Supplementary Information

1 Methods

1.1 Real-time Learning Benchmark

1.1.1 Network Pretraining

The networks tested in the real-time learning benchmark were pretrained on ImageNet [7] and VGGFace2 [1]. We randomly sampled 1281167 images from VGGFace2 and combined it with ImageNet, which also has 1281167 images. In addition to this mixing, we also added a data-augmentation pipeline, which is added right after the random-resized cropping operation that ends with a 224×224 image, then randomly scales the image into a smaller image of shape $S \times S$ (S is randomly sampled from 224 to 50), and padding the smaller image using gray pixels (pixel value 127) to return to shape 224×224 with the smaller image in the center. This pipeline is randomly applied with a probability of 0.6. After pretraining, we selected the checkpoint with the highest initial d' values, which was typically an earlier checkpoint than the last one. We find both of these two additional settings (cotraining with VGGFace2 and the random-resizing-gray-padding augmentation pipeline) are needed to get the reasonable initial d' on the face test images. We provide all the pretrained checkpoints in the Supplementary Materials.

1.1.2 Video Stream Construction

For one model under one continual learning setting, we ran 15 experiments through starting from the same checkpoint but varying the selection of face pairs from six faces. For each experiment, all the three conditions (Non-Swap, Swap, and Switch) were ran independently. For each condition, a control face pair was first randomly selected and fixed across this condition, which is also how the experiments on humans were run. This control face pair contains two faces that are different from the selected experiment face pair. For the test phase, each test trial was constructed through first showing the test image, then simulating four saccades from the two middle-sized face images with 0.6s between two saccades, and finally ending with 0.5s gray images (pixel value 127). In total, one test trial took 3s. One test phase included 200 test trails randomly sampled from the tests on the selected face pair and on the control face pair. The construction of the exposure phase has been described in the main text (see Fig 5 **A** for examples).

We also provide a pseudo-code description of this construction procedure in Alg 1.

1.1.3 d' and Learning Effects Computation

To compute d', the hit rate (H = hit/(hit + miss)) and the false alarm rate (F = false alarm/(false alarm + correct rejection)) are first computed. Then d' is defined as norm.ppf(H) − norm.ppf(F), where norm.ppf is the percent point function of a normal distribution. The learning effect at test phase i is computed by $(d_i'^{exp} - d_0'^{exp}) - (d_i'^{control} - d_0'^{control})$, where $d_i'^{exp}$ is the d' of the experiment object pair at this test phase and $d_i'^{control}$ is the d' of the corresponding control object pair. $i = 0$ is the first test phase.

1.1.4 Mismatch Computation

For one test phase of one condition, the learning effects of the model across different object pairs are sampled in a bootstrap fashion to get 1000 samples and then averaged to get one mean learning effect, which is called one bootstrapped model mean effect, or e^M_{boot}. This is then compared to the bootstrapped human mean effect of the corresponding phase (e^H_{boot}) to get the absolute difference between these two mean effects. This comparison is performed for 1000 times to get the mean absolute difference (mean(abs($e^M_{boot} - e^H_{boot}$))). This measure is normalized by mean(abs(mean(e^H_{boot}) − e^H_{boot})) to get the final mismatch score, whose minimal value is 1.

1.1.5 Learning-Rate Search and Number of Steps per Phase

For most of the training settings such as the batch size and the optimizer, we followed the exactly same settings used during the ImageNet + VGGFace2 pretraining. For one model in one continual learning setting, we varied the learning rate with the fixed number of steps per phase (150). For the three experiment conditions and all 15 face pairs within each experiment condition, the same learning rate was used. The learning rate was varied with respect to each model and the continual learning setting, for typically at least three values. All the effects are provided in the Supplementary Material (see "all_effects.tar"). As for the fixed number of steps per phase, this number has proven enough in the manually selected aggregation pair experiments, where all models show low mismatch scores. We have also tested longer steps (300 and 600) for the BYOL-More-MLPs model in the high-diversity settings and find that the mismatch scores are very similar to those from the 150-step setting.

1.1.6 Initial d' and Mismatch

For each object pair, we find that its learning effect in the Non-Swap condition is typically lower than $4 - d'_{init}$ and the absolute value of its learning effect in the Swap condition is usually smaller than d'_{init} (Fig 8 **A**). So the models with the wrong initial d' fail to match the human learning effects as it is easy for

these models to show increasingly better performance in the Non-Swap condition but hard to show increasingly worse performance in the Swap condition. This observation also explains why proper pre-training to have the correct initial d' is important and why catastrophic forgetting leads to worse mismatch, as networks have wrong initial d' after catastrophic forgetting. However, this observation by itself cannot fully explain the failure of the models using the naturally emerging aggregation pairs, since BYOL has a good final d' on non-exposed objects but still shows a large mismatch (Fig 8 **B**).

