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

Anonymous Author(s)
Affiliation
Address
email

Abstract

The common self-supervised pre-training practice requires collecting massive unlabeled data together and then trains a representation model, dubbed **joint training**. However, in real-world scenarios where data are decentralized or collected in a streaming fashion, the joint training scheme is storage-heavy, time-consuming, and even infeasible. A more efficient alternative is to train a model continually with streaming data, dubbed **sequential training**, which, however, has not been investigated by previous works. To this end, in this paper, we conduct thorough experiments to investigate self-supervised pre-training with streaming data. Specifically, we evaluate and compare the transfer performance of self-supervised models between joint training and sequential training. We pre-train over 400 models on 4 types of pre-training streaming data from ImageNet and DomainNet, and evaluate them on 3 kinds of downstream tasks and 12 different downstream datasets. Surprisingly, we find that (1) as for self-supervised pre-training, with the help of simple data replay or parameter regularization, sequential training is promising to exhibit comparable transfer ability to joint training on various streaming data, and (2) when sequentially trained with streaming data chunks, self-supervised models have visibly less knowledge forgetting of the first data chunk than supervised models. Based on our findings, we believe sequential self-supervised training is a **more efficient yet performance-competitive** representation learning practice for real-world pre-training applications.

1 Introduction

Relying on supervised learning with large-scale labeled data, deep neural networks (DNNs) are able to extract transferable features beneficial to various visual tasks [1, 2]. As a result, it has been a popular paradigm to first pre-train a DNNs model on a large-scale labeled database (e.g., ImageNet [3]) and then transfer learned features to target downstream tasks. However, supervised pre-training requires massive labeled data, which are usually difficult to collect and annotate. To exempt expensive labeling, existing works have resorted to self-supervised learning (SSL) with large-scale unlabeled data. SSL aims to learn useful features via solving various pretext tasks [4–8] using labels generated from unlabeled data themselves. Recent advances in SSL [8, 9] demonstrate comparable or even better transfer performance on various downstream tasks, compared with supervised learning (SL).

Although SSL waives the cost of human labeling, it usually requires massive unlabeled data to learn a good representation model. Meanwhile, it is desirable to leverage significantly large-scale unlabeled data, e.g., billion-scale data to pre-train a strong model [7]. However, it is not easy to collect together such a large amount of unlabeled data. In realistic scenarios, data are usually streaming, generated and collected sequentially chunk by chunk, or data cannot be distributed elsewhere due to the privacy

Submitted to NeurIPS 2021Workshop on ImageNet: Past, Present, and Future. Do not distribute.

and can only be visited sequentially. How to leverage these streaming data to pre-train a strong self-supervised representation model is well-worth studying.

The common pre-training scheme with SSL collects massive unlabeled data together and trains a SSL model jointly using these data, dubbed **joint training** (JT). For a sequential data chunks, joint training requires to store all seen data chunks and re-train the representation model with both the new chunk and historical chunks. Drawbacks are evident that joint training is extremely storage-heavy, time-consuming, and not able to learn with decentralized data. A more efficient learning scheme for supervised learning with streaming data is to continually train a supervised model, dubbed **sequential training** (ST). Sequential training is assumed to suffer from catastrophic forgetting in supervised learning [10–12], showing significant performance degradation of previously learned tasks. Though more efficient and applicable to various streaming data, sequential training has not been investigated in SSL so far. Whether sequential SSL with streaming data suffers the similar degradation of transfer performance on downstream tasks is still unclear.

In this work, we empirically study the transfer learning behavior of SSL models sequentially pre-trained on streaming data chunks. To make a comprehensive investigation, we consider 4 types of streaming data with different degrees of data distributions shifts, including ImageNet-based streaming data, i.e., the instance incremental sequence, the random class incremental sequence, and the distant class incremental sequence, and the DomainNet-based streaming data [13], i.e., the domain incremental sequence. As for downstream evaluation, following [14], we conduct 3 downstream tasks, including few-shot evaluation and linear evaluation on 12 image classification datasets [15], and Pascal VOC [16] detection. Besides the effect of streaming pre-training data and downstream tasks on the transfer learning performance, we investigate the effect of different SSL methods, and the potential help of continual learning methods. We further analyze the resource efficiency and knowledge forgetting behaviour of sequential SSL. We summarize findings and takeaways as below:

