EVOLVE: Enhancing Unsupervised Continual Learning with Multiple Experts Supplementary Material

In the supplementary material, we include more details on the following aspects:

- In Section 1, we discuss the connections and differences between EVOLVE and a typical online optimization problem.
- In Section 2, we list the implementation details of EVOLVE, the self-supervised learning baselines and the continual learning baselines.
- In Section 3, we provide details on data setup and how to construct data streams from each dataset.
- In Section 4, we show that EVOLVE is capable of surpassing the best expert if trained with multiple epochs. With both self-supervised learning and expert-guided learning, EVOLVE is not bounded by the performance of experts.
- In Section 5, we present additional accuracy results as a complement to the results in the main paper.
- In Section 6, we show additional visualization for dynamic weight update using Multiplicative Weight (MW) Update and EVOLVE.
- In Section 7, we conduct more sensitivity experiments on the hyperparameters λ and α.
- In Section 8, we analyze the time complexity of EVOLVE.

1. Connection to Online Optimization

The proposed expert aggregation and dynamic weight adjustment have close connections to the MW algorithm [1, 4] in online optimization. In particular, the proposed unsupervised continual learning with multiple experts can be rewritten as an online optimization problem [4]:

1. At timestamp t, given the sample $\mathbf{x}_t$, the task is to predict one augmentation $\mathbf{x}_t^A = t^A(\mathbf{x}_t)$ from augmentation $\mathbf{x}_t^B = t^B(\mathbf{x}_t)$, where $t^A(\cdot)$ and $t^B(\cdot)$ are two random augmentations.

2. Each expert e makes a prediction $q_e^t \in [0, 1]$. The loss incurred by the expert e can be calculated as $\ell_e^t = |1 - q_e^t|$.

3. Each expert has a weight w_e^t. The prediction of the learner is:
$$q^t = \frac{\sum_e w_e^t q_e^t}{\sum_e w_e^t} \tag{1}$$
which is the aggregated predictions of all experts, weighted by the corresponding weights. The loss incurred by the learner can be computed as $\ell^t = |1 - q^t|$.

4. Intuitively, if the expert e has a higher loss, the continual learner should decrease its weight. The weight update can be written as,
$$w_e^t = w_e^t + (1 - \alpha)(q_e^t - w_e^t) \tag{2}$$

5. At timestamp T, the cumulative regret of the continual learner with respect to the expert e can be written as,
$$\mathcal{L}_e = \sum_{t=1}^{T} (l^t - l_e^t) \tag{3}$$

It is important to recognize that our approach differs from online optimization in that our objective is not only to minimize regret with respect to the best expert. Rather, we seek to leverage the experts to guide the continual learning process that leads to improved representations, especially in scenarios where prior knowledge is limited. In contrast to the MW algorithm, our proposed method places greater emphasis on the current predictions of the experts. As a result, it exhibits greater responsiveness to dynamic shifts in data streams. For further comparative insights between the MW algorithm and our proposed update, refer to Fig. 2.

2. Implementation Details

2.1. Implementation of EVOLVE

The hyperparameters of EVOLVE across all datasets are summarized in Table 1. The values for learning rate lr, batch size b, memory buffer size m and batch size for sampled memory data $b_{\mathcal{M}}$ are the same as in [11]. For all methods in EVOLVE, we use SGD optimizer with a learning rate of 0.03, a momentum of 0.9 and a weight decay of $1e^{-4}$ as reported in [11]. The selection of hyperparameters α and λ is based on a validation set.

Table 1. Hyperparameters of EVOLVE in all datasets.

Param.	Meaning	Value
lr	Learning rate	0.03
b	Batch size for streaming data	128
m	Memory buffer size	256
$b_{\mathcal{M}}$	Batch size for sampled memory data	128
τ	Temperature for the confidence metric	0.1
α	Decay rate to update each expert's confidence	0.95
λ	Relative weight for SSL and expert aggregation loss	1.0

Data augmentation. The two-way augmentation is a pivotal element within the SSL backbones adopted in EVOLVE, a common practice in self-supervised learning [2,5,6,9,19]. Recognizing the informative potential of augmentation, we highlight that we use the general augmentation techniques for image datasets. Consistency is maintained as the identical augmentation procedure is implemented across all methods for a given dataset.

