

Scalable Continual Deep Learning With Computational Cost Considerations



Ameya Pandurang Prabhu
Linacre College
University of Oxford

This dissertation is submitted for the degree of

Doctor of Philosophy

February 2025

To being kinder to ideas I disagree with, there is much at stake!

Declaration

I hereby declare that this thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work based on publications in collaboration with the co-authors where due acknowledgement is made.

Ameya Pandurang Prabhu
Linacre College

Acknowledgements

I first thank Philip Torr for taking a chance on me and being my shield, an extremely underrated aspect. For two years, from October '20 to '22, I admittedly had little research output, largely attributable to me. I would have been fired long ago if not for his shielding me from the pressure of funders. The fact that I am writing this thesis now is due to his belief in me and ensuing patience.

While the writing and procedural aspects of this thesis was an exercise in sustained suffering, in contrast, the journey itself was one of the most memorable. I am especially thankful to Arnab Ghosh for his PhD advice, which I am extremely grateful to have followed. I fondly recall being my quirky self with Tom Joy and Andrew Gambardella, the encouragement and support from Francesco Pinto and Will Lin that helped me get through difficult times, the memorable moments of research inspiration from Shuyang Sun, Fabio Pizzati, Alasdair Paren, Ashkan Khakzar, Christian Schroeder de Witt, and the long nights spent in fervent discussions with Kalyan R., Tuan Nguyen, Bruno Andreis. The good times spent with fellow PhD candidates around me were my greatest treasure over these years—thank you to Aleksander Petrov, Botos Csaba, Jishnu Mukhoti, Francisco Eiras, Runjia Li and Joanna Materzynska.

I love research, but working with seriously cool people made the journey so much more memorable! I would like to express my gratitude to Hasan Hammoud, my greatest influence, whose infectious energy has transformed me. I look back with fondness on the challenging problems we tackled together and kicked ass. Hasan has been an inspiration for me in many ways, his superhumanness ranging from project execution to his dedication to staying swole. Same applies for Shashwat Goel. Thank you Shashwat and Hasan, who spent several hundreds to thousands of hours on long videocalls with me, their influence which still feels under-rated despite this lengthy paragraph. I am very grateful and thank my supervisors on PhD projects – Puneet Dokania, Ozan Sener, Vladlen Koltun and my advisors Adel Bibi and Philip Torr, who significantly shaped me and instilled their understanding and way of approaching problems. I am proud of every paper we wrote.

Controversially, I don't really regret my underproductivity. In Oxford, I evolved from a shy CS nerd to a better persona with cool hobbies and dreams outside computers. My biggest influences outside research was my best friend Lewis Hart, without whom I cannot imagine my PhD years, and all the crazy times, through highest highs and lowest lows, we spent. I will dearly miss my housemates, notably Franziska Schantz, Tommaso De Zan and Pink Kulthanan for the hikes,

extensive nerdy gadget-mania and all the chocolate-shake walks during covid days. I cherish the people in the EA community; they are some of the kindest and coolest individuals I've met — Peter Wallich, Tushita Jha, and Mathieu Putz were particularly influential on my growth, and I can't thank them enough. I thank some exceptionally cool people: Asher Soryl, Bella Forristal, Noah Wescombe, Aimee Watts, Mrinank Sharma, Isaac Dunn, Kajuli Claeys for their significant influence on me. I thank my party people — Shehryar Saroya, Alice Evatt, Alexi Siddiqui, Stephanie Sheir — for the best times and good vibes, the vagueness being necessary ;) I am still best described as a nerd, and I must thank the best fellow nerds I've met: Elle Yang, Mitch Beard, Bianca Bauer, Sophia Ungermann, Carl Mohr, Holly Russell. Special thanks to Teddy Cai, my greatest influence from October '22 to '23, who instilled in me the passion for climbing and with whom I've shared some of my best times. Lastly, I acknowledge those who truly made me a better person — Sara, Phoebe, and Jean — and I wish them much success on their journey ahead. Lastly, I cannot thank my parents enough for their unwavering support. My parents were strong pillars standing behind me, being my safety net and giving me the courage to pursue my wildest aspirations.

I have sacrificed everything for my love of research, with hard life choices like moving far away from my family and long-time friends several times, wondering whether it was worth it all. I am less disillusioned now as this has sunk-in — the research system is not my friend, but it is not my enemy either. It is a pseudo-random entity that sometimes grants acceptance or rejection to my ideas, as it does to everyone else. Many smart folks have tried to improve it — I am no smarter. However, I believe that kindness is key. Kindness, not only to those around me but also to ideas I disagree with, for there is much at stake! I hope to always remember all the kindness you folks showed to me and my ideas along the way. I am about to sacrifice my wonderful Oxford life again for research, moving to Germany, but I shall dearly miss all of you — thank you for the precious memories — to the many friends and colleagues I've met in Oxford.

Abstract

Continual Learning (CL) is an emerging field that focuses on developing models capable of learning continuously from an incoming stream of data, as opposed to hundreds of passes over static, curated datasets. These models aim to retain previously acquired knowledge while seamlessly integrating new information, often with constraints like limited storage capacities. To advance this field, we first pinpoint the current research paradigm’s limitations. We address these by (i) implementing more realistic constraints, such as optimizing learning within limited computational resources, and (ii) demonstrating the effectiveness of simple, straightforward algorithms. Additionally, we address flaws in existing metrics and enhance data collection methods to improve the efficiency and applicability of continual models in practical scenarios. We detail our contributions below:

1. General Formulation & Concerns on Relevance of the Online Stream. We pose a general formulation of the CL problem for classification, inspired by the open-set recognition problem, which requires the model to adapt continually to new classes from a stream of data. We detail how some assumptions in continual learning potentially oversimplify the problem, thus reducing their practical relevance. Our GDumb model demonstrates that even simple ablations like not using the online stream can match or outperform continual learning algorithms designed for specific continual learning scenarios, challenging the progress in this field.

2. Challenges in Continual Representation Learning. Our RanDumb model explores continual representation learning by using a fixed random transform and a simple linear classifier, revealing that using no representation learning outperforms continual learned representations using deep networks across a variety of standard continual learning benchmarks. This finding suggests a need for a principled re-evaluation of how we design and train models for effective continual learning.

3. Introducing Computational Constraints. Current literature often overlooks the practical constraints of computational and time budgets, focusing instead on unrestricted access to data. We develop large-scale datasets like ImageNet2K and Continual Google Landmarks V2, and extensively evaluate continual learning under compute-constrained settings, and show that traditional CL methods, even with strategic sampling and distillation, cannot outperform even the simple experience replay baselines. This finding indicates that existing CL approaches are too compute-intensive for real-world applications.

4. Online Continual Learning without Storage Constraints. We propose a simple algorithm that updates a kNN classifier continually with a fixed, pretrained feature extractor, suitable for scenarios with rapid data stream changes and minimal computational budgets. This method significantly outperforms existing methods in accuracy while maintaining low computational and storage requirements.

5. Rapid Adaptation Metrics in Online Continual Learning. We critique the traditional metrics used in online continual learning. Our findings suggest that existing methods might not genuinely adapt but rather memorize irrelevant data patterns. We alleviate this by proposing a new metric focused on 'near-future' sample accuracy to avoid learning spurious label correlations and better reflects the ability of a model to genuinely adapt to new, incoming data.

6. Leveraging Webly-Supervised Data. Tackling the impracticality of continually sourcing large, annotated datasets, we introduce a novel continual data-acquisition paradigm that allows models to adapt to new categories using only category names. We propose a simple method which uses uncurated webly-supervised data, and not only reduces reliance on costly manual annotation but also demonstrates the potential of internet-sourced data to support effective learning. This is evidenced by our creation of the EvoTrends dataset, which reflects real-world trends with minimal cost budget.

Overall, this thesis lays the foundations and advocates for a shift towards more computationally efficient continual learning methods that are better suited for real-world applications.

Contents

List of Figures	vi
List of Tables	xiv
1 Introduction	1
1.1 A Very Brief Introduction to Continual Learning	2
1.2 Why Computational Cost Considerations?	2
1.3 Thesis Outline and Contributions	4
2 GDumb: A Simple Approach that Questions Our Progress in Continual Learning	9
2.1 Introduction	11
2.2 Problem Formulation, Assumptions, and Trends	11
2.2.1 Simplifying Assumptions in Continual Learning	12
2.2.2 Recent Trends in Continual Learning	15
2.3 Greedy Sampler and Dumb Learner (GDumb)	16
2.4 Experiments	19
2.4.1 Results and Discussions	20
2.4.2 Resources Needed	24
2.4.3 Potential Future Extensions	24
2.5 Conclusion	24
3 Randumb: A Simple Approach that Questions the Efficacy of Continual Representation Learning	25
3.1 Introduction	27
3.1.1 Randumb: Construction and Performance	28
3.2 Randumb: Mechanism & Intuitions	30
3.3 Experiments	31
3.3.1 Results	35
3.3.2 Randumb: Analysis	38
3.4 Discussion and Concluding Remarks	40

4	Computationally Budgeted Continual Learning: What Does Matter?	41
4.1	Introduction	43
4.2	Continual Learning with Limited Compute	44
4.2.1	Problem Formulation	44
4.2.2	Key Differences with Prior Art	44
4.2.3	Constructing the Stream	46
4.3	Dissecting Continual Learning Systems	46
4.4	Experiments	49
4.4.1	Experimental Setup and Details	49
4.4.2	Budgeted Continual Learning	52
4.4.3	Sensitivity Analysis	55
4.4.4	Exploring Partial Training	57
4.5	Conclusion	58
5	Online Continual Learning Without the Storage Constraint	59
5.1	Introduction	61
5.2	Related Work	62
5.3	Our Approach: Adaptive Continual Memory	64
5.3.1	Computational Cost and Storage Considerations	66
5.4	Experiments	66
5.4.1	Comparison of ACM with Online Continual learning Approaches	68
5.4.2	Comparison of ACM with Approaches Leveraging a Fixed Backbone	70
5.4.3	Analyzing Our Method: Adaptive Continual Memory	71
5.4.4	Discussion	72
5.5	Conclusion	73
6	Rapid Adaptation in Online Continual Learning: Are We Evaluating It Right	74
6.1	Introduction	76
6.2	OCL Evaluation: A Review	77
6.3	Correcting Evaluations of Rapid Adaptation	80
6.3.1	Evaluating OCL: A Problem Formulation	81
6.3.2	Isolating and Quantifying Label Correlations	81
6.3.3	Can Deep Networks Learn Label Correlations?	82
6.3.4	Exploring Properties of OverAdapt	83
6.3.5	Removing Correlations to Evaluate Adaptation	86
6.4	OCL Evaluation by Near-Future Accuracy	87
6.4.1	Experimental Setup	88

6.4.2	Results: Varying Stream Speeds	89
6.4.3	Further Discussion	90
6.5	Conclusion	91
7	From Categories to Classifier: Name-Only Continual Learning by Exploring the Web	92
7.1	Introduction	94
7.2	Name-Only Continual Learning: Problem Formulation	95
7.3	Our Approach: Categories to Classifier By Exploring the Web	96
7.4	Evaluating Capabilities of our Approach	99
7.4.1	Experimental Details	99
7.4.2	How Reliable is our Proposed Approach?	100
7.4.3	Comparison with Name-Only Approaches	101
7.5	Continual Webly-Supervised Learning	103
7.5.1	Experimental Details	104
7.5.2	Results	105
7.5.3	EvoTrends: The First Continual Name-Only Classification Benchmark	107
7.6	Conclusion	108
8	Conclusion	109
8.1	Summary of Contributions	110
8.2	The Road Ahead and Open Problems	112
8.3	Concluding Remarks	113
	Appendices	114
A	GDumb: A Simple Approach that Questions Our Progress in Continual Learning	115
A.1	Additional Experiments	116
A.2	Experiment formulations	117
B	Computationally Budgeted Continual Learning: What Does Matter?	120
B.1	Motivation: Cost and Privacy Reasons for Restricting Memory are Flawed	121
B.2	Dataset Construction	122
B.2.1	Constructing Imagenet2K	122
B.2.2	Constructing Continual Google Landmarks V2	123
B.3	Estimation of Equivalent Iterations	123
B.4	Additional Results	124
B.4.1	Effect of Weight Decay	131
B.4.2	Effect of Batch Size	131
B.4.3	Effect of Increasing Computational Budget on Distillation	131

C	Online Continual Learning Without the Storage Constraint	133
C.1	Implementation Details	134
C.1.1	Dataset Details	134
C.1.2	Model and Optimization Details	135
C.2	Evaluating Rapid Adaptation Using Near-Future Accuracy	136
C.3	Additional Experiments	137
D	Rapid Adaptation in Online Continual Learning: Are We Evaluating It Right?	139
D.1	Effect of Sampling Strategies	140
D.1.1	CGLM	140
D.1.2	CLOC	140
D.2	Sensitivity Analysis: Learning Rate and Weight Decay	141
D.2.1	Sensitivity to Weight Decay	142
D.2.2	Sensitivity to Learning Rate	142
D.3	Fixed-Feature Extractor Based Methods: NCM, SDLA, and ACM	142
D.4	Limitations & Future Directions	143
E	From Categories to Classifier: Name-Only Continual Learning by Exploring the Web	145
E.1	Connections to Past Literature in Webly Supervised Learning	146
E.2	Appendix: How Reliable is Uncurated Webly-supervised Data?	148
E.2.1	Analysis: Why Does Uncurated Web Data Outperform Manually Annotated Datasets?	148
E.3	Appendix: Comparison with Name-Only Classification Strategies	152
E.4	Appendix: Continual Webly Supervised Learning	155
E.4.1	EvoTrends	155
E.4.2	Computational Budgets	155
E.4.3	Investigating Performance Gaps Between Our Webly-Supervised Data and MA Data	158
E.5	Ethics	162
F	Publications	163
F.1	GDumb: A Simple Approach that Questions Our Progress in Continual Learning (ECCV 2020, Oral)	163
F.2	RanDumb: A Simple Approach that Questions the Efficacy of Continual Represen- tation Learning (Under Review)	163
F.3	Computationally Budgeted Continual Learning: What Does Matter? (CVPR 2023)	164
F.4	Online Continual Learning Without the Storage Constraint (Arxiv)	164

F.5	Rapid Adaptation in Online Continual Learning: Are We Evaluating It Right (ICCV 2023)	165
F.6	From Categories to Classifier: Name-Only Continual Learning by Exploring the Web (Under Review)	165
Bibliography		166

List of Figures

2.2.1 Our approach (GDumb): The sampler greedily stores samples while balancing the classes. When asked, the learner trains a network from scratch on memory $\mathcal{D}_t$ provided by the sampler. If a mask $\mathbf{m}$ is given at inference, GDumb classifies on the subset of labels provided by the mask. Depending on the mask, GDumb’s inference can vary between two extremes: CI (class-incremental) and TI (task-incremental) formulations.	17
3.1.1 Mechanism of Randumb. Project raw pixels to a high dimensional space using random fourier projections (φ), then decorrelate the features using Mahalanobis distance McLachlan [1999] and classify with the nearest class mean. The online update only involves updating a single covariance matrix and class-means.	27
3.2.1 Mechanism of Randumb. Randumb projects the input datapoints to a high-dimensional space to create a clearer separation between classes. Subsequently, it corrects the anisotropy across different feature dimensions, scaling them to be unit variance each. This allows a nearest class mean classifier to accurately rely on cosine similarity. The figure is adapted from Pilario et al. [2020].	33
3.3.1 Impact of Embedding Dimensions. We show the variation of accuracy of Randumb with embedding dimensionality across datasets. We see decreasing marginal returns with increasing dimensions, obtaining relatively minor improvements after 15K dimensions.	39
4.1.1 Main Findings. Under per time step computationally budgeted continual learning, classical continual learning methods, <i>e.g.</i> , sampling strategies, distillation losses, and fully connected (FC) layer correction based methods such as calibration, struggle to cope with such a setting. Most proposed continual algorithms are particularly useful only when large computation is available, where, otherwise, minimalistic algorithms (ERM) are superior.	43

4.4.1 Effective Training Epochs Per Time Step. Our default setting sets a total training budget over all 20 time steps of 8000 and 2000 iterations for ImageNet2K and CGLM ,respectively, with a per iteration batch size of $\mathcal{B} = 1500$. Effectively, this reflects to training on 25-50% of the stored data, except in the first few time steps on CGLM. Note that for ImageNet2K, we assume that ImageNet1K of 1.2M samples is available in memory.	50
4.4.2 DI-ImageNet2K (400 Iterations). ERM-Naive, a non-continual learning algorithm, is compared against inexpensive sampling strategies (first four plots) with 400 training iterations and the costly KMeans (fifth plot) with 200 iterations. All CL methods perform similarly but worse than ERM-Naive. This is the case for FIFO that suffers from forgetting the KMeans due to its expensive nature. ImageNet2K experiments performance can be decomposed into (i) accuracy on classes seen during pre-training on ImageNet1K and (ii) accuracy on newly seen classes in ImageNet2K, allowing analysis of forgetting old classes and learning newly introduced classes.	50
4.4.3 CI-ImageNet2K (400 Iterations). Similarly, ERM-Naive is compared on CI-ImageNet2K. Both FIFO, which suffers from forgetting, and KMeans due to its expensive nature, struggle to compete against simpler inexpensive methods as Class-Balanced. However, overall, all other methods perform very similarly with no clear advantage. ImageNet2K experiments performance can be decomposed into (i) accuracy on classes seen during pre-training on ImageNet1K and (ii) accuracy on newly seen classes in ImageNet2K, allowing analysis of forgetting old classes and learning newly introduced classes.	50
4.4.4 CGLM (100 Iterations). All inexpensive methods perform overall similarly with the exception for KMeans due to its expensive nature. This highlights that simplicity is key under a budgeted continual learning setting. CGLM is not an extension of ImageNet1K and involves a different task: landmark classification. Hence, we measure only the stream accuracy resulting in two lines instead of six.	51
4.4.5 Distillation in Data and Class Incremental Settings. Naive, which does not employ any distillation loss, outperforms all distillation methods (MSE, and Cosine) across all three settings.	51
4.4.6 FC Layer Correction. Even though loss functions (CosineFC and ACE) might outperform Naive in the first few time steps, eventually Naive catches up. Overall, Naive consistently outperforms all considered calibration methods too, namely, BIC and WA.	53
4.4.7 CGLM Distillation with Different Number of Time Steps. Under larger number of time steps, where total number of iterations is normalized accordingly, Naive outperforms distillation in both settings, namely, 50 and 200 time steps.	54

4.4.8	CGLM Distillation with Different Computational Budgets. Naive outperforms distillation methods under the restricted 40 and the larger 400 iterations (originally 100). Distillation methods become competitive when enough budget is available.	55
4.4.9	Linear vs Full Model Training. Performing Linear fine tuning allows to leveraging the computational budget efficiently improving the gap compared to full model training particularly for better pretrained models, <i>e.g.</i> , Instagram1B+ImNet1K.	57
5.3.1	Adaptive Continual Memory (ACM) performs Memory.Retrieve and Memory.Insert on new incoming samples, extracted by a fixed, pretrained deep network.	64
5.4.1	Online Continual Learning Performance. We observe that ACM outperforms existing methods by a large margin despite being far cheaper computationally. Traditional methods perform poorly given unrestricted access to past seen data indicating continual learning is a hard problem even without storage constraints.	69
5.4.2	Online Continual Learning Performance. ACM outperforms existing methods which leverage a fixed backbone. This highlights the importance of preserving the consistency property in online continual learning. The collapse of other approaches for the CLOC dataset indicates the hardness of the continual learning scenario.	70
5.4.3	Left: Wall clock time overhead of using ACM Memory after feature extraction (x-axis is log-scaled) on a 16-core i7 CPU server. The time increases logarithmically with dataset size, with a 8ms overhead at 40M samples. Right: Contribution of kNN and contribution of the original backbone along with the MLP. We observe most of the performance is attributable to the kNN.	72
6.1.1	Effect of Spurious Correlations. Current methods in OCL perform well on rapid adaptation but suffer from abysmal information retention (top). We show that these trends are due to idiosyncrasies in the data stream such as label correlations that favor methods that inadvertently overfit to the latest data. Using our proposed metric that removes these correlations (bottom), different sets of methods achieve good performance, with high information retention being key for achieving high adaptability.	76
6.2.1	Blind Classifier. Performance of a blind classifier on the CGLM (left) and CLOC (right) datasets across varying shift S towards future for selecting evaluation samples alongside the optimal context window size k for the selected shift. (Section 6.3.2) A blind classifier achieves unrealistically high accuracy on current evaluation of adaptation (dark blue line, $S=0$) on both datasets despite being a trivial baseline. (Section 6.3.5) The accuracy of the blind classifier on incoming samples with increasing shift S into the future decreases significantly, eventually converging to near-random performance, indicating a lack of label correlations to the current training samples.	80

6.3.1 OverAdapt Classifier. We compare the adaptation performance of our OverAdapt classifier on CGLM (left) and CLOC (right) datasets using online accuracy as a measure. The three top performing OCL methods on CLOC are benchmarked on the CGLM dataset. We see that OverAdapt outperforms recent OCL methods by a large margin of 10-40% in accuracy.	82
6.3.2 Sensitivity Analysis of OverAdapt Classifier. We illustrate the performance of various components of our OverAdapt classifier on CGLM (top) and CLOC (bottom) datasets measuring: (Section 6.3.4) information retention (left) and rapid adaptation capability in terms of online accuracy ($\mathcal{S} = 0$, center) (Section 6.3.5) rapid adaptation performance in terms of online accuracy ($\mathcal{S} = 0$, center) with our proposed near-future accuracy ($\mathcal{S} = 256$ in CGLM and , $\mathcal{S} = 16384$ in CLOC). Note that information retention (left) is independent of the shift $\mathcal{S}$. The results demonstrate that introducing the shift removes label correlations, leading to dramatic loss in adaptation performance for our OverAdapt method while the uniform baseline stays unaffected.	84
6.3.3 Our Evaluation Method. We compare the samples selected for evaluating adaptation performance by our evaluation method. The blue and green arrows indicate the latest batch received from the stream for learning and the samples selected for evaluation respectively. Current evaluation proposes to compute accuracy on the immediate incoming samples ($\mathcal{S} = 0$), <i>i.e.</i> online accuracy, while we propose to evaluate on the closest future samples ($\mathcal{S} = 64$), <i>i.e.</i> near-future accuracy, which is free from spurious correlations with the seen batch of images.	86
6.4.1 Performance of Methods on Near-Future Accuracy. We evaluate CL methods in slow and fast streams similar to [Ghunaim et al., 2023]. Near-future accuracy, which eliminates spurious label correlations, shows that retaining and reusing past information is crucial for better generalization and that algorithms not designed for adaptation in OCL can perform well. Hence retention and adaptation are closely linked.	89
7.3.1 Continual Name-Only Classification: Our Approach. At each timestep t , the learner receives a list of class categories without any training samples. We start by collecting webly-supervised data through querying and downloading data from multiple search engines. We then extract features using a frozen backbone, and subsequently train a linear layer on those features. The same process is repeated for the next timestep.	98

B.4.1 Expensive Sampling. As mentioned in the manuscript, KMeans performs the best among expensive sampling techniques such as Max Loss and Uncertainty Loss. Nevertheless, the performance of the expensive sampling methods is worse than simple Naive.	125
B.4.2 Distillation Methods. All four studied distillation methods under perform compared to the simple Naive baseline in all three studied settings. ImageNet experiments are allowed 400 iterations whereas CGLM is allowed 100 iterations.	126
B.4.3 DI-ImageNet2K 50 Time Steps. As observed in the manuscript, when the number of time steps, distillation methods still under perform compared to the Naive baseline.	126
B.4.4 DI-ImageNet2K 200 Time Steps. As observed in the manuscript, when the number of time steps, distillation methods still under perform compared to the Naive baseline.	127
B.4.5 CI-ImageNet2K 50 Time Steps. As observed in the manuscript, when the number of time steps, distillation methods still under perform compared to the Naive baseline.	127
B.4.6 CI-ImageNet2K 200 Time Steps. As observed in the manuscript, when the number of time steps, distillation methods still under perform compared to the Naive baseline.	127
B.4.7 CLGM 50 Time Steps. As observed in the manuscript, when the number of time steps, distillation methods still under perform compared to the Naive baseline. . . .	128
B.4.8 CGLM 200 Time Steps. As observed in the manuscript, when the number of time steps, distillation methods still under perform compared to the Naive baseline. . . .	128
B.4.9 DI-ImageNet2K - 100 Iterations. As observed in the manuscript, with reduced compute, distillation methods still under perform compared to the Naive baseline. The compute budget of the Naive baseline, C , is set to 100 iterations whereas that of the distillation methods is $2/3 C = 67$ iterations.	128
B.4.10 DI-ImageNet2K - 1200 Iterations. As observed in the manuscript, with increased compute, distillation methods still under perform compared to the Naive baseline. The compute budget of the Naive baseline, C , is set to 1200 iterations whereas that of the distillation methods is $2/3 C = 800$ iterations.	129
B.4.11 CI-ImageNet2K - 100 Iterations. As observed in the manuscript, with reduced compute, distillation methods still under perform compared to the Naive baseline. The compute budget of the Naive baseline, C , is set to 100 iterations whereas that of the distillation methods is $2/3 C = 67$ iterations.	129
B.4.12 CI-ImageNet2K - 1200 Iterations. As observed in the manuscript, with increased compute, distillation methods still under perform compared to the Naive baseline. The compute budget of the Naive baseline, C , is set to 1200 iterations whereas that of the distillation methods is $2/3 C = 800$ iterations.	129

B.4.1	CGLM 40 - Iterations. As observed in the manuscript, with reduced compute, distillation methods still under perform compared to the Naive baseline. The compute budget of the Naive baseline, C , is set to 40 iterations whereas that of the distillation methods is $2/3 C = 27$ iterations.	130
B.4.1	CGLM 400 - Iterations. As observed in the manuscript, with increased compute, distillation methods still under perform compared to the Naive baseline. The compute budget of the Naive baseline, C , is set to 400 iterations whereas that of the distillation methods is $2/3 C = 267$ iterations.	130
B.4.1	Effect of Weight Decay. Increasing the weight decay causes a slight drop in performance. Setting $wd=0$ gave the best results during our parameter cross-validation.	131
B.4.1	Effect of Batch Size. The conclusions presented in our work hold even when the batch size while is changed while the same overall computational budget.	132
B.4.1	Effect of Increasing Computation Budget on Distillation. As the computation budget is increased while maintaining a normalized compute among different methods, Naive baseline still outperforms distillation based methods.	132
C.2.1	Online Continual Learning Performance. We observe that ACM outperforms existing methods on the near-future accuracy metric by 5-10%, maintaining state-of-the-art performance.	136
C.3.1	Left: Ablating the effect of first-session adaptation using an MLP. We observed first-session adaptation caused minor improvement in performance on XCIT model with 512 dimensional features, however provided a major performance boost for 2048 dimensional ResNet50 features. Right: Ablating the effect of embedding dimensions. We observe that varying the embedding size allows a tradeoff between computational cost and accuracy. However, it does not cause dramatic increase or decrease in performance.	137
D.1.1	Effect of Sampling Strategy on Online Accuracy. In terms of online accuracy, FIFO sampling which focuses on the latest samples performs best where uniform performs the worst and mixed performs somewhere in the middle. In terms of retention however, the order is reversed.	140
D.1.2	Effect of Sampling Strategy on Near-Future Accuracy. In terms of near-future accuracy, Uniform and Mixed sampling perform almost the same, however FIFO sampling is no where close to them. In terms of retention, Uniform still takes the lead.	141
D.1.3	Effect of Sampling Strategy on Online Accuracy. Unlike what was observed for CLGM, for CLOC, the mixed sampling performs the best in terms of online accuracy followed followed by FIFO and then Uniform. However, Uniform still performs the best in terms of information retention with a large gap with a large margin.	141

D.1.4	Effect of Sampling Strategy on Near-Future Accuracy In terms of near-future accuracy, mixed outperforms other samplers by a significant margin. However, it achieves significantly worse performance compared to uniform sampling in terms of information retention.	142
D.2.1	Sensitivity to Weight Decay on ER (Replay Only). Weight decay seems to have minimal effect on both near-future accuracy and backward transfer.	143
D.2.2	Effect of Learning Rate on the ER Baseline on CGLM. Both near future-accuracy and backward transfer are highly sensitive to the selection of the learning rate. . .	143
D.3.1	Fixed-Feature Extractor Based Methods. When compared to NCM and SLDA, ACM performs the best in terms of Backward Transfer, Online Accuracy and Near-Future Accuracy metrics.	144
E.4.1	EvoTrends: A Dynamic Dataset Reflecting Real-world Trends. This illustration showcases the dataset we have curated using internet sources. EvoTrends consists of 21 timesteps, spanning the years 2000 to 2020. Each timestep presents the most trending products of the respective year, challenging the learner to adapt to these evolving trends. Unlike artificial scenarios, this dataset accurately reflects a real class-incremental setting, where classes emerge based on actual trends observed in the world.	155
E.4.2	Effective Training Epochs Per Time Step. Each row represents one of the computational budgets: tight, normal, and relaxed in sequential order. The normal budget is carefully chosen to allow the manually annotated datasets to undergo one epoch of training during the initial timestep (depicted by the green plot in the left side figures). The tight budget (blue) is half the budget of normal, while the relaxed budget (yellow) is four times the budget of normal. As the webly-supervised data surpasses the manually curated datasets in size, they undergo fewer epochs within each of the three budget regimes. At each timestep more data is presented, hence the effective number of epochs decays with time.	157
E.4.3	Domain Gap in CLEAR10 Dataset. In the CLEAR10 dataset, the original paper describes buses as the "exterior" of the bus, while the test set predominantly consists of images showcasing the "interior" of the bus. In contrast, our web-collected data primarily comprises "exterior" photos of the buses. Considering the inherent dissimilarity between our buses and the test set, this justifies the 10% performance gap observed when using our webly-supervised data compared to manually annotated datasets. It is important to note that this discrepancy does not apply to other classes within CLEAR10, as demonstrated by the camera class for reference.	159

E.4.4	Comparison of PACS and Webly-Supervised Data. The upper two rows depict the sketch and painting domains of the manually annotated PACS dataset, while the last two rows showcase the sketch and painting domains of our webly-supervised data. Although there is some resemblance between the painting domains of both datasets, a significant domain gap becomes apparent when comparing the sketches. In the PACS dataset, sketches refer to quick drawings, whereas in our web search, sketches are composed of line drawings and detailed sketches.	160
E.4.5	Comparison of CIFAR100 and Webly-Supervised Data. The upper row displays 32×32 samples obtained from CIFAR100, while the second row showcases images collected through our webly-supervised approach. Evidently, a substantial domain gap exists between the CIFAR100 images, characterized by their low quality, centered composition, and outdated nature, and the samples we collect, which are recent, not necessarily centered, and of significantly higher image quality. To bridge this gap, we employ downsampling techniques to resize our images to 32×32 . It is noteworthy that through this downsampling process, we achieve an improvement in the performance of our webly-supervised approach on CIFAR100 with a normal budget, boosting performance by approximately 8% from 30.3% to 38.7%.	161

List of Tables

1.2.1 The cost of storing data has decreased rapidly, allowing the storage of a large dataset for a negligible cost compared to the cost of computation for both cloud and mobile systems.	3
2.2.1 Here we categorize various recently proposed CL approaches depending on the underlying simplifying assumptions they impose.	14
2.3.1 Various CL formulations we considered in this chapter to evaluate GDumb. These formulations differ in terms of simplifying assumptions (refer Table 2.2.1) and also in terms of resources used. We ensure that selected benchmarks are diverse, covering all popular categorizations. Note, in B3 and D, memory is not constant– it increases over tasks uniformly by (+size) for xtasks times.	18
2.4.1 (CI-Online-Disjoint) Performance on formulation A1.	19
2.4.2 (CI-Online-Disjoint) Performance on formulations A2 (left) and A3 (right).	21
2.4.3 (CI-Offline-Disjoint) Performance on B1, B2, and B3.	22
2.4.4 (TI-Offline-Disjoint) Performance on C1 (left) and C2 (middle). (TI-Online-Disjoint) Performance on D (right).	22
2.4.5 (CI-Online-Joint) Performance on E (left). Note, this is particularly challenging as the tasks here are non-disjoint (blurry task boundary) with class-imbalance. On the (right), we benchmark resource consumption in terms of training time and memory usage. Memory cost is provided in terms of the total parameters P , the size of the minibatch B , the total size of the network hidden state H (assuming all methods use the same architecture), the size of the episodic memory M per task. GDumb, at the very least, is 7.5x times faster than the existing efficient CL formulations.	23
3.1.1 Online Continual Learning. Performance comparison of RanDumb on the PEC setup [Zajac et al., 2024] and VAE-GC [Van De Ven et al., 2021]. Setup and numbers borrowed from PEC [Zajac et al., 2024]. RanDumb outperforms the best exemplar-free OCL method. Comparing representation learning with no representation learning, <i>a fixed, random function outperforms training a deep network continually, i.e. $RepGain(\mathcal{D}) < 0$</i> . Furthermore, RanDumb nearly matches performance of one-pass joint, demonstrating the inefficacy of current benchmarks.	28

3.1.2	Offline Continual Learning. Performance comparison with ImageNet21K ViT-B16 model using 2 init classes and 1 new class per task. RanDumb here assumes the features as input pixels. SLCA Zhang et al. [2023a] and RanPAC McDonnell et al. [2023] results reproduced from original codebase. Comparing representation learning with no representation learning, <i>a fixed, pretrained feature extractor outperforms further finetuning a deep network continually, i.e. $\text{RepGain}(\mathcal{D}) < 0$</i> . RanDumb nearly matches performance of joint, demonstrating the inefficacy of current benchmarks. *here representation learning collapses, accuracy reported without PETL. +here representation learning and random projection both collapse, reporting last stable accuracy.	29
3.2.1	Benchmark (Overview). We evaluate RanDumb across a diverse set of benchmarks that progressively relax constraints compared to RanDumb. Benchmark A closely matches RanDumb with one class per timestep and no stored exemplars. Benchmark B, D, E progressively relax the constraints on exemplars and classes per timestep. Benchmark C and E remove the online constraint by allowing unrestricted training and sample access within a task without exemplar-storage of past tasks. Benchmark F allows using large pretrained models, modified with one class per task, inspired by challenges detailed in Benchmark A.	32
3.2.2	Benchmark A (Ref: Table 1 from PEC Zajac et al. [2024]). We perform comprehensive comparisons of RanDumb with popular online continual learning approaches in a 1-class per task setting referred as ‘Dataset (num_tasks/1)’. This setting is a challenging setting for OCL methods except GDumb. We observe that RanDumb outperforms all approaches across all datasets by 2-6% margins, with an exception of latest work PEC Zajac et al. [2024] on CIFAR10.	32
3.3.1	Benchmark B.1 (Ref: Table adopted from OnPro Wei et al. [2023], OCMGuo et al. [2022]) We perform comprehensive comparisons of RanDumb with popular online continual learning approaches in less ideal many-classes per task setting referred as ‘Dataset (num_tasks/num_classes_per_task)’. Furthermore, results are categorized by memory buffer sizes given in ‘M’ subcolumn for that dataset. Comparing across rows, we observe that RanDumb outperforms the best among the compared approaches without heavy-augmentations by 3-20% margins despite being exemplar free. Only in one case, it is second best, but after GDumb. Comparing the two columns of CIFAR100, we observe a large drop in performance of OCL methods on benchmarks with longer timesteps indicating analyzing longer timesteps is important in continual learning.	34

- 3.3.2 **(Left) Benchmark B.2** (*Ref: Table from OnPro Wei et al. [2023]*) We now compare with state-of-the-art contrastive representation learning based online continual learning approaches which additionally use sophisticated augmentations. These augmentations and additional loss function can improve any of the OCL methods described in Setup B.1 by large margins Guo et al. [2022]. We observe that RanDumb often outperforms these sophisticated methods despite all of these factors on small-exemplar settings. **(Right) Benchmark C** (*Ref: Table 2 from Smith et al. [2023b]*). We compare the performance of RanDumb with latest efforts on improving rehearsal-free methods. We outperform them by 4% margins. 35
- 3.3.3 **(Left) Benchmark D** (*Ref: Table 2 from VR-MCL Anonymous [2024]*) We compare RanDumb with meta-continual learning approaches operating in a high memory setting, allowing buffer sizes up to 1K exemplars. Despite catering representations specifically to combat forgetting, all methods with the exception of VR-MCL are outperformed by RanDumb on TinyImageNet. RanDumb also surpasses all prior work by a substantial 9.1% on CIFAR100. Allowing generous replay buffers shifts scenarios to a high exemplar regime where GDumb performs the best on CIFAR10. Yet RanDumb competes favorably even under these conditions not optimized for its approach. **(Middle) Benchmark E** (*Ref: Table 1 from SEDEM Ye and Bors [2023]*) We compare RanDumb with recent network expansion based online continual learning approaches. Despite allowing access to much larger memory buffers, RanDumb matches the performance of best method SEDEM on MNIST, while exceeding it by 0.3% on CIFAR10 and 3.8% on CIFAR100. **(Right) Architectures** (*Ref: Table 1 from Mirzadeh et al. [2022]*) RanDumb surpasses continual representation learning across a wide range of architectures, achieving close to 94% of the joint performance. 35
- 3.3.4 **Benchmark F** We compare RanDumb with other prompt-tuning based continual learning approaches using ViT-B/16 ImageNet-21K/1K pretrained models using 2 init classes and 1 class per task setting. Most prompt-tuning based methods collapse and simple baselines which do not pretrain the architecture like NCM Janson et al. [2022] or RanDumb achieve state-of-the-art performance. '-' indicates that despite reasonable efforts, we could not run the method. +here representation learning and random projection both collapse, reporting last stable accuracy. 37

3.3.5 RanDumb (Analysis). We thoroughly analyse the key components of RanDumb, and observe: (1) contributions of decorrelation and embedding, finding them interdependent and integral to performance; (2) consistent gains from augmentation across datasets; (3) marginal returns and saturation with higher embedding sizes, enabling computational tradeoffs; (4) increase in the optimal regularisation parameter for more complex datasets; (5) comparisons with alternate embeddings (or lack thereof) favoring RanDumb’s approach.	38
4.2.1 Primary Directions of Progress in CL. Analysis of recent replay-based systems, with bold highlighting the primary contribution. We observe that there are three primary directions of improvement. “App.” denotes the applicability to our setting based on whether they are scalable to large datasets and applicable beyond the class-incremental stream.	45
4.4.1 Experimental Details. The first block shows the various considered settings in the experiments section. The second block denotes the effective training iterations C for each class of methods due to their over head extra computation. The last block details the setup for our sensitivity analysis.	53
5.2.1 Breakdown of popular OCL systems, with key contributions in red . Most methods focus on sampling techniques for storing datapoints, which cannot transfer here as we store all past samples.	63
6.2.1 Properties of OCL Benchmarks. We list benchmarks in the field of OCL along with the works which introduced them. We compare them across six properties: Realistic Data Streams (DO), No Storage Constraints (SC), Rapid Adaptation (RA), Information Retention (IR), Restricts Computational Budgets (CB), Evaluates on long stream sizes (LB), Leverages Pretrained Models (PT). Note that ✓ is better than ✗ in all columns.	78
7.4.1 Performance Analysis between Uncurated Webly-Supervised Data (C2C) and Manually Annotated Training (MA) Data. Despite utilizing uncurated web data, our results demonstrate competitive or even better performance than that of manually annotated datasets in fine-grained categorization tasks. The most notable improvements are observed when using MLP-adapters.	100
7.4.2 Comparison with Name-Only Classification Techniques with ResNet50: When comparing with existing state-of-the-art name-only classification techniques, we show that our method outperforms those methods by margins ranging from 2% to 25%.	100

7.4.3	Impact of Size of Queried Webly-Supervised Data on Performance. This table illustrates the influence of downsizing our queried web data by considering only the top- k queries for download. Notably, a substantial performance drop occurs as the dataset size decreases. Despite the higher quality of the top- k samples, their limited quantity adversely affects performance. We use Manually Annotated Training (MA) Data as a reference point.	103
7.5.1	Comparison with Name-Only Classification Techniques with ResNet50: When comparing with existing state-of-the-art name-only classification techniques, we show that our method outperforms those methods by margins ranging from 2% to 25%.	103
7.5.2	Linear Probe Performance in Continual Learning Scenarios (Avg. Acc. $\uparrow$). Our uncurated webly-supervised data achieves average accuracy close to manually annotated (MA) datasets in a continual learning context with relatively small performance gaps.	106
7.5.3	Comparing Last-Layer Based Continual Learning Approaches in Name-Only Continual Learning (Avg. Acc. $\uparrow$). We evaluate the average accuracy of various continual learning methods in a name-only continual learning scenario with constrained computational resources. Surprisingly, KNN achieves superior results compared to linear probing, even while operating within a lower computational budget than the "tight" setting.	107
7.5.4	(Left) Time Benchmark (in minutes): Evaluation of the approximate total time required for continual learning across all time steps using a linear classifier. Our pipeline demonstrates exceptional time efficiency compared to manual sample annotation. (Right) EvoTrend Results: Comparative performance analysis of Linear Probing on C2C using MoCoV3 ResNet50 against zero-shot CLIP with ResNet50, indicating improved performance.	107
A.0.1	Minimizing the resource consumption (storage size and parameters).	116
A.1.1	Accuracy of tweaked (NIN/DenseNet) GDumb models with number of passes. The bolded accuracies represent the reported results in previous experiments, while accuracies in (brackets) are obtained without cutmix regularization. We can halve the training time with slight tradeoff of upto 1% accuracy.	117
B.2.1	Cost of Memory vs Computation. Google Cloud Standard Storage (2¢per GB per month for 1 month) and Compute Cost measured as running cost of an A2 instance (3\$ per hour for 1 GPU). Number of iterations (forward+backward passes) for training a CL model on that dataset listed for comparison invariant to input image and model sizes. We observe that computational costs for running an experiment far outweigh the costs for storing replay samples.	122

E.2.1 Dataset Statistics. Our uncured webly-supervised data is significantly larger in size compared to manually annotated (MA) datasets. Leveraging multiple engines could allow to query the web for samples enabling the creation of datasets up to 155× larger than manually curated ones.	148
E.2.2 Impact of Architecture and Pretraining Algorithm. Irrespective of the selected network architecture and pretraining algorithm, our uncured webly-supervised data consistently exhibit superior performance over manually annotated datasets by substantial margins. The improvement is primarily attributed to the scale of web data.	149
E.2.3 Performance Across Search Engines. We find that different search engines demonstrate varying levels of performance across different datasets, and their collective utilization yields superior results compared to any individual search engine. It is worth noting that Flickr encountered difficulties in querying and downloading images for certain classes in the Cars dataset, hence the performance for those classes is not reported.	150
E.2.4 Effect of Architecture and Pretraining Algorithm. We observe different search engines performing well across different datasets, with their combined data consistently outperforming any individual search engine.	150
E.3.1 ViTB/16: Comparison with Other Approaches for Name-Only Classification Our uncured webly-supervised data consistently outperform existing name-only classification techniques, with the exception of the Stanford Cars dataset where Priming outperforms our approach. * indicates that the method significantly differs from the cited work. However, it performs SSL finetuning with retrieved samples, emulating a 1-step Internet-explorer algorithm. Further exploration of LAION2B is not possible here due to the lack of a training set.	153
E.3.2 Dataset Statistics. Our uncured webly-supervised data is significantly larger in size compared to other approaches for acquiring a training set. Leveraging multiple engines could allow to query the web for samples enabling the creation of datasets up to 15 – 155× larger than alternative approaches for creating a training set in a name-only classification setting.	154
E.4.1 Consistency of Linear Probing Performance across Computational Budgets. Our findings reveal a remarkable consistency in the results in terms of performance gap across different computational budgets.	156

E.4.2 CLEAR10 Class-wise Performance. The analysis of class-wise performance in CLEAR10 highlights a significant decline in accuracy specifically for the bus class. This discrepancy can be attributed to the fact that CLEAR10 test set includes images of the "interior" of the bus, whereas our webly-supervised data primarily consists of images depicting the "exterior" of the bus. As a result, the accuracy for our bus class is notably low. Please refer to Figure E.4.3 for a visual representation of this disparity.	158
E.4.3 Domain-specific Accuracy in Continual PACS. Our domain-incremental setup incorporates a dataset consisting of four domains: Photo, Art, Cartoon, and Sketch. It is important to note that there is a notable disparity between the sketches in PACS, which are characterized as quick drawings, and the sketches in our webly-supervised data, which encompass more detailed sketches and line drawings. This domain shift results in a significant performance drop for our approach in the Sketch domain. This domain shift can be observed in Figure E.4.4.	159
E.4.4 Statistics of Continual Datasets. In contrast to the datasets collected in the <i>name-only</i> classification setup, the scale of the datasets used in the <i>continual name-only</i> classification scenario is maximally $5.5\times$ larger than the manually curated ones, whereas in the previous scenario it was $144\times$ larger. This arises from the fact that the continual datasets we compare against are already big in size.	161

1

Introduction

This is an integrated thesis where submitted or published work is reformatted as different chapters. In the Introduction, we provide a brief introduction to continual learning and motivate why computational cost considerations, review shortcomings in traditional approaches for tackling this problem, and present the outline of proposed solutions in this thesis, and list the relevant publications. The introduction is organized as follows:

Contents

1.1	A Very Brief Introduction to Continual Learning	2
1.2	Why Computational Cost Considerations?	2
1.3	Thesis Outline and Contributions	4

1.1 A Very Brief Introduction to Continual Learning

Deep learning has excelled in various computer vision tasks [He et al., 2016, Schmidhuber, 2015, Bommasani et al., 2021, LeCun et al., 2015] by performing hundreds of shuffled passes over a static labeled dataset. However, the fundamental characteristic of humans is their ability to quickly learn concepts, more generally, continually learn new concepts while remembering information about the old ones. Realizing that very objective in machines is precisely the motivation behind continual learning (CL).

What does realizing this objective in machine looks like? We imagine designing machine learning systems which can continually learn from streaming data. This can revolutionize a wide range of real-world systems, e.g., Instagram, TikTok, and Flickr, which experience high throughput of a constantly changing stream of data for, which say, misinformation or hate speech, are time-sensitive in which long training times before deployment are simply not an option. Otherwise, new stream data will accumulate until training is completed, causing server delays and worsening user experience. Continual learning (CL) algorithms update a model continuously from data streams with a non-stationary distribution where samples are presented incrementally over time, often referred to as time steps. This requires them to solve a number of problems: they needs to successfully learn the main task (accuracy), rapidly adapt to changes in the distribution, and retain information from the past.

However, retain information from the past while rapidly learning new (or even related) tasks continually is extremely difficult for deep models as, in such scenarios, they are prone to the phenomenon called catastrophic forgetting [McCloskey and Cohen, 1989, Goodfellow et al., 2013]. Significant attention has been paid recently to this problem [Li et al., 2017a, Rebuffi et al., 2017b, Zenke et al., 2017, Kirkpatrick et al., 2017, Lopez-Paz and Ranzato, 2017, Chaudhry et al., 2018] and a diverse set of approaches have been proposed in the literature (refer to De Lange et al. [2020] for an overview of traditional methods).

1.2 Why Computational Cost Considerations?

Current approaches for continual learning impose different sets of simplifying constraints to the continual learning problem and propose tailored algorithms for the same. Sometimes these constraints are so rigid that they even break the notion of learning continually, for example, one such constraint would be knowing a priori the subset of labels a given input might take. In addition, these approaches are never tested exhaustively on scalable realistic scenarios. Keeping this observation in mind, we suggest that it is of paramount importance to understand the caveats in these simplifying

Table 1.2.1: The cost of storing data has decreased rapidly, allowing the storage of a large dataset for a negligible cost compared to the cost of computation for both cloud and mobile systems.

Cloud Storage	1987	1997	2007	2017	Mobile NAND	2000	2005	2010	2017	2022
\$/MB	83.33	0.22	0.0003	0.00002	\$/MB	1100	85	1.83	0.25	0.03
Cost of storing CLOC (\$)	350M	920K	1250	83	Cost of storing CLOC (\$)	4M	300K	6350	850	70
Training Cost (ER)			>2000\$		Training Cost (ER)			Not Currently Feasible		

assumptions, understand why these simplified forms are of little practical usability under typical commercial settings. We shall elaborate on these problems in Chapters 2 and 3.

As a first step to address the limitations of current continual learning systems, we ask: what is required of continual learning algorithms in commercial settings? A scalable continual learning algorithm must deliver accurate predictions, scale to very large data streams encountered during its operational lifetime, and operate within the system’s total cost budget (in dollars). The economics of data storage have been studied since 1987 [Gray and Putzolu, 1987, Gray and Graefe, 1997, Graefe, 2009, Appuswamy et al., 2017]. Table 1.2.1 summarizes the trends, show a rapid decline in storage costs over time ($\sim$ \$100 to store CLOC, the largest dataset for OCL [Cai et al., 2021], in 2017). In contrast, training an ER baseline [Cai et al., 2021], the state-of-the-art OCL method on CLOC currently costs over \$2000 on a GCP server and is still not commercially feasible on mobile devices. Moreover, on Google Cloud Standard Storage (2¢ per GB per month), it costs no more than 6¢ to store the entire CLEAR benchmark [Lin et al., 2021], a recent large-scale CL dataset. Consequently, computational constraints are the primary concern, with storage costs being relatively insignificant. Therefore, as long as computational costs are controlled, economically storing the entire incoming data stream is feasible. This is the primary motivation for pursuing continual learning with computational cost considerations.

However, continual learning has primarily been studied under limited storage constraints [Lopez-Paz and Ranzato, 2017, Chaudhry et al., 2019a, Aljundi et al., 2019b, Rebuffi et al., 2017b, Hou et al., 2019, Kirkpatrick et al., 2017], with learners only allowed to store a subset of incoming data but do not have restrictions on the computational training budget per time step. This constraint has led to many algorithms focusing on identifying a representative data subset [Aljundi et al., 2019b, Yoon et al., 2022, Chrysakis and Moens, 2020, Bang et al., 2021, Sun et al., 2022, Koh et al., 2022]. While limited storage aligns with the practical constraints of biological learning agents and offline embodied artificial agents, deep learning-based systems are largely compute-constrained because computation is expensive but storage is cheap. Therefore, it is prudent to have computationally budgeted methods. The problems of traditional continual learning remain because

under computational cost considerations the memory size, as a consequence, is still implicitly restricted. This is because, under a computational budget, it is no longer possible to revisit all previous data even if they were all stored in memory (given their low memory costs). We shall study methods to overcome this challenge in Chapters 4 and 5.

What are other impacts of considering the problem of cost minimization seriously? Continual Learning (CL) predominantly rely on annotated data streams, i.e., a common underlying assumption is the availability of well-curated, annotated datasets. However, the financial and temporal costs associated with continual annotation is staggering. To illustrate this, annotating 30K samples in the CLEAR10 dataset [Lin et al., 2021], a popular CL dataset, despite using optimized annotation workflows with large CLIP models [Radford et al., 2021], cost \$4,500 and more than a day worth of annotation time. In contrast, businesses like Amazon and Fast Fashion companies constantly need to update their image classification models and associated recommendation engines due to changing inventory, seasonal and customer trends. Annotating labeled training sets every time for commercial classification models with thousands of categories and millions of samples is unrealistic, as it would take weeks and cost hundreds of thousands of dollars. In short, manual data collection and annotation are expensive and time-consuming, posing a bottleneck in real-world continual learning. We develop strategies for alleviating continual labeling costs in Chapter 7.

Lastly, we investigate metrics in continual learning. We argue that current metrics measuring rapid adaptation have deviated significantly from measuring true adaptation capabilities, apart from neglect in measuring the capacity of models to retain previous knowledge. As a consequence, several online continual learning algorithms achieve high online accuracy Cai et al. [2021] but poor performance on other important metrics like *information retention* which measure catastrophic forgetting of past seen data. We demonstrate that this is due to critical issues in the online accuracy metric, and propose solutions in Chapter 6.

1.3 Thesis Outline and Contributions

The main contribution of this thesis is to provide a continual learning system with computational cost constraints. To provide a brief outline, Chapter 2 and Chapter 3 discover fundamental problems in continual learning formulations. Chapter 4 and Chapter 5 propose an alternate formulation for the case of offline continual learning and online continual learning respectively, along with investigating state-of-the-art approaches to adopt in these setups. Combining these two components would give us the slow-learning and fast-learning components of a continual learning system. Chapter 6

and Chapter 7 fill the gaps among the other components in a comprehensive continual learning setup – tackling critical gaps in metrics and labeling costs respectively. Overall, they provide a comprehensive continual learning system under computational cost considerations.

Please note that this is an integrated thesis where published or submitted work in peer-reviewed conferences is reformatted as different chapters. Hence, each of the chapters 2– 7 is based on a different published or submitted work at a leading peer-reviewed conference in machine learning and computer vision (*: Equal contribution). Each of these chapters is self-contained with its own related work section centred around the contribution of the paper. These publications are discussed in more detail in Appendix F. In the remainder of this section, we discuss the contributions of each chapter in detail.

Chapter 2 This chapter discusses a general approach to the Continual Learning (CL) problem for classification. It proposes a general formulation inspired by the open-set recognition problem and critiques the oversimplification of CL in recent methods, concluding they lack practical importance. It achieves this by introducing a new method called GDumb, which greedily stores samples in memory as they arrive and, at test time, trains a model from scratch using only these samples. Remarkably, GDumb, despite not being specifically designed for CL problems, achieves state-of-the-art performance and often outperforms methods tailored for specific CL scenarios. This suggests exploration of alternative, better formulations for continual learning.

The corresponding publication is–

[Prabhu et al., 2020]: **Ameya Prabhu**, Philip H.S. Torr, and Puneet K. Dokania.
“GDumb: A Simple Approach that Questions Our Progress in Continual Learning.”
European Conference on Computer Vision (ECCV), 2020 - Oral

Full Text. [Link To Paper](#)

Reproducibility. [Link To Code](#)

Chapter 3 This chapter proposes RanDumb to examine the efficacy of continual representation learning. RanDumb embeds raw pixels using a fixed random transform which approximates an RBF-Kernel, initialized before seeing any data, and learns a simple linear classifier on top. We present a surprising and consistent finding: RanDumb significantly outperforms the continually learned representations using deep networks across numerous continual learning benchmarks, demonstrating the poor performance of representation learning in these scenarios. RanDumb stores no exemplars and performs a single pass over the data, processing one sample at a time. It complements the

above chapter, GDumb [Prabhu et al., 2020], operating in a low-exemplar regime where GDumb has especially poor performance. We reach the same consistent conclusions when RandDumb is extended to scenarios with pretrained models replacing the random transform with pretrained feature extractor. Our investigation is both surprising and alarming as it questions our understanding of how to effectively design and train models that require efficient continual representation learning.

The corresponding publication is–

[Prabhu et al., 2024]: **Ameya Prabhu***, Shiven Sinha*, Ponnurangam Kumaraguru, Philip H.S. Torr, Ozan Sener⁺, Puneet K. Dokania⁺. “RandDumb: A Simple Approach that Questions the Efficacy of Continual Representation Learning.” *Under Review*

Full Text. [Link To Paper](#)

Reproducibility. [Link To Code](#)

Chapter 4 This chapter challenges the current focus of offline Continual Learning (CL) literature on limited access to previous data without considering computational budget constraints. It introduces a large-scale benchmark for analyzing traditional CL methods under computational limitations, revealing that these methods fail to outperform a basic strategy that uniformly samples from memory. By conducting experiments over 1500 GPU-hours on large-scale datasets like ImageNet2K and Continual Google Landmarks V2, it shows that most existing offline CL methods are computationally expensive for realistic deployment, highlighting the need for more efficient approaches in computationally constrained environments.

The corresponding publication is–

[Prabhu et al., 2023b]: **Ameya Prabhu***, Hasan Abed Al Kader Hammoud*, Puneet K. Dokania, Philip H.S. Torr, Ser-Nam Lim, Bernard Ghanem, Adel Bibi. “Computationally Budgeted Continual Learning: What Does Matter?” *Computer Vision and Pattern Recognition (CVPR)*, 2023.

Full Text. [Link To Paper](#)

Reproducibility. [Link To Code](#)

Chapter 5 This chapter introduces the challenge of learning under severe computational constraints in an online continual learning (OCL). It proposes propose a novel algorithm using a kNN classifier and pre-trained feature extractors that can compactly store and process the entire data stream with minimal computational resources. The selected algorithm is surprisingly suitable for online continual learning. It can adapt to rapidly changing streams, has zero stability gap, operates

within tiny computational budgets, has low storage requirements by only storing features, and has a consistency property: It *never forgets* previously seen data. The algorithm sets a new benchmark on large-scale OCL datasets like Continual LOCalization and Continual Google Landmarks V2, outperforming existing methods in reducing catastrophic forgetting and adapting to changing data streams under tiny computational budgets.

The corresponding publication is–

[Prabhu et al., 2023a]: **Ameya Prabhu**, Zhipeng Cai, Puneet K. Dokania, Philip H.S. Torr, Vladlen Koltun, Ozan Sener. “Online Continual Learning Without the Storage Constraint” *Arxiv*.

Full Text. [Link To Paper](#)

Reproducibility. [Link To Code](#)

Chapter 6 This chapter critically examines the prevalent methods of evaluating Online Continual Learning (OCL) algorithms, particularly the use of online accuracy as a metric. It highlights that this metric can be misleading, as it may favor algorithms that exploit spurious label correlations, leading to high accuracy but poor information retention. The chapter proposes a new metric focusing on accuracy for near-future samples, eliminating these spurious correlations. This approach, tested on large-scale datasets, shows that better generalization is achievable by retaining and reusing previously seen information, pointing towards a more effective way to develop adaptive OCL methods.

The corresponding publication is–

[Hammoud et al., 2023]: Hasan Abed Al Kader Hammoud*, **Ameya Prabhu***, Ser-Nam Lim, Philip H.S. Torr, Adel Bibi, Bernard Ghanem. “Rapid Adaptation in Online Continual Learning: Are We Evaluating It Right?” *International Conference on Computer Vision (ICCV)*, 2023.

Full Text. [Link To Paper](#)

Reproducibility. [Link To Code](#)

Chapter 7 This chapter presents a framework to alleviate the labeling bottlenecks in continual learning, where algorithms need to adapt to new category shifts using only category names, without the need for annotated training data. It proposes an approach utilizing the vast and evolving internet to query and download relevant data, demonstrating that it can often outperform traditional name-only classification methods by large margins. It further shows that the proposed data collection method can effectively adapt to varying continual learning contexts with minimal computational

costs, and introduce EvoTrends, a class-incremental dataset made from the web to illustrate this effect. This work underlines the potential of using webly-supervised data to overcome challenges associated with manual data labeling in CL.

The corresponding publication is–

[Prabhu et al., 2023c]: **Ameya Prabhu***, Hasan Abed Al Kader Hammoud*, Ser-Nam Lim, Philip H.S. Torr, Adel Bibi, Bernard Ghanem. “From Categories to Classifier: Name-Only Continual Learning by Exploring the Web” *Under Review*.