- Sequential SSL exhibits almost the same transfer performance as joint SSL on streaming data with mild distribution shifts. As for streaming data with large distribution shifts, i.e., the distant class sequence and the domain sequence, there exist evident transfer performance gaps between sequential SSL and joint SSL. Such performance gaps, however, can be mitigated effectively and efficiently with unsupervised parameter regularization [17] and simple data replay.
- The common joint training practice may be unnecessary for SSL to obtain a good representation model with streaming data. Instead, sequential SSL is performance-competitive but more time-efficient and storage-saving, well worth considering as common practice for self-supervised pre-training with streaming data.
- When sequentially trained with streaming data, representations of SSL models exhibits visibly less forgetting than those of SL model. We believe such good property of knowledge forgetting will inspire potential applications of SSL in continual learning tasks.

2 Problem Setting

For the illustration purpose, we adopt the prevailing SSL method, MoCo-v2 [18], to investigate the transfer learning performance of SSL with streaming data. See Appendix B on how to apply MoCo-v2 in sequential training and joint training, respectively.

Pre-training with streaming data. We design 4 types of streaming data to mimic practical data collection scenarios: instance incremental sequence, random class incremental sequence, distant class incremental sequence and domain incremental sequence.

We first consider ImageNet [3] as the streaming data and split it into 4 disjoint data chunks, which means the data sequence length is $B = 4$. For the **instance incremental sequence**, we randomly shuffle and split all samples into 4 IID parts. For the **random class incremental sequence**, we randomly split ImageNet into 4 disjoint data chunks with each chunk having 250 classes. For the **distant class incremental sequence**, inspired by [19], we split ImageNet into 4 class-even chunks according to WordNet Tree [20] while maximize the semantic dissimilarity across splits. In this case, the labels of data in different splits do not have common parent nodes under the 9-th level of taxonomy. Finally, to obtain a **domain incremental sequence**, we adopt a multi-domain dataset called DomainNet [21] for pre-training. Following [21], we evenly choose samples from four domains

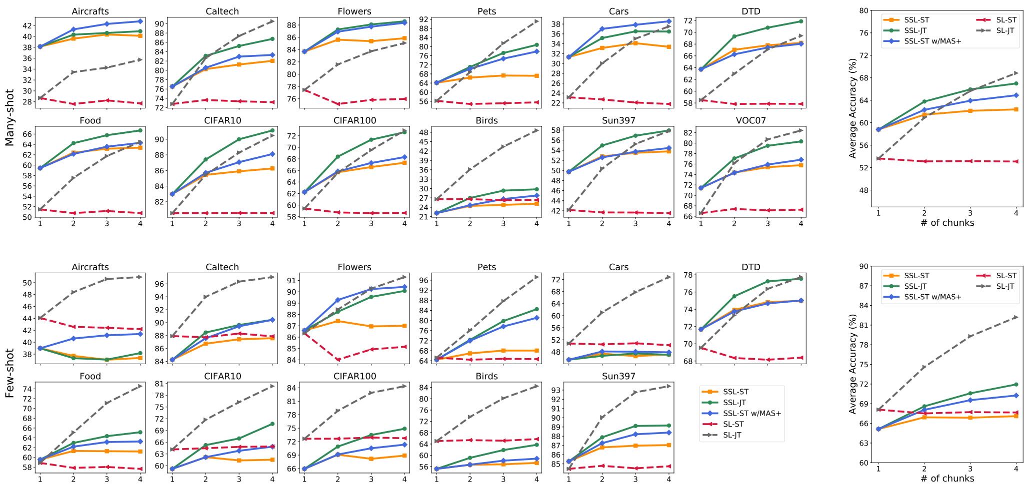


Figure 2: Comparisons of transfer learning performance among models of SSL-ST, SSL-ST w/MAS, SSL-ST w/MAS+, SSL-JT, SL-ST, and SL-JT, when pre-trained with the **distant class incremental sequence**. On the right, we show the average accuracy for two downstream tasks across all the datasets, respectively.

including Real, Clipart, Sketch and Painting. The illustration of above 4 types of streaming data is shown in Figure 1. See Appendix C for more introductions.