During the training phase, our data augmentation procedure normalizes the data using mean and variance. All data are resized to 32×32.

- For CIFAR-10, we apply a random scaling 0.2-1, a random horizontal flip, a random color jitter of brightness 0.6-1.4, contrast 0.6-1.4, saturation 0.6-1.4, hue 0.9-1.1, and a random gray scale with $p = 0.2$.
- For TinyImageNet, we apply a random scaling 0.08-1 with random aspect ratio 0.75-1.33 and bicubic interpolation.
- For both CORe50 and Stream-51, we apply a random horizontal flip.

During the evaluation phase, we only normalize and resize the data but do not use any other augmentations.

2.2. SSL Baseline Implementation

In Section 4 of the main paper, we experiment the following state-of-the-art self-supervised learning baselines which are adapted from their official implementations:

- **SimCLR** [5] uses a Siamese structure and enhances the similarity between augmented pairs of the same sample. We set temperature $\tau = 0.1$.
- **SimSiam** [6] uses a simple Siamese structure with a stop-gradient operation to avoid collapsing.
- **BYOL** [9] creates a target network via moving average and learns without collapsing by predicting the representation of the target network. The moving average decay for target network updates is set to 0.99.
- **BarlowTwins** [19] improves representation learning by aligning the cross-correlation matrix (between the two branches of the Siamese network) with the identity matrix. The coefficient for off-diagonal elements is $5e^{-3}$ as in the original implementation.
- **VICReg** [2] further improves BarlowTwins by adding extra invariance and variance loss terms. The coefficients for the invariance, variance and covariance terms are 25.0, 25.0 and 1.0 respectively.

2.3. Implementation of Unsupervised Continual Learning Baselines

The following unsupervised continual learning baselines are used to compare with SCALE:

- **PNN** [16]: Progressive Neural Network gradually expands the network architecture.
- **SI** [20]: Synaptic Intelligence performs online per-synapse consolidation as a typical regularization technique.
- **DER** [3]: Dark Experience Replay retains existing knowledge by matching the network logits across a sequence of tasks.
- **CaSSLe** [8] proposes a general framework that extracts the best possible representations invariant to task shifts.
- **LUMP** [11] interpolates the current batch with the memory samples to alleviate catastrophic forgetting.

We did not compare with STAM because their architecture needs dataset-specific tuning and the original paper did not consider large image-based or video-based datasets [17].

More implementation details are grouped and summarized as follows:

- **PNN**, **SI**, **DER**, **LUMP** are adapted from the official framework in [11] using their default hyperparameters[1]. PNN, SI and DER are originally designed for supervised lifelong learning but are adapted to ULL tasks as described in the paper. For DER and LUMP, we use the identical memory size m and batch size for memory samples $b_{\mathcal{M}}$ as EVOLVE.
- We use a modified version of **CaSSLe** based on the original implementation[2]. Specifically, we remove task labels and the model is used to predict the representations generated by a frozen model from the previous time stamp, from the current representations. The predictor network is implemented as a two-layer MLP with 2048 hidden neurons and ReLU activation as the predictor network to predict the past representations from the current ones.

[1] https://github.com/divyam3897/UCL
[2] https://github.com/DonkeyShot21/cassle

3. Data Stream Construction

In this section, we detail our dataset setup and data stream construction. For each of the four data sets, we test all the methods on the sequential class-incremental streams **(Seq)** and the sequential imbalanced class-incremental streams **(Seq-imb)**.

- **CIFAR-10 [12] and TinyImageNet [7].** To construct the Seq stream, we sample 5000 and 500 samples from each class in CIFAR-10 (10 classes in total) and TinyImageNet (100 classes in total), then feed them class by class to the model. The order of samples inside each class is random. The setup of the Seq-imb stream is almost the same, except that for each class, we randomly sample a subset that has more than half of the total samples in that class. Specifically, suppose that there are U samples in that class. We first uniformly sample an integer $V \in [0.5U, U]$, then randomly select V samples from that class.

