Supplementary Materials: Tuned Compositional Feature Replays for Efficient Stream Learning

Morgan B. Talbot*, Rushikesh Zawar*, Rohil Badkundri, Mengmi Zhang†, and Gabriel Kreiman†
*Equal contribution †Corresponding authors

S1. Compositional Replay Using Memory Blocks (CRUMB) outperforms competing algorithms in the class-iid setting in most cases

We report top-1 accuracy results (measured on all tasks/classes in each dataset at the end of stream learning training) for CRUMB and all competing baseline algorithms on five video streaming datasets (CORe50 [1], Toybox [2], iLab [3], iLab+CORe50, and iCub [4]) and two image datasets (Online-CIFAR100 [5], Online-Imagenet [6]) in both class-iid and class-instance training protocols in Table I in the main text. For CRUMB and a subset of baseline algorithms, we illustrate task-by-task top-1 accuracy (on all previously seen classes) for the five video datasets in the class-iid setting in Fig. S1. Class-instance plots for video datasets, and class-iid plots for image datasets, are shown in Fig. 3 (main text).

S2. CRUMB's performance is competitive with baseline algorithms even when memory buffer size is unlimited

Our primary baseline comparison experiments in the main text focus on comparing CRUMB with competing algorithms under a fixed memory budget: methods that store and replay entire raw images cannot store as many training examples, affecting their continual learning performance. CRUMB is specifically designed with memory-constrained conditions in mind, and compresses each training example stored in its replay buffer to occupy only 3.6% as much memory as an entire image (e.g., as stored by image-based replay baseline iCARL [7]). Nonetheless, we demonstrate here in Fig. S2 that CRUMB obtains competitive performance even when memory usage is unlimited, greatly outperforming iCARL. CRUMB's accuracy after training on all tasks is slightly lower than that of REMIND [8] in the class-instance setting under these conditions, with both methods close to the offline upper bound in both class-instance and class-iid.

S3. CRUMB maintains its bias towards object shape information after class-iid stream learning on video datasets

In Section IV.D.3 and Fig. 4 in the main text, we observe that pretraining CRUMB on ImageNet [6] induces a bias in the CNN towards attending to object shape information more than image texture information, an effect that has been shown to mitigate catastrophic forgetting by flattening the loss minimum of each task [9]. Fig. 4 in the main text visualizes the extent of this "shape bias" for CRUMB trained on video datasets in the class-instance setting. Fig. S3 shows results in the class-iid setting. As for class-instance, we observe that CRUMB mostly retains its shape bias during class-iid stream learning.

S4. Data analysis

A. Data cleaning

For our main results on the video datasets CORe50, Toybox, and iLab, we noticed that a small subset of runs for some models had markedly reduced accuracy on the first task compared to other runs. To facilitate fair comparisons among models, we excluded all runs with an initial task accuracy less than 80% from all analysis and results. For the small number of algorithm/dataset/protocol combinations for which no runs exceeded 80% on the first task, we filtered at a 60% threshold, or a 40% threshold if no runs exceeded 60%. We did not encounter this issue for any runs of CRUMB on any dataset, or for any method on Online-CIFAR100 and Online-Imagenet.

B. Statistics for model analysis experiments

Our model analysis experiments in main-text Section IV.D compared the performance of CRUMB with various ablated or otherwise perturbed versions of CRUMB. For each comparison with the original algorithm, we evaluated statistical significance of pairwise differences using the following method:

i. Divide the test set from the dataset being used into batches of 100 images. The images should be randomly sampled without replacement, and the sampling should be done only once (or, using a fixed random seed) for all experiments such that each version of the algorithm is evaluated on the exact same batches of images.
ii. Evaluate CRUMB and each experimentally perturbed version of CRUMB on the same set of image batches and record mean top-1 accuracy on each batch. This is done for each of the 5 independent training runs, and accuracies are pooled across runs. Therefore, for each training protocol (class-instance and class-iid, for which all analyses are kept separate), each version of the algorithm has $n_r \times n_b$ top-1 accuracy estimates, where n_r is the number of runs and n_b is the number of 100-image batches in the test set. Conceptually, we treat the accuracy on each batch as an independent sample indicating the accuracy of the corresponding algorithm on a roughly continuous scale, with each run of each algorithm tested on the exact same batches of images.
iii. Perform a paired-samples t-test for each comparison, using accuracy on each image batch of CRUMB and the

