How Well Does Self-Supervised Pre-Training Perform with Streaming ImageNet?

Dapeng Hu^{*1}, **Shipeng Yan**^{*2}, **Qizhengqiu Lu**^{3}, **Lanqing Hong**^{4},
Hailin Hu^{3}, **Yifan Zhang**^{1}, **Zhenguo Li**^{4}, **Xinchao Wang**^{1}, **Jiashi Feng**^{1}
^{1}National University of Singapore ^{2}ShanghaiTech University
^{3}AARC, Huawei Technologies ^{4}Huawei Noah's Ark Lab

{dapeng.hu, yifan.zhang}@u.nus.edu, yanshp@shanghaitech.edu.cn
{luqizhengqiu, honglanqing, huhailin2, li.zhenguo}@huawei.com
xinchao@nus.edu.sg, jshfeng@gmail.com

Abstract

Prior works on self-supervised pre-training focus on the joint training scenario, where massive unlabeled data are assumed to be given as input all at once, and only then is a learner trained. Unfortunately, such a problem setting is often impractical if not infeasible since many real-world tasks rely on sequential learning, e.g., data are decentralized or collected in a streaming fashion. In this paper, we conduct the first thorough and dedicated investigation on self-supervised pre-training with streaming data, aiming to shed light on the model behavior under this overlooked setup. Specifically, we pre-train over 500 models on four categories of pre-training streaming data from ImageNet and DomainNet and evaluate them on three types of downstream tasks and 12 different downstream datasets. Our studies show that, somehow beyond our expectation, with simple data replay or parameter regularization, sequential self-supervised pre-training turns out to be an efficient alternative for joint pre-training, as the performances of the former are mostly on par with those of the latter. Moreover, catastrophic forgetting, a common issue in sequential supervised learning, is much alleviated in sequential self-supervised learning (SSL), which is well justified through our comprehensive empirical analysis on representations and the sharpness of minima in the loss landscape. Our findings, therefore, suggest that, in practice, for SSL, the cumbersome joint training can be replaced mainly by sequential learning, which in turn enables a much broader spectrum of potential application scenarios.

1 Introduction

Recent advances in self-supervised learning (SSL) [1–4] demonstrate competitive or even better transfer learning performance on downstream tasks, compared with supervised pre-training. Although waiving the cost of human labeling, SSL usually requires massive unlabeled data to learn a powerful representation model and benefits from significantly large-scale pre-training data [1]. The common pre-training practice follows the **joint training** (JT) setup, where data are collected together before model training. In reality, however, it is usually difficult to access a large amount of collective unlabeled data at once. Instead, real-world data are usually accessed in a streaming fashion, e.g., data are generated and collected sequentially chunk by chunk [5], or even decentralized and stored in different servers [6]; such a learning setup is known as **sequential training** (ST). Despite promising results achieved by JT, it inevitably suffers from heavy data storage and prolonged training time with increasing volume of training data. For ST, on the other hand, a learner can be sequentially trained with disjoint data chunks, making it much more efficient than JT.

*contributed equally.

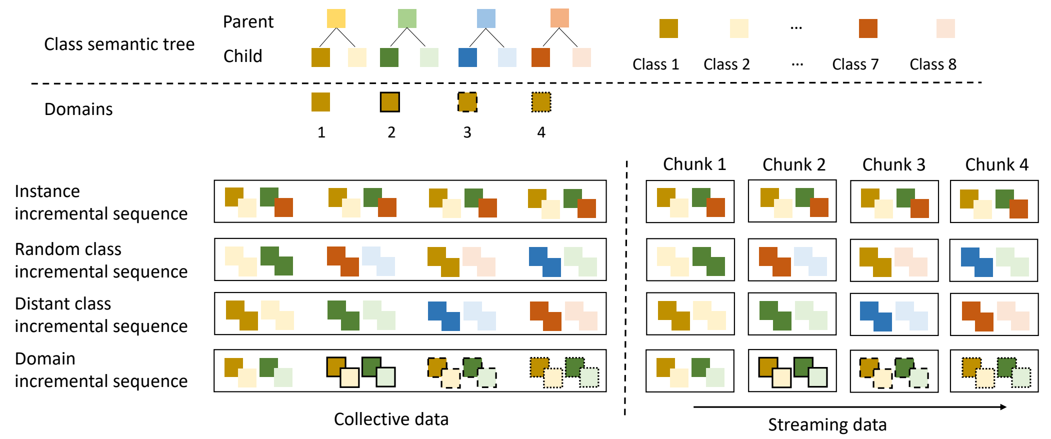


Figure 1: Illustration of streaming data and the corresponding collective data. Different colors denote different classes, and border types distinguish different domains. We use the WordNet Tree [7] to measure the semantic similarity of classes. Classes having the same parent or ancestor in WordNet, marked with similar colors, share similar semantics in the class semantic tree.

Unlike the well-studied supervised continual learning problem [5], how to effectively and efficiently pre-train a representation model under the ST setup has been an open problem. To fill the research gap, we provide a thorough empirical study on the transfer learning behavior of SSL models pre-trained with streaming data. In the pre-training stage, to mimic real-world data collection scenarios and for the ease of dissection of sequential SSL, we consider streaming data with different degrees of distribution shift. As shown in Figure 1, we obtain four types of streaming data, including (1) the instance incremental sequence with negligible data distribution shift, by randomly splitting ImageNet-1K [8] into four identically distributed (IID) data chunks, (2) the random class incremental sequence with moderate data distribution shift, by randomly splitting 1K classes of images into four independent chunks each with 250 classes, (3) the distant class incremental sequence with severe data distribution shift, by splitting 1K classes of data into four chunks while maximizing the semantical dissimilarity among chunks, and (4) the domain incremental sequence with severe domain distribution shift, by taking each domain of data in DomainNet [9] as a data chunk.

As for the evaluation, we consider three downstream tasks following [10], including few-shot evaluation and linear evaluation (also named many-shot classification) on 12 image classification datasets [11], and the Pascal VOC [12] detection task. Through extensive experiments with more than 500 pre-trained models, we thoroughly investigate key roles in sequential SSL, including streaming data, downstream tasks and datasets, continual learning methods, SSL methods, and the method efficiency in terms of time and storage. We also thoroughly investigate knowledge forgetting of sequential SSL and SL models and provide an empirical analysis of the underlying reason.

To the best of our knowledge, we are among the first to explore the sequential self-supervised pre-training setting and the first to provide a thorough empirical study on self-supervised pre-training with streaming data. We summarize the takeaways as well as our contributions as: (1). Sequential SSL models exhibit the on par transfer learning performance as joint SSL models on streaming data with negligible or mild distribution shift. As for streaming data with severe distribution shifts or longer sequences, i.e., the distant class incremental sequence, evident performance gaps exist between sequential SSL and joint SSL models. Such performance gaps, however, can be mitigated effectively and efficiently with unsupervised parameter regularization [13] and simple data replay. (2). Based on the above finding, the standard joint training paradigm may be unnecessary for SSL pre-training. Instead, sequential SSL is performance-competitive but more time-efficient and storage-saving and is well worth considering as the practical practice for self-supervised pre-training with streaming data. (3). Compared with supervised learning (SL) models, SSL models consistently show smaller performance gaps between ST and JT. Our comprehensive investigation of learned representations demonstrates that sequential SSL models are less prone to catastrophic forgetting than SL models. Moreover, through the empirical analysis on the sharpness minima in the loss landscape, we find that SSL models have wider minima than SL models, which we argue is the probable reason for less forgetting of SSL models.

2 Problem Setting

In pre-training, we train representation models on large-scale datasets, such as ImageNet [8], and evaluate the transferability of representations on various downstream tasks [14]. In our empirical study, we adopt the prevailing MoCo-v2 [15] method to pre-train SSL models with streaming data.

Types of streaming data. In pre-training, we consider streaming data with various distribution shifts to mimic practical data collection scenarios. As shown in Figure 1, each type of streaming data consists of sequential and disjoint data chunks, while collective data cover all available data. See Appendix B.1 for a detailed description of the four types of streaming data.

Model pre-training. With these streaming data, we study both sequential training (ST) and joint training (JT) for model pre-training. As illustrated in Figure 1, in sequential training, a model is sequentially trained with streaming data chunks, while in joint training, a model is repeatedly re-trained with collective data, i.e., all seen data chunks. Moreover, we compare SSL with supervised learning (SL) and mainly study the following pre-trained models: sequentially trained SSL models (SSL-ST), jointly trained SSL models (SSL-JT), sequentially trained SL models (SL-ST), and jointly trained SL models (SL-JT). See Appendix B.2 for details of the pre-training stage.