Transferring to downstream tasks. To thoroughly evaluate the transfer learning ability of SSL pre-trained models with streaming data, we evaluate them on 3 typical downstream tasks, including linear evaluation, few-shot classification and detection. Following [22], we consider 12 diverse image classification datasets including Food-101 [23], CIFAR10 [24], CIFAR100 [24], Birdsnap [25], SUN397 [26], Standard Cars [27], FGVC Aircraft [28], VOC2007 [16], DTD [29], Oxford-IIIT Pets [30], Caltech-101 [31] and Oxford 102 Flowers [32]. On these datasets, we evaluate the pre-trained models via the linear probe and few-shot classification (except VOC2007). Both classification protocols are the same as [14]. In addition, we evaluate the pre-trained models on the PASCAL VOC detection task, following the same transfer protocol of MoCo [7]. We mainly make comparisons among the following training models: sequentially trained SSL models (SSL-ST), jointly trained SSL models (SSL-JT), sequentially trained SSL models using MAS [17] (SSL-ST w/MAS), sequentially trained SSL models using MAS and replay of 10% old data (SSL-ST w/MAS+), sequentially trained SL models (SL-ST), and jointly trained SL models (SL-JT).

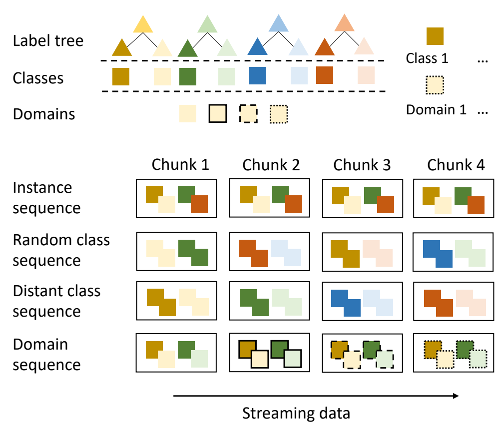


Figure 1: Illustration of 4 different types of streaming data used for pre-training. Each color means one class, where similar colors refer to similar semantics in the label semantic tree. Border types mean domain styles.

3 Sequential SSL: Resource-efficient and Performance-competitive

In Table 1, we compare the required training time and storage between sequentially trained models and the model of joint training (JT). As shown in Table 1, JT is very time-consuming especially when data amount is large, while ST is able to save a large amount of time under sequential training scenarios. 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. In summary, sequential SSL pre-training is much more time-efficient and storage-saving than JT, especially when the data amount is large and

Table 1: The comparison of SSL pre-training methods in terms of the resource efficiency. 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 after each data chunk. Note that all the following statistics are recorded under the same hardware environment. The lower value means the better efficiency.

Time (Storage) / Chunk	2	3	4
SSL-ST	**17 (35)**	**34 (35)**	**51 (35)**
SSL-ST w/MAS	18 (35)	36 (35)	54 (35)
SSL-ST w/MAS+	22 (39)	46 (42)	72 (46)
SSL-JT	31 (70)	78 (105)	145 (140)

Table 2: The comparison of pre-training methods in terms of the transfer performance gap between ST and JT models. We report the averaged accuracy gaps of linear evaluation across 12 downstream datasets. 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**

grows quickly. Such a result suggests that sequential SSL is a more favorable choice for real-world applications, where data come in sequentially and grow daily.

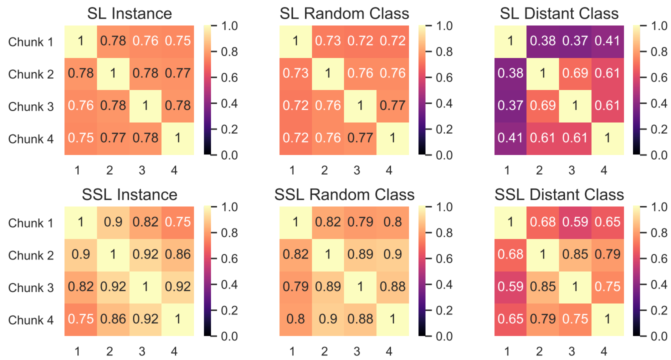
In Figure 2, we make comparisons of the transfer learning performance among different models on the most challenging distant class incremental sequence. We find SSL have much smaller accuracy gap between ST models and JT models, compared with SL. Besides, simple yet efficient continual learning methods bring visible improvement over sequential SSL. In Table 2, we show the mean accuracy gap between ST model and the corresponding JT model under the linear evaluation protocol. On the easiest instance incremental sequence, SL shows an obvious accuracy gap while the gap of SSL is negligible. On the medium-hard random class incremental sequence, SL exhibits a much larger accuracy gap while SSL still keeps the negligible accuracy gap. On the hard distant class incremental sequence, SL shows a much larger accuracy gap and SSL has an obvious accuracy gap. But such an accuracy gap of SSL can be effectively mitigated with simple continual learning methods like MAS or data replay. Generally, when learned with various streaming data, sequential SSL can achieve comparable transfer performance to joint SSL, with the help of continual learning methods.