Full Text. [Link To Paper](#)

Reproducibility. [Link To Code](#)

2

GDumb: A Simple Approach that Questions Our Progress in Continual Learning

Contents

2.1	Introduction	11
2.2	Problem Formulation, Assumptions, and Trends	11
2.2.1	Simplifying Assumptions in Continual Learning	12
2.2.2	Recent Trends in Continual Learning	15
2.3	Greedy Sampler and Dumb Learner (GDumb)	16
2.4	Experiments	19
2.4.1	Results and Discussions	20
2.4.2	Resources Needed	24
2.4.3	Potential Future Extensions	24
2.5	Conclusion	24

Abstract

We first discuss a general formulation for the Continual Learning (CL) problem for classification—a learning task where a stream provides samples to a learner and the goal of the learner, depending on the samples it receives, is to continually upgrade its knowledge about the old classes and learn new ones. Our formulation takes inspiration from the open-set recognition problem where test scenarios do not necessarily belong to the training distribution. We also discuss various quirks and assumptions encoded in recently proposed approaches for CL. We argue that some oversimplify the problem to an extent that leaves it with very little practical importance, and makes it extremely easy to perform well on. To validate this, we propose GDumb that (1) greedily stores samples in memory as they come and; (2) at test time, trains a model from scratch using samples only in the memory. We show that even though GDumb is not specifically designed for CL problems, it obtains state-of-the-art accuracies (often with large margins) in almost all the experiments when compared to a multitude of recently proposed algorithms. Surprisingly, it outperforms approaches in CL formulations for which they were specifically designed. This, we believe, raises concerns regarding our progress in CL for classification. Overall, we hope our formulation, characterizations and discussions will help in designing realistically useful CL algorithms, and GDumb will serve as a strong contender for the same.

2.1 Introduction

In this chapter, we first provide a *general* formulation of CL for classification. Then, we investigate popular variants of existing CL algorithms, and categorize them based on the simplifying assumptions they impose over the said general formulation. We discuss how each of them impose constraints either over the growing nature of the label space, the size of the label space, or over the resources available. One of the primary drawbacks of these restricted settings is that algorithms tailored towards them fail miserably when exposed to a slightly different variant of CL, making them extremely specific to a particular situation. We would also like to emphasize that there is no explicit consensus among researchers regarding which formulation of CL is the most appropriate, leading to a diverse experimental scenarios, none of which actually mimic the general form of CL problem one would face when exposed to the real-world.

Then, we take a step back and design an extremely simple algorithm with almost no simplifying assumptions compared to the recent approaches. We call this approach GDumb (Greedy Sampler and Dumb Learner). As the name suggest, the two core components of our approach are a *greedy sampler* and a *dumb learner*. Given a memory budget, the sampler greedily stores samples from a data-stream while making sure that the classes are balanced, and, *at inference*, the learner (neural network) is trained from scratch (hence dumb) using all the samples stored in the memory. When tested on a variety of scenarios on which various recent works have proposed highly tuned algorithms, GDumb surprisingly provides state-of-the-art results with large margins in almost all the cases.

The fact that GDumb, even though not designed to handle the intricacies in the challenging CL problems, outperforms recently proposed algorithms in their own experimental set-ups, is alarming. It raises concerns relating to the popular and widely used assumptions, evaluation metrics, and also questions the efficacy of various recently proposed algorithms for continual learning.

2.2 Problem Formulation, Assumptions, and Trends

To provide a *general* and practically useful view of CL problem, we begin with the following example. Imagine a robot walking in a living room solving a task that requires it to identify all the objects it encounters. In this setting, the robot will be identifying known objects that it has learned in the past, will be learning about a few *unknown* objects by asking an oracle to provide labels for them, and, at the same time, will be updating its information about the known objects if the new instances of them provided extra cues useful for the task. In a nutshell, the robot

begins with some partial information about the world and keeps on improving its knowledge about it as it explores new parts of the world.

Inspired by this example, a realistic formulation of continual learning for classification would be where there is a stream of training samples or data accessible to a learner, each sample comprising a two-tuple $(\mathbf{x}_t, \mathbf{y}_t)$, where t represents the timestamp or the sample index. Let $\mathcal{Y}_t = \cup_{i=1}^t \mathbf{y}_i$ be the set of labels seen until time t , then it is trivial to note that $\mathcal{Y}_{t-1} \subseteq \mathcal{Y}_t$. This formulation implies that the stream might give us a sample that either belongs to a new class or to the old ones. Under this setting, at any *given* t , the objective is to provide a mapping $f_{\theta_t} : \mathbf{x} \rightarrow \mathbf{y}$ that can accurately map a sample $\mathbf{x}$ to a label $\mathbf{y} \in \mathcal{Y}_t \cup \bar{\mathbf{y}}$, where $\bar{\mathbf{y}}$ indicates that the sample does not belong to any of the learned classes. Notice, addition of this extra label $\bar{\mathbf{y}}$ assumes that while training, there is incomplete knowledge about the world and a test sample might come from outside the training distribution. Interestingly, it connects an instance of CL very well with the well known open-set classification problem [Scheirer et al., 2012]. However, in CL, the learner, with the help of an oracle (*e.g.*, active learning), could improve its knowledge about the world by learning the semantics of samples inferred as $\bar{\mathbf{y}}$.

2.2.1 Simplifying Assumptions in Continual Learning

The above discussed formulation is general in the sense that it does not put any constraints whatsoever on the growing nature of the label space, nature of test samples, and size of the output space $|\lim_{t \rightarrow \infty} \mathcal{Y}_t|$. It does not put any restrictions on the resources (compute and storage) one might pick to get a reliable mapping $f_{\theta_t}(\cdot)$ either, however, the lack of information about the nature and the size of the output space makes the problem extremely hard. This has compelled almost all the work in this direction to impose additional simplifying constraints or assumptions. These assumptions are so varied that it is difficult to compare one CL algorithm with another as a slight variation in the assumption might change the complexity of the problem dramatically. For better understanding, below we discuss all the popular assumptions, highlight their drawbacks, and categorize various recently proposed CL algorithms depending on the simplifying assumptions they make. One assumption common to all is that the test samples always belong to the training distribution.

Disjoint task formulation: This formulation is being used in almost all the recent works [Rebuffi et al., 2017b, Zenke et al., 2017, Kirkpatrick et al., 2017, Lopez-Paz and Ranzato, 2017, Chaudhry et al., 2018] whereby the assumption made is that at a particular *duration* in time, the data-stream will provide samples specific to a task, in a pre-defined order of *tasks*, and the aim is to learn the mapping by learning each task at a time sequentially. In particular, let $\mathcal{Y} = \lim_{t \rightarrow \infty} \mathcal{Y}_t$ be the set of labels that the stream might present until it runs out of samples. Recall, in the general CL formulation, the size

of $\mathcal{Y}$ is unknown and the samples can be presented in any order. This label space $\mathcal{Y}$ is then divided into different disjoint subsets (could be a random or an informed split), where each label subset $\mathcal{Y}_i$ represents a task and the *sharp* transition between these sets is called *task boundaries*. Let there be m splits (typically the split is balanced with nearly equal number of classes) then $\mathcal{Y} = \cup_i^m \mathcal{Y}_i$, and $\mathcal{Y}_i \cap \mathcal{Y}_j = \emptyset, \forall i \neq j$. An easy and widely used example is to divide ten digits of MNIST into 5 disjoint tasks where each task comprises of the samples from two consecutive digits and the stream is *controlled* to provide samples for each task in a pre-defined order, say $\{0, 1\}, \dots, \{8, 9\}$. This formulation simplifies the general CL problem to a great extent as the unknown growing nature of the label space is now being restricted and is known. It provides a very strong prior to the learner and helps in deciding both the space budget and the family of functions $f_\theta(\cdot)$ to learn.

Task-incremental v/s Class-incremental: To further make the training and the inference easier, a popular choice of CL formulation is the task-incremental continual learning (TI-CL) [Lopez-Paz and Ranzato, 2017] where, along with the disjoint task assumption, the task information (or id) is also passed by an oracle during training and inference. Thus, instead of a two-tuple, a three-tuple $(\mathbf{x}, \mathbf{y}, \alpha)$ is given where $\alpha \in \mathbb{N}$ represents the task identifier. This formulation is also known as *multi-head* and is an extremely simplified form of the continual learning problem [Chaudhry et al., 2018]. For instance, in the above mentioned MNIST example, at inference, if the input is $(\mathbf{x}, \alpha = 3)$, it implies that the sample either belongs to class 4, or to 5. Knowing this subset of labels a-prior dramatically reduces the label space during training and inference, and is relatively impractical to know in real-world scenarios. Whereas, in a class-incremental formulation (CI-CL) [Rebuffi et al., 2017b, Chaudhry et al., 2018], also known as the *single-head*, we do not have any such information about the task id.

Online CL v/s Offline CL: Note, the disjoint task formulation placed a restriction on the growing nature of the label space and inherently restricted the size of it, however, it did not put any constraints on the learner itself. Therefore, the learning paradigm may store task-specific samples coming from the stream depending on the space budget and then use them to update the parameters. Under this setting, in the *online* CL formulation, even though the learner is allowed to store samples as they come, they are not allowed to use a sample more than once for parameter update. Thus, the learner can not use the same sample (unless it is in the memory) multiple times at different iterations of the learning process. In contrast, *offline* CL allows unrestricted access to the *entire* dataset corresponding to a particular task (not to the previous ones) and one can use this dataset to learn the mapping by revisiting the samples again and again while performing multiple passes over the data [Rebuffi et al., 2017b].

Form.	CI-CL	Online	Disjoint	Papers	Regularize	Memory	Distill	P. iso
A	✓	✓	✓	MIR[Aljundi et al., 2019a], GMED[Jin et al., 2021]	×	✓	×	×
				LwM[Dhar et al., 2019], DMC[Zhang et al., 2020b]	×	×	✓	×
				SDC [Yu et al., 2020]	✓	×	×	×
				BiC[Wu et al., 2019], iCARL[Rebuffi et al., 2017b]				
B	✓	×	✓	UCIR[Hou et al., 2019], EEIL[Castro et al., 2018]	×	✓	✓	×
				IL2M[Belouadah and Popescu, 2019], WA[Zhao et al., 2020]				
				PODNet[Douillard et al., 2020], MCIL[Liu et al., 2020]				
				RPS-Net[Rajasegaran et al., 2019], iTAML[Rajasegaran et al., 2020]	×	✓	✓	✓
				CGATE[Abati et al., 2020]	×	✓	×	✓
				RWALK[Chaudhry et al., 2018]	✓	✓	×	×
C	×	×	✓	PNN[Rusu et al., 2016], DEN[Yoon et al., 2018]	×	×	×	✓
				DGR [Shin et al., 2017]	×	✓	×	×
				LwF[Li et al., 2017a]	×	×	✓	×
				P&C[Schwarz et al., 2018]	×	×	✓	✓
				APD[Yoon et al., 2020]	✓	×	×	✓
				VCL[Nguyen et al., 2018]	✓	✓	×	×
				MAS[Aljundi et al., 2018], IMM[Lee et al., 2017]				
				SI[Zenke et al., 2017], Online-EWC[Schwarz et al., 2018]	✓	×	×	×
				EWC[Kirkpatrick et al., 2017]				
				TinyER[Chaudhry et al., 2019b]	×	✓	×	×
D	×	✓	✓	HAL[Chaudhry et al., 2021]	×	✓	×	×
				GEM[Lopez-Paz and Ranzato, 2017]	✓	✓	×	×
				AGEM[Chaudhry et al., 2019a]	✓	✓	×	×
E	✓	✓	×	GSS[Aljundi et al., 2019b]	×	✓	×	×

Table 2.2.1: Here we categorize various recently proposed CL approaches depending on the underlying simplifying assumptions they impose.

Memory based CL: As mentioned earlier, we only have access to all/subset of samples corresponding to the *current* task. This restriction makes it extremely hard for the model to perform well, in particular, on CI-CL setting as the absence of samples from the previous tasks makes it difficult to learn to distinguish samples from the current and the previous tasks due to catastrophic forgetting. Very little forgetting is normally observed in TI-CL as the given task identifier works as the indicator of task boundary, thus the model does not have to learn to differentiate labels among tasks. To reduce forgetting, a common practice, inspired by the complementary learning systems theory [Tulving, 2002, Norman and O’Reilly, 2003], is to store a subset of samples from each task and use them while training on the current task. There primarily are two components under this setting: a *learner* and a *memorizer* (or sampler). The learner has the goal of obtaining representations which generalize beyond current task. The memorizer, on the other hand, deals with remembering (storing) a collection of episode-like memories from the previous tasks. In recent approaches [Rebuffi et al., 2017b, Lopez-Paz and Ranzato, 2017, Chaudhry et al., 2018], the learner is modeled by a neural network and the memorizer is modeled by memory slots which store samples previously encountered.

2.2.2 Recent Trends in Continual Learning

Typically, continual learning approaches are categorized by ways they tackle forgetting such as (1) regularization-based, (2) replay (or memory)-based, (3) distillation-based, and (4) parameter-isolation based (for details refer [De Lange et al., 2020]). However, they do vary in terms of simplifying assumptions they encode, and we argue that keeping track of these assumptions is extremely important for fair comparisons, and also to understand the limitations of each of them. Since all these algorithms in some sense use combinations of the above discussed simplifying assumptions, to give a bird’s eye view over all the recently proposed approaches, we categorize them in Table 2.2.1 depending on the simplifying assumptions they make. For example, Table 2.2.1 indicates that RWalk [Chaudhry et al., 2018] is an approach designed for a CL formulation that is offline, class-incremental, and assumes sharp task boundaries. Algorithmically, it is regularization based and uses memory. Note, one can potentially modify these approaches to apply to other settings as well. For example, the same RWalk can also be used without memory, or can be applied on task-incremental offline formulation. However, we focus on the formulation these methods were originally designed for. We now discuss some high-level problems associated with the simplifying assumptions these approaches make.

Most models, metrics, classifiers, and samplers for CL inherently encode disjoint task (or sharp task boundary) assumption into their design, hence fail to generalize even with slight deviation from the this formulation. Similarly, popular metrics like forgetting and intransigence [Lopez-Paz and Ranzato, 2017, Chaudhry et al., 2018] are designed with this specific formulation encoded in their formal definition, and break with simple modifications like blurry boundaries (class-based, instead of sample-based, definitions of forgetting would appear as classes mix because of blurred boundaries).

Moving to TI-CL v/s CI-CL, these are two extreme cases where CI-CL (single-head) faces scaling issues as there is no restriction on the size of $|\lim_{t \rightarrow \infty} \mathcal{Y}_t|$, and TI-CL (multi-head) imposes a fixed, coherent two-level hierarchy among classes with oracle labels. This formulation is unrealistic in the sense that it does not allow dynamic contexts [Ren et al., 2020].

Lastly, Offline CL v/s Online CL is normally defined depending on whether an algorithm is allowed to revisit a sample repeatedly (unless it is in the memory) during the training process or not. The intention here is to make the continual learning algorithm fast enough so that it can learn *quickly* from a single (or few) sample without having the need of revisiting it. This distinction makes sense if we imagine a data stream spitting samples very fast, then the learner has to adapt itself very quickly. Therefore, the algorithm must provide an acceptable trade-off between space (number of samples to store) and time (training complexity) budgets. However, because of the

Algorithm 1 Greedy Balancing Sampler

```
1: Init: counter  $C_0=\{\}$ ,  $\mathcal{D}_0=\{\}$  with capacity  $k$ . Online samples arrive from  $t=1$ 
2:
3: function SAMPLE( $x_t, y_t, \mathcal{D}_{t-1}, \mathcal{Y}_{t-1}$ ) ▷ Input: New sample and past state
4:    $k_c = \frac{k}{|\mathcal{Y}_{t-1}|}$ 
5:   if  $y_t \notin \mathcal{Y}_{t-1}$  or  $C_{t-1}[y_t] < k_c$  then
6:     if  $\sum_i C_i \geq k$  then ▷ If memory is full, replace
7:        $y_r = \operatorname{argmax}(C_{t-1})$  ▷ Select largest class, break ties randomly
8:        $(x_i, y_i) = \mathcal{D}_{t-1}.\operatorname{random}(y_r)$  ▷ Select random sample from class  $y_r$ 
9:        $\mathcal{D}_t = (\mathcal{D}_{t-1} - (x_i, y_i)) \cup (x_t, y_t)$ 
10:       $C_t[y_r] = C_{t-1}[y_r] - 1$ 
11:     else ▷ If memory has space, add
12:        $\mathcal{D}_t = \mathcal{D}_{t-1} \cup (x_t, y_t)$ 
13:     end if
14:      $\mathcal{Y}_t = \mathcal{Y}_{t-1} \cup y_t$ 
15:      $C_t[y_t] = C_{t-1}[y_t] + 1$ 
16:   end if
17:   return  $\mathcal{D}_t$ 
18: end function
```

lack of proper definition and evaluation schemes, there are algorithms doing very well on one end (use a sample only once), however, performing very poorly on the other end (very expensive learning process). For example, GEM [Lopez-Paz and Ranzato, 2017], a widely known online CL algorithm, uses a sample only once, however, solves a quadratic program for parameter updates which is very time consuming. Therefore, without proper metrics or procedures to quantify how well various CL algorithms balance both space and time complexities, categorizing them into offline vs online might lead to wrong conclusions.

2.3 Greedy Sampler and Dumb Learner (GDumb)

We now propose a simple approach that does not put any restrictions, as discussed above, over the growing nature of the label space, task boundaries, online vs offline, and the ordering of the samples in which the data-stream provides them. Thus, can easily be applied to all the CL formulations discussed in Table 2.2.1. The only requirement is to be allowed to store some episodic memories. We emphasize that we do not claim that our approach solves the general CL problem. Rather, we experimentally show that our simple approach, that does not encode anything specific to the challenging CL problem at hand, is surprisingly effective compared to other approaches

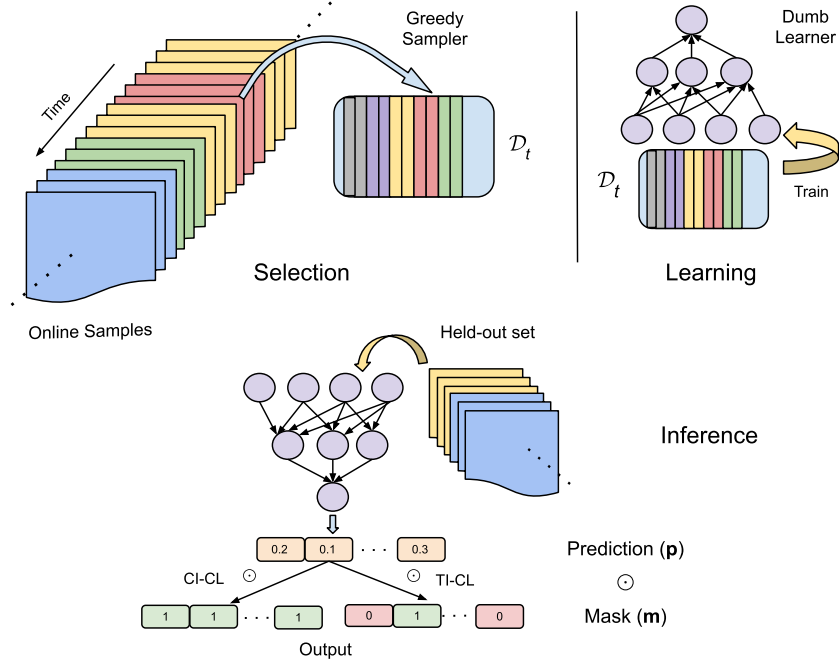


Figure 2.2.1: Our approach (GDumb): The sampler greedily stores samples while balancing the classes. When asked, the learner trains a network from scratch on memory $\mathcal{D}_t$ provided by the sampler. If a mask $\mathbf{m}$ is given at inference, GDumb classifies on the subset of labels provided by the mask. Depending on the mask, GDumb’s inference can vary between two extremes: CI (class-incremental) and TI (task-incremental) formulations.

over all the formulations discussed previously, and also exposes important shortcomings with recent formulations and algorithms.

As illustrated in Figure 2.2.1, our approach comprises of two key components: a *greedy balancing sampler* and a *learner*. Given a memory budget, say k samples, the sampler greedily stores samples from the data-stream (max k samples) with the constraint to asymptotically balance class distribution (Algorithm 1). It is greedy in the sense that whenever it encounters a new class, the sampler simply creates a new bucket for that class and starts removing samples from the old ones, in particular, from the one with a maximum number of samples. Any tie is broken randomly, and a sample is also removed randomly assuming that each sample is equally important. Note, this sampler does not rely on task boundaries or any information about the number of samples in each class.

Let the set of samples greedily stored by the sampler in the memory at any instant in time be $\mathcal{D}_t$ (a dataset with $\leq k$ samples). Then, the objective of the learner, a deep neural network in our experiments, is to learn a mapping $f_{\theta_t} : \mathbf{x} \rightarrow \mathbf{y}$, where $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_t$. This way, using a small dataset that the sampler has stored, the learner learns to classify all the labels seen until time t . Let $\mathcal{Y}_t$

Form.	Designed in	Model (Dataset)	memory (k)	Metric	CI-CL	Online	Disjoint
A1	[Aljundi et al., 2019a]	MLP-400 (MNIST); ResNet18 (CIFAR10)	300, 500; 200, 500, 1000	Acc. (at end)			
A2	[Jin et al., 2021]	MLP-400 (MNIST); ResNet18 (CIFAR10)	500; 500	Acc. (at end)	✓	✓	✓
A3	[Ji et al., 2020]	MLP-400 (MNIST); ResNet18 (CIFAR10)	500; 1000	Acc. (at end)			
B1	[Hsu et al., 2018]; [Rajasegaran et al., 2019]	MLP-400 (MNIST); ResNet18 (SVHN)	4400	Acc. (at end)			
B2	[Rebuffi et al., 2017b]	ResNet32 (CIFAR100)	2000	Acc. (avg in t)	✓	×	✓
B3	[Douillard et al., 2020]	ResNet32 (CIFAR100); ResNet18 (ImageNet100)	1000-2000 (+20) x50	Acc. (avg in t)			
C1	[Hsu et al., 2018]	MLP-400 (MNIST)	4400	Acc. (at end)	×	×	✓
C2	[De Lange et al., 2020]	Many (TinyImageNet)	4500,9000	Acc. (at end)			
D	[Chaudhry et al., 2019a]	ResNet-18-S (CIFAR10)	0-1105 (+65) x17	Acc. (at end)	×	✓	✓
E	[Aljundi et al., 2019b]	MLP-100 (MNIST); ResNet-18 (CIFAR10)	300; 500	Acc. (at end)	✓	✓	×

Table 2.3.1: Various CL formulations we considered in this chapter to evaluate GDumb. These formulations differ in terms of simplifying assumptions (refer Table 2.2.1) and also in terms of resources used. We ensure that selected benchmarks are diverse, covering all popular categorizations. Note, in B3 and D, memory is not constant– it increases over tasks uniformly by (+size) for x tasks times.

represents the set of labels in $\mathcal{D}_t$. Then, at inference, given a sample $\mathbf{x}$, the prediction is made as

$$\hat{\mathbf{y}} = \arg \max \mathbf{p} \odot \mathbf{m}, \quad (2.1)$$

where, $\mathbf{p}$ is the softmax probabilities over all the classes in $\mathcal{Y}_t$, $\mathbf{m} \in \{0, 1\}^{|\mathcal{Y}_t|}$ is a user-defined mask, and $\odot$ denotes the Hadamard product. Note, our prediction procedure allows us to mask any combination of labels at inference. When $\mathbf{m}$ consists of all ones, the inference is exactly the same as that of *single-head* or class-incremental, and when the masking is done depending on the subset of classes in a particular task, it is exactly the same as *multi-head* or task-incremental. Since our sampler does not put any restrictions on the flow of the samples from the data-stream, and our learner does not require any task-boundaries, our overall approach puts minimal restrictions on the general continual learning problem. We would also like to emphasize that we do not use the class $\bar{\mathbf{y}}$ as discussed in our general formulation in Section 2.2, we leave that for future work. However, our objective does encapsulate all the recently proposed CL formulations with minimal possible assumptions, allowing us to provide a fair comparison.

Method k	MNIST	
	(300)	(500)
MLP-100		
FSS-Clust [Aljundi et al., 2019b]	75.8 ± 1.7	83.4 ± 2.6
GSS-Clust [Aljundi et al., 2019b]	75.7 ± 2.2	83.9 ± 1.6
GSS-IQP [Aljundi et al., 2019b]	75.9 ± 2.5	84.1 ± 2.4
GSS-Greedy [Aljundi et al., 2019b]	82.6 ± 2.9	84.8 ± 1.8
GDumb (Ours)	88.9 ± 0.6	90.0 ± 0.4
MLP-400		
GEN [Riemer et al., 2019]	-	75.5 ± 1.3
GEN-MIR [Aljundi et al., 2019a]	-	81.6 ± 0.9
ER [Rolnick et al., 2019]	-	82.1 ± 1.5
GEM [Lopez-Paz and Ranzato, 2017]	-	86.3 ± 1.4
ER-MIR [Aljundi et al., 2019a]	-	87.6 ± 0.7
GDumb (Ours)	-	91.9 ± 0.5

(A1)

Method k	CIFAR10		
	(200)	(500)	(1000)
GEM [Lopez-Paz and Ranzato, 2017]	16.8 ± 1.1	17.1 ± 1.0	17.5 ± 1.6
iCARL [Rebuffi et al., 2017b]	28.6 ± 1.2	33.7 ± 1.6	32.4 ± 2.1
ER [Rolnick et al., 2019]	27.5 ± 1.2	33.1 ± 1.7	41.3 ± 1.9
ER-MIR [Aljundi et al., 2019a]	29.8 ± 1.1	40.0 ± 1.1	47.6 ± 1.1
ER5 [Aljundi et al., 2019a]	-	-	42.4 ± 1.1
ER-MIR5 [Aljundi et al., 2019a]	-	-	49.3 ± 0.1
GDumb (Ours)	35.0 ± 0.6	45.8 ± 0.9	61.3 ± 1.7

(A1)

Table 2.4.1: (CI-Online-Disjoint) Performance on formulation A1.

2.4 Experiments

We now compare GDumb with various existing algorithms for several recently proposed CL formulations. As shown in Table 2.2.1, there broadly are five such formulations $\{A, B, \dots, E\}$. Since even within a formulation there can be sub-categories depending on the resources used, we further enumerate them and present a more detailed categorization, keeping fair comparisons in mind, in Table 2.3.1. For example, B1 and B2 belong to the same formulation B, however, they differ in terms of architectures, datasets, and memory sizes used in their respective papers. Therefore, in total, we pick 10 different formulations, most of them having multiple architectures and datasets (refer Appendix B for details).

Implementation details: GDumb uses the same fixed training settings, with *no* hyperparameter tuning whatsoever, in all the CL formulations. This is possible because of the fact that GDumb does not impose any simplifying assumptions. All results measure accuracy (fraction of correctly classifications) evaluated on the held-out test set. For all the formulations, GDumb uses an SGD optimizer, fixed batch size of 16, learning rates $[0.05, 0.0005]$, an SGDR [Loshchilov and Hutter, 2017] schedule with $T_0 = 1$, $T_{mult} = 2$ and warm start of 1 epoch. Early stopping with patience of 1 cycle of SGDR, along with standard data augmentation is used. GDumb uses cutmix [Yun et al., 2019] with $p=0.5$ and $\alpha=1.0$ for regularization on all datasets except MNIST. The training set-up comprises of an Intel i7 4790, 32GB RAM and a single GTX 1070 GPU. All results are averaged over 3 random seeds, each with different class-to-task assignments. In formulations B2 and B3, we strictly follow class order specified in iCARL [Rebuffi et al.,

2017b] and PODNet [Douillard et al., 2020]. Our pytorch implementation is publicly available at: <https://github.com/drimpossible/GDumb>.

2.4.1 Results and Discussions

Class Incremental Online CL with Disjoint Tasks (Form. A):

The first sub-category under this formulation is A1, which follows exactly the same setting, on Split-MNIST and Split-CIFAR10, as presented in MIR [Aljundi et al., 2019a]. Results are shown in Table 2.4.1. We observe that on both MNIST and CIFAR10 for all choices of k , GDumb outperforms all the approaches by a large margin. For example, in the case of MNIST with memory size $k = 500$, GDumb outperforms ER-MIR [Aljundi et al., 2019a] by around 4.3%. Similarly, on CIFAR10 with memory sizes of 200, 500, and 1K, our approach outperforms current approaches by nearly 5%, 6% and 11%, respectively, convincingly achieving state-of-the-art results. Note, increasing the memory size from 200 to 1K in CIFAR10 increases the performance of GDumb by 26.3% (expected as GDumb is trained only on memory), whereas, this increase is only 18% in the case of ER-MIR [Aljundi et al., 2019a]. Similar or even much worst improvements are noticed in other recent approaches, suggesting they might not be utilizing the memory samples efficiently.

We now benchmark our approach on Split-MNIST and Split-CIFAR10 as detailed by parallel works GMED [Jin et al., 2021] (sub-category A2) and ARM [Ji et al., 2020] (sub-category A3). We present results in Table 2.4.2 and show that GDumb outperforms parallel works like HAL [Chaudhry et al., 2021], QMED [Jin et al., 2021], ARM [Ji et al., 2020] in addition to outperforming recent works GSS [Aljundi et al., 2019b], MIR [Aljundi et al., 2019a], and ADI [Yin et al., 2020], consistently across datasets. It outperforms the best alternatives in QMED [Jin et al., 2021] by over 4% and 10% on MNIST and CIFAR10, respectively, and in ARM [Ji et al., 2020] by over 5% and 13% on MNIST and CIFAR10 datasets, respectively. Results from ARM [Ji et al., 2020] indicate—(i) GDumb consistently outperforms other experience replay approaches and (ii) experience replay methods obtain much better performance than generative replay with much smaller memory footprint.

Class Incremental Offline CL with Disjoint Tasks (Form. B):

We proceed next to offline CI-CL formulations. We first compare our proposed approach with 12 popular methods on sub-category B1. Results are presented in Table 2.4.3 (left). Our approach outperforms all memory-based methods like GEM, RtF, DGR by over 5% on MNIST. We outperform the recent RPS-Net [Rajasegaran et al., 2019] and OvA-INN [Hocquet et al., 2020] by over 1%,

Method k	MNIST (500)	CIFAR-10 (500)	Method	MNIST		CIFAR10	
				Memory	Accuracy	Memory	Accuracy
Fine tuning	18.8 $\pm$ 0.6	18.5 $\pm$ 0.2	Finetune	0	18.8 $\pm$ 0.5	0	15.0 $\pm$ 3.1
AGEM [Chaudhry et al., 2019a]	29.0 $\pm$ 5.3	18.5 $\pm$ 0.6	GEN [Shin et al., 2017]	4.58	79.3 $\pm$ 0.6	34.5	15.3 $\pm$ 0.5
BGD [Zeno et al., 2018]	13.5 $\pm$ 5.1	18.2 $\pm$ 0.5	GEN-MIR [Aljundi et al., 2019a]	4.31	82.1 $\pm$ 0.3	38.0	15.3 $\pm$ 1.2
GEM [Lopez-Paz and Ranzato, 2017]	87.2 $\pm$ 1.3	20.1 $\pm$ 1.4	LwF [Li et al., 2017a]	1.91	33.3 $\pm$ 2.5	4.38	19.2 $\pm$ 0.3
GSS-Greedy [Aljundi et al., 2019b]	84.2 $\pm$ 2.6	28.0 $\pm$ 1.3	ADI [Yin et al., 2020]	1.91	55.4 $\pm$ 2.6	4.38	24.8 $\pm$ 0.9
HAL [Chaudhry et al., 2021]	77.9 $\pm$ 4.2	32.1 $\pm$ 1.5	ARM [Ji et al., 2020]	1.91	56.2 $\pm$ 3.5	4.38	26.4 $\pm$ 1.2
ER [Rolnick et al., 2019]	81.0 $\pm$ 2.3	33.3 $\pm$ 1.5	ER [Rolnick et al., 2019]	0.39	83.2 $\pm$ 1.9	3.07	41.3 $\pm$ 1.9
MIR [Aljundi et al., 2019a]	84.9 $\pm$ 1.7	34.5 $\pm$ 2.0	ER-MIR [Aljundi et al., 2019a]	0.39	85.6 $\pm$ 2.0	3.07	47.6 $\pm$ 1.1
GMED (ER) [Jin et al., 2021]	82.7 $\pm$ 2.1	35.0 $\pm$ 1.5	iCarL [Rebuffi et al., 2017b] (5 iter)	-	-	3.07	32.4 $\pm$ 2.1
GMED (MIR) [Jin et al., 2021]	87.9 $\pm$ 1.1	35.5 $\pm$ 1.9	GEM [Lopez-Paz and Ranzato, 2017]	0.39	86.3 $\pm$ 0.1	3.07	17.5 $\pm$ 1.6
GDumb (Ours)	91.9 $\pm$ 0.5	45.8 $\pm$ 0.9	GDumb (ours)	0.39	91.9 $\pm$ 0.5	3.07	61.3 $\pm$ 1.7

(A2)

(A3)

Table 2.4.2: (CI-Online-Disjoint) Performance on formulations A2 (left) and A3 (right).

and are as good as iTAML [Rajasegaran et al., 2020], on MNIST. On SVHN, we outperform recently proposed methods like RPS-Net by 4.5% and far exceeding methods like GEM. Note, we achieve the same accuracy as the best offline CL method iTAML [Rajasegaran et al., 2020] despite using an extremely simple approach in online fashion.

We now discuss two very interesting sub-categories B2 (as in iCARL [Rebuffi et al., 2017b]) and B3 (from a very recent work PODNet [Douillard et al., 2020]). The primary difference between B2 and B3 merely lies in the number of classes per task. However, as will be seen, this minor difference changes the complexity of the problem dramatically. In the case of B2, CIFAR100 is divided into 20 tasks, whereas, B3 starts with a network trained on 50 classes and then learns one class per task incrementally (leading to 50 new tasks). Performance of GDumb on B2 and B3 formulations are shown in Table 2.4.3 (center) and Table 2.4.3 (right), respectively. An interesting observation here is that GDumb which performed nearly 20% worse than BiC and iCARL in B2, performs over 10-15% better than BiC, UCIR and iCARL in B3. This drastic shift against previous results might suggest that having higher number of classes per task and less number of tasks might give added advantage to scaling/bias correction type approaches, which otherwise would quickly deteriorate over greater timesteps. Furthermore, we note that our simple baseline narrowly outperforms PODNet (CNN) on both CIFAR100 and ImageNet100 datasets.

Task Incremental Offline CL with Disjoint Tasks (Form. C):

We now proceed to compare the performance of GDumb in task incremental formulation. Recall, GDumb does not put any restrictions such as task vs class incremental, or online vs offline. However,

Method	MNIST	SVHN
No memory		
MAS [Aljundi et al., 2018]	19.5 ± 0.3	17.3
SI [Zenke et al., 2017]	19.7 ± 0.1	17.3
EWC [Kirkpatrick et al., 2017]	19.8 ± 0.1	18.2
Online EWC [Schwarz et al., 2018]	19.8 ± 0.04	18.5
LwF [Li et al., 2017a]	24.2 ± 0.3	-
OvA-INN [Hocquet et al., 2020]	96.4	-
$k=4400$		
DGR [Shin et al., 2017]	91.2 ± 0.3	-
DGR+Distill	91.8 ± 0.3	-
GEM [Lopez-Paz and Ranzato, 2017]	92.2 ± 0.1	75.6
RtF [van de Ven and Tolias, 2018]	92.6 ± 0.2	-
RPS-Net [Rajasegaran et al., 2019]	96.2	88.9
iTAML [Rajasegaran et al., 2020]	98.72	96.18
GDumb (Ours)	97.8 ± 0.2	93.4 ± 0.4

(B1)

CIFAR100		
Method	Acc. (Avg)	Acc. (last)
No memory		
Finetune	17.8 ± 0.72	5.9 ± 0.15
SI [Zenke et al., 2017]	23.6 ± 1.90	13.3 ± 1.14
MAS [Aljundi et al., 2018]	24.7 ± 1.76	10.4 ± 0.80
EWC [Kirkpatrick et al., 2017]	25.4 ± 1.99	9.5 ± 0.83
RWALK [Chaudhry et al., 2018]	25.6 ± 1.92	11.1 ± 2.14
LwF [Li et al., 2017a]	32.3 ± 1.92	14.1 ± 0.87
DMC [Zhang et al., 2020b]	45.0 ± 1.96	23.8 ± 1.90
$k=2000$		
GDumb (Ours)	45.2 ± 1.70	24.1 ± 0.97
DMC++ [Zhang et al., 2020b]	56.8 ± 0.86	-
iCARL [Rebuffi et al., 2017b]	58.8 ± 1.90	42.9 ± 0.79
EEIL [Castro et al., 2018]	63.4 ± 1.6	-
BiC [Wu et al., 2019]	63.8	46.9

(B2)

Method	CIFAR100	ImageNet100
iCaRL [Rebuffi et al., 2017b]	44.2 ± 1.0	54.97
BiC [Wu et al., 2019]	47.1 ± 1.5	46.49
UCIR (NME) [Hou et al., 2019]	48.6 ± 0.4	55.44
UCIR (CNN) [Hou et al., 2019]	49.3 ± 0.3	57.25
PODNet (CNN) [Douillard et al., 2020]	58.0 ± 0.5	62.08
GDumb (CNN)	58.4 ± 0.8	62.86
PODNet (NME) [Douillard et al., 2020]	61.4 ± 0.7	-

(B3)

Table 2.4.3: (CI-Offline-Disjoint) Performance on B1, B2, and B3.

Method		Parameters	Regularization	Accuracy	
No stored samples					
Method (k)	MNIST (4400)	mean-IMM [Lee et al., 2017]	3.5M	none	32.42
		mode-IMM [Lee et al., 2017]	9.0M	dropout	42.41
		SI [Zenke et al., 2017]	3.5M/9.0M	L2/dropout	43.74
		HAT [Serra et al., 2018]	3.5M/9.0M	L2	44.19
		EWC [Kirkpatrick et al., 2017]	613K	none	45.13
		LwF [Li et al., 2017a]	9.0M	L2	48.11
		EBLL [Rannen et al., 2017]	9.0M	L2	48.17
		MAS [Aljundi et al., 2018]	3.5M/9.0M	none	48.98
		PackNet [Mallya and Lazebnik, 2018]	613K/3.5M	L2/dropout	55.96
			k=4500		
		GEM [Lopez-Paz and Ranzato, 2017]	613K/3.5M	none/dropout	44.23
		GDumb	834K	cutmix	45.50
		iCARL [Rebuffi et al., 2017b]	613K/3.5M	dropout	48.55
k=9000					
		GEM [Lopez-Paz and Ranzato, 2017]	613K/3.5M	none/dropout	45.27
		iCARL [Rebuffi et al., 2017b]	613K/3.5M	dropout	49.94
		GDumb	834K	cutmix	57.27

(C1)

Method	CIFAR100 (1105)
RWalk [Chaudhry et al., 2018]	40.9 ± 3.97
EWC [Kirkpatrick et al., 2017]	42.4 ± 3.02
Base	42.9 ± 2.07
MAS [Aljundi et al., 2018]	44.2 ± 2.39
SI [Zenke et al., 2017]	47.1 ± 4.41
iCARL [Rebuffi et al., 2017b]	50.1
S-GEM [Chaudhry et al., 2019a]	56.2
PNN [Rusu et al., 2016]	59.2 ± 0.85
GEM [Lopez-Paz and Ranzato, 2017]	61.2 ± 0.78
A-GEM [Chaudhry et al., 2019a]	63.1 ± 1.24
TinyER [Chaudhry et al., 2019b]	68.5 ± 0.65
GDumb	60.3 ± 0.85

(D)

(C2)

Table 2.4.4: (TI-Offline-Disjoint) Performance on C1 (left) and C2 (middle). (TI-Online-Disjoint) Performance on D (right).

in the case of GDumb, we use masking (subset of labels in a task) over softmax probabilities at test time to mimic TI-CL formulation.

Table 2.4.4 (left) shows the results on C1, a very widely used and most popular offline TI-CL (or multi-head) formulation for CL on Split-MNIST. We now move to C2 on Split-TinyImagenet (offline TI-CL formulation as in [De Lange et al., 2020]). Note, in this particular formulation we used a different architecture called DenseNet-100-BC [Huang et al., 2017]. Note that the methods presented here use different architectures, and hence we also chose a standard efficient architecture. However, we provide an architectural ablation across these architectures in Appendix A. Results are presented in Table 2.4.4 (middle). We observe that for $k = 9000$, we outperform

Method	MNIST	CIFAR10	Method	Train time	Memory (Train)	Memory (Test)
Reservoir [Riemer et al., 2019]	69.12	-	Base	105s	$P + B^*H$	$P + B^*H$
GSS-Clust [Aljundi et al., 2019b]	-	25.0	EWC [Kirkpatrick et al., 2017]	250s	$4^*P + B^*H$	$P + B^*H$
FSS-Clust [Aljundi et al., 2019b]	-	26.0	PNN [Rusu et al., 2016]	409s	$2^*P^*T + B^*H^*T$	$2^*P^*T + B^*H^*T$
GSS-IQP [Aljundi et al., 2019b]	76.49	29.6	GEM [Lopez-Paz and Ranzato, 2017]	5238s	$P^*T + (B+M)^*H$	$P + B^*H$
GSS-Greedy [Aljundi et al., 2019b]	77.96	29.6	A-GEM [Chaudhry et al., 2019a]	449s	$2^*P + (B+M)^*H$	$P + B^*H$
GDumb (Ours)	88.93	45.8	GDumb (Ours)	60s	$P + M^*H$	$P + B^*H$
(E)			(Resources)			

Table 2.4.5: (CI-Online-Joint) Performance on E (left). Note, this is particularly challenging as the tasks here are non-disjoint (blurry task boundary) with class-imbalance. On the (right), we benchmark resource consumption in terms of training time and memory usage. Memory cost is provided in terms of the total parameters P , the size of the minibatch B , the total size of the network hidden state H (assuming all methods use the same architecture), the size of the episodic memory M per task. GDumb, at the very least, is 7.5x times faster than the existing efficient CL formulations.

all 10 approaches including GEM and iCARL by margins of atleast 7%. When the memory is halved to $k = 4500$, we perform slightly better than GEM and nearly 3% worse than iCARL. Since we used different architecture, we do not claim that we would notice similar improvements had we trained GDumb using the networks used in respective papers. However, these results are still encouraging as the approaches we compare against are trained in TI-CL manner and GDumb is always trained in CI-CL manner (much more difficult).

Task Incremental Online CL with Disjoint Tasks (Form. D):

We now compare GDumb with 12 TI-CL tuned online approaches with small memory (not a favourable setting for GDumb as it relies totally on the samples in the memory) and detail the results in Table 2.4.4 (right). We observe that GDumb outperforms 8 out of 11 approaches even though it is trained in CI-CL manner.

Class Incremental Online CL with Joint Tasks (Form. E):

We now measure impact of imbalanced data stream with blurry task boundaries [Aljundi et al., 2019b]. Results are presented in Table 2.4.5 (left). We outperform competing models by over 10% and 16%, overwhelming surpassing complicated methods attuned to this benchmark. This demonstrates that GDumb works well even when almost all simplifying assumptions are removed.

2.4.2 Resources Needed

It is important that our approach is in the ballpark of online continual learning constraints of memory and compute usage to achieve its performance. We benchmark our resource consumption against the efficient CL algorithms in Table 2.4.5 (right) benchmarked with a V100 GPU on formulation E [Chaudhry et al., 2019a]. We observe that we require only 60s on a slower GTX 1070 GPU (and 350s on a 4790 i7 CPU), performing several times efficient than various recently proposed algorithm. Note that sampling time is negligible, while testing time is not included in the above.

2.4.3 Potential Future Extensions

Active Sampling: Given an importance value $v_t \in \mathbb{R}^+$ (by active learner) along with sample (x_t, y_t) at time t , we can extend our sampler by having the objective of storing most important samples (maximizing $\sum_{i=1}^{|\mathcal{D}_c|} v_i$) for any given class c in its storage of size k . This will allow an algorithm to reject less important samples. Of course, it is not clear how to learn to quantify *importance* of a sample.

Dynamic Probabilistic Masking: It is possible to extend masking in GDumb beyond CI-CL and TI-CL to dynamic task hierarchies across video/scene types useful in recently proposed settings [Ren et al., 2020]. Since GDumb applies a mask (given a context) only at inference, we can dynamically adapt to the context to generate a mask. Similarly, we can extend GDumb beyond deterministic oracles ($m_i \in \{0, 1\}$) to probabilistic one ($m_i \in [0, 1]$). This delivers a lot of flexibility for diverse extensions like cost-sensitive classification, class-imbalance among others.

2.5 Conclusion

In this chapter, we provided a general view of a continual image classification problem. We then proposed a simple and general approach with minimal restrictions and empirically showed that it outperforms almost all the complicated state-of-the-art approaches *in their own formulations* for which they were specifically designed. We hope that our approach serves the purpose of a strong baseline to benchmark the effectiveness of any newly proposed CL algorithm. Our solution also raises a major concern to be investigated: Even though there are plenty of research articles focused on specific scenarios relating CL problem, are we really progressing in the right direction? We say no, as we focus on the wrong constraints. We shall further explore alternative directions adopted and issues with them in the next chapter.

3

RanDumb: A Simple Approach that Questions the Efficacy of Continual Representation Learning

Contents

3.1	Introduction	27
3.1.1	RanDumb: Construction and Performance	28
3.2	RanDumb: Mechanism & Intuitions	30
3.3	Experiments	31
3.3.1	Results	35
3.3.2	RanDumb: Analysis	38
3.4	Discussion and Concluding Remarks	40

Abstract

We propose Randumb to examine the efficacy of continual representation learning. Randumb embeds raw pixels using a fixed random transform which approximates an RBF-Kernel, initialized before seeing any data, and learns a simple linear classifier on top. We present a surprising and consistent finding: Randumb significantly outperforms the continually learned representations using deep networks across numerous continual learning benchmarks, demonstrating the poor performance of representation learning in these scenarios. Randumb stores no exemplars and performs a single pass over the data, processing one sample at a time. It complements GDumb [Prabhu et al., 2020], operating in a low-exemplar regime where GDumb has especially poor performance. We reach the same consistent conclusions when Randumb is extended to scenarios with pretrained models replacing the random transform with pretrained feature extractor. Our investigation is both surprising and alarming as it questions our understanding of how to effectively design and train models that require efficient continual representation learning, and necessitates a principled reinvestigation of the widely explored problem formulation itself. Our code is available [here](#).

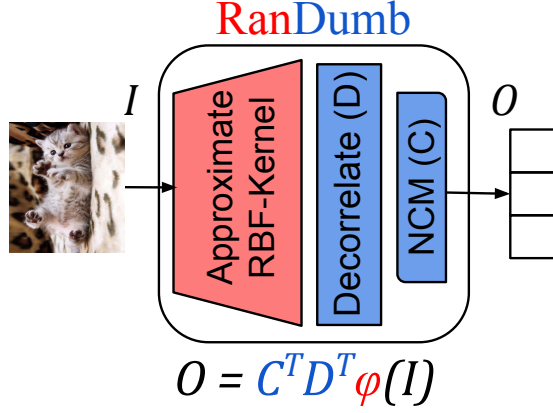


Figure 3.1.1: Mechanism of RanDumb. Project raw pixels to a high dimensional space using random fourier projections (φ), then decorrelate the features using Mahalanobis distance McLachlan [1999] and classify with the nearest class mean. The online update only involves updating a single covariance matrix and class-means.

3.1 Introduction

Building on the foundations of supervised deep learning, the prevalent approach in continual learning has been to jointly train representations alongside classifiers. This approach simply follows from the assumption that learned representations are expected to outperform fixed representation functions such as kernel classifiers, as demonstrated in supervised deep learning. However, this assumption is never validated in continual learning, with scenarios having limited updates where networks might not be trained until convergence, such as online continual learning (OCL).

In this chapter, we study the efficacy of representations derived from continual learning algorithms. Surprisingly, our findings suggest that these representations might not be as beneficial as presumed. To test this, we introduce a simple baseline method named RanDumb, which combines a random representation function with a straightforward linear classifier, illustrated in detail in Figure 3.1.1. Our empirical evaluations, summarized in Table 3.1.1 (left, top), reveal that despite replacing the representation learning with a pre-defined random representation, RanDumb surpasses current state-of-the-art methods in latest online continual learning benchmarks Zajac et al. [2024].

We further expand our evaluations to scenarios incorporating methods that use pre-trained feature extractors Wang et al. [2022c]. By substituting our random projections with these feature extractors and retaining the linear classifier, RanDumb again outperforms leading methods as shown in Table 3.1.2 (right, top).

Table 3.1.1: Online Continual Learning. Performance comparison of RanDumb on the PEC setup [Zajac et al., 2024] and VAE-GC [Van De Ven et al., 2021]. Setup and numbers borrowed from PEC [Zajac et al., 2024]. RanDumb outperforms the best exemplar-free OCL method. Comparing representation learning with no representation learning, *a fixed, random function outperforms training a deep network continually, i.e. $RepGain(\mathcal{D}) < 0$* . Furthermore, RanDumb nearly matches performance of one-pass joint, demonstrating the inefficacy of current benchmarks.

Method	MNIST	CIFAR10	CIFAR100	m-ImNet
Comparison with Best Method				
Best (PEC)	92.3	58.9	26.5	14.9
RanDumb (Ours)	98.3	55.6	28.6	17.7
Improvement	+6.0	-3.3	+2.1	+2.8
Only Ablating Representation Learning: Random v/s Deep Embedding				
VAE-GC	84.0	42.7	19.7	12.1
RanDumb (Ours)	98.3	55.6	28.6	17.7
Improvement	+14.3	+12.9	+8.9	+5.6
Scope of Improvement				
Joint (One Pass)	98.3	74.2	33.0	25.3
RanDumb (Ours)	98.3	55.6	28.6	17.7
Gap Covered. (%)	100%	75%	87%	70%

3.1.1 RanDumb: Construction and Performance

Mechanism. Our mechanism is illustrated in Figure 3.1.1. RanDumb first projects input pixels into a high-dimensional space using a fixed kernel based on random Fourier basis, which is a low-rank data-independent approximation of the RBF Kernel Rahimi and Recht [2007]. Then, we use a simple linear classifier which first normalizes distances across different feature dimensions (anisotropy) with Mahalanobis distance McLachlan [1999] and then uses nearest class means for classification Mensink et al. [2013]. In scenarios with pretrained feature extractors, we use the fixed pretrained model as embedder and learn a linear classifier over it, similar to Hayes and Kanan [2020].

Properties. RanDumb needs no storage of exemplars and requires only one pass over the data in a one-sample-per-timestep fashion. Furthermore, it only requires online estimation of the covariance matrix and nearest class mean, hence is invariant to data order, representing the accuracy in the worst-case ordering.

Poor Representation Learning. We first compare RanDumb with VAE-GC [Van De Ven et al., 2021] in Table 3.1.1 (middle) and SLCA Zhang et al. [2023a] in Table 3.1.2 (middle). The only distinction between them is their embedding functions: RanDumb uses a fixed function (random/pretrained network) for the embedder, whereas VAE-GC and SLCA further continually trained deep networks, using them as the embedder. Astonishingly, RanDumb consistently surpasses

Table 3.1.2: Offline Continual Learning. Performance comparison with ImageNet21K ViT-B16 model using 2 init classes and 1 new class per task. RanDumb here assumes the features as input pixels. SLCA Zhang et al. [2023a] and RanPAC McDonnell et al. [2023] results reproduced from original codebase. Comparing representation learning with no representation learning, *a fixed, pretrained feature extractor outperforms further finetuning a deep network continually, i.e. $RepGain(\mathcal{D}) < 0$* . RanDumb nearly matches performance of joint, demonstrating the inefficacy of current benchmarks. *here representation learning collapses, accuracy reported without PETL. +here representation learning and random projection both collapse, reporting last stable accuracy.

Method	CIFAR	IN-A	IN-R	CUB	OB	VTAB	Cars
Comparison with Best Method							
Best (RanPAC)	89.6	26.8 ⁺	67.3	87.2 ⁺	77.9*	88.2 ⁺	53.7*
RanDumb (Ours)	86.8	42.2	64.9	88.5	75.3	92.3	69.1
Improvement	-2.9	+15.4	-2.4	+1.3	-2.6	+3.7	+15.4
Only Ablating Representation Learning: No Finetuning v/s Full Finetuning							
SLCA	86.8	-	54.2	82.1	-	-	18.2
RanDumb (Ours)	86.8	42.2	64.9	88.5	75.3	92.3	69.1
Improvement	+0.0	-	+10.7	+6.4	-	-	+50.9
Scope of Improvement							
Joint	93.8	70.8	86.6	91.1	83.8	95.5	86.9
RanDumb (Ours)	86.8	42.2	64.9	88.5	75.3	92.3	69.1
Gap Covered. (%)	93%	60%	75%	97%	92%	97%	80%

VAE-GC and SLCA baselines by wide margins of 5-15%. This shows that state-of-the-art online continual learning algorithms fail to learn effective representations across standard exemplar-free continual learning benchmarks.

Benchmarks Over-Constrained. The reader might now ask: With better representation learning, can one outperform RanDumb by large margins in these settings? To study this, we compare the performance of RanDumb with Joint in both online and offline settings respectively in Table 3.1.1 (bottom) and Table 3.1.2 (bottom). We observe that our simple baseline RanDumb recovers 70-90% of the gap to the respective joint classifiers across both settings. Improving forgetting in continual representation learning Hess et al. [2023], Mukhoti et al. [2023] offers limited scope for improvement, as current benchmarks might be too constrained to have effective representation learning at all.

In the next sections, we shall first describe our proposed method and then walk through our findings in detail.

3.2 RanDumb: Mechanism & Intuitions

In this section, we thoroughly examine RanDumb, focusing on its two main elements: random projection and the dumb learner. We illustrate the mechanism of RanDumb using three toy examples in Figure 3.2.1.

We first describe the dumb classifier. To classify a test sample $\mathbf{x}_{\text{test}}$, we start with a simple classifier, the nearest class mean (NCM) classifier. It predicts the class among C classes by highest value of the similarity function f among class means μ_i :

$$y_{\text{pred}} = \arg \max_{i \in \{1, \dots, |C|\}} f(\mathbf{x}_{\text{test}}, \mu_i), \text{ where} \\ f(\mathbf{x}_{\text{test}}, \mu_i) := \mathbf{x}_{\text{test}}^\top \mu_i \quad (3.1)$$

and μ_i are the class-means in the pixel space:

$$\mu_i = \frac{1}{|C_i|} \sum_{\mathbf{x}} \mathbf{x} \quad \forall \mathbf{x} \in C_i$$

RanDumb adds two additional components to this classifier: 1) Kernelization and 2) Decorrelation.

① Kernelization: Why & How? Classes are typically not linearly separable in the pixel space, unlike in the feature space of deep models. Hence, we apply the kernel trick to embed the pixels in a better representation space, computing all distances between the data and class-means in this embedding space. This phenomena is illustrated on three toy examples to build intuitions in Figure 3.2.1 (Embed). We use an RBF-Kernel, which for two points $\mathbf{x}$ and $\mathbf{y}$ is defined as: $K_{\text{RBF}}(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \|\mathbf{x} - \mathbf{y}\|^2)$ where γ is a scaling parameter. However, calculating the RBF kernel is not possible due to the online continual learning constraints preventing computation of pairwise-distance between all points. Hence, we use a data-independent approximation, random Fourier projection $\phi(\mathbf{x})$, as given in [Rahimi and Recht, 2007]:

$$K_{\text{RBF}}(\mathbf{x}, \mathbf{y}) \approx \phi(\mathbf{x})^T \phi(\mathbf{y})$$

where the random Fourier features $\phi(\mathbf{x})$ are defined by first sampling D vectors $\{\omega_1, \dots, \omega_D\}$ from a Gaussian distribution with mean zero and covariance matrix $2\gamma\mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Then $\phi(\mathbf{x})$ is a $2D$ -dimensional feature, defined as:

$$\phi(\mathbf{x}) = \frac{1}{\sqrt{D}} \left[\cos(\omega_1^T \mathbf{x}), \sin(\omega_1^T \mathbf{x}), \dots, \cos(\omega_D^T \mathbf{x}), \sin(\omega_D^T \mathbf{x}) \right]$$

We keep these ω bases fixed throughout online learning. Thus, we obtain our modified similarity function from Equation 3.1 as:

$$f(\mathbf{x}_{\text{test}}, \mu_i) := \phi(\mathbf{x}_{\text{test}})^\top \bar{\mu}_i \quad (3.2)$$

where $\bar{\mu}_i$ are the class-means in the kernel space:

$$\bar{\mu}_i = \frac{1}{|C_i|} \sum_{\mathbf{x}} \phi(\mathbf{x}) \quad \forall \mathbf{x} \in C_i$$

② *Decorrelation: Why & How?* Projected raw pixels, similar to features of deep models, have feature dimensions with different variances (anisotropic). Hence, instead of naively computing $\phi(\mathbf{x}_{\text{test}})^\top \bar{\mu}_i$, we further decorrelate the feature dimensions using a Mahalanobis distance with the empirical covariance matrix $\mathbf{S}$. We illustrate this phenomena as well on three toy examples in Figure 3.2.1 (Decorrelate) to build intuitions. Our similarity function finally is:

$$f(\mathbf{x}_{\text{test}}, \mu_i) := (\phi(\mathbf{x}_{\text{test}}) - \bar{\mu}_i)^\top \mathbf{S}^{-1} (\phi(\mathbf{x}_{\text{test}}) - \bar{\mu}_i) \quad (3.3)$$

Online Computation. Our random projection is computed and fixed before seeing any real data. In the continual learning process, we only update the running class mean and empirical covariance matrix, which is accurately estimable online¹.

Note on Equivalences. For the curious reader we draw equivalences between the linear classifier used in this work and related works. These equivalences rely on the assumption that the classes are equiprobable, which is the case for most datasets here. In these cases, nearest class mean classifier with the Mahalanobis distance metric is equivalent to linear discriminant analysis classifier McLachlan [2005]. Hence, one could say RanDumb is equivalent to a Streaming LDA classifier with an approximate RBF Kernel. Alternatively, one could think of the decorrelation operation as explicitly decorrelating the features with ZCA whitening Bell and Sejnowski [1996]. We hope these alternative perspectives allow for a better understanding of the RanDumb classifier.

3.3 Experiments

We compare RanDumb with algorithms across online continual learning benchmarks with an emphasis on exemplar-free and low-exemplar storage regime complementing GDumb Prabhu et al. [2020] in regimes where it has poor performance.

¹Online prediction is possible using the Sherman–Morrison formula to update the inverse of the covariance matrix. However, we only needed to compute inverse at the last timestep for all our experiments, hence we do not experiment with this.

Table 3.2.1: Benchmark (Overview). We evaluate RanDumb across a diverse set of benchmarks that progressively relax constraints compared to RanDumb. Benchmark A closely matches RanDumb with one class per timestep and no stored exemplars. Benchmark B, D, E progressively relax the constraints on exemplars and classes per timestep. Benchmark C and E remove the online constraint by allowing unrestricted training and sample access within a task without exemplar-storage of past tasks. Benchmark F allows using large pretrained models, modified with one class per task, inspired by challenges detailed in Benchmark A.

Setup	Num Passes	#Classes Per Task	#Samples Per Step	#Stored Exemplars	Contrastive Augment
RanDumb (Ours)	1	1	1	0	No
A (Zajac et al. [2024])	1	1	10	0	No
B1 (Guo et al. [2022])	1	2	10	100-2000	No
B2 (Guo et al. [2022])	1	2	10	100-1000	Yes
C (Smith et al. [2023b])	Many	10	All	0	No
D (Anonymous [2024])	1	2-10	10	1000	No
E (Ye and Bors [2023])	1	2-5	10	1000-5000	No
F (Wang et al. [2022c])	Many	1	All	0	No

Table 3.2.2: Benchmark A (Ref: Table 1 from PEC Zajac et al. [2024]). We perform comprehensive comparisons of RanDumb with popular online continual learning approaches in a 1-class per task setting referred as ‘Dataset (num_tasks/1)’. This setting is a challenging setting for OCL methods except GDumb. We observe that RanDumb outperforms all approaches across all datasets by 2-6% margins, with an exception of latest work PEC Zajac et al. [2024] on CIFAR10.