Transfer to downstream tasks. We evaluate the transfer learning performance of pre-trained models using three typical downstream tasks: many-shot classification, few-shot classification, and object detection. See Appendix B.3 for more details of the downstream evaluation.

3 Dissection of Sequential Self-Supervised Pre-Training

In this section, we provide a comprehensive empirical study on SSL models pre-trained with streaming data. Specifically, we study several key factors affecting the downstream transfer learning performance of sequential SSL models, including types of streaming data in Section 3.1, downstream tasks and datasets in Section 3.1, continual learning methods in Section 3.2, and the adopted SSL methods in Section 3.3. In addition, we analyze the time efficiency and the storage efficiency of models trained with streaming data in Section 3.4.

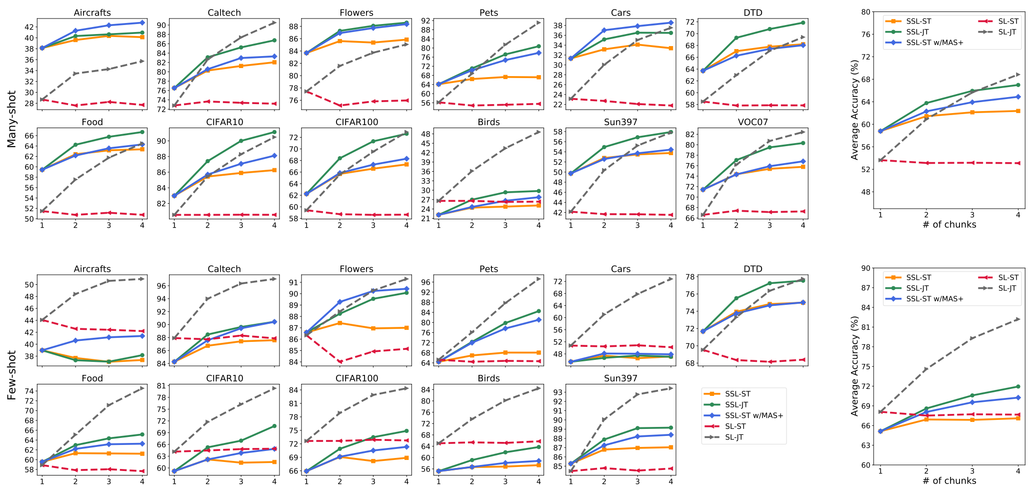


Figure 2: Linear and few-shot evaluation results of **distant class incremental sequence**. on the left are the results of each dataset. On the right are averaged results across all left datasets.

3.1 How does transfer learning performance vary with streaming data?

As shown in Figure 8 and Figures 6-9 in Appendix C.1, generally, performances of sequential SSL models generally increase with more streaming chunks, while sequential SL models do not. As for the performance on each type of streaming data, sequential SSL models are on par with joint SSL models on streaming data with mild distribution shift. On streaming data with severe distribution shift, joint SSL visibly outperform sequential SSL. In contrast, on all types of streaming data, sequential SL models perform obviously worse than joint SL models.

3.2 Do continual learning methods help sequential SSL?

We investigate two classic methods in continual learning [5], i.e., data replay and MAS [13], to mitigate possible performance gaps between SSL-ST and SSL-JT. We denote SSL models trained with data replay as SSL-ST w/Replay, SSL models trained with MAS as SSL-ST w/MAS, and SSL models trained with both methods as SSL-ST w/MAS+. In short, as shown in Figure 8 and Figure 8 in Appendix C.1, we find continual learning methods especially promising to improve performances of sequential SSL models challenging streaming data. See Appendix C.2 for implementations of MAS and data replay in sequential SSL.

3.3 How about SSL methods other than MoCo?

For simplicity, we choose MoCo-v2 [15] in experiments and illustrate sequential SSL is performance-promising. We further try BYOL [2] with the distant class incremental sequence and show results in Figure 10 in Appendix C.3. Similarly, we find SSL models exhibit much smaller performance gaps than SL models, which further validates the potential of sequential SSL in pre-training tasks. See detailed results in Appendix C.3.

Table 1: Resource efficiency of considered SSL pre-training methods. We take the distant class incremental sequence as an example and report the training time (h) and required storage (GB) of the model pre-trained with each data chunk. The lower value means better efficiency.

Time (Storage) / Chunk	2	3	4
SSL-ST	**16.5 (35)**	**16.5 (35)**	**16.6 (35)**
SSL-ST W/Replay	17.0 (35)	18.5 (42)	20.0 (46)
SSL-ST w/MAS	18.2 (35)	18.1 (35)	18.1 (35)
SSL-ST w/MAS+	22.4 (39)	24.4 (42)	26.4 (46)
SSL-JT	31.1 (70)	46.5 (105)	66.6 (140)

Table 2: Comparisons of the averaged accuracy gaps of linear evaluation between ST and JT models. The lower, the better.

Accuracy gap (%) / Chunk	2	3	4
SL-ST (Instance)	2.26	3.27	4.83
SSL-ST (Instance)	**0.41**	**1.02**	**1.04**
SL-ST (Random)	5.63	8.73	10.68
SSL-ST (Random)	**0.42**	**0.94**	**1.13**
SL-ST (Distant)	7.77	12.50	15.75
SSL-ST (Distant)	2.34	3.81	4.62
SSL-ST w/MAS (Distant)	1.82	2.73	3.17
SSL-ST w/MAS+ (Distant)	**1.47**	**2.01**	**2.10**

3.4 Analysis of method efficiency

We then discuss the time and memory consumption of different training methods of SSL, including sequential training, continual learning methods like data replay and MAS, and joint training. As shown in Table 1, sequential SSL is much more time-efficient and storage-saving than JT, especially when the data amount is large or grows quickly. See Appendix D.1 for detailed descriptions.

4 Self-Supervised Models Forget Less than Supervised Models

We compare SSL and SL in terms of the transfer performance gaps between ST models and the corresponding JT models, as shown in Table 2. We find that SL models generally show larger performance gaps than SSL models, which motivates us to further investigate the knowledge forgetting behavior of both SL and SSL models.

4.1 Backward and forward transfer analysis of sequential learning

Following continual learning works [16], we adopt the backward and forward transfer to measure the knowledge forgetting. Backward transfer refers to the improvement of performance on previously learned chunks when learning new chunks, where large negative transfer is also known as catastrophic forgetting. Forward transfer measures the improvement in performance on the novel chunk with the accumulation of knowledge from previous chunks. As shown in Table 3, we find SSL itself is less prone to catastrophic forgetting than

Table 3: Backward and forward transfer analysis of sequential learning.

Data	Method	BWT(%)		FWT(%)	
		Top-1	Top-5	Top-1	Top-5
Instance	SL	-9.45	-5.46	**8.64**	2.81
	SSL	**3.61**	**3.60**	7.55	**8.63**
Random	SL	-20.63	-7.03	-0.34	0.01
	SSL	**-5.17**	**-1.36**	**11.05**	**4.52**
Distant	SL	-40.43	-28.66	4.90	0.47
	SSL	**-13.24**	**-11.06**	**11.01**	**3.66**

SL, especially that SSL achieves positive backward transfer on instance incremental sequence. More details or observations are in Appendix D.3.

4.2 Representations analysis of sequential learning

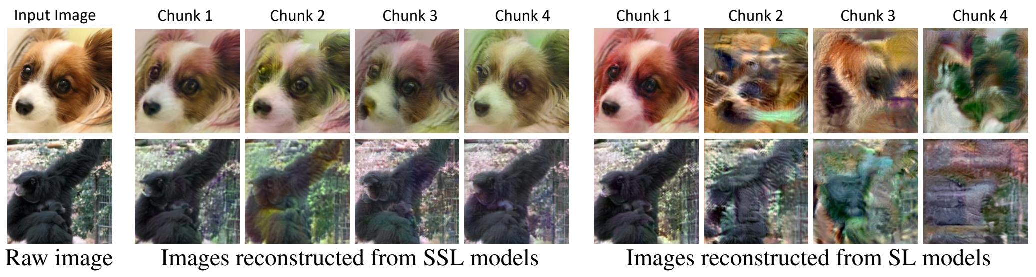


Figure 3: Feature reconstructions of both SSL models and SL models in sequential pre-training.

We investigate the forgetting of previous features and the similarity between ST and JT models.