  In CIFAR-10, we split the original test data set, using 9,000 samples for testing and 1000 samples for validation. In TinyImageNet, the total number of samples for testing and validation is 4500 and 500, respectively.

- **CORe50 [10].** CORe50 is a video dataset for classifying 50 domestic objects held by the operator. The dataset has been collected in different environments with different backgrounds and lighting, which are formulated to 11 distinct sessions (8 indoor and 3 outdoor). For each session and for each object class, a 15 seconds video (at 20 fps) has been recorded with a Kinect 2.0 sensor delivering 300 RGB-D frames.

  We generate the Seq streams following the most challenging New Instances and Classes (NIC) sequence setting in the original paper. NIC introduces both new classes and new instances (under different sessions) in subsequent training batches. A good model is expected to consolidate its knowledge about the known classes and to learn the new ones under the shift of environments. CORe50 has a balanced distribution among different classes, and we sample an equal amount of 960 samples from each class to construct the training Seq streams, and 180 samples per class for evaluation.

  The Seq-imb streams for CORe50 follow exactly the same temporal order as the Seq streams, while we randomly sample a subset from the sequence of each class. The method we employ to determine the subset size aligns with that of CIFAR-10 and TinyImageNet.

  For evaluation, we randomly sample separate subsets of 9,000 and 1000 images from the original test dataset, then use them for testing and validation.

- **Stream-51 [15].** Stream-51 is a recently introduced video dataset tailored for continual learning, encompassing temporally correlated images across 51 distinct object categories. Our focus is on the Seq-imb streams within Stream-51, given the inherent class imbalance in the dataset. Following the dataset's temporal order, we stochastically sample 40% of the training data, resulting in 60,000 training samples. Evaluation employs the original Stream-51 test dataset, which encompasses both familiar and novel categories in an open set classification context. Specifically, we engage with all images pertaining to the known 51 categories within the test dataset. Upon partitioning, we allocate 2400 samples for testing and 500 samples for validation.

4. Comparison with the Best Expert

One important question is whether, guided by pretrained expert models, EVOLVE can achieve comparable or even superior performance compared to the best expert. The answer is affirmative. In this experiment, we assess multiple epochs over the data at each timestamp, resembling the context of **offline unsupervised continual learning** [13, 14, 18]. Fig. 1 illustrates EVOLVE's kNN training accuracy over 10 epochs across all four datasets in the iid stream. This is compared with the kNN accuracy of the expert models, indicated by dashed horizontal lines.

Remarkably, EVOLVE outperforms the best expert in CIFAR-10 and CORe50, and closely approaches the performance of the best expert in TinyImageNet and Stream-51. Notably, while all experts employ sophisticated structures like ResNet-50 and the Swin Transformer, EVOLVE achieves this using the considerably smaller ResNet-18 model. EVOLVE has two components of self-supervised learning and expert-guided learning, allowing it to continuously improve beyond experts.

5. Complete Accuracy Results

As a complement to the accuracy results in the main paper, we present the rest accuracy results at the end of the supplementary material. In the paper, we focus on the sequential streams which aligns with real-world scenarios. Tab. 2-4 summarize the accuracy results on Seq CIFAR-10, TinyImageNet and CORe50 respectively. Note, that Stream-51 comes with imbalanced samples thus is evaluated only for the Seq-imb streams. Tab. 5 shows the accuracy results on Seq-imb CIFAR-10.

