschannen, M., Michalski, M., Bousquet, O., Gelly, S., and Houlsby, N. (2019). The visual task adaptation benchmark. `http://arxiv.org/abs/1910.04867`.

[111] Zheng, Z., Ma, M., Wang, K., Qin, Z., Yue, X., and You, Y. (2023). Preventing zero-shot transfer degradation in continual learning of vision-language models. *arXiv preprint arXiv:2303.06628*.

[112] Zhou, D.-W., Zhang, Y., Ning, J., Ye, H.-J., Zhan, D.-C., and Liu, Z. (2023). Learning without forgetting for vision-language models. *arXiv preprint arXiv:2305.19270*.

A TiC-CLIP: Benchmarks and Experimental Protocol

In this section, we introduce our benchmark (Fig. 2) focusing on the training of a vision-language foundation model with the objective of Contrastive Language Image Pretraining (CLIP) [76]). Notably, we train on image-text data that arrives sequentially unlike the conventional image-text datasets which are static (e.g. WiT in CLIP, DataComp in Gadre et al. [34]). We curate TIC-DataComp, TIC-YFCC, and TIC-RedCaps that are image-text pairs sourced from the internet which we augment with auxiliary time information. We also introduce dynamic evaluation tasks to assess performance of our continually trained models on data evolving with time. The goal of a learner is to train a *deployable* model at each step as new data becomes available with a fixed compute budget.

A.1 Benchmark Design: How we Create Time-Continual Datasets?

To instantiate continual training of CLIP, we extend existing image-text datasets with time information collected from the original source of the datasets. Our largest dataset is TIC-DataComp which contains 12.7 billion image-text pairs with "crawl-time" metadata. We create this dataset on top of the existing DataComp benchmark [34]. We also create TIC-YFCC and TIC-RedCaps on top of existing YFCC15M [94, 76] and Redcaps [25] datasets to highlight that our findings are broadly applicable to carefully curated datasets from diverse sources such as Reddit and Flickr. While time-related metadata is absent in the DataComp benchmark, it is available in the original releases of YFCC and Redcaps. Nevertheless, to the best of our knowledge, no prior work utilizes such time information for continual training of CLIP models. We show dataset statistics for all datasets, e.g., number of examples in each year in App. G.3.

TIC-DataComp We collect timestamps for the CommonPool dataset introduced in DataComp which contains 12.7B image-text pairs (not including 0.1B inaccessible ones). This dataset stands as the largest public image-text dataset to date. The source of DataComp is Common Crawl, which periodically releases web-crawled data snapshots, typically on a monthly basis since 2014 with new and updated webpages. To construct TIC-DataComp, we augment each image-text pair in DataComp with their *first* timestamp. We followed the same construction process as DataComp but retained only the image-text pair found in the earliest snapshot during the deduplication stage. This process provides timestamps at the granularity of months, spanning years 2014–2022. See App. G.7 for details on the construction process. We note that while this augmented time information may contain some noise, on average, we find it to be a reasonably accurate proxy for the upload time of web pages (see App. G.7).

Although our benchmark contains time information at the granularity of months, we limit our experiments to granularity of years by consolidating data for all months in a year. Similar to DataComp, our benchmark has an inclusive design, accommodating participants with varying levels of computational resources. In particular, we experiment with `medium`, `large`, and `xlarge` sizes from CommonPool. [34] leverage different filtering strategies to select the training subset. We are concerned that filtering techniques bias the selected training data. In App. G.1, we provide preliminary evidence that "Bestpool" filtering that uses off-the-shelf CLIP models, indeed biases the selected data to old time steps. Nevertheless, to highlight significance of our findings even for state-of-the filtering techniques, we experiment with both Bestpool and Basic filtering (no CLIP filtering) at `xlarge` scale. For `large` and `medium` scales, we only experiment with Basic filtering.

TIC-YFCC We experiment with the 15M subset of YFCC100M [94], namely YFCC15M, selected by OpenAI [76]. This filtering retains only images with natural text in captions. YFCC100M contains data from years 2008–2014 and was originally released with upload timestamps. We use this information to create continual splits at the granularity of years.

TIC-RedCaps RedCaps contains 12M image-caption pairs from manually curated set of subreddits across 2011–2020 [25]. We use the creation timestamps of the posts to create splits for continual learning. Similar to the other two datasets, we experiment at the granularity of years.

A.2 Evaluation Testbed

Dynamic tasks We leverage the temporal information in our benchmarks to create *dynamic evaluation* tasks. Here, the test data comprises samples varying over years as the world evolved. For our largest dataset which is TIC-DataComp, we create dynamic tasks for both retrieval and classification as described below. (examples in Figure 3 and additional examples in App. G.5):

I. *Dynamic retrieval task*: To create a retrieval task, we sample a batch of IID image-text pairs from different timestamps and evaluate text retrieval performance given the corresponding image (similarly, image retrieval given the corresponding text). We refer to the dataset as TIC-DataComp-Retrieval.

II. *Dynamic classification task*: We also create a classification dataset TIC-DataComp-Net with ImageNet classes from CommonPool and augmented with timestamps. Inspired by LAIONNet [88], we first filter examples where the corresponding caption contains one and only one of the synsets of ImageNet. Then we only retain examples where the similarity between ImageNet synset definition and the caption exceeds a threshold of 0.5. We evaluate the similarity using an off-the-shelf sentence embedding model [80]. Crucially, unlike LAIONNet, we do not filter the image-text pairs with CLIP similarity scores to avoid biasing the selection process. We describe the construction process in more details in App. G.5. On TIC-DataComp-Net, we report average accuracy over all classes and over selected nodes (e.g., motor vehicles) at each time step.

Similarly, we create retrieval tasks for TIC-YFCC and TIC-RedCaps. Note that we remove the extracted image-text pairs for dynamic retrieval and classification tasks from the training sets. Evaluations on dynamic tasks are done in a zero shot manner.

Static tasks We also evaluate models on numerous classification and retrieval tasks in a zero-shot manner as in Radford et al. [76]. In particular, we consider 28 standard tasks: 27 image classification tasks, e.g., ImageNet and its 6 distribution shifts (e.g., ImageNetv2, ImageNet-R, ImageNet-Sketch, and Objectnet), datasets from VTAB and Flickr30k retrieval task. We refer to these as *static evaluation* tasks. We list all the datasets in App. G.2.

Evaluation metrics We define metrics for classification tasks and retrieval tasks based on *accuracy* and *Recall@1*, respectively. Let T represent the number of time steps for which we have data. For each training method, we generate a total of T models, each corresponding to the end of training at a particular time step. For static datasets (e.g., ImageNet), we report average performance of T models. However, when dealing with dynamic evaluation datasets, we assess the performance of each of the T models on evaluation datasets collected at all time steps. Consequently, for each model and a dynamic evaluation task, we obtain T performance values. We represent these values using the performance matrix $\mathcal{E}$, where each entry $\mathcal{E}_{i,j}$ signifies the performance of the model obtained after observing training data at time step i when evaluated on a dataset from time step j. The performance matrix $\mathcal{E}$ can also be succinctly summarized using three standard metrics commonly employed in continual learning evaluations [59, 64, 26]:

- *In-domain performance*: average performance at each training time step (i.e., the diagonal of $\mathcal{E}$)
- *Backward transfer*: average on time steps before each training step (i.e., the lower triangular of $\mathcal{E}$)
- *Forward transfer*: average on time steps following each training step (i.e., the upper triangular of $\mathcal{E}$)

While the static tasks capture performance on standard benchmarks, dynamic tasks capture problems due to distribution shift (for forward transfer) and forgetting (for backward transfer). The goal in our benchmark is to develop continual learning methods that maximize performance on static tasks while simultaneously optimizing for performance on dynamic tasks.

