Supplemental Material

S1 Parameter Settings

Table S1. Training parameter settings for REMIND and Offline models.

Parameters	ImageNet	CORe50	TDIUC	CLEVR
Optimizer	SGD	SGD	Adamax	Adamax
Learning Rate	0.1	0.01	2e-3	3e-4
Momentum	0.9	0.9	-	-
Weight Decay	1e-4	1e-4	-	-
Streaming Batch Size	51	21	51	51
Offline Batch Size	128	256	512	64
Offline Epochs	90	40	20	20
Offline LR Decay	[30,60]	[15,30]	-	-

We provide parameter settings for REMIND and the offline models in Table S1. For the image classification experiments, we use the ResNet-18 implementation from the PyTorch Torchvision package. For the offline ImageNet model, we use standard data augmentation of random resized crops and random flips at 224×224 pixels. We employ per-class learning rate decay for REMIND on ImageNet, using 0.1 as the starting learning rate and decaying it such that the learning rate becomes 0.001 after seeing all new samples for a class, at a step size of 100 new instances. For the k-means variant of REMIND, we use a codebook size of 10000 for ImageNet, and we found that increasing the codebook size yielded only marginal performance improvements. For CORe50, we do not use data augmentation with REMIND, as it harms performance. Unlike batch methods, REMIND learns one class at a time instance-by-instance.

To train REMIND on ImageNet in the incremental batch setting, we follow a paradigm similar to the incremental batch paradigm used by [64, 77]. The base initialization stage for REMIND remains the same, where it trains offline on 100 classes and then subsequently trains the product quantizer and stores indices for previous examples in its memory buffer. We subject REMIND to the same buffer size during incremental batch learning as we do for streaming learning, which equates to 1.51 GB or compressed representations for 959665 examples. After base initialization, REMIND receives the next batch of 100 classes of data and mixes in all of the data from its replay buffer. It then loops over this data for 40 epochs, where the learning rate starts at 0.1 and is decayed by a factor of 10 at epochs 15 and 30. After looping over a batch, REMIND updates its replay buffer by storing new samples until it is full, and then randomly replacing samples from the class with the most examples. Consistent with our streaming experiments, the incremental batch version of REMIND uses random resized crops and mixup augmentation.

For ExStream on the image classification experiments, we use 20 prototype vectors per class and the same parameters as the offline models. For SLDA on all experiments, we use shrinkage regularization of 10^{-4}. Both ExStream and SLDA

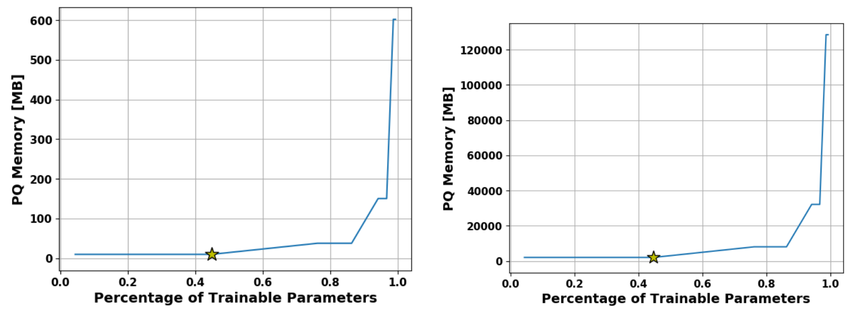


Fig. S1. Auxiliary storage required to store quantized CNN features for the entire dataset as a function of the percentage of ResNet-18 parameters used in the top of the CNN, $F(\cdot)$, which are updated during streaming learning in REMIND. Storage requirements are shown for CORe50 (left) and ImageNet (right). The star denotes parameters used for our main experiments.

learn classes one at a time, instance-by-instance. For ImageNet, the parameters of iCaRL are kept the same as [64]. Similarly, the parameters for Unified and BiC on ImageNet are from [27] and [77], respectively. For the batch versions of CORe50 with iCaRL, Unified, and BiC, we train each batch for 60 epochs with a batch size of 64, weight decay of 1e-4, and a learning rate of 0.01 that we lower at epochs 20 and 40 by a factor of 5. For the streaming versions of iCaRL, Unified, and BiC, we set the number of epochs to 1 and the batch size to 51 and 21 for ImageNet and CORe50, respectively.