For each setting, SSL acts as a backbone while additional continual learning techniques may improve or degrade the performance of SSL. In all settings, EVOLVE achieves the best performance and the final accuracy is of similar level regardless of which SSL is used. Such observation supports

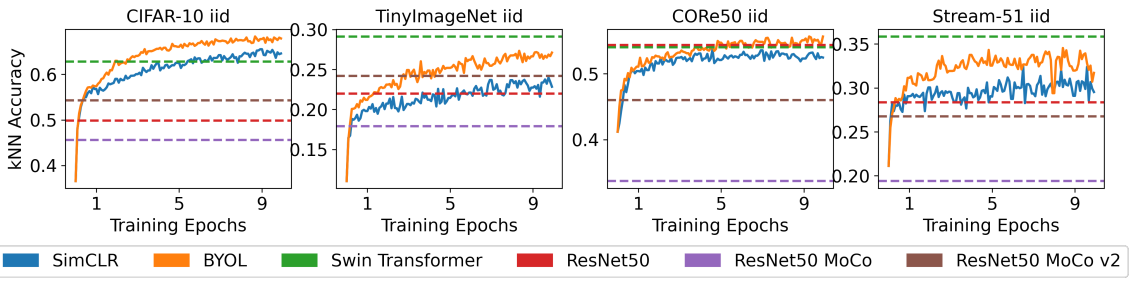

Figure 1. EVOLVE can achieve comparable or better accuracy compared with the experts with a much smaller model size. The kNN accuracy during training by EVOLVE, compared to the kNN accuracy of the frozen experts. We select the best two SSL, SimCLR and BYOL, as backbone and experiment under the iid streaming data.

the strong learning capability of EVOLVE in scenarios with limited prior knowledge.

6. More Weight Updates Results

In Fig. 2, a comprehensive visualization across all datasets featuring sequential streams is presented. For each configuration, we depict the expert weights during training, contrasting the Multiplicative Weights (MW) update method with the proposed moving average-based update method in EVOLVE. The weight w_e^t of each expert is updated based on the current and historical confidence q_e^t, but using different mechanisms. A higher q_e^t (and subsequently, a higher w_e^t) means that the expert is more confident about the current input batch, implying a better quality of the expert in the current dataset.

It can be observed in Fig. 2 that different expert models have varied w_e^t on different datasets, which supports the motivation of EVOLVE which distills from a diverse set of experts for richer information. CIFAR-10 and CORe50 display one expert's consistent dominance during sequential training, whereas TinyImageNet and Stream-51 exhibit alternating confidence between two experts with incoming batches. In the latter case, MW cannot effectively capture this alternation due to accumulation. The proposed EVOLVE is able to reflect the alternation by maintaining a moving average of the past and latest confidence.

7. More Sensitivity Analysis

In Fig. 3, we present more sensitivity results of λ and α on various datasets using different underlying SSL. To remind the reader, we set λ based on the relative loss scale, and experiment $\lambda = \psi\frac{|\mathcal{L}_{SSL}|}{|\mathcal{L}_E|}$ with $\psi \in \{0.5, 0.75, 1.0, 1.25, 1.5\}$. We further evaluate EVOLVE using SimCLR on Seq-imb Stream-51 streams, and using BYOL on Seq-imb CORe50 streams. We show the average and standard deviation of the final kNN accuracy in 3 random trials.

We highlight that our selection of hyperparameters α and λ is based on a validation set, while the testing results in a specific scenario is jointly affected by the dataset, the