1.2 Life-long Learning Benchmark

1.2.1 Training Settings

Unless otherwise specified, all networks were trained for 100 segments. LARS optimizer [12] was used with learning rate 4.8, momentum 0.9, and weight decay 1e-6. The learning rate started from 0 and linearly increased to 4.8 in 10 epochs (segments) with changes in each step. A cosine decay was applied to the learning rate to finally reduce it to 0. Each batch contained 512 pairs of images, but the gradients were averaged across 8 steps and then applied, making the effective batch size 4096. Each segment contained 2502 batches. Typical data augmentation pipeline included the following: random resized crop, random horizontal flip, random color jitter, random gray scale, and random Gaussian filter. The input images were then color-normalized using ImageNet color mean and std. Different methods may have their modified data augmentation pipeline. The input images to the DNNs were in resolution 112 to make the training faster. The main results of the life-long learning benchmark were also validated in larger resolution (224) and longer training steps (300 segments). All embeddings were L2-normalized before being used for loss computation. MAE was trained with larger batch size (1024) and also more batches per-segment (5004), making the overall number of image-pair-presentations four times of the typical number. This was following the original practice in He et al. [10], where the authors compared the results from 400 epochs with the results of other models from 100 epochs. This was also due to the high mask ratio (0.75) used in the model.

1.2.2 Mini-ImageNet Evaluation

At the end of every 10 segments, the outputs of the ResNet18 encoder in the shape 512 were extracted on the Mini-ImageNet images, which were resized to make the shortest edge 128 and center cropped to get $112 * 112$ images. A linear SVM was trained on the training subset to do the multi-class object recognition and evaluated on the evaluation subset. We reported the best performance among linear SVMs with α values chosen from {1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1, 1e1, 1e2}. The Mini-ImageNet was constructed through randomly sampling 100 classes and 200 images per class. The evaluation subset was the corresponding validation images for these classes.

1.2.3 Algorithm Loss Definitions

SimCLR. [4] For one batch containing bs items, of which one item contains two images x_0^i and $x_1^i, i \in \{1, 2, ..., bs\}$, and the two sampled data augmentations (v_i^0 and v_i^1), the optimized loss is defined as follows:

$$\mathcal{L}_{\text{SimCLR}} = -\frac{1}{bs}\sum_{i=0}^{bs}\sum_{j=0,1}\log\frac{\exp({e_i^j}^T e_i^{1-j}/\tau)}{\sum_{k=0}^{bs}\sum_{l=0,1}\exp({e_i^j}^T e_k^l/\tau)} \tag{1}$$

where τ is 0.1, $e_i^j = f(v_i^j(x_j^i))$.

MoCo v2. [9, 6] For the momentum encoder ($\hat{f}$) and $e^1 = \hat{f}(v^1(x_1))$, the loss is defined as follows:

$$\mathcal{L}_{\text{MoCo v2}} = -\log(\frac{\exp({e^0}^T e^1/\tau)}{\sum_{k=0}^{\text{q}}\exp({e^0}^T e_k/\tau)}) \tag{2}$$

where q is the size of the queue (typically 65536) and τ is 0.2. The momentum encoder is updated with fixed momentum 0.999 (meaning that after each step, $\hat{\theta} = \hat{\theta} * 0.999 + \theta * 0.001$, where $\hat{\theta}$ is the parameters for $\hat{f}$ and θ represents the parameters for f).

BYOL. [8] Similar as MoCo v2, BYOL also maintains the momentum encoder as the target network. However, BYOL does not use other embeddings as reference points. Instead, it only maximizes the correctness of predicting the target embedding ($e^1 = \hat{f}(v^1(x_1))$) from the other embedding ($e^0 = f(v^0(x_0))$) using a Multi-Layer-Perceptron (MLP), called the "predictor". The loss is defined as follows:

$$\mathcal{L}_{\text{BYOL}} = 1 - \cos\_\text{sim}(e^1, \text{MLP}(e^0)) \tag{3}$$

where cos_sim represents cosine similarity.

SimSiam. [5] SimSiam can be characterized as BYOL without momentum encoder. Therefore, given $e^0 = f(v^0(x_0))$ and $e^1 = f(v^1(x_1))$, the loss is defined as follows:

$$\mathcal{L}_{\text{SimSiam}} = 2 - \cos\_\text{sim}(\text{SG}(e^0), \text{MLP}(e^1)) - \cos\_\text{sim}(\text{SG}(e^1), \text{MLP}(e^0)) \tag{4}$$

where SG represents stop gradient operation.