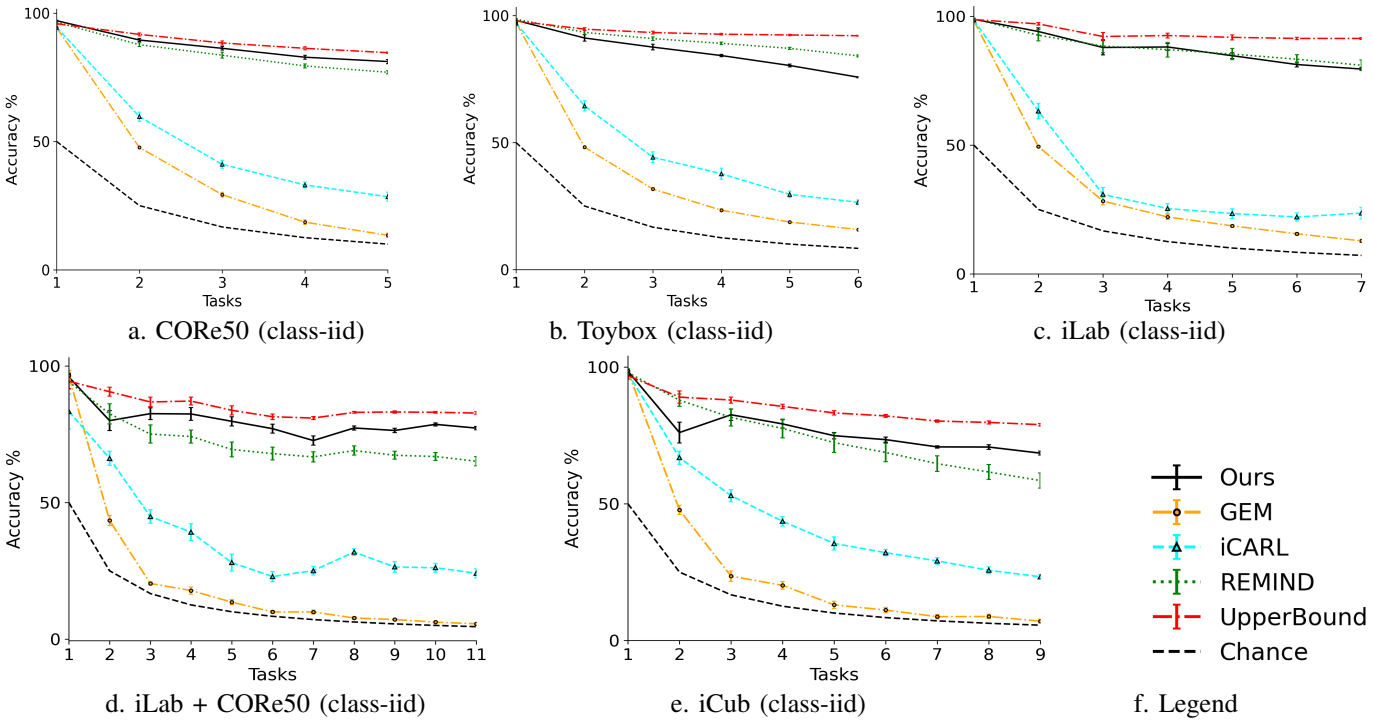


Fig. S1. **In the class-iid setting, CRUMB outperforms most baseline algorithms and performs near the the upper bound on some datasets**. Line plots show top-1 accuracy in online stream learning on video datasets (a) CORe50, (b) Toybox, (c) iLab, (d) iLab + CORe50, and (e) iCub in the class-iid setting. All models train on the first task for many epochs, but view each image only once on all subsequent tasks. Accuracy estimates are the mean from 10 runs, where each run has different class and image/video clip orderings. Error bars show the root-mean-square error (RMSE) among runs. Results for all baselines are in Table I in the main text.

perturbed version of CRUMB as a sample pair and pooling sample pairs across runs. We used a global p-value cutoff of $p < 0.01$ to report the statistical significance of t-test results for each comparison between CRUMB and a perturbed version of CRUMB.