Method	Memory	MNIST (10/1)	CIFAR-10 (10/1)	CIFAR-100 (100/1)	miniImageNet (100/1)	
<i>Fine-tuning</i>	all	10.1± 0.0	10.0± 0.0	1.0± 0.0	1.0± 0.0	
<i>Joint, 1 epoch</i>	all	98.3± 0.0	74.2± 0.1	33.0± 0.2	25.3± 0.2	
Rehearsal-Based	ER Chaudhry et al. [2019b] (ICML-W '19)	500	84.4± 0.3	40.6± 1.1	12.5± 0.3	5.7± 0.2
	A-GEM Chaudhry et al. [2019a] (ICLR '19)	500	59.8± 0.8	10.2± 0.1	1.0± 0.0	1.1± 0.1
	iCaRL Rebuffi et al. [2017b] (CVPR '17)	500	83.1± 0.3	37.8± 0.4	5.7± 0.1	7.5± 0.1
	BiC Wu et al. [2019] (CVPR '19)	500	86.0± 0.4	35.9± 0.4	6.4± 0.3	1.5± 0.1
	ER-ACE Caccia et al. [2022a] (ICLR '22)	500	87.8± 0.2	39.9± 0.5	8.2± 0.2	5.7± 0.2
	DER Buzzega et al. [2020] (NeurIPS '20)	500	91.7± 0.1	40.0± 1.5	1.0± 0.1	1.0± 0.0
	DER++ Buzzega et al. [2020] (NeurIPS '20)	500	91.9± 0.2	35.6± 2.4	6.2± 0.4	1.4± 0.1
	X-DER Boschini et al. [2022] (TPAMI '22)	500	83.0± 0.1	43.2± 0.5	15.6± 0.1	8.2± 0.4
GDumb Prabhu et al. [2020] (ECCV '20)	500	91.0± 0.2	50.7± 0.7	8.2± 0.2	-	
Rehearsal-Free	EWC Kirkpatrick et al. [2017] (PNAS '17)	0	10.1± 0.0	10.6± 0.4	1.0± 0.0	1.0± 0.0
	SI Zenke et al. [2017] (ICML '17)	0	12.7± 1.0	10.1± 0.1	1.1± 0.0	1.0± 0.1
	LwF Li et al. [2017a] (TPAMI '17)	0	11.8 ± 0.6	10.1± 0.1	0.9± 0.0	1.0± 0.0
	LT Zeno et al. [2018] (Arxiv '18)	0	10.9± 0.9	10.0± 0.2	1.1± 0.1	1.0± 0.0
	Gen-NCM Janson et al. [2022] (NeurIPS-W '22)	0	82.0± 0.0	27.7± 0.0	10.0± 0.0	7.5± 0.0
	Gen-SLDA Hayes and Kanan [2020] (CVPR-W '20)	0	88.0± 0.0	41.4± 0.0	18.8± 0.0	12.9± 0.0
	VAE-GC Van De Ven et al. [2021] (CVPR-W '21)	0	84.0± 0.5	42.7± 1.3	19.7± 0.1	12.1± 0.1
	PEC Zajac et al. [2024] (ICLR '24)	0	92.3± 0.1	58.9± 0.1	26.5± 0.1	14.9± 0.1
	RanDumb (Ours)	0	98.3 (+5.9)	55.6 (-3.3)	28.6 (+2.1)	17.7 (+2.8)

Benchmarks. We illustrate the benchmarks used in our experiments along with key differences to training run for RanDumb in Table 3.2.1. We aim for a truly comprehensive coverage and show results on four different benchmarks (A, B, D, E) which reflect the latest trends in online continual learning (’22-’24) across exemplar-free, contrastive-training², meta-continual learning and network-expansion based approaches respectively and a rehearsal-free offline continual learning

²Benchmark B is split into two distinct sections: (i) methods that do not rely on contrastive learning and heavy augmentation, named B1, and (ii) approaches that incorporate contrastive learning and extra augmentations, named as Scenario B2.

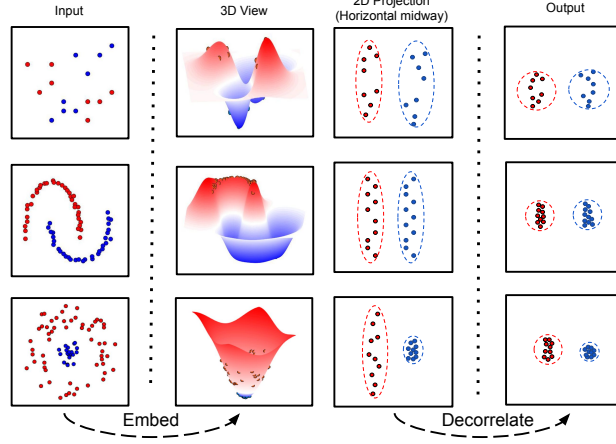


Figure 3.2.1: Mechanism of Randumb. Randumb projects the input datapoints to a high-dimensional space to create a clearer separation between classes. Subsequently, it corrects the anisotropy across different feature dimensions, scaling them to be unit variance each. This allows a nearest class mean classifier to accurately rely on cosine similarity. The figure is adapted from Pilario et al. [2020].

benchmark C. These benchmarks are ordered by increasingly relaxed constraints, moving further away from the training scenario of Randumb. These benchmarks train models from scratch. Hence, Randumb utilizes a random Fourier transform for embedding.

We further test on exemplar-free scenarios in offline continual learning using Benchmark F Wang et al. [2022c] with the challenging one-class per task constraint borrowed from Zajac et al. [2024]. This benchmark allows using pretrained models along with unrestricted training time and access to all class samples at each timestep. However, Randumb is restricted to learning from a single pass seeing only one sample at a time. Randumb only learns a linear classifier over a given pretrained model in Benchmark F. We use LAMDA-PILOT codebase³ for all methods, except RanPAC and SLDA which use their codebase. All methods use the original hyperparameters. We only change initial classes to 2 and number of classes per task to 1 in the scripts and test using both ImageNet21K and ImageNet1K ViT-B/16 models. For detailed assumptions, task orderings, implementation details and other specific information of these benchmarks and approaches, readers are encouraged to consult the LAMDA-PILOT codebase.

Implementation Details (Randumb). We evaluate Randumb using five datasets commonly used for online continual learning: MNIST, CIFAR10, CIFAR100, TinyImageNet200, and miniImageNet100. For the latter two datasets, we downscale all images to 32x32. We augment each datapoint with flipped version, hence two images are seen by the classifier at each timestep (except for MNIST and Benchmark F). We normalize all images and flatten them into vectors, obtaining 784-dim input

³<https://github.com/sun-hailong/LAMDA-PILOT>

Table 3.3.1: Benchmark B.1 (Ref: Table adopted from OnPro Wei et al. [2023], OCMGuo et al. [2022]) We perform comprehensive comparisons of Randumb with popular online continual learning approaches in less ideal many-classes per task setting referred as ‘Dataset (num_tasks/num_classes_per_task)’. Furthermore, results are categorized by memory buffer sizes given in ‘M’ subcolumn for that dataset. Comparing across rows, we observe that Randumb outperforms the best among the compared approaches without heavy-augmentations by 3-20% margins despite being exemplar free. Only in one case, it is second best, but after GDumb. Comparing the two columns of CIFAR100, we observe a large drop in performance of OCL methods on benchmarks with longer timesteps indicating analyzing longer timesteps is important in continual learning.

Method	MNIST (5/2)	CIFAR10 (5/2)		CIFAR100 (10/10)		CIFAR100 (50/2)	TinyImageNet (100/2)	
	M = 0.1k	M = 0.1k	M = 0.2k	M = 0.5k	M = 1k	M = 1k	M = 1k	M = 2k
AGEM Chaudhry et al. [2019a] (ICLR '19)	56.9±5.2	17.7±0.3	22.7±1.8	5.8±0.2	5.9±0.1	1.8±0.2	0.8±0.1	0.9±0.1
GSS Aljundi et al. [2019b] (NeurIPS '19)	70.4±1.5	18.4±0.2	26.9±1.2	8.1±0.2	11.1±0.2	4.3±0.2	1.1±0.1	3.3±0.5
ER Chaudhry et al. [2019b] (ICML-W '19)	78.7±0.4	19.4±0.6	29.7±1.0	8.7±0.3	15.7±0.3	8.3±0.3	1.2±0.1	5.6±0.5
ASER Shim et al. [2021] (AAAI '21)	61.6±2.1	20.0±1.0	27.8±1.0	11.0±0.3	16.4±0.3	9.6±1.3	2.2±0.1	5.3±0.3
MIR Aljundi et al. [2019a] (NeurIPS '19)	79.0±0.5	20.7±0.7	37.3±0.3	9.7±0.3	15.7±0.2	12.7±0.3	1.4±0.1	6.1±0.5
ER-AML Caccia et al. [2022a] (ICLR '22)	76.5±0.1	-	40.5±0.7	-	16.1±0.4	-	-	5.4±0.2
iCaRL Rebuffi et al. [2017b] (CVPR '17)	-	31.0±1.2	33.9±0.9	12.8±0.4	16.5±0.4	-	5.0±0.3	6.6±0.4
DER++ Buzzega et al. [2020] (NeurIPS '20)	74.4±1.1	31.5±2.9	44.2±1.1	16.0±0.6	21.4±0.9	9.3±0.3	3.7±0.4	5.1±0.8
GDumb Prabhu et al. [2020] (ECCV '20)	81.2±0.5	23.3±1.3	35.9±1.1	8.2±0.2	18.1±0.3	18.1±0.3	4.6±0.3	12.6±0.1
CoPE De Lange and Tuytelaars [2021] (CVPR '21)	-	33.5±3.2	37.3±2.2	11.6±0.4	14.6±1.3	-	2.1±0.3	2.3±0.4
DVC Guo et al. [2022] (CVPR '22)	-	35.2±1.7	41.6±2.7	15.4±0.3	20.3±1.0	-	4.9±0.6	7.5±0.5
Co ² L Cha et al. [2021] (ICCV '21)	83.1±0.1	-	42.1±1.2	-	17.1±0.4	-	-	10.1±0.2
R-RT Bang et al. [2021] (CVPR '21)	89.1±0.3	-	45.2±0.4	-	15.4±0.3	-	-	6.6±0.3
CCIL Mittal et al. [2021] (CVPR '21)	86.4±0.1	-	50.5±0.2	-	18.5±0.3	-	-	5.6±0.9
IL2A Zhu et al. [2021] (NeurIPS '21)	90.2±0.1	-	54.7±0.5	-	18.2±1.2	-	-	5.5±0.7
BiC Wu et al. [2019] (CVPR '19)	90.4±0.1	-	48.2±0.7	-	21.2±0.3	-	-	10.2±0.9
SSIL Ahn et al. [2021] (ICCV '21)	88.2±0.1	-	49.5±0.2	-	26.0±0.1	-	-	9.6±0.7
<i>Rehearsal-Free</i>								
PASS Zhu et al. [2021] (NeurIPS '21)	-	33.7±2.2	33.7±2.2	7.5±0.7	7.5±0.7	-	0.5±0.1	0.5±0.1
Randumb (Ours)	98.3 (+7.8)	55.6 (+20.4)	55.6 (+5.9)	28.6 (+12.6)	28.6 (+2.6)	28.6 (+10.5)	11.6 (+6.6)	11.6 (-1.0)

vectors for MNIST and 3072-dim input vectors for all the other datasets. For Benchmark F, we compare Randumb on seven datasets used in LAMDA-PILOT, replacing ObjectNet with Stanford Cars as ObjectNet license prohibits training models on it. We use the 768-dimensional features from the same pretrained ViT-B models used in this benchmark. We measure accuracy on the test set of all past seen classes after completing the full one-pass over the dataset. We take the average accuracy after the last task on all past tasks Zajac et al. [2024], Guo et al. [2022], Wang et al. [2022c]. In Benchmark A and F, since we have one class per task, the average accuracy across past tasks is the same regardless of the task ordering. In Benchmarks A-E, all datasets have the same number of samples, hence similarly the average accuracy across past tasks is the same regardless of the task ordering. We used the Scikit-Learn implementation of Random Fourier Features Rahimi and Recht [2007] with 25K embedding size, $\gamma = 1.0$. We use progressively increasing ridge regression parameter (λ) with dataset complexity, using $\lambda = 10^{-6}$ for MNIST, $\lambda = 10^{-5}$ for CIFAR10/100 and $\lambda = 10^{-4}$ for TinyImageNet200/miniImageNet100. All experiments were conducted on a CPU server with a 48-core Intel Xeon Platinum 8268 CPU and 392GB of RAM. Our code is available at <https://github.com/drimpossible/Randumb>.

Table 3.3.2: (Left) Benchmark B.2 (Ref: Table from OnPro Wei et al. [2023]) We now compare with state-of-the-art contrastive representation learning based online continual learning approaches which additionally use sophisticated augmentations. These augmentations and additional loss function can improve any of the OCL methods described in Setup B.1 by large margins Guo et al. [2022]. We observe that Randumb often outperforms these sophisticated methods despite all of these factors on small-exemplar settings. **(Right) Benchmark C** (Ref: Table 2 from Smith et al. [2023b]). We compare the performance of Randumb with latest efforts on improving rehearsal-free methods. We outperform them by 4% margins.

Method	MNIST (5/2) $M = 0.1k$	CIFAR10 (5/2) $M = 0.1k$	CIFAR100 (10/10) $M = 0.5k$	TinyImageNet (100/2) $M = 1k$	Method	CIFAR100 (10/10)
SCR Mai et al. [2021] (CVPR '21)	86.2±0.5	40.2±1.3	19.3±0.6	8.9±0.3	<i>Rehearsal-Free</i>	
OCM Guo et al. [2022] (ICML '22)	90.7±0.1	47.5±1.7	19.7±0.5	10.8±0.4	PredKD Li et al. [2017a]	24.6
OnPro Wei et al. [2023] (ICCV '23)	-	57.8±1.1	22.7±0.7	11.9±0.3	PredKD + FeatKD	12.4
<i>Rehearsal-Free</i>					PredKD + EWC	23.3
Randumb (Ours)	98.3 (+7.5)	55.6 (-2.2)	28.6 (+5.9)	11.6 (-0.3)	PredKD + L2	21.5
					Randumb	28.6 (+4.0)

Table 3.3.3: (Left) Benchmark D (Ref: Table 2 from VR-MCL Anonymous [2024]) We compare Randumb with meta-continual learning approaches operating in a high memory setting, allowing buffer sizes up to 1K exemplars. Despite catering representations specifically to combat forgetting, all methods with the exception of VR-MCL are outperformed by Randumb on TinyImageNet. Randumb also surpasses all prior work by a substantial 9.1% on CIFAR100. Allowing generous replay buffers shifts scenarios to a high exemplar regime where GDumb performs the best on CIFAR10. Yet Randumb competes favorably even under these conditions not optimized for its approach. **(Middle) Benchmark E** (Ref: Table 1 from SEDEM Ye and Bors [2023]) We compare Randumb with recent network expansion based online continual learning approaches. Despite allowing access to much larger memory buffers, Randumb matches the performance of best method SEDEM on MNIST, while exceeding it by 0.3% on CIFAR10 and 3.8% on CIFAR100. **(Right) Architectures** (Ref: Table 1 from Mirzadeh et al. [2022]) Randumb surpasses continual representation learning across a wide range of architectures, achieving close to 94% of the joint performance.

Method	CIFAR10 (5/2) $M = 1k$	CIFAR100 (10/10) $M = 1k$	TinyImageNet (20/10) $M = 1k$	Method	MNIST (5/2) $M = 2k$	CIFAR10 (5/2) $M = 1k$	CIFAR100 (20/5) $M = 5k$	Model	CIFAR100
Finetune	17.0 ± 0.6	5.3 ± 0.3	3.9 ± 0.2	Finetune	19.8 ± 0.1	18.5 ± 0.3	3.5 ± 0.1	Joint	79.58
A-GEM Chaudhry et al. [2019a] (ICLR '19)	18.4 ± 0.2	6.0 ± 0.2	4.0 ± 0.2	MIR Aljundi et al. [2019a] (NeurIPS '19)	93.2 ± 0.4	42.8 ± 2.2	20.0 ± 0.6	CNN x1	62.2 ± 1.35
IS Zenke et al. [2017] (ICML '17)	17.4 ± 0.2	5.2 ± 0.2	3.3 ± 0.3	GEM Chaudhry et al. [2019a] (ICLR '19)	93.2 ± 0.4	24.1 ± 2.5	11.1 ± 2.4	CNN x2	66.3 ± 1.12
MER Riemer et al. [2019] (ICLR '19)	36.9 ± 2.4	-	-	iCARL Rebuffi et al. [2017b] (CVPR '17)	83.9 ± 0.2	37.3 ± 2.7	10.8 ± 0.4	CNN x4	68.1 ± 0.5
La-MAML Gupta et al. [2020] (NeurIPS '20)	33.4 ± 1.2	11.8 ± 0.6	6.74 ± 0.4	G-MED Jin et al. [2021] (NeurIPS '21)	82.2 ± 2.9	47.5 ± 3.2	19.6 ± 1.5	CNN x16	69.9 ± 0.62
GDumb Prabhu et al. [2020] (ECCV '20)	61.2 ± 1.0	18.1 ± 0.3	4.6 ± 0.3	GSS Aljundi et al. [2019b] (NeurIPS '19)	92.5 ± 0.9	38.5 ± 1.4	13.1 ± 0.9	ResNet-18	76.8 ± 0.76
ER Chaudhry et al. [2019b] (ICML-W '19)	43.8 ± 4.8	16.1 ± 0.9	11.1 ± 0.4	CoPE De Lange and Tuytelaars [2021] (CVPR '21)	93.9 ± 0.2	48.9 ± 1.3	21.6 ± 0.7	ResNet-34	45.0 ± 0.63
DER Buzzega et al. [2020] (NeurIPS '20)	29.9 ± 2.9	6.1 ± 0.1	4.1 ± 0.1	CURL Rao et al. [2019] (NeurIPS '19)	92.6 ± 0.7	-	-	ResNet-50	44.8 ± 2.34
DER++ Buzzega et al. [2020] (NeurIPS '20)	52.3 ± 1.9	11.8 ± 0.7	8.3 ± 0.3	CNDPM Lee et al. [2020] (ICLR '20)	95.4 ± 0.2	48.8 ± 0.3	22.5 ± 1.3	ResNet-101	56.2 ± 0.88
CLSER Arani et al. [2022] (ICLR '22)	52.8 ± 1.7	17.9 ± 0.7	11.1 ± 0.2	Dynamic-OCM Ye and Bors [2022] (ECCV '22)	94.0 ± 0.2	49.2 ± 1.5	21.8 ± 0.7	WRN-10-2	56.8 ± 1.62
OCM Guo et al. [2022] (ICML '22)	53.4 ± 1.0	14.4 ± 0.8	4.5 ± 0.5	SEDEM Ye and Bors [2023] (ICCV '23)	98.3 ± 0.2	55.3 ± 1.3	24.8 ± 1.2	WRN-10-10	50.5 ± 2.65
ER-OBK Chrysakos and Moens [2023] (ICLR '23)	54.8 ± 2.2	17.2 ± 0.9	11.5 ± 0.2	<i>Rehearsal-Free</i>				WRN-16-2	56.8 ± 2.03
VR-MCL Anonymous [2024] (ICLR '24)	56.5 ± 1.8	19.5 ± 0.7	13.3 ± 0.4	Randumb (Ours)	98.3 (0.0)	55.6 (+0.3)	28.6 (+3.8)	WRN-16-10	44.6 ± 2.81
<i>Rehearsal-Free</i>								WRN-16-10	51.3 ± 1.47
Randumb (Ours)	55.6 (-5.6)	28.6 (+9.1)	11.6 (-1.7)					WRN-28-2	46.6 ± 2.27
								WRN-28-10	49.3 ± 2.02
								VIT-512/1024	51.7 ± 1.4
								VIT-1024/1546	60.4 ± 1.56
								Randumb (Ours)	74.8 (-2.0)

3.3.1 Results

We extensively evaluate Randumb and detail our findings on each benchmark in this section.

Benchmark A. Benchmark A assesses continual learning models in the challenging setup of one class per timestep, closely mirroring our training assumptions. We present our results in Table 3.2.2. Comparing across rows, and see that Randumb improves over prior state-of-the-art

across all datasets with 2-6% margins. The only exception is PEC on CIFAR10, where Randumb underperforms by 3.3%. Nonetheless, it still outperforms the second-best model, GDumb with a 500 memory size, by 4.9%.

Benchmark B1. We present our results comparing with non-contrastive methods in Table 3.3.1. We notice that scenario allows two classes per task and relaxes the memory constraints for online continual learning methods, allowing for higher accuracies compared to Benchmark A. Despite that, Randumb outperforms latest OCL algorithms on MNIST, CIFAR10 and CIFAR100—often by margins exceeding >10%. The lone exception is GDumb achieving a higher performance with 2K memory samples on TinyImageNet, indicating that this already is in the high-memory regime. Overall, Randumb is the best algorithm across the benchmark.

Benchmark B2. We additionally compare our performance with the latest online CL approaches using contrastive losses with sophisticated data augmentations. As shown in in Table 3.3.2 (Left), these advancements provide large performance improvements over methods from Benchmark B.1. To compensate, we compare on lower exemplar budgets. The best approach, OnPro Wei et al. [2023], which outperforms Randumb on CIFAR10 by 2.2% and TinyImageNet by 0.3%, but falls significantly short on CIFAR100 by 5.9%. Overall, Randumb achieves strong results compared to highly performant representation learning using state-of-the-art contrastive learning approaches customized to continual learning, despite storing no exemplars.

Benchmark C. We now compare against offline rehearsal-free continual learning approaches in Table 3.3.2 (Right) on CIFAR100 dataset. Despite online training, Randumb outperforms PredKD by over 4% margins.

Benchmark D. We now compare performance of Randumb against meta-continual learning methods, which require large exemplars with buffer sizes of 1K in Table 3.3.3 (left). Randumb achieves strong performance under these conditions, exceeding all prior work by a large margin of 9.1% on CIFAR100 and outperforms all but VR-MCL approach on the TinyImageNet dataset. GDumb performs the best on CIFAR10, indicating this is already in a large-exemplar regime uniquely unsuited for Randumb.

Benchmark E. We compare Randumb against recent network expansion-based online continual learning methods in Table 3.3.3 (middle). These approaches grow model capacity to mitigate forgetting while dealing with shifts in the data distribution, and are allow larger memory buffers. Randumb matches the performance of the state-of-the-art method SEDEM Ye and Bors [2023] on MNIST, while exceeding it by 0.3% on CIFAR10 and 3.8% on CIFAR100. The results indicate current benchmarks remain too restrictive for effective continual representation learning.

Table 3.3.4: Benchmark F We compare RanDumb with other prompt-tuning based continual learning approaches using ViT-B/16 ImageNet-21K/1K pretrained models using 2 init classes and 1 class per task setting. Most prompt-tuning based methods collapse and simple baselines which do not pretrain the architecture like NCM Janson et al. [2022] or RanDumb achieve state-of-the-art performance. ‘-’ indicates that despite reasonable efforts, we could not run the method. ⁺here representation learning and random projection both collapse, reporting last stable accuracy.

Method	CIFAR	IN-A	IN-R	CUB	VTAB
ViT-B/16 (IN-1K Pretrained)					
Finetune	1.0	1.2	1.1	1.0	2.1
L2P Wang et al. [2022c] (CVPR ’22)	2.4	0.3	0.8	1.4	1.3
DualPrompt Wang et al. [2022b] (ECCV ’22)	2.3	0.3	0.8	0.9	4.2
CODA-Prompt Smith et al. [2023a] (CVPR ’23)	2.6	0.3	0.8	1.9	6.3
Adam-Adapt Zhou et al. [2023a] (Arxiv ’23)	76.7	49.3	62.0	85.2	83.6
Adam-SSF Zhou et al. [2023a] (Arxiv ’23)	76.0	47.3	64.2	85.6	84.2
Adam-VPT Zhou et al. [2023a] (Arxiv ’23)	79.3	35.8	61.2	83.8	86.9
Adam-FT Zhou et al. [2023a] (Arxiv ’23)	72.6	49.3	61.0	85.2	83.8
Memo Zhou et al. [2022] (ICLR ’23)	69.8	-	-	81.4	-
iCARL Rebuffi et al. [2017b] (CVPR ’17)	72.4	-	35.2	72.4	-
Foster Wang et al. [2022a] (ECCV ’22)	52.2	-	76.8	86.6	-
NCM Janson et al. [2022] (NeurIPS-W ’22)	76.2	49.4	61.2	85.2	83.6
SLCA Zhang et al. [2023a] (ICCV ’23)	86.3	-	52.8	84.7	-
RanPAC McDonnell et al. [2023] (NeurIPS ’23)	88.2	39.0 ⁺	72.8	77.7 ⁺	93.0
RanDumb (Ours)	84.5	47.8	66.9	88.0	93.4
ViT-B/16 (IN-21K Pretrained)					
Finetune	2.8	0.5	1.2	1.2	0.5
Adam-Adapt Zhou et al. [2023a] (Arxiv ’23)	82.4	48.8	55.4	86.7	84.4
Adam-SSF Zhou et al. [2023a] (Arxiv ’23)	82.7	46.0	59.7	86.2	84.9
Adam-VPT Zhou et al. [2023a] (Arxiv ’23)	70.8	34.8	53.9	84.0	81.1
Adam-FT Zhou et al. [2023a] (Arxiv ’23)	65.7	48.5	56.1	86.5	84.4
Foster Wang et al. [2022a] (ECCV ’22)	87.3	-	5.1	86.9	-
iCARL Rebuffi et al. [2017b] (CVPR ’17)	71.6	-	35.1	71.6	-
NCM Janson et al. [2022] (NeurIPS-W ’22)	81.3	48.9	54.6	86.7	84.4
SLCA Zhang et al. [2023a] (ICCV ’23)	86.8	-	54.2	82.1	-
RanPAC McDonnell et al. [2023] (NeurIPS ’23)	89.6	26.8 ⁺	67.3	87.2 ⁺	88.2 ⁺
RanDumb (Ours)	86.8	42.2	64.9	88.5	92.3

Benchmark F. We compare performance of approaches which do not further train the deep network like RanDumb (ours)⁴ and NCM Janson et al. [2022] against popular prompt-tuning approaches in Table 3.3.4. and discover that prompt-tuning approaches completely collapse under large timesteps and approaches which do not finetune their pretrained model achieve strong performance, even under challenging one class per timestep constraint. Note that there are variants of RanDumb (*e.g.* RanPAC - RP - PETL McDonnell et al. [2023]) that achieve higher accuracies due to minor modifications which are designed for this benchmark.

Conclusion. Overall, despite RanDumb being exemplar-free and modeling worst-case ordering, it outperforms nearly all online continual learning methods across various tasks when exemplar storage is limited. We specifically benchmark on lower exemplar sizes to complement settings in which GDumb does not perform well.

⁴Recall that RanDumb in Benchmark F is quite similar to SLDA Hayes and Kanan [2020].

Table 3.3.5: RanDumb (Analysis). We thoroughly analyse the key components of RanDumb, and observe: (1) contributions of decorrelation and embedding, finding them interdependent and integral to performance; (2) consistent gains from augmentation across datasets; (3) marginal returns and saturation with higher embedding sizes, enabling computational tradeoffs; (4) increase in the optimal regularisation parameter for more complex datasets; (5) comparisons with alternate embeddings (or lack thereof) favoring RanDumb’s approach.

Method	MNIST (10/1)	CIFAR10 (10/1)	CIFAR100 (10/1)	T-ImNet (200/1)	m-ImNet (100/1)
Ablating Components of RanDumb					
RanDumb	98.3	55.6	28.6	11.1	17.7
-Decorrelate	83.8 (-14.5)	30.0 (-25.6)	12.0 (-16.6)	4.7 (-6.4)	8.9 (-8.8)
-Embed	88.0 (-10.3)	41.6 (-14.0)	19.0 (-9.6)	8.0 (-3.1)	12.9 (-4.8)
-Both	82.1 (-16.2)	28.5 (-27.1)	10.4 (-18.2)	4.1 (-7.0)	7.28 (-10.4)
Effect of Adding Flip Augmentation					
With	-	55.6	28.6	11.1	17.7
Without	98.3	52.5 (-3.1)	26.9 (-1.7)	10.7 (-0.4)	16.6 (-1.1)
Variation with Ridge Parameter λ					
$\lambda = 10^{-6}$	98.3	53.9	27.8	10.3	15.8
$\lambda = 10^{-5}$	-	55.6	28.6	11.1	15.9
$\lambda = 10^{-4}$	96.6	52.6	26.1	11.6	17.7
Variation Across Embedding Projections					
No-Embed	88.0	41.6	19.0	8.0	12.9
RP+ReLU	95.2	48.8	23.1	9.7	15.7
RanDumb (Ours)	98.3 (+3.1)	55.6 (+6.8)	28.6 (+5.5)	11.1 (+1.4)	17.7 (+2.0)

3.3.2 RanDumb: Analysis

We extensively analyse RanDumb for the traditional OCL benchmarks in this subsection. First, we ablate the two components (random embedder and decorrelation) in RanDumb to ensure both components are necessary. Then we study effect of varying the embedding dimensionality, shrinkage and removing flipping augmentation and compare with alternate random embeddings proposed in literature McDonnell et al. [2023] and across architectures Mirzadeh et al. [2022].

Ablating Components of RanDumb. We ablate the contribution of only using Random Fourier features for embedding and decorrelation to the overall performance of RanDumb in Table 3.3.5 (top). Ablating the decorrelation and relying solely on random Fourier features, colloquially dubbed Kernel-NCM, has performance drops ranging from 6-25% across the datasets. Replacing random Fourier features with raw features, colloquially the SLDA baseline, leads to pronounced drop in performance ranging from 3-14% across the datasets. Moreover, abating both components, resulting in the base nearest class mean classifier exhibits the poorest performance with an average reduction of 17%. The poor performance when abating both decorrelation and embedding highlights their interdependence.

Impact of Embedding Dimensions. We vary the dimensions of the random Fourier features ranging from compressing 3K input dimensions to 1K to projecting it to 25K dimensions and

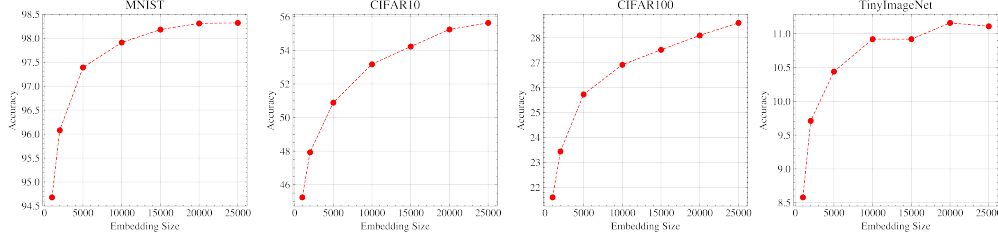


Figure 3.3.1: Impact of Embedding Dimensions. We show the variation of accuracy of RanDumb with embedding dimensionality across datasets. We see decreasing marginal returns with increasing dimensions, obtaining relatively minor improvements after 15K dimensions.

evaluate its impact on performance in Figure 3.3.1. Surprisingly, the random projection to a 3x compressed 1K dimensional space allows for significant performance improvement over not using embedding, given in Table 3.3.5 (top). Furthermore, increasing the dimension from 1K to 25K results in improvements of 3.6%, 10.4%, 7.0%, and 2.5% on MNIST, CIFAR10, CIFAR100, and TinyImageNet respectively. Increasing the embedding sizes beyond 15K, however, only results in modest improvements of 0.1%, 1.4%, 1.1% and 0.2% on the same datasets, indicating 15K dimensions would be a good point for a performance-computational cost tradeoff.

Impact of Flip Augmentation. We evaluate the impact of adding the flip augmentation on the performance of RanDumb in Table 3.3.5 (middle). Note that MNIST was not augmented. Augmentation provided large gains of 3.1% on CIFAR10, 1.7% on CIFAR100, and 0.4% on TinyImageNet. We did not try further augmenting the data with RandomCrop transform as done with standard augmentations.

Impact of Varying Ridge Parameter. All prior experiments use a ridge parameter (λ) that increases with dataset complexity: $\lambda = 10^{-6}$ for MNIST, 10^{-5} for CIFAR10 and CIFAR100, and 10^{-4} for TinyImageNet and miniImageNet. Table 3.3.5 (middle) shows the effect of varying λ on RanDumb’s performance. With a smaller $\lambda = 10^{-6}$, CIFAR10, CIFAR100, TinyImageNet and miniImageNet all exhibit minor drops of 0.1%-1.7%, 0.8%, 0.8%. Increasing shrinkage to a $\lambda = 10^{-4}$ reduces CIFAR10 and CIFAR100 performance more substantially by 3% and 2.5% versus their optimal $\lambda = 10^{-5}$. On the other hand, this larger λ leads to improvements of 0.5% and 1.8% on TinyImageNet and miniImageNet. This aligns with the trend that datasets with greater complexity benefit from more regularisation, with the optimal λ surfacing as a result of balancing under- and over-regularisation effects.

Comparison with RanPAC. We compared our random Fourier features with projection based on a random weight matrix and ReLU McDonnell et al. [2023] (RP+ReLU) in Table 3.3.5 (middle) with their best embedding size. Our method performs significantly better on each dataset, averaging

a gain of 3.4%. We believe this is attributable to using a theoretical grounded random projection compared to RanPAC McDonnell et al. [2023], which is specifically designed for the setting using pretrained transformers.

Comparisons across Architectures. In table 3.3.3 (right), we compare whether using random Fourier features as embeddings outperforms models across various architectures for continual representation learning. We use experience replay (ER) baseline in the task-incremental CIFAR100 setup (for details, see Mirzadeh et al. [2022] as it differs significantly from earlier setups). Our comparison involved across various architectures. The findings revealed that RanDumb surpassed the performance of nearly all considered architectures, and achieved close to 94% of the joint multi-task performance. This suggests that RanDumb outperforms continual representation learning across a wide range of architectures.

Conclusion. Overall, both random embedding and decorrelation are critical components in the performance of RanDumb. Using random fourier features is substantially better than RanPAC. Lastly, one can substantially reduce the embedding dimension without a large drop in performance for large gains in computational cost and RAM usage, additional augmentation may further significantly help performance and optimal shrinkage parameter increases with dataset complexity. RanDumb outperforms continual representation learning across a wide range of architectures.

3.4 Discussion and Concluding Remarks

Our investigation reveals a surprising result, empirically demonstrating that currently the representation gain is negative, i.e. $RepGain(\mathcal{D}) < 0$ — simply using random embedding (RanDumb) consistently outperforms learned representations from methods specifically designed for continual training. Furthermore, using fixed pretrained networks consistently outperforms continually fine-tuned method. Using random/pretrained features also recovers 70-90% of the gap to joint learning, leaving limited room for improvement in representation learning techniques. Our investigation is both surprising and alarming as it questions our understanding of how to effectively design and train models that require efficient continual representation learning, and necessitates a re-investigation of the widely explored problem formulation itself. We believe adoption of computationally bounded scenarios without memory constraints Prabhu et al. [2023b,a], Garg et al. [2023] could be a promising way forward. We develop what would continual learning look like under computational cost considerations as an alternative formulation in the next chapter.

4

Computationally Budgeted Continual Learning: What Does Matter?

Contents

4.1	Introduction	43
4.2	Continual Learning with Limited Compute	44
4.2.1	Problem Formulation	44
4.2.2	Key Differences with Prior Art	44
4.2.3	Constructing the Stream	46
4.3	Dissecting Continual Learning Systems	46
4.4	Experiments	49
4.4.1	Experimental Setup and Details	49
4.4.2	Budgeted Continual Learning	52
4.4.3	Sensitivity Analysis	55
4.4.4	Exploring Partial Training	57
4.5	Conclusion	58

Abstract

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

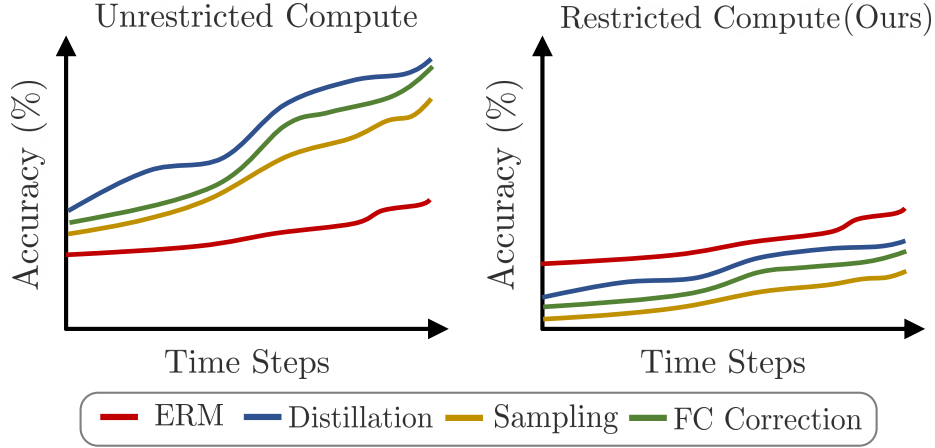


Figure 4.1.1: Main Findings. Under per time step computationally budgeted continual learning, classical continual learning methods, *e.g.*, sampling strategies, distillation losses, and fully connected (FC) layer correction based methods such as calibration, struggle to cope with such a setting. Most proposed continual algorithms are particularly useful only when large computation is available, where, otherwise, minimalistic algorithms (ERM) are superior.

4.1 Introduction

In this chapter, we study the question: “*Do existing continual learning algorithms perform well under per step restricted computation?*” To address this question, we exhaustively study continual learning systems, analyzing the effect of the primary directions of progress proposed in the literature in the setting where algorithms are permitted fixed computational budget per stream time step. We evaluate and benchmark at scale various classical CL sampling strategies (Uniform, Class-Balanced[Prabhu et al., 2020], Recency-Biased[Lin et al., 2021], FIFO [Chaudhry et al., 2019b, Cai et al., 2021], Max Loss, Uncertainty Loss [Bang et al., 2021], and KMeans [Chaudhry et al., 2019b]), CL distillation strategies (BCE [Rebuffi et al., 2017b], MSE [Buzzega et al., 2020], CrossEntropy [Wu et al., 2019], and Cosine [Hou et al., 2019]) and FC layer corrections (ACE [Zeno et al., 2018, Mai et al., 2022, Caccia et al., 2022b], BiC [Wu et al., 2019], CosFC [Hou et al., 2019], and WA [Zhao et al., 2020]) that are common in the literature. Evaluation is carried on two large-scale datasets, amounting to a total of 1500 GPU-hours, namely ImageNet [Deng et al., 2009] and Continual Google Landmarks V2 [Prabhu et al., 2023a] (CGLM) under various stream settings, namely, data incremental, class incremental, and time incremental settings. We compare against Naive; a simple baseline that, utilizing all the per step computational budget, trains while sampling from previous memory samples.

Conclusions. We summarize our empirical conclusions in three folds. (1) None of the proposed

CL algorithms, see Table 4.1.1 for considered methods, can outperform our simple baseline when computation is restricted. **(2)** The gap between existing CL algorithms and our baseline becomes larger with harsher compute restrictions. **(3)** We find that training a minimal subset of the model can close the performance gap compared to our baseline in our setting, but only when supported by strong pretrained models.

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

4.2 Continual Learning with Limited Compute

4.2.1 Problem Formulation

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

4.2.2 Key Differences with Prior Art

(1) Tasks: In most prior work, CL is simplified to the problem of learning a set of non-overlapping tasks, *i.e.*, distributions, *with known boundaries* between them [Rebuffi et al., 2017b, Lopez-Paz

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

Dir.	Reference	Applicability (our setup)	Distillation	MemUpdate	Components MemRetrieve	FC Correction	Others
Distillation	Naive	✓	-	Random	Random	-	-
	iCARL [Rebuffi et al., 2017b]	✓	BCE	Herd	Random	-	NCM
	LUCIR [Hou et al., 2019]	✓	Cosine	Herd	MargRank	CosFC	NCM
	PODNet [Douillard et al., 2020]	✓	POD	Herd	Random	LSC	Imprint, NCM
	DER [Buzzega et al., 2020]	✓	MSE	Reservoir	Random	-	-
	CO ² L [Cha et al., 2021]	×	IRD	Random	Random	Asym.SupCon	-
Sampling	SCR [Mai et al., 2021]	✓	-	Reservoir	Random	SupCon	NCM
	TinyER [Chaudhry et al., 2019b]	✓	-	FIFO,KMeans,Reservoir	-	-	-
	GSS [Aljundi et al., 2019b]	×	-	GSS	Random	-	-
	MIR [Aljundi et al., 2019a]	×	-	Reservoir	MIR	-	-
	GDumb [Prabhu et al., 2020]	✓	-	Balanced	Random	-	MemOnly
	Mnemonics [Liu et al., 2020]	×	-	Mnemonics	-	-	BalFineTune
	OCS [Yoon et al., 2022]	×	-	OCS	Random	-	-
	InfoRS [Sun et al., 2022]	×	MSE	InfoRS	Random	-	-
	RMM [Liu et al., 2021]	×	-	RMM	-	-	-
	ASER [Shim et al., 2021]	×	-	SV	ASV	-	-
	RM [Bang et al., 2021]	✓	-	Uncertainty	Random	-	AutoDA
	CLIB [Koh et al., 2022]	×	-	Max Loss	Random	-	MemOnly, AdaLR
FC Layer	BiC [Wu et al., 2019]	×	CrossEnt	Random	Random	BiC	-
	WA [Zhao et al., 2020]	×	CrossEnt	Random	Random	WA	-
	SS-IL [Ahn et al., 2021]	×	TKD	Random	Balanced	SS	-
	CoPE [De Lange and Tuytelaars, 2021]	✓	-	Balanced	Random	PPPLoss	-
	ACE [Caccia et al., 2022b]	✓	-	Reservoir	Random	ACE	-

Table 4.2.1: Primary Directions of Progress in CL. Analysis of recent replay-based systems, with **bold** highlighting the primary contribution. We observe that there are three primary directions of improvement. “App.” denotes the applicability to our setting based on whether they are scalable to large datasets and applicable beyond the class-incremental stream.

and Ranzato, 2017, Chaudhry et al., 2018]. In particular, the data of a given distribution $\mathcal{D}_j$ is given all at once for the model to train. This is as opposed to our setup, where there is no knowledge about the distribution boundaries, since they are often gradually changing and not known a priori. As such, continual learning methods cannot train only just before the distribution changes.

(2) *Computational Budget*: A key feature of our work is that, per time step, CL methods are given a fixed computational budget C_t to train on $\{(x_i^t, y_i^t)\}_{i=1}^{n_t}$. For ease, we assume throughout that $C_t = C \forall t$, and that $n_t = n \forall t$. Although C can be represented in terms of wall clock training time, for a given f_θ and stream $\mathcal{S}$, and comparability between GPUs, we state C in terms of the number of training iterations instead. This avoids hardware dependency or suboptimal implementations when comparing methods. This is unlike prior work, which do not put hard constraints on compute per step [Rebuffi et al., 2017b, Hou et al., 2019, Buzzega et al., 2020] giving rise to degenerate but well-performing algorithms such as GDumb [Prabhu et al., 2020]. Concurrent works [Ghunaim et al., 2023, Prabhu et al., 2023a, Bornschein et al., 2022] restrict the computational budget, however, they focus on adaptation to future whereas we focus on alleviating catastrophic forgetting.

(3) *Memory Constraints*: Prior work focuses on a fixed, small memory buffer for learning and thereof proposing various memory update strategies to select samples from the stream. We assume that all the samples seen so far can be stored at little cost. However, given the restricted

imposed computation C , CL methods cannot revisit or learn from all stored samples. For example, as shown in Figure 4.4.1, consider performing continual learning on ImageNet2K, composed of 1.2M samples from ImageNet1K and 1.2M samples from ImageNet21K forming 2K classes, which will be detailed later, over 20 time steps, where the stream reveals sequentially $n = 60K$ images per step. Then, under a computation budget of 8000 iterations, the model cannot revisit more than 50% of all seen data at any given time step, *i.e.* 600K samples. Our proposed setting is closer to realistic scenarios that cope with high-throughput streams, where *computational bottlenecks impose implicit constraints on learning from past samples that can be too many to be revisited during training*.

4.2.3 Constructing the Stream

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

(1) *Data Incremental Stream*: In this setting, there are no constraints imposed on the incoming distribution $\mathcal{D}_j$ over time, an assumption critical to continual learning because data distributions vary unpredictably in real-world tasks. This has often been neglected to be emphasized in prior work.. We randomly shuffle all data and then reveal it sequentially over steps, which could lead to a varying distribution $\mathcal{D}_j$ over steps in which there are no clear distribution boundaries.

(2) *Time Incremental Stream*: In this setting, the stream data is ordered by the upload timestamp to a server, reflecting a natural distribution change $\mathcal{D}_j$ across the stream as it would in real scenarios. There is a recent shift toward studying this ordering as apparent in recent CL benchmarks, *e.g.*, CLEAR [Lin et al., 2021], Yearbook [Yao et al., 2022] and FMoW [Yao et al., 2022], NEVIS22 [Bornschein et al., 2022], Continual YFCC100M [Cai et al., 2021] and Continual Google Landmarks V2 [Prabhu et al., 2023a].

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

4.3 Dissecting Continual Learning Systems

Continual learning methods typically propose a system of multiple components that jointly help improve learning performance. For example, LUCIR[Hou et al., 2019] is composed of a cosine linear layer, a cosine distillation loss function, and a hard-negative mining memory-based selector. In this section, we analyze continual learning systems and dissect them into their underlying components.

This helps to analyze and isolate the role of different components under our budgeted computation setting and helps us to understand the most relevant components. In Table 4.2.1, we present the breakdown of novel contributions that have been the focus of recent progress in CL. The columns indicate the major directions of change in the CL literature. Overall, there have been three major components on which advances have focused, namely distillation, sampling, and FC layer correction. These three components are considered additions to a naive baseline that simply performs uniform sampling from memory. We refer to this baseline as Naive in Table 4.2.1.

(1) *Distillation*: One popular approach towards preserving model performance on previous distributions has been through distillation. It enables student models, *i.e.*, current time step model, to learn from a teacher model, *i.e.*, one that has been training for many time steps, through the logits providing a rich signal. In this chapter, we consider four widely adopted distillation losses, namely, Binary CrossEntropy (BCE) [Rebuffi et al., 2017b], CrossEntropy [Wu et al., 2019, Zhao et al., 2020, Liu et al., 2020], Cosine Similarity (Cosine)[Hou et al., 2019], and Mean Square Error (MSE)[Buzzega et al., 2020, Sun et al., 2022] Loss.

(2) *Sampling*: Rehearsing samples from previous distributions is another popular approach in CL. However, sampling strategies have been used for two objectives. Particularly when access to previous samples is restricted to a small memory, they are used to select which samples from the stream will update the memory (MemUpdate) or to decide on which memory samples are retrieved for rehearsal (MemRetrieve). In our unconstrained memory setup, simply sampling uniformly over the joint data of past and current time step data (as in Naive) exposes a particular shortcoming. When training for a large number of time steps, uniform sampling reduces the probability of selecting samples from the current time step. For that, we consider various sampling strategies, *e.g.*, recency sampling [Lin et al., 2021] that biases toward sampling current time step data, and FIFO [Chaudhry et al., 2019b, Cai et al., 2021] that exclusively samples from the current step. We do not consider Reservoir, since it approximates uniform sampling in our setup with no memory restrictions. In addition to benchmarking the sampling strategies mentioned above, we also consider approaches that evaluate the contribution of each memory sample to learning [Toneva et al., 2019]. For example, Herding [Rebuffi et al., 2017b], K-Means [Chaudhry et al., 2019b], OCS [Yoon et al., 2022], InfoRS [Sun et al., 2022], RM [Bang et al., 2021], and GSS [Aljundi et al., 2019b] aim to maximize diversity among samples selected for training with different metrics. MIR [Aljundi et al., 2019a], ASER[Shim et al., 2021], and CLIB [Koh et al., 2022] rank the samples according to their informativeness and select the top- k . Lastly, balanced sampling [Prabhu et al., 2020, De Lange and Tuytelaars, 2021, Chrysakis and Moens, 2020] select samples such that an equal distribution of

classes is selected for training. In our experiments, we only consider previous sampling strategies that are applicable to our setup and compare them against Naive.

(3) *FC Layer Correction*: It has been hypothesized that the large difference in the magnitudes of the weights associated with different classes in the last fully connected (FC) layer is among the key reasons behind catastrophic forgetting [Wu et al., 2019]. There has been a family of different methods addressing this problem.

These include methods that improve the design of FC layers, such as CosFC [Hou et al., 2019], LSC [Douillard et al., 2020], and PPP [De Lange and Tuytelaars, 2021], by making the predictions independent of their magnitude. Other approaches such as SS-IL [Ahn et al., 2021] and ACE [Zeno et al., 2018, Mai et al., 2022, Caccia et al., 2022b] mask out unaffected classes to reduce their interference during training. In addition, calibrating the FC layer in post-training, *e.g.*, BiC [Wu et al., 2019], WA [Zhao et al., 2020], and IL2M [Belouadah and Popescu, 2019] is widely used. Note that the calibration techniques are only applicable to the class-incremental setup. We benchmark existing methods applicable to our setting against the Naive approach that does not implement any FC layer correction.

(4) *Model Expansion Methods*: Several works attempt to adapt the model architecture according to the data. This is done by only training part of the model [Mallya and Lazebnik, 2018, Mallya et al., 2018, Abati et al., 2020, Aljundi et al., 2017, Rajasegaran et al., 2019] or by directly expanding the model when data is presented [Rusu et al., 2016, Yan et al., 2021, Yoon et al., 2018, Zhang et al., 2020a, Wang et al., 2017, Rebuffi et al., 2017a]. However, most of the previous techniques in this area do not apply to our setup. Most of this line of work [Mallya and Lazebnik, 2018, Mallya et al., 2018, Rebuffi et al., 2017a] assumes a task-incremental setting, wherein at each time step, new samples are always labeled with new, disjoint classes. Furthermore at test time, one is provided which task any sample belongs to as additional information, *i.e.*, the distribution boundaries are known, even at test time. To overcome these limitations, newer methods [Rajasegaran et al., 2020, Abati et al., 2020] use a bilevel prediction structure, predicting the task at one level and the label within the task at the second level. They are restricted to the class-incremental setting as they assume each task corresponds to a set of non-overlapping classes. We seek to understand the limitation of partial retraining in a network; hence, instead, we compare Naive against a setting where only the FC layer is being trained, thus minimally training the network per time step. In addition, we examine the role of pretraining which has recently become a widely popular direction for exploration in continual learning [Wu et al., 2022].

4.4 Experiments

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

4.4.1 Experimental Setup and Details

Model. We use a standard ResNet50 following prior work on continual learning [Cai et al., 2021]. The model is ImageNet1K pre-trained used as a backbone throughout all experiments.

Datasets. We conduct experiments using two large-scale datasets, namely ImageNet2K and Continual Google Landmarks V2 (CGLM). We construct ImageNet2K by augmenting ImageNet1K with 1.2M images from ImageNet21K [Deng et al., 2009], thus, adding 1K new non-overlapping classes with ImageNet1K amounting to a total of 2K classes.

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

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

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

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

Computational Budget. We set the computational budget C to 400 training iterations per time step ($8000 = 20 \text{ time steps} \times 400$) for ImageNet2K, *i.e.*, DI-ImageNet2K and CI-ImageNet2K, and set C to 100 training iterations for CGLM. In each iteration, a batch of images is used to update the model in training where we set the training batch size $\mathcal{B}$ to 1500. We set the constrained

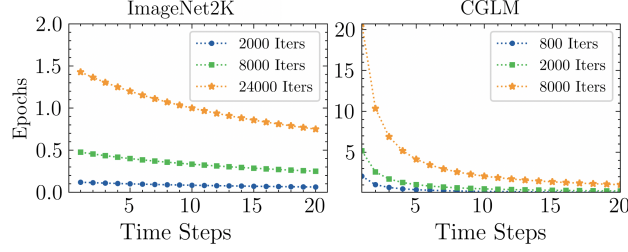


Figure 4.4.1: Effective Training Epochs Per Time Step. Our default setting sets a total training budget over all 20 time steps of 8000 and 2000 iterations for ImageNet2K and CGLM ,respectively, with a per iteration batch size of $\mathcal{B} = 1500$. Effectively, this reflects to training on 25-50% of the stored data, except in the first few time steps on CGLM. Note that for ImageNet2K, we assume that ImageNet1K of 1.2M samples is available in memory.

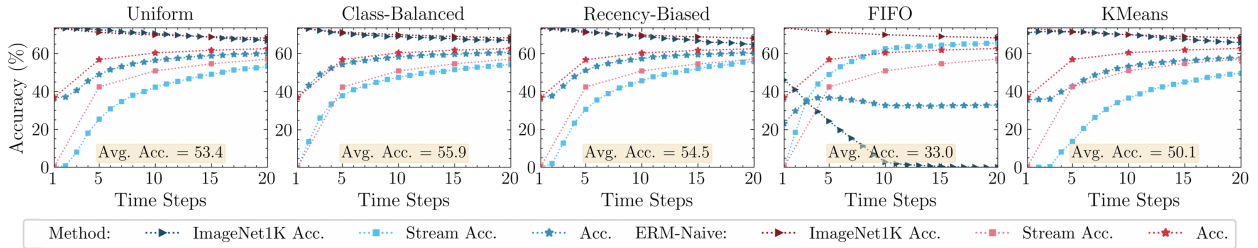


Figure 4.4.2: DI-ImageNet2K (400 Iterations). ERM-Naive, a non-continual learning algorithm, is compared against inexpensive sampling strategies (first four plots) with 400 training iterations and the costly KMeans (fifth plot) with 200 iterations. All CL methods perform similarly but worse than ERM-Naive. This is the case for FIFO that suffers from forgetting the KMeans due to its expensive nature. ImageNet2K experiments performance can be decomposed into (i) accuracy on classes seen during pre-training on ImageNet1K and (ii) accuracy on newly seen classes in ImageNet2K, allowing analysis of forgetting old classes and learning newly introduced classes.

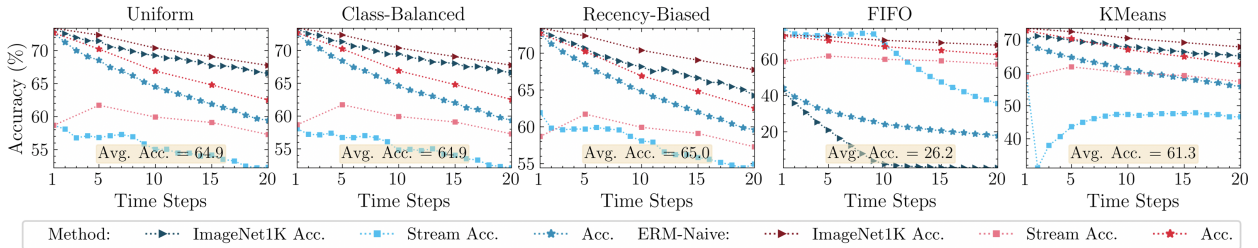


Figure 4.4.3: CI-ImageNet2K (400 Iterations). Similarly, ERM-Naive is compared on CI-ImageNet2K. Both FIFO, which suffers from forgetting, and KMeans due to its expensive nature, struggle to compete against simpler inexpensive methods as Class-Balanced. However, overall, all other methods perform very similarly with no clear advantage. ImageNet2K experiments performance can be decomposed into (i) accuracy on classes seen during pre-training on ImageNet1K and (ii) accuracy on newly seen classes in ImageNet2K, allowing analysis of forgetting old classes and learning newly introduced classes.

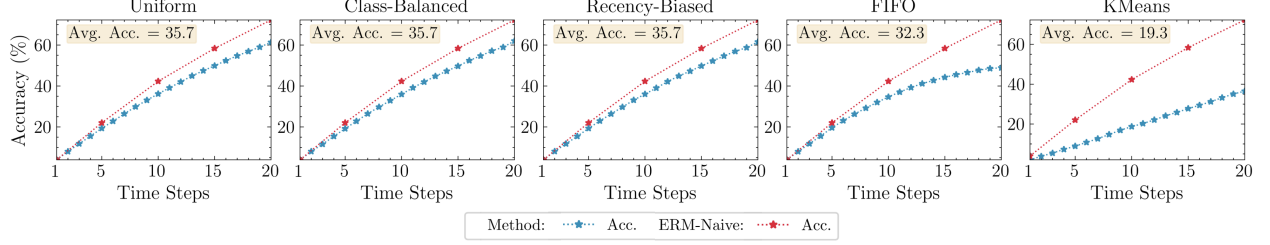


Figure 4.4.4: CGLM (100 Iterations). All inexpensive methods perform overall similarly with the exception for KMeans due to its expensive nature. This highlights that simplicity is key under a budgeted continual learning setting. CGLM is not an extension of ImageNet1K and involves a different task: landmark classification. Hence, we measure only the stream accuracy resulting in two lines instead of six.

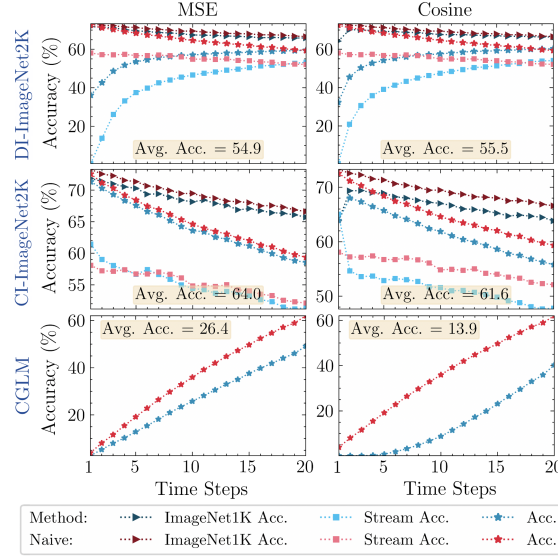


Figure 4.4.5: Distillation in Data and Class Incremental Settings. Naive, which does not employ any distillation loss, outperforms all distillation methods (MSE, and Cosine) across all three settings.

compute budget C such that, at any given step, one can train models using data equivalent to one pass over 25–50% of the total data observed thus far. For example, as highlighted in Figure 4.4.1 for ImageNet2K, time step $t = 5$. corresponds to training only on about 40% of the complete observed data at this step, *i.e.*, $400 \times 1500 / (1.2M + 5 \times 60K) \approx 0.4$ of an epoch where $1.2M$ denotes the ImageNet1K samples. Furthermore, we set C to 100 iterations for CGLM, since the dataset contains $1/4$ of the total data in ImageNet2K. Note that after 20 time steps on CGLM, the data that would have been seen is $20 \times 29K$ images, as opposed to $1.2M + 20 \times 60K$ images for ImageNet2K experiments.

Metrics. We report the accuracy (Acc) on a separate test set after training at each time step. This test set simply comprises the joint test set for all classes seen up to the current time step. Moreover, for ImageNet2K, we decompose the test accuracy into the accuracy on ImageNet1K

(ImageNet1K Acc), which measures forgetting, and the accuracy on the stream (Stream Acc), which measures adaptation. For GCLM, we only report stream accuracy.

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

We summarize all the settings with all the benchmark parameters in the first part of Table 4.4.1.