DINO. [3] DINO is like BYOL on transformers, though it has additional mechanisms like centering the teacher embedding and softmaxing both the student and the teacher embeddings. More specifically, the loss is defined as follows:

$$\mathcal{L}_{\text{DINO}} = -\text{SG}(\text{softmax}(e^1 - e_{\text{center}})) * \log(\text{softmax}(e^0)) \tag{5}$$

where e_{center} is the exponential average of the batch-averaged e^1 across its history. DINO also has very large output dimensionality for both e^0 and e^1 (like 65536), which is bottlenecked by a typical embedding dimensionality like 256.

SwAV. [2] SwAV maintains a trainable set of K prototypes C and uses these prototypes to compute a cluster assignment (q^0, q^1) for embeddings (e^0, e^1). The loss is then to "swap-predict" the assignment from the other embedding, defined as follows:

$$\mathcal{L}_{\text{SwAV}} = -\sum_{j=0,1}\sum_{k} q_k^j * \log\frac{\exp({e^{(1-j)}}^T c_k)}{\sum_{k'}({e^{(1-j)}}^T c_{k'})} \tag{6}$$

where c_k means the k-th prototype and q_k^j means the k-th value of the cluster assignment.

Barlow-Twins. [13] This method first compute the cross-correlation matrix $\mathcal{C}$ of the embeddings as follows:

$$\mathcal{C}_{ij} = \frac{\sum_b e^0_{b,i} e^1_{b,j}}{\sqrt{\sum_b (e^0_{b,i})^2}\sqrt{\sum_b (e^1_{b,j})^2}} \tag{7}$$

where $e^0_{b,i}$ is the value at the i-th dimension for e^0_b. The loss is then defined as follows:

$$\mathcal{L}_{\text{BT}} = \sum_i (1-\mathcal{C}_{ii})^2 + \lambda \sum_i \sum_{j\neq i} \mathcal{C}_{ij}^2 \tag{8}$$

where λ is 0.0051.

BYOLNeg. Intuitively, BYOLNeg is something like "SimCLR with a momentum encoder," or "MoCo v2 with negative samples from the current batch." Following the same notations used in the main text, the loss is then $\mathcal{L}_{\text{BYOLNeg}} = -\frac{1}{\text{bs}}\sum_{i=0}^{\text{bs}} \log\frac{\exp({e_i^0}^T e_i^1/\tau)}{\sum_{k=0}^{\text{bs}}\exp({e_i^0}^T e_k^1/\tau)}$. where τ is 0.1, $e^1 = \hat{f}(v^1(x_1))$, and $e^0 = f(v^0(x_0))$. It has no "predictor", as we find that adding it makes the performance much worse.

DINONeg. The loss of DINONeg is in fact the same as BYOLNeg, however, we followed other practices of DINO like linearly increasing the weight decay to 0.4.

1.2.4 Models with More MLPs

For SimCLR and BYOL, we find that increasing the number of layers in the MLPs used between the ResNet-18 backbone and the embedding output helps the performance in life-long benchmark in general. The original number of layer is 2 and the models with "More-MLPs" have 4 layers. For BYOL, this change was applied to both the "predictor" and the "projector".

1.2.5 BYOL with Other Hyperparameters

As shown in Fig 7, we varied the starting momentum, weight decay value, number of layers in the MLPs, and even the backbone architecture for BYOL, but its poor performance in lower-diversity conditions seems to still be true.

1.2.6 EWC on Pure-Continual Curriculum

To explore whether methods like Elastic Weight Consolidation (EWC) [11] can be used to avoid catastrophic forgetting when training the DNNs without any replay from the memory, we trained SimCLR with EWC on pure-continual curriculum ($R = 1 : 0$). As shown in Fig 6, the performance of models with EWC is barely different from the pure-continual one, even after varying two important hyperparameters in the EWC algorithm (λ and γ). Here λ controls the weighting between the losses from the current batch and the "old tasks" (see Eq. 3 in Kirkpatrick et al. [11]). γ controls the updating speed of the Fisher information: $F_{new} = \gamma F_{old} + (1 - \gamma)F_{now}$. The default value of γ is 0.9. Although these experiments were run with 300-segment settings and resolution 224, the results should be the same for the typical setting (100-segment setting and resolution 112).

1.3 Computational Resources

We use TITAN Xp gpus on our internal cluster to train our models. For life-long learning benchmark, as one model roughly needs 2 gpus for 2 days and there were 13 algorithms trained with 6 continual learning settings, all results took $13 * 6 * 2 * 2 = 312$ gpu$*$days. For real-time learning benchmark, as testing one model in one continual learning setting for all three conditions with one learning rate took 4 gpus for 6 hours. There were 13 models, 6 learning settings, around 3 learning rates in each combination. So all results took $13 * 6 * 3 * 4 * 0.25 = 234$ gpu$*$days.