4 SSL Models Forget Less Than SL Models

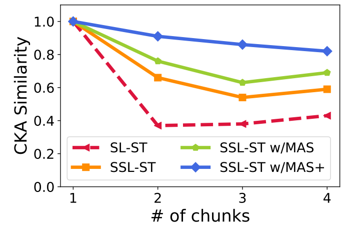
To further understand why SSL has smaller accuracy gaps between sequential models and joint models, we analyze features of sequentially trained models via Centered Kernel Alignment (CKA) [33].

How do features forget in sequential training? We first study how learned features forget in sequential training via the CKA. Specifically, we randomly sample 5,000 images from the first data chunk for each streaming data. We use these samples and the sequentially trained models for CKA similarity analysis. We report CKA values under three sequential training settings in Figure 3(a). Each value in Figure 3 (a) is obtained by inputting these samples to two different models and computing the CKA similarity value between the output two features. We find SSL always has higher feature similarity than SL. This suggests that features of SSL forget less than features of SL during sequential training. Moreover, for the distant class incremental sequence, equipped with the MAS regularization and data replay, sequential SSL features are almost the same as the initial features with a CKA similarity over 0.9. Such a result shows that with these two simple techniques, the model continually trained by SSL exhibits almost no forgetting of previous knowledge in sequential training.

Sequential training v.s. Joint training. We then evaluate CKA similarity between features from the jointly trained model and features from the sequentially trained model for each data chunk. For example, as shown in Figure 3(b), at the second data chunk, we compute the CKA similarity between features of the sampled data from the model jointly trained with the first two data chunks and features from the model sequentially trained with the second data chunk. The corresponding CKA similarity value is 0.4, which means for SL, the difference between joint training and sequential training is very large. In contrast, SSL has a higher similarity between sequential learning and joint training. Particularly, with MAS and data replay, the model trained by sequential SSL extracts nearly the same features as the jointly trained model. This illustrates that, even for the challenging distant class incremental sequence, one can also replace the joint training by sequential SSL pre-training.



(a) CKA scores across ST models trained after streaming data chunks.



(b) CKA similarity scores between ST models and corresponding JT models at the point of each data chunk on the distant class incremental sequence.

Figure 3: CKA similarity analysis of representations learned from different methods w.r.t. each sequential chunk. Given a set of images, figures (a-c) show the feature similarity between the model pre-trained with the first data chunk and the model sequentially trained with the current data chunk under 3 sequential training settings, respectively. Moreover, figure (d) shows the feature similarity between the jointly trained model and the sequentially trained model with different methods w.r.t. each data chunk on the distant class incremental sequence.

Feature reconstruction. Similar to [34], in Figure 4, we visualize feature reconstructions of both sequential SL models and sequential SSL models using deep image prior (DIP) [35]. To be specific, we choose 4 images in the first data chunk of the distant class incremental sequence and visualize features of 4 sequential sequentially learned models of SSL and SL, respectively. As is shown in Figure 4 in the Appendix, In the sequential training process, features of SSL model can always perfectly reconstruct the main information in original images, while features of SL models would lose lots of detailed information, which indicates SSL is much better at countering the knowledge forgetting in sequential training. Considering the evolving CKA similarity shown in Figure 3(a), the good property of knowledge forgetting does not means SSL models stop learning new knowledge, but it indicates that SSL does well in learning new knowledge while keeping old knowledge.

5 Discussions

In this paper, we have conducted the first thorough empirical evaluation to investigate how well self-supervised learning (SSL) performs under sequential training scenarios. Our results show two main findings as follows: 1). **Joint training is not necessary for SSL, while sequential training with suitable strategies is a good alternative.** In the scenarios where distribution shifts within streaming data are mild (e.g., instance and random class incremental sequence), it is more favorable to directly conduct sequential SSL training that is far more efficient with negligible performance loss. On the other hand, if distribution shifts between streaming data are large, sequential SSL training with MAS+ is well worth considering. 2). **Sequential self-supervised pre-training shows a better capability of overcoming catastrophic forgetting than supervised pre-training**. One reason is that the features learned by contrastive SSL have been shown to be uniformly distributed over the feature space [36], which means the learned representations shift less during sequential training, as demonstrated by Section 4. In addition, features learned by the self-supervised task of instance discrimination are able to keep more visual information than the features learned by supervised pre-training [34], which weakens the effect of knowledge forgetting during sequential training.