A.3 Experimental Protocol For Training

Streaming protocol We follow a streaming protocol, where data is progressively revealed to the learner in large batches with the objective of achieving a deployable model as early as possible after each batch arrives. We conduct experiments with data streaming at the granularity of years and our benchmark supports future research at the granularity of months. Additionally, as the amount of data from earlier time steps is limited (see App. G.3), we aggregate data from the earlier time steps into a single larger batch and timestamp it by the latest year in the range. After this aggregation, we have 7 time steps for TIC-DataComp (2016–2022) and 4 for both TIC-YFCC (2011–2014) and TIC-RedCaps (2017–2020). While the number of image-text pairs revealed at each time step are of similar orders of magnitude, the exact number does vary across steps and we do not artificially alter the sizes.

Memory budget We allow methods to use the last model checkpoint at each step as the cost of keeping one checkpoint per month is often negligible. In contrast, the cost of retaining old data can be high and might not be permitted due to data expiration policies. Thus, along with studying methods that retain all old data, we also explore strategies that restrict data persistence (see Sec. B for details).

Compute budget To ensure a fair comparison between methods, we establish a consistent total compute budget, quantified in terms of Multiply-Accumulate Operations (MACs), and allocate it

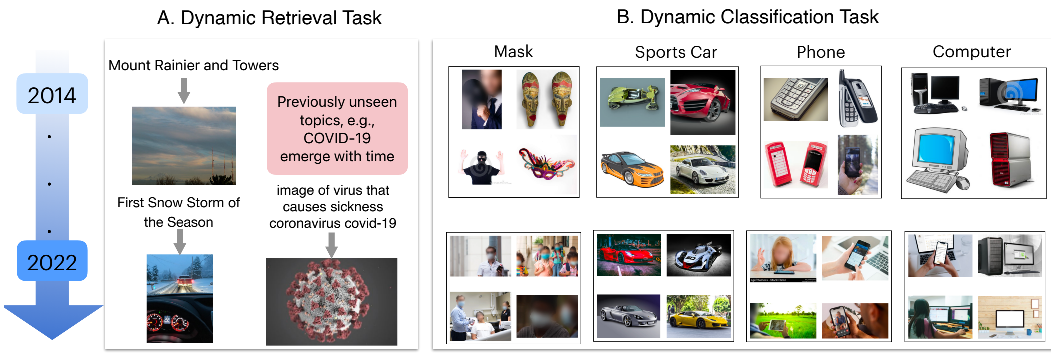


Figure 3: **Distribution of examples changes from 2014 to 2022 in our dynamic evaluation tasks.** *(Left)* Samples for text to image retrieval. For new timestamps, images from novel concepts appear (e.g., COVID-19). *(Right)* Samples from our classification task for 4 categories. We observe that not only objects evolve over time but also images from recent timestamps are captured more in the wild.

evenly for training at every time step. Unless specified otherwise, for all methods except Oracle and LwF, we use the same compute budget. For experiments on TIC-DataComp, we refer to compute configurations in DATACOMP for overall compute. For TIC-RedCaps and TIC-YFCC, we use compute of order `medium` scale in TIC-DataComp. Compute budget details are in App. G.4.

A.4 Analyzing Distribution Shifts in the Constructed Benchmarks

TIC-DataComp analysis through the lens of constructed evaluation tasks First, we qualitatively analyze the examples in our retrieval and classification dataset (Fig. 3). We observe that over time, in the retrieval task, new concepts like COVID-19 emerge. Likewise, certain ImageNet classes evolve, such as the shift from "masquerad" masks to "surgical/protective" masks in their definitions. Moreover, as time evolves, we observe that image quality improves and more images tend to appear in the wild in contrast to centered white background images. Next, we compare performance of OpenAI and OpenCLIP models on our datasets. Here, we only present the main findings, and delegate a detailed discussion to App. G.6. We observe a significant performance gap between OpenAI and OpenCLIP models on our dynamic retrieval task (Fig. 1). This gap widens notably on retrieval queries where captions mention COVID-19. On the other hand, OpenAI and OpenCLIP models exhibit similar robustness for retrieval on data coming from Flickr highlighting that data from some domains do not exhibit shifts that cause performance drops. For our classification task, we observe a very small drop ($\approx 1\%$) when averaged across all categories. However, we observe a substantial gap on specific subtrees in ImageNet. For example, classes in "motor vehicle" subtree show an approximate 4% performance drop, when comparing OpenAI and OpenCLIP models. These findings highlight that while overall ImageNet classes may remain timeless, certain categories tend to evolve faster than others. Our qualitative and quantitative analysis on TIC-DataComp clearly highlights evolution of distributions and captures different properties than standard benchmarks.

Quantitative analysis on TIC-YFCC We analyze TIC-YFCC using off-the-shelf sentence and image encoders. We first embed images from different time steps with an OpenAI CLIP encoder and then compute Frechet Inception Distance (FID; Seitzer [87]). As time progresses, we observe that FID distance increases with respect to data from first time step (Fig. 15 in App. G.6). Similarly, we use pretrained sentence transformer to extract top-5 categories from Wordnet Nouns for each caption. We observe that the TV distance over distribution of WordNet Nouns evolves over time when compared to data from the first time step. More details in App. G.6.

B TiC-CLIP: How to Continually Train CLIP Models?

In this section, we lay out different methods specifically focus on the following questions (Tab. 3): (i) How to utilize/replay data from previous time steps; (ii) How to leverage previously trained model checkpoints? (iii) What should be the training/optimization procedure?

Table 3: Table summarizing our methods. D: data size in each step, T total time steps, t: current time step, C: compute budget (iterations).

Method	Each Step			Total
	Train Size	Init.	Compute	Compute
Cumulative-All	tD	Last	C	TC
Cumulative-Exp	$2D$	Last	C	TC
Cumulative-Equal	$2D$	Last	C	TC
Sequential	D	Last	C	TC
Restart	tD	Rand	C	TC
Patching	D	Last Patch	C	TC
LwF	D	Last	$1.2 \times C$	$1.2 \times TC$
Oracle**	tD	Rand	tC	$\frac{(T+1)T}{2}C$

Data replay methods initialized from the last checkpoint demonstrate strong performance on standard continual learning benchmarks (Appendix D.1). We consider replay methods with/without initialization from last checkpoint(s):

I. **Oracle**: Train a CLIP model from scratch (i.e., random initialization) on all image-text data received till time t using a large compute budget of $t \times C$. Oracle represents a *prohibitively expensive* method that is the most common practice in training large-scale foundation models. The goal of other methods is to perform as close as possible to the Oracle within their limited budget.

II. **Cumulative**: Train each model initialized from last checkpoint on the union of all data up to t with compute budget C. This method is analogous to Experience Replay (ER; [81, 38]) but with substantially larger buffers than common in the continual learning literature. Given a fixed buffer size for each past step, we observe minimal to no difference between random subsampling and other strategies. After sampling the replay data, we randomly shuffle it together with new data for training. We consider the following strategies for sampling buffer sizes per step:

- **-All**: Replay all previous data.
- **-Exp**: Replay a buffer of size D and reduce the amount of old data by half at each step. For example, at 4-th time step, we retain $D/2, D/2, D$ of old data and at 4-th, we retain $D/8, D/8, D/4, D/2$ of old data. Along with D data from current step, this method trains on at most $2D$ data in each step.
- **-Equal**: Replay a buffer of size D but split the buffer equally among all previous years. For example, at 4-th step, we retain $D/3, D/3, D/3$ of old data. Along with D data from current time step, this method trains on at most $2D$ data in each step.

III. **Sequential**: Train *only* on the new data starting from the best checkpoint of the previous time step. Sequential is similar to Cumulative but without any replay buffer.

IV. **Restart**: Train each model from scratch (i.e., random initialization) on all the data till time t for compute budget C. Restart is similar to the Oracle but with compute budget C at each time step and similar to Sequential but with random initialization. As such, Restart helps us understand the *forward transfer* and *loss of plasticity* in our benchmark [3, 28].