1.4 Codes and Model Weights

Source codes are provided with the Supplementary Materials. The link to download the model checkpoints we have used for the real-time learning benchmark is also released with the source codes. See the "source_codes.tar" file.

2 Figures

3 Tables

Table 1: Life-long Learning results. The column names are correspondingly the window length (W) and the current-memory mix ratio (R).

Alg.	20m, 1:3	0.5m, 1:3	20m, 1:1	0.5m, 1:1	20m, 3:1	0.5m, 3:1
SimCLR-More-MLPs	0.34	0.33	0.32	0.29	0.28	0.20
SimCLR	0.31	0.31	0.30	0.28	0.27	0.23
MoCo-v2	0.25	0.26	0.26	0.23	0.25	0.15
Barlow-Twins	0.29	0.28	0.28	0.27	0.24	0.26
BYOLNeg	0.32	0.32	0.31	0.30	0.30	0.24
SwAV	0.33	0.30	0.33	0.12	0.30	0.05
BYOL-More-MLPs	0.29	0.28	0.26	0.19	0.15	0.03
BYOL	0.25	0.27	0.26	0.13	0.24	0.07
SimSiam	0.21	0.23	0.17	0.04	0.05	0.03
SimCLR-Res50	0.37	0.37	0.36	0.33	0.34	0.18
DINO-ViT-S	0.39	0.37	0.37	0.35	0.34	0.26
DINONeg-ViT-S	0.38	0.37	0.37	0.31	0.33	0.23
MAE-ViT-S	0.28	0.28	0.27	0.28	0.27	0.26

Table 2: Real-time Learning results. The column names are correspondingly the window length (W) and the current-memory mix ratio (R). Numbers after $\pm$ are standard deviations across bootstrapped examples.

Alg.	20m, 1:3	0.5m, 1:3	20m, 1:1	0.5m, 1:1	20m, 3:1	0.5m, 3:1
SimCLR-More-MLPs	1.74 $\pm$ 0.48	1.66 $\pm$ 0.41	1.39 $\pm$ 0.38	1.71 $\pm$ 0.45	1.49 $\pm$ 0.36	1.78 $\pm$ 0.37
SimCLR	1.98 $\pm$ 0.49	2.53 $\pm$ 0.56	1.43 $\pm$ 0.43	1.56 $\pm$ 0.44	1.49 $\pm$ 0.37	1.91 $\pm$ 0.47
MoCo-v2	1.70 $\pm$ 0.46	1.78 $\pm$ 0.48	1.68 $\pm$ 0.42	1.92 $\pm$ 0.45	1.67 $\pm$ 0.43	1.72 $\pm$ 0.43
Barlow-Twins	2.50 $\pm$ 0.69	1.78 $\pm$ 0.49	1.96 $\pm$ 0.53	2.02 $\pm$ 0.43	1.95 $\pm$ 0.51	1.81 $\pm$ 0.49
BYOLNeg	2.29 $\pm$ 0.45	1.94 $\pm$ 0.49	2.30 $\pm$ 0.42	2.00 $\pm$ 0.38	2.13 $\pm$ 0.45	2.14 $\pm$ 0.45
SwAV	2.47 $\pm$ 0.53	2.82 $\pm$ 0.57	2.64 $\pm$ 0.58	2.20 $\pm$ 0.47	2.41 $\pm$ 0.43	2.21 $\pm$ 0.44
BYOL-More-MLPs	2.26 $\pm$ 0.60	2.79 $\pm$ 0.55	2.93 $\pm$ 0.52	2.67 $\pm$ 0.52	2.16 $\pm$ 0.50	2.61 $\pm$ 0.47
BYOL	2.62 $\pm$ 0.54	3.09 $\pm$ 0.51	2.55 $\pm$ 0.56	3.11 $\pm$ 0.65	2.70 $\pm$ 0.56	2.38 $\pm$ 0.49
SimSiam	2.79 $\pm$ 0.54	2.68 $\pm$ 0.48	3.18 $\pm$ 0.58	3.47 $\pm$ 0.55	2.51 $\pm$ 0.57	3.15 $\pm$ 0.50
SimCLR-Res50	2.43 $\pm$ 0.54	2.57 $\pm$ 0.48	2.19 $\pm$ 0.49	1.80 $\pm$ 0.47	1.91 $\pm$ 0.47	1.47 $\pm$ 0.40
DINO-ViT-S	2.30 $\pm$ 0.47	2.13 $\pm$ 0.46	2.27 $\pm$ 0.42	2.24 $\pm$ 0.44	1.73 $\pm$ 0.44	1.72 $\pm$ 0.45
DINONeg-ViT-S	1.98 $\pm$ 0.45	2.41 $\pm$ 0.51	2.04 $\pm$ 0.42	2.47 $\pm$ 0.51	2.67 $\pm$ 0.50	1.91 $\pm$ 0.47
MAE-ViT-S	3.40 $\pm$ 0.53	2.90 $\pm$ 0.53	3.07 $\pm$ 0.55	2.93 $\pm$ 0.52	3.18 $\pm$ 0.52	3.49 $\pm$ 0.56