Future directions. We first call for more attention to sequential self-supervised learning for understanding its underlying theories and devising better approaches. Also, we recommend considering sequential self-supervised training as a more efficient representation learning practice for real-world applications. Moreover, we will further investigate different self-supervised learning methods on various network architectures under sequential pre-training.

References

[1] Donahue, J., Y. Jia, O. Vinyals, et al. Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learning*. 2014.

[2] Zeiler, M. D., R. Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*. 2014.

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

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

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

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

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

[8] 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.

[9] 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.

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

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

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

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

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

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

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

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

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

[19] Huh, M., P. Agrawal, A. A. Efros. What makes imagenet good for transfer learning? *arXiv*, 2016.

[20] Miller, G. A. *WordNet: An electronic lexical database*. MIT press, 1998.

[21] Saito, K., D. Kim, S. Sclaroff, et al. Semi-supervised domain adaptation via minimax entropy. In *Proceedings of the IEEE International Conference on Computer Vision*. 2019.

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

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

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

[25] 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.

[26] 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.

[27] 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.

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

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

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

[31] 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.

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

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

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

[35] Ulyanov, D., A. Vedaldi, V. Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454. 2018.

[36] Wang, T., P. Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*. 2020.

[37] Larsson, G., M. Maire, G. Shakhnarovich. Learning representations for automatic colorization. In *European conference on computer vision*. 2016.

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

[39] Tian, Y., D. Krishnan, P. Isola. Contrastive multiview coding. *arXiv*, 2019.

[40] Misra, I., L. v. d. Maaten. Self-supervised learning of pretext-invariant representations. In *Computer Vision and Pattern Recognition*. 2020.

[41] Tian, Y., C. Sun, B. Poole, et al. What makes for good views for contrastive learning. In *International Conference on Learning Representations*. 2020.

[42] Xie, E., J. Ding, W. Wang, et al. Detco: Unsupervised contrastive learning for object detection. *arXiv*, 2021.

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

[44] Thomee, B., D. A. Shamma, G. Friedland, et al. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 2016.

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

[46] 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.

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

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

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

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

[51] 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.

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

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

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

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

[56] Han, J., X. Liang, H. Xu, et al. SODA10M: towards large-scale object detection benchmark for autonomous driving. *CoRR*, abs/2106.11118, 2021.

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

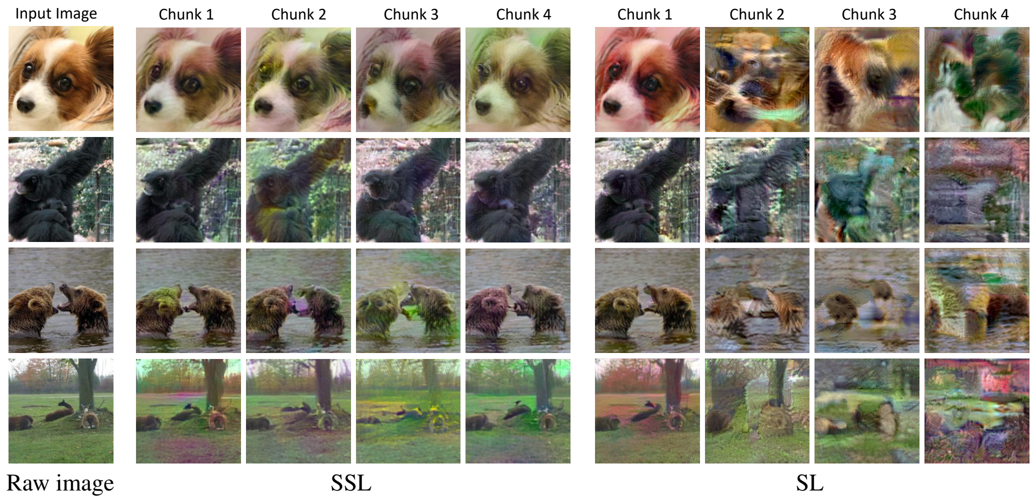


Figure 4: Features reconstruction of both SL-ST and SSL-ST models in sequential training.