V. **LwF**: Train only on the new data with an additional loss that regularizes the model by KL divergence between the image-text similarity matrix of last checkpoint and current model on each mini-batch [58, 27].

VI. **Patching**: We use sequential patching from Ilharco et al. [43] and initialize from a patched model of last step and train only on the new data. To obtain patched model at each time step, we apply weight interpolation with the patched model (if any) trained at time step $t-1$ and model trained at time step t. We tune the mixing coefficients by optimizing average retrieval performance on previous tasks.

Learning rate schedule The defacto Learning Rate (LR) schedule for training CLIP models is an initial linear increase to a maximum value, i.e., warm up, followed by a cosine decay [76, 34]. We default to using cosine LR schedule for each sequential run, resulting in a cyclic schedule and observe a significant increase in training loss early in subsequent runs when the LR is high. However, as training progresses, we observe that the increased loss decreases at a faster rate (when compared to training from scratch) allowing us to train with cyclic schedules.

Other Training details and hyperparameters Unless specified otherwise, we closely follow the original CLIP training recipe [76]. We train the CLIP variant with ViT-B/16 as the image encoder [29]. All training and hyperparameters can be found in App. H.2.

Table 4: **Zero shot performance on our time-continual benchmarks.** * and ** denote methods that violate the compute budget. For static tasks, we tabulate accuracy of the models obtained on the final timestamp. For dynamic tasks, we tabulate forward/backward transfer and ID performance on retrieval tasks (Sec. A.3). For TIC-DataComp (XL), we include results with Bestpool filtering (basic filtering in Table 7). For all metrics, higher is better.

Benchmark	**Method**	**Compute** (MACs)	**Static Tasks**				**Dynamic Retrieval Tasks**		
			ImageNet	ImageNet dist. shift	Flickr30k	Average over 28 datasets	Backward Transfer	ID Performance	Forward Transfer
TIC-YFCC	Restart	3.4×10^{18}	5.2	3.6	3.0	4.0	13.2	41.4	18.6
	Sequential	3.4×10^{18}	17.3	10.5	15.9	11.6	42.2	48.4	23.7
	Patching	3.4×10^{18}	18.9	11.3	18.5	12.1	44.7	53.4	24.5
	Cumulative-Exp	3.4×10^{18}	24.1	14.3	20.4	15.4	60.4	60.1	27.1
	Cumulative-Equal	3.4×10^{18}	23.9	13.8	20.5	14.7	60.4	60.4	27.1
	Cumulative-All	3.4×10^{18}	**29.3**	**17.6**	**26.8**	**18.0**	**66.4**	**60.2**	**27.6**
	LwF*	4.1×10^{18}	16.9	9.8	14.7	10.5	36.6	56.0	23.2
	Cumulative-All*	3.6×10^{18}	**29.2**	**17.5**	**27.4**	**18.1**	**66.8**	**60.3**	**27.6**
	Oracle**	8.5×10^{18}	**29.2**	**17.0**	**25.9**	**18.0**	**66.1**	**61.8**	**26.9**
TIC-RedCaps	Restart	3.4×10^{18}	11.7	8.5	3.7	7.6	21.3	25.4	22.4
	Sequential	3.4×10^{18}	19.3	13.7	6.2	11.9	33.0	33.6	27.5
	Patching	3.4×10^{18}	21.3	15.2	7.7	14.0	34.8	34.8	27.8
	Cumulative-Exp	3.4×10^{18}	27.3	19.1	10.5	16.3	44.5	42.0	32.6
	Cumulative-Equal	3.4×10^{18}	27.8	19.4	10.0	16.7	44.4	42.0	32.6
	Cumulative-All	3.4×10^{18}	**32.2**	18.7	14.5	**19.7**	**48.9**	**43.2**	**33.4**
	LwF*	4.1×10^{18}	21.6	14.8	8.2	13.5	35.4	36.0	28.4
	Cumulative-All*	3.6×10^{18}	**32.9**	**23.7**	**14.1**	**20.1**	**49.0**	**43.4**	**33.4**
	Oracle**	8.5×10^{18}	**32.7**	**22.7**	**14.3**	**20.6**	**48.5**	**43.1**	**33.4**
TIC-DataComp (M)	Sequential	3.0×10^{18}	19.2	16.4	16.4	15.0	25.7	26.4	14.9
	Patching	3.0×10^{18}	19.3	16.8	18.5	14.7	26.9	25.4	14.5
	Cumulative-Exp	3.0×10^{18}	22.1	18.4	20.4	16.7	31.7	27.1	**15.2**
	Cumulative-Equal	3.0×10^{18}	22.1	18.4	19.2	17.1	31.8	26.8	15.1
	Cumulative-All	3.0×10^{18}	24.0	20.2	20.9	17.9	33.8	26.4	15.1
	LwF*	3.8×10^{18}	19.2	16.5	17.7	14.3	25.6	26.6	14.9
	Cumulative-All*	3.9×10^{18}	**30.0**	**25.0**	**28.6**	**22.3**	**36.7**	**28.3**	**15.5**
	Oracle**	1.2×10^{19}	25.5	21.2	23.3	19.0	34.9	27.8	**15.6**
TIC-DataComp (L)	Sequential	2.7×10^{19}	44.7	37.4	48.4	32.7	52.6	**58.4**	41.1
	Patching	2.7×10^{19}	45.8	38.9	49.7	33.6	55.2	57.5	40.9
	Cumulative-Exp	2.7×10^{19}	47.3	39.6	50.8	35.0	60.4	**58.4**	**41.4**
	Cumulative-Equal	2.7×10^{19}	47.7	40.3	51.8	36.0	60.9	**58.2**	**41.4**
	Cumulative-All	2.7×10^{19}	48.9	41.3	50.9	36.3	62.1	57.3	41.2
	Cumulative-All*	4.1×10^{19}	53.0	**44.3**	**54.4**	**39.0**	63.0	57.8	41.2
	Oracle**	1.1×10^{20}	**53.6**	44.0	53.9	38.0	**64.3**	**58.6**	**41.8**
TIC-DataComp (XL)	Sequential	2.7×10^{20}	66.5	54.2	61.2	51.7	63.1	68.9	56.8
	Cumulative-All	2.7×10^{20}	71.6	58.8	65.1	55.7	**70.7**	**68.5**	**57.1**
	Cumulative-All*	3.5×10^{20}	**72.8**	60.4	66.5	**57.7**	**71.0**	**68.6**	**57.1**
	Oracle**	1.1×10^{21}	**73.3**	**61.3**	**68.0**	**58.1**	-	-	-

C Experiments and Results

Our main results are in Table 4 and more detailed plots on each dataset are in App. F.1. Recall, our goal is compete with an Oracle that re-trains from scratch every time new data is observed, both on dynamic and static tasks, while being computationally efficient. Here, we summarize our key findings:

Cumulative-All saves up to $4\times$ the cost. On dynamic evaluation tasks, we observe that Cumulative-All where we replay all the past data, achieves performance close to the Oracle (within 1%) using significantly less compute ($4\times$ less on TIC-DataComp and $2.5\times$ less on TIC-YFCC and TIC-RedCaps). On static tasks, the gap remains small at small scales but grows to 4.7% on `large`, 1.8% on `xlarge` Bestpool, and 4% on `xlarge` Basic (see Table 4 and Table 7). In these cases, training Cumulative models with slightly extra compute bridges the gap while remaining at least $2.7\times$ more computationally efficient (see rows with * in Table 4). This highlights that with unconstrained access to past data, we can simply train sequentially and save significant computational resources.

At scale, Sequential has strong forward transfer but lacks on static tasks. On TIC-YFCC and TIC-RedCaps, which are at the smallest scale, we observe a significant gap ($> 10\%$) between Sequential (with no data replay) and Oracle on all tasks. On the other hand, on all scales in TIC-DataComp, Sequential shows strong performance on forward transfer and ID dynamic evaluations. However, on static tasks and backward transfer evaluations, Sequential significantly underperforms the Oracle.