For our experiments on shape-texture bias, we employ a similar approach by first calculating accuracy on batches of 100 images at a time. For each batch, we subtract the model's perturbed accuracy (i.e., after spatial, feature, or style perturbation, see Section IV.D.3 in the main text) from the unperturbed accuracy, and divide the result by the unperturbed accuracy to obtain the relative accuracy drop for each perturbation. We compare the relative accuracy drops for CRUMB and a control network on all batches, pooling across runs with different data orderings, using the Wilcoxon signed-rank test for paired samples. We apply a global p-value cutoff of $p < 0.01$ to report the significance of any differences, visualized as CRUMB's relative accuracy advantage being either above or below zero in main-text Fig. 4 and supplementary Fig. S3.

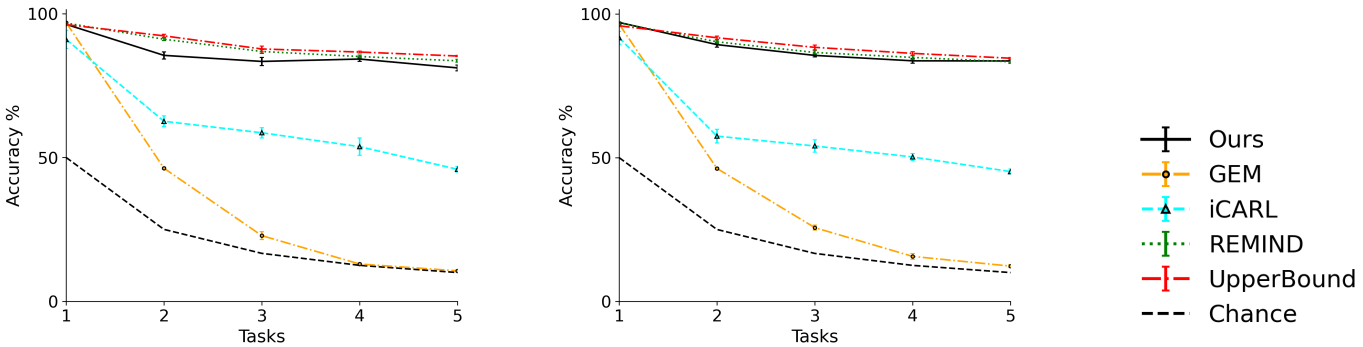
S5. Replay buffer size calculations

For replay-based baseline algorithms, we limit the number of examples that can be stored in the buffer to fit within a memory budget that is held constant for all methods in our main results (main text Table I). We do not apply this constraint for weight regularization approaches. To calculate the maximum number of training examples we can store in the replay buffer for each experiment, we first set the number of examples n_{raw} that raw-image replay methods such as iCARL may store, then calculate how many examples (n_x) CRUMB can fit into the same amount of memory usin the formula:

$$n_x = \frac{n_r(3w_ih_i) - bd}{swh/d} \tag{1}$$

Where w_i and h_i are raw image width and height respectively (224×224 for our experiments), the codebook matrix has dimensions $b \times d$ (b memory blocks, each of dimension d), and the feature map has dimensions $s \times w \times h$ (s features in a $w \times h$ spatial grid). The numerator corresponds to the number of 8-bit RGB values needed to store one image, subtracting a discounting factor for the number of values in the memory blocks themselves. The denominator corresponds to the number of 8-bit integer indices required to encode one feature map. Concretely, the memory budgets are 2.2 MB on CORe50, Toybox, and iLab, 14.3 MB on CIFAR100, and 1.44 GB on ImageNet based on the number of 8-bit integers each method stores per training example.