How do features forget in sequential training? We find features of SSL models forget less information and evolve more slowly than those of SL models in sequential training. The feature reconstructions via deep image prior [17] are shown in Figure 3. Details are in Appendix D.4.

How are ST models similar to JT models? In summary, we find that SSL representations have larger similarity between ST models and JT models than SL representations. Details are in Appendix D.4.

4.3 Why do self-supervised models forget less?

Table 4: The sharpness results. Lower is flatter in the loss landscape. ϵ is the perturbation range.

	Instance		Random Class		Distant Class	
	$\epsilon = 0.1$	$\epsilon = 0.3$	$\epsilon = 0.1$	$\epsilon = 0.3$	$\epsilon = 0.1$	$\epsilon = 0.3$
SL	0.47	0.94	0.21	0.94	0.19	0.94
SSL	**0.14**	**0.68**	**0.08**	**0.66**	**0.06**	**0.71**

We provide an explanation for the forgetting behavior of different training approaches by analyzing the flatness of minima in the loss landscape. Flat minima is the minima in which the change in objective is slow in its large neighborhood. When starting with flat minima, it is expected that learning new chunks will have a smaller effect on performance of the existing chunks, as escaping the flat basin is more difficult [18]. Therefore, we hypothesize that SSL encourages the model to seek out more flat minima, which increases SSL's resistance to catastrophic forgetting. As sharpness results shown in Table 4, we can see that SSL does indeed discover a more flat minima compared to SL, which verifies our hypothesis and providing an explanation for why SSL suffers less forgetting than SL. More implementation details can be found in Appendix D.5.

5 Discussions

This paper has conducted the first thorough empirical evaluation to investigate how well self-supervised learning (SSL) performs with streaming data. Our results show the two main findings as follows: 1). Joint training is unnecessary for SSL, while sequential training with suitable continual learning strategies is performance-competitive yet more efficient, well worth considering as a good alternative. 2). Sequential self-supervised pre-training shows a better capability of overcoming catastrophic forgetting than sequential supervised pre-training. We hypothesize the reason that SSL models have wider minima than SL models in the loss landscape and verify it by experiments.

As for future directions, we first call for more attention to sequential self-supervised learning for understanding its underlying theories of knowledge forgetting and devising better approaches. Also, we recommend considering sequential self-supervised training as a more efficient representation learning paradigm for real-world applications.

Acknowledgements

The authors would like to thank Jiawei Du for his help with implementing sharpness analysis and Yujun Shi for his discussion with the feature decorrelation in SSL. This work was partially supported by AISG-100E-2019-035, MOE2017-T2-2-151, NUS ECRA FY17 P08 and CRP20-2017-0006.

References

[1] He, K., H. Fan, Y. Wu, et al. Momentum contrast for unsupervised visual representation learning. In *Computer Vision and Pattern Recognition*. 2020.

[2] Grill, J.-B., F. Strub, F. Altché, et al. Bootstrap your own latent: A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*. 2020.

[3] Caron, M., I. Misra, J. Mairal, et al. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems*. 2020.

[4] Jure, Z., J. Li, M. Ishan, et al. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*. 2021.

[5] Delange, M., R. Aljundi, M. Masana, et al. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[6] Lange, M. D., X. Jia, S. Parisot, et al. Unsupervised model personalization while preserving privacy and scalability: An open problem. In *Computer Vision and Pattern Recognition*. 2020.

[7] Miller, G. A. *WordNet: An Electronic Lexical Database*. MIT press, 1998.

[8] Russakovsky, O., J. Deng, H. Su, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 2015.

[9] Peng, X., Q. Bai, X. Xia, et al. Moment matching for multi-source domain adaptation. In *International Conference on Computer Vision*. 2019.

[10] Ericsson, L., H. Gouk, T. M. Hospedales. How well do self-supervised models transfer? In *Computer Vision and Pattern Recognition*. 2021.

[11] Kornblith, S., J. Shlens, Q. V. Le. Do better imagenet models transfer better? In *Computer Vision and Pattern Recognition*. 2019.

[12] Everingham, M., L. Van Gool, C. K. Williams, et al. The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision*, 2010.

[13] Aljundi, R., F. Babiloni, M. Elhoseiny, et al. Memory aware synapses: Learning what (not) to forget. In *European Conference on Computer Vision*. 2018.

[14] Chen, T., S. Kornblith, M. Norouzi, et al. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*. 2020.

[15] Chen, X., H. Fan, R. Girshick, et al. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[16] Yan, S., J. Xie, X. He. Der: Dynamically expandable representation for class incremental learning. In *Computer Vision and Pattern Recognition*. 2021.

[17] Ulyanov, D., A. Vedaldi, V. Lempitsky. Deep image prior. In *Computer Vision and Pattern Recognition*. 2018.

[18] Keskar, N. S., D. Mudigere, J. Nocedal, et al. On large-batch training for deep learning: Generalization gap and sharp minima. 2016.

[19] Gidaris, S., P. Singh, N. Komodakis. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations*. 2018.

[20] Noroozi, M., P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*. 2016.

[21] Larsson, G., M. Maire, G. Shakhnarovich. Learning representations for automatic colorization. In *European Conference on Computer Vision*. 2016.

[22] Caron, M., P. Bojanowski, A. Joulin, et al. Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision*. 2018.

[23] Wu, Z., Y. Xiong, S. X. Yu, et al. Unsupervised feature learning via non-parametric instance discrimination. In *Computer Vision and Pattern Recognition*. 2018.

[24] Caron, M., P. Bojanowski, J. Mairal, et al. Unsupervised pre-training of image features on non-curated data. In *International Conference on Computer Vision*. 2019.

[25] Thomee, B., D. A. Shamma, G. Friedland, et al. YFCC100M: The new data in multimedia research. *Communications of the ACM*, 2016.

[26] Mahajan, D., R. Girshick, V. Ramanathan, et al. Exploring the limits of weakly supervised pretraining. In *European Conference on Computer Vision*. 2018.

[27] Gururangan, S., A. Marasović, S. Swayamdipta, et al. Don't stop pretraining: adapt language models to domains and tasks. In *Annual Meeting of the Association for Computational Linguistics*. 2020.

[28] Reed, C. J., X. Yue, A. Nrusimha, et al. Self-supervised pretraining improves self-supervised pretraining. In *International Conference on Computer Vision*. 2021.

[29] Kirkpatrick, J., R. Pascanu, N. Rabinowitz, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 2017.

[30] Zenke, F., B. Poole, S. Ganguli. Continual learning through synaptic intelligence. *Proceedings of Machine Learning Research*, 2017.

[31] Rebuffi, S.-A., A. Kolesnikov, G. Sperl, et al. icarl: Incremental classifier and representation learning. In *Computer Vision and Pattern Recognition*. 2017.

[32] Rolnick, D., A. Ahuja, J. Schwarz, et al. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*. 2019.

[33] Lopez-Paz, D., M. Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*. 2017.

[34] Wang, L., K. Yang, C. Li, et al. Ordisco: Effective and efficient usage of incremental unlabeled data for semi-supervised continual learning. In *Computer Vision and Pattern Recognition*. 2021.

[35] Serra, J., D. Suris, M. Miron, et al. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*. 2018.

[36] Mallya, A., S. Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Computer Vision and Pattern Recognition*. 2018.

[37] Rao, D., F. Visin, A. Rusu, et al. Continual unsupervised representation learning. In *Advances in Neural Information Processing Systems*. 2019.

[38] Aljundi, R., K. Kelchtermans, T. Tuytelaars. Task-free continual learning. In *Computer Vision and Pattern Recognition*. 2019.

[39] Parisi, G. I., R. Kemker, J. L. Part, et al. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.

[40] Yosinski, J., J. Clune, Y. Bengio, et al. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*. 2014.

[41] Han, J., X. Liang, H. Xu, et al. SODA10M: Towards large-scale object detection benchmark for autonomous driving. *arXiv preprint arXiv:2108.12178*, 2021.

[42] Oord, A. v. d., Y. Li, O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[43] He, K., X. Zhang, S. Ren, et al. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*. 2016.

[44] Bossard, L., M. Guillaumin, L. Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*. 2014.

[45] Krizhevsky, A., G. Hinton, et al. Learning multiple layers of features from tiny images. *Master's thesis, University of Tront*, 2009.

[46] Berg, T., J. Liu, S. Woo Lee, et al. Birdsnap: Large-scale fine-grained visual categorization of birds. In *Computer Vision and Pattern Recognition*. 2014.