Patching and LwF only improve slightly over Sequential but significantly lag behind Cumulative-All. On static tasks, LwF improves over Sequential by <1.5%. However, its computation cost is higher than even Cumulative-All* which outperforms LwF on all tasks. Similarly, Patching improves over Sequential on backward transfer on all datasets (e.g., 3% boost on TIC-DataComp L) highlighting that Patching combines benefits of the previously patched model and the new Sequential model.

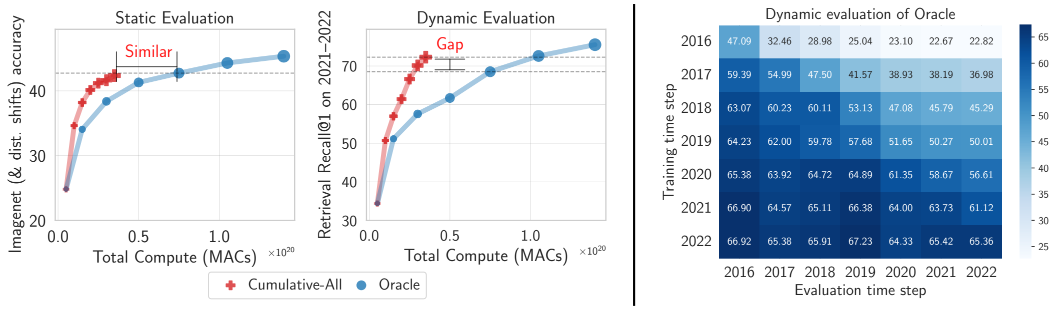


Figure 4: *(Left)* **Dynamic and static evaluations rank models differently**. Models with similar performance on static datasets, have $> 6\%$ difference on retrieval task from 2021-2022 TIC-DataComp (L). Different points denote models trained sequentially over time. *(Right)* **Performance of Oracle on future time steps drops highlighting distribution shift in dataset**. Each row evaluates the Oracle trained on TIC-DataComp (L) at a particular time step across all dynamic retrieval tasks.

However, the benefits on static tasks are relatively modest. These results hint that to continuously improve on static tasks with time, replaying old data as in Cumulative-All plays a crucial role.

-Exp and -Equal significantly reduce replay buffer size and maintain static task performance and backward transfer. Recall, that -Exp and -Equal reduce the replay buffer size to a D of old data and overall reduce the buffer size by $3.5\times$ for TIC-DataComp datasets. While reducing the buffer sizes, these methods still achieve performance close to Cumulative-All (within 2%) on both static and dynamic tasks, with -Equal often better than -Exp. As we go to large scale, e.g., from `medium` to `large`, the gap between these methods and Cumulative-All reduces. These findings demonstrate that even a small amount of replay data from old time steps stays competitive with replaying all data and significantly improves over no replay at all.

Warm up helps training on data from first time step, but hurts on subsequent time steps. Cosine LR is commonly coupled with an initial warm-up that linearly increases the LR from zero to maximum LR. We investigate the effectiveness of warm-up in first versus subsequent time steps. Surprisingly, we observe that not using warmup for subsequent training runs is *strictly* more beneficial than using warm up on both static and dynamic tasks. In particular, on TIC-DataComp (L), we observe about 2.5% improvement in accuracy on ImageNet when not using warmup with Cumulative (see App. F.3). Moreover, we also ablate over not using warm up for the first training run and observe a drop of approximately 4.7% accuracy in the first time step on TIC-DataComp (L). Hence, we default to using warmup when training on the first time step and not using it on the subsequent time steps with all methods except for training on TIC-DataComp (XL) where we add a smaller warm up (10% of the warm up iterations used in first step) to stabilize training.

Same maximum LR works best across all runs when using cosine schedule. We ablate on TIC-DataComp (M) to investigate how to change LR after training on data from the first time step. Unlike conventional pretraining and finetuning settings where LR is typically decreased for subsequent training, we observe that decaying maximum LR for subsequent steps in our setup hurts on static and dynamic tasks and consequently, we use same maximum LR across our runs (see App. F.3).

Filtering strategy changes the ordering of performance on static and dynamic retrieval tasks. We observe that while bestpool filtering models outperform basic filterining models on TIC-DataComp (XL) by 6% on static tasks, they underperform by over 5% on dynamic retrieval task (see Fig. 9).

Dynamic tasks provide complimentary information for model selection compared to static tasks. Choosing models solely based on static task performance may inadvertently select models that underperform on dynamic tasks. For example, Cumulative models that show relatively modest improvements on static tasks continue to improve by $> 6\%$ for retrieval on 2021-2022 (Fig. 4).

Cumulative-All remains competitive to Oracle even on ImageNet on up to 8 splits. CLIP models are often trained for fewer epochs and are typically not trained until they reach an "overfitting" regime. Moreover, CLIP models are trained with noisy supervision of web data with a contrastive loss. Here, we investigate how Cumulative-All

Table 5: ImageNet continual training with up to 8 splits. Cumulative-All remains close to Oracle.

Method	Number of splits			
	1 (Oracle)	2	4	8
Cumulative-All	80.9	80.8	80.6	80.0

(with no extra budget) performs when compared to Oracle when training is done for longer with high-quality data. Specifically, we assess Cumulative-All on 2, 4 and 8 IID splits including the full dataset (see App. H.1 for details). Table 5 summmarizes our key findings. Notably, even with up to 8 splits, the difference in accuracy between Oracle and Cumulative-All remains below 0.9% on the ImageNet holdout test set. These results underscore the feasibility of continual training with Cumulative-All even on ImageNet.

D Continual Learning benchmarks

We introduce a large-scale image-text benchmark with web scale streaming image text pairs especially developed for studying how efficiently one can get a fresh CLIP model with new incoming batches of data. Table 6 compares the proposed benchmark with the existing ones. It's noteworthy to say that this table is not aimed to be an exhaustive list of all CL datasets, but, is only intended to present the most popular benchmarks in each domain for the sake of comparison with our curated benchmarks. We note that for language modeling tasks the number of examples/documents and for detection tasks the number of labeled objects/bounding boxes is reported as the number of samples.

Table 6: Comparison with continual learning benchmarks.

Benchmark	# Samples	Years	Time-Continual	Image-Text	Task
Split-MNIST [36]	60K	1998	✗	✗	Classification
Perm-MNIST [36]	60K	1998	✗	✗	Classification
Rot-MNIST [64]	60K	1998	✗	✗	Classification
Split-CIFAR-100 [109]	50K	2008	✗	✗	Classification
Split-MINI-ImageNet [15]	50K	2009	✗	✗	Classification
Split-ImageNet [103]	1.2M	2009	✗	✗	Classification
Split-ImageNet-R [102]	30K	2019	✗	✗	Classification
CORe50 [62]	165K	2017	✗	✗	Detection
CLAD [99]	23K	2021	✗	✗	Detection
WANDERLUST [101]	326K	2021	✓	✗	Detection
Inc-PASCAL [66]	11K	2012	✗	✗	Segmentation
Inc-ADE20K [13]	20K	2012	✗	✗	Segmentation
StreamingQA [60]	100K	2007–2020	✓	✗	Question Answering
TemporalWiki [45]	32M	2021	✓	✗	Language Modeling
CKL [46]	30K	2019-2021	✗	✗	Language Modeling
CTrL [98]	300K	1998-2017	✗	✗	Classification
CLOC [12]	39M	2006-2014	✓	✗	Classification
CLEAR [59]	7.8M	2004-2014	✓	✗	Classification
NEVIS [10]	8M	1992-2021	✓	✗	Classification
Mod-X [70]	156K	2014	✗	✓	Retrieval
CLiMB [89]	1.3M	2013-2021	✗	✓	Classification
TiC-YFCC	15M	2008-2014	✓	✓	Retrieval / ZS Classification
TiC-RedCaps	12M	2011-2020	✓	✓	Retrieval / ZS Classification
TiC-DataComp	100M/1B/12B	2014-2022	✓	✓	Retrieval / ZS Classification

D.1 Other Related Work

Neural networks trained on new data suffer from catastrophic forgetting of prior knowledge [92, 36]. Continual learning literature has focused on benchmarks and methods to address this challenge [37] while concentrating on domain, class, or task incremental benchmarks [42, 96] with artificial task boundaries (e.g., Split-CIFAR, Perm-MNIST). This results into benchmarks with minimal or no meaningfully evolution between adjacent tasks. Continual learning methods can be categorized broadly into i) regularization ii) replay, and iii) architecture-based methods. Regularization methods push the model to change slowly in the directions of prior knowledge and often incur additional memory/compute costs [51, 67, 68, 32]. Data replay methods retain all or a subset of the prior data for either retraining or regularization [64, 78, 14]. Simple replay-based baselines can surpass various methods on standard benchmarks [63, 4, 75]. Lastly, architecture-based methods expand the model as new tasks arrive which limits their applicability in continually evolving environments [86, 84].