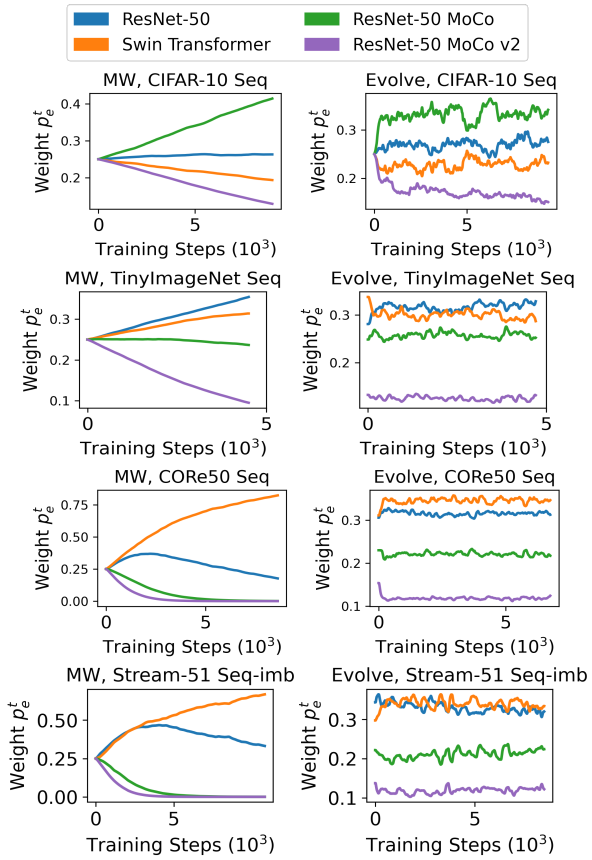

Figure 2. The proposed EVOLVE can better capture the dynamics of the data streams and the change of experts' confidence by maintaining a moving average of the past and latest confidence. **Left column**: results of the MW algorithm in different settings. **Right column**: results of the proposed moving average-based update.

streaming pattern, the quality of experts and the underlying SSL. $\psi = 1.5$ achieves the best kNN accuracy in the two testing scenarios in Fig. 3. We believe this can be attributed to the challenging nature of the Seq-imb video data streams, thus placing higher weight on $\mathcal{L}_E$ will lead to a better performance at the end of the single-pass continuum. In terms of α, $\alpha = 0.97$ produces the highest kNN accuracy on Seq-imb Stream-51 streams with SimCLR, while $\alpha = 0.95$

Table 2. Comparison of EVOLVE and unsupervised continual learning baselines on the **Seq CIFAR-10** streams. Bold and underlined values show the best and second best results on the basis of each SSL.

Method	kNN Accuracy(↑)					Linear Evaluation Accuracy(↑)				
	SimCLR	**BYOL**	**SimSiam**	**BarlowTwins**	**VICReg**	**SimCLR**	**BYOL**	**SimSiam**	**BarlowTwins**	**VICReg**
SSL	46.3±0.4	38.5±1.1	38.0±0.9	36.7±5.8	40.4±0.9	38.8±1.2	35.3±1.0	33.7±1.7	31.3±8.4	42.9±0.3
SI	38.6±0.5	28.3±1.2	33.9±1.4	34.5±0.3	40.5±0.7	28.6±1.2	22.0±0.9	28.9±2.8	26.3±0.5	33.4±1.1
PNN	37.9±0.6	27.8±0.6	33.9±1.4	34.7±0.2	40.9±0.4	27.7±1.1	21.0±0.9	28.4±3.7	26.3±1.6	32.6±1.1
DER	39.5±0.7	28.5±0.6	32.1±2.1	35.6±0.5	40.7±0.6	39.0±0.2	26.7±1.1	24.7±1.0	35.3±1.2	42.0±0.0
CaSSLe	28.3±1.3	37.3±0.2	37.3±0.8	29.5±3.8	40.2±0.6	13.2±2.6	35.8±0.7	35.6±1.0	21.0±0.9	42.2±0.5
LUMP	38.1±0.6	34.2±1.3	34.4±0.7	24.9±0.6	42.8±1.2	31.7±0.9	28.4±1.3	27.9±0.4	17.0±1.3	43.1±1.1
EVOLVE	**50.9±0.5**	**54.1±0.3**	**51.8±0.3**	**49.6±1.4**	**52.5±0.3**	**53.2±0.4**	**55.7±0.3**	**51.3±1.2**	**51.2±1.3**	**54.0±0.5**

Table 3. Comparison of EVOLVE and unsupervised continual learning baselines on the **Seq TinyImageNet** streams. Bold and underlined values show the best and second best results based on each SSL.