4.4.2 Budgeted Continual Learning

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

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

To normalize for the effective C due to the overhead of associated extra forward passes to decide on the sampling, costly sampling strategies are allowed $C/2$ training iterations, where the exact calculation is left for the Appendix. That is to say, costly sampling strategies perform 200 training iterations for ImageNet2K and 50 training iterations for CGLM as the rest of the budget is for the extra forward passes. We report the performance of the five sampling strategies consisting of the inexpensive and the best performing costly sampling strategy (KMeans), presented in shades of blue, in Figures 4.4.2, 4.4.3, and 4.4.4 for DI-ImageNet2K, CI-ImageNet2K, and CGLM, respectively. Other methods are listed in the Appendix due to lack of space. We compare against a non-continual learning oracle that performs classical empirical risk minimization at every step on $\mathcal{T}_t = \cup_{r=1}^t \{(x_i^r, y_i^r)\}$ with a computational budget of $C \times t$, which we refer to as ERM-Naive; this is as opposed to the previously mentioned continual learning methods that have only C per step t spent equally over all steps. ERM-Naive acts as a training method with hindsight, spending the complete computational budget at once after collecting the full dataset. This acts as a very strong baseline against all continual learning methods. We report it in shades of red in the same figures. We also report the average accuracy, averaged over all time steps, for each sampling method in the yellow box in each figure.

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

Table 4.4.1: Experimental Details. The first block shows the various considered settings in the experiments section. The second block denotes the effective training iterations C for each class of methods due to their over head extra computation. The last block details the setup for our sensitivity analysis.

Conclusion. First, we observe that the top inexpensive sampling strategies perform very similarly to each other. This is consistent across settings, CI-ImageNet2K, and CGLM, on both ImageNet1K accuracy and Stream Accuracy. There are some advantages for Class-Balanced over other sampling strategies, *e.g.*, gaining an average accuracy of 2.5% over Uniform in DI-ImageNet2K. However, sampling strategies such as FIFO completely forget ImageNet1K (dark blue line), leading to poor performance over all three settings. Interestingly, costly sampling strategies perform significantly worse in CL performance over the simple Uniform sampling when subjected to an effectively similar computational budget. This observation is different from previous settings [Masana et al., 2022], as the additional computational overhead of costly sampling does not seem worthwhile to improve performance.

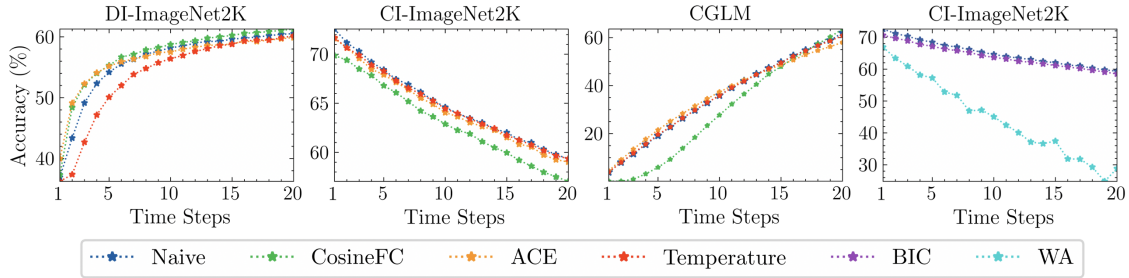


Figure 4.4.6: FC Layer Correction. Even though loss functions (CosineFC and ACE) might outperform Naive in the first few time steps, eventually Naive catches up. Overall, Naive consistently outperforms all considered calibration methods too, namely, BIC and WA.

2. Does Distillation Matter? We evaluate four well-known distillation losses in our benchmark,

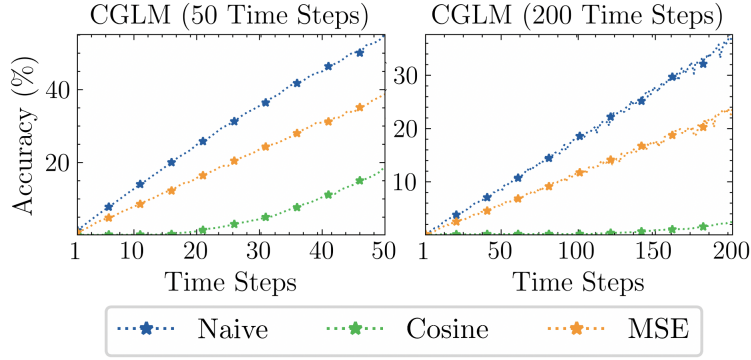


Figure 4.4.7: CGLM Distillation with Different Number of Time Steps. Under larger number of time steps, where total number of iterations is normalized accordingly, Naive outperforms distillation in both settings, namely, 50 and 200 time steps.

namely, Cosine, CrossEntropy, BCE, and MSE losses. Given that Class-Balanced is a simple inexpensive sampling procedure that performed slightly favorably, as highlighted in the previous section, we use it as a default sampling strategy from now onward, where the number of samples used per training step is equal over all classes. We refer to this basic approach with a cross entropy loss as Naive. To fairly factor in the overhead of an additional forward pass, distillation approaches are allowed $2C/3$ iterations compared to Naive with C training iterations. That is, the distillation losses perform 267 iterations for ImageNet2k and 67 iterations for CGLM compared to 400 and 100 iterations for Naive. We report the results for Cosine and MSE on DI-ImageNet2K, CI-ImageNet2K, and CGLM datasets in the first, second, and third rows of Figure 4.4.5, respectively. Other methods are left for the Appendix due to lack of space. Distillation methods are shown in shades of blue, whereas Naive is shown in shades of red. We report the average accuracy, averaged over all time steps, for each distillation method in the yellow box in each figure.

Conclusion. In all three settings, distillation methods underperform compared to Naive. Even in ImageNet1K Acc, which measures forgetting, Naive performs similarly or slightly better than all distillation methods in DI-ImageNet2K and CI-ImageNet2K streams. The results in Figure 4.4.5 show that the top distillation methods, such as MSE, perform only slightly worse compared to Naive (54.9 vs 55.9 on DI-ImageNet2K and 64 vs 64.9 on CI-ImageNet2K). However, in CGLM they perform significantly worse (26.4 compared to 35.7) due to the limited iterations. We attribute this to the fact that distillation methods often require a larger number of training samples, and thereof a large enough computational budget per time step.

3. Does FC Layer Correction Matter? We evaluate five FC layer correction approaches from two different families. A family of methods that modifies the FC layer directly, including

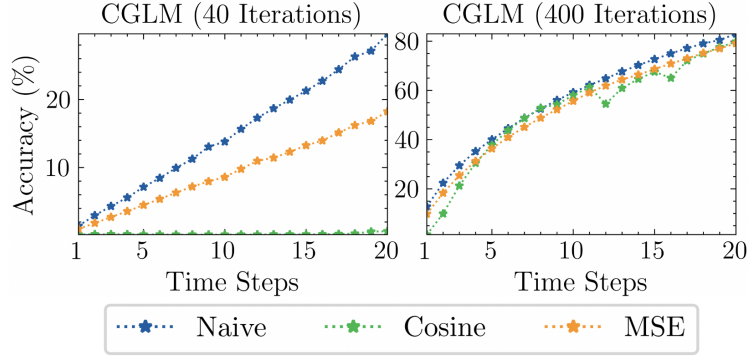


Figure 4.4.8: CGLM Distillation with Different Computational Budgets. Naive outperforms distillation methods under the restricted 40 and the larger 400 iterations (originally 100). Distillation methods become competitive when enough budget is available.

CosineFC [Hou et al., 2019] and ACE [Zeno et al., 2018, Mai et al., 2022, Caccia et al., 2022b]. The other family of methods applies post-training calibration including BiC [Wu et al., 2019], WA [Zhao et al., 2020], along with temperature scaling [Guo et al., 2017]. All methods employ Class-Balanced as a sampling strategy and compare against Naive (Class-balanced with cross entropy loss) with no corrections in FC layer. The first three subplots of Figure 4.4.6 correspond to comparisons of direct FC layer modification methods against Naive on DI-ImageNet2K, CI-ImageNet2K, and CGLM. Since calibration methods tailored for Class Incremental settings, in the rightmost plot of Figure 4.4.6, we report comparisons with Naive on CI-ImageNet2K. Since all FC layer corrections are with virtually no extra cost, the number of training iterations per time step is set to C , *i.e.*, 400 for ImageNet2K and 100 for CGLM.

Conclusion. No method consistently outperforms Naive in computationally budgeted continual learning. The first family of methods helps in DI-ImageNet2K, particularly in the initial steps due to class imbalance, but no method outperforms Naive in the CI-ImageNet2K set-up. Calibration-based methods, such as BIC, are somewhat competitive with Naive, but WA fails. Surprisingly, even under various FC correction approaches, all methods fail to outperform Naive in computationally budgeted continual learning.

4.4.3 Sensitivity Analysis

We have analyzed the performance of various CL methods under budgeted computation. We have consistently observed over a variety of settings on large-scale datasets that a simple method, *i.e.*, Naive, simply sampling with a Class-Balanced strategy and a cross entropy loss outperforms all existing methods. However, all reported results were for 20 time steps with $C = 400$ or

$C = 100$ training iterations for ImageNet2K and CGLM, respectively, in which expensive methods were normalized accordingly. Now, we analyze the sensitivity of our conclusions over different time steps and iterations C .

Does the Number of Time Steps Matter? Prior art, such as GDumb [Prabhu et al., 2020], found that the relative performance of CL methods changes drastically when the number of time steps is varied. Subsequently, we increased the number of time steps to 50 and 200 from 20, a more extreme setting than explored in recent works, while maintaining the same overall computational budget C eliminating any source of performance variation due to a different total computational budget. This is since per time step, the stream reveals fewer number of samples n with an increased number of time steps. We report experiments in the CGLM setting where Naive will receive only 40 and 10 iterations for the 50 and 200 time steps, respectively. We consider distillation approaches where they are permitted $2/3C$, which is 27 and 7 iterations, respectively, on the 50 and 200 time steps, respectively. Note that, in these settings, per time step, methods observe $2/3 \times 11.6K$ and $2/3 \times 2.9K$ samples, respectively. We leave the experiments on ImageNet2K for the Appendix due to space constraints. We compare two distillation methods against Naive in Figure 4.4.7. Other methods are presented in the Appendix.

Conclusion. We still consistently observe that Naive outperforms all distillation methods on both the 50 and the 200 time steps. Moreover, the relative performance across distillation methods is preserved similarly to the 20 time steps setup. That is, our conclusions are largely robust under different number of time steps. This is contrary to the observation of the prior art [Prabhu et al., 2020], this is because unlike our setting, [Prabhu et al., 2020] does not scale the compute with increased number of time steps.

Does the Compute Budget Matter? Finally, we explore the impact of changing the computational budget on the performance of different distillation methods on CGLM under 20 time steps. We study two scenarios, one where the budget is increased to $C = 400$ and where it is reduced to $C = 40$, originally $C = 100$ for CGLM. Hence, distillation would be allocated 267 and 27 iterations in this setting, respectively. As shown in Figure 4.4.1, the higher budget setting allows approximately a full pass per time step over all stored data. We leave the experiments on ImageNet2K for the Appendix. We compare two distillation methods with Naive in Figure 4.4.8. The remaining methods are presented in the Appendix.

Conclusion. Again, we observe that Naive outperforms all distillation methods in both increased and decreased compute budget settings. The final gap between MSE distillation and Naive is 11.41% for $C = 40$, this gap is reduced to 3.85% for $C = 400$. Surprisingly, even with increased compute

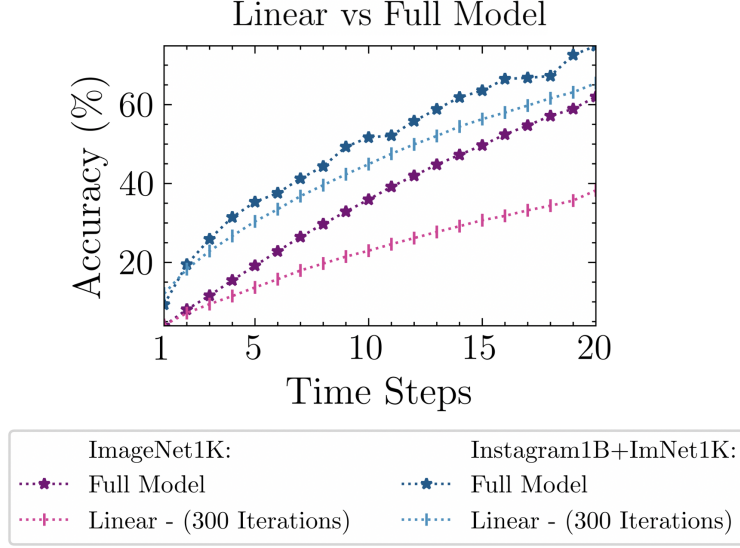


Figure 4.4.9: Linear vs Full Model Training. Performing Linear fine tuning allows to leveraging the computational budget efficiently improving the gap compared to full model training particularly for better pretrained models, *e.g.*, Instagram1B+ImNet1K.

budget, distillation methods still fall behind Naive. However, the reduced gap in performance compared to that of Naive is a strong indication that the reason behind the failure of distillation methods is indeed the limited computation.

4.4.4 Exploring Partial Training

We now investigate the reallocation of the computational budget through partial training of the model, which is a model expansion method that involves pre-selecting the subnetwork to be trained. This approach is more computationally efficient, especially on large-scale datasets. The top (FC) layer is the smallest part of the network that can be retrained. We compare partial training of the network, *i.e.*, FC layer only, to training the full model (Naive) using two different model initializations, ImageNet1K pretraining [He et al., 2016] and Instagram1B+ImageNet1K pretraining [Mahajan et al., 2018]. Note that Instagram1B+ImageNet1K is a stronger pretrained model, with better feature representations. To normalize the computation for the FC partial training, we permit $3C$ training iterations compared to full model training (Naive) with C training iterations. Hence, CGLM FC partial training performs 300 iterations compared to 100 iterations for training Naive. We present our results in Figure 4.4.9, where the shades of purple and blue represent models trained from pretrained ImageNet1K and Instagram1B+ImageNet1K models, respectively.

Conclusion. There exists a gap between full model training and partial FC layer training

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

4.5 Conclusion

In this chapter, we concluded that existing CL algorithms, such as sampling strategies, distillation, and FC layer corrections, fail in a budgeted computational setup. Simple Naive methods based on experience replay outperform all the considered CL methods. This conclusion was persistent even under various computational budgets and an increased number of time steps. We further improve continual learning methods by incorporating a fast learning mechanism, using the online context. Hence, we investigate effective methods for online continual learning in the next chapter.

5

Online Continual Learning Without the Storage Constraint

Contents

5.1	Introduction	61
5.2	Related Work	62
5.3	Our Approach: Adaptive Continual Memory	64
5.3.1	Computational Cost and Storage Considerations	66
5.4	Experiments	66
5.4.1	Comparison of ACM with Online Continual learning Approaches	68
5.4.2	Comparison of ACM with Approaches Leveraging a Fixed Backbone	70
5.4.3	Analyzing Our Method: Adaptive Continual Memory	71
5.4.4	Discussion	72
5.5	Conclusion	73

Abstract

Traditional online continual learning (OCL) research has primarily focused on mitigating catastrophic forgetting with fixed and limited storage allocation throughout an agent’s lifetime. However, a broad range of real-world applications are primarily constrained by computational costs rather than storage limitations. In this chapter, we target such applications, investigating the online continual learning problem under relaxed storage constraints and limited computational budgets. We contribute a simple algorithm, which updates a kNN classifier continually along with a fixed, pretrained feature extractor. We selected this algorithm due to its exceptional suitability for online continual learning. It can adapt to rapidly changing streams, has zero stability gap, operates within tiny computational budgets, has low storage requirements by only storing features, and has a consistency property: It *never forgets* previously seen data. These attributes yield significant improvements, allowing our proposed algorithm to outperform existing methods by over 20% in accuracy on two large-scale OCL datasets: Continual LOCalization (CLOC) with 39M images and 712 classes and Continual Google Landmarks V2 (CGLM) with 580K images and 10,788 classes, even when existing methods retain all previously seen images. Furthermore, we achieve this superior performance with considerably reduced computational and storage expenses.

5.1 Introduction

In online continual learning (OCL), the focus is on rapid adaptation to incoming data rather than preserving information about all past knowledge. That is, the algorithm may gracefully forget past knowledge if it does not benefit future predictions. In this chapter, we propose a system based on approximate k-nearest neighbor (kNN) algorithm [Malkov and Yashunin, 2018] for online continual learning, following its five desirable properties. i) Approximate kNNs are inherently incremental algorithms with explicit insert and retrieve operations allowing it to rapidly adapt to incoming data; ii) With a suitable representation, approximate kNN algorithms are exceptionally effective models at large-scale [Efros, 2017] iii) Approximate kNN algorithms are computationally cheap, with a graceful logarithmic scaling of computation despite compactly storing and using all past samples; iv) kNN does not forget past data. In other words, if a data point from history is queried again, the query yields the same label; v) It has no stability gap [De Lange et al., 2022]¹.

Ideally, a continual learner should learn and update the feature representations over time. We defer this interesting problem to future work and instead show that combined with kNNs, it is possible for feature representations pre-trained on a relatively smaller dataset (Imagenet1K) to reasonably tackle complex tasks such as geolocalization over 39 million images (CLOC), and long-tailed fine-grained classification (CGLM). While this does not solve the underlying continual representation learning problem, it does show the effectiveness of a simple method on large-scale online continual learning problems, demonstrating viability to many real-world applications. Additionally, our approach overcomes a significant limitation of existing gradient-descent-based methods: the ability to learn from a single example. Updating a deep network for every incoming sample is computationally infeasible. In contrast, a kNN can efficiently learn from this sample, enabling rapid adaptation. We argue that the capacity to adapt to a single example while leveraging all past seen data is essential for truly online operation, allowing our simple method to outperform existing continual learning baselines.

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

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

¹We expand the discussion on these properties in detail in Section 5.3

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

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

5.2 Related Work

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

Methods. Traditional methods of adapting to concept drift [Gama et al., 2014] include a variety of approaches based on SVMs [Laskov et al., 2006, Zheng et al., 2013], random forests [Gomes et al., 2017, Ristin et al., 2015, Mourtada et al., 2019], and other models [Oza and Russell, 2001, Mensink et al., 2013]. They offer incremental additional and querying properties, most similar to our

Table 5.2.1: Breakdown of popular OCL systems, with key contributions in **red**. Most methods focus on sampling techniques for storing datapoints, which cannot transfer here as we store all past samples.

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

method, but have not been compared with recent continual learning approaches [Ostapenko et al., 2022, Hayes and Kanan, 2020, Hayes et al., 2019]. We perform extensive comparisons with them.

The (online) continual learning methods designed for deep networks are typically based on experience replay [Chaudhry et al., 2019b] and change a subset of the three aspects summarized in Table 5.2.1: (i) the loss function used for learning, (ii) the algorithm to sample points into the replay buffer, and (iii) the algorithm to sample a batch from the replay buffer. Methods to sample points into the replay buffer include GSS [Aljundi et al., 2019b], RingBuffer [Chaudhry et al., 2019b], class-balanced reservoir [Chrysakis and Moens, 2020], greedy balancing [Prabhu et al., 2020], rainbow memory [Bang et al., 2021], herding [Rebuffi et al., 2017b], coreset selection [Yoon et al., 2022], information-theoretic reservoir [Sun et al., 2022], and samplewise importance [Koh et al., 2022]. These approaches do not apply to our setting because we simply remove the storage constraint. Approaches to sampling batches from the replay buffer include MIR [Aljundi et al., 2019a], ASER [Shim et al., 2021], and AML [Caccia et al., 2022a]. These require mining hard negatives or performing additional updates for importance sampling over the stored data, which face scaling issues to large-scale storage as in our work. We compare with some of the above approaches, including ER as proposed in Cai et al. [2021], that finetune the backbone deep network with one gradient update for incoming data, with unrestricted access to past samples for replay.

Pretrained representations. Pretrained representations [Yuan et al., 2021, Caron et al., 2021, Chen et al., 2021a, Ali et al., 2021] have been utilized as initializations for continual learning, but in settings with harsh constraints on memory [Wu et al., 2022, Ostapenko et al., 2022]. Inspired by Ostapenko et al. [2022], we additionally explore suitability of different pretrained representations. Another emerging direction for using pretrained models in continual learning has been prompt-

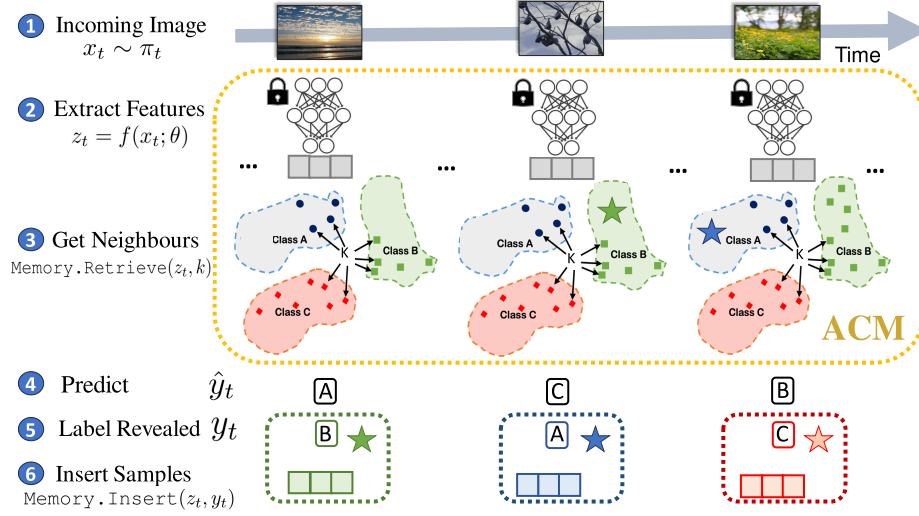


Figure 5.3.1: Adaptive Continual Memory (ACM) performs `Memory.Retrieve` and `Memory.Insert` on new incoming samples, extracted by a fixed, pretrained deep network.

tuning as it produces accurate classifiers while being computationally efficient [Wang et al., 2022c,b, Chen et al., 2023]. However, Janson et al. [2022] show that simple NCM classification outperforms complex prompt tuning strategies. Lastly, the direction most similar to ours is methods which use kNN classifiers alongside deep networks for classification [Nakata et al., 2022, Iscen et al., 2022]. We operate in significantly different setting and constraints, use far weaker pretrained representations (ImageNet1K) and benchmark on far larger online classification datasets.

5.3 Our Approach: Adaptive Continual Memory

We use pre-trained feature representations and only learn using the approximate k-nearest neighbor algorithm. Hence, our algorithm is rather simple. We refer to our algorithm as Adaptive Continual Memory (ACM) and refer to the kNN neighbour set as Memory. At each time step, our continual learner performs the following steps:

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

6. Learner inserts new data: `Memory.Insert( $z_t, y_t$ )`.

We summarize this approach in Figure 5.3.1. Before presenting further implementation details, we discuss four properties of this method in detail.

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

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

Zero Stability Gap. When learning new points, traditional continual learning algorithms first drop in performance on past samples due to a large drift from current minima and gradually recover performance on convergence. This phenomena is called as the stability gap [De Lange et al., 2022]. Approximate kNN inherently does not have this optimization issue, hence enjoy zero stability gap.

Efficient Online Hyperparameter Optimization. Hyperparameter optimization is a critical issue during the online continual learning phase because, as distributions shifts, hyperparameters must be recalibrated. Selecting hyperparameters relevant to optimization like learning rate and batch size, can be nuanced; an incorrect choice has the potential to indefinitely impede future performance. Common strategies include executing multiple simultaneous online learning tasks using diverse parameters [Cai et al., 2021]. However, this can be prohibitively resource-intensive. In contrast, our method has a single hyperparameter (k), which only affects the immediate prediction, and can be recalibrated during the online continual learning phase with minimal computational cost. To determine the optimal number of neighbors (k) for our algorithm, we implement an online hyperparameter search. First, we retrieve and cache the 512 nearest neighbors for each data point, sorted by proximity to avoid repeated queries. We then perform a brute-force search by testing k values that are powers of two (e.g., 256, 128, 64, 32, 16, 8, 4, 2, 1) within this ranked list. For each k , we use the nearest k neighbors to predict the next point for a small set of incoming datapoints, simulating the arrival of previous samples and measuring prediction accuracy. Through this iterative

process, we identify the k that achieves the highest accuracy and apply it to the next 2048 data points and repeat this process. This method ensures efficient on-the-fly hyperparameter tuning of k .

5.3.1 Computational Cost and Storage Considerations

In the presented algorithm above for our method, feature extraction (step 2) and prediction (step 4) have a fixed overhead cost. However, nearest-neighbour retrieval (step 3) and inserting new data (step 6) can have high computational costs if done naively. However, literature in approximate k-nearest neighbours [Shakhnarovich et al., 2006] has shown that we can achieve high performance while significantly reducing computational complexity from linear $O(n)$ to logarithmic $O(\log n)$, where n is the number of data points in memory. By switching from exact kNN to HNSW-kNN, we reduce the comparisons from 30 million to a few hundred, while maintaining a similar accuracy. We utilize the HNSW algorithm from HNSWlib because of its high accuracy, approximate guarantees and practically fast runtime on ANN Benchmarks [Aumüller et al., 2020]. We perform a wall-clock time analysis quantifying this speed in Section 5.4.3.

5.4 Experiments

We first describe our experimental setup below and then provide comprehensive comparisons of our method against existing incremental learning approaches.

Datasets. We used a subset of Google Landmarks V2 and YFCC-100M datasets for online image classification. These datasets are ordered by the timestamps of image uploads, and our task is to predict the label of incoming images. We followed the online continual learning (OCL) protocol as described in Chaudhry et al. [2019a]: We first tune hyperparameters of all OCL algorithms on a pretraining set, continually train the methods on the online training set while measuring rapid adaptation performance, and finally evaluated information retention on a unseen test set. Further dataset details are available in the Appendix.

Metrics. We follow Cai et al. [2021], measuring average online accuracy until the current timestep t (a_t) as a metric for measuring rapid adaptation, given by $a_t = 1/t \sum_{i=1}^t \mathbb{1}_{y_i=\hat{y}_i}$ where $\mathbb{1}_{(\cdot)}$ is the indicator function. We additionally measure information retention, i.e. mitigating catastrophic forgetting, after online training on unseen samples from a test set. Formally, information retention for h timesteps (IR_h) at time T , is defined as $IR_h = 1/h \sum_{t=T-h}^T \mathbb{1}_{y_t=\hat{y}_t}$.

Computational Budget and Pretraining. To ensure fairness among compared methods, we restrict the computational budget for all methods to one gradient update using the naive ER method.

All methods were allowed to access all past samples with no storage restrictions. All methods started with a pretrained ResNet50 model on the ImageNet1K dataset for fairness. Note that we select the ImageNet1K pretrained ResNet50 because despite being a good initialization, is not sufficient by itself to perform well on the selected continual learning benchmarks. We select a fine grained landmark recognition benchmark over 10788 categories (CGLM), and a harder geolocalization task over a far larger dataset of 39 million samples (CLOC).

OCL Approaches. We compared five popular OCL approaches as described in [Ghunaim et al., 2023] on the CLOC dataset. For CGLM, we compare among the top two performing methods from CLOC. We provide a brief summary of the approaches:

1. *ER* [Cai et al., 2021]: We use vanilla ER without PoLRS and ADRep [Cai et al., 2021] as they did not improve performance.
2. *MIR* [Aljundi et al., 2019a]: It additionally uses MIR as the selection mechanism for choosing samples for training (in a task-free manner).
3. *ACE* [Caccia et al., 2022a]: ACE loss is used instead of cross entropy to reduce class interference.
4. *LwF* [Li et al., 2017a]: It adds a distillation loss to promote information retention.
5. *RWalk* [Chaudhry et al., 2018]: This method adds a regularization term based on Fisher information matrix and optimization-path based importance scores. We treat each incoming batch of samples as a new task.

Training Details for Baselines. CGLM has the same optimal hyperparameters for CLOC. The ResNet50 model was continually updated using the hyperparameters outlined in [Ghunaim et al., 2023]. We used a batch size of 64 for CGLM and 128 for CLOC to control the computational costs. Predictions are made on the next batch of 64/128 samples for CGLM/CLOC dataset respectively using the latest model. The model uses a batch size of 128/256 respectively for training, with the remaining batch used for replaying samples from storage.

Fixed Feature Extractor based Approaches. In this section, we ablate capabilities specifically contributed by kNN in ACM compared to other continual learning methods which use a common fixed feature extractor. However, the compared baselines do not have the consistency property provided by ACM, ablating the contribution of this property. We use a 2 layer embedder MLP to project the 2048 dimensional features of ResNet to 256 dimensions using the pretrain set. This adapts the pretrained features to domain of the tested dataset, while providing compact storage and

increasing processing speed. All below methods operate on these fixed 256 dimensional features for fairness and operate on features normalized by an online scaler for best performance, with one sample incoming at a timestep. Note that the full model continual learning methods did not benefit significantly by this additional adaptation step. We detail the approaches below:

1. *Nearest Class Mean (NCM)* [Mensink et al., 2013, Rebuffi et al., 2017b, Janson et al., 2022]: This method maintains a mean feature for each class and classifies new samples by measuring cosine similarity with the mean feature.
2. *Streaming LDA (SLDA)* [Hayes et al., 2019]: This is the current state-of-the-art online continual method using fixed-feature extractors. We use the code provided by the authors with $1e - 4$ being optimal shrinkage parameter.
3. *Incremental Logistic Classification* [Tsai et al., 2014]: We include traditional incremental logistic classification. We use scikit-learn `SGDClassifier` with Logistic loss.
4. *Incremental SVM* [Laskov et al., 2006]: We include traditional online support vector classification. We use scikit-learn `SGDClassifier` with Hinge loss.
5. *Adaptive Random Forests (ARF)* [Gomes et al., 2017]: We chose the best performing method from benchmarks provided by the River library ² called Adaptive Random Forests.
6. *Eigen Memory Trees (EMT)* [Rucker et al., 2022]: This is the current state-of-the-art incremental learning method using Trees, outperforming Sun et al. [2019] by large margins.

5.4.1 Comparison of ACM with Online Continual learning Approaches

Online adaptation. We compare the average online accuracy of ACM to state-of-the-art approaches on CGLM and CLOC datasets in Figure 5.4.1. We observe that ACM significantly outperforms previous methods, achieving a 35% and 5% higher absolute accuracy margin on CGLM and CLOC, respectively. This improvement is due to the capability of ACM to rapidly learn new information.

Information retention. We compare backward transfer of ACM to current state-of-the-art approaches on CGLM and CLOC in Figure 5.4.1. We find that ACM preserves past information much better than existing approaches, achieving 20% higher accuracy on both datasets. On the larger CLOC dataset, we discover that existing methods catastrophically forget nearly all past knowledge, while ACM maintains a fairly high cumulative accuracy across past timesteps. This

²<https://riverml.xyz/0.19.0/>

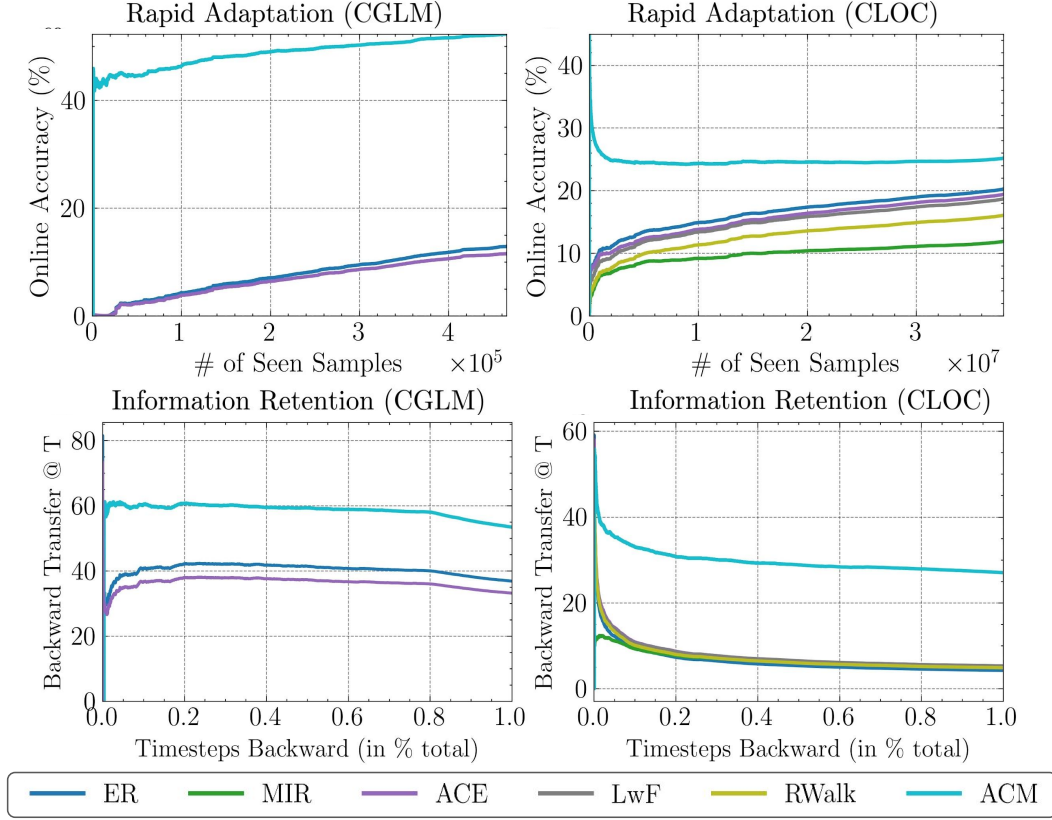


Figure 5.4.1: Online Continual Learning Performance. We observe that ACM outperforms existing methods by a large margin despite being far cheaper computationally. Traditional methods perform poorly given unrestricted access to past seen data indicating continual learning is a hard problem even without storage constraints.

highlights the advantages of the consistency property, allowing perfect recall of past train samples and subsequently, good generalization ability on similar unseen test samples to past data.

Moreover, comparing the performance of methods to those in the *fast stream* from Ghunaim et al. [2023], it becomes evident that removing memory restrictions (from 40,000 samples) did not substantially alter the performance of traditional OCL methods. This emphasizes that online continual learning with limited computation remains challenging even without storage constraints.

Key Takeaways. ACM demonstrates significantly better performance in both rapid adaptation and information retention when compared to popular continual learning algorithms which can update the base deep network on any of the past seen samples with no restrictions. We additionally highlight that ACM has a substantially lesser computational cost compared to traditional OCL methods.

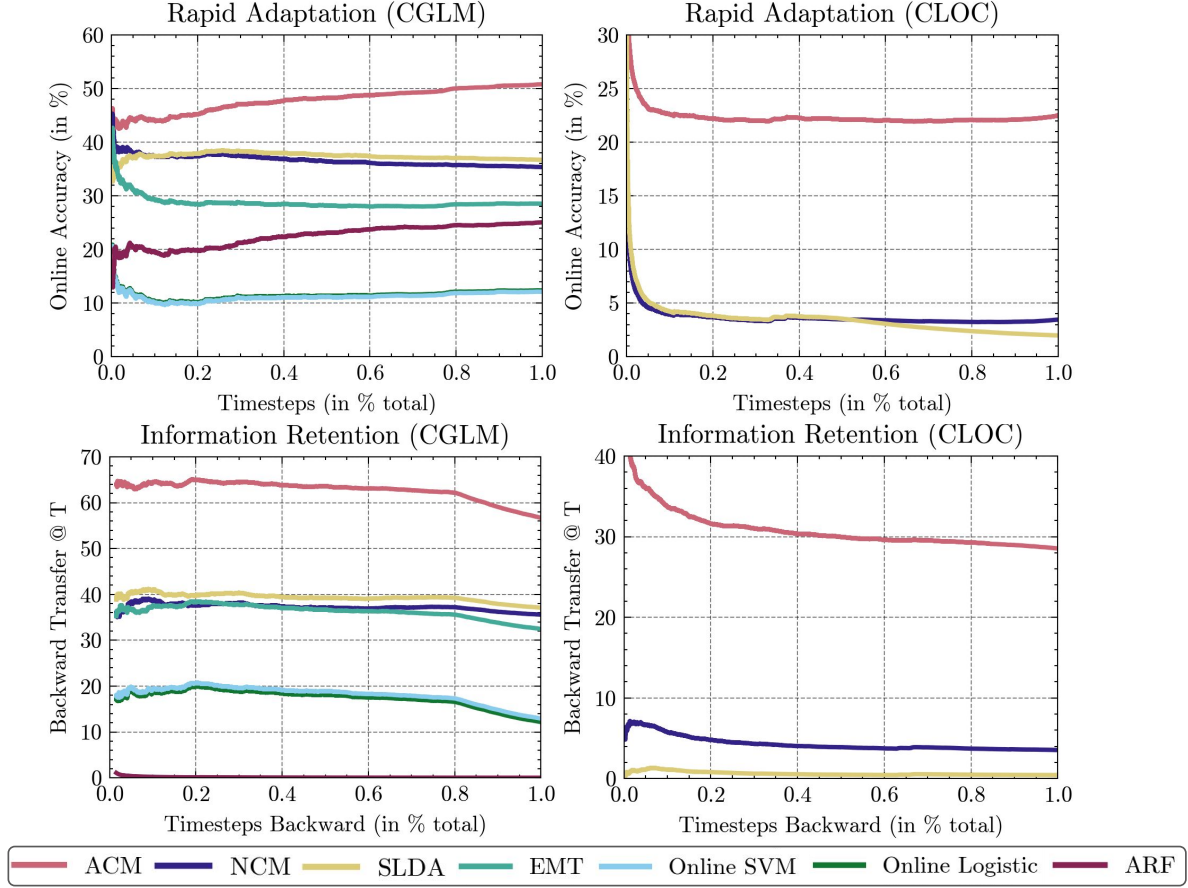


Figure 5.4.2: Online Continual Learning Performance. ACM outperforms existing methods which leverage a fixed backbone. This highlights the importance of preserving the consistency property in online continual learning. The collapse of other approaches for the CLOC dataset indicates the hardness of the continual learning scenario.

5.4.2 Comparison of ACM with Approaches Leveraging a Fixed Backbone

Online adaptation. We compare average online accuracy of ACM against recent continual learning approaches that also employ a fixed feature extractor on CGLM and CLOC datasets in Figure 5.4.2. We find that ACM outperforms these alternative approaches by significant margins, achieving 10% and 20% higher absolute accuracy on CGLM and CLOC respectively. All approaches here can rapidly adapt to every incoming sample, achieving higher accuracy than traditional OCL approaches. However, ACM can additionally utilize past seen samples when necessary. Notably, in CLOC, the best alternative approaches collapse to random performance, highlighting that pretrained feature representations are not sufficient, and the effectiveness of kNN despite its simplicity.

Information retention. We compare backward transfer of ACM compared to other fixed-feature

based OCL approaches on CGLM and CLOC dataset in Figure 5.4.2. We observe that ACM outperforms other approaches by 20% on both datasets, demonstrating its remarkable ability to preserve past knowledge over time. Even after 39 million online updates on the CLOC, ACM preserves information from the earliest samples. In contrast, existing fixed-feature online continual learning methods collapse to random performance.

Key Takeaways. The impressive performance of ACM is evident in both rapid adaptation and information retention even amongst latest approaches which similarly use a fixed feature extractor. This further demonstrates the impact of preserving consistency.

A Note on Time. ACM and NCM were the fastest approaches among the compared methods. All other approaches were considerably slower, with a 5 to 100 fold increase in runtime compared with NCM or ACM. However, this could be attributed to codebases, although we used open-source, fast libraries such as River and Scikit-learn.

5.4.3 Analyzing Our Method: Adaptive Continual Memory

Contribution of kNN. Here, we aim to disentangle the benefit provided by the pretrained backbone and the domain tuning by the MLP, with the contribution of kNN to ACM. To test this, we perform online continual learning by replacing the kNN with the fixed MLP classifier. The performance obtained on rapid adaptation on ablating the kNN will be the effect of strong backbone and first session tuning using MLP on pretrain set [Panos et al., 2023]. We conducted this experiment using two additional pretrained backbones stronger than our ResNet50 trained on ImageNet1K: A ResNet50 trained on Instagram 1B dataset and the best DINO model XCIT-DINO trained on Imagenet1K to vary the pretraining dataset and architecture.

The results presented in Figure 5.4.3 (right). We observe that removing the kNN for classification leads to a drastic decline in performance, indicating that kNN is the primary driver of performance, with performance gains of 20-30%. The decline in performance, losing over 10%, compared to initialization is attributable to distribution shift across time. This is consistently seen across model architectures, indicating that CGLM remains a challenging task with a fixed feature extractor despite backbones far stronger than ResNet50 trained on ImageNet1K.

These findings suggest that kNN is the primary reason for rapid adaptation gains to distribution shifts. Having high-quality feature representations alone or first-session adaptation is insufficient for a satisfactory online continual learning performance.

Time Overhead of ACM. We provide a practical analysis of time to ground the logarithmic computational complexity of ACM. Figure 5.4.3 (left) provides insights into the wall-clock time

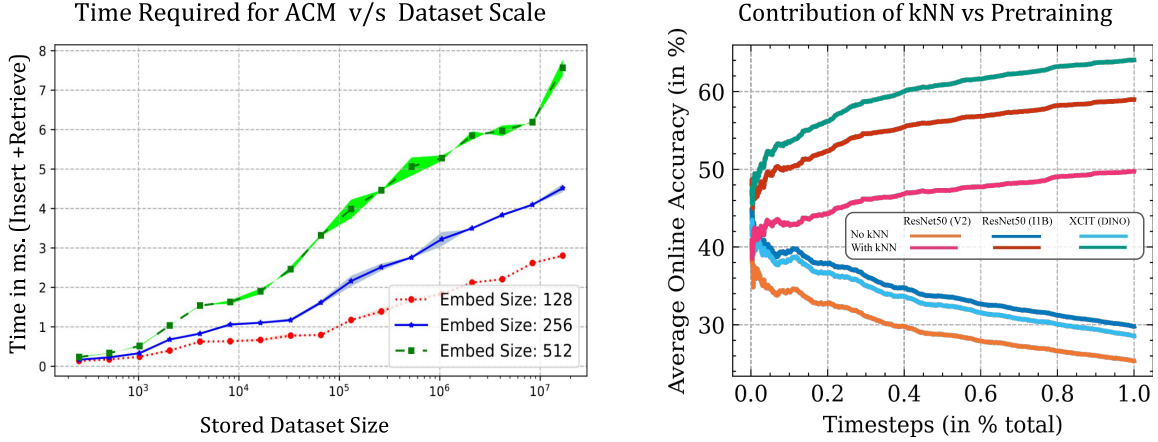


Figure 5.4.3: Left: Wall clock time overhead of using ACM Memory after feature extraction (x-axis is log-scaled) on a 16-core i7 CPU server. The time increases logarithmically with dataset size, with a 8ms overhead at 40M samples. **Right:** Contribution of kNN and contribution of the original backbone along with the MLP. We observe most of the performance is attributable to the kNN.

required for the overhead cost imposed by ACM when scaling to datasets of 40 million samples. We observe that the computational overhead while using ACM scales logarithmically, reaching a maximum of approximately 5 milliseconds for 256 dimensional embeddings. In comparison, the time required for the classification of a single sample for deep models like ResNet50 is approximately 10 milliseconds on an Intel 16-core CPU. It’s important to note that when using ACM, the total inference cost of ACM inference would be 15 milliseconds, representing a 50% inference overhead as a tradeoff for high rapid adaptation and information retention performance.

5.4.4 Discussion

A notable limitation of our approach is its dependency on the presence of pretrained features. Consequently, our method may not be appropriate for situations where such features are unavailable. While this limitation is important to acknowledge, it doesn’t diminish the relevance of our approach in situations where pretrained features are available. Our method can be applied effectively in a wide range of visual continual learning scenarios. We have been selective in our choice of models and experiments, opting for pretrained models on ImageNet1K rather than larger models like CLIP or DINOv2 to demonstrate this applicability. Moreover, we tested our approach on more complex datasets like Continual YFCC-100M, which is 39 times larger than ImageNet1K and includes significantly more challenging geolocation tasks.

Memory constraints are often linked to privacy concerns. However, it’s crucial to understand that merely avoiding data storage does not guarantee privacy in continual learning. Given the

tendency of deep neural networks to memorize information, ensuring privacy becomes a much bigger challenge. While a privacy-preserving adaptation of our method is beyond this chapter’s scope, one can employ differentially private feature extractors, as suggested by [Ma et al., 2022], to build privacy-conscious ACM models. We conjecture that as more advanced privacy-preserving feature extractors become available, privacy concerns can be addressed in parallel.

Finally, to contextualize the computational budget with tangible figures, we envision a hypothetical system necessitating real-time operation on a video stream, facilitated by a 16-core i7 CPU server. Given a feature size of 256 and drawing insights from Figure 5.4.3, our method is projected to sustain real-time processing at 30 frames per second for an impressive span of up to 71 years, without necessitating further optimization. Such a configuration would consume approximately 900 GB of storage annually, translating to a cost of roughly \$20 per year, as indicated in Table 1.2.1. Thus, the ACM stands out as practical, even for the prolonged deployment of continual learning systems.

5.5 Conclusion

In this chapter, we explored online continual learning without any storage constraints. Our reformulation emerges from an analysis of both economic and computational attributes of computing systems. We introduce an approximate kNN-based approach that stores the entirety of the data, adapts on a per-sample basis at each timestep, and still retains computational efficiency. Upon evaluation on large-scale OCL benchmarks, our system yields significant improvements over existing methods. Our approach is computationally cheap and scales gracefully to large-scale datasets. This covers the efforts in this thesis to discover better algorithms for continual learning in a computationally budgeted scenario. In the next chapter, we explore metrics useful for continual learning.

6

Rapid Adaptation in Online Continual Learning: Are We Evaluating It Right

Contents

6.1	Introduction	76
6.2	OCL Evaluation: A Review	77
6.3	Correcting Evaluations of Rapid Adaptation	80
6.3.1	Evaluating OCL: A Problem Formulation	81
6.3.2	Isolating and Quantifying Label Correlations	81
6.3.3	Can Deep Networks Learn Label Correlations?	82
6.3.4	Exploring Properties of OverAdapt	83
6.3.5	Removing Correlations to Evaluate Adaptation	86
6.4	OCL Evaluation by Near-Future Accuracy	87
6.4.1	Experimental Setup	88
6.4.2	Results: Varying Stream Speeds	89
6.4.3	Further Discussion	90
6.5	Conclusion	91

Abstract

We revisit the common practice of evaluating adaptation of Online Continual Learning (OCL) algorithms through the metric of *online accuracy*, which measures the accuracy of the model on the immediate next few samples. However, we show that this metric is unreliable, as even vacuous blind classifiers, which do not use input images for prediction, can achieve unrealistically high online accuracy by exploiting spurious label correlations in the data stream. Our study reveals that existing OCL algorithms can also achieve high online accuracy, but perform poorly in retaining useful information, suggesting that they unintentionally learn spurious label correlations. To address this issue, we propose a novel metric for measuring adaptation based on the accuracy on the *near-future* samples, where spurious correlations are removed. We benchmark existing OCL approaches using our proposed metric on large-scale datasets under various computational budgets and find that better generalization can be achieved by retaining and reusing past seen information. We believe that our proposed metric can aid in the development of truly adaptive OCL methods.

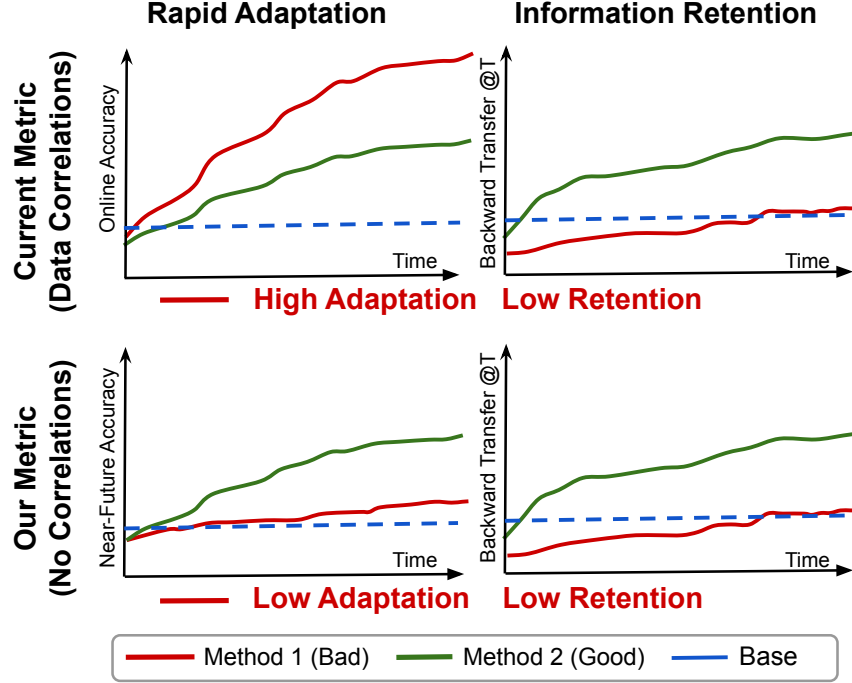


Figure 6.1.1: Effect of Spurious Correlations. Current methods in OCL perform well on rapid adaptation but suffer from abysmal information retention (top). We show that these trends are due to idiosyncrasies in the data stream such as label correlations that favor methods that inadvertently overfit to the latest data. Using our proposed metric that removes these correlations (bottom), different sets of methods achieve good performance, with high information retention being key for achieving high adaptability.

6.1 Introduction

In this chapter, we demonstrate that the conventional metric of measuring online accuracy may lead to misleading conclusions about the adaptation capabilities of OCL algorithms due to the idiosyncrasies of the data stream. To support our claim, we first demonstrate that a simple blind classifier that relies solely on spurious label correlations in the stream achieves unrealistically high online accuracy on large-scale OCL datasets. Furthermore, we show that this behaviour can be exhibited by deep OCL algorithms, where we propose *OverAdapt* which achieves state-of-the-art performance in online accuracy despite the very poor performance in information retention. We confirm the findings on two large-scale OCL datasets collected from different sources. This indicates that the problem is not limited to a specific dataset, but may be a general concern.

To overcome this issue, we introduce a new metric called *near-future accuracy* that measures the adaptation capabilities of OCL algorithms by evaluating the accuracy on samples after a shift of $\mathcal{S}$ steps. We choose the minimum shift $\mathcal{S}$ which ensures that the samples no longer have spurious

label correlations with the current training data. Choosing the minimum value helps minimize any distribution drift between the training and evaluation samples. The case $\mathcal{S} = 0$ recovers the online accuracy metric in case of no spurious correlations. We observe a significant drop in performance on this metric compared to their online accuracy with several state-of-the-art OCL algorithms. This suggests that these algorithms inadvertently rely on label correlations in the stream for predictions. On the contrary, we find that algorithms designed primarily to improve information retention perform exceptionally well in our evaluation strategy. This suggests that a better generalization can be achieved by effectively retaining and reusing past information. Figure 6.1.1 illustrates that the best algorithms are those that have little discrepancy between information retention and adaptation. These results challenge current beliefs and demonstrate the importance of using a more precise measure of adaptation in evaluating OCL algorithms.

Overall, our contributions can be summarized as follows:

- We demonstrate that popular metrics like online accuracy can be unreliable in measuring rapid adaptation due to the presence of spurious label correlations in OCL datasets.
- We introduce a novel baseline, OverAdapt, which achieves high online accuracy by relying on label correlations, despite forgetting almost all previously seen data.
- We propose a novel metric that evaluates the adaptation capabilities of OCL algorithms by measuring accuracy on near-future samples. We select the smallest value of $\mathcal{S}$ that removes label correlations and use this metric to evaluate state-of-the-art OCL methods.
- Our findings show that existing OCL algorithms perform poorly when evaluated using our proposed evaluation strategy. Algorithms which prevent catastrophic forgetting perform significantly better, suggesting that better adaptation performance can be achieved by retaining and reusing past seen information.

Our results challenge the current emphasis on adapting to the latest samples and demonstrate the importance of seeking a more precise measure of adaptation. We note that our contributions are robust, with our findings valid across various OCL approaches, optimization strategies, and large-scale datasets from different sources.

6.2 OCL Evaluation: A Review

We start with an overview of benchmarks in OCL across different properties of interest. For a more detailed review of the literature, we refer the reader to [Wang et al., 2023].

Table 6.2.1: Properties of OCL Benchmarks. We list benchmarks in the field of OCL along with the works which introduced them. We compare them across six properties: Realistic Data Streams (DO), No Storage Constraints (SC), Rapid Adaptation (RA), Information Retention (IR), Restricts Computational Budgets (CB), Evaluates on long stream sizes (LB), Leverages Pretrained Models (PT). Note that ✓ is better than ✗ in all columns.

DataOrder	Benchmark	DO	SC	RA	IR	CB	LB	PT
Task-Inc.	A-GEM[Chaudhry et al., 2019a]	✗	✗	✗	✓	✗	✓	✗
Class-Inc.	MIR[Aljundi et al., 2019a]	✗	✗	✗	✓	✗	✓	✗
	DER[Buzzega et al., 2020]							
	GSS[Aljundi et al., 2019b]							
Blurry	RM [Bang et al., 2021]	✗	✗	✗	✓	✗	✓	✗
	CLIB[Koh et al., 2022]							
	CLEAR[Lin et al., 2021]	✓	✗	✓	✓	✗	✗	✓
Natural	BudgetCL[Prabhu et al., 2023b]	✗	✓	✗	✓	✓	✓	✓
	CLOC[Cai et al., 2021]	✓	✗	✓	✗	✗	✓	✗
	DelayOCL[Ghunaim et al., 2023]	✓	✗	✓	✗	✓	✓	✓
	ACM[Prabhu et al., 2023a]	✓	✓	✓	✓	✓	✓	✓

1. Realistic Data Streams (DO). The OCL community has been moving towards increasingly realistic benchmarks in terms of data ordering and task setup. Early works [Lopez-Paz and Ranzato, 2017, Chaudhry et al., 2019a] assumed access to which subset of classes a test sample is from (task-incremental setup). Subsequent work removed this constraint, requiring models to predict across all seen classes so far (class-incremental setup) [Aljundi et al., 2019b,a]. However, these works used incoming samples from disjoint data distributions with artificial task boundaries, resulting in unrealistic ordering. Recent works [Bang et al., 2021, Koh et al., 2022, Prabhu et al., 2023b] improved the ordering by mixing samples from different disjoint tasks. Latest works [Cai et al., 2021, Prabhu et al., 2023a, Lin et al., 2021] eliminate the need for generating artificial data streams simply by using real-world timestamps to order the data stream. Our work focuses on these real-world data streams on large-scale datasets.

2. Towards No Storage Constraints (SC). The majority of prior benchmarks on continual learning constrain the problem by prohibiting access to previously received data [Li et al., 2017a, Kirkpatrick et al., 2017], with only a small portion of data allowed to be stored in memory [Lopez-Paz and Ranzato, 2017, Chaudhry et al., 2018, Buzzega et al., 2020, Aljundi et al., 2019b, Koh et al., 2022, Lin et al., 2021]. This constraint is often justified by two reasons: (i) storage space is expensive and (ii) access to previous data is prohibited due to privacy constraints. However, recent work [Prabhu et al., 2023a,b] shows critical shortcomings of these justifications. They show that (a) storage costs are negligible compared to the computational costs of training a model, and (b) simply restricting access to previous data does not address privacy considerations, as samples can be reconstructed from model weights [Haim et al., 2022] or detectably change the model output

[Shokri et al., 2017]. Interestingly, Goel *et al.* [Goel et al., 2023] shows that removing data from the model can be done by catastrophic forgetting, which is antithetical to the objective of OCL approaches. Hence, we do not impose any memory constraints on our evaluation.

3. Towards Better Information Retention (IR). Interestingly, most of the previous benchmarks in OCL focus on preventing catastrophic forgetting of previously learned information, rather than evaluating the ability of models to rapidly learn new concepts [Lopez-Paz and Ranzato, 2017, Rebuffi et al., 2017b, Chaudhry et al., 2019a, Bang et al., 2021, Aljundi et al., 2019b, Lin et al., 2021]. This emphasis on information retention can be attributed to disjoint set-based data ordering, which often involves little to no change in distribution for most of the stream, making it difficult to evaluate rapid adaptation. However, recent works have shifted their focus to rapid adaptation performance in real-world ordered data streams [Cai et al., 2021, Ghunaim et al., 2023]. In our benchmark, our objective is to maintain an additional high degree of retained knowledge from past data as demonstrated in [Prabhu et al., 2023a, Cai et al., 2022].

4. Towards Enabling Rapid Adaptation (RA). The goal of *online* continual learning algorithms is to rapidly adapt to incoming data from changing distributions. Similarly to past work [Prabhu et al., 2023a, Cai et al., 2021, Ghunaim et al., 2023], we place a strong emphasis on achieving this goal with better metrics.

5. Towards Computational Budgets (CB). Continual learning without memory constraints is primarily a problem of achieving high performance with computational efficiency, as retraining from scratch is an ideal solution to the problem. This setting addresses the real-world problem of reducing the cost of updating a given model with incoming data from a stream [Prabhu et al., 2023a]. We evaluate OCL approaches under the maximum computational budget for fairness [Ghunaim et al., 2023], testing across two different budgets.

6. Towards Large-Scale OCL (LB). OCL benchmarks have historically focused on learning over samples incoming from a stream over long data streams since the development of GEM [Lopez-Paz and Ranzato, 2017]. Recent OCL benchmarks [Aljundi et al., 2019b, Koh et al., 2022, Prabhu et al., 2023a] have preserved this useful characteristic, while scaling up to larger and more complex data streams. In this work, we use two large-scale OCL benchmarks: *Continual Google LandMarks (CGLM)* [Prabhu et al., 2023a] and *Continual LOCALization (CLOC)* [Cai et al., 2021]. CGLM is a landmark classification dataset that consists of a long-tailed distribution with 10,788 classes

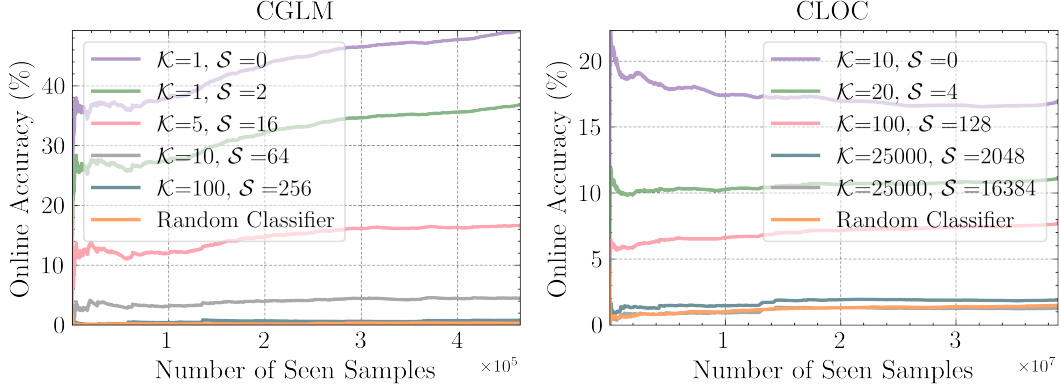


Figure 6.2.1: Blind Classifier. Performance of a blind classifier on the CGLM (left) and CLOC (right) datasets across varying shift S towards future for selecting evaluation samples alongside the optimal context window size k for the selected shift. (Section 6.3.2) A blind classifier achieves unrealistically high accuracy on current evaluation of adaptation (dark blue line, $S=0$) on both datasets despite being a trivial baseline. (Section 6.3.5) The accuracy of the blind classifier on incoming samples with increasing shift S into the future decreases significantly, eventually converging to near-random performance, indicating a lack of label correlations to the current training samples.

that simulate images that arrive on a Wikimedia Commons server. CLOC is a dataset focused on geolocation at scale consisting of 713 classes with 39M images simulating images arriving on a Flickr server. As these datasets come from two different sources, the issue of label correlation that we observe is likely to be a repeating pattern across future OCL datasets.