Real-world machine learning has recently been dominated by training of large-scale foundation models that flourish with scale [50, 18]. Particularly, vision-language models have demonstrated scalability with data size leading to growing compute requirements [76, 47, 57]. Continual learning of foundation models would significantly reduce the costs and increase quick adaptability. While some recent works have started to introduce large-scale continual learning benchmarks, they are not naturally time-continual and are comparatively much smaller in scale [70, 89]. Proposing methods, [93] use the zero-shot capability of CLIP to evaluate on standard continual learning benchmarks without any training. [27] focus on continual fine-tuning of CLIP models on classification tasks and proposes an adaptation of LwF. Model averaging methods aim at interpolating between the performance of multiple contrastively pretrained models and classification-finetuned copies [106, 43, 91].

E Takeaways and Future Work

In conclusion, we view TIC-DataComp as the initial stride toward the continual training of large-scale vision-language foundation models. We aspire to empower the research on large-scale continual-learning through our new benchmark and preliminary results obtained using simple baselines.

There are several pivotal directions for future work: (i) Reduce the replay buffer size while maintaining the performance on static evaluation tasks and backward-transfer; (ii) Compare our baselines on continually streaming data at finer granularity, e.g., streaming data at the monthly level; (iii) Investigate alternate learning rate schedules (e.g., Const-Cosine) that are forward looking, and are better suited to continual learning; (iv) Better data filtering techniques that are more inclusive of future data; (v) Expand our problem setup to encompass the training of other large-scale foundation models.

F Additional Experimental Results

F.1 Detailed Results on Our Benchmarks

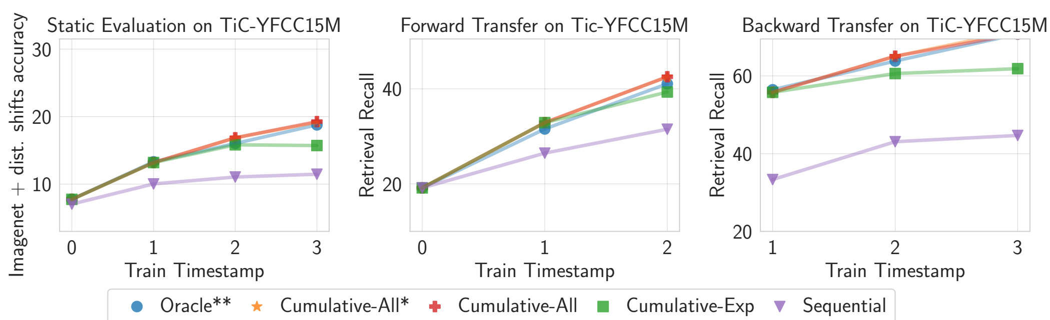


Figure 5: *Static and dynamic evaluation with selected methods on* TIC-*YFCC.* As we get more data, models (with all methods) improve on both static and forward transfer on dynamic tasks but methods with limited replay buffer start performing slightly worse for backward transfer.

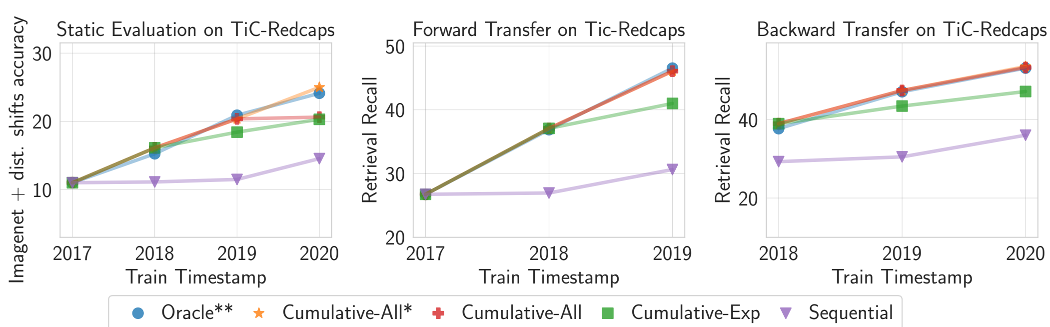


Figure 6: *Static and dynamic evaluation with selected methods on* TIC-*RedCaps.* As we get more data, models (with all methods) improve on both static and forward transfer on dynamic tasks but methods with limited replay buffer start performing slightly worse for backward transfer.

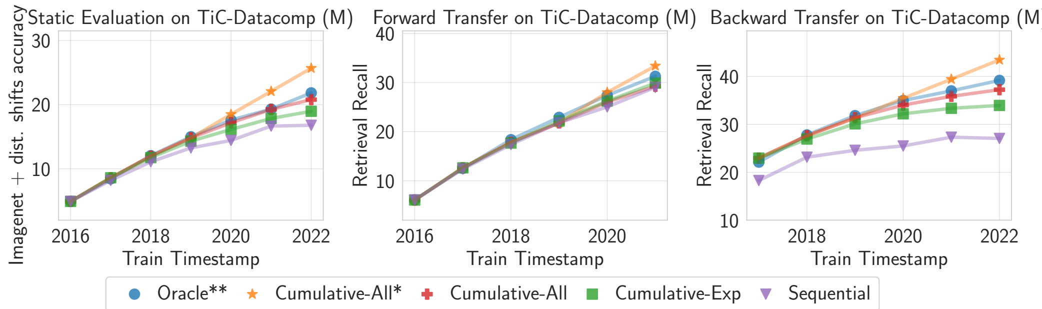


Figure 7: *Static and dynamic evaluation with selected methods on* TIC-*DataComp (medium).* As we get more data, models (with all methods) improve on both static and forward transfer on dynamic tasks but methods with limited replay buffer start performing slightly worse for backward transfer.

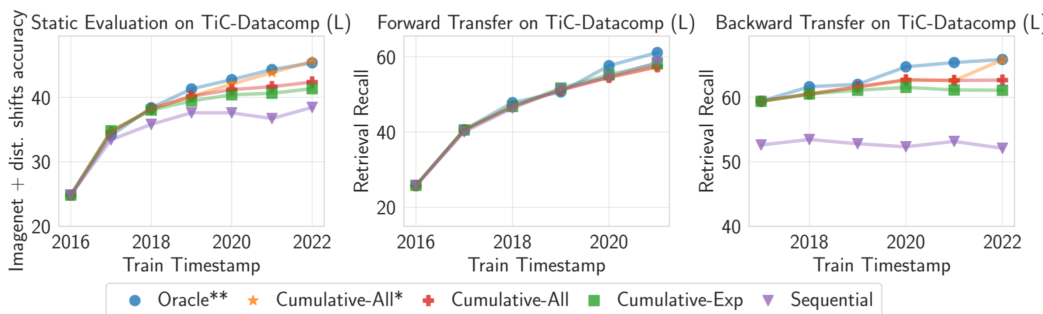


Figure 8: *Static and dynamic evaluation with selected methods on* TIC-*DataComp (large).* As we get more data, models (with all methods) improve on both static and forward transfer on dynamic tasks but methods with limited replay buffer start performing slightly worse for backward transfer.