[47] Xiao, J., J. Hays, K. A. Ehinger, et al. Sun database: Large-scale scene recognition from abbey to zoo. In *Computer Vision and Pattern Recognition*. 2010.

[48] Krause, J., J. Deng, M. Stark, et al. Collecting a large-scale dataset of fine-grained cars. In *Workshop on Fine-Grained Visual Categorization*. 2013.

[49] Maji, S., E. Rahtu, J. Kannala, et al. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[50] Cimpoi, M., S. Maji, I. Kokkinos, et al. Describing textures in the wild. In *Computer Vision and Pattern Recognition*. 2014.

[51] Parkhi, O. M., A. Vedaldi, A. Zisserman, et al. Cats and dogs. In *Computer Vision and Pattern Recognition*. 2012.

[52] Fei-Fei, L., R. Fergus, P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *Computer Vision and Pattern Recognition Workshop*. 2004.

[53] Nilsback, M.-E., A. Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics & Image Processing*. 2008.

[54] Chen, T., S. Kornblith, K. Swersky, et al. Big self-supervised models are strong semi-supervised learners. In *Advances in Neural Information Processing Systems*. 2020.

[55] Ren, S., K. He, R. Girshick, et al. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*. 2015.

[56] Goodfellow, I. J., M. Mirza, D. Xiao, et al. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[57] McCloskey, M., N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*. Elsevier, 1989.

[58] Tian, Y., Y. Wang, D. Krishnan, et al. Rethinking few-shot image classification: a good embedding is all you need? In *European Conference on Computer Vision*. 2020.

[59] Wang, Z., Z. Dai, B. Póczos, et al. Characterizing and avoiding negative transfer. In *Computer Vision and Pattern Recognition*. 2019.

[60] Newell, A., J. Deng. How useful is self-supervised pretraining for visual tasks? In *Computer Vision and Pattern Recognition*. 2020.

[61] Kornblith, S., M. Norouzi, H. Lee, et al. Similarity of neural network representations revisited. In *International Conference on Machine Learning*. 2019.

[62] Zhao, N., Z. Wu, R. W. Lau, et al. What makes instance discrimination good for transfer learning? In *International Conference on Learning Representations*. 2020.

[63] Wen, W., Y. Wang, F. Yan, et al. Smoothout: Smoothing out sharp minima to improve generalization in deep learning. *arXiv preprint arXiv:1805.07898*, 2018.

[64] Mirzadeh, S. I., M. Farajtabar, D. Gorur, et al. Linear mode connectivity in multitask and continual learning. In *International Conference on Learning Representations(ICLR)*. 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] The limitations are discussed in 'Future direction' part of the last section.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] This is a fundamental research and does not have potential negative social impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A] The code is proprietary, but will be made public upon acceptance.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] All the assets are publicly available.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] The data we are using do not contain personally identifiable information or offensive content.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

6 Appendix

A Related Work

Self-supervised learning (SSL). SSL learns useful features by solving various pretext tasks using supervisions generated from unlabeled training data, e.g., predicting rotations [19], solving jigsaw puzzles [20], predicting colorization [21], predicting cluster assignments [22], and solving instance discrimination [23, 14, 1, 2]. To achieve better performance in the downstream task, recent studies of SSL have made efforts in either upstream pre-training or downstream transfer. Previous works [24, 1] have leveraged especially large datasets for pre-training, such as YFCC 100M [25] and Instagram 1B [26]. Some recent works [27, 28] propose to pre-train with the downstream dataset for a better transfer. Our work still focuses on the downstream-agnostic model pre-training. However, in realistic scenarios, access to massive data is often streaming, and how to perform SSL with streaming data has not been studied before, motivating our work.

Continual learning. Existing studies of continual learning (CL) [5] mainly focus on supervised tasks and can be summarized into three categories, including regularization, replay, and parameter-isolation. In regularization-based CL, knowledge preserving is achieved by regularizing the parameter posterior of the new task not to deviate drastically from the prior [13, 29, 30]. Replay-based CL methods overcome forgetting by saving samples of previous tasks in a replay buffer [31–34] and using them to regularize the learning of new tasks. Last, isolation-based CL methods leverage different parameters for learning each task to preserve the learned knowledge [35, 36]. Although works [37, 38] explore continual learning for some specific unsupervised tasks, few have studied the transfer learning performance of sequential self-supervised models.

B Experimental Setups

B.1 Types of streaming Data

We consider four kinds of streaming data for the study, i.e., the instance incremental sequence, the random class incremental sequence, the distant class incremental sequence, and the domain incremental sequence. To exclude the effect of the number of images, we make sure that data chunks in the same sequence have almost the same data amount. Here we provide more details about these data sequences.

Instance incremental sequence. In the instance incremental sequence, streaming data chunks are almost IID, which simulates the scenario where data are continually collected under the same condition. In this case, there is negligible distribution shift among sequential data chunks. Specifically, we split the ImageNet [8] training data that consists of 1.28 million images with 1,000 classes into four even chunks. We ensure that each data chunk includes the same 1,000 classes with the same number of images for each class, which means these data chunks are independent and identically distributed (IID).

Random class incremental sequence. In the random class incremental sequence, data in disjoint chunks belong to different classes, which mimics the scenario where data are collected by random keyword search on the Internet [39]. Here the distribution shift is moderate. Specifically, we randomly split the 1,000 classes of ImageNet into four parts where each part has 250 classes. Since each class of ImageNet has around 1,000 images, we can directly obtain four data chunks with almost the same amount of images.

Distant class incremental sequence. To explore the data sequence with severe distribution shift among data chunks, we consider the distant class incremental sequence. The distant class incremental sequence is similar to the random class incremental sequence except that the semantic gaps between sequential data chunks in the distant sequence are larger, i.e., images from different data chunks are semantically dissimilar. This data sequence has severe distribution shift between chunks. It mimics the scenario where data are crawled from websites with different subjects. Following [40], rather than randomly splitting the 1,000 classes, we leverage the WordNet Tree [7] to obtain four even data chunks sharing the minimal semantic overlapping. We first build a 1000*1000 adjacent matrix among the 1,000 classes by setting the value of similar classes as 1 and the value of dissimilar classes as 0. To be specific, we take classes sharing the common parent node beneath the ninth depth in the WordNet Tree as similar classes and vice versa. Using the semantic similarity described in the adjacent matrix, we then split the 1,000 classes into independent connected components as shown in Figure 4. Finally, we merge these imbalanced components into four almost even data chunks.

Domain incremental sequence. In the domain incremental sequence, data chunks are collected from different domains with severe domain distribution shift. A typical example is that large-scale autonomous driving data in [41] are collected in different domains, such as different weather conditions and cities, but share similar classes. The first three types of streaming data are designed with ImageNet [8], while the domain incremental sequence consists of five domains in DomainNet [9]. Specifically, we consider a multi-domain dataset called DomainNet [9]. In our work, we adopt a domain incremental data sequence made of four distant domains including 'sketch', 'real', 'painting' 'quickdraw', and 'clipart'. There exist severe domain distribution shift among data in these five domains. Specifically, data in the domain 'quickdraw' mostly contain only lines without visual textures. As a result, images from 'quickdraw' are less informative and more visually distinct, compared with images from those four domains, as shown in Figure 5. For each domain, we randomly select 48,129 images as a data chunk, except for 'quickdraw' where we select 47,687 images.

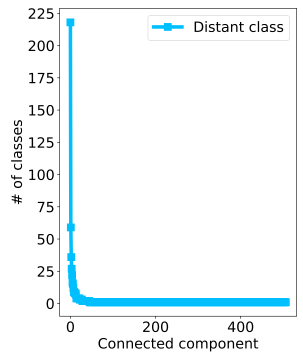


Figure 4: The number of classes for each connected component from the adjacent matrix of 1,000 classes in ImageNet.

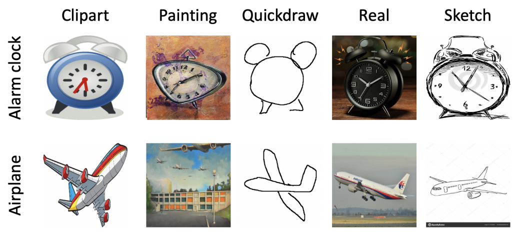


Figure 5: Example images in the five domains of DomainNet.