7. Leveraging Pretraining for Effective OCL (PT). Traditional methods in online continual learning start with randomly initialized models [Lopez-Paz and Ranzato, 2017, Aljundi et al., 2019a]. However, continual learning approaches can leverage the abundance of large-scale pretrained models to improve computational efficiency [Ostapenko et al., 2022, Wu et al., 2022, Prabhu et al., 2023a]. We focus on an evaluation setting that starts with ImageNet1K pretrained models to allow approaches to leverage pretrained information.

6.3 Correcting Evaluations of Rapid Adaptation

We delve into our exploration of the limitations of using online accuracy as a metric to measure adaptation to distribution shifts and present our efforts to alleviate the shortcomings of this metric. We start by identifying the label correlations that affect online accuracy, and then we unveil a technique that enables deep networks to artificially achieve high performance on this metric. This highlights the dependence of the networks on these label correlations. Finally, we introduce a novel

evaluation metric, near-future accuracy, and a test to detect and remove label correlations allowing for accurate measurement of the effectiveness of OCL methods in rapidly adapting to new distributions.

6.3.1 Evaluating OCL: A Problem Formulation

In traditional OCL setups with real-world stream orders [Cai et al., 2021, Prabhu et al., 2023a], the learner is fed a continuous stream of training samples over time steps $t \in \{1, 2, \dots, \infty\}$, consisting of an input datapoint $\mathbf{x}_t \in \mathcal{X}$ and its corresponding label $\mathbf{y}_t \in \mathcal{Y}$, where $(\mathbf{x}_t, \mathbf{y}_t) \sim \mathcal{D}_{j \leq t}$. The distribution $\mathcal{D}_j$ can change at any stream timestep. The aim is to train a classifier $f_{\theta_t} : \mathcal{X} \rightarrow \mathcal{Y}$ at any given t , which accurately maps a new sample $\mathbf{x}$ to its label $\mathbf{y}$ while adapting to the latest information and incorporating it with the historical knowledge acquired from previous data.

To evaluate the adaptation ability of f_{θ_t} , online accuracy measures the performance on the next unseen sample, $(\mathbf{x}_{t+1}, \mathbf{y}_{t+1})$, *i.e.*, $a_t = \mathbb{1}\{f_{\theta_t}(\mathbf{x}_{t+1}) = \mathbf{y}_{t+1}\}$. Online accuracy is updated in an online fashion given by $A_t^{RA} = \frac{1}{t} (A_{t-1}^{RA} \cdot (t-1) + a_t)$. For OCL approaches training deep networks, the running average is updated on a batch $\mathcal{B}$ of the next unseen samples as opposed to a single sample. After the evaluation is carried out, the model f_{θ_t} is updated using the same samples.

The ability of f_{θ_t} to retain information is measured by evaluating its accuracy on an unseen test stream that is similar to the training stream. In particular, per time step t , the model trained at the end of step t is evaluated on all test samples from step 0 to step t . We measure Backward Transfer @ T [Cai et al., 2021] where T is the last time step of the stream. That is to say, the model at the last step of the stream is evaluated on an increasing test test from step 0 to the end of the stream. However, traditional OCL methods targeting natural distribution shifts have neglected this metric [Cai et al., 2021, Ghunaim et al., 2023].

6.3.2 Isolating and Quantifying Label Correlations

Our objective is to investigate whether using online accuracy to measure adaptation in OCL suffers from drawbacks due to label correlations. To accomplish this, we introduce a very simple algorithm that leverages label correlations and performs well in terms of online accuracy.

Blind Classifier. We define a blind classifier, similar to prior works [Cai et al., 2021, Prabhu et al., 2023a], as a model that predicts the mode of the last $\mathcal{K}$ samples seen without access to the input images. Formally, at a given new step t , the blind classifier predicts the class of $\mathbf{x}_t$ as the mode of the last revealed labels $\cup_{j=1, \dots, \mathcal{K}} \{\mathbf{y}_{t-j}\}$. The optimal context window size $\mathcal{K}$ is selected by a search over different sizes on the hyperparameter search set.

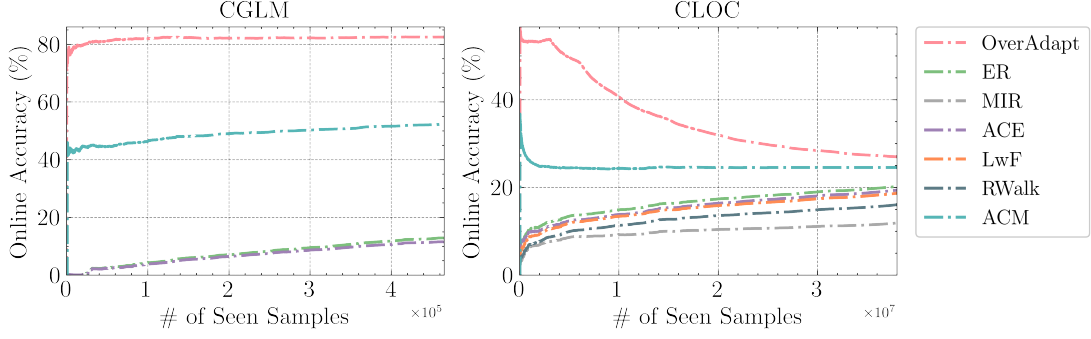


Figure 6.3.1: OverAdapt Classifier. We compare the adaptation performance of our OverAdapt classifier on CGLM (left) and CLOC (right) datasets using online accuracy as a measure. The three top performing OCL methods on CLOC are benchmarked on the CGLM dataset. We see that OverAdapt outperforms recent OCL methods by a large margin of 10-40% in accuracy.

Results. We present the performance of the blind classifier on the online accuracy in Figure 6.2.1. For this section, we focus solely on measuring online accuracy (indicated by the topmost purple line). Our results indicate that the blind classifier performs remarkably well achieving an average accuracy of 50% on the CGLM dataset, which consists of 10,788 classes, using only the labels of the last seen training sample ($\mathcal{K} = 1$). Moreover, the blind classifier achieves 17% on the large-scale CLOC dataset consisting of 718 classes with $\mathcal{K} = 10$ achieving the best performance. These findings suggest an unusually high degree of label correlation between the past 1-10 samples and the immediate incoming sample in the data stream, especially given that the blind classifier achieves 50% and 17% accuracy, while the random baseline achieves 0.01% and 2% on CGLM and CLOC datasets, respectively.

Conclusion. Despite having never processed any input images, the blind classifier achieves a remarkably high online accuracy. This is surprising because it was expected to perform no better than a random classifier. However, it achieves this result by exploiting the label correlations present in OCL datasets. This is a critical drawback in using online accuracy to measure adaptation. In the following sections, we will delve deeper into this issue.

6.3.3 Can Deep Networks Learn Label Correlations?

After demonstrating the effectiveness of the blind classifier in exploiting label correlations in the two large-scale datasets used in OCL settings, a pertinent inquiry arises: Can deep networks inadvertently learn these same label correlations leading to unreasonably high online accuracies?

This question warrants investigation to determine to what extent deep OCL algorithms can leverage label correlations and the impact this may have on their performance.

An Overfitting-Based Baseline. One approach for DNNs to exploit label correlations is by overfitting to recent data. We investigate algorithmic choices for designing an OCL method that can overfit the most recent labels in a data stream. Specifically, we make two important design choices for our baseline: (i) we update the model only on recent data to achieve overfitting while sacrificing most of the useful information from the past. To this end, we adopt FIFO sampling, a widely-used technique in OCL, to select batches of samples for training. (ii) We fix the feature representations and train only the last linear layer to prevent degradation of feature representations due to overfitting. To improve the quality of the representations, we use a ResNet50 model pretrained on Instagram1B [Mahajan et al., 2018]. We refer to this model as *OverAdapt*. OverAdapt is computationally efficient since training the Linear layer over multiple updates is relatively inexpensive. Moreover, we adopt the fast stream evaluation setting [Ghunaim et al., 2023], *i.e.* a setting with a strictly limited computational budget. We set the upper bound for the computation to one gradient update per incoming batch of samples for all methods with no restrictions on storage space.

Results. We present the results of the proposed OverAdapt compared to state-of-the-art OCL methods trained using an ImageNet1K pretrained ResNet50 model on the CGLM and CLOC datasets in Figure 6.3.1. OverAdapt achieves remarkable performance, surpassing the best-performing method ACM on the CGLM dataset by more than 30%, and outperforming all but the ACM baseline on the CLOC dataset by 10-35%. Notably, OverAdapt achieves an impressive 80% accuracy on the CGLM dataset.

Conclusion. After demonstrating that OverAdapt can effectively leverage label correlations from the data stream, we conclude that traditional design choices like the FIFO buffer in past OCL setups [Cai et al., 2021] improved overfitting to the next sample. It is worrying that potential future work or existing works could have inadvertently made similar choices leading to perceived improvement in algorithms when instead it is a subtle case of overfitting to the online accuracy metric. However, we further ensure the validity of our conclusions by investigating whether the effectiveness of this model is due to a good training algorithm or to potential limitations of the evaluation metric. We explore this in the next subsections.

6.3.4 Exploring Properties of OverAdapt

We investigate the reasons behind the impressive performance of our proposed OverAdapt.

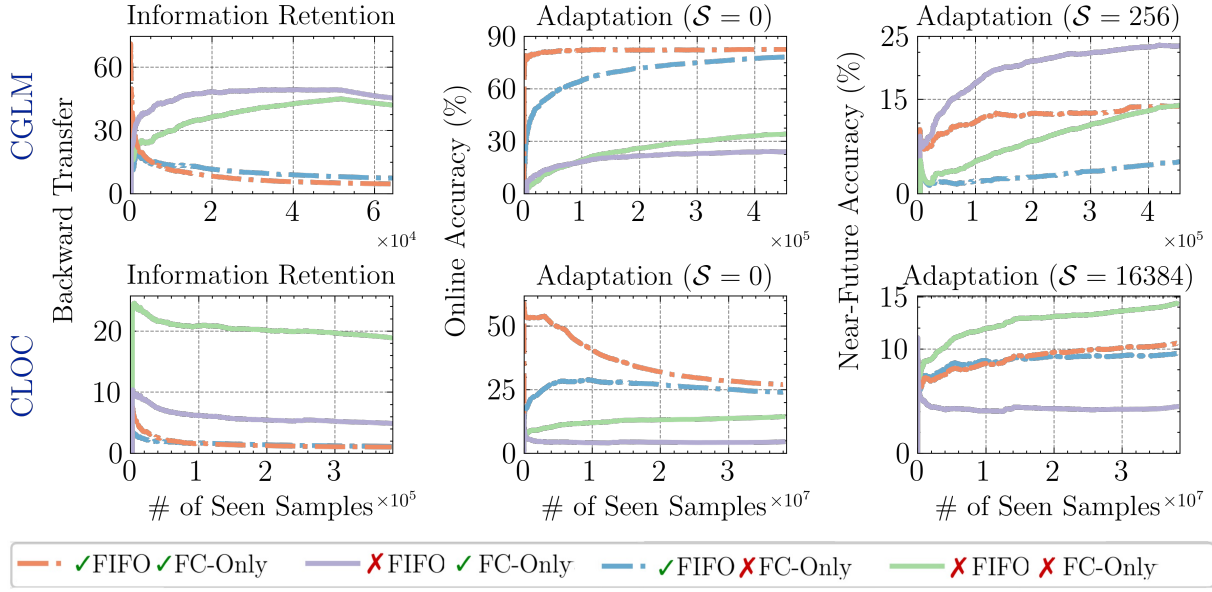


Figure 6.3.2: Sensitivity Analysis of OverAdapt Classifier. We illustrate the performance of various components of our OverAdapt classifier on CGLM (top) and CLOC (bottom) datasets measuring: (Section 6.3.4) information retention (left) and rapid adaptation capability in terms of online accuracy ($\mathcal{S} = 0$, center) (Section 6.3.5) rapid adaptation performance in terms of online accuracy ($\mathcal{S} = 0$, center) with our proposed near-future accuracy ($\mathcal{S} = 256$ in CGLM and $\mathcal{S} = 16384$ in CLOC). Note that information retention (left) is independent of the shift $\mathcal{S}$. The results demonstrate that introducing the shift removes label correlations, leading to dramatic loss in adaptation performance for our OverAdapt method while the uniform baseline stays unaffected.

Sensitivity Analysis. We start with the sensitivity study of OverAdapt by analyzing the impact of modifying its two key components: (i) FC-Only training with large-data pretrained initialization and (ii) FIFO Sampling. To study the effect of FC-Only training, we replace it with full-model training using an ImageNet1K-pretrained ResNet50 which has been common in prior OCL work [Ghunaim et al., 2023]. We also analyze the effect of FIFO sample selection by comparing it with uniform sampling, which has been shown to be a simple but effective strategy in limited-compute regimes [Prabhu et al., 2023b]. We evaluate their performance in terms of online accuracy and information retention at time T after training on the entire stream following prior art [Cai et al., 2021] as discussed in Section 6.3.1.

Results. Our results are shown in Figure 6.3.2 (left & center).

OverAdapt. Strikingly, our analysis reveals that **OverAdapt** ($\checkmark$ FIFO $\checkmark$ FC-Only) achieves remarkable performance in rapid adaptation as measured by online accuracy achieving **82%** and **27%** on CGLM and CLOC, respectively. However, this seemingly impressive performance is

coupled with an abysmally low information retention when measured by Backward Transfer with **OverAdapt** performing with 5% and 1% accuracy on CGLM and CLOC, respectively. These results demonstrate that **OverAdapt** is not a superior OCL algorithm but rather a product of a flawed rapid adaptation evaluation.

FIFO Sampling. Our analysis comparing sampling strategies reveals that models trained using the FIFO sample selection strategy with both **FC-Only training** (✓FIFO ✓FC-Only) and **full-model training** (✓FIFO ✗FC-Only) achieve significantly higher accuracy of **82/78%** and **27/24%** on CGLM and CLOC, respectively, on online accuracy metric compared to those trained using uniform sampling-based with **FC-Only training** (✗FIFO ✓FC-Only) and **full-model training** (✗FIFO ✗FC-Only). This clearly indicates that FIFO sampling is primarily responsible for the ability to leverage the latest label correlations. However, models trained with FIFO sampling achieve significantly lower accuracy of **5/7%** and **1/1%** on CGLM and CLOC, respectively, in terms of information retention as measured by backward transfer in the last timestep, performing close to a random classifier. This highlights their poor representation quality and confirms that models using the FIFO strategy simply overfit to the latest samples.

FC-Only Training. Comparing **FC-Only training** (✓FIFO ✓FC-Only) to **full-model training** (✓FIFO ✗FC-Only) under FIFO sampling, we observe that **FC-Only training** achieves a higher accuracy of **82%** and **27%** on CGLM and CLOC, respectively, in online accuracy. However, for information retention, both models perform poorly due to the impact of FIFO sampling. In contrast **full-model training** under uniform sampling (✗FIFO ✗FC-Only) achieved a higher accuracy of **34%** and **14%** on CGLM and CLOC, respectively, for online accuracy, compared to **full-model training** (✗FIFO ✓FC-Only). Additionally, **full-model training** achieved **42%** and **18%** accuracy on CGLM (comparable to **full-model training**) and CLOC (higher than **full-model training**), respectively, in information retention. In general, it is not conclusive whether FC-Only training could substitute full model training, as it performed significantly better in some cases and significantly worse in others.

Conclusion. Our experiments highlighted critical drawbacks in assessing the rapid adaptation of deep models using online accuracy, due to spurious label correlations in the data stream. Sensitivity analysis revealed that both FIFO sampling and FC-Only training are critical components that helped OverAdapt leverage label correlations where FIFO sampling being the primary contributor. Furthermore, our findings suggest that information retention is a useful indicator for identifying drawbacks of OCL algorithms, such as inadvertent overfitting.

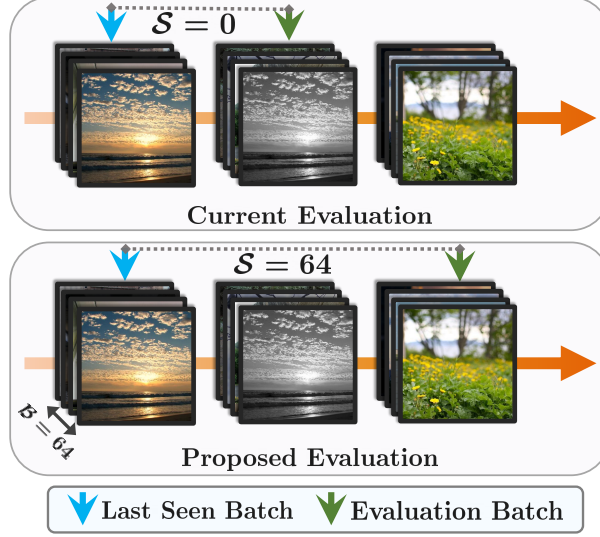


Figure 6.3.3: Our Evaluation Method. We compare the samples selected for evaluating adaptation performance by our evaluation method. The blue and green arrows indicate the latest batch received from the stream for learning and the samples selected for evaluation respectively. Current evaluation proposes to compute accuracy on the immediate incoming samples ($S = 0$), *i.e.* online accuracy, while we propose to evaluate on the closest future samples ($S = 64$), *i.e.* near-future accuracy, which is free from spurious correlations with the seen batch of images.

6.3.5 Removing Correlations to Evaluate Adaptation

To address the limitations of online accuracy, we propose a new metric, near-future accuracy, which involves evaluating the model on near-future samples rather than immediate next samples. This can mitigate the adverse effects of label correlations on the performance by obtaining a more accurate assessment of the adaptation ability of OCL algorithms.

Formulation of Proposed Evaluation. As illustrated in Figure 6.3.3, we introduce a shift S that represents the number of samples in the future, after which we evaluate the performance of the model. Instead of evaluating on the immediate next sample $(\mathbf{x}_{t+1}, \mathbf{y}_{t+1})$, we use the next near-future sample after the smallest shift S , namely $(\mathbf{x}_{t+1+S}, \mathbf{y}_{t+1+S})$, so that it has no label correlation with them. We select the smallest shift S instead of an arbitrarily large shift S so that the test sample comes from the same distribution as the latest seen train samples. We calculate the online accuracy as before, where $\hat{a}_t$ is the running average of the accuracy calculated by first calculating the accuracy $a_t = \mathbb{1}\{f_{\theta_t}(x_{t+1+S}) = y_{t+1+S}\}$ and then updating the running average using the formula $A_t^{RA} = \frac{1}{t} (A_{t-1}^{RA} \cdot (t-1) + a_t)$.

How To Set the Shift S ? The shift S is the smallest value in the future that has no label correlation

with the last seen train samples. We select the smallest S so that the blind classifier performs similarly to a random classifier. As shown in Figure 6.2.1 for both CGLM (left) and CLOC (right) datasets, we observe that increasing the shift S decreases the accuracy of the blind classifier down to a similar performance of a random classifier. In particular, at a shift of $S = 256$ for CGLM and $S = 16384$ for CLOC, the label correlations are effectively eliminated. Therefore, we use these shift values for our proposed evaluation strategy. We evaluate OverAdapt and its variants on rapid adaptation with our proposed near-future accuracy and compare it with the previously adopted online accuracy ($S = 0$).

Results. Figure 6.3.2 presents the results of OverAdapt and its variants on rapid adaptation using two metrics: online accuracy (center) and near-future accuracy (right). When studying FIFO sampling with **FC-Only training** (✓FIFO ✓FC-Only) and with **full-model training** (✓FIFO ✗FC-Only), we observe a significant drop in the performance of the methods between the two metrics that overfit using the FIFO sampling by about 70% in CGLM and 20-40% on CLOC datasets, while the performance of uniform sampling for both **FC-Only training** (✗FIFO ✓FC-Only) and **full-model training** (✗FIFO ✗FC-Only) remains similar. Interestingly, the trends reverse, with the uniform sampling variants outperforming the FIFO sampling strategy by a large margin. Furthermore, we observe that FC-Only training no longer consistently improves rapid adaptation, proving to be useful in some cases on the new evaluation method.

Conclusion. We propose evaluation using near-future accuracy which removes label correlations. OverAdapt performs poorly after removing label correlations, compared to a uniform baseline, emphasizing the importance of developing OCL algorithms that handle rapid adaptation without overfitting. Additionally, there is no surprising discrepancy between near-future accuracy and information retention across methods, unlike online accuracy. Our strategy allows for better evaluation of OCL algorithms on real-world ordered datasets. We provide code to reproduce our results at <https://github.com/drimpossible/EvalOCL>.

6.4 OCL Evaluation by Near-Future Accuracy

In this section, we provide a comprehensive analysis of various OCL approaches under near-future accuracy. We incorporate the latest advancements in setup design into our benchmark and take computational constraints into account for fair comparison. Following the prior art [Prabhu et al., 2023a,b], our study assumes no memory constraints, but rather sets constraints on computational budget following [Ghunaim et al., 2023, Prabhu et al., 2023a,b]. As addressed in prior work, budgeted computation per time step implicitly imposes a limited memory access.

6.4.1 Experimental Setup

Datasets and Metrics. We employ two large-scale online continual learning datasets, CGLM with a total of 460K images and CLOC comprising of 39M images in a stream, both of which contain natural distribution shifts [Cai et al., 2021]. That is to say, the images are temporally ordered. We measure rapid adaptation performance using the average accuracy on near-future samples, as described in Section 6.3.5, with a shift $S = 256$ for CGLM and $S = 16384$ for CLOC. The choice of both was made such that the blind classifier discussed in Section 6.3.2 achieves a random accuracy. Additionally, we measure information retention using the Backward Transfer metric following CLOC [Cai et al., 2021].

Model and Optimization. In all experiments, we use a ResNet50 with ImageNet1K initialization unless otherwise specified. We adopt the optimization procedure from [Ghunaim et al., 2023] and train all models using an SGD optimizer, a fixed learning rate of 0.005, and a weight decay of 10^{-4} . For both training and evaluation, we batch incoming samples with a batch size $\mathcal{B}$ of 64 for CGLM [Prabhu et al., 2023a] and 128 for CLOC [Ghunaim et al., 2023].

Compared OCL Approaches. We evaluate five online continual learning methods using our proposed near-future accuracy. For consistency, we use uniform sampling as the sampling strategy, since it was shown in Section 6.3.4 that it strikes a good balance between adaptation and retention. Again, in uniform sampling, the training batch is constructed by uniformly and sampling $\mathcal{B}$ samples from all past stored data. The Appendix provides a comprehensive analysis of other sampling techniques.

ER (Replay Only)[Prabhu et al., 2023b]. ER (Replay Only) is a simple approach that stores the incoming samples and trains the model on a batch sampled from the set of stored samples. It has demonstrated this to be the leading method in budgeted offline CL.

FC-Only. FC-Only, referred to as OverAdapt (~~X~~FIFO ✓FC-Only) in Section 6.3.4, is an adapted version of ER (Replay Only) that uses a ResNet50-I1B feature extractor and trains only the last linear layer. This method has been shown to be computationally efficient [Prabhu et al., 2023b].

ACE[Caccia et al., 2022b]. We replace the CrossEntropy loss in ER (Replay Only) with the ACE Loss proposed in to isolate the effect of ACE loss under the proposed evaluation setting.

ACM[Prabhu et al., 2023a]. ACM extracts features from a pretrained ResNet50 model (trained on ImageNet1k) and continually updates a kNN classifier, with $k = 2$ and cosine distance as distance metric, over those features. Additionally, we note that ACM does not train a deep network, thus incurring minimal computational cost. This sets the minimal accuracy requirements for OCL methods.

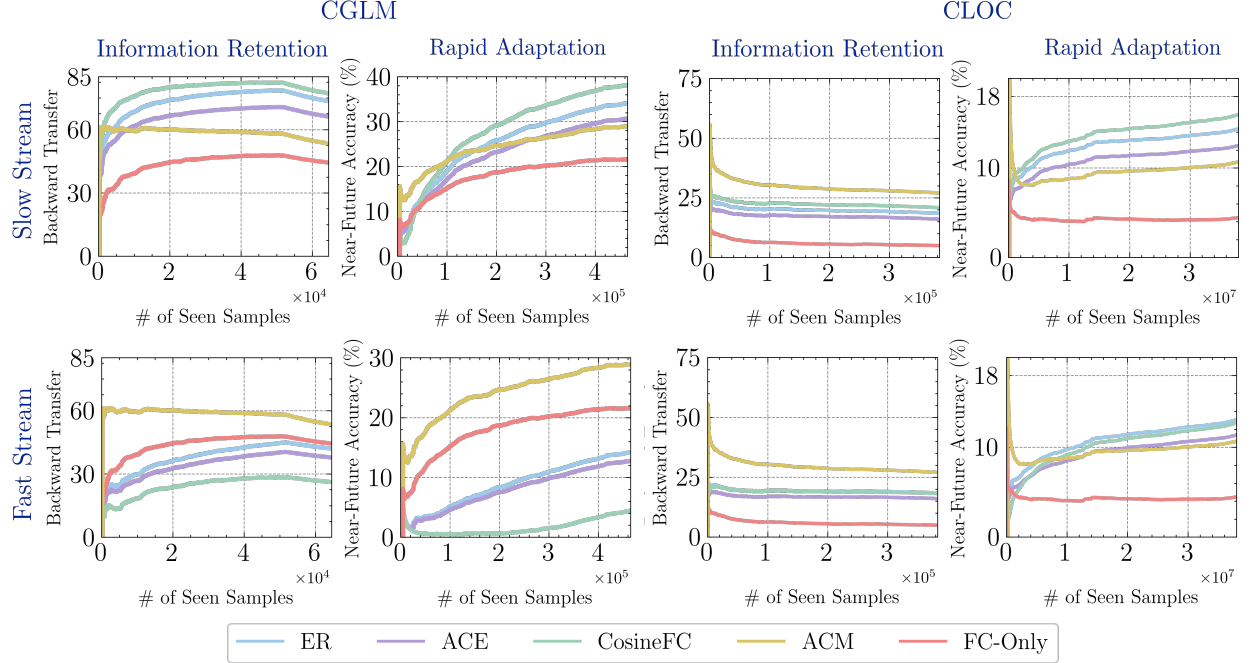


Figure 6.4.1: Performance of Methods on Near-Future Accuracy. We evaluate CL methods in slow and fast streams similar to [Ghunaim et al., 2023]. Near-future accuracy, which eliminates spurious label correlations, shows that retaining and reusing past information is crucial for better generalization and that algorithms not designed for adaptation in OCL can perform well. Hence retention and adaptation are closely linked.

CosineFC[Hou et al., 2019]. This approach replaces the linear layer of the ResNet50 model in the ER (Replay Only) baseline with a layer that computes the cosine distance between the weights and the features. We only perform this modification as distillation has been shown to be computationally ineffective [Prabhu et al., 2023b].

By comparing these approaches under our proposed near-future accuracy along with information retention, we aim to provide insights into the strengths and limitations of current OCL methods.

6.4.2 Results: Varying Stream Speeds

Following prior work on fair computationally normalized evaluation of online continual learners [Ghunaim et al., 2023], we evaluate the performance of OCL methods in two different time constraints, namely slow stream and fast stream scenarios.

Slow Stream. Slow stream is the scenario in which the computational budget for learning is not strictly limited. We enforce all methods to perform the equivalent of 10 and 5 model updates per

time step on CGLM and CLOC, respectively. Methods that require a significant overhead will thus perform a fewer number of model updates such that it matches the corresponding computational budget of 10 and 5 updates on the respective datasets. Note that ACM requires negligible training cost, and therefore OCL methods must at least outperform ACM to be considered useful.

Results. In Figure 6.4.1 (first row), we present the performance of various methods in terms of near-future accuracy for both CGLM and CLOC datasets. CosineFC, originally proposed for offline continual learning, achieves the highest accuracy with 38% and 16% accuracy on CGLM and CLOC, respectively. Furthermore, CosineFC also shows impressive information retention performance, achieving an accuracy of 77% in CGLM. As expected, FC-Only which only fine-tunes the last linear layer, has the worst retention capabilities due to the restricted access to full-model updates.

Conclusion. Our evaluation revealed that methods not explicitly designed for adaptation in the OCL setup perform well, indicating that retaining and reusing previously seen information is crucial for achieving better generalization, and that retention and adaptation are closely linked.

Fast Stream. Fast stream setting refers to the scenario where the computational budget for learning is limited. That is to say, the stream presents samples at a fast rate that provides limited time for learners to train [Ghunaim et al., 2023]. To evaluate the performance of OCL methods under this setting, we restrict the methods to only use a computation equivalent to one model update per incoming batch for both datasets.

Results. As shown in Figure 6.4.1 (second row), our results suggest that the two simplest baselines, namely ACM and FC-Only, achieve the highest accuracy on CGLM with 29% and 21%, respectively. On the other hand, ACM achieves the highest information retention of 27% while simple ER and CosineFC achieve the highest near-future accuracy of around 13%. These results indicate that the simple baseline methods outperform the more advanced OCL methods in both adaptation and information retention in this setup. This highlights that current OCL methods still have a long way to go to achieve satisfactory performance in the fast stream setting where compute is extremely limited.

Conclusion. Under our proposed evaluation approach, in which spurious label correlations are removed, the fast stream setting with its limited computational budget shows that simple baselines can outperform more advanced OCL methods in both adaptation and retention. This finding emphasizes the need to develop OCL methods that can adapt rapidly in severely budgeted settings.

6.4.3 Further Discussion

Our benchmark of OCL methods on large-scale datasets highlights two key observations. First, the best-performing online adaptation methods often suffer from forgetting due to overfitting to label

correlations, challenging the assumption that the performance gap between adaptation and retention is an inherent learning problem [Cai et al., 2022]. Our proposed near-future accuracy helps reveal label correlations and identifies whether the underlying algorithm is inadvertently exploiting them. Second, methods with better retention properties tend to generalize better and achieve improved performance on near-future accuracy in relaxed and restricted computational settings [Gulrajani and Lopez-Paz, 2021, Wang et al., 2021]. This aligns well with the domain generalization literature [Gulrajani and Lopez-Paz, 2021] and suggests that proper training on previous samples can lead to features that generalize well across domain shifts. Finally, we consistently observe that, under limited computational settings, simpler methods outperform their more computationally involved ones.

6.5 Conclusion

Our work proposes a new measure to evaluate adaptation in OCL algorithms and highlights its limitations compared to the standard online accuracy evaluation. We found that many current OCL methods overfit to idiosyncrasies of the stream rather than genuinely adapting to new data, as revealed by poor information retention. By using our proposed metric, near-future accuracy, we observed that algorithms with good retention also had better generalization and adaptation capabilities. Our proposed evaluation can serve as a sanity check for future OCL algorithms to ensure that they are not incorrectly learning spurious label correlations. This covers the efforts in this thesis to discover better metrics for continual learning in a computationally budgeted scenario. In the next chapter, we explore how to alleviate the labeling bottleneck for continual training.

7

From Categories to Classifier: Name-Only Continual Learning by Exploring the Web

Contents

7.1	Introduction	94
7.2	Name-Only Continual Learning: Problem Formulation	95
7.3	Our Approach: Categories to Classifier By Exploring the Web	96
7.4	Evaluating Capabilities of our Approach	99
7.4.1	Experimental Details	99
7.4.2	How Reliable is our Proposed Approach?	100
7.4.3	Comparison with Name-Only Approaches	101
7.5	Continual Webly-Supervised Learning	103
7.5.1	Experimental Details	104
7.5.2	Results	105
7.5.3	EvoTrends: The First Continual Name-Only Classification Benchmark . . .	107
7.6	Conclusion	108

Abstract

Continual Learning (CL) often relies on the availability of extensive annotated datasets, an assumption that is unrealistically time-consuming and costly in practice. We explore a novel paradigm termed name-only continual learning where time and cost constraints prohibit manual annotation. In this scenario, learners adapt to new category shifts using only category names without the luxury of annotated training data. Our proposed solution leverages the expansive and ever-evolving internet to query and download uncured webly-supervised data for image classification. We investigate the reliability of our web data and find them comparable, and in some cases superior, to manually annotated datasets. Additionally, we show that by harnessing the web, we can create support sets that surpass state-of-the-art name-only classification that create support sets using generative models or image retrieval from LAION-5B, achieving up to 25% boost in accuracy. When applied across varied continual learning contexts, our method consistently exhibits a small performance gap in comparison to models trained on manually annotated datasets. We present EvoTrends, a class-incremental dataset made from the web to capture real-world trends, created in just minutes. Overall, this paper underscores the potential of using uncured webly-supervised data to mitigate the challenges associated with manual data labeling in continual learning.

7.1 Introduction

In this chapter, we seek to alleviate the reliance of continual learning methods on labeled datasets. To this end, we explore a new scenario called *name-only continual learning*¹. As commonly done in the traditional continual learning, new categories or domain shifts are continuously introduced at each timestep and we need to quickly adapt the classification model to the changes in the stream; however, in this setting we cannot create annotated training datasets. At each timestep, the learner is only provided with category/class names and is allocated a computational budget to adapt to the new classes. At the end of each timestep, the learner is presented with test samples and its performance is assessed. To tackle this setting, we propose to leverage the ever-evolving internet by query and downloading uncurated webly-supervised data for continual image classification. This will dramatically speed up the process of continually updating classifiers, from once in several days to once practically every hour.

Why Revisit Webly-Supervised Learning [Fergus et al., 2005, Schroff et al., 2010]? Recently, countries like Japan² have enacted legislations allowing the use of online data for training deep models, irrespective of the copyright status. This follows the intuition that one can learn and be inspired from copyrighted materials so long they do not regenerate it or derivate works, such as with classification models. This allows us to leverage the internet, which functions as an ever-expanding database, continually updating itself with billions of new photos daily, staying current with the latest trends. Additionally, it provides search engines that traditionally offer highly relevant image results at scale, allowing us to query and download webly-supervised data cheaply and in just minutes. Being dynamic, the internet is ideal for continually updating to rapid changes in the stream. In this context, we address three crucial questions about the use of the web for training dataset creation:

① *How reliable is our uncurated webly-supervised data?* To assess its quality, we compare performance of deep learning models on our webly-supervised training data with manually annotated datasets for fine-grained image classification, which typically require expert annotations. We find that in some cases models trained on uncurated webly-supervised data can equal or even surpass the performance of those trained on manually annotated datasets. We show that this performance primarily results from our ability to cheaply gather much larger training sets than manual annotation allows.

¹We borrow the term name-only classification from [Udandarao et al., 2023]. We do not use zero-shot classification [Lampert et al., 2009] as it aims to generalize to unseen categories *without seeing any examples*, using attribute information whereas name-only setting allows access to public models and data.

²<https://aibusiness.com/data/japan-s-copyright-laws-do-not-protect-works-used-to-train-ai->

② *How does uncurated webly-supervised data compare to the latest name-only classification approaches?* We demonstrate that using uncurated webly-supervised data, one can outperform alternative methods of dataset generation used in state-of-the-art name-only classification approaches [Udandaraao et al., 2023, He et al., 2022, Wallingford et al., 2023] on the same CLIP model by an impressive 5-25% absolute accuracy improvement. Our approach can also generalize to vision-only self-supervised models like MoCoV3 ImageNet1K models [Chen et al., 2021b].

③ *Can we efficiently utilize uncurated webly-supervised data across various continual learning settings?* We apply our name-only webly-supervised approach to various continual learning situations such as class-incremental (new classes introduced over time), domain incremental (new domains introduced over time), and time incremental (mimicking a chronologically ordered class-annotated stream). In each of the above scenarios where we had access only to class names, our models trained on uncurated webly-supervised data only had a small performance gap compared to those trained on curated datasets. To illustrate our capabilities beyond existing datasets, we introduce *EvoTrends*, a continual learning dataset that introduces trending products year-by-year from 2000 to 2020. This underscores our ability to build classifiers and deploy them in a continual manner within minutes without relying on manually curated training datasets.

In summary, our primary contributions address the aforementioned three questions, conclusively showing that using uncurated webly-supervised data can significantly reduce the time and expense associated with manual annotation in the proposed name-only continual learning setting.

7.2 Name-Only Continual Learning: Problem Formulation

In the *name-only* classification setup, the target is to learn a function f_θ parameterized by θ , where here, unlike traditional classification tasks, the only given information is the class categories denoted by $\mathcal{Y}$. While additional context about the data distribution (*e.g.* cartoon, art, sketch,...) is allowed to be given in $\mathcal{Y}$, no training samples are provided. In contrast to the zero-shot setting, the learner is allowed to use publicly available data and models, with the exception of the original training set and models trained on it. For example, the use of prominent backbones like GPT [OpenAI, 2023], DALL-E [Ramesh et al., 2022] and assembling a training set from public datasets such as LAION5B [Schuhmann et al., 2022] is allowed to obtain the classifier. The performance of the learner is subsequently assessed on a curated test set, $\mathcal{X}^*$.

We extend the *name-only* classification paradigm to continual learning, dubbing this *name-only continual learning*. In this setup, we perform *name-only* classification across multiple timesteps,

$t \in \{1, 2, 3, \dots\}$. For each timestep t , a data stream $\mathcal{S}$, unveils a distinct set of class categories, $\mathcal{Y}_t$. Notably, $\mathcal{Y}_t$ might introduce categories absent in preceding timesteps; that is, a category $y_t \in \mathcal{Y}_t$ might not belong to $\mathcal{Y}_j$ for all $j < t$. Subsequently, at each t , the algorithm must continually update the classifier f_θ by using prominent backbones or publicly available data.

Formally, the primary goal in continual learning, is to learn a classifier $f_{\theta_t} : \mathcal{X} \rightarrow \bigcup_{i=1}^t \mathcal{Y}_i$, parameterized by θ_t , that correctly classifies a category from all the introduced class categories up to the current timestep. Given that evaluation samples could originate from any past class categories, *i.e.* $y_i \in \bigcup_{i=1}^t \mathcal{Y}_i$, the updated model f_{θ_t} must maintain its capabilities in classifying earlier seen classes. In summary, at every timestep t :

1. The data stream, $\mathcal{S}$, presents a set of categories, $\mathcal{Y}_t$, to be learned.
2. Under a given computational budget, C_t , the classifier $f_{\theta_{t-1}}$ is updated to f_{θ_t} .
3. To evaluate the learner, the stream $\mathcal{S}$ presents test samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with y_i belonging to the collective set $\bigcup_{i=1}^t \mathcal{Y}_i$.

In Step 3, it is important to note that the annotated test set is reserved solely for evaluation. Neither the images nor the labels from the test set are available for the model in any future training steps. Moreover, it is worth noting that computational budgeting has become the prevailing standard in CL Prabhu et al. [2023b]. This practice involves setting limits, either in terms of computation or time, hence on the number of samples that can be generated or annotated for training purposes.

7.3 Our Approach: Categories to Classifier By Exploring the Web

Without access to training data, one might be tempted to use generative models to create training data. However, as explained in Section 7.2, the continual learner is constrained by a budget limit C_t . This budget constraint makes generative methods computationally impractical due to their high computational requirements. Hence, we discuss our approach, “C2C”, for transitioning from class categories to classifiers within a computational budget. At each timestep t , our approach involves takes main steps: (1) collecting data from the web, which we refer to as uncured webly-supervised data and (2) training a classifier using this data.

Step 1. Querying and Downloading Uncured Webly-Supervised Training Data. There are several challenges associated with querying the web which raises questions that we address below:

How to design web queries? The web is expansive and noisy, and simply class categories provided by stream are often not specific enough. Consider the category name “snapdragon”:

on its own, search engines might yield images of computer chips. Hence, we design a simple querying strategy of adding an auxiliary suffix to refine our queries. Our searches follow the pattern: **Category Name + Auxiliary Suffix**. When building a flower dataset and querying “snapdragon”, appending the suffix “flower” refines the query to focus on the desired botanical images. Moreover, within domain-incremental settings, we can adapt our search by using domain-specific suffixes like “cartoon” for cartoon images. In summary, this addition offers a richer context, steering the search engine more precisely.

How do we prevent unintentional download of explicit images? Past webly supervised methods have unintentionally collected explicit content from online sources [Birhane and Prabhu, 2021]. To address this, we implemented some cost-effective safeguards. First, we enabled strict safe-search feature on our search engines, which helps filter out explicit or inappropriate content. Second, we ensure that class-categories $\mathcal{Y}_t$ do not have explicit terms by manually checking the queries and replacing possible offensive terms with less offensive ones, e.g. “african ass” would be replaced by “african wild donkey” or “Equus africanus”. We manually inspected a few hundred of the downloaded images with random sampling and found no explicit content providing preliminary evidence of effectiveness of the safeguards.

Improvements in the speed of querying and download. The end-to-end scraping and downloading time required for 39 million flickr samples in a stress test required 12 days using a standard Python query and download pipeline. We optimized and reduced it to just 2 days - a 600% improvement - using the same computational resources. We applied the same pipeline to accelerate querying and downloading of uncured internet data, we utilize parallelization across multiple dimensions: (1) We query four major search engines concurrently - Bing, Flickr, Google and DuckDuckGo - using separate CPU nodes in a cluster. This allows for simultaneous querying across engines. (2) We use an efficient multi-threaded querying tool³ that handles image search queries in parallel for each engine. This tool utilizes FIFO threaded queues to concurrently manage the search and download workflows for each query. (3) After aggregating image links from different engines, we leverage a parallelized image downloading tool⁴, which additionally applies postprocessing such as resizing. In conclusion, the key factors were concurrent querying across multiple search engines, fast multi-threaded querying per engine, and parallelized downloading and resizing of images.

Step 2. Classifier Training. Once we have uncured webly-supervised data the next step is to train a classifier. At each timestep t , the learner is assigned a computational budget, denoted as C_t .

³<https://github.com/hellock/icrawler>

⁴<https://github.com/rom1504/img2dataset>

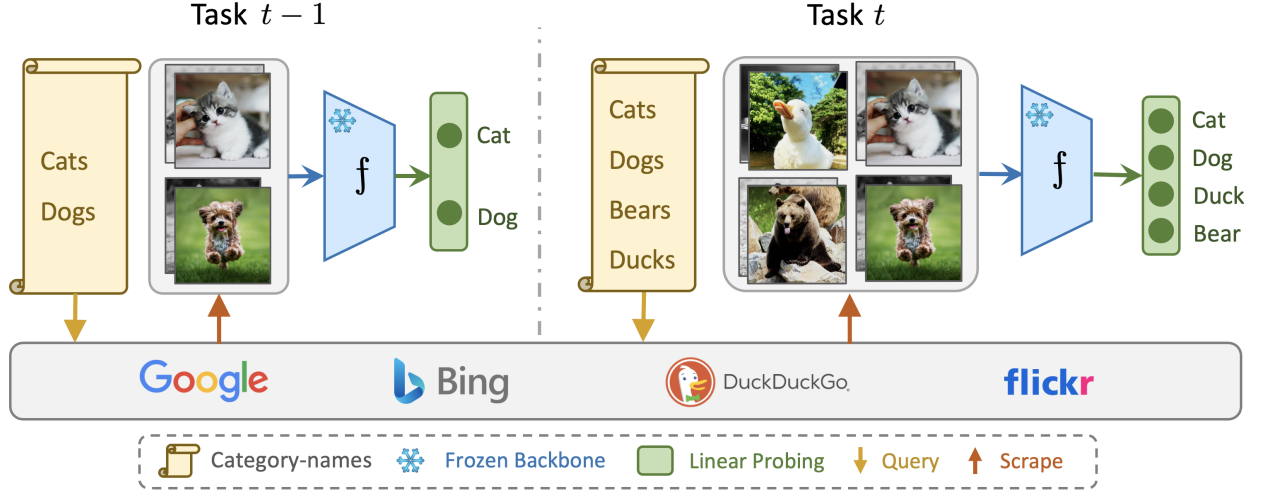


Figure 7.3.1: Continual Name-Only Classification: Our Approach. At each timestep t , the learner receives a list of class categories without any training samples. We start by collecting webly-supervised data through querying and downloading data from multiple search engines. We then extract features using a frozen backbone, and subsequently train a linear layer on those features. The same process is repeated for the next timestep.

Ideally, this budget should include the entire data collection process, whether it involves querying and downloading from the web or manual annotation. It is important to note that including this overhead within the budget would make it challenging or even impossible for manually annotated datasets to receive sufficient training, as their annotation pipeline incurs significant costs. We test three budgets: tight, normal, and relaxed. Normal budget allows for a training equivalent to 1 epoch on the first timestep of the manually annotated datasets (details in Appendix D). The “tight” budget is half of the normal, while the “relaxed” budget is four times the normal, as done in [Prabhu et al., 2023b]. Under this budgeted setup, we compare three continual learning baselines (a) Linear Probing, (b) NCM [Mensink et al., 2013, Janson et al., 2022] and (c) KNN [Malkov and Yashunin, 2018, Prabhu et al., 2023a], providing insights into efficient CL methods with fixed feature extractor.

Our approach is summarized in Figure 7.3.1. In our *continual name-only* classification setting, for each timestep t , we query and download webly-supervised data based on the provided class categories $\mathcal{Y}_t$, by following the recipe described in Step 1. Once we complete downloading the data, the classifier is trained not only on the uncured data downloaded from the current timestep t but also on the uncured data downloaded from all prior timesteps.

7.4 Evaluating Capabilities of our Approach

We begin by examining two main questions: ❶ How reliable is uncured webly-supervised data? Specifically, can models trained on these training sets match the accuracy of those trained on expert-annotated training sets? ❷ How does the uncured webly-supervised data compare to the latest name-only classification approaches? For instance, can models trained on our data surpass the latest methods tailored for vision-language models, such as CLIP, in a name-only classification context? We analyze these questions in the better studied non-continual name-only classification setting where one is provided with a set of class categories to be learnt.

7.4.1 Experimental Details

Datasets. We focus primarily on fine-grained classification tasks due to two main reasons: (i) they present a greater challenge than coarse-grained datasets especially when sourced from noisy data sources such as the web, and (ii) they are widely used in name-only classification benchmarks, facilitating comprehensive comparisons with existing methods. We evaluate the classification accuracy across five benchmarks that contain a broad set of classes: (1) FGVC Aircraft [Maji et al., 2013], (2) Flowers102 [Nilsback and Zisserman, 2008], (3) OxfordIIITPets [Parkhi et al., 2012], (4) Stanford Cars [Krause et al., 2013], and (5) BirdSnap [Berg et al., 2014].

Models. We use a fixed backbone, ResNet50 MoCoV3 [Chen et al., 2021b], and experiment with two classifiers on top: (i) Linear Probe and (ii) MLP Adapter. The MLP Adapter consists of a three-layer model: $\text{input_dim} \rightarrow 512, 512 \rightarrow 256$, and $256 \rightarrow \text{num_classes}$, with Dropout(0.5) and ReLU nonlinearities. Additionally, we also try fine-tuning both the backbone and a linear layer.

Training Procedure. For linear probing and MLP adapter experiments, we freeze the backbone and extract features from both our uncured webly-supervised data and manually annotated (MA) datasets. We then perform linear probing and MLP adapter training on the extracted features. Our training uses an Adam optimizer with a batch size of 512 and a learning rate of 0.001. We use a LRonPlateau scheduler with a patience of 10 and a decay of 0.1. Models are trained for 300 epochs, reaching convergence within 10-50 epochs. For finetuning experiments for both our uncured webly-supervised data and manually annotated (MA) datasets, we use an SGD optimizer with a learning rate of 0.1 and a linear learning rate scheduler. A batch size of 128 and standard data augmentations are applied. Models are trained until convergence on both uncured web data and the manually annotated training sets, within 50 epochs for our uncured web data and up to 250 for manually annotated datasets. Class-balanced random sampling is used for all experiments, especially helpful for data downloaded from the internet given its natural long-tail distribution.

Table 7.4.1: Performance Analysis between Uncurated Webly-Supervised Data (C2C) and Manually Annotated Training (MA) Data. Despite utilizing uncurated web data, our results demonstrate competitive or even better performance than that of manually annotated datasets in fine-grained categorization tasks. The most notable improvements are observed when using MLP-adapters.

Evaluation	Training Dataset	Datasets				
		FGVC Aircraft	Flowers102	OxfordIIITPets	Stanford Cars	BirdSnap
Linear Probe	MA Data	38.5%	83.3%	89.8%	56.3%	46.2%
	C2C (Ours)	57.5% (+19.0%)	85.7% (+2.4%)	91.7% (+1.9%)	62.1% (+5.8%)	56.1% (+9.9%)
MLP Adapter	MA Data	46.0%	80.3%	89.7%	57.6%	47.7%
	C2C (Ours)	65.5% (+19.5%)	87.1% (+6.8%)	92.8% (+3.1%)	66.8% (+9.2%)	53.7% (+6.0%)
Finetune Backbone	MA Data	76.6%	94.3%	92.8%	91.6%	70.4%
	C2C (Ours)	94.8% (+18.2%)	93.3% (-1.0%)	94.7% (+1.9%)	92.8% (+1.2%)	69.9% (-0.5%)

Table 7.4.2: Comparison with Name-Only Classification Techniques with ResNet50: When comparing with existing state-of-the-art name-only classification techniques, we show that our method outperforms those methods by margins ranging from 2% to 25%.

Type	Method	Model	Birdsnap	Aircraft	Flowers	Pets	Cars	DTD
Data-Free	CLIP-ZS [Radford et al., 2021]	CLIP	32.6	19.3	65.9	85.4	55.8	41.7
	CaFo-ZS [Zhang et al., 2023b]	CLIP	-	17.3	66.1	85.8	55.6	50.3
	CALIP [Guo et al., 2023]	CLIP	-	17.8	66.4	86.2	56.3	42.4
	CLIP-DN [Zhou et al., 2023b]	CLIP	31.2	17.4	63.3	81.9	56.6	41.2
	CuPL [Pratt et al., 2023]	CLIP	35.8	19.3	65.9	85.1	57.2	47.5
	VisDesc [Menon and Vondrick, 2022]	CLIP	35.7	16.3	65.4	82.4	54.8	42.0
	SD-CLIP [Li et al., 2023a]	SD-2.0	-	26.4	66.3	87.3	-	-
Use-Data	GLIDE-Syn [He et al., 2022]	CLIP	38.1	22.0	67.1	86.8	56.9	43.2
	CaFo [Zhang et al., 2023b]	CLIP	-	21.1	66.5	87.5	58.5	50.2
	SuS-X-LC [Udandara et al., 2023]	CLIP	38.5	21.1	67.1	86.6	57.3	50.6
	SuS-X-SD [Udandara et al., 2023]	CLIP	37.1	19.5	67.7	85.3	57.2	49.2
	C2C (Ours-Linear Probe)	CLIP	48.1 (+9.6)	44.0 (+22.0)	82.0 (+14.3)	88.1 (+0.6)	71.3 (+12.8)	57.1 (+6.5)
	C2C (Ours-MLP Adapter)	CLIP	46.6 (+8.1)	48.9 (+26.9)	84.8 (+17.1)	89.4 (+1.9)	72.6 (+14.1)	57.6 (+7.0)
	C2C (Ours-Linear Probe)	MocoV3	56.1 (+17.6)	57.5 (+35.5)	85.7 (+18.0)	91.7 (+4.2)	62.1 (+3.6)	54.6 (+4.0)
	C2C (Ours-MLP Adapter)	MocoV3	53.7 (+15.2)	65.5 (+43.5)	87.1 (+19.4)	92.8 (+5.3)	66.8 (+8.3)	55.8 (+5.2)

7.4.2 How Reliable is our Proposed Approach?

We first address our first fundamental question: ① Can uncurated webly-supervised data serve as a substitute for meticulously curated training data? Put simply, can web data match the accuracy of manually annotated datasets?

Results and Key Findings. Table 7.4.1 contrasts the performance of our uncurated webly-supervised data with manually annotated datasets. Remarkably, classifiers trained on our webly-supervised data surpass those trained on manually annotated datasets by a margin of 1 – 19% (highlighted in green). In the worst-case scenario, there is a performance decline of less than 1% (marked in red). The most pronounced improvement, ranging from 3 – 19%, arises when our webly-supervised data is integrated with an MLP-Adapter. As anticipated, fine-tuning yields superior results compared to the MLP adapter, which in itself outperforms linear probing.

In summary, using uncurated webly-supervised data consistently outperform manually annotated datasets across different classifiers and datasets. This finding is counterintuitive, given that our web data is: (i) uncurated, (ii) noisy, and (iii) out-of-distribution with respect to the test set. The

reason behind this apparent paradox can be attributed to the dataset sizes. Details are provided in the Appendix B and summarized below.

How Does Scale Influence Our Performance? Our webly-supervised datasets are notably large due to cheap query and download properties, being approximately 15 to 50 times larger than the manually annotated datasets. Hence, we explore the impact of scale by limiting our queries to search engines to return only the top- k images in Table 7.4.3. Our results suggest that query size is the primary driver for performance gains. When we limit our query size to match the size of manually annotated datasets (using the top 10 or 20 images per engine per class), there is a drop in accuracy by 10-20% relative to manually curated datasets. However, as we gather more data, we consistently observe performance improvements. The scalability of our method, only possible due to virtually no scaling cost. The primary cause of superior performance is by scaling the size of downloaded data, without high costs of manual annotations or other checks.

In Appendix B, we explore various factors that, surprisingly, had little to no impact on the effectiveness of our approach. Our approach demonstrated strong performance across various model architectures and training protocols. Its strength was mostly evident when sourcing data from multiple web engines (Google, Flickr, DuckDuckGo Bing), effectively handling diverse data distributions. Surprisingly, even after cleaning our web data using deduplication and automatic removal of noisy samples, reducing the data size by 30%, the performance remained unaffected. This suggests that the main challenges are likely due to out-of-domain instances rather than reducing noise or duplicate samples. Lastly, class-balanced sampling does not affect the performance of our model, indicating that further exploration of long-tailed loss functions [Karthik et al., 2021], may not yield significant improvements.

7.4.3 Comparison with Name-Only Approaches

We now address our second question: ② How does the performance of webly-supervised datasets compare to the latest name-only classification approaches? Can web data surpass the latest methods tailored for vision-language models, such as CLIP, in a name-only classification context?

Comparison with Recent Advances. Traditional name-only classification methods are often built upon zero-shot CLIP (CLIP-ZS) [Radford et al., 2021]. CLIP-ZS works by using text prompts that contain category names to classify images. For each test data point, it predicts the class by finding the category name prompt that best matches the input image. Recent research has introduced improvements to this approach in three main areas:

(i) *Better Text Prompts:* Methods like VisDesc [Menon and Vondrick, 2022], CuPL [Pratt et al., 2023] and WaffleCLIP [Roth et al., 2023] have explored more effective text prompts to enhance

classification accuracy; (ii) *Creating Pseudo-training Datasets*: Approaches such as Glide-Syn [He et al., 2022] and Sus-X [Udandaraao et al., 2023], and Neural Priming [Wallingford et al., 2023] focus on creating training datasets either by retrieval from LAION5B or generating samples from diffusion models to improve model performance, with retrieval being better [Burg et al., 2023]; (iii) *Enhanced Adapters*: CALIP [Guo et al., 2023], along with Glide-Syn [He et al., 2022] and Sus-X [Udandaraao et al., 2023] propose improved adapters for CLIP models to enhance their classification abilities. There are alternative approaches, like SD-Clf [Li et al., 2023a], which shows the effectiveness of stable-diffusion models for classification tasks. Additionally, CaFo [Zhang et al., 2023b] explores chaining different foundation models for tasks including name-only classification. We describe these approaches in detail in Appendix C.

Results. To evaluate our approach, we compare it against recent strategies using the ResNet50 CLIP model for a fair comparison. The results are summarized in Table 7.5.1; comparisons on CLIP ViT-B/16 model can be found in Appendix C. Consistently, our approach outperforms other leading methods such as CaFo and SuS-X-LC, with performance improvements between 2-25%. Additionally, we apply our approach to vision-only ResNet50 MoCoV3 model trained on ImageNet1K. Notably, this often performs significantly better out-of-the-box than CLIP variants, with impressive improvements of 2-8%, offering new insights on recent works [Li et al., 2023b]. Moreover, employing an MLP Adapter results in a 1-4% boost in performance over linear probing, and this is achieved with minimal added computational cost when compared to extracting features from a ResNet50 model.

Why Does Our Webly-Supervised Data Outperform Other Approaches? A fundamental factor in the superior performance of our approach is again the scale of our uncured webly-supervised data. We download roughly ten times larger than what is used in alternative approaches (detailed in Appendix C). One might wonder: why not just scale up the datasets used by other methods? Retrieval-augmented techniques such as SuS-X [Udandaraao et al., 2023] and Neural Priming—our [Wallingford et al., 2023] closest competitors performance-wise—experience stagnation or even a decline in results when expanded to 100 samples per class, as illustrated in Figure 6 of Udandaraao et al. [2023] and discussed in Appendix B of Wallingford et al. [2023]. Conversely, our method still achieves marked improvements in accuracy even as dataset sizes approach 500 – 750 samples per class, as previously highlighted in Table 7.4.3. Alternative dataset generation methods, like Diffusion models [He et al., 2022, Zhang et al., 2023b], come with a significant computational cost, yet they do not surpass retrieval methods such as LAION-5B in performance [Udandaraao et al., 2023, Burg et al., 2023]. To provide some context, producing a dataset equivalent in size to ours (~ 150K samples) using generative techniques like stable-diffusion demands a staggering

Table 7.4.3: Impact of Size of Queried Webly-Supervised Data on Performance. This table illustrates the influence of downsizing our queried web data by considering only the top- k queries for download. Notably, a substantial performance drop occurs as the dataset size decreases. Despite the higher quality of the top- k samples, their limited quantity adversely affects performance. We use Manually Annotated Training (MA) Data as a reference point.

Eval.	Dataset Size	Datasets				
		FGVC Aircraft	Flowers102	OxfordIIITPets	Stanford Cars	BirdSnap
Linear Probe	MA Data	38.5%	83.3%	89.8%	56.3%	46.2%
	Top-10/engine/class	18.3% (-20.2%)	64.3% (-19.0%)	82.0% (-7.8%)	34.3% (-22.0%)	36.4% (-9.8%)
	Top-20/engine/class	33.8% (-4.7%)	71.7% (-11.6%)	87.3% (-2.5%)	45.7% (-10.6%)	42.1% (-4.1%)
	Top-50/engine/class	40.8% (+2.3%)	77.7% (-5.6%)	88.7% (-1.1%)	57.9% (+1.6%)	48.5% (+2.3%)
	Top-100/engine/class	52.4% (+13.9%)	80.8% (-2.5%)	90.1% (+0.3%)	64.6% (+8.3%)	52.4% (+6.2%)
	Top-200/engine/class	56.6% (+18.1%)	82.7% (-0.6%)	90.7% (+0.9%)	67.8% (+11.5%)	54.6% (+8.4%)
	All Data	57.5% (+19.0%)	85.7% (+2.4%)	91.7% (+1.9%)	62.1% (+5.8%)	56.1% (+9.9%)
MLP-Adapter	MA Data	46.0%	80.3%	89.7%	57.6%	47.7%
	Top-10/engine/class	32.2% (-13.8%)	66.0% (-14.3%)	86.7% (-3.0%)	39.8% (-17.8%)	36.4% (-11.3%)
	Top-20/engine/class	37.4% (-8.6%)	72.6% (-7.7%)	88.1% (-1.6%)	49.5% (-8.1%)	42.3% (-5.4%)
	Top-50/engine/class	51.4% (+5.4%)	77.3% (-3.0%)	89.9% (+0.2%)	61.1% (+3.5%)	47.9% (+0.2%)
	Top-100/engine/class	58.2% (+12.2%)	81.9% (+1.6%)	90.6% (+0.9%)	65.8% (+8.2%)	50.1% (+2.4%)
	Top-200/engine/class	63.4% (+17.4%)	84.1% (+3.8%)	91.9% (+2.2%)	70.3% (+12.7%)	54.4% (+6.7%)
	All Data	65.5% (+19.5%)	87.1% (+6.8%)	92.8% (+3.1%)	66.8% (+9.2%)	53.7% (+6.0%)

Table 7.5.1: Comparison with Name-Only Classification Techniques with ResNet50: When comparing with existing state-of-the-art name-only classification techniques, we show that our method outperforms those methods by margins ranging from 2% to 25%.