F.2 Results with Basic Filtering on TIC-DataComp XL

Table 7: **Zero shot performance on our time-continual benchmarks.** * and ** denote methods that violate the compute budget and use extra compute. For static tasks, we tabulate accuracy of the models obtained on the final timestamp. For dynamic tasks, we tabulate forward transfer, backward transfer and ID performance. For all metrics, higher is better.

Benchmark	**Method**	**Compute** (MACs)	Static Tasks				Dynamic Retrieval Tasks		
			ImageNet	ImageNet dist. shift	Flickr30k	Average over 28 datasets	Backward Transfer	ID Performance	Forward Transfer
TIC-DataComp (XL)	Cumulative-All	2.7×10^{20}	63.5	52.0	62.8	47.5	64.6	55.5	47.6
	Sequential	2.7×10^{20}	60.2	48.9	62.4	44.4	51.6	50.3	45.0
	Oracle**	1.1×10^{21}	66.0	54.0	63.8	49.6	-	-	-

Comparing basic and bestpool filtering. We observe that while bestpool filtering models outperform basic filterining models on TIC-DataComp (XL) by 6% on static tasks, they underperform by over 5% on dynamic retrieval task (see Fig. 9).

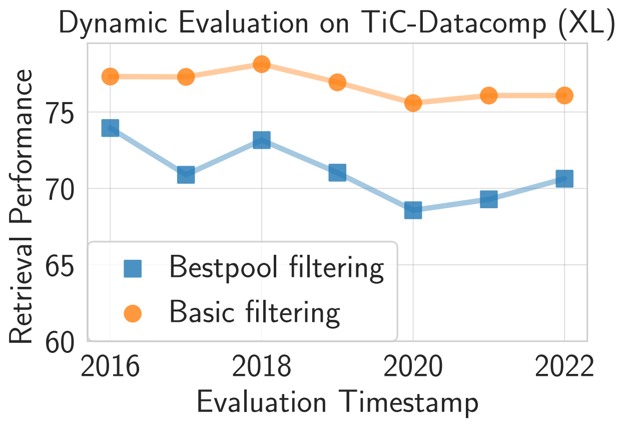


Figure 9: Comparing Oracle models trained on Bestpool and Basic filtering. Our results clearly highlight that Basic filtering performs better than Bestpool filtering on dynamic retrieval task. However, on static tasks, the order is reversed.

F.3 Ablations over learning rate warmup and Max LR

Table 8: **Zero shot performance on our time-continual benchmarks with and without initial LR wamrup for subsequent runs.** For all metrics, higher is better.

Benchmark	Method	Static Tasks				Dynamic Retrieval Tasks		
		ImageNet	ImageNet dist. shift	Flickr30k	Average over 28 datasets	Backward Transfer	ID Performance	Forward Transfer
TIC-DataComp (M)	Cumulative-All (w/o warmup)	24.0	20.2	20.9	17.9	33.8	26.4	15.1
	Cumulative-All (w warmup)	23.0	19.5	19.2	17.4	33.1	26.1	14.8
TIC-DataComp (L)	Cumulative-All (w/o warmup)	48.9	41.3	50.9	36.3	62.1	57.3	41.2
	Cumulative-All (w warmup)	46.4	39.3	47.8	34.9	60.4	56.3	39.6

Table 9: Cumulative experiments on TIC-DataComp (M) with different maximum learning rates. Our default choice is 0.00025. Performance reported on ImageNet. At maximum learning rate 0.001, the runs crashed with Nan in loss.

Method	Max LR				
	0.00005	0.0001	0.00025	0.0005	0.001
Cumulative-All	16.3	19.0	24.0	10.1	–

F.4 OpenCLIP models obtained by retraining after removing any duplicate examples from the test set

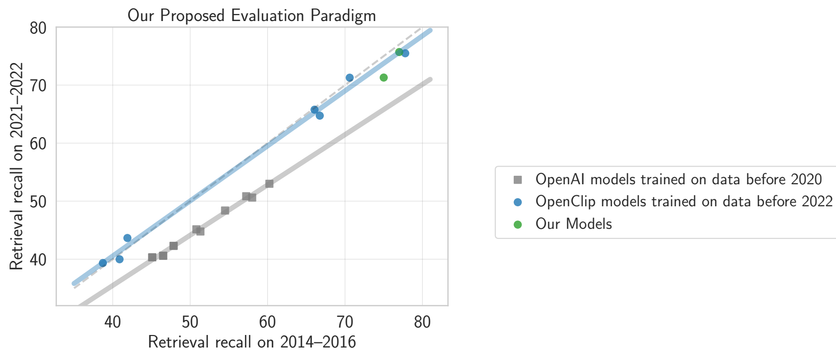


Figure 10: We replicate Open CLIP models by training from scratch and removing duplicates from the evaluation dataset. We observe that trends continue to hold.

G Additional Benchmark Details

G.1 Filtering ablations on TIC-DataComp

For Basic Filtering, Gadre et al. [34] performs the following three steps: filter by English language (using fasttext [49]), filter by caption length over two words and 5 characters, and filter by image sizes with smallest dimension over 200 pixels and aspect ratio above 3. We do not default to other filtering techniques that use off-the-shelf CLIP models from Gadre et al. [34] to avoid biasing dataset selection from each time step. In Fig. 11 we show that "Bestpool" filtering (which filters image-text pairs with CLIP scores and ImageNet image embeddings) biases dataset selection to preferring old time step data over new timestamp data. Moreover, we also show that models trained with Bestpool filtering are less robust when evaluated on our dynamic tasks from 2021-2022. Nevertheless, for completeness and to highlight significance of our findings even for state-of-the-art filtering techniques, we perform one set of continual learning experiments with Bestpool filtering at `xlarge` scale.

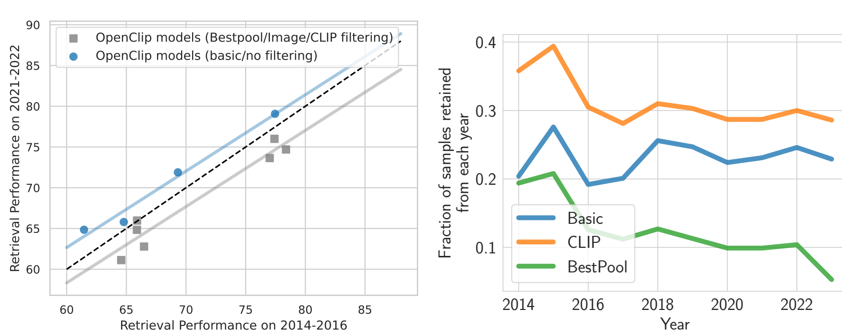


Figure 11: (Left) Gap in retrieval performance for different OpenCLIP models that use different filtering techniques. (Right) Reduction in TIC-DataComp data at different times with different filtering techniques. This clearly highlights that there is a selection bias towards retaining more old data for CLIP/BestPool filtering. No such bias exists for basic filtering.

G.2 Static Datasets considered for evaluation

Table 10: Evaluation tasks borrowed from Gadre et al. [34].