A Related Work

Self-supervised learning (SSL). SSL learns useful representations by solving various pretext tasks using supervisions generated from unlabeled training data, e.g., predicting rotations [5], solving jigsaw puzzles [4], predicting colorization [37], predicting cluster assignments [6] and solving instance discrimination [38, 22, 7, 8]. Recently, instance discrimination has become the most popular pre-text task for SSL, which motivates various contrastive SSL methods [39, 22, 7, 18, 40–42]. Contrastive SSL usually leverages a contrastive loss [43] to maximize the similarity of features from the same image and minimize the similarity of features from different images, where massive pairwise comparisons among different images are required. Many strategies are proposed to improve contrastive learning, including maintaining a memory bank of all features [38], using a large chunk size [22] and using momentum encoders [7, 18]. To further improve the representation model, recent studies of SSL have proposed to pre-train a representation model with increasingly large datasets such as YFCC 100M [44] or even Instagram 1B [45]. Despite the desirable transfer performance [7], in realistic scenarios, it is not easy to acquire massive data at a time and unlabeled data are mostly streaming. However, how to efficiently and effectively perform SSL with streaming data remains open, which motivates our study.

Continual learning. Existing studies of continual learning (CL) [46] 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 [17, 11, 47]. Replay-based CL methods overcome forgetting by saving samples of previous tasks in a replay buffer [48–51] 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 [52, 53]. Although works [54, 55] explore continual learning for some specific unsupervised tasks, few have studied the transfer performance of sequential self-supervised representation learning.

B Training of MoCo-v2 on streaming data

Formally, we consider the unlabeled dataset $D = \bigcup_{b=1}^{B} D_b$ with B chunks of data, where $D_b = \bigcup\{(x_i)\}$ represents the b-th data chunk in the stream. Without loss of generality, we assume that these data come from C classes although the labels are unavailable for model training.

Sequential training. In sequential training, data samples used for model training are divided into disjoint chunks, i.e., $D = \bigcup_{b=1}^{B} D_b$, where B is the total number of data chunks. In sequential self-supervised pre-training, both the representation network f_θ and the projection head f_w are continually trained. Specifically, the b-th time sequential training starts from the pre-trained network including f_θ^{b-1} and f_w^{b-1}, only involving samples of data chunk D_b in the model training. When the b-th time training finishes, only f_θ^b and f_w^b are saved for sequential learning with next independent data chunk. Given the same training epoch, sequential training is much more efficient than joint training as only new data are used for the continual pre-training at each chunk. Continual learning techniques including data replay and unsupervised parameter regularization methods e.g. Memory Aware Synapses (MAS) [17] may be used to further improve the performance of sequential training.

Joint training. In joint training, all available data are randomly shuffled to jointly train a model until convergence. Joint training is the common practice in SSL [8, 9]. As for pre-training with streaming data, each data chunk has a joint training result. At b-th data chunk, joint training requires all previously seen data chunks, i.e., $\{D_1, ..., D_{b-1}, D_b\}$, for jointly training a representation network f_θ from scratch. When the data sequence is long and each data chunk has a large amount of data, joint training is very storage-heavy and time-consuming.

C Introduction on Streaming Data

Instance incremental sequence Here we consider the sequential training of *new instance*. It assumes that streaming data are independent and identically distributed (IID), where each sequence chunk contains all the C classes but new instances come in sequentially. This kind of data stream is often encountered when samples are sequentially collected under the same conditions.

Random class incremental sequence Similar to the classic class incremental learning, we then consider the random class incremental sequence for representation learning. In a typical example, each chunk of image data is obtained by a random key word from the Internet using search engines.

Distant class incremental sequence Extending from the random class incremental sequence, we intentionally split the data classes w.r.t. the semantic similarity of classes to enlarge the data distribution gaps among chunks. In particular, images in the same data chunk share similar semantics while images from different data chunks are semantically dissimilar. This setting is designed to evaluate how well self-supervised pre-training performs on streaming data with large data distribution shifts.

Domain incremental sequence Complementary to the above settings, we also consider a domain-incremental setting, where data chunks in the stream come from different image domains. For example, the first data chunk is realistic photos while the second data chunk are paintings. Such a data sequence mimics streaming data with domain distribution shifts, where different data chunks in the sequence may share the same classes or similar semantics. A typical example can be found in [56], when autonomous driving data with similar semantics are collected under different weathers or light conditions.

D Results of Object Detection

The results of object detection are shown in Figure 8.

E Results of BYOL

The results of BYOL on the distant class incremental sequence are shown in Figure 9.