Type	Method	Model	Birdsnap	Aircraft	Flowers	Pets	Cars	DTD
Data-Free	CLIP-ZS [Radford et al., 2021]	CLIP	32.6	19.3	65.9	85.4	55.8	41.7
	CaFo-ZS [Zhang et al., 2023b]	CLIP	-	17.3	66.1	85.8	55.6	50.3
	CALIP [Guo et al., 2023]	CLIP	-	17.8	66.4	86.2	56.3	42.4
	CLIP-DN [Zhou et al., 2023b]	CLIP	31.2	17.4	63.3	81.9	56.6	41.2
	CuPL [Pratt et al., 2023]	CLIP	35.8	19.3	65.9	85.1	57.2	47.5
	VisDesc [Menon and Vondrick, 2022]	CLIP	35.7	16.3	65.4	82.4	54.8	42.0
	SD-Clf [Li et al., 2023a]	SD-2.0	-	26.4	66.3	87.3	-	-
Use-Data	GLIDE-Syn [He et al., 2022]	CLIP	38.1	22.0	67.1	86.8	56.9	43.2
	CaFo [Zhang et al., 2023b]	CLIP	-	21.1	66.5	87.5	58.5	50.2
	SuS-X-LC [Udandara et al., 2023]	CLIP	38.5	21.1	67.1	86.6	57.3	50.6
	SuS-X-SD [Udandara et al., 2023]	CLIP	37.1	19.5	67.7	85.3	57.2	49.2
	C2C (Ours-Linear Probe)	CLIP	48.1 (+9.6)	44.0 (+22.0)	82.0 (+14.3)	88.1 (+0.6)	71.3 (+12.8)	57.1 (+6.5)
	C2C (Ours-MLP Adapter)	CLIP	46.6 (+8.1)	48.9 (+26.9)	84.8 (+17.1)	89.4 (+1.9)	72.6 (+14.1)	57.6 (+7.0)
	C2C (Ours-Linear Probe)	MocoV3	56.1 (+17.6)	57.5 (+35.5)	85.7 (+18.0)	91.7 (+4.2)	62.1 (+3.6)	54.6 (+4.0)
	C2C (Ours-MLP Adapter)	MocoV3	53.7 (+15.2)	65.5 (+43.5)	87.1 (+19.4)	92.8 (+5.3)	66.8 (+8.3)	55.8 (+5.2)

32 hours of computation on an 8 A100 GPUs. In contrast, our approach collects the same dataset in around 15 minutes using a basic CPU machine.

7.5 Continual Webly-Supervised Learning

③ Building upon our prior observations regarding the efficiency of collecting webly-supervised data and its effectiveness for *name-only* classification, we now test this approach in the context

of *continual name-only* classification. Within this framework, the learner is solely provided with category names, and potentially descriptions, necessitating the continuous and streamlined construction of data and updation of the classifier. To assess the robustness and adaptability of our approach, we subject it to a diverse range of data streams encountered in various continual learning scenarios, namely: (i) *class-incremental*: the incremental addition of classes, (ii) *domain-incremental*: incremental adaptation to known domain shifts, and (iii) *time-incremental*: the gradual incorporation of new data over time. The subsequent subsection presents a comprehensive overview of the experimental setup and the corresponding results obtained from these three scenarios.

7.5.1 Experimental Details

Datasets: We assess the effectiveness of our uncurated webly-supervised data in three different continual learning (CL) scenarios. For each scenario, we compare the performance of our downloaded data with manually annotated data. This evaluation setup aligns with the traditional CL paradigm, where labeled training data is revealed sequentially in the data stream. It is worth noting that methods with access to manually annotated data naturally have an inherent advantage. In principle, manually annotated data serves as a soft upper bound to our webly supervised approach. However, our primary goal is to determine to what extent web-supervised datasets can bridge this performance gap, with extreme limits of < 1 hour and cost $< \$15$ on AWS servers. Our experiments focus on the following three CL setups:

Class-Incremental: In this setting, we use CIFAR100, which is partitioned into ten timesteps, where at each timestep ten new class categories are introduced. CIFAR100 exhibits a notable domain gap due to its samples being old, centered, and downsampled to 32x32 pixels. To match this resolution, we downscale our images to 32x32 as well. The queries provided in this case simply consist of the class names for all previously encountered classes.

Domain-Incremental: In this setting, we use PACS [?] dataset, which comprises four timesteps and is suitable for the domain-incremental setup. Each timestep introduces new domains, namely Photos, Art, Cartoon, and Sketches. The primary challenge here lies in adapting to the distinct visual styles associated with each domain. The queries in this case are composed of a combination of class names and the names of the visual domains.

Time-Incremental: In this setting, we use CLEAR10 [Lin et al., 2021] dataset, a recently popular CL dataset, by incorporating timestamps from the CLEAR benchmark into web queries⁵. Our web

⁵Timestamps from: https://github.com/linzhiqu/continual-learning/blob/main/clear_10_time.json

queries for categories are consistent across timesteps, however, samples are filtered by timestamp to match CLEAR time categorization. Here we only use Flickr as it supports timestamped querying.

Optimizing Data Collection. To optimize the creation of our webly-supervised datasets while adhering to time constraints, we conduct additional experiments involving the retrieval of only the top- k most relevant samples per search engine. Specifically, we explore two settings: $k = 20$ and $k = 50$. This approach significantly diminishes the cost and time associated with querying the web and feature extraction processes.

Training Models. We note that we do not restrict the storage of past samples, unlike previous literature as download links largely remain accessible. If some download link expires then we do not that sample, allowing realistic privacy evaluation. However, we note that no links expired in the duration of our study and only a small fraction ($<5\%$) of links of CLOC dataset collected until 2014 have become invalid until today. Hence, we follow the constraints specified in Prabhu et al. [2023b] limiting the computational budgets and without storage constraints. We train a linear probe under varying Linear probing results are compared to NCM [Mensink et al., 2013, Janson et al., 2022] and KNN [Malkov and Yashunin, 2018, Prabhu et al., 2023a] classifiers. We use a ResNet50 MoCoV3 backbone for all experiments since SSL training has been shown to help in CL tasks [Gallardo et al., 2021]. For linear probing, we use the same optimization parameters provided earlier except that we constrain the iterations according to our compute budgets C_t . For more details about the computational budgets please refer to Appendix D. We set $k = 1$ and use cosine distance for KNN. During the training process, we implement experience replay and utilize class-balanced sampling to select training batches from the previously collected samples.

Metrics. We compute the average incremental accuracy for all three settings [Rebuffi et al., 2017b]. Briefly, we compute the accuracy on the available test set after finishing training of each timestep, which gives us a graph of accuracies over time. The average incremental accuracy is the aggregate measure of these incremental accuracies, which gives average performance of the method over time.

7.5.2 Results

We evaluate the efficacy of our uncured web data in the context of continual name-only learning and compare the results with manually annotated datasets in various scenarios. Linear probing results are presented in Table 7.5.2, while additional results for NCM and KNN can be found in the Appendix. Despite having access solely to class/category names, our uncured webly-supervised data achieves accuracies that are close to training on the manually annotated datasets. We note that

Table 7.5.2: Linear Probe Performance in Continual Learning Scenarios (Avg. Acc. $\uparrow$). Our uncurated webly-supervised data achieves average accuracy close to manually annotated (MA) datasets in a continual learning context with relatively small performance gaps.

Eval Dataset	Split-CIFAR100	Split-PACS	CLEAR10
MA Data	43.2%	82.8%	70.0%
C2C (Ours)	38.7% (-4.5%)	80.8% (-2.0%)	65.3% (-4.7%)
C2C (Top-20/engine/class)	39.2% (-4.0%)	79.9% (-2.9%)	62.0% (-8.0%)
C2C (Top-50/engine/class)	39.5% (-3.7%)	78.6% (-4.2%)	60.8% (-9.2%)

the performance on manually annotated datasets serves as an upper bound and not a fair comparison as they require expensive curation process, they are well-aligned with the test distribution as both sets are created from the same sampling of data. The performance gap between them is small, ranging from 2-5%, with the exception of CLEAR10 where it reaches 5-10%. In Table 7.5.4, we also consider the time required for querying and downloading our uncurated continual webly-supervised data. Remarkably, we are able to generate the web data within minutes, instead of days, across a variety of continual learning scenarios allowing more budget for computational resources. All experiments were completed in < 1 hour and cost $< \$15$ on AWS servers. This approach delivers both a performance comparable to manually annotated datasets and significantly reduces associated expenses, which typically exceed \$4500.

Understanding the Performance Gap: While our webly-supervised dataset (C2C) has shown promise, a performance discrepancy exists when compared to the manually annotated data (MA Data). This performance lag, is only slightly behind the ideal scenario of using in-distribution annotated data. The natural question that arises is: Why cannot we bridge this performance gap and possibly exceed it, as observed in Section 7.4? Two primary distinctions from Section 7.4 can explain this: **(i)** The current training operates within a limited computational budgets, and **(ii)** The size difference between our webly-supervised continual datasets and manually annotated datasets has notably shrunk, transitioning from a substantial 30 – 100 $\times$ difference to a mere 2 – 3 $\times$. It is important to note that in Section 7.4, when we match the size of the manually annotated datasets by considering only the top-20 web queries, we observe a similar gap to that in this section. Nevertheless, the improvements in speed and reduction in annotation costs significantly outweigh this gap.

Firstly, in the case of PACS, a significant domain gap arises between web sketches which refer to line-drawings and manually annotated sketches which refer to quick-draws. This domain shift results in a performance gap, which is challenging to bridge with the inclusion of additional sketch data from the internet. Second, in CIFAR100, images are carefully selected and often do not reflect real-world data streams. Specifically, they consist of older, centered, and downsampled

Table 7.5.3: Comparing Last-Layer Based Continual Learning Approaches in Name-Only Continual Learning (Avg. Acc. $\uparrow$). We evaluate the average accuracy of various continual learning methods in a name-only continual learning scenario with constrained computational resources. Surprisingly, KNN achieves superior results compared to linear probing, even while operating within a lower computational budget than the "tight" setting.

Classifier	Budget	Split-CIFAR100	Split-PACS	CLEAR10
C2C-NCM	<Tight	48.9%	77.4%	60.7%
C2C-Linear	Tight	31.9%	75.8%	56.1%
C2C-Linear	Normal	38.7%	80.8%	65.3%
C2C-Linear	Relaxed	49.9%	84.2%	71.6%
C2C-KNN	<Tight	59.8% (9.9%)	89.5% (5.3%)	81.2% (9.6%)

Table 7.5.4: (Left) Time Benchmark (in minutes): Evaluation of the approximate total time required for continual learning across all time steps using a linear classifier. Our pipeline demonstrates exceptional time efficiency compared to manual sample annotation. **(Right) EvoTrend Results:** Comparative performance analysis of Linear Probing on C2C using MoCoV3 ResNet50 against zero-shot CLIP with ResNet50, indicating improved performance.

Components	Benchmarks			Method	Budget	Avg Acc. ($\uparrow$)
	Split-CIFAR100	Split-PACS	CLEAR10			
Download Images	~15 mins	~5 mins	~14 mins	ZS-CLIP RN50	-	62.9%
Extract Features	~15 mins	~6 mins	~10 mins	C2C-Linear	Tight	57.7% (-5.2%)
Model Training	~2 mins	~0.5 min	~1 min	C2C-Linear	Normal	69.0% (+6.1%)
				C2C-Linear	Relaxed	72.8% (+9.9%)
Overall	~ 32 mins	~ 12 mins	~ 25 mins	C2C-NCM	<Tight	65.7% (+2.8%)
				C2C-KNN	<Tight	78.6% (+15.7%)

images, which strongly contrast with the dynamic and varied nature of web data harnessed in our approach. This difference highlights the importance of considering more realistic data streams, over handpicked and potentially unrepresentative datasets. Lastly, in the context of CLEAR10, our analysis uncovers data collection inaccuracies, particularly in the bus class. While our web datasets consist of images depicting the exterior of buses, the manually annotated CLEAR10 dataset primarily includes interior images of buses in the train/test set. Given that the bus class constitutes one out of ten classes, our webly-supervised approach directly incurs a 10% performance loss illustrated in Figure 4 in Appendix D. We delve deeper into this domain gap issue in Appendix D.

7.5.3 EvoTrends: The First Continual Name-Only Classification Benchmark

To demonstrate the adaptability, speed, and effectiveness of our webly-supervised approach to continual name-only classification, we introduce a novel dataset titled “EvoTrends” (depicted in the Appendix D). Instead of resorting to synthetic methods that produce class-incremental streams based on arbitrarily chosen classes, EvoTrends offers a more natural scenario. Spanning two decades (2000 to 2020), it chronicles a sequence of trending products that have risen in popularity, from the PlayStation 2 in 2000 to face masks in 2020. This dataset spans 21 unique timesteps and comprises 39 different classes. EvoTrends, presents a real-world class-incremental

challenge where new “trend” classes emerge naturally over time. With our approach, and without requiring any specific alterations to our method, we query and download webly-supervised data for each timestep. Within minutes, our model could continually adapt and effectively categorize these evolving trends, establishing its viability for applications demanding rapid adaptation to unlabeled, continuously updated data or development of new classifiers. The dataset was divided into 36,236 training samples and 11,317 testing samples, with the test set cleaned using automated deduplication and anomaly detection pipelines.

Results. Preliminary evaluations on the EvoTrends dataset have yielded promising results. Our method was compared to the zero-shot CLIP. It is noteworthy that CLIP may have already been trained on several EvoTrends classes during its pretraining phase, as illustrated in Table 7.5.4 (right). Impressively, our model, while using a ResNet50 MoCoV3 and budgeting compute, outperformed CLIP-RN50 by 6.1%. This further underscores the potency of our method. Consistent with previous sections, KNN stands out as the most proficient performer within our computational parameters.

7.6 Conclusion

In conclusion, the traditional reliance on costly and time-intensive manual annotation for Continual Learning is increasingly becoming infeasible, especially in rapidly changing environments. Our exploration into the *name-only continual learning* scenario presents a challenging and novel CL setting. Our proposed solution which leverages uncured webly-supervised data (C2C), does not only manage to reduce the annotation costs significantly but also maintain a high level of classifier performance. Addressing key questions regarding the reliability, comparison to state-of-the-art name-only classification, and adaptability across various continual learning settings, we demonstrate that webly-supervised data can offer a robust and efficient alternative to manual data curation. We also present *EvoTrends* a dataset that highlights our capability to swiftly adapt to emerging trends using web-supervision. This covers the efforts in this thesis to alleviate labeling bottlenecks for large-scale continual learning in a computationally budgeted scenario. Lastly, I shall present the entire proposed system again alongside other concluding thoughts in the next chapter.

8

Conclusion

In this integrated thesis, we lay the foundations for designing a scalable continual learning system with a focus on computational cost considerations.

We first address the need for transitioning to this new paradigm by highlighting the limitations of current continual learning approaches in Chapters 2 and 3. In Chapter 2, we show that traditional methods often rely on memory-intensive techniques or fail to leverage all available data effectively, underscoring the necessity for removing the memory limitations and focusing on minimizing computational cost. In Chapter 3, we show that continual representation learning has been hindered by small datasets and a lack of scalable solutions, motivating the scalable continual learning focus of this thesis.

Chapter 4 and 5 introduced the new paradigm of continual learning under computational considerations, while Chapters 6 and 7 investigate metrics and data aspects of this framework respectively. This framework challenges existing practices by proposing a slow-fast learner combination system. Chapter 4 introduces the "slow learner" aspect of the system, demonstrating why simply continually finetunes a learner class-balanced sampling for replay seems to be current best alternative. Chapter 5 discusses "fast learner" aspect of the system, concluding that an approximate k-NN is especially suitable for rapid learning, both chapters demonstrating viability for large-scale applications. Chapter 6 investigates critical issues in current metrics, motivates the intuitions behind why they are bad and extends this to suggest simple but effective alternative metrics. Finally, Chapter 7 further explores strategies for data acquisition within the same cost constraints.

We take a closer look into these contributions in Section 8.1, outline some interesting open problems in Section 8.2, and conclude with final thoughts in Section 8.3.

8.1 Summary of Contributions

We detail contributions made in each chapter below:

Chapter 2– GDumb: A Simple Approach that Questions Our Progress in Continual Learning

This chapter introduces GDumb, a straightforward strategy that merely storing a subset of samples and retraining the model from scratch when tested. The fact that GDumb outperforms complex, state-of-the-art continual learning algorithms, despite its simplicity, raises critical questions about the progress and direction of continual learning research. Specifically, it questions the progress in the limited memory setting. This experiment indicates continual learning approaches are only retaining information present in samples in their memory, and not preserving past knowledge. Another point raised is the efficacy of rewinding from the initial pretrained model as continual learning approaches seem to strictly decline in quality over time.

Chapter 3– RanDumb: A Simple Approach that Questions the Efficacy of Continual Representation Learning

This chapter introduces RanDumb, the result of our investigation into continual representation learning, which reveals that using random embeddings or fixed pre-trained networks often surpasses the performance of continually trained representations. This observation is both surprising and concerning, as it contradicts the expected benefits of continual learning in enhancing representation learning over time. Our results suggest a re-examination of current approaches and advocate for the consideration of alternative, more promising settings with scalable datasets to better advance the field.

Chapter 4– Computationally Budgeted Continual Learning: What Does Matter?

This chapter introduces the paradigm of offline continual learning (slow learner) with computational constraints considerations. It first investigates the suitability of existing continual learning methods under computational constraints, demonstrating that simple naive strategies based on experience

replay often outperform more sophisticated algorithms. This finding challenges the practical applicability of current CL methods and emphasizes the necessity for more efficient solutions that are feasible within real-world computational budgets. Currently, we advocate a simple system of continual finetuning with all past data, samples selected by class balanced sampling strategy. We recommend future efforts should enhance the efficiency and effectiveness of these simple methods.

Chapter 5– Online Continual Learning Without the Storage Constraint

This chapter introduces the paradigm of online continual learning (fast learner) in environments where computational resources are more constrained than storage. This chapter demonstrates employing an approximate kNN-based approach that allows for rapid adaptation to new data without forgetting previous information, our method demonstrates significant improvements over existing CL algorithms. This work advocates for studying computational efficiency and scaling online continual learning to handle rapid updates in large-scale datasets between training runs of the slow learner.

Chapter 6– Rapid Adaptation in Online Continual Learning: Are We Evaluating It Right?

In this chapter, we critique the standard metric of online accuracy for evaluation of online continual learning algorithms, proposing a new metric that better captures the essence of adaptation. Our investigations reveal that many current OCL methods fail to adapt effectively to new data, instead overfitting to peculiarities of the data stream. We introduce near-future accuracy, a more reliable measure for evaluating adaptation in online continual learning, highlighting the need for algorithms that genuinely learn and adapt rather than memorize transient patterns.

Chapter 7– From Categories to Classifier: Name-Only Continual Learning by Exploring the Web

This chapter first demonstrates the impracticality of traditional data collection and labeling at scale. It explores the potential of leveraging webly-supervised data for continual learning. By adapting to new categories using just names and uncured web data, our method demonstrates a promising direction for reducing reliance on manual annotation. Our findings, supported by the introduction of the EvoTrends dataset, illustrate the feasibility of utilizing web-supervised data for efficient and effective continual learning, opening new avenues for research in this area.

We highlight some exciting open problems in the next section.

8.2 The Road Ahead and Open Problems

There are still several important challenges remaining to make continual learning systems deployable in computationally constrained settings beyond extensions to broader computer vision tasks such as dense prediction [Gui et al., 2024]. We list a few exciting avenues below for the curious reader.

Continual Learning Methods. Our innovative approach opens up several promising avenues for enhancing models in a computationally efficient way.

(i) *Integrating Slow and Fast Updates into a Unified System.* We have developed simple yet effective solutions for continual learning that include fine-tuning models to prevent catastrophic forgetting in Chapter 4 and employing a kNN-based system for rapidly adapting to new, incoming information in Chapter 5. The goal is to create a scalable system that merges these slow and fast updating mechanisms into a unified framework. This integration would use backfilling techniques [Seo et al., 2023] to reduce feature drift between backbone updates for the kNN, ensuring a smooth transition and maintaining model integrity over time.

(ii) *Transferring Knowledge Across Architectures.* As new techniques are proposed, there is a frequent change in components of large ML models like LLMs. A crucial area of research is exploring how to transfer accumulated knowledge across different model architectures with minimal loss of information, which involves managing complex, non-linear dynamics that can significantly impact model performance. A particularly challenging yet unexplored aspect of this is adding new modalities to multimodal models.

(iii) *Mitigating the Stability Gap [De Lange et al., 2022].* Integrating new knowledge into models often requires high learning rates or similar adjustments, which, while increasing adaptability, can lead to significant forgetting of a vast amount of existing knowledge — a problem known as concept forgetting [Mukhoti et al., 2023]. An exciting research question is how low-rank adapters, which are gaining popularity, can be used to effectively bridge this stability gap over extended periods, ensuring both stability and plasticity within the learning process, as done in [Harun and Kanan, 2023].

Improving Online Continual Learning Metrics. Metrics for computing rapid adaptation need a lot more improvement beyond our contributions in Chapter 6. We list some exciting avenues for extending our exploration below:

(i) *Beyond Label Correlations.* We currently only measure and remove correlations from labels $p(y)$, while covariate correlations $p(X)$ are harder to reliably isolate, requiring further investigation.

(ii) *Why aren't information retention and adaptation at odds?* Our results suggest that there is still a lot of room for improvement in OCL methods, as we are far from the pareto frontier where information retention and rapid adaptation are at odds.

Improving Continual Data Collection. Continual data-collection can be a promising direction on the intersections of continual learning and data-centric ML, two emerging fields of literature. We provide some open problems for extending the work in Chapter 7 below:

(i) *Extending the Setting to Test-Time Adaptation using the Web:* An interesting extension is to incorporate unlabeled test images together with their corresponding categories, enabling test-time adaptation. This allows retrieval-based refinement of web search results [Oquab et al., 2023]. This additionally allows gathering webly-supervised data close to test samples using reverse-image search-based retrieval on the web.

(ii) *Online Continual Learning:* Our method is not yet equipped to handle the challenges posed by online continual learning streams, where we have to rapidly adapt to incoming stream of images. It may worsen the issues related to data correlations due to repeated scraping of trending images from various sources [Hammoud et al., 2023]. Cleaning and analyzing name-only continual learning scenarios in a online fashion, with rapid distribution shifts, presents unique challenges, especially without labels, which are currently beyond the scope of our work.

(iii) *Data Collection Limitations:* Although our approach is effective in many scenarios, it may not be suitable for a significant portion of niche domain datasets, particularly in fields like medicine where manual collection remains the primary source due to data sharing constraints. It is important to verify and acknowledge the limitations of the name-only classification setting itself.

8.3 Concluding Remarks

Collectively, this thesis present a comprehensive overview of current challenges and an introduction to the new paradigm of computationally budgeted continual learning, and the effective use of emerging data sources like the web. We particularly emphasize the need for focusing on practical, scalable solutions that address the real-world challenges of computational resources and data annotation capabilities.

Appendices



GDumb: A Simple Approach that Questions Our Progress in Continual Learning

In the appendix, we provide additional information and experiments to supplement the observations in the main paper. The appendix is organized as follows:

Contents

A.1 Additional Experiments	116
A.2 Experiment formulations	117

Form.	Dataset	Model	#Params	k	Acc
B1	MNIST	MLP-400	479K	4400	97.8
		NIN	260K	250	97.2
		Improv	1.8x	17.6x	-0.6%
B1	SVHN	ResNet18	11.2M	4400	93.4
		NIN	968K	2000	89.3
		Improv	11.6x	2.2x	-4.1%
A1/A2	CIFAR10	ResNet18	11.2M	500	45.8
		DenseNet100	769K	500	47.5
		Improv	14.6x	-	1.7%
D	CIFAR100-Task	ResNet18-S	1108K	1105	60.3
		DenseNet100	794K	1105	62.1
		Improv	1.4x	-	1.8%
B2	CIFAR100-Class	ResNet18	11.2M	2000	24.1
		DenseNet100	800K	2000	27.5
		Improv	14.6x	-	3.4%

Table A.0.1: Minimizing the resource consumption (storage size and parameters).

A.1 Additional Experiments

We perform additional experiments to study variance of performance across memory sizes, models, training timesteps on each dataset. We first study the variation across models and different values of k and tabulate these results in Table A.0.1. For MNIST and SVHN, we could reduce the replay buffer size by over 17x and 2x respectively with small impact on performance, a good tradeoff. For CIFAR10 and CIFAR100, we substitute the bulky ResNet18 model with a DenseNetBC-100-12 [Huang et al., 2017] which gives us a major decrease of over 14x in parameters. In CIFAR10 and CIFAR100 (Task-IL and Class-IL), we additionally improve accuracy by 1.7%, 1.8% and 3.4% respectively after compressing the models. On MNIST and SVHN, we could compress the size of stored dataset k , showing existing formulations use too much memory. But as dataset complexity increases, larger memory size is required. We could also compress the parameter storage by large margins while increasing accuracy by substituting popular but efficient models, showing the scope for using more efficient models for CL formulations.

We further study the trade-off between accuracy and training time of our improved models on the improved k values by varying the passes on memory. We present the results in Table A.1.1, with **bold** being the selected models, along with (brackets) representing accuracies obtained by an ablation without cutmix regularization. We show that we can further reduce our training time by

Passes/Form.	B1-MNIST	B1-SVHN	A1-CIFAR10	D	B2	C2
k	250	2000	500	1105	2000	9000
	NIN	NIN	DenseNet100	DenseNet100	DenseNet100	DenseNet100
8	91.7 (96.0)	72.5 (81.4)	28.4 (28.4)	49.5 (50.8)	7.3 (8.8)	32.3 (33.9)
16	95.9 (96.3)	85.1 (85.6)	32.3 (33.5)	52.6 (53.0)	10.0 (11.4)	39.4 (41.9)
32	96.9 (97.2)	88.8 (87.3)	37.8 (36.7)	56.2 (58.3)	15.0 (18.1)	47.2 (48.4)
64	97.4 (97.2)	89.2 (87.5)	39.9 (40.1)	60.6 (61.2)	21.9 (22.5)	54.0 (54.3)
128	97.4 (97.2)	89.2 (88.9)	43.9 (42.9)	62.1 (61.9)	26.5 (24.1)	56.9 (56.3)
256	96.6 (97.5)	88.4 (89.7)	46.4 (43.9)	62.0 (62.4)	27.6 (25.7)	54.1 (54.8)
512	96.1 (97.7)	86.9 (88.4)	47.5 (45.9)	62.3 (61.5)	27.6 (25.8)	54.0 (54.7)

Table A.1.1: Accuracy of tweaked (NIN/DenseNet) GDumb models with number of passes. The bolded accuracies represent the reported results in previous experiments, while accuracies in (brackets) are obtained without cutmix regularization. We can halve the training time with slight tradeoff of upto 1% accuracy.

half with a minor (approx 1%) tradeoff in accuracy. We also observe that cutmix regularization improves performance by 0.2% to 1.5% margins. We strongly recommend regularization in GDumb since it only has access to the few samples in memory.

A.2 Experiment formulations

We first detail each of the seven selected formulation and then present results on each of them.

Formulation A1 ([Aljundi et al., 2019a]): We benchmark on two datasets: MNIST and CIFAR10 from this setting. They randomly divided MNIST and CIFAR10 into 5 disjoint tasks of 2 classes each. The architectures used for MNIST is a MLP with 2 hidden layers of 400 nodes and ResNet18 for CIFAR10. They use a small limit (k) of 300 and 500 stored samples for MNIST and 200, 500 and 1000 stored samples for CIFAR10. In MNIST, they select 1000 samples per task, while in CIFAR10 they utilize 9,750 samples per task in the online stream. On MNIST, we additionally compare with GSS [Aljundi et al., 2019b] who use the same setup except they have an MLP with 100 hidden size. We compare accuracy on the hold-out test set after all tasks are done.

Formulation A2 ([Jin et al., 2021]): They split MNIST and CIFAR10 into 5 disjoint subsets by their labels as different tasks. Each task consists of 1,000 online training examples in MNIST and 10,000 training examples in CIFAR10 similar to the above setup. The goal is to classify over all 10 classes on a held-out test set when the training ends. The architectures used for MNIST is a MLP with 2 hidden layers of 100 nodes and ResNet18 for CIFAR10, identical to the above setup. However, since the accuracies obtained for ER, GEM and ER-MIR are very different, we list them as a different formulation and compare our performance against them.

Formulation A3 ([Ji et al., 2020]): They split MNIST and CIFAR10 into 5 disjoint subsets. Each task consists of 1,000 online training examples in MNIST and 9,750 training examples in CIFAR10 similar to the A1. The goal is to classify over all 10 classes on a held-out test set when the training ends. The architectures used for MNIST is a MLP with 2 hidden layers of 100 nodes and ResNet18 for CIFAR10, identical to the above setup. However, since the accuracies obtained for ER, GEM and ER-MIR are very different (for MNIST), we list them as a different formulation as merging tables is not possible.

Formulation B1 ([Hsu et al., 2018, Rajasegaran et al., 2019]): It consists of two datasets: MNIST and SVHN, randomly divided into 5 disjoint tasks of 2 classes each. The architecture used is a MLP with 2 hidden layers of 400 nodes for MNIST and ResNet18 for SVHN. The formulation controls the total static memory overhead among all proposed approaches, resulting in storage capacity of 4400 for memory-based approaches [Hsu et al., 2018].

Formulation B2 ([Rebuffi et al., 2017b]): The formulation use CIFAR100, split into 20 tasks of 5 classes each. They use a ResNet-32 model and a limit of 2000 stored samples (k). We additionally report the average of accuracy after each task as described in [Rebuffi et al., 2017b] referred to as accuracy (avg in t), along with accuracy after all tasks referred to as accuracy.

Formulation B3 ([Douillard et al., 2020]): This formulation tests small-task increments on CIFAR100 and Imagenet100. Given a class-order, they use the first 50 tasks for pretraining and then subsequent 50 tasks with 1 class each in a CI-CL fashion including the initial 50 classes. Hence, they have a CI-CL classification with 51-100 classes. They start off with 1000 samples in memory and add 20 samples to memory for each subsequent task added. They use ResNet32 on CIFAR100 and ResNet18 on Imagenet100 to match the formulation and provide 3 class-orders for CIFAR100 and 1 class-order for Imagenet100 which we use. We measure average and last-task accuracy on using the same set of class-orders, using the same memory and networks as specified in the formulation.

Formulation C1 ([Hsu et al., 2018]): It splits MNIST into 5 disjoint tasks of 2 classes each. The architecture used is a MLP with 2 hidden layers of 400 nodes for MNIST. The formulation controls the total static memory overhead, resulting in storage capacity of 4400 for memory-based approaches. We use three different class-to-task mappings to get performance.

Formulation C2 ([De Lange et al., 2020]): TinyImagenet is divided into 10 disjoint tasks of 20 classes each in this formulation. We compare with the overall best accuracies obtained (Table 10 in [De Lange et al., 2020]). The table lists their best performance observed over different architectures, regularization strategies, hyperparameter searches, etc. We test for two stored sample limit (k): 4500

and 9000. We use DenseNetBC-100-12 architecture as detailed in subsequent sections. Note that although it differs from the architectures tested; it is a fairly standard efficient architecture.

Formulation D ([Chaudhry et al., 2019a]): We borrow the benchmark on CIFAR100, consisting of 20 disjoint tasks of 5 classes each. We train 17 tasks, store upto 13 samples per class and use the same reduced ResNet18 architecture.

Formulation E ([Aljundi et al., 2019b]): It consists of two datasets (MNIST and CIFAR10) with class-imbalanced, blurry boundary setting. Each has 5 tasks, each task having 2 classes each. MNIST has 2000 online samples of current task and 200 from each other tasks for every task in the formulation. Similarly, in CIFAR10 we keep 90% of the data for each task, and introduce 10% of data from the other tasks. The same architectures are used as in GSS [Aljundi et al., 2019b], which are a 2-layer MLP with hidden size 100 for MNIST and a ResNet18 for CIFAR10. The limit on stored samples (k) is 300 for MNIST and 500 for CIFAR10.

B

Computationally Budgeted Continual Learning: What Does Matter?

In the appendix, we provide additional information and experiments to supplement the observations in the main paper. The appendix is organized as follows:

Contents

B.1	Motivation: Cost and Privacy Reasons for Restricting Memory are Flawed	121
B.2	Dataset Construction	122
B.2.1	Constructing Imagenet2K	122
B.2.2	Constructing Continual Google Landmarks V2	123
B.3	Estimation of Equivalent Iterations	123
B.4	Additional Results	124
B.4.1	Effect of Weight Decay	131
B.4.2	Effect of Batch Size	131
B.4.3	Effect of Increasing Computational Budget on Distillation	131

B.1 Motivation: Cost and Privacy Reasons for Restricting Memory are Flawed

Prior art on continual learning [Lopez-Paz and Ranzato, 2017, Rebuffi et al., 2017b, Chaudhry et al., 2019a, Buzzega et al., 2020, Aljundi et al., 2019b, Lin et al., 2021] motivate the problem from the aspect of prohibited access to previously received data; except for a small portion that is allowed to store in memory. The two principal motivations behind restricting the access to past samples in the literature are two folds. **(i)** Storage space is expensive. **(ii)** Access to previous data is prohibitive due to privacy and GDPR constraints.

Cost. As for the first argument used as a motivation for limiting the memory size, as we have elaborated in Section 1 of the main paper and similarly further detail in Table B.2.1, the cost of storing data is insignificant. This is particularly the case when considering the associated computational costs of training deep models. For example, as per Table B.2.1, it costs 2 cents to store the entirety of the CLEAR dataset, among the largest datasets for continual learning, while it costs about 100\$ to train a model continually on the same dataset. If reducing costs are the key issue, limited computational budget and not memory, as argued earlier, is the way forward.

Privacy. A classical argument is that due to GDPR requirements, data needs to be removed or company privacy policies, it can no longer accessible after certain period, typically a few months. Any previous benchmark with memory constraints already violates this privacy consideration. This is since, which data is to be made private and shall be deleted should not be up to the learning algorithm to decide. But, we further argue that simply not storing any samples still does not address privacy considerations. Deep models store information about trained samples [Jagielski et al., 2020, 2022]. A glaring violation additionally was presented in Haim *et al.* [Haim et al., 2022], showing that we can reconstruct a large number of training samples only given the trained model. Without removing the information explicitly from the deep model, allowing no access to past samples does not alleviate privacy concerns. Goel *et al.* [Goel et al., 2023] presents a catastrophic forgetting baseline for inexact machine unlearning, indicating that forgetting might be the very objective of sustaining privacy, antithetical to the objective of continual learning.

Table B.2.1: Cost of Memory vs Computation. Google Cloud Standard Storage (2¢per GB per month for 1 month) and Compute Cost measured as running cost of an A2 instance (3\$ per hour for 1 GPU). Number of iterations (forward+backward passes) for training a CL model on that dataset listed for comparison invariant to input image and model sizes. We observe that computational costs for running an experiment far outweigh the costs for storing replay samples.

Ref.	Dataset	Ordering	Memory	Cost	Iters	Cost
[Buzzega et al., 2020]	CIFAR10	Cls Inc	1-25MB	0.05¢	250K-375K	20\$
[Rebuffi et al., 2017b]	CIFAR100	Cls Inc	10 MB	0.02¢	50K	8\$
[Douillard et al., 2020, Hou et al., 2019]					125K	15\$
[Buzzega et al., 2020]	TinyImageNet	Cls Inc	5-20 MB	0.04¢	350K-500K	25\$
[Hou et al., 2019, Douillard et al., 2020]	ImageNet100	Cls Inc	0.3-1 GB	2¢	100K	50\$
[Hou et al., 2019, Douillard et al., 2020]	ImageNet1K	Cls Inc	33GB	66¢	1M	500\$
[Lin et al., 2021]	CLEAR	Dist Shift	0.4-1.2GB	2¢	300K	100\$
[He et al., 2016]	ResNet50 (bs=256)		22GB			
Ours	CGLM	Dist Shift	90GB	2\$	2K	10\$
	ImageNet2K	ClsInc, DataInc	400GB	10\$	8K	35\$

B.2 Dataset Construction

B.2.1 Constructing Imagenet2K

ImageNet2K train set is constructed using all training images in ImageNet1K dataset [Deng et al., 2009] for 1K classes as an initialization, with selecting an additional 1K non-overlapping classes from ImageNet21K dataset [Deng et al., 2009] to form the ImageNet2K dataset. We illustrate the creation of the test, validation and train sets below:

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

Validation Set: We use ImagenetV2 dataset [Recht et al., 2019] as the validation set from Imagenet1K data. We separate 50 images per class, not used in the test set to create the val set for the new 1K classes. We combine these two sets to create the overall validation set for experiments. The validation set for every timestep consists of classes from this validation set which have been seen so far.

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

B.2.2 Constructing Continual Google Landmarks V2

Continual Google Landmarks V2 (CGLM) consists of 580K samples. To obtain this subset, we start with the train-clean subset of the Google Landmarks V2 available from the Google Landmarks V2 dataset website¹. We apply the following preprocessing steps in order:

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

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

B.3 Estimation of Equivalent Iterations

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

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

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

The computational cost of training for expensive sampling methods is $1/2C$, as the selected sample set is half the size compared to the naive baseline. The computational cost of selecting the samples is $1/2 \times 3 \times 1/3C = 1/2C$ (forward pass requires $1/3^{rd}$ of the total cost, on $3\times$ the

¹<https://github.com/cvdfoundation/google-landmark>

²<https://www.lesswrong.com/posts/jJApGWG95495pYM7C/how-to-measure-flop-s-for-neural-networks-empirically>

required data, the required data size being $1/2$ when compared to naive). The combined cost is the sum of selection and training cost, which is C . Hence, expensive sampling methods have an equivalent budget with $1/2$ training iterations.

B.4 Additional Results

Due to limited space, some of the experiments in the manuscripts were deferred to the Appendix. In this section we present results for all mentioned costly sampling methods and distillation methods. Additional results for *Section 4.2: 1 “Do Sampling Strategies Matter?”* are presented in Figure B.4.1 where all three costly sampling methods are presented, namely Max Loss, Uncertainty Loss, and KMeans. Similarly, additional results for *Section 4.2: 2 “Does Distillation Matter?”* are presented in Figure B.4.2 where all four distillation methods are presented, namely BCE, MSE, Cosine, and CrossEntropy. We observe that all previous conclusions consistently hold, *i.e.* the Naive baseline is still leading in comparison to all previous approaches.

We also extend the time steps and the number of iterations sensitivity experiments to all four considered distillation methods in all three setups, DI-ImageNet2K, CI-ImageNet2K and CGLM. We present additional results for *Section 4.3: 1. Does the Number of Time Steps Matter?* with results for DI-ImageNet2K presented in Figures B.4.3 and B.4.4 with 50 and 200 time steps respectively, CI-ImageNet2K in Figure B.4.5 and B.4.6 with 50 and 200 time steps respectively and CGLM in Figures B.4.7 and B.4.8 with 50 and 200 time steps respectively. We observe that all previous conclusions consistently hold, *i.e.* the conclusions are robust to changing time steps for a given cost C .

We present additional results for *Section 4.3: 2. Does the Compute Budget Matter?* on DI-ImageNet2K in Figures B.4.9 and B.4.10 for 100 and 1200 iterations respectively, and CI-ImageNet2K in Figures B.4.11 and B.4.12 for 100 and 1200 iterations respectively and on CGLM in Figures B.4.13 and B.4.14 for 40 and 400 iterations respectively. We observe that all previous conclusions consistently hold, *i.e.* the conclusions are robust to changing computational cost, towards both harsher and laxer computational constraint regimes.

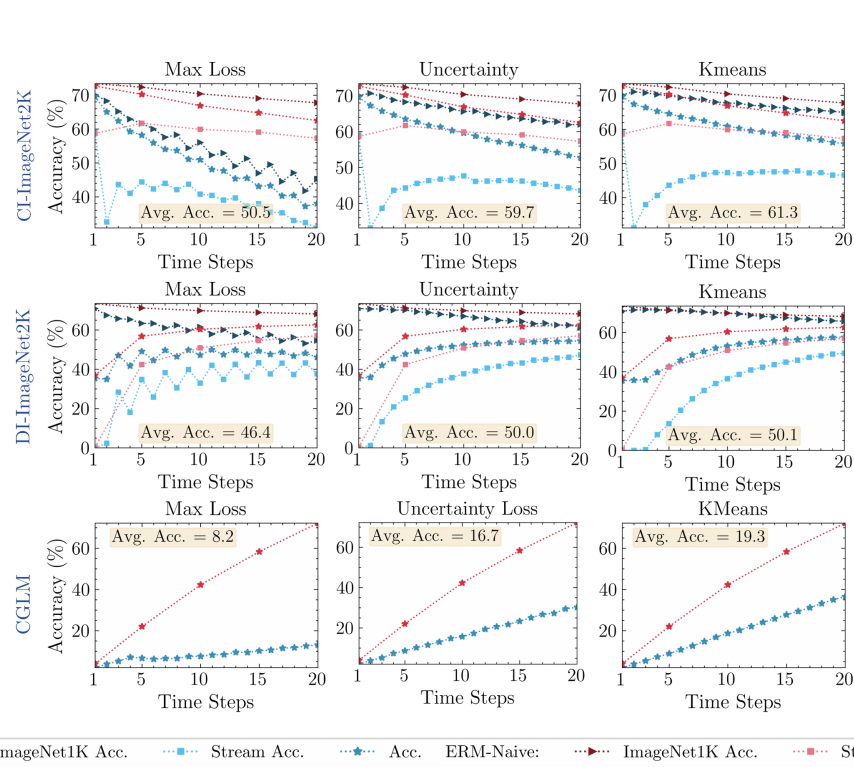


Figure B.4.1: Expensive Sampling. As mentioned in the manuscript, KMeans performs the best among expensive sampling techniques such as Max Loss and Uncertainty Loss. Nevertheless, the performance of the expensive sampling methods is worse than simple Naive.

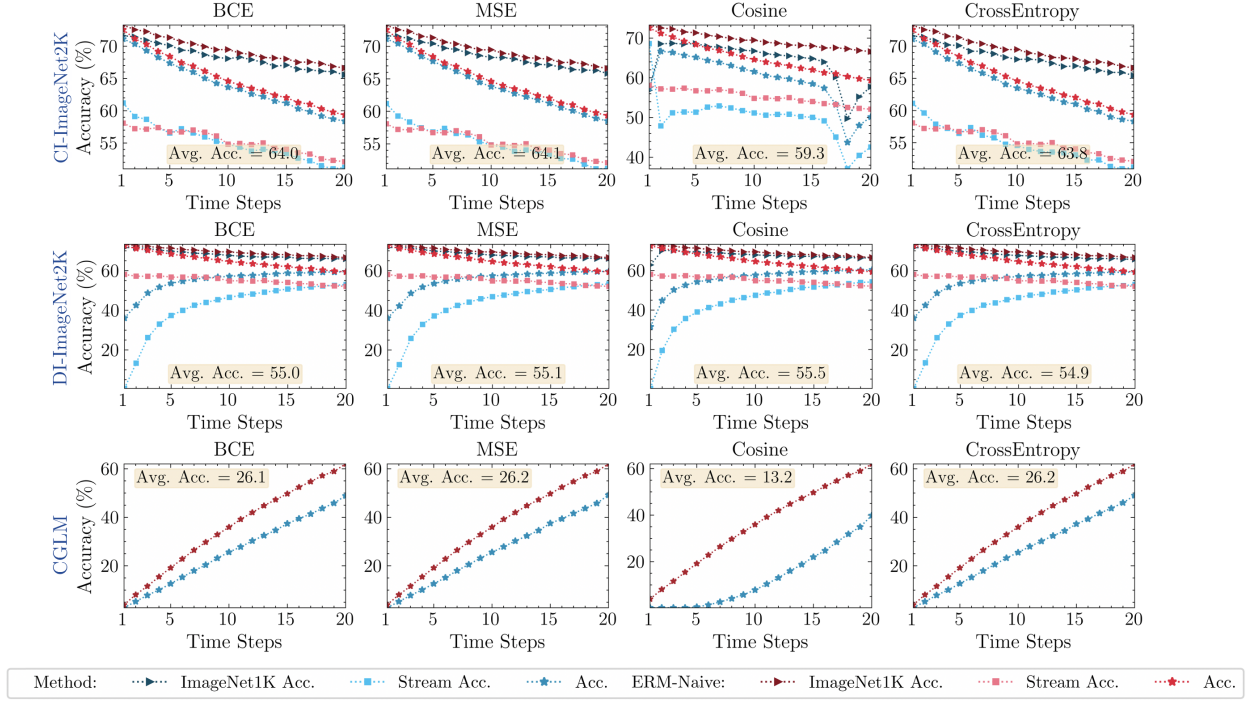


Figure B.4.2: Distillation Methods. All four studied distillation methods under perform compared to the simple Naive baseline in all three studied settings. ImageNet experiments are allowed 400 iterations whereas CGLM is allowed 100 iterations.

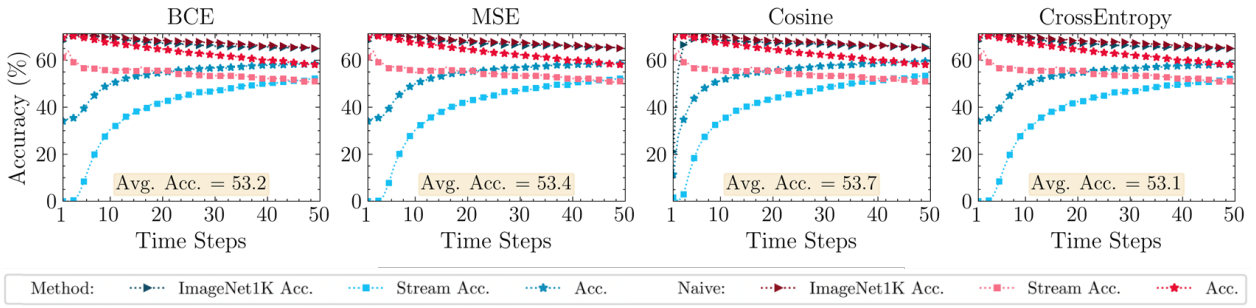


Figure B.4.3: DI-ImageNet2K 50 Time Steps. As observed in the manuscript, when the number of time steps, distillation methods still under perform compared to the Naive baseline.

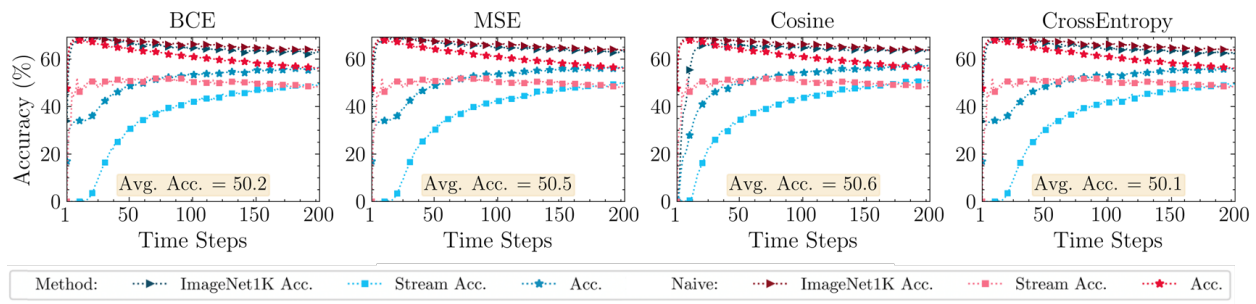


Figure B.4.4: DI-ImageNet2K 200 Time Steps. As observed in the manuscript, when the number of time steps, distillation methods still under perform compared to the Naive baseline.

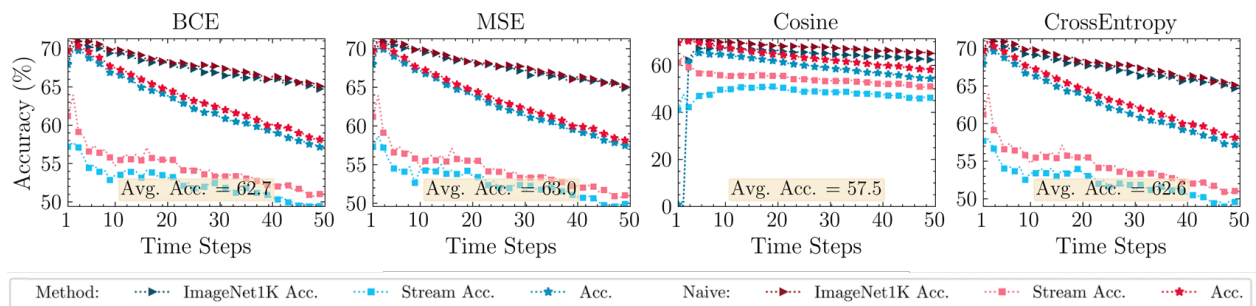


Figure B.4.5: CI-ImageNet2K 50 Time Steps. As observed in the manuscript, when the number of time steps, distillation methods still under perform compared to the Naive baseline.

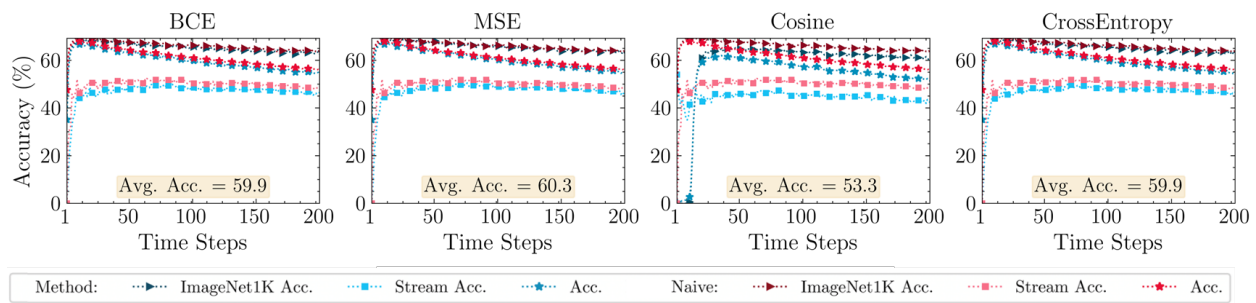


Figure B.4.6: CI-ImageNet2K 200 Time Steps. As observed in the manuscript, when the number of time steps, distillation methods still under perform compared to the Naive baseline.

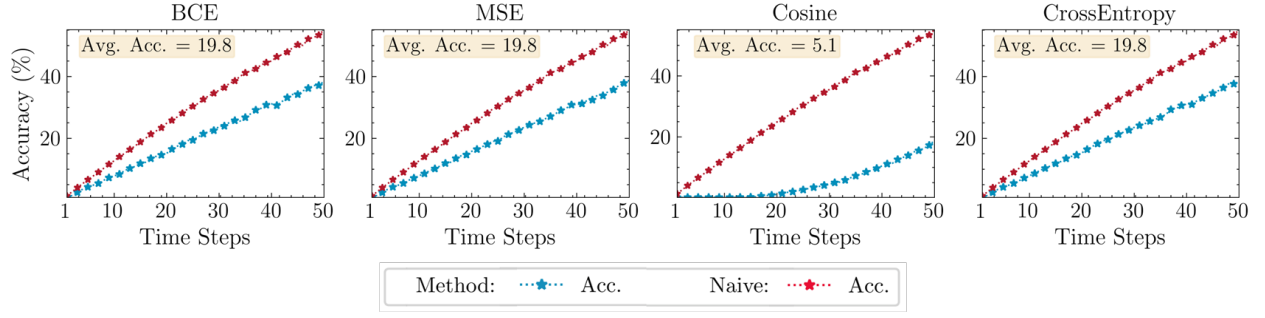


Figure B.4.7: CLGM 50 Time Steps. As observed in the manuscript, when the number of time steps, distillation methods still under perform compared to the Naive baseline.

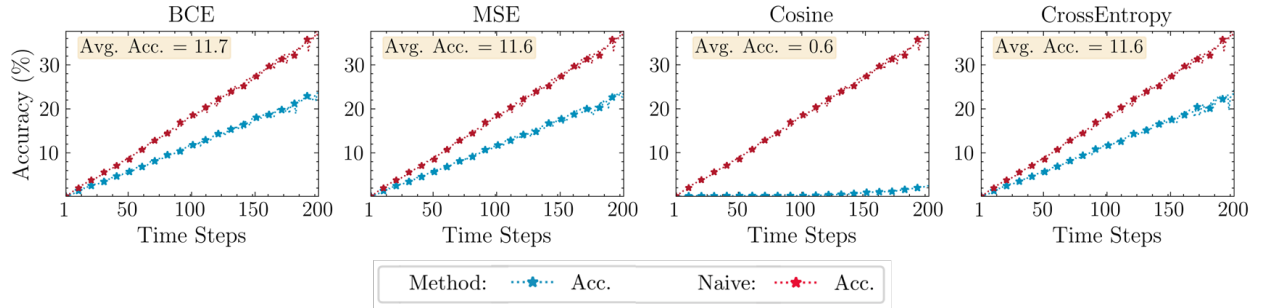


Figure B.4.8: CGLM 200 Time Steps. As observed in the manuscript, when the number of time steps, distillation methods still under perform compared to the Naive baseline.

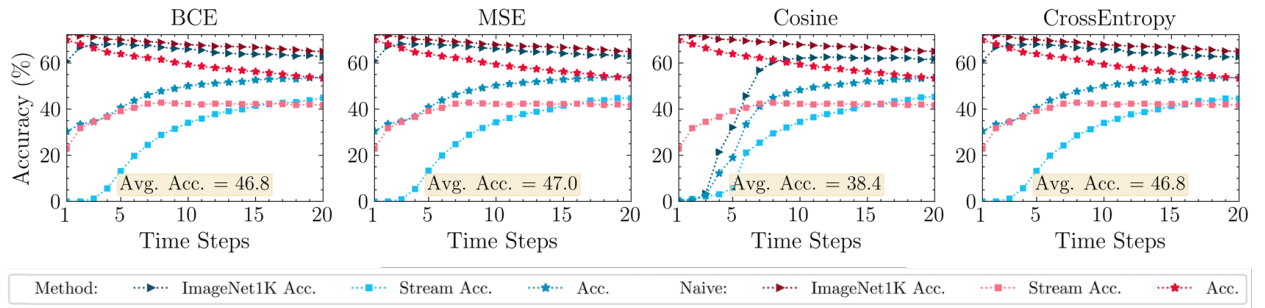


Figure B.4.9: DI-ImageNet2K - 100 Iterations. As observed in the manuscript, with reduced compute, distillation methods still under perform compared to the Naive baseline. The compute budget of the Naive baseline, C , is set to 100 iterations whereas that of the distillation methods is $\frac{2}{3} C = 67$ iterations.

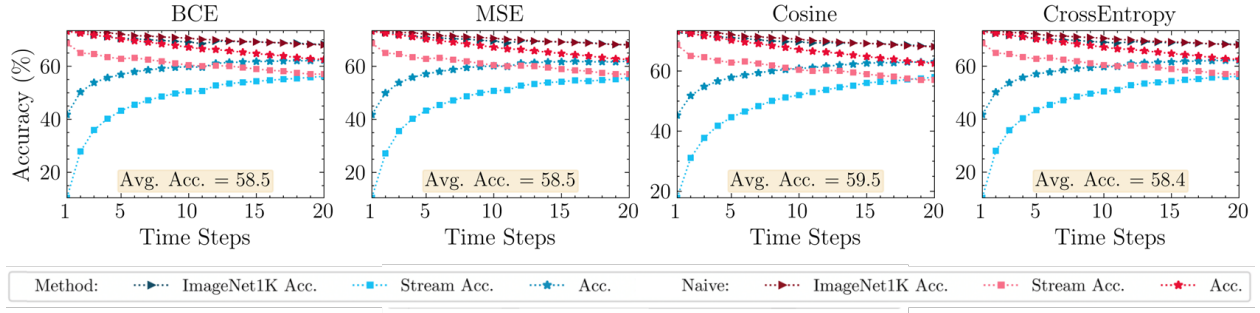


Figure B.4.10: DI-ImageNet2K - 1200 Iterations. As observed in the manuscript, with increased compute, distillation methods still under perform compared to the Naive baseline. The compute budget of the Naive baseline, C , is set to 1200 iterations whereas that of the distillation methods is $2/3 C = 800$ iterations.

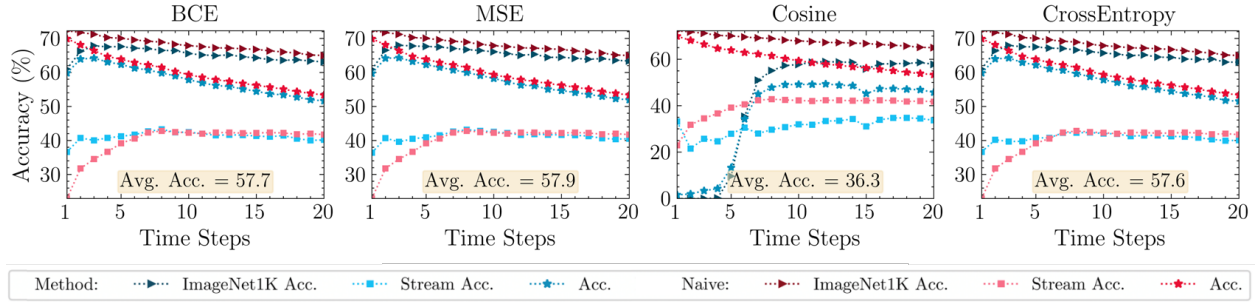


Figure B.4.11: CI-ImageNet2K - 100 Iterations. As observed in the manuscript, with reduced compute, distillation methods still under perform compared to the Naive baseline. The compute budget of the Naive baseline, C , is set to 100 iterations whereas that of the distillation methods is $2/3 C = 67$ iterations.

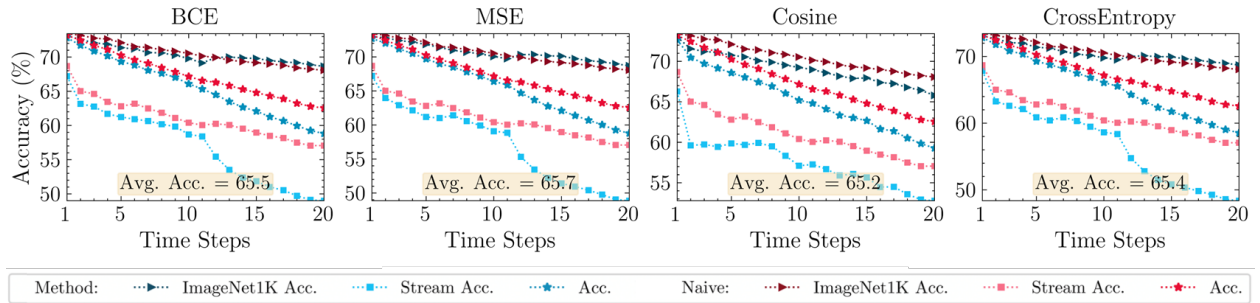


Figure B.4.12: CI-ImageNet2K - 1200 Iterations. As observed in the manuscript, with increased compute, distillation methods still under perform compared to the Naive baseline. The compute budget of the Naive baseline, C , is set to 1200 iterations whereas that of the distillation methods is $2/3 C = 800$ iterations.