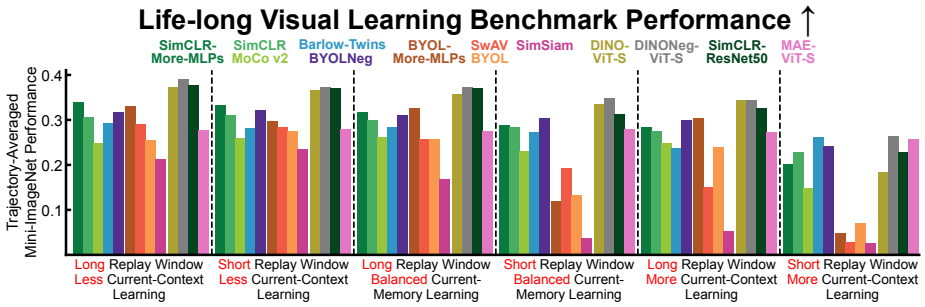

Figure 1: **Life-Long Learning results.** Life-long benchmark performance measured by the trajectory-averaged Mini-ImageNet performance. Long replay window means $W = 20m$ and short window means $W = 0.5m$. More current-context learning means $R = 3 : 1$, balanced means $R = 1 : 1$, and less means $R = 1 : 3$. In all conditions, $T = 0.2s$.

References

[1] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*, pages 67–74. IEEE, 2018.

[2] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33:9912–9924, 2020.

[3] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9650–9660, 2021.

[4] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

[5] X. Chen and K. He. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*, 2020.

[6] X. Chen, H. Fan, R. Girshick, and K. He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. 2009.

[8] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

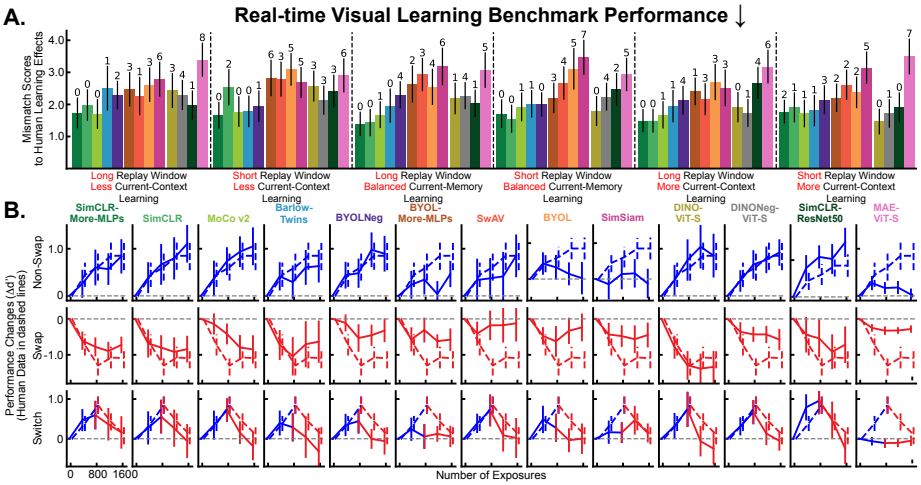

Figure 2: **Real-time Learning results. A.** Real-time benchmark performance. Long replay window means $W = 20m$ and short window means $W = 0.5m$. More current-context learning means $R = 3:1$, balanced means $R = 1:1$, and less means $R = 1:3$. In all conditions, $T = 0.2s$. **B.** Learning effects of humans in dashed lines and the unsupervised DNNs under their best conditions in solid lines.

[9] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[10] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021.

[11] J. Kirkpatrick, R. Pascanu, N. C. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796, 2016. URL `http://arxiv.org/abs/1612.00796`.

[12] Y. You, I. Gitman, and B. Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

[13] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. Barlow twins: Self-supervised learning via redundancy reduction. *arXiv preprint arXiv:2103.03230*, 2021.