B.2 Details of pre-training

MoCo-v2. For the illustration purpose, we adopt a prevailing self-supervised learning (SSL) method, MoCo-v2 [15], to investigate the performance of SSL with streaming data. MoCo-v2 uses a Siamese network consisting of two encoders. These two encoders are designed for query images and key images, respectively, and share the same architecture where an MLP projection head f_w is on top of a backbone network f_θ. Only the query encoder is updated by the gradients backpropagation while the key encoder is updated by the moving average with a momentum. MoCo-v2 maintains an additional dictionary as a queue of features for contrastive learning. Specifically, features in the dictionary are progressively updated. The current mini-chunk features from the key encoder are enqueued and the same number of oldest features are dequeued. MoCo-v2 uses InfoNCE [42], a variant of contrastive loss (CL), to maximize the similarity of features from positive pairs and minimize the similarity of features from negative pairs. The contrastive loss is formalized as below.

$$\mathcal{L}_{cl} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{e^{(z_i^\top z_i^+/\tau)}}{e^{(z_i^\top z_i^+/\tau)} + \sum_{z_i^- \in Z^-} e^{(z_i^\top z_i^-/\tau)}}, \tag{1}$$

where N is the number of samples, z_i is the L2-normalized projected feature from the query encoder, z_i^+ is the L2-normalized projected feature of the same input image from the key encoder, Z^- are the negative history features stored in the dictionary and τ is the temperature.

Pre-training. For self-supervised pre-training, we follow the protocol of MoCo-v2 [15], i.e., using the standard ResNet50 backbone [43]. The implementation is based on OpenSelfSup[2]. For both

[2]https://github.com/open-mmlab/OpenSelfSup

joint training and sequential training, the number of training epochs is 200 for each model training, where the convergence of loss is observed. For the instance incremental sequence, we consider one random sequence as the data are randomly divided. While for the random class incremental sequence, distant class incremental sequence, and domain incremental sequence, we experiment with different sequences of data chunks. In particular, considered sequences are obtained through right circular shift operations. For example, if the data sequence length is 4, after splitting all the data into four chunks A, B, C, and D, four sequences, namely A-B-C-D, B-C-D-A, C-D-A-B, and D-A-B-C are used for the sequential pre-training a representation model. The results from different sequences are averaged to obtain the final performance. For comparison, supervised pre-training is also implemented using OpenSelfSup following the recommended training protocol of ImageNet. For supervised pre-training, the classifier layer is reset at a new data chunk.

B.3 Details of downstream tasks

We evaluate the transfer performance of the pre-trained models using three different downstream tasks. Following [14], we consider 12 diverse image classification datasets including Food-101 [44], CIFAR10 [45], CIFAR100 [45], Birdsnap [46], SUN397 [47], Standard Cars [48], FGVC Aircraft [49], VOC2007 [12], DTD [50], Oxford-IIIT Pets [51], Caltech-101 [52] and Oxford 102 Flowers [53]. On these datasets, we evaluate the pre-trained models via the many-shot classification and the few-shot classification (except VOC2007). Both classification protocols are the same as [10]. In addition, we evaluate the pre-trained models on the PASCAL VOC detection task, following the same transfer protocol of MoCo [1]. The training data of detection come from VOC2007 and VOC2012, and the test data come from VOC2007.

Table 5: The inverse of regularization strength (weight decay value) used in many-shot logistic regression evaluation on 12 different downstream classification datasets. SSL models: self-supervised models. SL models: supervised models.

Dataset	SSL Models	SL Models
Aircraft	5623.413277133687	9.99999985098839
Caltech-101	316227.7712565657	0.3162277621819913
Flowers	31622.77530666721	999.999952502551
Pets	999.999952502551	562.3413185099295
Cars	5623.413277133687	17.782794106882072
DTD	1778.2794843157246	0.0177827946252197
Food	177827.94843157247	0.0562341298247638
CIFAR10	316227.7712565657	0.0562341298247638
CIFAR100	100.00000223517424	0.0562341298247638
Birdsnap	1778.27948431572	0.1
SUN397	100.00000223517424	0.0177827946252197
VOC2007	9.99999985098839	0.005623413223739

Many-shot classification. Many-shot classification is a widely used evaluation protocol [54, 1]. To evaluate the pre-trained representations, a linear classifier is directly added to the pre-trained feature encoder. During the downstream task evaluation, only the added linear classifier is fine-tuned using a substantial amount of downstream labeled data while the feature encoder is frozen. In this way, the downstream transfer performance can directly reflect the generalization ability of the pre-trained representation models.

Few-shot classification. Few-shot classification reflects how well the pre-trained models perform on downstream tasks in the few-shot learning regime. Specifically, we consider 5-way 5-shot few-shot tasks on 11 downstream classification datasets, following the few-shot setting in [10]. Concretely, the pre-trained model is fixed for extracting representations. In contrast to many-shot evaluation, in few-shot evaluation, only a few downstream labeled data are provided to obtain prototypes for different categories and then the classification is based on the nearest prototype.

Detection. To further evaluate the transferability of the pre-trained models on more downstream scenarios, we consider object detection as a downstream task, where the fine-grained spatial location information is more important, compared with classification tasks. To be specific, we follow the settings in [1], i.e., adopting the Faster-RCNN [55] with a backbone of R50-dilated-C5 and fine-tuning all layers including the pre-trained representation network.

We perform no hyper-parameter tuning for few-shot evaluation and detection evaluation. As for the linear evaluation protocol, we adopt the logistic regression and only tune the weight decay value. The inversed weight decay values for all downstream classification datasets are given in Table 5.

C More Experimental Results

C.1 Results of other streaming data

We pre-train representation models on four types of streaming data and evaluate pre-trained models on 12 downstream datasets with three downstream evaluation tasks. Note that models pre-trained with ImageNet-based streaming data are evaluated on all three downstream tasks. Models trained with the domain incremental sequence are only evaluated with few-shot classification, considering that the size of each data chunk in DomainNet is only $1/5$ that of each chunk in ImageNet. Besides the distant class incremental sequence, we also report downstream evaluation results of the instance incremental sequence, random class incremental sequence, and the domain incremental sequence in Figure 6, Figure 7, and Figure 8, respectively. We also evaluate three types of ImageNet-based streaming data on object detection and illustrate results in Figure 9.

Transfer learning results of self-supervised pre-training with the instance incremental sequence are evaluated on all three downstream tasks. For results of both many-shot classification and few-shot classification in Figure 6, we find sequential SSL performs comparably with joint SSL on all downstream datasets, with the average performance gap between sequential training and joint training less than 1%, while there exists evident gaps, more than 4%, between sequential supervised learning and joint supervised learning.

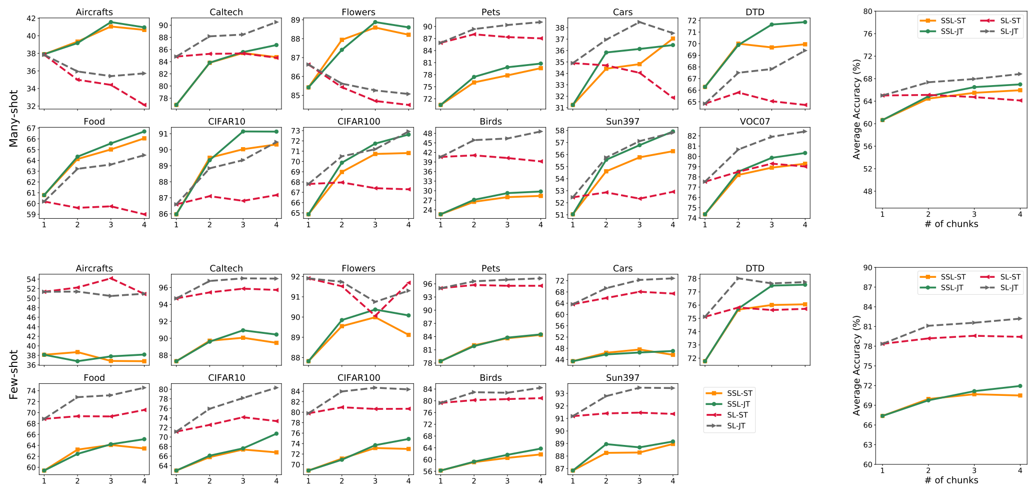


Figure 6: Linear and few-shot evaluation results of **instance incremental sequence**. on the left are the results of each dataset. On the right are averaged results across all left datasets.

C.2 Details of continual learning methods

Continual learning is assumed to suffer from catastrophic forgetting of previously learned knowledge in supervised learning [56, 29, 57], leading to significant performance degradation of previous tasks. Here we introduce the continual learning techniques we adopt, including data replay [32, 31] and regularization-based method, e.g., Memory Aware Synapses (MAS) [13].