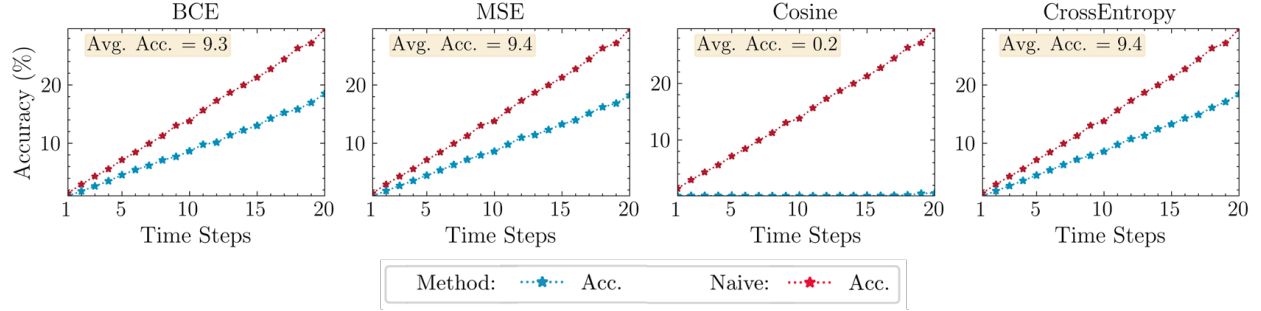


Figure B.4.13: CGLM 40 - Iterations. As observed in the manuscript, with reduced compute, distillation methods still under perform compared to the Naive baseline. The compute budget of the Naive baseline, C , is set to 40 iterations whereas that of the distillation methods is $\frac{2}{3} C = 27$ iterations.

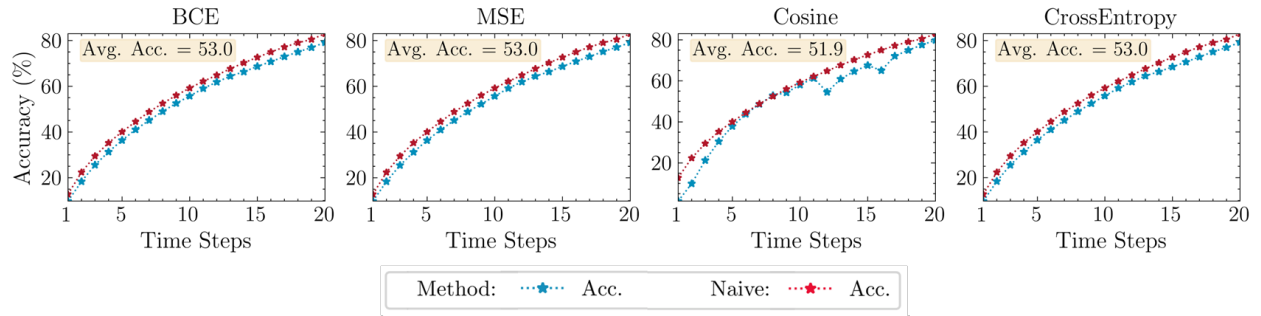


Figure B.4.14: CGLM 400 - Iterations. As observed in the manuscript, with increased compute, distillation methods still under perform compared to the Naive baseline. The compute budget of the Naive baseline, C , is set to 400 iterations whereas that of the distillation methods is $\frac{2}{3} C = 267$ iterations.

B.4.1 Effect of Weight Decay

The choice of weight decay, $wd = 0$, in the manuscript was based on result of cross-validation from the set. More specifically, we try weight decays of $\{5 \times 10^{-5}, 1 \times 10^{-4}\}$. We observe a minor difference in performance between various weight decays, with a $wd=0$ consistently being slightly better. The results are shown in Figure B.4.15

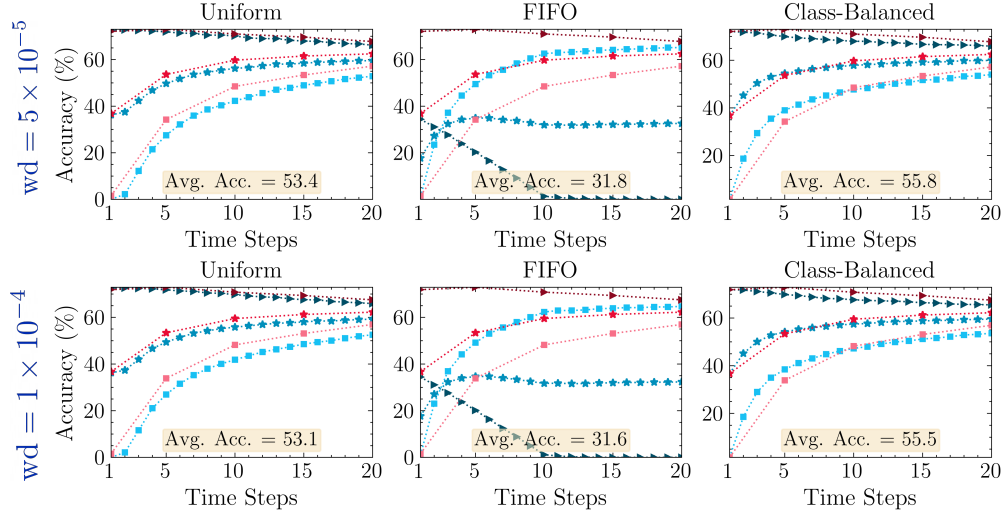


Figure B.4.15: Effect of Weight Decay. Increasing the weight decay causes a slight drop in performance. Setting $wd=0$ gave the best results during our parameter cross-validation.

B.4.2 Effect of Batch Size

In all experiments, we fixed the batch size (BS) to 1500 to optimize the utilization of our hardware resources and minimize the training time. As shown in the literature, BS and learning rate (LR) are closely related. The selected LR was tuned to fit the selected BS. Regardless, we present varying BS experiments in Fig (1) where we study the latest FC correction method, ACE, and the distillation method, MSE, for BS of 250 and 500 with increased iterations of 600 and 300, respectively, and crossvalidated learning rates. We observe that the Naive baseline is still superior even when the batch size is adjusted. Our findings are summarized in Figure B.4.16.

B.4.3 Effect of Increasing Computational Budget on Distillation

We complement the results in Figure 4.4.8 with additional experiments using 800 and 1200 iterations. The observed results align with our previous observations; as long as the computation is normalized across methods, naive, the most simplest among the considered methods, outperforms existing

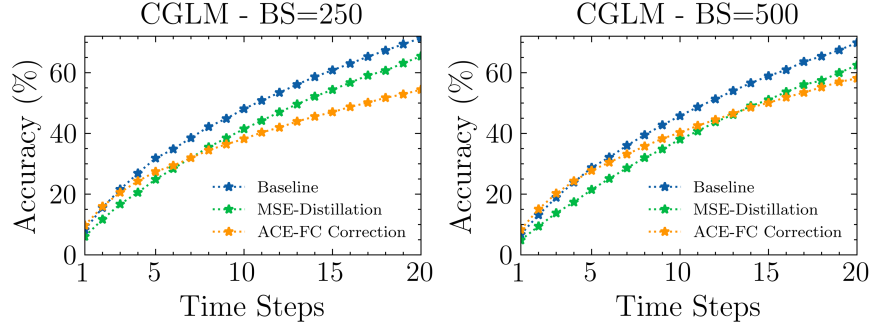


Figure B.4.16: Effect of Batch Size. The conclusions presented in our work hold even when the batch size while is changed while the same overall computational budget.

methods. This is as opposed to prior art comparison that does not normalize compute, which puts the Naive baseline in disadvantage. The results are shown in Figure B.4.17.

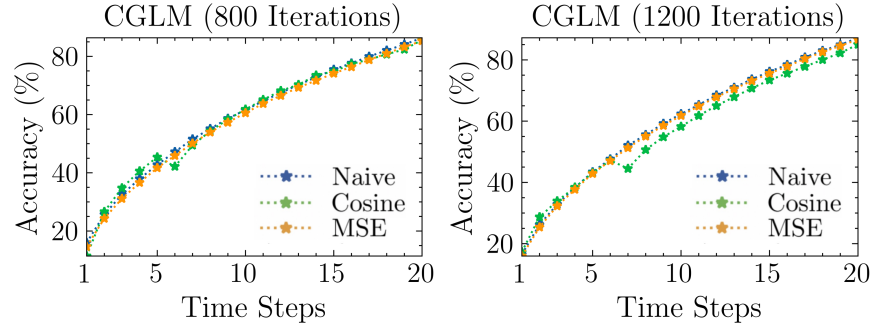


Figure B.4.17: Effect of Increasing Computation Budget on Distillation. As the computation budget is increased while maintaining a normalized compute among different methods, Naive baseline still outperforms distillation based methods.



Online Continual Learning Without the Storage Constraint

In the appendix, we provide additional information and experiments to supplement the observations in the main paper. The appendix is organized as follows:

Contents

C.1	Implementation Details	134
C.1.1	Dataset Details	134
C.1.2	Model and Optimization Details	135
C.2	Evaluating Rapid Adaptation Using Near-Future Accuracy	136
C.3	Additional Experiments	137

C.1 Implementation Details

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

C.1.1 Dataset Details

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

We start with the train-clean subset of the GLDv2 available from the dataset website¹. We apply the following preprocessing steps in order:

1. Filter out images which do not have timestamp metadata available.
2. Remove images of classes that have less than 25 samples in total
3. Order data by timestamp

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

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

Continual YFCC-100M (CYFCC) [Thomee et al., 2016]: The subset of YFCC100M, which has date and time annotations [Cai et al., 2021]. We follow their dataset splits. We order the images by timestep and iterate over 39 million online timesteps, one image at a time, with evaluation on the next image in the stream. Note that in contrast, Cai et al. [2021] uses a more restricted protocol assuming 256 images per timestep and evaluates on images uploaded by a different user in the next batch of samples. We download the images and the metadata as given by Cai et al. [2021] from their github repository. We provide a guide for downloading alongside in our codebase.

¹<https://github.com/cvdfoundation/google-landmark>

C.1.2 Model and Optimization Details

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

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

C.2 Evaluating Rapid Adaptation Using Near-Future Accuracy

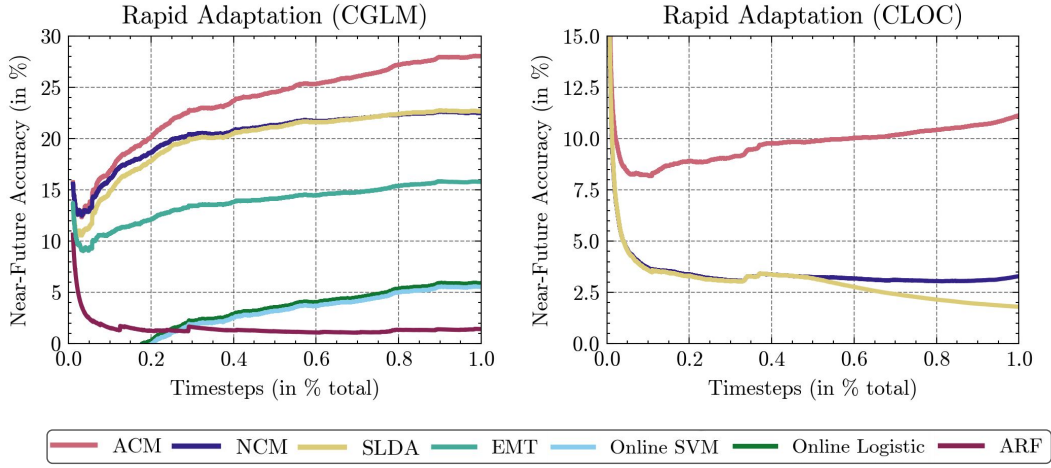


Figure C.2.1: Online Continual Learning Performance. We observe that ACM outperforms existing methods on the near-future accuracy metric by 5-10%, maintaining state-of-the-art performance.

Evaluation of Rapid Adaptation using Near-Future Accuracy. Hammoud et al. [2023] discovered label correlations in the original online accuracy metric used for evaluating online continual learning methods. However, ACM remained state-of-the-art performance on their *fast stream* setting, despite performance of other methods falling below offline learning baselines.

In this section, we compare the performance of ACM to other online learning approaches with a fixed feature extractor on CGLM and CLOC datasets with the near-future accuracy metric using the same delay parameters [Hammoud et al., 2023]. We present our results in Figure C.2.1. We observe that ACM achieves state-of-the-art performance among online continual learning approaches, outperforming them by 5-10% margins. This provides us with additional evidence to indicate that ACM did not exploit on label correlations to achieve high online accuracy.

C.3 Additional Experiments

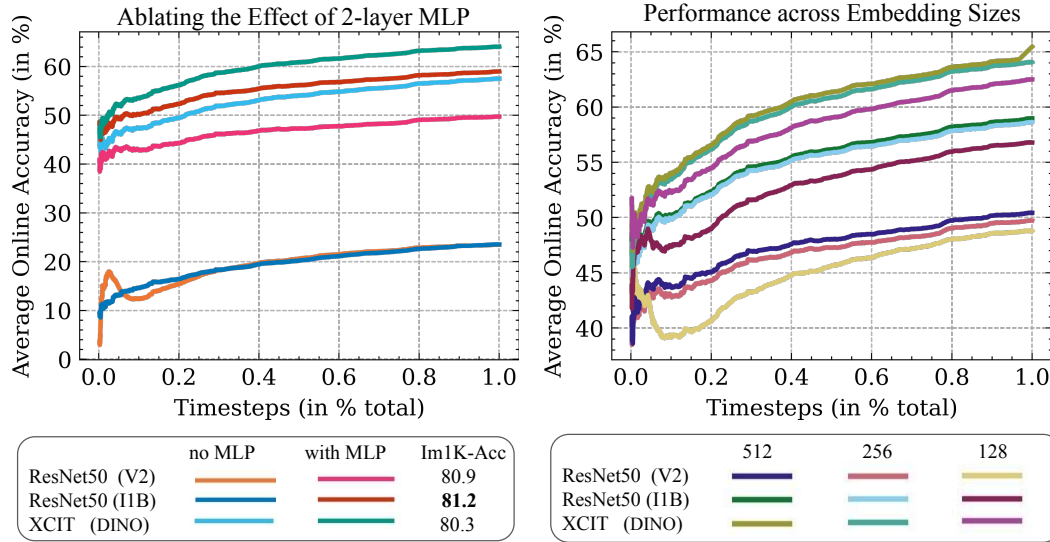


Figure C.3.1: Left: Ablating the effect of first-session adaptation using an MLP. We observed first-session adaptation caused minor improvement in performance on XCIT model with 512 dimensional features, however provided a major performance boost for 2048 dimensional ResNet50 features. **Right: Ablating the effect of embedding dimensions.** We observe that varying the embedding size allows a tradeoff between computational cost and accuracy. However, it does not cause dramatic increase or decrease in performance.

Ablating the contribution of the 2-layer MLP. We ablate the contribution of first-session adaptation [Panos et al., 2023] on the pretraining set. We compare online learning performance with and without using the MLP and present the results across different models in Figure C.3.1 (left). We observe that the performance in XCIT DINO model achieves a performance boost of 5% when using the MLP, indicating that first-session adaptation allows for an improvement in performance. However, both ResNet50 models gain large improvements of over 30% in online accuracy due to the MLP. We notice that this is attributable to the curse of dimensionality. ResNet50 architecture has a far larger feature dimension of 2048 in compared to XCIT-DINO having 512 dimensions. Comparing models with MLP, we surprisingly observe that ResNet50-I1B performs worse than XCIT-DINO model despite ResNet50-I1B achieving better performance across traditional benchmarks, indicative of robust and generalizable features. We conclude that ResNet50 architecture is a poor fit for ACM due to 2048 dimensional features.

Conclusion. Models with high-dimensional embeddings perform much more poorly in combination with a kNN despite better representational power due to the curse of dimensionality.

Ablating the effect of the embedding size. We now know that embedding dimensions can significantly affect performance. We vary the embedding dimensions to explore the sensitivity

of ACM to embedding dimensions. We start with 512 dimensional XCIT feature dimensions, varying the embedding sizes to 512, 256 and 128.

We present the results in Figure C.3.1 (right). First, we observe that decreasing the embedding dimension to 256 results in minimal drop in accuracy across all the three models but reduces the computational costs by half as shown in Figure 4 in the main paper. Further reduction in embedding dimension leads to considerable loss of performance. An embedding size of 256 achieves the best tradeoff between speed and accuracy.



Rapid Adaptation in Online Continual Learning: Are We Evaluating It Right?

In the appendix, we provide additional information and experiments to supplement the observations in the main paper. The appendix is organized as follows:

Contents

D.1	Effect of Sampling Strategies	140
D.1.1	CGLM	140
D.1.2	CLOC	140
D.2	Sensitivity Analysis: Learning Rate and Weight Decay	141
D.2.1	Sensitivity to Weight Decay	142
D.2.2	Sensitivity to Learning Rate	142
D.3	Fixed-Feature Extractor Based Methods: NCM, SDLA, and ACM	142
D.4	Limitations & Future Directions	143

D.1 Effect of Sampling Strategies

In Figure 4 of the manuscript, we present the baseline ER for two sampling strategies, namely FIFO and uniform. FIFO selects the latest seen samples to train on, whereas Uniform simply randomly and uniformly samples a set of previously seen samples to train on. Mixed sampling is a mix of both FIFO and Uniform, where half of the batch is constructed using FIFO sampling, and the other half using Uniform sampling.

D.1.1 CGLM

The results for various samplers for both Online Accuracy (Figure D.1.1) and Near-Future Accuracy (Figure D.1.2) on CGLM dataset are summarized below:

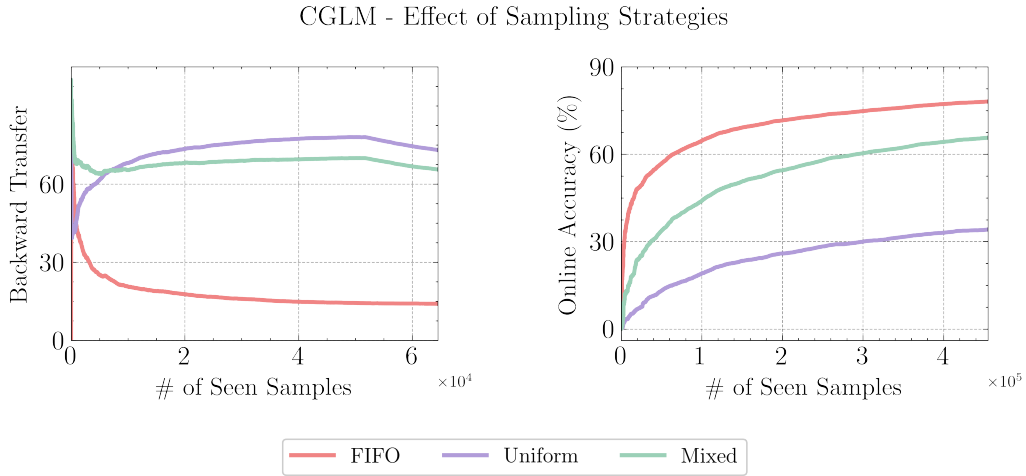


Figure D.1.1: Effect of Sampling Strategy on Online Accuracy. In terms of online accuracy, FIFO sampling which focuses on the latest samples performs best where uniform performs the worst and mixed performs somewhere in the middle. In terms of retention however, the order is reversed.

Conclusion. Interestingly, mixed sampling is competitive with uniform sampling in near future accuracy, but performs worse in information retention.

D.1.2 CLOC

The results for various samplers for both Online Accuracy (Figure D.1.3) and near-future accuracy (Figure D.1.4) on CLOC dataset are summarized below.

Conclusion. Mixed sampling is competitive with uniform sampling in near future accuracy, however, its information retention capabilities are half that of the ER baseline.

CGLM - Effect of Sampling Strategies

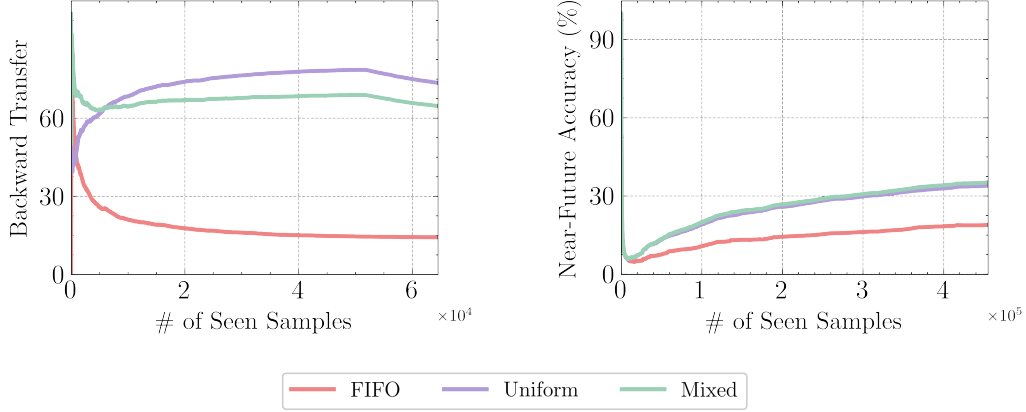


Figure D.1.2: Effect of Sampling Strategy on Near-Future Accuracy. In terms of near-future accuracy, Uniform and Mixed sampling perform almost the same, however FIFO sampling is no where close to them. In terms of retention, Uniform still takes the lead.

CLOC - Effect of Sampling Strategies

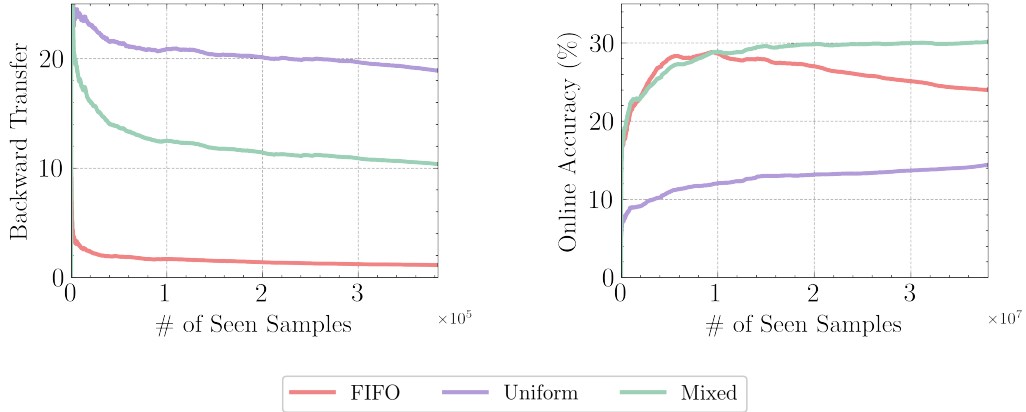


Figure D.1.3: Effect of Sampling Strategy on Online Accuracy. Unlike what was observed for CLGM, for CLOC, the mixed sampling performs the best in terms of online accuracy followed followed by FIFO and then Uniform. However, Uniform still performs the best in terms of information retention with a large gap with a large margin.

D.2 Sensitivity Analysis: Learning Rate and Weight Decay

In OCL, changing hyperparameters across datasets is uncertain as the stream might have a significant distribution shift from the pretrain. Hence, we use the hyperparameters for our model from Ghunaim *et al.*[Ghunaim et al., 2023] for all our experiments. However, how do the selected hyperparameters

CLOC - Effect of Sampling Strategies

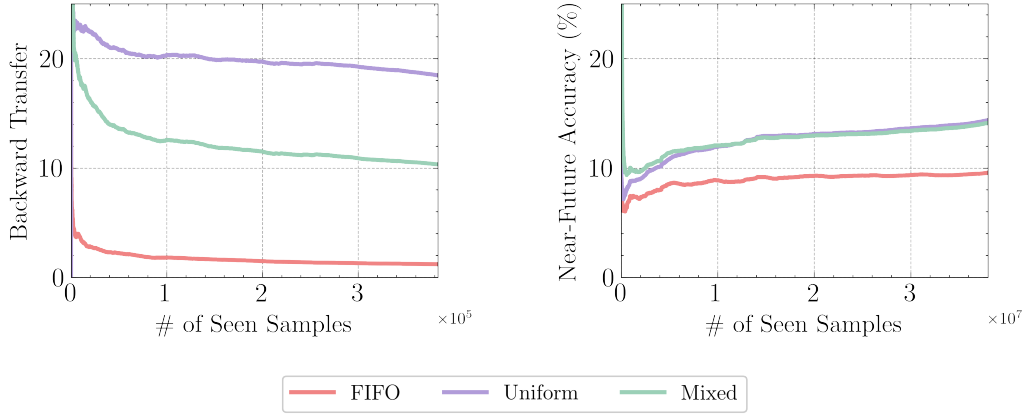


Figure D.1.4: Effect of Sampling Strategy on Near-Future Accuracy In terms of near-future accuracy, mixed outperforms other samplers by a significant margin. However, it achieves significantly worse performance compared to uniform sampling in terms of information retention.

transfer to CGLM is an interesting question. We demonstrate the sensitivity of the hyperparameters: learning rate and weight decay below:

D.2.1 Sensitivity to Weight Decay

Results and Conclusion. We present our results in Figure D.2.1. We conclude that weight decay has minimal effect on the performance of the ER (Replay Only) method.

D.2.2 Sensitivity to Learning Rate

Results and Conclusion. We present our results in Figure D.2.2. An order magnitude change in learning rate in either direction leads to a decrease in both information retention performance and near-future accuracy. The learning rate of 0.005 transfers well to CGLM dataset in terms of both near-future accuracy and information retention (Backward Transfer).

D.3 Fixed-Feature Extractor Based Methods: NCM, SDLA, and ACM

In this section, we compare ACM with other popular fixed-feature extractor based methods like NCM mensink2013distance, ostapenko2022continual, janson2022simple and SLDA [Hayes and

CGLM - Effect of Weight Decay

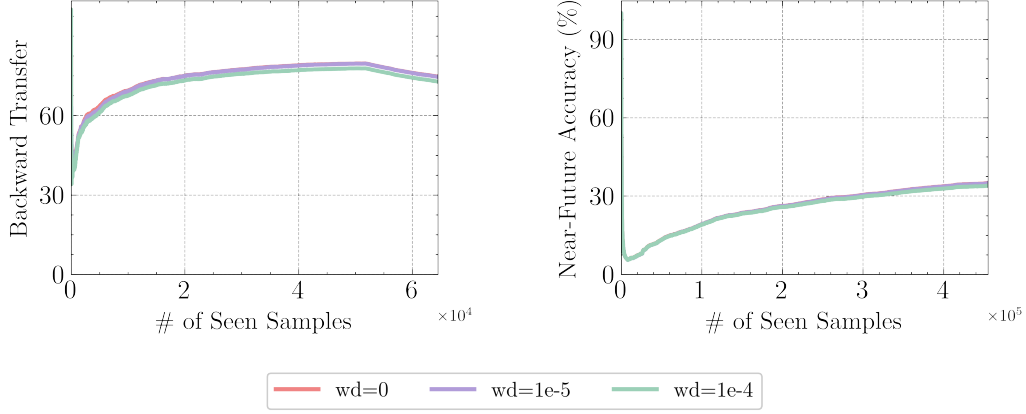


Figure D.2.1: Sensitivity to Weight Decay on ER (Replay Only). Weight decay seems to have minimal effect on both near-future accuracy and backward transfer.

CGLM - Effect of Learning Rate

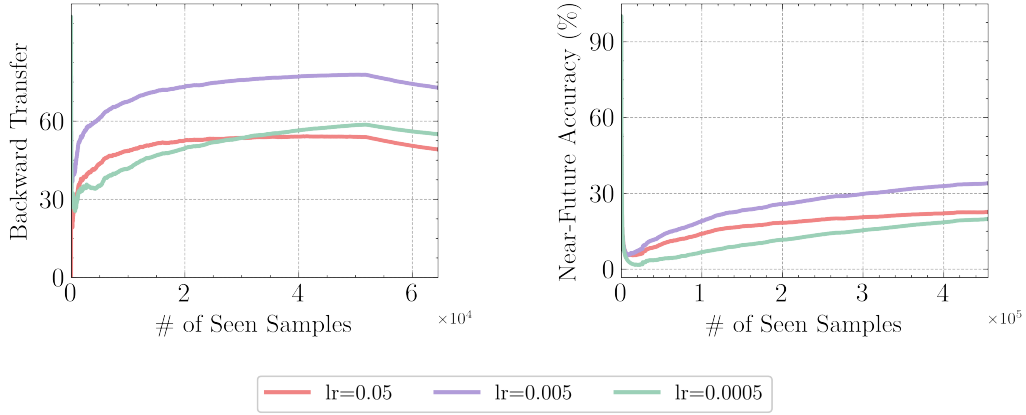


Figure D.2.2: Effect of Learning Rate on the ER Baseline on CGLM. Both near future-accuracy and backward transfer are highly sensitive to the selection of the learning rate.

Kanan, 2020]. The results for running NCM, SLDA, and ACM on CGLM dataset are shown in Figure D.3.1 where we see that ACM performs best in terms of backward transfer, online accuracy, and near-future accuracy.

D.4 Limitations & Future Directions

Limitations. While our work discovers interesting phenomena in OCL, it has important limitations:

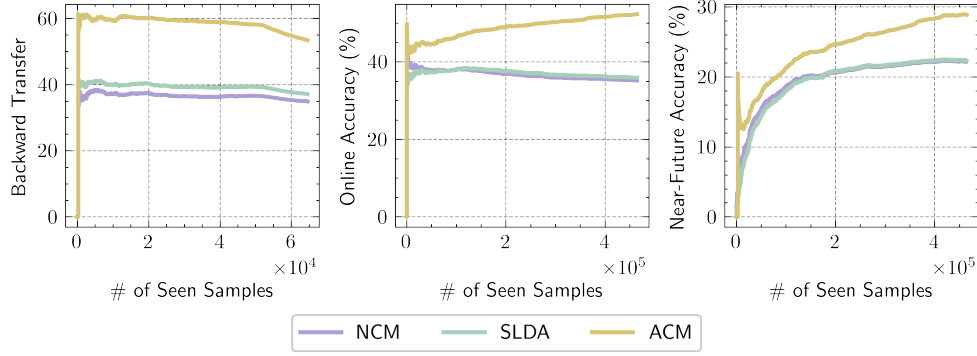


Figure D.3.1: Fixed-Feature Extractor Based Methods. When compared to NCM and SLDA, ACM performs the best in terms of Backward Transfer, Online Accuracy and Near-Future Accuracy metrics.

- **Is near-future accuracy definitively measuring adaptation?** It is unclear why our proposed evaluation approach definitively measures rapid adaptation, and further investigation is needed to determine the problem exists in future OCL scenarios.
- **Dependency on shift S .** Our proposed metric, near-future accuracy, depends on the calculated value (fixed) for the shift, S , however in reality the stream label correlations could be dynamically changing with the stream and an adaptive value of S would be required in that case.
- **Mitigating Monitoring Costs.** In our experiments, we did not explicitly store the model itself as we were able to access future samples solely for evaluation purposes. In practice, a larger memory allocation would be necessary. To reduce storage costs we recommend performing this evaluation periodically instead of on every incoming sample to reduce storage requirements.
- **Why use a new metric instead of changing the data stream?** A question that might arise is why use a new metric if we could simply remove the correlated samples? The data streams used in our work are *naturally* ordered by timestamps, therefore changing the datasets would make it less natural or even complete unnatural if the correlation is long enough. Therefore, in our work we propose a new metric which could address the label correlations without modifying the datasets.

E

From Categories to Classifier: Name-Only Continual Learning by Exploring the Web

In the appendix, we provide additional information and experiments to supplement the observations in the main paper. The appendix is organized as follows:

Contents

E.1	Connections to Past Literature in Webly Supervised Learning	146
E.2	Appendix: How Reliable is Uncurated Webly-supervised Data?	148
E.2.1	Analysis: Why Does Uncurated Web Data Outperform Manually Annotated Datasets?	148
E.3	Appendix: Comparison with Name-Only Classification Strategies	152
E.4	Appendix: Continual Webly Supervised Learning	155
E.4.1	EvoTrends	155
E.4.2	Computational Budgets	155
E.4.3	Investigating Performance Gaps Between Our Webly-Supervised Data and MA Data	158
E.5	Ethics	162

E.1 Connections to Past Literature in Webly Supervised Learning

Webly-supervised learning has been extensively explored in the past two decades. We summarize a few directions in this section.

Weakly/Noisy-Label Supervised Learning: Early works attempt to leverage alternative forms of input signals, such as predicting words [Joulin et al., 2016] or n-grams [Li et al., 2017a]. Subsequent work [Mahajan et al., 2018, Singh et al., 2022, Beal et al., 2022, Ghadiyaram et al., 2019] in this direction focused on significantly increasing the size of pretraining datasets, often by incorporating hashtags. This expansion of pretraining data was aimed at improving model performance. More recently, there has been a surge in interest in vision-language pretraining [Radford et al., 2021, Jia et al., 2021, Cherti et al., 2023], particularly with alt-text training, which has gained popularity due to its out-of-the-box, zero-shot capabilities. However, it is important to note that these methods often grapple with the issues of poor ground truth quality leading to a noisy training signal. Interestingly, they can be matched in performance by self-supervised learning methods [Oquab et al., 2023] which do not rely on any annotations and often involve fewer images in their training process.

Augmenting Training Datasets with Internet. An promising alternative is to augment training datasets with web images Zheng et al. [2020]. Recently, [Oquab et al., 2023] obtained highly transferable representations by augmenting training dataset with high quality web images by similarity-search for self-supervised pretraining. Additionally, parallel work [Li et al., 2023b] explored targeted pretraining, making that step highly efficient given a downstream task. However, our setting differs from these works as they focus on pretraining and have supplementary train data available, we focus on the label-only setting with no reference train set in the downstream task.

Name-only Classification. Image search engines traditionally provide highly relevant results, using text, linked images and user query based recommendation engines. Hence, researchers have long-since used internet in the name-only classification setting as a form of supervision [Fergus et al., 2005, Vijayanarasimhan and Grauman, 2008, Schroff et al., 2010]. Several datasets including Webvision [Li et al., 2017b], JFT-300M [Sun et al., 2017], Instagram-1B [Mahajan et al., 2018] have been introduced for pretraining. Several works have specifically targeted fine-grained classification [Krause et al., 2016, Sun et al., 2021] creating very large scale pretraining datasets for fine-grained classification. Recent approaches have primarily focused on open-ended large-vocabulary visual category learning [Kamath et al., 2022]. Past works have diversified this to object detection [Chen and Gupta, 2015] and object segmentation [Shen et al., 2018, Jin et al., 2017, Sun et al., 2020]

with tackling challenges of the weak category supervision. We do not focus on the pretraining task, where self-supervised learning has achieved dominance. Instead, we focus on building targeted representations for downstream classification tasks.

Continual Webly-Supervised Learning. This line of research which attempts to learn datasets continually [Li and Fei-Fei, 2010] is much sparser, with classical works [Divvala et al., 2014, Chen et al., 2013] attempting to learn exhaustively about visual categories. In contrast, we focus on more recent formulation of the continual learning problem with a targeted set of categories, computationally budgeted learning when given class categories.

Active Learning. An interesting avenue of research is actively labeling portions of dataset with humans-in-the-loop allowing high quality annotations within limited time. A primary bottleneck here is the computationally expensiveness in scaling these approaches to large datasets, tackled in [Coleman et al., 2019, Prabhu et al., 2019]. However, their primary quality is they allow learning rare, underspecified concepts [Coleman et al., 2022, Hayes et al., 2022, Stretcu et al., 2023] in a continual fashion [Mundt et al., 2023, Munagala et al., 2022]. However, these methods are complementary to name-only classification approaches which generate the unlabeled pool of images for these approaches to select informative samples from.

E.2 Appendix: How Reliable is Uncurated Webly-supervised Data?

How Big Is Our Dataset? We present the dataset set statistics of both the manually curated and the web data in Table E.2.1. The scale of the uncurated webly-supervised data is significantly bigger than the manually annotated ones. Additionally, cleaning the web data using automated removal of noisy samples and deduplication using DinoV2 ViT-G [Oquab et al., 2023] features could significantly reduce the size of the web data, indicating severe duplication amongst search engines.

Training Dataset	Datasets				
	FGVC Aircraft	Flowers102	OxfordIIITPets	Stanford Cars	BirdSnap
<i>Size Relative to Ground Truth Train Sets</i>					
MA Data	3.3K	1.0K	3.7K	8.1K	42K
C2C (Ours)	158K (48x)	155K (155x)	53K (15x)	184K (23x)	557K (13x)
<i>Ablating Dataset Cleaning</i>					
Before Cleaning	158K	155K	53K	184K	557K
Duplicates	57K	51K	10K	57K	-
Noisy Samples	15K	23K	2.3K	24K	-
After Cleaning	90K (0.57x)	97K (0.62x)	40K (0.75x)	111K (0.60x)	-
<i>Ablating Search Engines</i>					
All	158K	155K	53K	184K	557K
Google Only	30K	29K	11K	60K (1.3x)	142K
Bing Only	29K	30K	8K	47K	104K
DuckDuckGo Only	29K	32K	8K	55K	166K (1.1x)
Flickr Only	70K (2.3x)	65K (2.0x)	26K (2.4x)	21K	145K

Table E.2.1: Dataset Statistics. Our uncurated webly-supervised data is significantly larger in size compared to manually annotated (MA) datasets. Leveraging multiple engines could allow to query the web for samples enabling the creation of datasets up to 155× larger than manually curated ones.

E.2.1 Analysis: Why Does Uncurated Web Data Outperform Manually Annotated Datasets?

We now present a comprehensive analysis of factors that we thoroughly examined but found to have no impact on our performance.

1. Uncurated Webly-Supervised Data on Different Architectures and Training Procedures

To investigate this aspect, we conducted an experiment employing three additional backbones: (a) a supervised ImageNet1K ResNet50, (b) a DeiT-III ViT-B/16 ImageNet21K, and (c) a weakly-supervised CLIP ResNet50 [Radford et al., 2021]. The results of this experiment are summarized in Table E.2.2. Our findings reveal that neither the architecture nor the training procedure is

		Datasets				
Eval.	Dataset	FGVC Aircraft	Flowers102	OxfordIIITPets	Stanford Cars	BirdSnap
ResNet50 (Supervised)						
Linear Probing	MA Data	31.7%	79.9%	92.1%	48.8%	50.4%
	C2C (Ours)	44.2% (+12.5%)	83.5% (+3.6%)	92.9% (+0.8%)	58.1% (+9.3%)	56.2% (+5.8%)
MLP Adapter	MA Data	35.5%	78.8%	92.8%	45.6%	49.2%
	C2C (Ours)	47.3% (+11.8%)	84.5% (+5.7%)	93.8% (+1.0%)	58.0% (+12.4%)	54.5% (+5.3%)
ViT-B/16 (DeiT-III)						
Linear Probing	MA Data	49.0%	98.0%	94.3%	64.5%	73.7%
	C2C (Ours)	63.5% (+14.5%)	92.9% (-5.1%)	94.8% (+0.5%)	71.5% (+7.0%)	73.4% (-0.3%)
MLP Adapter	MA Data	48.9%	98.3%	94.3%	65.7%	72.9%
	C2C (Ours)	67.4% (+18.5%)	93.6% (-4.7%)	95.0% (+0.7%)	73.9% (+8.2%)	75.4% (+2.5%)
ViT-B/32 (CLIP)						
Linear Probing	MA Data	40.5%	91.2%	89.5%	80.8%	56.1%
	C2C (Ours)	51.7% (+11.2%)	86.2% (-5.0%)	91.8% (+2.3%)	79.3% (-1.5%)	58.4% (+2.3%)
MLP Adapter	MA Data	44.8%	88.9%	90.9%	79.9%	55.3%
	C2C (Ours)	57.8% (+13.0%)	86.8% (-2.1%)	92.6% (+1.7%)	81.1% (+1.2%)	57.1% (+1.8%)
ResNet50-CLIP						
Linear Probing	MA Data	31.9%	80.5%	81.8%	69.2%	44.7%
	C2C (Ours)	44.0% (+12.1%)	82.0% (+1.5%)	88.1% (+6.3%)	71.3% (+2.1%)	48.1% (+3.4%)
MLP Adapter	MA Data	34.3%	78.3%	85.7%	69.0%	45.2%
	C2C (Ours)	48.9% (+14.6%)	84.8% (+6.5%)	89.4% (+3.7%)	72.6% (+3.6%)	46.6% (+1.4%)

Table E.2.2: Impact of Architecture and Pretraining Algorithm. Irrespective of the selected network architecture and pretraining algorithm, our uncuration webly-supervised data consistently exhibit superior performance over manually annotated datasets by substantial margins. The improvement is primarily attributed to the scale of web data.

the underlying cause behind the observed performance. Remarkably, our web data consistently outperform manually annotated datasets across various classifiers and training procedures. In conclusion, our study consistently demonstrates comparable or superior performance to manually annotated datasets when considering different classifiers and training procedures.

2. Influence of Specific Search Engines on Accuracy Improvement

To investigate the impact of individual search engines on our overall performance, we compare the performance achieved by each of the four search engines individually, as presented in Table E.2.3. Our analysis reveals that the collection of data from all four search engines, despite encountering challenges such as duplicate search results, consistently results in a 3-10% higher absolute accuracy improvement compared to using a single search engine. It is worth noting that Flickr consistently outperforms the other engines, while Google exhibits the lowest performance. In summary, the performance of our webly-supervised data cannot be solely attributed to a specific search engine. Each search engine contributes a distinct distribution of samples, and the combination of these

Eval.	Training Dataset	Datasets				
		FGVC Aircraft	Flowers102	OxfordIIITPets	Stanford Cars	BirdSnap
Linear Probe	All	57.5%	85.7%	91.7%	62.1%	56.1%
	Google Only	42.0% (-15.5%)	75.7% (-10.0%)	88.9% (-2.8%)	63.2% (+1.1%)	48.9% (-7.2%)
	Bing Only	48.9% (-8.6%)	81.3% (-4.4%)	88.8% (-2.9%)	55.7% (-6.4%)	47.2% (-8.9%)
	DuckDuckGo Only	47.4% (-10.1%)	80.4% (-5.3%)	88.8% (-2.9%)	59.0% (-3.1%)	46.7% (-9.4%)
	Flickr Only	58.9% (+1.4%)	82.5% (-3.2%)	89.9% (-1.8%)	-%	52.8% (-3.3%)
MLP-Adapter	All	65.5%	87.1%	92.8%	66.8%	53.7%
	Google Only	49.2% (-16.3%)	78.3% (-8.8%)	90.3% (-2.5%)	65.6% (-1.2%)	48.9% (-4.8%)
	Bing Only	54.5% (-11.0%)	81.7% (-5.4%)	90.1% (-2.7%)	60.3% (-6.5%)	48.7% (-5.0%)
	DuckDuckGo Only	54.0% (-11.5%)	81.6% (-5.5%)	90.1% (-2.7%)	62.4% (-4.4%)	47.5% (-6.2%)
	Flickr Only	62.8% (-2.7%)	83.0% (-4.1%)	91.7% (-1.1%)	-%	50.7% (-3.0%)

Table E.2.3: Performance Across Search Engines. We find that different search engines demonstrate varying levels of performance across different datasets, and their collective utilization yields superior results compared to any individual search engine. It is worth noting that Flickr encountered difficulties in querying and downloading images for certain classes in the Cars dataset, hence the performance for those classes is not reported.

Evaluation	Training Dataset	Datasets				
		FGVC Aircraft	Flowers102	OxfordIIITPets	Stanford Cars	BirdSnap
Ablating Dataset Cleaning						
Linear Probing	Before Cleaning	57.5%	85.7%	91.7%	62.1%	56.1%
	After Cleaning	57.3% (-0.2%)	83.3% (-2.4%)	91.7%	69.2% (+7.1%)	-
MLP Adapter	Before Cleaning	65.5%	87.1%	92.8%	66.8%	53.7%
	After Cleaning	64.6% (-0.9%)	84.8% (-2.3%)	93.1% (+0.3%)	71.0% (+4.2%)	-
Ablating Balanced Training						
Linear Probing	Class-Balanced	57.5%	85.7%	91.7%	62.1%	56.1%
	Uniform	61.2% (+3.7%)	84.6% (-1.1%)	91.8% (+0.1%)	68.7% (+6.6%)	56.8 %(+0.7%)
MLP Adapter	Class-Balanced	65.5%	87.1%	92.8%	66.8%	53.7%
	Uniform	66.2% (+0.7%)	86.4% (-0.7%)	92.6% (-0.2%)	70.1% (+3.3%)	53.8% (+0.1%)

Table E.2.4: Effect of Architecture and Pretraining Algorithm. We observe different search engines performing well across different datasets, with their combined data consistently outperforming any individual search engine.

distributions leads to enhanced generalization and improved performance.

3. Does Cleaning Webly-supervised Data Help? To clean our data, we perform removal of noisy samples along with deduplication using DinoV2 ViT-G features for reliability. We defined duplicates as samples with a cosine similarity higher than 0.99. We defined noisy samples as samples which have a cosine similarity lower than 0.2 for 50% or more samples in their category. We checked that our thresholds were reasonable by manual inspection of identified duplicate and noisy samples. In table E.2.4, we investigate the impact of cleaning the noisy webly-supervised data. We find that cleaning our data with did not lead to substantial performance improvements,

indicating that the main challenge may be the out-of-domain nature of web data rather than the inherent noise in the queried samples.

4. Does Class Balancing Help? In Table E.2.4 we investigate the importance of class balanced sampling in our setup. We find that class-balanced sampling, did not impact performance much and only lead to drops in performance in a few cases. This suggests that further improvements in this direction, such as using long-tailed loss functions, might not yield significant improvements. Our investigation suggests that categories for which we can only collect a limited number of samples from the web are typically associated with noisy samples. Consequently, these categories do not contribute significantly to performance improvement.

E.3 Appendix: Comparison with Name-Only Classification Strategies

We provide detailed descriptions of the compared approaches below:

1. **CALIP** [Guo et al., 2023]: This approach aims to enhance the alignment between textual and visual features for improved similarity score estimation. It achieves this by utilizing textual class-prompts to generate attention maps that correlate better with the input images.
2. **CLIP-DN** [Zhou et al., 2023b]: Similar to CALIP, CLIP-DN focuses on enhancing the alignment between textual and visual features. It achieves this through test-time adaptation by estimating input normalization.
3. **CuPL** [Pratt et al., 2023]: CuPL takes a different approach by improving prompts at a class-wise granularity across datasets. Leveraging GPT3, it generates enhanced prompts for each class.
4. **VisDesc** [Menon and Vondrick, 2022]: VisDesc improves prompts by computing similarity with multiple textual descriptors that capture the visual characteristics of the target class.
5. **Glide-Syn** [He et al., 2022]: Glide-Syn generates realistic synthetic images using the ALIGN model. It then uses classifier tuning, as proposed in [Wortsman et al., 2022], to classify these synthetic images.
6. **SuS-X-LC and SuS-X-SD** [Udandaraao et al., 2023]: SuS-LC and SuS-SD experiment with different techniques. SuS-LC retrieves images similar to the class-prompt from the LAION-5B dataset using a ViT-L model. SuS-SD, on the other hand, generates images based on a class-description using a stable diffusion model. These images are then utilized in combination with an adapter module called Tip-X for inference.
7. **SD-Classifier** [Li et al., 2023a]: SD-Classifier employs a diffusion-classifier framework to fine-tune the text conditioning $\mathbf{c}$ for each class. It predicts the noise added to the input image that maximizes the likelihood of the true label.
8. **CaFo** [Zhang et al., 2023b]: CaFo leverage GPT-3 to produce textual inputs for CLIP and for querying a DALL-E model in the non-zs variant. The DALL-E model then generates a classification dataset with 16 samples per class, using a combination of CLIP and DINO features.

9. **Retrieval+SSL** [Li et al., 2023b, Wallingford et al., 2023]: This implements a self-supervised finetuning step with the base backbone before classification with label information. However, this is a 1-shot internet explorer algorithm, which might not be faithful to the internet explorer idea as repeated exploration and refinement is not possible in a name-only setting with no access to training dataset.
10. **Neural Priming** [Wallingford et al., 2023]: Similar to SuS-X-LC, it retrieves samples relevant to the downstream task from LAION2B. However, instead of relying on semantic search on CLIP features, they adopt a search using language for fast initial filtering and image search for accurate retrieval. Additionally, they retrieve samples only from LAION2B, their pretraining set, unlike past works and use a weighted combination of Nearest Class Mean (NCM) and Linear probe for classification.

Type	Method	Pretraining	Birdsnap	Aircraft	Flowers-102	Pets	Cars	DTD
Data-Free	CLIP-ZS [Radford et al., 2021]	CLIP	39.1	27.1	70.4	88.9	65.6	46.0
	CLIP-ZS	OpenCLIP-2B	-	25.9	71.7	90.2	87.4	-
	CuPL [Pratt et al., 2023]	OpenCLIP-2B	-	29.6	72.3	91.2	88.6	-
	VisDesc [Menon and Vondrick, 2022]	CLIP	-	-	-	86.9	-	45.6
	VisDesc [Menon and Vondrick, 2022]	OpenCLIP-2B	-	28.5	72.1	90.2	87.9	-
Use-Data	GLIDE-Syn [He et al., 2022]	CLIP	46.8	30.8	72.6	90.4	66.9	44.9
	SuS-LC [Udandarao et al., 2023]	CLIP	47.7	30.5	73.8	91.6	65.9	55.3
	SuS-SD [Udandarao et al., 2023]	CLIP	45.5	28.7	73.1	90.6	66.1	54.6
	Retrieval+SSL* [Li et al., 2023b]	OpenCLIP-2B	-	26.2	72.1	90.4	88.0	-
	Priming [Wallingford et al., 2023]	OpenCLIP-2B	-	33.0	79.8	91.9	89.3	-
	Priming+CuPL [Wallingford et al., 2023]	OpenCLIP-2B	-	36.0	80.0	91.9	90.2	-
	C2C (Ours-Linear)	CLIP	62.7 (+15.0)	62.0 (+26.0)	87.8 (+7.8)	94.1 (+2.2)	83.8 (-6.2)	61.0 (+5.7)
	C2C (Ours-MLP Adapter)	CLIP	64.4 (+16.7)	66.2 (+30.2)	89.9 (+9.9)	94.5 (+2.6)	85.5 (-4.7)	61.8 (+6.5)
	C2C (Ours-Linear)	DeIT-III (1m21K)	73.4 (+25.7)	63.5 (+27.5)	92.9 (+12.9)	94.8 (+2.9)	71.5 (-18.7)	57.6 (+2.3)
	C2C (Ours-MLP Adapter)	DeIT-III (1m21K)	75.4 (+27.7)	67.4 (+31.4)	93.6 (+13.6)	95.0 (+3.1)	73.9 (-16.3)	60.4 (+5.1)

Table E.3.1: ViTB/16: Comparison with Other Approaches for Name-Only Classification Our uncuration webly-supervised data consistently outperform existing name-only classification techniques, with the exception of the Stanford Cars dataset where Priming outperforms our approach. * indicates that the method significantly differs from the cited work. However, it performs SSL finetuning with retrieved samples, emulating a 1-step Internet-explorer algorithm. Further exploration of LAION2B is not possible here due to the lack of a training set.

Additional Results. Table E.3.2 shows the number of samples used in prior-art compared to our webly-supervised data. Our webly-supervised data uses up-to 155× more samples than previous methods. Table E.3.1 shows similar results and trends as presented in the manuscript for CLIP-ResNet50 but for CLIP-ViTB/16. Our method outperforms prior art by large margins with the exception of Stanford Cars where Neural Priming outperforms our approach.

Training Dataset	Datasets				
	FGVC Aircraft	Flowers102	OxfordIIITPets	Stanford Cars	BirdSnap
<i>Size Relative to Ground Truth Train Sets</i>					
CaFo [Zhang et al., 2023b]	1.6K	1.6K	0.6K	3.1K	8.0K
GLIDE-Syn [He et al., 2022]	3.3K	1.0K	3.7K	8.1K	42K
SuS-X [Udandara et al., 2023]	7.9K	3.2K	2.6K	1K	39K
C2C (Ours)	158K (48x)	155K (155x)	53K (15x)	184K (23x)	557K (13x)

Table E.3.2: Dataset Statistics. Our uncured webly-supervised data is significantly larger in size compared to other approaches for acquiring a training set. Leveraging multiple engines could allow to query the web for samples enabling the creation of datasets up to 15 – 155× larger than alternative approaches for creating a training set in a name-only classification setting.

E.4 Appendix: Continual Webly Supervised Learning

E.4.1 EvoTrends

EvoTrends: Figure E.4.1 visually depicts the construction of our dataset, EvoTrends. This dataset captures the trends of products over a span of 21 years, starting from the year 2000 and continuing until 2020. Each year showcases a distinct trending product, such as the PlayStation 2 in 2000 and N95 Masks in 2020. What sets EvoTrends apart is its truly class-incremental setup, where the learner must quickly adapt to emerging concepts and categories, as opposed to the conventional class-incremental setup where classes are incremented arbitrarily.

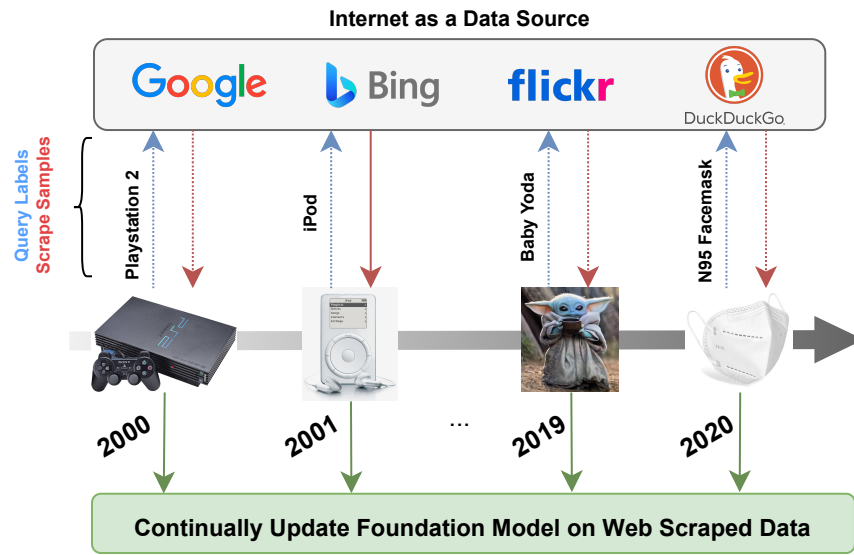


Figure E.4.1: EvoTrends: A Dynamic Dataset Reflecting Real-world Trends. This illustration showcases the dataset we have curated using internet sources. EvoTrends consists of 21 timesteps, spanning the years 2000 to 2020. Each timestep presents the most trending products of the respective year, challenging the learner to adapt to these evolving trends. Unlike artificial scenarios, this dataset accurately reflects a real class-incremental setting, where classes emerge based on actual trends observed in the world.

E.4.2 Computational Budgets

Computational Budgets. In Figure E.4.2 we present a visual representation of the impact of different computational budgets on the training epochs. Each row in the figure corresponds to one of the computational budgets: tight, normal, and relaxed, in sequential order. Specifically, the normal budget is selected to allow the manually annotated datasets to undergo one epoch of training during the initial time step, as depicted by the green plot in the left side figures. Comparatively, the tight

budget, represented by the blue plot, is half the size of the normal budget, while the relaxed budget, indicated by the yellow plot, is four times the size of the normal budget. As the webly-supervised data surpasses the manually curated datasets in terms of size, the number of epochs they undergo within each of the three budget regimes decreases. This is due to the increasing amount of data presented at each time step, resulting in a decay in the effective number of epochs over time.

Dataset Size Comparison and Computational Budget Analysis. We experiment with different levels of computational constraints and present the results in Appendix Table E.4.1. We find that accuracy predictably improves with increasing computational budgets, while the gap between our webly-supervised approach and manually annotated datasets remains relatively similar.

Eval.	Training Dataset	Avg. Acc. on Benchmarks		
		Split-PACS	Split-CIFAR100	CLEAR10
Tight	GT Data	77.1%	35.9%	62.1%
	C2C (Ours)	75.8%	31.9%	56.1%
	C2C (Top-20/engine/class)	73.3%	31.1%	52.3%
	C2C (Top-50/engine/class)	72.8%	31.3%	56.5%
Relaxed	GT Data	88.6%	57.7%	81.8%
	C2C (Ours)	84.2%	49.9%	71.6%
	C2C (Top-20/engine/class)	82.9%	48.0%	66.9%
	C2C (Top-50/engine/class)	84.1%	50.0%	70.0%

Table E.4.1: Consistency of Linear Probing Performance across Computational Budgets. Our findings reveal a remarkable consistency in the results in terms of performance gap across different computational budgets.

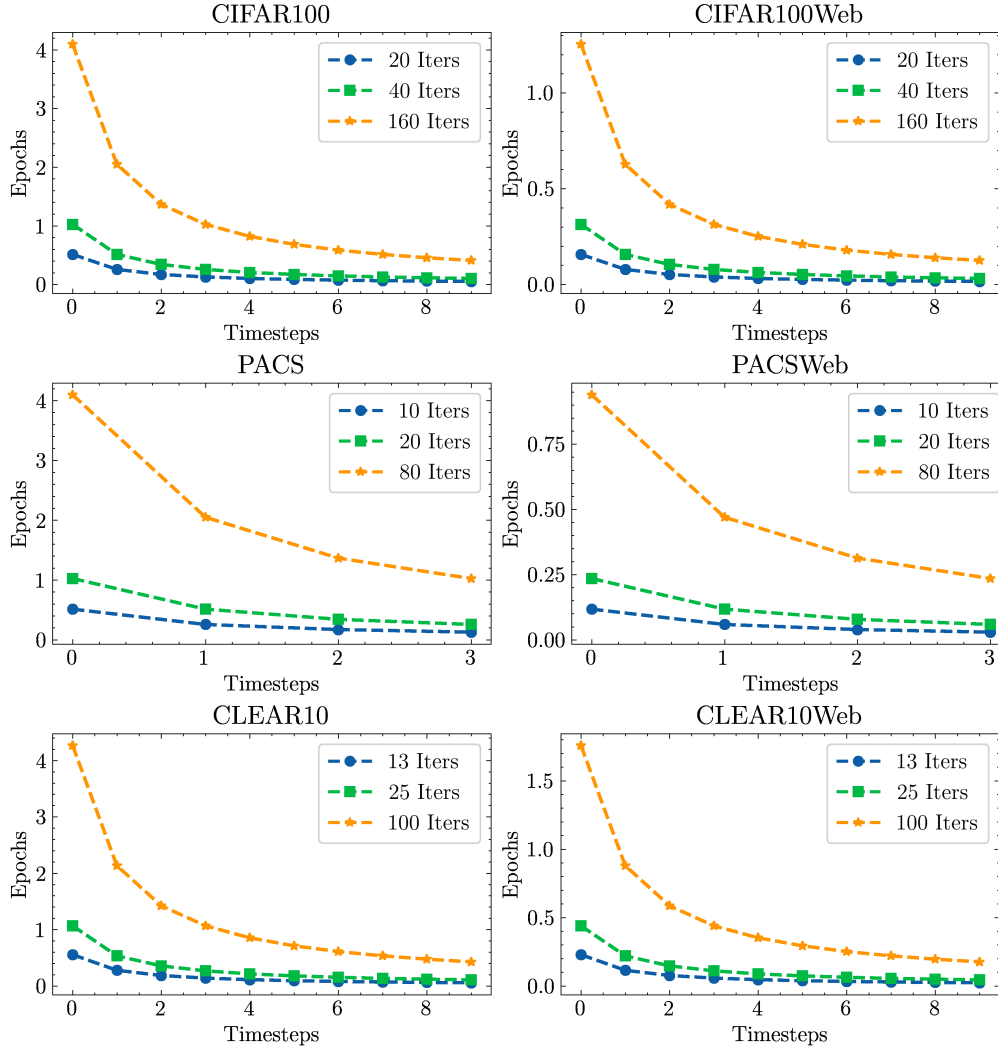


Figure E.4.2: Effective Training Epochs Per Time Step. Each row represents one of the computational budgets: tight, normal, and relaxed in sequential order. The normal budget is carefully chosen to allow the manually annotated datasets to undergo one epoch of training during the initial timestep (depicted by the green plot in the left side figures). The tight budget (blue) is half the budget of normal, while the relaxed budget (yellow) is four times the budget of normal. As the webly-supervised data surpasses the manually curated datasets in size, they undergo fewer epochs within each of the three budget regimes. At each timestep more data is presented, hence the effective number of epochs decays with time.