Method	kNN Accuracy(↑)					Linear Evaluation Accuracy(↑)				
	SimCLR	**BYOL**	**SimSiam**	**BarlowTwins**	**VICReg**	**SimCLR**	**BYOL**	**SimSiam**	**BarlowTwins**	**VICReg**
SSL	15.3±0.2	11.2±0.0	10.1±0.2	7.9±0.4	14.5±0.4	11.0±0.3	6.8±0.5	4.5±0.5	3.5±0.4	16.6±0.2
SI	13.4±0.5	9.6±0.2	9.4±0.7	8.2±1.2	13.1±0.2	9.4±0.0	8.2±1.0	5.9±0.0	4.7±1.0	11.7±0.8
PNN	13.7±0.6	10.2±0.1	9.9±1.1	8.1±0.9	13.1±0.3	11.7±0.0	11.9±0.8	5.7±1.7	5.0±0.0	11.5±0.0
DER	13.4±0.3	8.9±0.3	9.7±0.6	8.0±0.5	12.9±0.2	12.6±0.0	12.7±1.0	6.1±0.0	4.4±0.0	12.6±0.0
CaSSLe	8.6±0.2	10.8±0.2	10.6±0.2	7.2±0.5	14.8±0.5	3.2±0.9	8.2±2.5	5.0±0.3	1.9±0.3	16.6±0.1
LUMP	13.1±0.3	8.9±0.2	9.1±0.6	7.0±0.2	14.5±0.2	11.3±0.7	12.5±0.3	8.2±0.8	6.1±0.9	15.9±0.7
EVOLVE	**18.6±0.8**	**20.2±0.5**	**18.3±0.2**	**12.7±2.9**	**19.8±0.6**	**18.7±3.6**	**16.1±0.7**	**19.6±0.6**	**11.3±5.2**	**24.5±0.4**

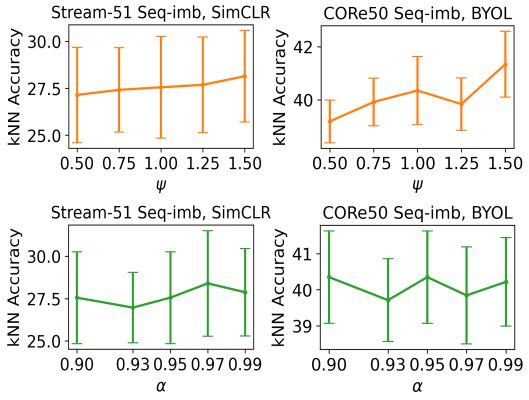

Figure 3. More sensitivity experiments of various hyperparameters in EVOLVE.

achieves the best result on Seq-imb CORe50 streams with BYOL.

8. Time Complexity Analysis

EVOLVE is efficient during both inference and training. During inference, all experts are discarded, and only the continual learner is used for inference. For training, we improve efficiency by reusing computation, and the time complexity of EVOLVE is analyzed below.

As explained in the main paper, EVOLVE has two loss terms: the self-supervised learning loss $\mathcal{L}_{SSL}$ and the expert aggregation loss $\mathcal{L}_E$. The time complexity of the SSL side is determined by the SSL method. One major novelty of EVOLVE lies in the expert aggregation loss, which takes the following steps to compute:

(1) Given the stacked augmented samples $\mathbf{x}_t$, EVOLVE first passes the samples into all expert models and the continual learner.

(2) For each expert model e that generates the representations $\mathbf{h}_e$, EVOLVE computes the kernel matricx $\mathbf{K}_e^t$. $\mathbf{L}^t$ is the kernel matrix computed on the local client by the continual learner, and shared with the cloud.

(3) EVOLVE computes the HSIC score for each expert model as well as the continual learner and obtains $\text{HSIC}(\mathbf{K}_e^t, \mathbf{L}_e^t)$, $\text{HSIC}(\mathbf{K}^t, \mathbf{L}^t)$. To remind the readers, $\text{HSIC}(\mathbf{K}, \mathbf{L}) = \frac{1}{(n-1)^2}\text{Tr}(\mathbf{KHLH})$, where $\mathbf{H} = \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T$ is the centering matrix.

(4) Last but not least, EVOLVE computes the dynamic weight p_e^t associated with each expert based on the confidence metric q_e^t. Now $\mathcal{L}_E$ can be calculated as Eq.(2) in the main paper.