Task type	Dataset	Task	Test set size	Number of classes	Main metric
Classification	Caltech-101 [33]	Object recognition	6,085	102	mean per class
	CIFAR-10 [54]	Visual recognition	10,000	10	accuracy
	CIFAR-100 [54]	Visual recognition	10,000	100	accuracy
	CLEVR Counts [48, 110]	Counting	15,000	8	accuracy
	CLEVR Distance [48, 110]	Distance prediction	15,000	6	accuracy
	Country211 [76, 95]	Geolocation	21,100	211	accuracy
	DTD [20]	Texture classification	1,880	47	accuracy
	EuroSAT [39, 110]	Satellite imagery recognition	5,400	10	accuracy
	FGVC Aircraft [65]	Aircraft recognition	3,333	100	mean per class
	Food-101 [11]	Food recognition	25,250	101	accuracy
	GTSRB [90]	Traffic sign recognition	12,630	43	accuracy
	ImageNet 1k [24]	Visual recognition	50,000	1,000	accuracy
	ImageNet Sketch [100]	Visual recognition	50,889	1,000	accuracy
	ImageNet V2 [79]	Visual recognition	10,000	1,000	accuracy
	ImageNet-A [41]	Visual recognition	7,500	200	accuracy
	ImageNet-O [41]	Visual recognition	2,000	200	accuracy
	ImageNet-R [40]	Visual recognition	30,000	200	accuracy
	KITTI distance [35, 110]	Distance prediction	711	4	accuracy
	MNIST [56]	Digit recognition	10,000	10	accuracy
	ObjectNet [6]	Visual recognition	18,574	113	accuracy
	Oxford Flowers-102 [71]	Flower recognition	6,149	102	mean per class
	Oxford-IIIT Pet [73, 110]	Pet classification	3,669	37	mean per class
	Pascal VOC 2007 [31]	Object recognition	14,976	20	accuracy
	PatchCamelyon [97, 110]	Metastatic tissue cls.	32,768	2	accuracy
	Rendered SST2 [110]	Sentiment classification	1,821	2	accuracy
	RESISC45 [16, 110]	Satellite imagery recognition	6,300	45	accuracy
	Stanford Cars [53]	Vehicle recognition	8,041	196	accuracy
	STL-10 [21]	Visual recognition	8,000	10	accuracy
	SUN-397 [107]	Scene recognition	108,754	397	accuracy
	SVHN [69, 110]	Digit recognition	26032	10	accuracy
	iWildCam [7, 52]	Animal recognition	42,791	182	macro F1 score
	Camelyon17 [5, 52]	Metastatic tissue cls.	85,054	2	accuracy
	FMoW [19, 52]	Satellite imagery recognition	22,108	62	worst-region acc.
	Dollar Street [82]	Object recognition	3,503	58	worst-income top-5 acc.
	GeoDE [77]	Object recognition	12,488	40	worst-region acc.
Retrieval	Flickr30k [108]	Image and text retrieval	31,014	N/A	R@1
	WinoGAViL [8]	Commonsense association	3,563	N/A	Jaccard score

G.3 Our Benchmark Statistics

Table 11: Number of examples in TiC-RedCaps in each year.

Dataset	Year			
	2017	2018	2019	2020
TiC-RedCaps	4220262	1660003	2526575	3115715

Table 12: Number of examples in TiC-YFCC in each year.

Dataset	Year			
	2004–2008	2009–2010	2011–2012	2012–2014
TiC-YFCC	4337727	4050166	3976339	2312753

Table 13: Number of examples in TiC-DataComp in each year before filtering.

Dataset	Year								
	2014	2015	2016	2017	2018	2019	2020	2021	2022
TiC-DataComp (no filter)	244802598	175648045	666019511	1906357755	1877561875	2016011588	1778751066	2044463701	1442233121
TiC-DataComp (basic filter)	52764775	50757898	133333267	400225598	501347511	519575760	417067014	494038122	371748613

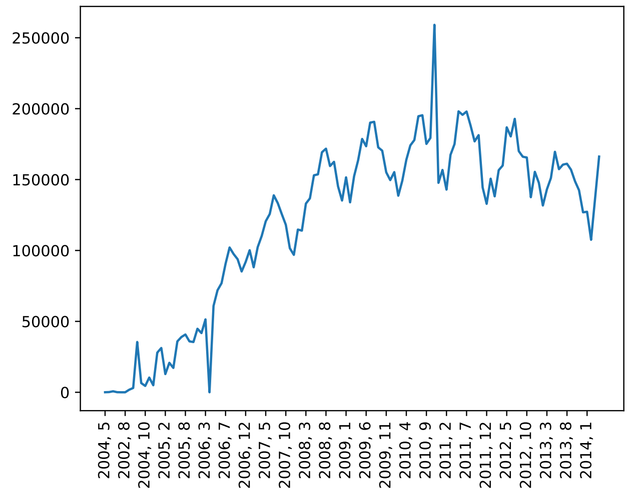


Figure 12: Number of examples in each year in original YFCC 15M. X-axis the upload month and y-axis is the number of examples in that month.

G.4 Compute Constraints for Different Datasets

We closely follow compute budget constraints from Gadre et al. [34]. In particular, on TIC-DataComp, we restrict to using exactly the same amount of overall compute as fixed in Gadre et al. [34]. Below we list exact total MACs on each dataset:

- TIC-YFCC: Total MACs: 3.4×10^{18}
- TIC-RedCaps: Total MACs: 3.4×10^{18}
- TIC-DataComp `medium`: Total MACs: 3.0×10^{18}
- TIC-DataComp `large`: Total MACs: 2.7×10^{19}
- TIC-DataComp `xlarge`: Total MACs: 2.7×10^{20}

For a ViT-B architecure, these values correspond to 20k iterations on TIC-YFCC (batch size: 8192), TIC-RedCaps (batch size: 8192), 35k iterations on TIC-DataComp (M) (batch size: 4096), 157k iterations on TIC-DataComp (L) (batch size: 8192), and 143.5k iterations on TIC-DataComp (XL) (batch size: 90100).

G.5 Creation Pipeline for Evaluation Datasets

We create our dynamic classification dataset TIC-DataComp-Net with ImageNet classes from the CommonPool data augmented with temporal information. Our construction process draws inspiration from the LAIONNet construction process described in Shirali and Hardt [88]. In particular, we first filter examples where the corresponding caption contains one and only one of the synsets of ImageNet. We also apply additional basic filtering [34] to make sure that images are atleast 200 in smalltest dimension and the caption contains atleast 2 words. After filtering for examples with ImageNet synsets, we only retain examples where the similarity—as evaluated by an off-the-shelf sentence embedding model [80]—between imagenet synset definition and the caption exceeds a threshold of 0.5. The goal of this filtering step is to restrict examples with 'high' alignment between caption and imagenet synset definition. Crucially, unlike LAIONNet, we do not filter the image-text pairs with CLIP similarity scores to avoid biasing the dataset selection process.

G.6 Distribution Shift Analysis on Proposed benchmarks

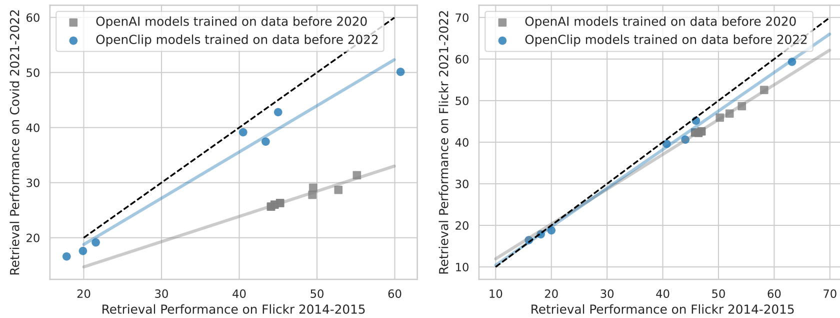


Figure 13: (Left) Comparison of retrieval performance on COVID queries versus Flickr queries. (Right) Comparison on old Flickr versus new Flickr data. Clearly, we observe that while gap on old versus new flickr data is small, the gap is significantly larger on Covid queries.