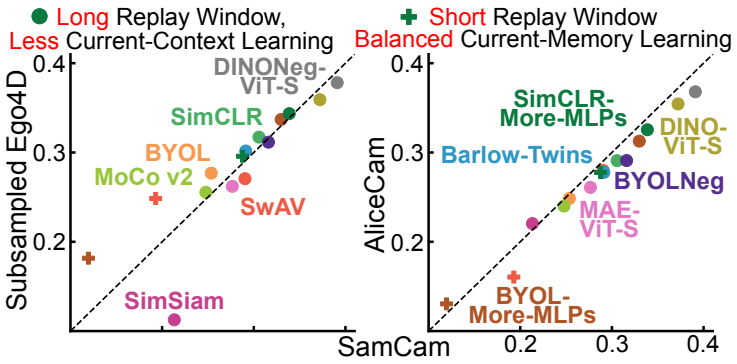

Figure 3: **Generalizability of the life-long benchmark results.** Each dot or cross represents a model trained under the specific condition. $T = 0.2s$ for both conditions. Both axes represent the trajectory-averaged Mini-ImageNet performance. For both panels, x-axis represents the performance of models trained on SamCam. For the left panel, y-axis is the performance of models trained on the subsampled Ego4D dataset. For the right panel, y-axis is the performance of models trained on AliceCam. For Ego4D, we sampled 300 videos that are at least one hour long are subsampled and randomly sorted to form the 100 segments. Each Ego4D segment contains three videos.

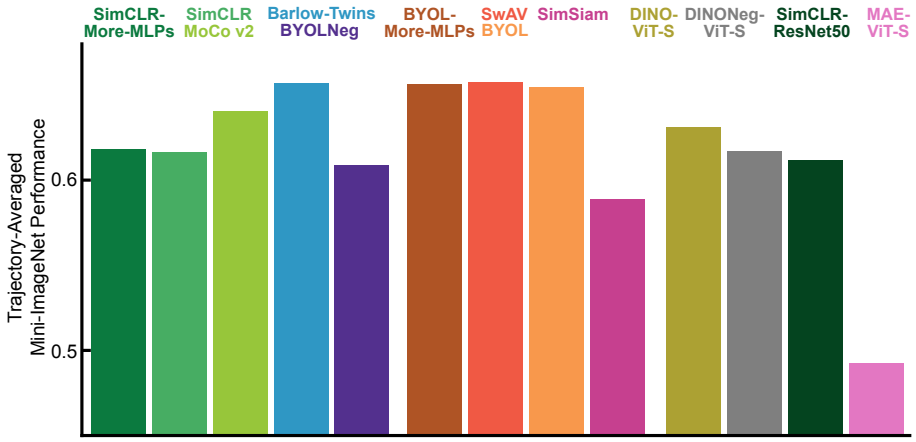

Figure 4: **ImageNet offline learning results.** ImageNet offline learning performance measured by the trajectory-averaged Mini-ImageNet performance. The significantly low performance of MAE-ViT-S on ImageNet is also validated using the official repository by the authors of this model (https://github.com/facebookresearch/mae). This result is also consistent with the findings in the original paper that the MAE-trained ViTs perform less well in linear-probing evaluations compared to other methods. All models were trained with resolution 224 and 300 epochs. DINO-ViT-S, DINONeg-ViT-S, and SimCLR-ResNet50 were trained for 100 epochs. MAE-ViT-S was trained for 400 epochs.

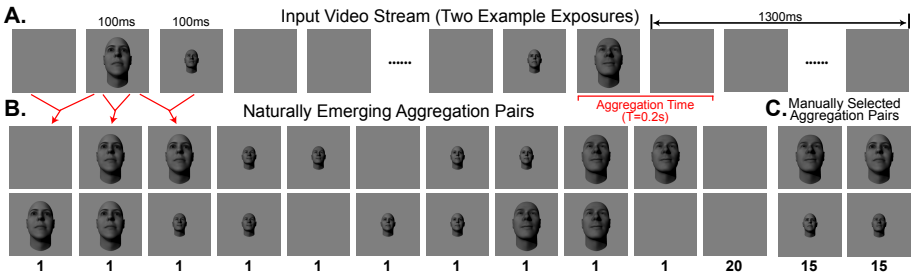

Figure 5: **Naturally emerging and manually selected aggregation pair examples. A.** Video stream of two example exposures in the swapped condition. **B.** Aggregation pairs naturally emerging from the two example exposures with aggregation time $T = 0.2s$. **C.** Aggregation pairs manually selected.