E.4.3 Investigating Performance Gaps Between Our Webly-Supervised Data and MA Data

In this section, we inspect the reasons for the performance gap between using our webly-supervised data compared to the manually annotated data in the *continual name-only* setup.

Domain Gaps.

CLEAR10. In Table E.4.2 we show class-wise performance analysis of our approach on CLEAR10 dataset. This analysis reveals a noticeable decrease in accuracy for the bus class. Upon further inspection, we found that this decline can be attributed to the disparity between the CLEAR10 test set, which contains images of the bus’s interior, and our webly-supervised data, which predominantly consists of images depicting the bus’s exterior. Consequently, the accuracy for our bus class is significantly lower. For a visual representation of this discrepancy, please refer to Figure E.4.3.

Query Category	Accuracy
Baseball	81.4%
Bus	19.5%
Camera	86.0%
Cosplay	96.2%
Dress	69.6%
Hockey	88.2%
Laptop	95.8%
Racing	77.6%
Soccer	65.2%
Sweater	89.4%

Table E.4.2: CLEAR10 Class-wise Performance. The analysis of class-wise performance in CLEAR10 highlights a significant decline in accuracy specifically for the bus class. This discrepancy can be attributed to the fact that CLEAR10 test set includes images of the "interior" of the bus, whereas our webly-supervised data primarily consists of images depicting the "exterior" of the bus. As a result, the accuracy for our bus class is notably low. Please refer to Figure E.4.3 for a visual representation of this disparity.

PACS. In Table E.4.3 we show an analysis of domain-specific accuracy in Continual PACS which reveals important insights. Our domain-incremental setup involves a dataset comprising four distinct domains: Photo, Art, Cartoon, and Sketch. It is crucial to acknowledge the substantial disparity between the sketches in PACS and our webly-supervised data. While sketches in PACS are characterized as quick drawings, our webly-supervised data encompasses more detailed sketches and line drawings. This domain shift has a significant impact on the performance of our approach, particularly in the Sketch domain, where we observe a notable drop in accuracy. To visually comprehend this domain shift, please refer to Figure E.4.4.

CIFAR100. In Figure E.4.5 we present a comparison between manually annotated CIFAR100 and our webly-supervised version of it which reveals a substantial domain gap. The upper row

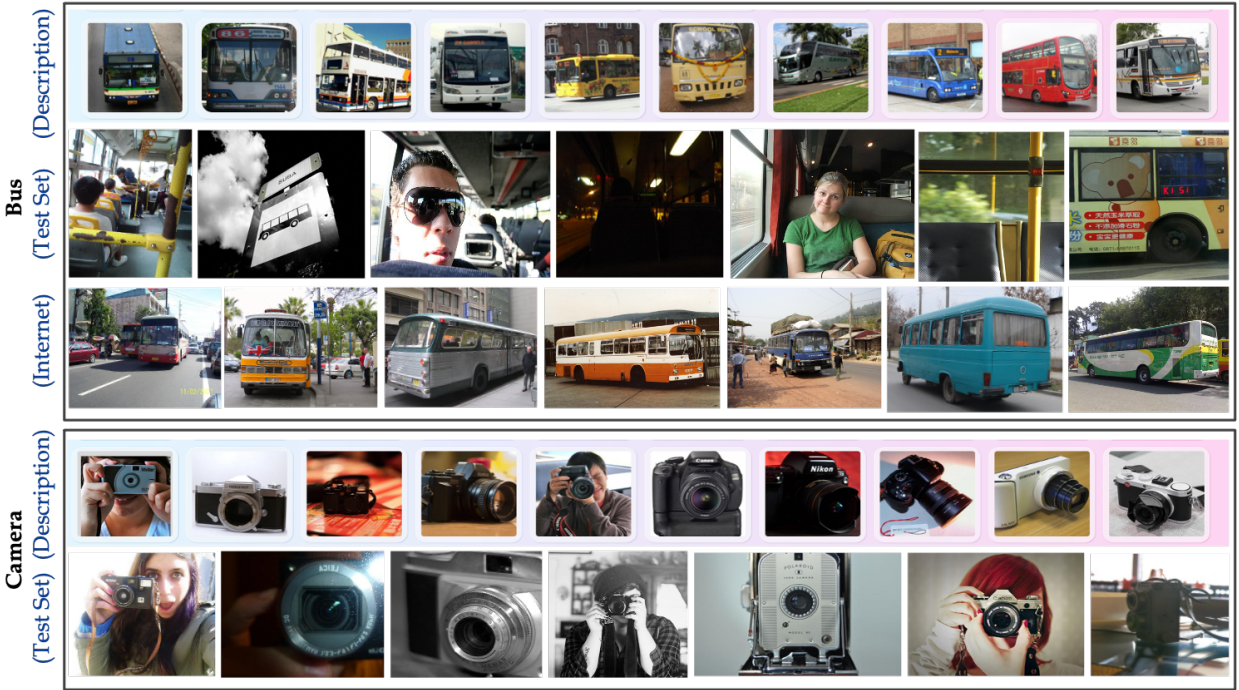


Figure E.4.3: Domain Gap in CLEAR10 Dataset. In the CLEAR10 dataset, the original paper describes buses as the "exterior" of the bus, while the test set predominantly consists of images showcasing the "interior" of the bus. In contrast, our web-collected data primarily comprises "exterior" photos of the buses. Considering the inherent dissimilarity between our buses and the test set, this justifies the 10% performance gap observed when using our webly-supervised data compared to manually annotated datasets. It is important to note that this discrepancy does not apply to other classes within CLEAR10, as demonstrated by the camera class for reference.

Dataset	Photo	Art	Cartoon	Sketch
MA Data	99.4%	91.6%	85.9%	77.4%
C2C (Ours)	96.4% -3.0%	90.7% -0.9%	83.0% -2.9%	66.3% -11.1%

Table E.4.3: Domain-specific Accuracy in Continual PACS. Our domain-incremental setup incorporates a dataset consisting of four domains: Photo, Art, Cartoon, and Sketch. It is important to note that there is a notable disparity between the sketches in PACS, which are characterized as quick drawings, and the sketches in our webly-supervised data, which encompass more detailed sketches and line drawings. This domain shift results in a significant performance drop for our approach in the Sketch domain. This domain shift can be observed in Figure E.4.4.

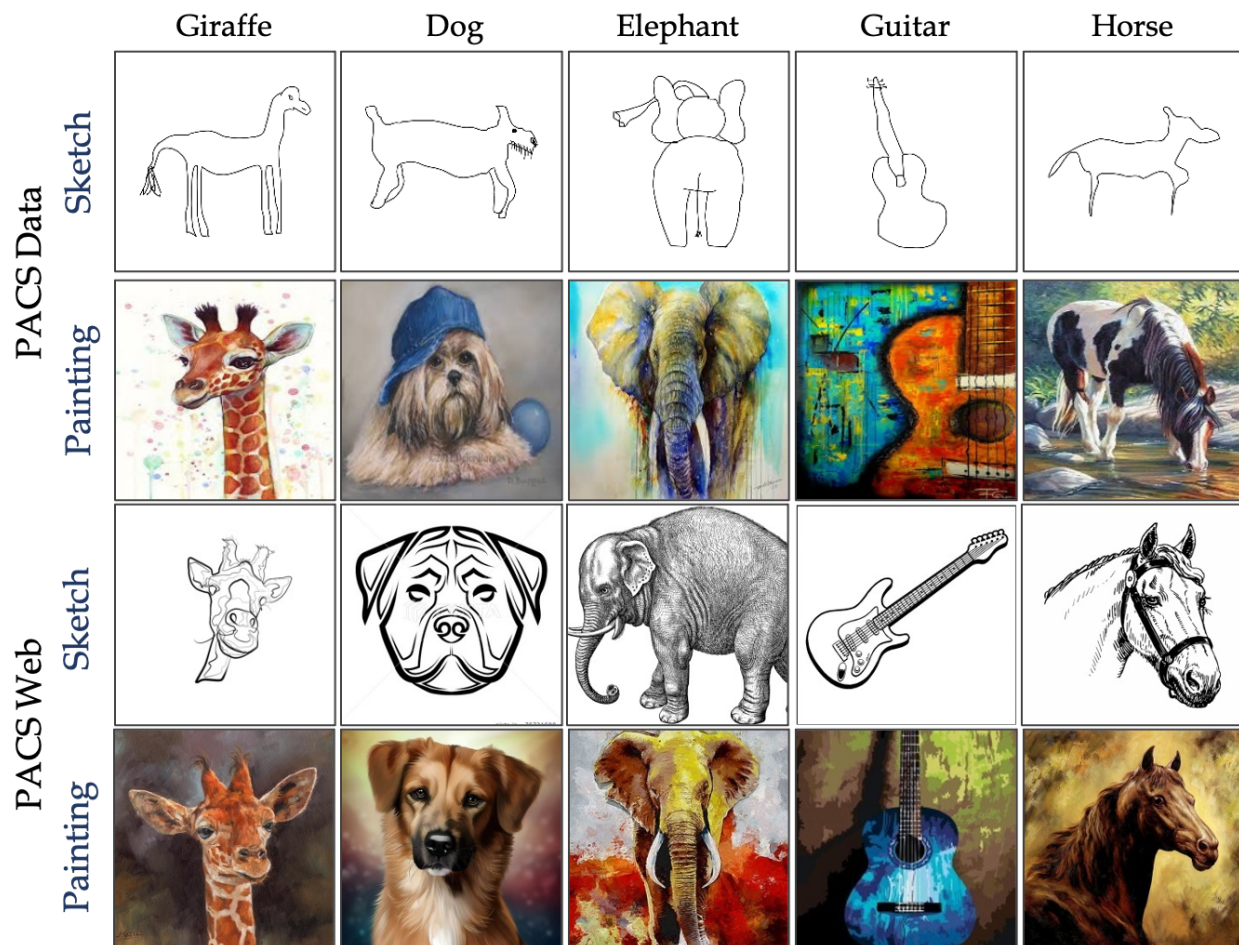


Figure E.4.4: Comparison of PACS and Webly-Supervised Data. The upper two rows depict the sketch and painting domains of the manually annotated PACS dataset, while the last two rows showcase the sketch and painting domains of our webly-supervised data. Although there is some resemblance between the painting domains of both datasets, a significant domain gap becomes apparent when comparing the sketches. In the PACS dataset, sketches refer to quick drawings, whereas in our web search, sketches are composed of line drawings and detailed sketches.

showcases the 32×32 samples obtained from CIFAR100, while the second row displays images collected through our webly-supervised approach. It is evident that the CIFAR100 images exhibit characteristics such as low quality, centered composition, and an outdated nature. In contrast, the samples we collect through our webly-supervised approach are more recent, not necessarily centered, and of significantly higher image quality. To bridge this gap and align the two datasets, we employ downsampling techniques to resize our images to 32×32 pixels. This downsampling process proves to be effective in improving the performance of our webly-supervised approach on CIFAR100, specifically when using a normal budget. We observe a notable enhancement in

Eval. Dataset	Datasets		
	Split-CIFAR100	Split-PACS	CLEAR10
<i>Size Relative to Ground Truth Train Sets</i>			
MA Data	50K	8K	30K
C2C (Ours)	163K (3.3x)	44K (5.5x)	73K (2.4x)
C2C (Top 20/engine/class)	8K (0.16x)	2.2K (0.3x)	2K (0.07x)
C2C (Top 50/engine/class)	20K (0.4x)	5.6K (0.7x)	5K (0.17x)

Table E.4.4: Statistics of Continual Datasets. In contrast to the datasets collected in the *name-only* classification setup, the scale of the datasets used in the *continual name-only* classification scenario is maximally 5.5 $\times$ larger than the manually curated ones, whereas in the previous scenario it was 144 $\times$ larger. This arises from the fact that the continual datasets we compare against are already big in size.

performance, with an approximate 8% increase from 30.3% to 38.7%.

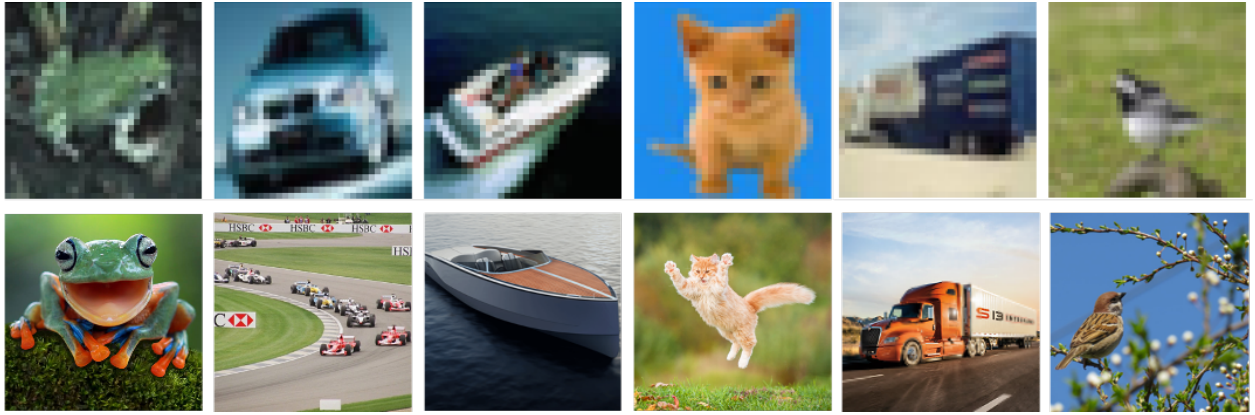


Figure E.4.5: Comparison of CIFAR100 and Webly-Supervised Data. The upper row displays 32×32 samples obtained from CIFAR100, while the second row showcases images collected through our webly-supervised approach. Evidently, a substantial domain gap exists between the CIFAR100 images, characterized by their low quality, centered composition, and outdated nature, and the samples we collect, which are recent, not necessarily centered, and of significantly higher image quality. To bridge this gap, we employ downsampling techniques to resize our images to 32×32 . It is noteworthy that through this downsampling process, we achieve an improvement in the performance of our webly-supervised approach on CIFAR100 with a normal budget, boosting performance by approximately 8% from 30.3% to 38.7%.

Scale of Datasets. Examining the statistics of continual datasets presented in Table E.4.4, we observe an interesting contrast. Unlike the datasets collected in the traditional *name-only* classification setup, the scale of the datasets used in the *continual name-only* classification scenario is maximally 5.5 $\times$ larger than the manually curated ones. This stands in a big contrast to the previous scenario (*name-only* classification), where the scale was 144 $\times$ larger. The reason behind this disparity lies in the fact that the continual datasets we compare against are already substantial in size.

E.5 Ethics

We took steps to ensure that we do not violate copyright laws and avoiding explicit content from the internet. Steps for avoiding explicit content from the internet are detailed in Section 3 in “How do we prevent unintentional download of explicit images?”. We clarify that we will only distribute links of the images for all C2C datasets with the class-names, including EvoTrends for reproducibility. Distributing links of images does not violate copyright laws while allowing reproducibility of our method. Similarly, to the best of our knowledge, training classification models on copyrighted data for this work is covered by copyright laws in UK ¹ and EU ².

¹<https://www.gov.uk/guidance/exceptions-to-copyright#non-commercial-research-and-private-study>

²<https://digital-strategy.ec.europa.eu/en/library/copyright-digital-single-market-brochure>

F

Publications

My thesis contains parts that were completed in collaboration and following is the detailed breakdown of research contributions:

F.1 GDumb: A Simple Approach that Questions Our Progress in Continual Learning (ECCV 2020, Oral)

Contributions:

- Ameya Prabhu designed the GDumb baseline and conducted the experiments.
- Puneet Dokania led the writing of the manuscript, and contextualized the method in the broader literature.
- Supervision and support was provided by Philip Torr and Puneet Dokania.

F.2 RanDumb: A Simple Approach that Questions the Efficacy of Continual Representation Learning (Under Review)

Contributions:

- Ameya Prabhu designed the RanDumb baseline idea.
- Ameya Prabhu and Shiven Sinha led the experiments.

- Ameya Prabhu wrote the manuscript with help from Puneet Dokania and Ozan sener for writing of the manuscript and contextualized the method in the broader literature.
- Supervision and support was provided by Philip Torr and Ponnurangam Kumaraguru.

F.3 Computationally Budgeted Continual Learning: What Does Matter? (CVPR 2023)

Contributions:

- Ameya Prabhu developed the idea and did preliminary experiments with close guidance from Puneet Dokania.
- Ameya Prabhu and Hasan Hammoud scaled the idea up, extending experiments across models/datasets/tasks.
- Adel Bibi was instrumental in revamping the experiments and formalizing the setup constraints.
- Supervision and support was provided by Philip Torr and Bernard Ghanem.
- Writing was led by Hasan Hammoud, Adel Bibi and Ameya Prabhu, with feedback from Philip Torr and Bernard Ghanem.

F.4 Online Continual Learning Without the Storage Constraint (Arxiv)

Contributions:

- A preliminary version of the idea was developed by Ameya Prabhu, Vladlen Koltun redesigned it entirely and simplified it.
- Ozan Sener led the project, with supervision from Puneet Dokania, Philip H. S. Torr and Vladlen Koltun.
- Ameya Prabhu wrote the code and conducted the experiments and analysis.
- Zhipeng Cai contributed an earlier version for the CLOC baseline, later improved by Ameya Prabhu.

F.5 Rapid Adaptation in Online Continual Learning: Are We Evaluating It Right (ICCV 2023)

Contributions:

- Ameya Prabhu and Hasan Hammoud set up the initial version of the idea, with a significantly modified version proposed by Philip Torr and Adel Bibi.
- The code and experiments were led by Ameya Prabhu and Hasan Hammoud.
- Hasan Hammoud made the code fast and efficient, and led the scaling the experimentation pipeline and filling in critical gaps.
- Hasan Hammoud and Ameya Prabhu led the writing, with extensive feedback from Adel Bibi under the supervision of Philip Torr and Bernard Ghanem.

F.6 From Categories to Classifier: Name-Only Continual Learning by Exploring the Web (Under Review)

Contributions:

- Ameya Prabhu and Hasan Hammoud set up the initial version of the idea.
- The code and experiments were developed jointly by Ameya Prabhu and Hasan Hammoud.
- Hasan Hammoud and Ameya Prabhu led the writing, with extensive feedback from Adel Bibi under the supervision of Philip Torr and Bernard Ghanem.

Bibliography

- Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Cited on pages 14 and 48.
- Hongjoon Ahn, Jihwan Kwak, Subin Lim, Hyeonsu Bang, Hyojun Kim, and Taesup Moon. Ss-il: Separated softmax for incremental learning. In *International Conference on Compute Vision (ICCV)*, 2021. Cited on pages 34, 45, and 48.
- Alaaeldin Ali, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, et al. Xcit: Cross-covariance image transformers. *Conference on Neural Information Processing Systems (NeurIPS)*, 2021. Cited on page 63.
- Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. Cited on page 48.
- Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *European conference on computer vision (ECCV)*, 2018. Cited on pages 14 and 22.
- Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Laurent Charlin, and Tinne Tuytelaars. Online continual learning with maximally interfered retrieval. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019a. Cited on pages 14, 18, 19, 20, 21, 34, 35, 45, 47, 62, 63, 67, 78, 80, and 117.
- Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019b. Cited on pages 3, 14, 18, 19, 20, 21, 23, 34, 35, 45, 47, 62, 63, 78, 79, 117, 119, and 121.
- Anonymous. Meta continual learning revisited: Implicitly enhancing online hessian approximation via variance reduction. In *ICLR*, 2024. URL <https://openreview.net/forum?id=TpD2aG1h0D>. Cited on pages xvi, 32, and 35.
- Raja Appuswamy, Renata Borovica-Gajic, Goetz Graefe, and Anastasia Ailamaki. The five-minute rule thirty years later and its impact on the storage hierarchy. In *ADMS@VLDB*, 2017. Cited on page 3.
- Elahe Arani, Fahad Sarfraz, and Bahram Zonooz. Learning fast, learning slow: A general continual learning method based on complementary learning system. In *International Conference on Learning Representations (ICLR)*, 2022. Cited on page 35.

- Martin Aumüller, Erik Bernhardsson, and Alexander John Faithfull. Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Inf. Syst.*, 2020. Cited on page 66.
- Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. Cited on pages 3, 34, 43, 45, 47, 62, 63, 78, and 79.
- Josh Beal, Hao-Yu Wu, Dong Huk Park, Andrew Zhai, and Dmitry Kislyuk. Billion-scale pretraining with vision transformers for multi-task visual representations. In *Winter Conference on Applications of Computer Vision (WACV)*, 2022. Cited on page 146.
- Anthony Bell and Terrence J Sejnowski. Edges are the ‘independent components’ of natural scenes. In *NeurIPS*, 1996. Cited on page 31.
- Eden Belouadah and Adrian Popescu. Il2m: Class incremental learning with dual memory. In *International Conference on Computer Vision (ICCV)*, 2019. Cited on pages 14 and 48.
- Thomas Berg, Jiongxin Liu, Seung Woo Lee, Michelle L Alexander, David W Jacobs, and Peter N Belhumeur. Birdsnap: Large-scale fine-grained visual categorization of birds. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. Cited on page 99.
- Abeba Birhane and Vinay Uday Prabhu. Large image datasets: A pyrrhic win for computer vision? In *Winter Conference on Applications of Computer Vision (WACV)*, 2021. Cited on page 97.
- Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021. Cited on page 2.
- Jorg Bornschein, Alexandre Galashov, Ross Hemsley, Amal Rannen-Triki, Yutian Chen, Arslan Chaudhry, Xu Owen He, Arthur Douillard, Massimo Caccia, Qixuang Feng, et al. Nevis’22: A stream of 100 tasks sampled from 30 years of computer vision research. *arXiv preprint arXiv:2211.11747*, 2022. Cited on pages 45 and 46.
- Matteo Boschini, Lorenzo Bonicelli, Pietro Buzzega, Angelo Porrello, and Simone Calderara. Class-incremental continual learning into the extended der-verse. *TPAMI*, 2022. Cited on page 32.
- Max F Burg, Florian Wenzel, Dominik Zietlow, Max Horn, Osama Makansi, Francesco Locatello, and Chris Russell. A data augmentation perspective on diffusion models and retrieval. *arXiv preprint arXiv:2304.10253*, 2023. Cited on page 102.
- Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. Cited on pages 32, 34, 35, 43, 45, 47, 78, 121, and 122.
- Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New insights on reducing abrupt representation change in online continual learning. *International Conference on Learning Representations (ICLR)*, 2022a. Cited on pages 32, 34, 63, and 67.

- Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New insights on reducing abrupt representation change in online continual learning. In *International Conference on Learning Representations (ICLR)*, 2022b. Cited on pages 43, 45, 48, 55, and 88.
- Zhipeng Cai, Ozan Sener, and Vladlen Koltun. Online continual learning with natural distribution shifts: An empirical study with visual data. In *International Conference on Computer Vision (ICCV)*, 2021. Cited on pages 3, 4, 43, 46, 47, 49, 61, 62, 63, 65, 66, 67, 78, 79, 81, 83, 84, 88, and 134.
- Zhipeng Cai, Vladlen Koltun, and Ozan Sener. Improving information retention in large scale online continual learning. *arXiv preprint*, 2022. Cited on pages 79 and 91.
- Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *International Conference on Computer Vision (ICCV)*, 2021. Cited on page 63.
- Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *European conference on computer vision (ECCV)*, 2018. Cited on pages 14 and 22.
- Hyuntak Cha, Jaeho Lee, and Jinwoo Shin. Co2l: Contrastive continual learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. Cited on pages 34 and 45.
- Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *European conference on computer vision (ECCV)*, 2018. Cited on pages 2, 12, 13, 14, 15, 22, 45, 67, and 78.
- Arslan Chaudhry, Marc’Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *International Conference on Learning Representations (ICLR)*, 2019a. Cited on pages 3, 14, 18, 21, 22, 23, 24, 32, 34, 35, 66, 78, 79, 119, and 121.
- Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc’Aurelio Ranzato. Continual learning with tiny episodic memories. In *International Conference on Machine Learning Workshops (ICMLW)*, 2019b. Cited on pages 14, 22, 32, 34, 35, 43, 45, 47, and 63.
- Arslan Chaudhry, Albert Gordo, David Lopez-Paz, Puneet K. Dokania, and Philip Torr. Using hindsight to anchor past knowledge in continual learning, 2021. Cited on pages 14, 20, 21, and 63.
- Haoran Chen, Zuxuan Wu, Xintong Han, Menglin Jia, and Yu-Gang Jiang. Promptfusion: Decoupling stability and plasticity for continual learning. *arXiv preprint arXiv:2303.07223*, 2023. Cited on page 64.
- Xinlei Chen and Abhinav Gupta. Webly supervised learning of convolutional networks. In *International Conference on Computer Vision (ICCV)*, 2015. Cited on page 146.
- Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. Neil: Extracting visual knowledge from web data. In *International Conference on Computer Vision (ICCV)*, 2013. Cited on page 147.
- Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *International Conference on Computer Vision (ICCV)*, 2021a. Cited on page 63.

- Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. *International Conference on Computer Vision (ICCV)*, 2021b. Cited on pages 95 and 99.
- Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. Cited on page 146.
- Aristotelis Chrysakis and Marie-Francine Moens. Online continual learning from imbalanced data. In *International Conference on Machine Learning (ICML)*, 2020. Cited on pages 3, 47, and 63.
- Aristotelis Chrysakis and Marie-Francine Moens. Online bias correction for task-free continual learning. In *ICLR*, 2023. Cited on page 35.
- Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. *arXiv preprint arXiv:1906.11829*, 2019. Cited on page 147.
- Cody Coleman, Edward Chou, Julian Katz-Samuels, Sean Culatana, Peter Bailis, Alexander C Berg, Robert Nowak, Roshan Sumbaly, Matei Zaharia, and I Zeki Yalniz. Similarity search for efficient active learning and search of rare concepts. In *AAAI Conference on Artificial Intelligence*, 2022. Cited on page 147.
- Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. In *International Conference on Computer Vision (ICCV)*, 2021. Cited on pages 34, 35, 45, 47, 48, and 63.
- Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory G. Slabaugh, and Tinne Tuytelaars. Continual learning: A comparative study on how to defy forgetting in classification tasks. In *TPAMI*, 2020. Cited on pages 2, 15, 18, 22, 62, and 118.
- Matthias De Lange, Gido van de Ven, and Tinne Tuytelaars. Continual evaluation for lifelong learning: Identifying the stability gap. *arXiv preprint arXiv:2205.13452*, 2022. Cited on pages 61, 65, and 112.
- Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. Cited on pages 43, 49, and 122.
- Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyang Wu, and Rama Chellappa. Learning without memorizing. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on page 14.
- Santosh K Divvala, Ali Farhadi, and Carlos Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. Cited on page 147.
- Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *European conference on computer vision (ECCV)*, 2020. Cited on pages 14, 18, 20, 21, 22, 45, 48, 118, and 122.
- Alexei Efros. How to stop worrying and learn to love nearest neighbors, 2017. URL <https://nn2017.mit.edu/wp-content/uploads/sites/5/2017/12/Efros-NIPS-NN-17.pdf>. Cited on page 61.

- Robert Fergus, Li Fei-Fei, Pietro Perona, and Andrew Zisserman. Learning object categories from google’s image search. In *International Conference on Computer Vision (ICCV)*, 2005. Cited on pages 94 and 146.
- Jhair Gallardo, Tyler L Hayes, and Christopher Kanan. Self-supervised training enhances online continual learning. *British Machine Vision Conference (BMVC)*, 2021. Cited on page 105.
- João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys*, 2014. Cited on page 62.
- Saurabh Garg, Mehrdad Farajtabar, Hadi Pouransari, Raviteja Vemulapalli, Sachin Mehta, Oncel Tuzel, Vaishaal Shankar, and Fartash Faghri. Tic-clip: Continual training of clip models. *ArXiv*, 2023. Cited on page 40.
- Deepti Ghadiyaram, Du Tran, and Dhruv Mahajan. Large-scale weakly-supervised pre-training for video action recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on page 146.
- Yasir Ghunaim, Adel Bibi, Kumail Alhamoud, Motasem Alfarra, Hasan Abed Al Kader Hammoud, Ameya Prabhu, Philip HS Torr, and Bernard Ghanem. Real-time evaluation in online continual learning: A new paradigm. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. Cited on pages ix, 45, 67, 69, 78, 79, 81, 83, 84, 87, 88, 89, 90, and 141.
- Shashwat Goel, Ameya Prabhu, Amartya Sanyal, Ser-Nam Lim, Philip Torr, and Ponnurangam Kumaraguru. Towards adversarial evaluations for inexact machine unlearning. *arXiv preprint arXiv:2201.06640v3*, 2023. Cited on pages 79 and 121.
- Heitor M Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfahringer, Geoff Holmes, and Talel Abdessalem. Adaptive random forests for evolving data stream classification. *Machine Learning*, 2017. Cited on pages 62 and 68.
- Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013. Cited on page 2.
- Goetz Graefe. The five-minute rule 20 years later (and how flash memory changes the rules). *Communications ACM*, 2009. Cited on page 3.
- Jim Gray and Goetz Graefe. The five-minute rule ten years later, and other computer storage rules of thumb. *ACM SIGMOD*, 1997. Cited on page 3.
- Jim Gray and Franco Putzolu. The 5 minute rule for trading memory for disc accesses and the 10 byte rule for trading memory for cpu time. In *ACM SIGMOD*, 1987. Cited on page 3.
- Zhongrui Gui, Shuyang Sun, Runjia Li, Jianhao Yuan, Zhaochong An, Karsten Roth, Ameya Prabhu, and Philip Torr. knn-clip: Retrieval enables training-free segmentation on continually expanding large vocabularies. *arXiv preprint arXiv:2404.09447*, 2024. Cited on page 112.
- Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. *International Conference on Learning Representations (ICLR)*, 2021. Cited on page 91.

- Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning (ICML)*, 2017. Cited on page 55.
- Yiduo Guo, Bing Liu, and Dongyan Zhao. Online continual learning through mutual information maximization. In *ICML*, 2022. Cited on pages xv, xvi, 32, 34, and 35.
- Ziyu Guo, Renrui Zhang, Longtian Qiu, Xianzheng Ma, Xupeng Miao, Xuming He, and Bin Cui. Calip: Zero-shot enhancement of clip with parameter-free attention. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2023. Cited on pages 100, 102, 103, and 152.
- Gunshi Gupta, Karmesh Yadav, and Liam Paull. Look-ahead meta learning for continual learning. *NeurIPS*, 2020. Cited on page 35.
- Niv Haim, Gal Vardi, Gilad Yehudai, Ohad Shamir, and Michal Irani. Reconstructing training data from trained neural networks. *Conference on Neural Information Processing Systems (NeurIPS)*, 2022. Cited on pages 78 and 121.
- Hasan Abed Al Kader Hammoud, Ameya Prabhu, Ser-Nam Lim, Philip HS Torr, Adel Bibi, and Bernard Ghanem. Rapid adaptation in online continual learning: Are we evaluating it right? *ICCV*, 2023. Cited on pages 7, 113, and 136.
- Md Yousuf Harun and Christopher Kanan. Overcoming the stability gap in continual learning. *arXiv preprint arXiv:2306.01904*, 2023. Cited on page 112.
- Tyler L Hayes and Christopher Kanan. Lifelong machine learning with deep streaming linear discriminant analysis. In *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020. Cited on pages 28, 32, 37, 63, and 142.
- Tyler L Hayes, Nathan D Cahill, and Christopher Kanan. Memory efficient experience replay for streaming learning. In *ICRA*, 2019. Cited on pages 63 and 68.
- Tyler L Hayes, Maximilian Nickel, Christopher Kanan, Ludovic Denoyer, and Arthur Szlam. Can i see an example? active learning the long tail of attributes and relations. *arXiv preprint arXiv:2203.06215*, 2022. Cited on page 147.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016. Cited on pages 2, 57, and 122.
- Ruifei He, Shuyang Sun, Xin Yu, Chuhui Xue, Wenqing Zhang, Philip Torr, Song Bai, and Xiaojuan Qi. Is synthetic data from generative models ready for image recognition? *International Conference on Representation Learning (ICLR)*, 2022. Cited on pages 95, 100, 102, 103, 152, 153, and 154.
- Timm Hess, Eli Verwimp, Gido M van de Ven, and Tinne Tuytelaars. Knowledge accumulation in continually learned representations and the issue of feature forgetting. *arXiv preprint arXiv:2304.00933*, 2023. Cited on page 29.
- Guillaume Hocquet, Olivier Bichler, and Damien Querlioz. Ova-inn: Continual learning with invertible neural networks. In *IJCNN*, 2020. Cited on pages 20 and 22.

- Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on pages 3, 14, 22, 43, 45, 46, 47, 48, 55, 89, and 122.
- Yen-Chang Hsu, Yen-Cheng Liu, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. In *Conference on Neural Information Processing Systems Workshops (NeurIPS-W)*, 2018. Cited on pages 18 and 118.
- Hexiang Hu, Ozan Sener, Fei Sha, and Vladlen Koltun. Drinking from a firehose: Continual learning with web-scale natural language. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, to appear, 2022. Cited on page 62.
- Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. Cited on pages 22 and 116.
- Ahmet Iscen, Thomas Bird, Mathilde Caron, Alireza Fathi, and Cordelia Schmid. A memory transformer network for incremental learning. *British Machine Learning Conference (BMVC)*, 2022. Cited on page 64.
- Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High accuracy and high fidelity extraction of neural networks. In *USENIX security symposium (USENIX)*, 2020. Cited on page 121.
- Matthew Jagielski, Om Thakkar, Florian Tramer, Daphne Ippolito, Katherine Lee, Nicholas Carlini, Eric Wallace, Shuang Song, Abhradeep Thakurta, Nicolas Papernot, et al. Measuring forgetting of memorized training examples. *arXiv preprint arXiv:2207.00099*, 2022. Cited on page 121.
- Paul Janson, Wenxuan Zhang, Rahaf Aljundi, and Mohamed Elhoseiny. A simple baseline that questions the use of pretrained-models in continual learning. *Conference on Neural Information Processing Systems Workshops (NeurIPS-W)*, 2022. Cited on pages xvi, 32, 37, 64, 68, 98, and 105.
- Xu Ji, Joao Henriques, Tinne Tuytelaars, and Andrea Vedaldi. Automatic recall machines: Internal replay, continual learning and the brain. *Conference on Neural Information Processing Systems Workshops (NeurIPS-W)*, 2020. Cited on pages 18, 20, 21, and 118.
- Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning (ICML)*, 2021. Cited on page 146.
- Bin Jin, Maria V Ortiz Segovia, and Sabine Susstrunk. Webly supervised semantic segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. Cited on page 146.
- Xisen Jin, Junyi Du, and Xiang Ren. Gradient based memory editing for task-free continual learning. 2021. Cited on pages 14, 18, 20, 21, 35, and 117.
- Armand Joulin, Laurens Van Der Maaten, Allan Jabri, and Nicolas Vasilache. Learning visual features from large weakly supervised data. In *European Conference on Computer Vision (ECCV)*, 2016. Cited on page 146.

- Amita Kamath, Christopher Clark, Tanmay Gupta, Eric Kolve, Derek Hoiem, and Aniruddha Kembhavi. Webly supervised concept expansion for general purpose vision models. In *European Conference on Computer Vision*, pages 662–681. Springer, 2022. Cited on page 146.
- Shyamgopal Karthik, Jérôme Revaud, and Boris Chidlovskii. Learning from long-tailed data with noisy labels. *arXiv preprint arXiv:2108.11096*, 2021. Cited on page 101.
- James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. In *PNAS*, 2017. Cited on pages 2, 3, 12, 14, 22, 23, 32, and 78.
- Hyunseo Koh, Dahyun Kim, Jung-Woo Ha, and Jonghyun Choi. Online continual learning on class incremental blurry task configuration with anytime inference. *International Conference on Learning Representations (ICLR)*, 2022. Cited on pages 3, 45, 47, 62, 63, 78, and 79.
- Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *International conference on computer vision workshops (ICCV-W)*, 2013. Cited on page 99.
- Jonathan Krause, Benjamin Sapp, Andrew Howard, Howard Zhou, Alexander Toshev, Tom Duerig, James Philbin, and Li Fei-Fei. The unreasonable effectiveness of noisy data for fine-grained recognition. In *European Conference on Computer Vision (ECCV)*, 2016. Cited on page 146.
- Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. Cited on page 94.
- Pavel Laskov, Christian Gehl, Stefan Krüger, Klaus-Robert Müller, Kristin P Bennett, and Emilio Parrado-Hernández. Incremental support vector learning: Analysis, implementation and applications. *JMLR*, 2006. Cited on pages 62 and 68.
- Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 2015. Cited on page 2.
- Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2017. Cited on pages 14 and 22.
- Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim. A neural dirichlet process mixture model for task-free continual learning. In *International Conference on Learning Representations (ICLR)*, 2020. Cited on page 35.
- Alexander C Li, Mihir Prabhudesai, Shivam Duggal, Ellis Brown, and Deepak Pathak. Your diffusion model is secretly a zero-shot classifier. *International Conference on Computer Vision (ICCV)*, 2023a. Cited on pages 100, 102, 103, and 152.
- Alexander Cong Li, Ellis Langham Brown, Alexei A Efros, and Deepak Pathak. Internet explorer: Targeted representation learning on the open web. In *International conference on machine learning (ICML)*, 2023b. Cited on pages 102, 146, and 153.

- Ang Li, Allan Jabri, Armand Joulin, and Laurens Van Der Maaten. Learning visual n-grams from web data. In *International Conference on Computer Vision (ICCV)*, 2017a. Cited on pages 2, 14, 21, 22, 32, 35, 67, 78, and 146.
- Li-Jia Li and Li Fei-Fei. Optimol: automatic online picture collection via incremental model learning. *International Journal of Computer Vision (IJCV)*, 2010. Cited on page 147.
- Wen Li, Limin Wang, Wei Li, Eirikur Agustsson, and Luc Van Gool. Webvision database: Visual learning and understanding from web data. *arXiv preprint arXiv:1708.02862*, 2017b. Cited on page 146.
- Zhiqiu Lin, Jia Shi, Deepak Pathak, and Deva Ramanan. The clear benchmark: Continual learning on real-world imagery. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021. Cited on pages 3, 4, 43, 46, 47, 62, 78, 79, 104, 121, and 122.
- Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Cited on pages 14, 45, and 47.
- Yaoyao Liu, Bernt Schiele, and Qianru Sun. Rmm: Reinforced memory management for class-incremental learning. *Conference on Neural Information Processing Systems (NeurIPS)*, 2021. Cited on page 45.
- David Lopez-Paz and Marc’Aurelio Ranzato. Gradient episodic memory for continual learning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2017. Cited on pages 2, 3, 12, 13, 14, 15, 16, 19, 21, 22, 23, 44, 62, 78, 79, 80, and 121.
- Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. Cited on page 19.
- Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022. Cited on page 73.
- Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. Exploring the limits of weakly supervised pretraining. In *European Conference on Computer Vision (ECCV)*, 2018. Cited on pages 57, 83, and 146.
- Zheda Mai, Ruiwen Li, Hyunwoo Kim, and Scott Sanner. Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning. In *Conference on Computer Vision and Pattern Recognition Workshop (CVPR-W)*, 2021. Cited on pages 35 and 45.
- Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 2022. Cited on pages 43, 48, and 55.
- Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013. Cited on page 99.
- Yu A Malkov and Dmitry A Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *TPAMI*, 2018. Cited on pages 61, 98, and 105.

- Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. Cited on pages 22 and 48.
- Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *European conference on computer vision (ECCV)*, 2018. Cited on page 48.
- Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost van de Weijer. Class-incremental learning: survey and performance evaluation on image classification. *TPAMI*, 2022. Cited on page 53.
- Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*. 1989. Cited on page 2.
- Mark D McDonnell, Dong Gong, Amin Parveneh, Ehsan Abbasnejad, and Anton van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. In *NeurIPS*, 2023. Cited on pages xv, 29, 37, 38, 39, and 40.
- Geoffrey J McLachlan. *Discriminant analysis and statistical pattern recognition*. John Wiley & Sons, 2005. Cited on page 31.
- Geoffrey J McLachlan. Mahalanobis distance. *Resonance*, 4(6):20–26, 1999. Cited on pages vi, 27, and 28.
- Sachit Menon and Carl Vondrick. Visual classification via description from large language models. *International Conference on Representation Learning (ICLR)*, 2022. Cited on pages 100, 101, 103, 152, and 153.
- Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *TPAMI*, 2013. Cited on pages 28, 62, 68, 98, and 105.
- Seyed Iman Mirzadeh, Arslan Chaudhry, Dong Yin, Timothy Nguyen, Razvan Pascanu, Dilan Gorur, and Mehrdad Farajtabar. Architecture matters in continual learning. *arXiv preprint arXiv:2202.00275*, 2022. Cited on pages xvi, 35, 38, and 40.
- Sudhanshu Mittal, Silvio Galesso, and Thomas Brox. Essentials for class incremental learning. In *CVPR*, 2021. Cited on page 34.
- Jaouad Mourtada, Stéphane Gaïffas, and Erwan Scornet. Amf: Aggregated mondrian forests for online learning. *Journal of the Royal Statistical Society*, 2019. Cited on page 62.
- Jishnu Mukhoti, Yarin Gal, Philip HS Torr, and Puneet K Dokania. Fine-tuning can cripple your foundation model; preserving features may be the solution. *arXiv preprint arXiv:2308.13320*, 2023. Cited on pages 29 and 112.
- Sri Aurobindo Munagala, Sidhant Subramanian, Shyamgopal Karthik, Ameya Prabhu, and Anoop Namboodiri. Clactive: Episodic memories for rapid active learning. In *Conference on Lifelong Learning Agents*, pages 430–440. PMLR, 2022. Cited on page 147.

- Martin Mundt, Yongwon Hong, Iuliia Pliushch, and Visvanathan Ramesh. A wholistic view of continual learning with deep neural networks: Forgotten lessons and the bridge to active and open world learning. *Neural Networks*, 160:306–336, 2023. Cited on page 147.
- Kengo Nakata, Youyang Ng, Daisuke Miyashita, Asuka Maki, Yu-Chieh Lin, and Jun Deguchi. Revisiting a knn-based image classification system with high-capacity storage. In *European Conference on Computer Vision (ECCV)*, 2022. Cited on page 64.
- Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. In *International Conference on Learning Representations (ICLR)*, 2018. Cited on page 14.
- Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian conference on computer vision, graphics & image processing (ICVGIP)*, 2008. Cited on page 99.
- Kenneth A Norman and Randall C O’Reilly. Modeling hippocampal and neocortical contributions to recognition memory: a complementary-learning-systems approach. *Psychological review*, 2003. Cited on page 14.
- OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023. URL <https://api.semanticscholar.org/CorpusID:257532815>. Cited on page 95.
- Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. Cited on pages 113, 146, and 148.
- Oleksiy Ostapenko, Timothee Lesort, Pau Rodríguez, Md Rifat Arefin, Arthur Douillard, Irina Rish, and Laurent Charlin. Continual learning with foundation models: An empirical study of latent replay. In *Conference on Lifelong Learning Agents (CoLLAs)*, 2022. Cited on pages 63 and 80.
- Nikunj C Oza and Stuart J Russell. Online bagging and boosting. In *International Workshop on Artificial Intelligence and Statistics*. PMLR, 2001. Cited on page 62.
- Aristeidis Panos, Yuriko Kobe, Daniel Olmeda Reino, Rahaf Aljundi, and Richard E Turner. First session adaptation: A strong replay-free baseline for class-incremental learning. *arXiv preprint arXiv:2303.13199*, 2023. Cited on pages 71 and 137.
- German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019. Cited on page 62.
- Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *Conference on computer vision and pattern recognition (CVPR)*, 2012. Cited on page 99.
- Karl Ezra Pilario, Mahmood Shafiee, Yi Cao, Liyun Lao, and Shuang-Hua Yang. A review of kernel methods for feature extraction in nonlinear process monitoring. *Processes*, 8(1), 2020. ISSN 2227-9717. doi: 10.3390/pr8010024. URL <https://www.mdpi.com/2227-9717/8/1/24>. Cited on pages vi and 33.
- Ameya Prabhu, Charles Dognin, and Maneesh Singh. Sampling bias in deep active classification: An empirical study. *arXiv preprint arXiv:1909.09389*, 2019. Cited on page 147.

- Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *European conference on computer vision (ECCV)*, 2020. Cited on pages 5, 6, 26, 31, 32, 34, 35, 43, 45, 47, 56, 62, and 63.
- Ameya Prabhu, Zhipeng Cai, Puneet Dokania, Philip Torr, Vladlen Koltun, and Ozan Sener. Online continual learning without the storage constraint. *arXiv preprint arXiv:2305.09253*, 2023a. Cited on pages 7, 40, 43, 45, 46, 78, 79, 80, 81, 87, 88, 98, and 105.
- Ameya Prabhu, Hasan Abed Al Kader Hammoud, Puneet Dokania, Philip HS Torr, Ser-Nam Lim, Bernard Ghanem, and Adel Bibi. Computationally budgeted continual learning: What does matter? In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023b. Cited on pages 6, 40, 78, 84, 87, 88, 89, 96, 98, and 105.
- Ameya Prabhu, Hasan Abed Al Kader Hammoud, Ser-Nam Lim, Bernard Ghanem, Philip HS Torr, and Adel Bibi. From categories to classifier: Name-only continual learning by exploring the web. *arXiv preprint arXiv:2311.11293*, 2023c. Cited on page 8.
- Ameya Prabhu, Shiven Sinha, Ponnuram Kumaraguru, Philip HS Torr, Ozan Sener, and Puneet K Dokania. Randumb: A simple approach that questions the efficacy of continual representation learning. *arXiv preprint arXiv:2402.08823*, 2024. Cited on page 6.
- Sarah Pratt, Ian Covert, Rosanne Liu, and Ali Farhadi. What does a platypus look like? generating customized prompts for zero-shot image classification. *International Conference on Computer Vision (ICCV)*, 2023. Cited on pages 100, 101, 103, 152, and 153.
- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and Others. Learning transferable visual models from natural language supervision. In *International conference on machine learning (ICML)*, 2021. Cited on pages 4, 100, 101, 103, 146, 148, and 153.
- Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *NeurIPS*, 2007. Cited on pages 28, 30, and 34.
- Jathushan Rajasegaran, Munawar Hayat, Salman Khan, Fahad Shahbaz Khan, and Ling Shao. Random path selection for incremental learning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019. Cited on pages 14, 18, 20, 22, 48, and 118.
- Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. itaml: An incremental task-agnostic meta-learning approach. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Cited on pages 14, 21, 22, and 48.
- Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022. Cited on page 95.
- Amal Rannen, Rahaf Aljundi, Matthew B Blaschko, and Tinne Tuytelaars. Encoder based lifelong learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. Cited on page 22.
- Dushyant Rao, Francesco Visin, Andrei Rusu, Razvan Pascanu, Yee Whye Teh, and Raia Hadsell. Continual unsupervised representation learning. *NeurIPS*, 32, 2019. Cited on page 35.

- Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. *Conference on Neural Information Processing Systems (NeurIPS)*, 30, 2017a. Cited on page 48.
- Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017b. Cited on pages 2, 3, 12, 13, 14, 18, 19, 21, 22, 32, 34, 35, 37, 43, 44, 45, 47, 63, 68, 79, 105, 118, 121, and 122.
- Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning (ICML)*, 2019. Cited on page 122.
- Mengye Ren, Michael L Iuzzolino, Michael C Mozer, and Richard S Zemel. Wandering within a world: Online contextualized few-shot learning. *arXiv preprint arXiv:2007.04546*, 2020. Cited on pages 15 and 24.
- Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations (ICLR)*, 2019. Cited on pages 19, 23, and 35.
- Marko Ristin, Matthieu Guillaumin, Juergen Gall, and Luc Van Gool. Incremental learning of random forests for large-scale image classification. *TPAMI*, 2015. Cited on page 62.
- David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P Lillicrap, and Greg Wayne. Experience replay for continual learning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019. Cited on pages 19 and 21.
- Karsten Roth, Jae Myung Kim, A. Sophia Koepke, Oriol Vinyals, Cordelia Schmid, and Zeynep Akata. Waffling around for performance: Visual classification with random words and broad concepts. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. Cited on page 101.
- Mark Rucker, Joran T Ash, John Langford, Paul Mineiro, and Ida Momennejad. Eigen memory tree. *arXiv preprint arXiv:2210.14077*, 2022. Cited on page 68.
- Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. Cited on pages 14, 22, 23, and 48.
- Walter Scheirer, Anderson Rocha, Archana Sapkota, and Terrance Boult. Towards open set recognition. *TPAMI*, 2012. Cited on page 12.
- Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 2015. Cited on page 2.
- Florian Schroff, Antonio Criminisi, and Andrew Zisserman. Harvesting image databases from the web. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2010. Cited on pages 94 and 146.
- Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Conference on Neural Information Processing Systems (NeurIPS)*, 2022. Cited on page 95.

- Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning (ICML)*, 2018. Cited on pages 14 and 22.
- Seonguk Seo, Mustafa Gokhan Uzunbas, Bohyung Han, Sara Cao, Joena Zhang, Taipeng Tian, and Ser-Nam Lim. Online backfilling with no regret for large-scale image retrieval. *arXiv preprint arXiv:2301.03767*, 2023. Cited on page 112.
- Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning (ICML)*, 2018. Cited on page 22.
- Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. The MIT Press, 2006. Cited on page 66.
- Tong Shen, Guosheng Lin, Chunhua Shen, and Ian Reid. Bootstrapping the performance of webly supervised semantic segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. Cited on page 146.
- Dongsub Shim, Zheda Mai, Jihwan Jeong, Scott Sanner, Hyunwoo Kim, and Jongseong Jang. Online class-incremental continual learning with adversarial shapley value. In *Conference on Artificial Intelligence (AAAI)*, 2021. Cited on pages 34, 45, 47, and 63.
- Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2017. Cited on pages 14, 21, and 22.
- Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *IEEE S&P*, 2017. Cited on page 79.
- Mannat Singh, Laura Gustafson, Aaron Adcock, Vinicius de Freitas Reis, Bugra Gedik, Raj Prateek Kosaraju, Dhruv Mahajan, Ross Girshick, Piotr Dollár, and Laurens Van Der Maaten. Revisiting weakly supervised pre-training of visual perception models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. Cited on page 146.
- James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *CVPR*, 2023a. Cited on page 37.
- James Seale Smith, Junjiao Tian, Shaunak Halbe, Yen-Chang Hsu, and Zsolt Kira. A closer look at rehearsal-free continual learning. In *CVPR-W*, 2023b. Cited on pages xvi, 32, and 35.
- Otilia Stretcu, Edward Vendrow, Kenji Hata, Krishnamurthy Viswanathan, Vittorio Ferrari, Sasan Tavakkol, Wenlei Zhou, Aditya Avinash, Emming Luo, Neil Gordon Alldrin, et al. Agile modeling: From concept to classifier in minutes. In *International Conference on Computer Vision (ICCV)*, 2023. Cited on page 147.
- Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *International Conference on Computer Vision (ICCV)*, 2017. Cited on page 146.

- Guolei Sun, Wenguan Wang, Jifeng Dai, and Luc Van Gool. Mining cross-image semantics for weakly supervised semantic segmentation. In *European Conference on Computer Vision (ECCV)*, 2020. Cited on page 146.
- Shengyang Sun, Daniele Calandriello, Huiyi Hu, Ang Li, and Michalis Titsias. Information-theoretic online memory selection for continual learning. *International Conference on Learning Representations (ICLR)*, 2022. Cited on pages 3, 45, 47, and 63.
- Wen Sun, Alina Beygelzimer, Hal Daumé Iii, John Langford, and Paul Mineiro. Contextual memory trees. In *International Conference on Machine Learning (ICML)*, 2019. Cited on page 68.
- Zeren Sun, Yazhou Yao, Xiu-Shen Wei, Yongshun Zhang, Fumin Shen, Jianxin Wu, Jian Zhang, and Heng Tao Shen. Webly supervised fine-grained recognition: Benchmark datasets and an approach. In *International Conference on Computer Vision (ICCV)*, 2021. Cited on page 146.
- Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 2016. Cited on page 134.
- Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations (ICLR)*, 2019. Cited on pages 47 and 65.
- Cheng-Hao Tsai, Chieh-Yen Lin, and Chih-Jen Lin. Incremental and decremental training for linear classification. In *International Conference on Knowledge Discovery and Data mining (KDD)*, 2014. Cited on page 68.
- Endel Tulving. Episodic memory: From mind to brain. *Annual review of psychology*, 2002. Cited on page 14.
- Vishaal Udandaraao, Ankush Gupta, and Samuel Albanie. Sus-x: Training-free name-only transfer of vision-language models. *International Conference on Computer Vision (ICCV)*, 2023. Cited on pages 94, 95, 100, 102, 103, 152, 153, and 154.
- Gido M van de Ven and Andreas S Tolias. Generative replay with feedback connections as a general strategy for continual learning. *arXiv preprint arXiv:1809.10635*, 2018. Cited on page 22.
- Gido M Van De Ven, Zhe Li, and Andreas S Tolias. Class-incremental learning with generative classifiers. In *CVPR-W*, 2021. Cited on pages xiv, 28, and 32.
- Sudheendra Vijayanarasimhan and Kristen Grauman. Keywords to visual categories: Multiple-instance learning for weakly supervised object categorization. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. Cited on page 146.
- Matthew Wallingford, Vivek Ramanujan, Alex Fang, Aditya Kusupati, Roozbeh Mottaghi, Aniruddha Kembhavi, Ludwig Schmidt, and Ali Farhadi. Neural priming for sample-efficient adaptation. *Conference on Neural Information Processing Systems (NeurIPS)*, 2023. Cited on pages 95, 102, and 153.
- Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Foster: Feature boosting and compression for class-incremental learning. In *ECCV*, 2022a. Cited on page 37.

- Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, and Tao Qin. Generalizing to unseen domains: A survey on domain generalization. In *International Joint Conference on Artificial Intelligence*, 2021. Cited on page 91.
- Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *ArXiv*, abs/2302.00487, 2023. Cited on page 77.
- Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Growing a brain: Fine-tuning by increasing model capacity. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. Cited on page 48.
- Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision (ECCV)*, 2022b. Cited on pages 37 and 64.
- Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022c. Cited on pages 27, 32, 33, 34, 37, and 64.
- Yujie Wei, Jiaxin Ye, Zhizhong Huang, Junping Zhang, and Hongming Shan. Online prototype learning for online continual learning. In *ICCV*, 2023. Cited on pages xv, xvi, 34, 35, and 36.
- T. Weyand, A. Araujo, B. Cao, and J. Sim. Google Landmarks Dataset v2 - A Large-Scale Benchmark for Instance-Level Recognition and Retrieval. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Cited on page 134.
- Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. Robust fine-tuning of zero-shot models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. Cited on page 152.
- Tz-Ying Wu, Gurumurthy Swaminathan, Zhizhong Li, Avinash Ravichandran, Nuno Vasconcelos, Rahul Bhotika, and Stefano Soatto. Class-incremental learning with strong pre-trained models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. Cited on pages 48, 63, and 80.
- Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on pages 14, 22, 32, 34, 43, 45, 47, 48, and 55.
- Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. *arXiv preprint arXiv:2103.16788*, 2021. Cited on page 48.
- Huaxiu Yao, Caroline Choi, Yoonho Lee, Pang Wei Koh, and Chelsea Finn. Wild-time: A benchmark of in-the-wild distribution shift over time. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022. Cited on page 46.
- Fei Ye and Adrian G Bors. Continual variational autoencoder learning via online cooperative memorization. In *ECCV*, 2022. Cited on page 35.
- Fei Ye and Adrian G Bors. Self-evolved dynamic expansion model for task-free continual learning. In *ICCV*, 2023. Cited on pages xvi, 32, 35, and 36.

- Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Cited on pages 20 and 21.
- JaeHong Yoon, Jeongtae Lee, Eunho Yang, and Sung Ju Hwang. Lifelong learning with dynamically expandable network. In *International Conference on Learning Representations (ICLR)*, 2018. Cited on pages 14 and 48.
- Jaehong Yoon, Saehoon Kim, Eunho Yang, and Sung Ju Hwang. Scalable and order-robust continual learning with additive parameter decomposition. In *International Conference on Learning Representations (ICLR)*, 2020. Cited on page 14.
- Jaehong Yoon, Divyam Madaan, Eunho Yang, and Sung Ju Hwang. Online coreset selection for rehearsal-based continual learning. *International Conference on Learning Representations (ICLR)*, 2022. Cited on pages 3, 45, 47, and 63.
- Lu Yu, Bartłomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Cited on page 14.
- Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432*, 2021. Cited on page 63.
- Sangdo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on page 19.
- Michał Zając, Tinne Tuytelaars, and Gido M van de Ven. Prediction error-based classification for class-incremental learning. *ICLR*, 2024. Cited on pages xiv, xiv, xv, xv, 27, 28, 32, 33, and 34.
- Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning (ICML)*, 2017. Cited on pages 2, 12, 14, 22, 32, and 35.
- Chen Zeno, Itay Golan, Elad Hoffer, and Daniel Soudry. Task agnostic continual learning using online variational bayes. *arXiv preprint arXiv:1803.10123*, 2018. Cited on pages 21, 32, 43, 48, and 55.
- Gengwei Zhang, Liyuan Wang, Guoliang Kang, Ling Chen, and Yunchao Wei. Slca: Slow learner with classifier alignment for continual learning on a pre-trained model. In *ICCV*, 2023a. Cited on pages xv, 28, 29, and 37.
- Jeffrey O Zhang, Alexander Sax, Amir Zamir, Leonidas Guibas, and Jitendra Malik. Side-tuning: a baseline for network adaptation via additive side networks. In *European Conference on Computer Vision (ECCV)*, 2020a. Cited on page 48.
- Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasci, Larry Heck, Heming Zhang, and C-C Jay Kuo. Class-incremental learning via deep model consolidation. In *Winter Conference on Applications of Computer Vision (WACV)*, 2020b. Cited on pages 14 and 22.

- Renrui Zhang, Xiangfei Hu, Bohao Li, Siyuan Huang, Hanqiu Deng, Yu Qiao, Peng Gao, and Hongsheng Li. Prompt, generate, then cache: Cascade of foundation models makes strong few-shot learners. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023b. Cited on pages 100, 102, 103, 152, and 154.
- Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Cited on pages 14, 43, 45, 47, 48, and 55.
- Jun Zheng, Furao Shen, Hongjun Fan, and Jinxi Zhao. An online incremental learning support vector machine for large-scale data. *Neural Computing and Applications*, 2013. Cited on page 62.
- Wenbo Zheng, Lan Yan, Chao Gou, and Fei-Yue Wang. Webly supervised knowledge embedding model for visual reasoning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Cited on page 146.
- Da-Wei Zhou, Qi-Wei Wang, Han-Jia Ye, and De-Chuan Zhan. A model or 603 exemplars: Towards memory-efficient class-incremental learning. *arXiv preprint arXiv:2205.13218*, 2022. Cited on page 37.
- Da-Wei Zhou, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need. *arXiv preprint arXiv:2303.07338*, 2023a. Cited on page 37.
- Yifei Zhou, Juntao Ren, Fengyu Li, Ramin Zabih, and Ser-Nam Lim. Distribution normalization: An "effortless" test-time augmentation for contrastively learned visual-language models. *arXiv preprint arXiv:2302.11084*, 2023b. Cited on pages 100, 103, and 152.
- Fei Zhu, Zhen Cheng, Xu-yao Zhang, and Cheng-lin Liu. Class-incremental learning via dual augmentation. *NeurIPS*, 2021. Cited on page 34.