Data replay. Data relay is a simple yet effective method for alleviating the catastrophic forgetting problem during the continual learning process. Specifically, we need to maintain a replay buffer and store a selected subset of samples from each learned task in the buffer. Then we just retrain on samples in the replay buffer to revisit old tasks while training the model for a new task.

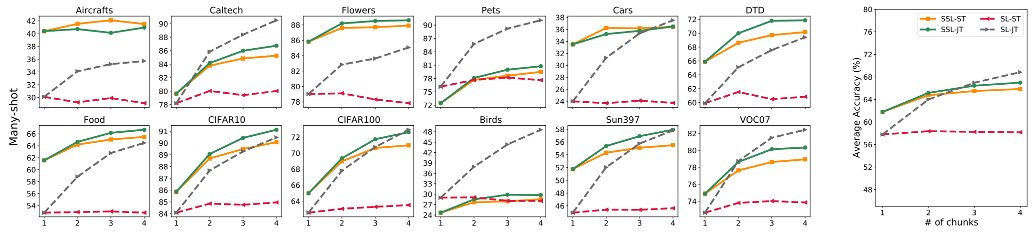
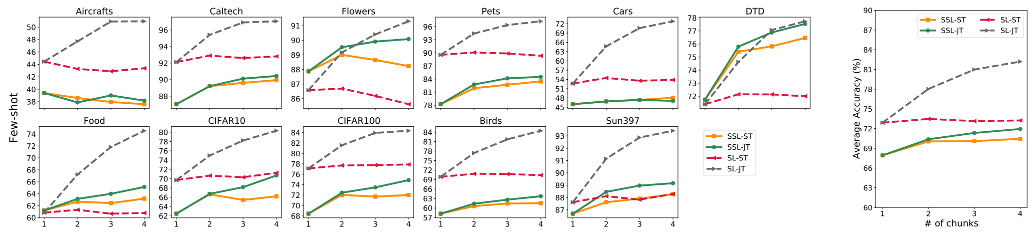



Figure 7: Linear and few-shot evaluation results of **random class incremental sequence**. On the left are the results of each dataset. On the right are averaged results across all left datasets.

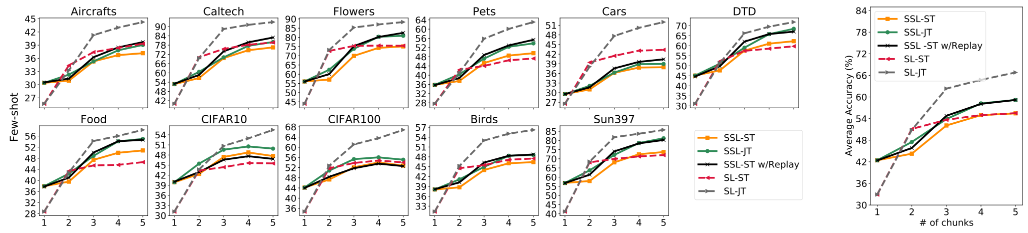
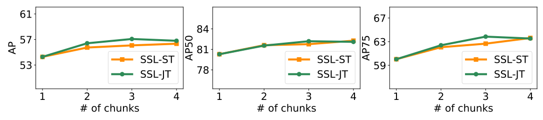
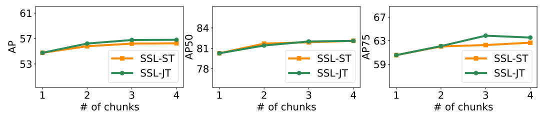


Figure 8: Few-shot evaluation results of **domain incremental sequence**. on the left are the results of each dataset. On the right are averaged results across all left datasets.


(a) Instance incremental sequence


(b) Random class incremental sequence

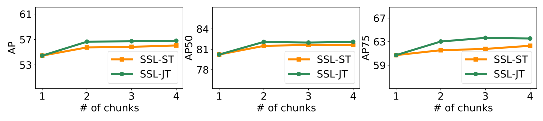

(c) Distant class incremental sequence

Figure 9: Object detection evaluation results of three types of ImageNet-based streaming data.

To perform data replay in the sequential training process, we maintain a replay buffer consisting of images sampled from previous data chunks. After finishing the sequential training with each data chunk, we randomly select 10% data of this chunk and store sampled data in the replay buffer. For the sequential training with the current data chunk, we directly mix current data with data in the replay buffer for self-supervised pre-training e.g. MoCo-v2 training using Eqn. (1).

MAS. Regularization-based methods aim to mitigate the catastrophic forgetting by consolidating previous knowledge with an added regularization term in the loss function. One typical unsupervised regularization-based method is MAS [13]. Specifically, MAS proposes to compute gradients the squared L2-norm of the encoder output f_θ as the importance weights of parameters.

$$\Omega_{ij} = \frac{1}{N}\sum_{k=1}^{N} \frac{\partial[\|f_\theta(x)\|_2^2)]}{\partial\theta_{ij}}. \tag{2}$$

With the parameter regularization term added, the resulting loss function with the coefficient λ is shown as below.

$$\mathcal{L}(\theta) = \mathcal{L}_{cl}(\theta) + \lambda\sum_{i,j}\Omega_{ij}(\theta_{ij} - \theta_{ij}^*)^2. \tag{3}$$

In MoCo-v2, the query encoder is considered as the representation network, we thus only impose the MAS regularization term on parameters of the query encoder. Specifically, the regularization coefficient λ is fixed to be 100. Following the prevailing use of MAS regularization [13, 38], we update the MAS importance weights Ω_{ij} to cover information of each data chunk in the sequential training process. To be specific, after finishing the sequential training with each data chunk, we leverage both the current data chunk and data in the replay buffer to estimate importance weights for the trained model using Eqn. (2). Then we update the stored sequential importance weights by a cumulative moving average of current and previous estimated importance weights, following [38]. As for the model training on the current data chunk, we apply the parameter regularization using previous importance weights and optimize the model using Eqn. (3).

To sum up, besides the sequentially trained model, both above methods require extra storage for sequential self-supervised pre-training. For the 10% data replay method, we need to only keep 10% data of each previous data chunk for sequential training. For the MAS regularization method, we only require to save a set of importance weights for the model and then update the importance weights sequentially.

As shown in Figure 8, data replay can totally eliminate the performance gaps between sequential SSL models and joint SSL models on the domain incremental sequence. Results in Figure 2 also validate the effectiveness of both continual learning methods in improving the transfer learning performance of sequential SSL models when faced with streaming data with severe distribution shift. In short, we find methods devised for supervised continual tasks are especially promising to make sequential SSL models perform comparably to joint SSL models on challenging streaming data.

C.3 Results of BYOL

To evaluate whether sequential training performs well for other SSL methods, we conduct the challenging distant class incremental sequence experiments with BYOL [2]. The results of BYOL are shown in Figure 10. Similar to the observations with MoCo-v2 in Section 3.1, sequential SSL is visibly inferior to joint SSL on streaming data with severe distribution shift, but sequential SL performs obviously worse than joint SL. In addition, compared with SL models, SSL models show significantly smaller performance gaps between sequential training and joint training.

C.4 Results on different downstream tasks and datasets

In pre-training tasks, we pay attention to the generalization of the learned representations to new data or tasks rather than the performance on the training dataset. As shown in Figures 7-8, sequential SSL is performance-promising across the three downstream tasks, supported by the average results across all datasets. Taking a closer look at the results, we observe that, although joint SSL models achieve comparable performance to joint SL models in linear evaluation, joint SL models significantly outperform joint SSL models in few-shot evaluation. This observation is also demonstrated in [58, 10]. The main difference between the two evaluation protocols is that linear evaluation involves more