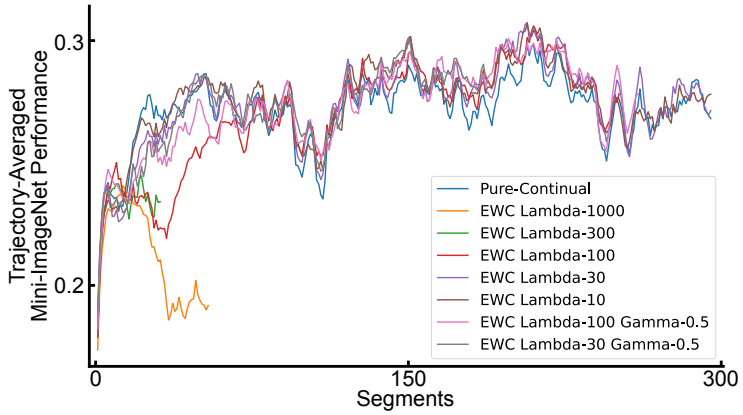

Figure 6: **Purely-Continual SimCLR (R=1:0) with EWC.** The λ and γ in EWC are varied (see Methods).

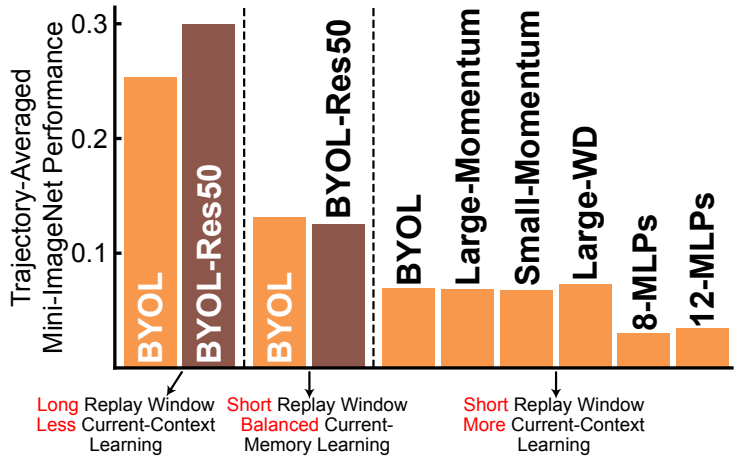

Figure 7: **BYOL with different hyperparameters.** Large momentum means setting the starting momentum to 0.995 and small momentum means 0.95. Large WD means setting the weight decay to $1e-4$ instead of $1e-9$.

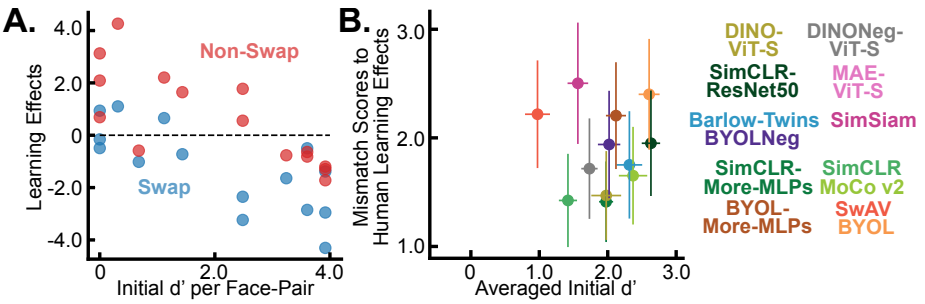

Figure 8: **Initial d' and mismatch score. A.** Scatter plot of the initial d' per face pair and its corresponding learning effect in Non-Swap and Swap conditions. Each dot represents one face-pair. This is from BYOL with manually selected aggregation pairs. **B.** Scatter plot of the averaged initial d' across all face pairs and the corresponding mismatch scores for all models using naturally emerging aggregation pairs. Each dot represents one model.

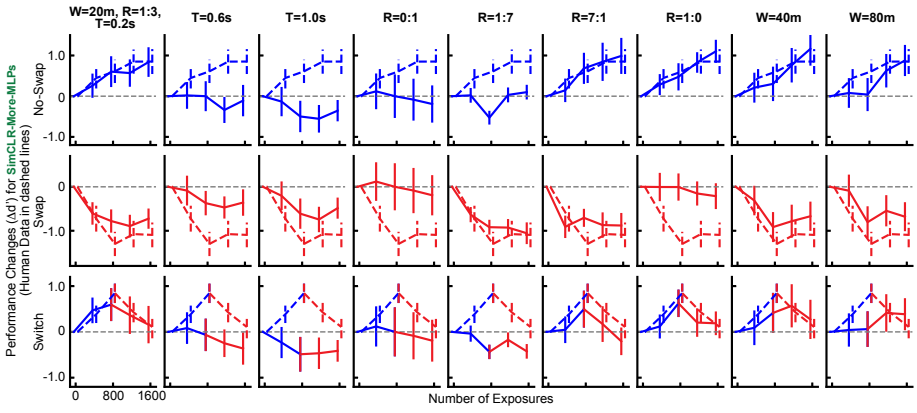

Figure 9: **Learning effects of SimCLR-More-MLPs with different continual learning settings.**