F Results on Other Streaming Data

The results of the instance incremental data is shown in Figure 5, the results of random class incremental data is shown in Figure 6, and the results of the domain incremental sequence is shown in Figure 7.

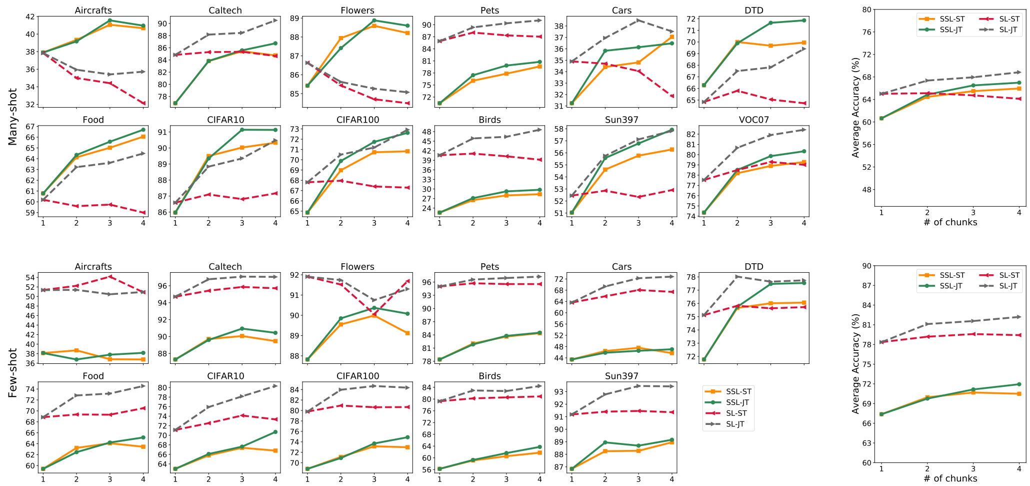


Figure 5: Comparisons of transfer performance between sequential training (ST) and joint training (JT) for self-supervised pre-training with the **instance incremental sequence**. ST shows similar transfer performance compared to JT on 12 downstream tasks under both many-shot and few-shot classification yet with much higher efficiency. On the right, we show the average performance for the two downstream tasks across all the datasets together with results of joint supervised pre-training (SL-ST) and sequential supervised pre-training (SL-ST).

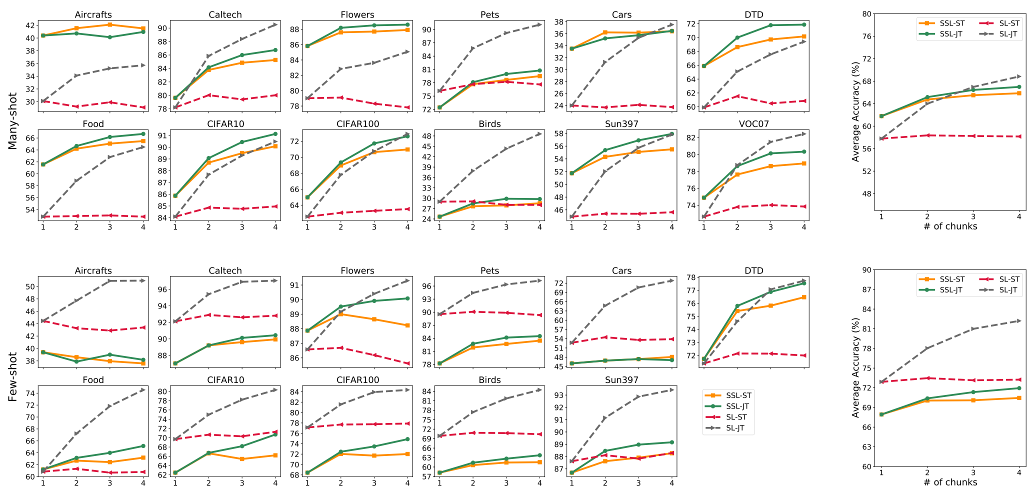


Figure 6: Comparison of transfer performance between sequential training (ST) and joint training (JT) for self-supervised pre-training with the **random class incremental sequence**. On the right, we show the average performance for the two downstream tasks across all the datasets together with results of joint supervised pre-training (SL-ST) and sequential supervised pre-training (SL-ST).