For direct comparisons between algorithms in our main results, we applied both CRUMB and REMIND to the SqueezeNet network architecture [11]. To calculate n_x for REMIND, we multiplied the compression ratio provided by the REMIND paper (959,665 feature maps/10,000 raw images) by the ratio of values in one feature map from ResNet18 (used in the REMIND paper, $512 \times 7 \times 7$) to those in one feature map from SqueezeNet ($512 \times 13 \times 13$) [8]. We then multiplied the resulting ratio of 278,246 feature maps/10,000 raw images by n_{raw} to obtain the corresponding n_x for each dataset.



a. CORe50 (class-instance) b. CORe50 (class-iid) c. Legend

Fig. S2. **CRUMB attains competitive levels of performance in conditions of unlimited memory usage**. Line plots show top-1 accuracy in online stream learning on the video dataset CORe50. For this comparison among replay methods, all models are allowed to store all previously encountered images in a replay buffer and intersperse them with images encountered while training on new tasks. All models train on the first task for many epochs, but view each image only once on all subsequent tasks. Accuracy estimates are the mean from 10 runs, where each run has different class and image/video clip orderings. Error bars show the root-mean-square error (RMSE) among runs.

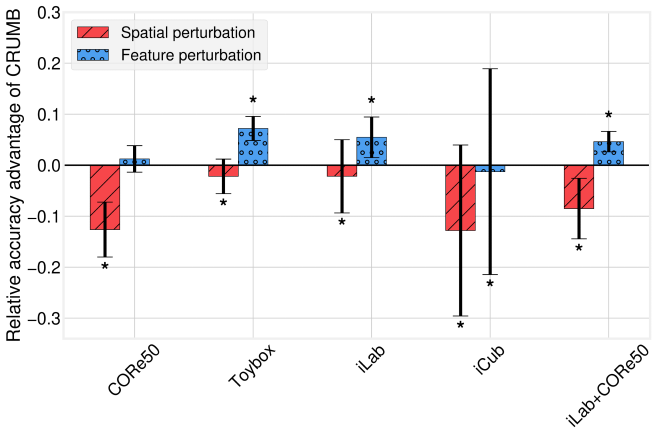


Fig. S3. **The bias towards shape information induced by CRUMB pretraining persists through stream learning in the class-iid setting**. The height of each bar shows how much smaller (or larger, if negative) CRUMB's drop in normalized test set accuracy under a perturbation is, in comparison to a control network. "Relative accuracy advantage" is calculated by dividing the difference in accuracy caused by a perturbation by the unperturbed accuracy, and then subtracting this result for CRUMB from that of the control network (see main-text Section IV.D.3). "Spatial perturbation" shuffles the spatial positions of all feature vectors in an intermediate feature map (at the same layer where it is reconstructed by CRUMB), "feature perturbation" randomly sets half of the feature map's features to zero, and "style perturbation" uses images from Stylized-ImageNet [10]. Streaming results (to the right of grey dotted line) are in the class-iid setting: class-instance shape-texture bias results for the video datasets, and class-iid results for Online-CIFAR100 and Online-ImageNet, are available in Fig. 4 in the main text. Error bars are standard errors of the mean of relative accuracy advantage among or 10 independent runs. * denotes a statistically significant difference from 0 (see Section S4).

References

[1] V. Lomonaco and D. Maltoni, "Core50: A new dataset and benchmark for continuous object recognition," in *Conference on Robot Learning*, PMLR, 2017, pp. 17–26.

[2] X. Wang, T. Ma, J. Ainooson, *et al.*, "The toybox dataset of egocentric visual object transformations," *arXiv preprint arXiv:1806.06034*, 2018.

[3] A. Borji, S. Izadi, and L. Itti, "Ilab-20m: A large-scale controlled object dataset to investigate deep learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2221–2230.

[4] G. Pasquale, C. Ciliberto, L. Rosasco, and L. Natale, "Object identification from few examples by improving the invariance of a deep convolutional neural network," in *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, IEEE, 2016, pp. 4904–4911.

[5] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[7] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "Icarl: Incremental classifier and representation learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010.

[8] T. L. Hayes, K. Kafle, R. Shrestha, M. Acharya, and C. Kanan, "Remind your neural network to prevent catastrophic forgetting," in *European Conference on Computer Vision*, Springer, 2020, pp. 466–483.

[9] Z. Shi, Y. Sun, J. H. Lim, and M. Zhang, "On the robustness, generalization, and forgetting of shape-texture debiased continual learning," *arXiv preprint arXiv:2211.11174*, 2022.

[10] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness.," in *International Conference on Learning Representations*, 2019.

[11] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.