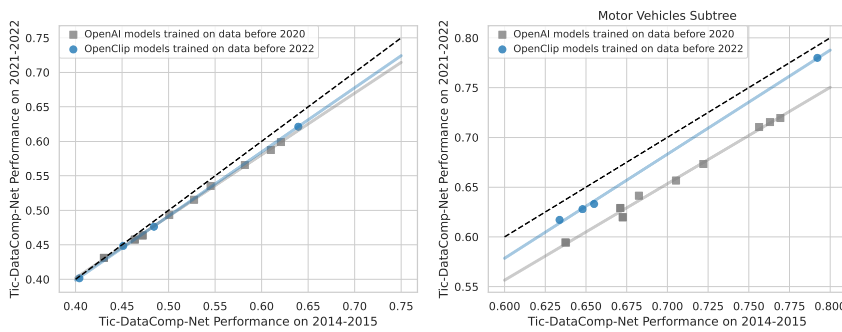
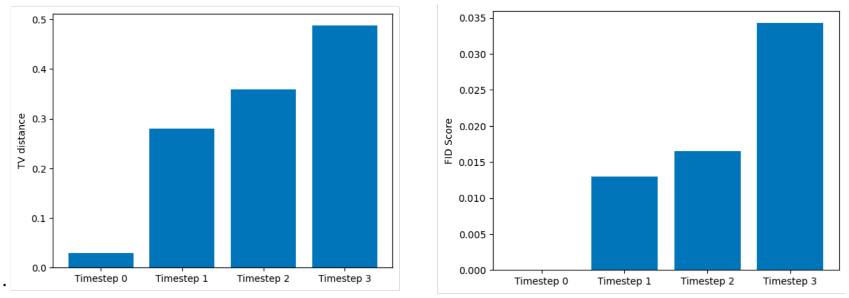


Figure 14: (Left) Comparison on old versus new data from TIC-DataComp-Net. (Right) Comparison on motor vehicles node from TIC-DataComp-Net. For our classification task, we observe a very small drop ($\approx 1\%$) when averaged across all categories. However, we observe a substantial gap on classes in "motor vehicle" subtree, when comparing OpenAI and OpenCLIP models. These findings highlight that while overall ImageNet classes may remain timeless, certain categories tend to evolve faster than others.

TIC-DataComp analysis through the lens of constructed evaluation tasks Here, we compare performance of OpenAI and OpenCLIP models on our datasets. We observe a significant performance gap between OpenAI and OpenCLIP models on our dynamic retrieval task (Fig. 1). This gap widens notably on retrieval queries where captions mention COVID-19. On the other hand, OpenAI and OpenCLIP models exhibit similar robustness for retrieval on data coming from Flickr highlighting that data from some domains do not exhibit shifts that cause performance drops. For our classification task, we observe a very small drop ($\approx 1\%$) when averaged across all categories. However, we observe a substantial gap on specific subtrees in ImageNet. For example, classes in "motor vehicle" subtree show an approximate 7% performance drop, when comparing OpenAI and OpenCLIP models. These findings highlight that while overall ImageNet classes may remain timeless, certain categories tend to evolve faster than others. Our qualitative and quantitative analysis on TIC-DataComp clearly highlights evolution of distributions and captures different properties than standard benchmarks.

Quantitative analysis on TIC-YFCC We analyze TIC-YFCC using off-the-shelf sentence and image encoders. We first embed images from different time steps with an OpenAI CLIP encoder and then compute Frechet Inception Distance (FID; Seitzer [87]). As time progresses, we observe that FID distance increases with respect to data from first time step (Fig. 15). Similarly, we use pretrained sentence transformer to extract top-5 categories from Wordnet Nouns for each caption. We observe that the TV distance over distribution of WordNet Nouns evolves over time when compared to data from the first time step.



Distance between categories from different timestep with respect to timestep 0

FID distance on CLIP embeddings with respect to timestep 0

Figure 15: YFCC15M distribution shift results. Analyze on TIC-YFCC using off-the-shelf sentence and image encoders. We first embed images from different time steps with an OpenAI CLIP encoder and then compute Frechet Inception Distance (FID; Seitzer [87]). As time progresses, we observe that FID distance increases with respect to data from first time step. Similarly TV distance over categorical distribution on Wordnet Noun synsets also increases with time when compared to categorical distribution on first timestep.

G.7 Creation Pipiline for TIC-DataComp

We collect timestamps for the CommonPool dataset introduced in DataComp. We repeat the crawling process described in Gadre et al. [34] to download WARC files from Common Crawl. After downloading the WARC files, we perform a join with the datacomp 12.8B examples. During this join, we lost approximately 0.1B of examples that are no longer available online. Moreover, while performing this join, we only retain examples with their first occurance. This is done before running any de duplication on image-text pairs for exact matches as done in Gadre et al. [34].

The source of DataComp is Common Crawl, which periodically releases web-crawled data snapshots, typically on a monthly basis since 2014 with new and updated webpages. This process provides timestamps at the granularity of months, spanning years 2014–2022.

We note that while this augmented time information may contain some noise, on average, we find it to be a reasonably accurate proxy for the upload time of web pages. To perform an initial check, we note that our data contains images from flickr which provides an API to query for true upload timestamp. So we extract 10k examples from our benchmark TIC-DataComp and query Flickr for their true timestamp. Fig. 16 summarizes true timestamps with timestamps extracted from CC.

H Additional Experimental Details

H.1 Additional details on ImageNet IID split continual learning experiment

With ImageNet data, we consider 2, 4 and 8 splits including the full dataset. This design is inspired by Ash and Adams [3]. We consider ViT-B/16 architecture trained for 300 epochs on full data and split the iterations corresponding to 300 epochs equally among k splits when training on k splits sequentially. We keep all other hyperparameters, such as learning rate, optimizer, and batch size, set to the standard values typically employed for training ViT-B/16 on the ImageNet dataset [29]. We also employ ℓ_2 regularization and augmentation on ImageNet training data. We evaluate the models on IID ImageNet test set.

H.2 Training and Hyperparameter Details

We create a common experimental setup by fixing the training procedure for sequential runs. Unless specified otherwise, we closely follow the CLIP training recipe proposed in [44, 76] where we train models with a contrastive objective over images and captions. Given a set of image-text pairs, we train an image encoder and a text encoder such that the similarity between the representations of images and their corresponding text is maximized relative to unaligned pairs. Only LwF deviates from this standard training procedure. For each benchmark, we pick Vision Transformers (ViTs) as

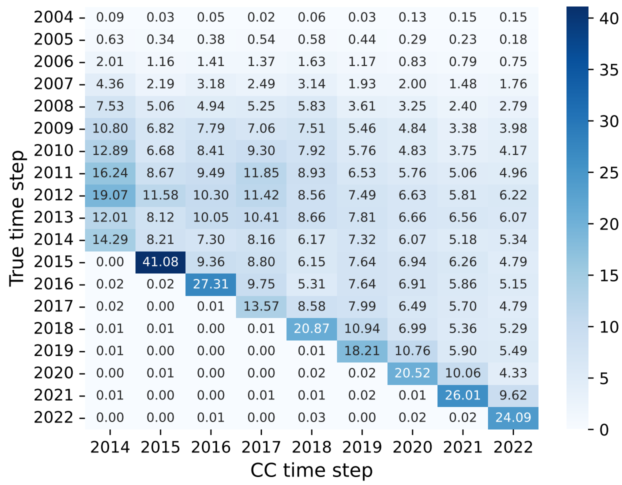


Figure 16: Comparison of Common Crawl assigned timestamp and true timestamp on a subset of 10k examples containing image-text pairs from Flickr. We observe a clear trend where CC timestamps correlate with true timestamps.

the image encoder, in particular, we fix the model architecture to ViT-B/16 [30]. We fix the Adam optimizer and its hyperparameters to values suggested in [44].

We primarily ablate over only two things: maximum learning rate with cosine learning schedule and warm up iterations for sequential training. For choosing other hyperparameters, we follow the OpenCLIP library [44].

H.3 Replay sizes with Exp and Equal strategies

We default to using 2D size of data where D represents incoming data size from new time step. As described in the main text, for -Exp, we reduce the buffer size by half of what we used at old time step and use rest of the half as data from previous time step. App. G.3 lists the dataset sizes for each benchmark which dictate the exact buffer sizes.