We focus on steps (2), (3), (4) in our complexity analysis for EVOLVE.

For simplicity, in the following analysis, we use $n = b + b_{\mathcal{M}}$ to denote the batch size of the combined data stream and memory samples. In step (2), the kernel matrix is computed as the pairwise dot product between $\mathbf{h}_e$ and $\mathbf{h}_e^\top$, thus the complexity is $O(n^2)$. In step (3), it takes $O(n^2)$ to compute $\mathbf{KH}$ and $\mathbf{LH}$ respectively, and the trace operation is $O(n)$. For step (4), Eq.(3) in the main paper lists the computation of q_e^t. Note that $\mathbf{h}_{e,i}^A \cdot \mathbf{h}_{e,k}^A$ in the nominator is exactly $\mathbf{K}_{e,ik}^t$, which has already been computed in step (2). The time complexity of q_e^t is $O(n^2)$ which is driven by $\mathbf{h}_{e,i}^A \cdot \mathbf{h}_{e,k}^B$. In summary, computing $\mathcal{L}_E$ in EVOLVE consumes $O(En^2)$ time given E pretrained experts. As $E << n$ in most cases,

Table 4. Comparison of EVOLVE and unsupervised continual learning baselines on the **Seq CORe50** streams. Bold and underlined values show the best and second best results based on each SSL.

Method	kNN Accuracy(↑)					Linear Evaluation Accuracy(↑)				
	SimCLR	**BYOL**	**SimSiam**	**BarlowTwins**	**VICReg**	**SimCLR**	**BYOL**	**SimSiam**	**BarlowTwins**	**VICReg**
SSL	23.5±0.8	25.8±0.4	20.2±0.9	31.3±0.0	23.4±0.8	13.3±1.4	14.4±1.7	9.7±1.8	26.4±1.2	18.0±0.9
SI	15.5±1.9	19.0±1.1	19.4±2.3	14.3±1.0	25.5±1.1	8.9±0.8	14.7±4.0	6.3±0.5	12.6±1.7	9.8±0.9
PNN	15.6±0.9	19.9±1.9	15.7±0.4	14.1±1.1	25.2±1.4	8.6±1.4	14.9±5.0	8.7±3.3	7.1±0.1	9.5±0.6
DER	21.3±1.6	22.8±0.7	21.1±1.8	16.2±1.1	25.6±0.7	14.1±1.4	14.7±2.4	12.5±3.1	12.5±3.1	9.9±1.3
CaSSLe	12.6±2.0	26.0±0.7	20.9±1.3	19.8±1.2	24.1±0.8	4.6±1.6	17.3±0.7	9.4±2.6	6.9±1.1	18.2±1.4
LUMP	27.3±0.9	15.5±1.2	17.4±0.8	29.8±0.5	21.0±0.5	18.8±1.4	9.8±2.0	4.1±0.3	7.8±1.2	28.4±0.4
EVOLVE	**44.1±1.4**	**43.6±1.3**	**45.2±0.2**	**43.7±2.0**	**40.3±0.9**	**53.0±1.4**	**57.3±0.5**	**54.9±0.3**	**46.8±4.6**	**39.4±1.1**

Table 5. Comparison of EVOLVE and unsupervised continual learning baselines on the **Seq-imb CIFAR-10** streams. Bold and underlined values show the best and second best results on the basis of each SSL.