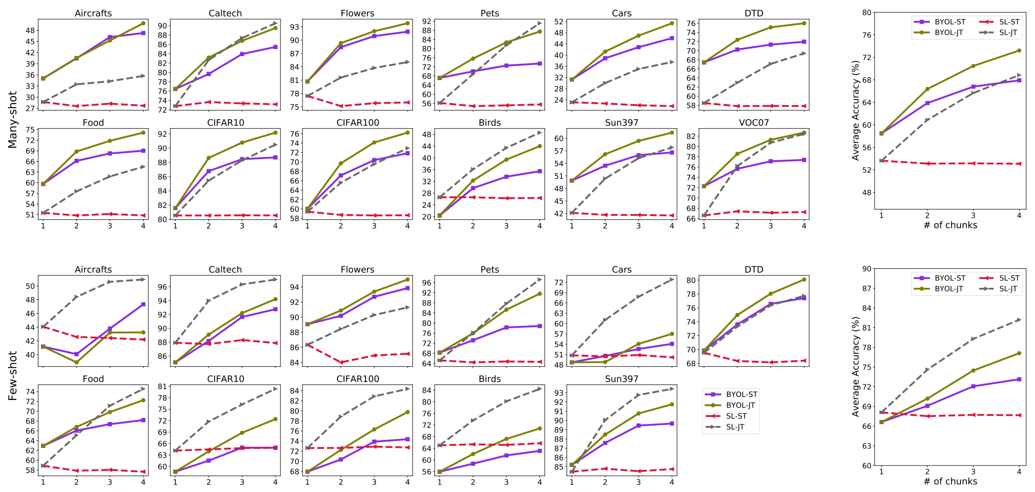


Figure 10: Linear and few-shot evaluation results of **distant incremental sequence** for BYOL. on the left are the results of each dataset. On the right are averaged results across all left datasets.

fine-tuning than few-shot evaluation, as introduced in Appendix B.3. Therefore, the underlying reason for the observation is that supervised features are correlated with labels and more discriminative, thus easy to directly transfer to downstream datasets similar to upstream pre-training data (DomainNet or ImageNet). For example, SL models dominate most few-shot object or scene classification tasks but fail on DTD [50], a texture classification dataset sharing no common classes with ImageNet or DomainNet. In contrast, self-supervised features are more generalized and comprehensive, thus requiring more fine-tuning for desirable downstream transfer. In addition, on some downstream datasets, we have seemingly abnormal observations that ST models may outperform JT models and the model performance may drop with the increase of chunk number. These phenomena are due to the so-called "negative transfer" [59], which is also discussed in other model pre-training studies [60, 27]. That is, pre-training with more data chunks generally improves the model ability, but does not necessarily benefit a specific downstream dataset if the added training data are irrelevant to the downstream dataset. Compared with sequential SSL, it is observed that sequential SL suffers more severe "negative transfer" on the performance of downstream datasets.

D More Empirical Analysis

D.1 Analysis of the efficiency

We then discuss the time and memory consumption of different training methods of SSL, including sequential training (SSL-ST), ST with data replay (SSL-ST w/Replay), ST with MAS (SSL-ST w/MAS), ST with MAS and data replay (SSL-ST w/MAS+), and joint training (SSL-JT). As shown in Table 1, JT is very time-consuming especially when the data amount is large, while ST is able to save a large amount of time under sequential training scenarios. To be specific, ST is about 2x faster than JT when there are 2 chunks of data, and is about 4x faster when the number of chunks is 4. Moreover, when we use MAS and data replay to improve the performance of ST, the time consumption of SSL increases a little but is still significantly faster than JT. As for storage consumption, we can observe a similar phenomenon as shown in Table 1. In summary, sequential SSL is much more time-efficient and storage-saving than JT, especially when the data amount is large or grows quickly. Such a result indicates that sequential SSL is a more favorable choice for real-world pre-training applications, where data come in sequentially and grow daily.

D.2 Pre-training loss

Figure 11 shows the training loss on each step for various types of streaming data. We can see that the training loss increases from instance incremental learning, random class incremental learning, to distant class incremental learning at the beginning of step 2. It reflects that the data distribution

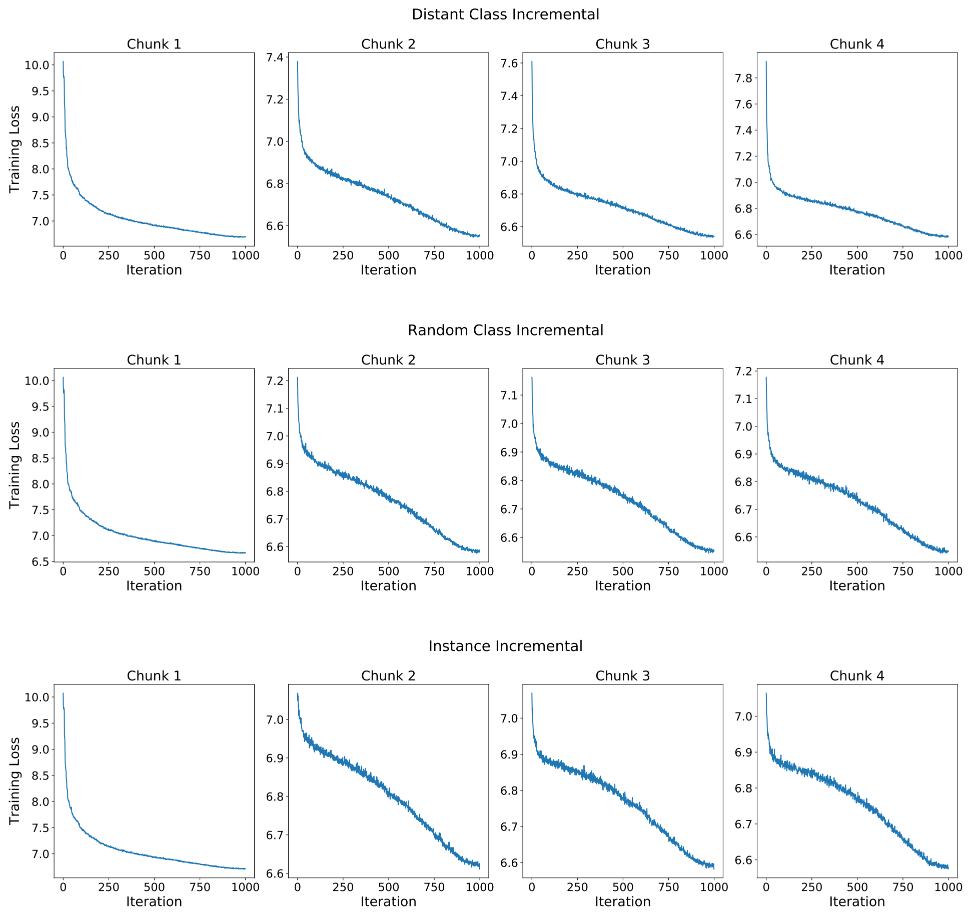


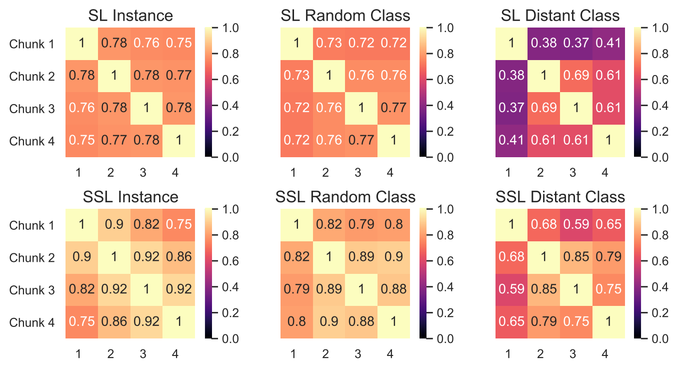
Figure 11: Training Loss on each step of various types of streaming data.

shift increases when changing the distribution from instance incremental learning, random class incremental learning to distant class incremental learning.

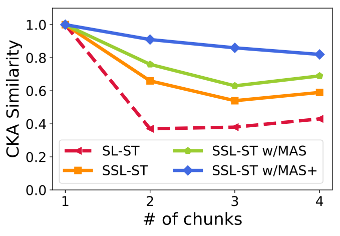
D.3 Details of BWT and FWT

For supervised continual learning, the performance is defined as the accuracy on the associated test set, which is meaningful due to the consistency of the training and test sets. However, the data on downstream task are significantly different from the training set. In this work, to measure the backward and forward transfer in self-supervised continual learning, we assume that the training labels are known and perform k-Nearest Neighbor(KNN) classification to evaluate the representation quality of pretrained model, similar to [23]. Concretely, the backward transfer BWT $= \frac{1}{T-1}\sum_{i=2}^{T}\frac{1}{i}\sum_{j=1}^{i} A^{i}_{\mathcal{Y}_j} - A^{j}_{\mathcal{Y}_j}$ and forward transfer FWT $= \frac{1}{T-1}\sum_{i=2}^{T} A^{i}_{\mathcal{Y}_i} - \tilde{A}_{\mathcal{Y}_j}$ metrics used in [16] where T is the sequence length, $A^{i}_{\mathcal{Y}_j}$ refers to the accuracy on the chunk j using model learned at step i where the label space includes all observed classes up to chunk j, and $\tilde{A}_{\mathcal{Y}_j}$ means the accuracy with the model learned from scratch. For accuracy $A^{i}_{\mathcal{Y}_j}$ on chunk j, we first extract the features for all the examples in the chunk j, and then perform KNN classification in the feature space. Specifically, we set the number of nearest neighbor k=200 for the KNN classification.