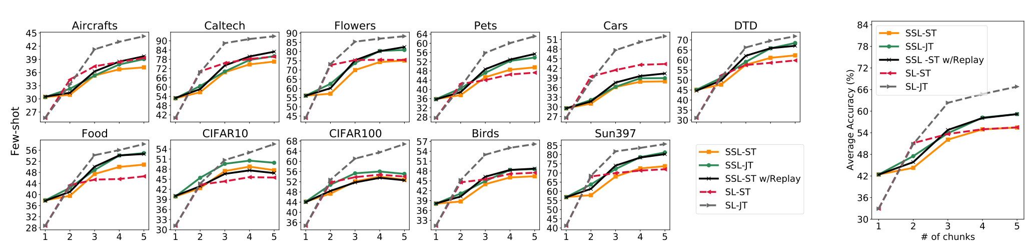
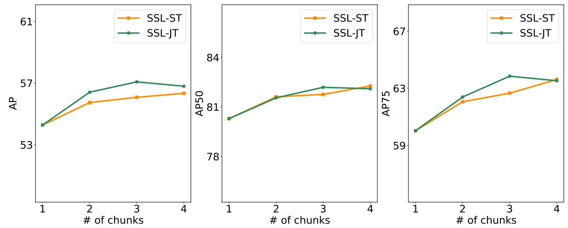
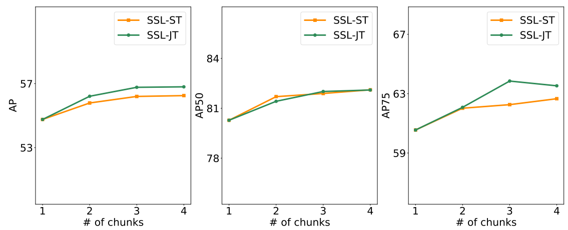


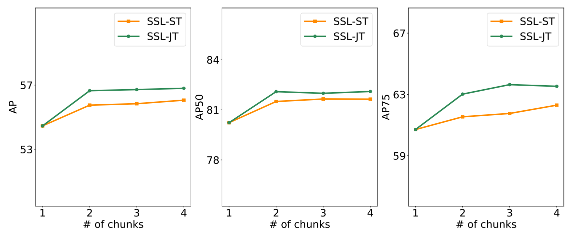
Figure 7: The average few-shot transfer performance across five sequences between sequential training (ST) and joint training (JT) for self-supervised pre-training with domain incremental streaming data. On the right, we show the average performance across all the datasets.



(a) Instance incremental sequence



(b) Random class incremental sequence



(c) Distant class incremental sequence

Figure 8: Comparisons of transfer performance between sequential training (ST) and joint training (JT) for self-supervised pre-training on the detection task.

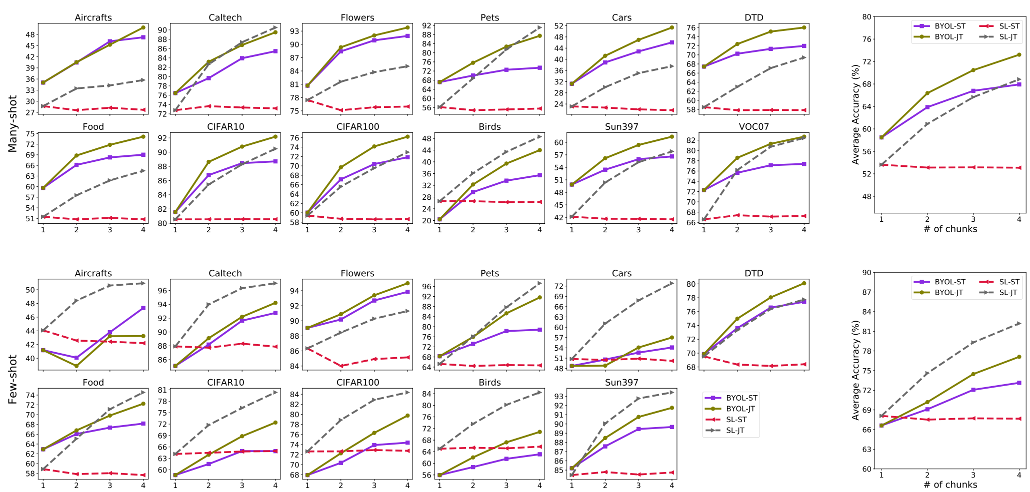


Figure 9: Comparisons of transfer performance between sequential training (ST) and joint training (JT) for BYOL with the **Distant class incremental sequence**. On the right, we show the average performance across all the datasets.