Method	kNN Accuracy(↑)					Linear Evaluation Accuracy(↑)				
	SimCLR	**BYOL**	**SimSiam**	**BarlowTwins**	**VICReg**	**SimCLR**	**BYOL**	**SimSiam**	**BarlowTwins**	**VICReg**
SSL	13.3±0.4	17.2±1.8	12.2±0.2	18.0±0.9	14.8±0.7	36.8±0.3	31.7±1.3	31.9±2.8	28.8±7.1	43.6±0.1
SI	38.0±0.6	29.3±1.1	34.1±0.8	32.5±0.3	39.8±0.9	28.2±0.5	22.6±2.2	29.2±0.7	24.5±1.3	31.9±1.0
PNN	38.1±0.7	28.2±0.4	33.5±1.7	33.4±0.8	40.2±0.8	27.7±1.1	21.0±0.9	28.4±3.7	26.3±1.6	32.6±1.1
DER	38.9±0.8	29.4±1.1	32.2±0.8	34.5±0.3	38.6±0.5	39.0±0.2	26.7±1.1	24.7±1.0	35.3±1.2	42.0±0.0
CaSSLe	30.7±2.5	37.2±0.8	35.9±0.2	30.4±3.3	41.6±0.4	12.9±1.7	36.6±2.2	33.2±1.7	25.5±7.2	43.3±0.6
LUMP	36.2±0.8	32.5±0.7	33.3±1.0	26.8±0.9	41.0±1.4	31.7±0.9	28.4±1.3	27.9±0.4	17.0±1.3	43.1±1.1
EVOLVE	**50.8±0.1**	**52.9±0.9**	**50.2±0.6**	**49.0±1.2**	**51.9±0.2**	**52.4±0.9**	**53.8±0.9**	**47.9±1.5**	**51.2±0.9**	**53.0±0.5**

we conclude that EVOLVE takes quadratic time as a function of the batch size n.

We emphasize that the $O(n^2)$ complexity to compute $\mathcal{L}_E$ is comparable with $\mathcal{L}_{SSL}$ for certain SSL methods. For BYOL and SimSiam, since they both employ MSE-based losses between the two views, it takes $O(n)$ to compute. Nevertheless, the time complexity for SimCLR, BarlowTwins and VICReg are $O(n^2)$: SimCLR computes the cosine similarity between all representations; BarlowTwins calculates the cross-correlation matrix; VICReg adds the MSE and variance losses on top of the co-variance loss from the cross-correlation matrix.

References

[1] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of computing*, 8(1):121–164, 2012. 1

[2] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021. 2

[3] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020. 2

[4] Nicolo Cesa-Bianchi, Yoav Freund, David Haussler, David P Helmbold, Robert E Schapire, and Manfred K Warmuth. How to use expert advice. *Journal of the ACM (JACM)*, 44(3):427–485, 1997. 1

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 2

[6] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021. 2

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 3

[8] Enrico Fini, Victor G Turrisi da Costa, Xavier Alameda-Pineda, Elisa Ricci, Karteek Alahari, and Julien Mairal. Self-supervised models are continual learners. *arXiv preprint arXiv:2112.04215*, 2021. 2

[9] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33:21271–21284, 2020. 2

[10] V Lomanco and Davide Maltoni. Core50: a new dataset and benchmark for continual object recognition. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 17–26, 2017. 3

[11] Divyam Madaan, Jaehong Yoon, Yuanchun Li, Yunxin Liu, and Sung Ju Hwang. Representational continuity for unsupervised continual learning. In *International Conference on Learning Representations*, 2022. 1, 2

[12] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 3

[13] Jason Ramapuram, Magda Gregorova, and Alexandros Kalousis. Lifelong generative modeling. *Neurocomputing*, 404:381–400, 2020. 3

[14] Dushyant Rao, Francesco Visin, Andrei A Rusu, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Continual unsupervised representation learning. *arXiv preprint arXiv:1910.14481*, 2019. 3

[15] Ryne Roady, Tyler L. Hayes, Hitesh Vaidya, and Christopher Kanan. Stream-51: Streaming classification and novelty detection from videos. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020. 3

[16] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. 2

[17] James Smith, Cameron Taylor, Seth Baer, and Constantine Dovrolis. Unsupervised progressive learning and the stam architecture. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2979–2987. International Joint Conferences on Artificial Intelligence Organization, 8 2021. Main Track. 2

[18] Fei Ye and Adrian G Bors. Learning latent representations across multiple data domains using lifelong vaegan. In *European Conference on Computer Vision*, pages 777–795. Springer, 2020. 3

[19] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pages 12310–12320. PMLR, 2021. 2

[20] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR, 2017. 2