We can obtain two more observations about forgetting: *(1) Types of streaming data:* The model suffers progressively severe forgetting when the distribution shift increases for both SSL and SL cases. *(2) Example forgetting:* It is observed that forgetting is less severe in top-5 classification than top-1 classification, which indicates that the knowledge is not fully forgotten.



(a) CKA scores between sequentially trained models.



(b) CKA similarity scores between the sequentially trained models and the corresponding jointly trained models at the step of each data chunk on the distant class incremental sequence.

Figure 12: CKA similarity analysis of sequentially trained models. Given images in the first data chunk, figure (a) shows the similarity of features between different sequential models on three types of ImageNet-based streaming data. With the same images, figure (b) shows the similarity of features between ST models and the corresponding JT models w.r.t. each data chunk on the distant class incremental sequence. The higher CKA similarity value, the more similar.

D.4 Details of CKA similarity analysis

To further understand the sequential self-supervised pre-training, we then take a closer look at the learned feature representations during the sequential training process. We leverage the linear centered kernel alignment (CKA) [61] to measure the similarity of output features between two different representation networks given the same data set as input. If we consider the size of the data set as n and the feature dimension for two networks as d_1 and d_2, respectively. We use the selected data set to extract features $X \in \mathbb{R}^{n\times d_1}$ from one representation network and features $Y \in \mathbb{R}^{n\times d_2}$ from another representation network. In our experiments, n is 50,000 and both d_1 and d_2 are 2,048. We first preprocess the two representation matrices by centering the columns. Then the linear CKA similarity between two representations X and Y can be computed as below:

$$CKA(X, Y) = \frac{\|X^TY\|_F^2}{\|X^TX\|_F^2\|Y^TY\|_F^2}.$$

How do features forget in sequential training? We first study how learned features forget in sequential training via the Centered Kernel Alignment (CKA) [61]. CKA is usually used to measure the similarity between two representations of the same given samples. Specifically, we randomly sample 50,000 images from the first data chunk on each type of streaming data. We use these samples and the sequentially trained models for CKA similarity analysis. We report the CKA similarity values on three types of ImageNet-based streaming data in Figure 12(a). Each value in Figure 12(a) is obtained by first extracting features of samples with two different models and then computing the CKA feature similarity value between the two features. On all streaming data, we have three consistent observations about the CKA similarity between sequential models: (1) SSL models all exhibit higher features similarity to the initial model, compared with SL models. (2) In general, SSL models show higher features similarity between two sequential models in sequential training, compared with SL models. (3) Features similarity between two sequential models decrease on streaming data with more severe distribution shift, for both SSL and SSL. These observations suggest that features of SSL models forget less information and evolve more slowly than those of SL models in sequential training.

How are ST models similar to JT models? We then evaluate CKA similarity, for each data chunk, between features from the sequentially trained model and features from the corresponding jointly trained model. The same 50,000 samples are used for CKA features similarity analysis. For example, as shown in Figure 12(b), at the step of the second data chunk, we compute the CKA similarity value between features of the model jointly trained with the first two data chunks and features from the model sequentially trained after the second data chunk. The corresponding CKA similarity value is 0.4, which indicates for SL, the difference between the ST model and the JT model is very large. In contrast, SSL has a higher similarity of 0.7 between the ST model and the JT model. Particularly, with MAS and data replay, the CKA similarity increases to about 0.9, which means the model trained by sequential SSL extracts nearly the same features as the jointly trained model does. Since the analysis is on the streaming data with severe distribution shift, this further reinforces our hypothesis that, with the help of suitable continual learning methods, sequential SSL pre-training is promising to replace joint training on streaming data with various distribution shift.

Feature reconstructions of sequential models. Similar to [62], in Figure 3, we visualize feature reconstructions of both sequential SL models and sequential SSL models using deep image prior (DIP) [17]. To be specific, we choose four images in the first data chunk of the challenging distant class incremental sequence and visualize features of four sequentially learned models for both SSL and SL, respectively. As shown in Figure 3, in sequential training, features of SSL models can always perfectly reconstruct the main information in original images. In contrast, features of SL models lose more detailed information with the increase of data chunks, which indicates SSL is much better at countering the knowledge forgetting in sequential training. Recalling the evolving CKA similarity shown in Figure 12(a), the perfect reconstruction results of sequential SSL models do not mean SSL models stop learning in sequential training, but it indicates that SSL does well in learning new knowledge while keeping previous knowledge.

D.5 Sharpness analysis

To verify this hypothesis, we undertake experiments to compare the sharpness of SL and SSL minimas. Concretely, we introduce the sharpness by adopting the similar metric in [18, 63]. The neighborhood $\mathcal{C}_\epsilon$ of mimina is defined as follows

$$\mathcal{C}_\epsilon = \{\boldsymbol{z} \in \mathbb{R}^n : -\epsilon||\boldsymbol{\theta}||_2 \leq ||\boldsymbol{z}||_2 \leq \epsilon||\boldsymbol{\theta}||_2\} \tag{4}$$

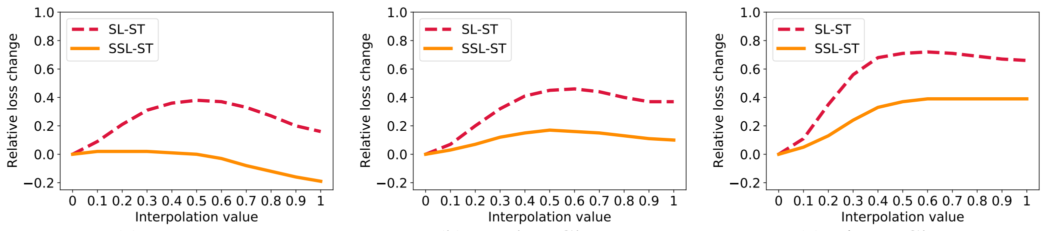
where n is the number of parameters and $\boldsymbol{\theta}$ refers to the model parameter after training. Note that it is unfair to compare the sharpness at minima of SL and SSL directly due to different losses used. However, the losses of SL and SSl can be considered as the proxies for 0-1 loss. As a result, we propose to compare the sharpness on the 0-1 loss for both SL and SSL, where we perform KNN classification to obtain 0-1 loss, i.e. classification accuracy. Then the sharpness $\Phi_{\theta,f}$ is defined as follows

$$\Phi_{\boldsymbol{\theta},f}(\epsilon) = \max_{\boldsymbol{\theta}' \in \mathcal{C}_\epsilon} g(\boldsymbol{\theta}') = \frac{f(\boldsymbol{\theta}) - \min_{\boldsymbol{\theta}'} f(\boldsymbol{\theta}')}{f(\boldsymbol{\theta})} \tag{5}$$

where $g(\boldsymbol{\theta}')$ means the relative loss change from minima $\boldsymbol{\theta}$ to the parameter $\boldsymbol{\theta}'$, and the loss function $f(\boldsymbol{\theta})$ is the negative KNN classification accuracy with model parameter $\boldsymbol{\theta}$.

As shown in Tab. 4, we can see that SSL does indeed discover a more flat minima compared to SL, which verifies our hypothesis and providing an explanation for why SSL suffers less forgetting than SL. More implementation details can be found in Appendix D.5. Moreover, we also conduct the visualization of relative loss change g over a linear path like [64] in Appendix D.5.

For calculation details, we compute the sharpness on chunk 1 for different types of splits in ImageNet. Considering the function f is not differentiable, we sample θ from $\mathcal{C}_\epsilon$ and run 50 times to take the minimal accuracy. For computational efficiency, we randomly sample 0.1M data points to perform kNN classification with k=200. For flatness visualization, we show the normalized loss along the specified path by performing linearly interpolation between the model after chunk 1 and the model after chunk 2 for different splits, as shown in Figure 13. We can see that the compared to SL, loss remains low for SSL along the linear path, which reflects SSL's superiority in terms of flatness.



(a) Instance (b) Random Class (c) Distant Class

Figure 13: Relation loss change for different interpolations of parameters