Algorithm 1: Build the Visual Stimuli Stream for Real-time Learning Benchmark.

```
begin
    Input  : o_0, o_1: experiment object pair; o_2, o_3: control object pair; c: condition
    Output : VS: a list of images, each of which represents the image of focus for 100ms
    Initialize the visual stimuli stream VS = [ ] ;
    Set GI as the gray background image, whose pixel values are all 127 ;
    for phase index i ∈ {0 to 8} do
        if i is even then
            for test event index j ∈ {0 to 199} do
                Add 5 GIs to VS ;
                Randomly choose the test event type t from { Big-Exp, Big-Control, Small-Exp, Small-Control } ;
                According to t, add a test image to VS, which is a big or small object in front of a randomly selected background ;
                for saccade index s ∈ {0 to 3} do
                    Add 6 copies of one randomly selected prototype image from two presented prototype images to VS, which is a middle-sized object image in front of a gray background ;
                end
            end
        end
        else
            for exposure event index j ∈ {0 to 399} do
                According to c, add two exposure object images to VS ;
                Add 13 GIs to VS ;
            end
        end
    end
    return VS ;
end
```

Algorithm 2: Real-time Learning Evaluation for Neural Networks.

begin
- Set W as the current-context replay window in the unit of minutes;
- Set T as the aggregation time in the unit of seconds;
- Set $R = X : Y$ as the current-memory mix ratio;
- Set the batch size of the current context as $bs_c = bs \times \frac{X}{X+Y}$;
- **for** *object pair* o_0, o_1 **do**
  - **for** *condition* $c \in \{\text{Non} - \text{Swap}, \text{Swap}, \text{Switch}\}$ **do**
    - Randomly determine the control object pair o_2, o_3 that are different from o_0, o_1;
    - Build VS according to o_0, o_1, o_2, o_3, and c using Alg. 1;
    - Load the pretrained network weights;
    - **for** *step index* $i \in \{0$ *to* $1349\}$ **do**
      - Set the phase index as $p = \text{floor}(i/150)$;
      - **if** *i is a multiple of 75 and p is even* **then**
        - `/* We evaluate twice in every test phase */`
        - Test the object-recognition accuracy of the network;
      - **end**
      - Set the batch buffer $B = [\,]$;
      - Get the corresponding time point $t = i/1350 * 90 * 60$ in the unit of seconds;
      - **for** *within-batch item index* $j \in \{1$ *to* $bs_c\}$ **do**
        - Randomly select a time point t_j from $(t - 60W, t)$;
        - Get the aggregation interval $(t_j - T, t_j)$;
        - Randomly sample two images from $VS[(t_j - T) \times 10 : t_j \times 10]$;
        - Apply data augmentation pipeline independently to these two images;
        - Add the pair of the augmented images to B as one item;
      - **end**
      - Sample $bs - bs_c$ items from the pretraining dataset and add them to B;
      - Train the network for one step using B;
    - **end**
  - **end**
- **end**

end

Algorithm 3: Life-long Learning Evaluation for Neural Networks.

begin
  Randomly initialize the network weights;
  Set $W, T, R = X : Y, bs_c$ as in Alg. 2;
  for *segment index* $s \in \{0$ *to* $99\}$ **do**
    Get the videos belonging to this segment;
    Get the frames FS belonging to these videos and order them according to the corresponding time index;
    Set L as the number of frames in FS;
    `/* As the frames are extracted in 25 FPS and one segment typically contains videos of 2hrs, L is typically 25 * 3600 * 2 */`
    for *step index* $i \in \{0$ *to* $2501\}$ **do**
      Set the batch buffer $B = [\,]$;
      Get the corresponding frame index $f = i/2502 * L$;
      for *within-batch item index* $j \in \{1$ *to* $bs_c\}$ **do**
        Randomly select a frame point f_j from $(f - 60 \times 25 \times W, f)$;
        Randomly sample two images from $FS[f_j - 25 \times T, f_j]$;
        Apply data augmentation pipeline independently to these two images;
        Add the pair of the augmented images to B as one item;
      end
      Sample $bs - bs_c$ items from the memory and add them to B;
      Train the network for one step using B;
    end
    if $s + 1$ *is a multiple of 10* **then**
      Test the Mini-ImageNet performance of the network;
    end
  end
end