For MAC, we use the publicly available PyTorch implementation (https://github.com/IBM/mi-prometheus). For SAN, we use our own PyTorch implementation. For ExStream on TDIUC, we use an MLP with layer sizes [4096, 1024, 1480], lr = 2e-3, dropout with probability 0.5, Adamax optimizer, batch size of 512, and store 2500 exemplars per class. For ExStream on CLEVR, we use an MLP with layer sizes [3072, 1024, 28], lr = 2e-3, dropout with probability 0.5, Adamax optimizer, batch size of 512, and store 65 exemplars per class. For TDIUC and CLEVR, we chose the number of exemplars to consist of roughly 10% of the dataset size.

S2 Where Should ResNet-18 be Quantized?

Following others, we used ResNet-18 for our incremental learning image classification experiments. This constrained the layers we could choose for quantization. If we quantized earlier in the network, the spatial dimensions of the feature tensor would be too large, resulting in much greater auxiliary storage requirements (see Fig. S1). For example, in our ImageNet experiments, if we chose layer 3 of ResNet-18 for quantization, it would require 129 GB to store a representation of the entire training dataset; in contrast, the layer we used in our main experiments would require only 2 GB to store the entire training set. It is also a more biologically sensible layer to choose based on the connectivity of the hippocampus to visual processing areas.

If the architecture of ResNet-18 was altered to decrease the spatial dimensions earlier in the network, with a corresponding increase in the feature dimensions, this would allow us to quantize earlier in the network. However, this would prevent us from comparing directly to prior work and may require a considerable amount of architectural search to find a good compromise.

S3 Additional Image Classification Experiments

S3.1 Buffer Size Comparisons

Since REMIND and several other comparison models use replay as their main mechanism for mitigating forgetting, we were interested in examining how changes to the replay buffer size affected model performance on both ImageNet and CORe50. In Fig. S2, we compare the performance of the incremental batch versions of iCaRL, Unified, and BiC on ImageNet at buffer sizes of 5K exemplars = 0.75 GB, 10K exemplars = 1.51 GB, and 20K exemplars = 3.01 GB, which are equivalent to REMIND storing 479665 compressed samples = 0.75 GB, 959665 compressed samples = 1.51 GB, and 1281167 compressed samples (full dataset) = 2.01 GB respectively. Note that this plot shows the performance of iCaRL, Unified, and BiC in the batch setting, but shows REMIND in the streaming setting, which is consistent with our main experiments.

These results demonstrate that REMIND and BiC are the top performers when a memory buffer of 0.75 GB is used, but BiC is the top performer when a memory buffer of 1.51 GB or 3.01 GB is used. However, REMIND rivals BiC's performance at both of these larger buffer sizes, only underperforming by 4% and 6.6% at 1.51 GB and 3.01 GB, respectively. It should be noted that BiC requires nearly 65 hours to train in incremental batch mode on ImageNet with a buffer size of 1.51 GB, whereas REMIND requires less than 12 hours with the same buffer size. Additionally, REMIND's performance is less dependent on the size of the buffer than BiC. That is, the difference between REMIND's performance at 0.75 GB and 3.01 GB is only 1.3% in terms of Ω_{all}, whereas the difference between BiC's performance is 7.9%, indicating that BiC is highly sensitive to the amount of storage allotted for replay. Additionally, while comparison models require 3.01 GB for the largest buffer size, REMIND's buffer size never exceeds 2.01 GB. Regardless, REMIND still achieves remarkable performance and rivals the state-of-the-art BiC model, even in the incremental batch setting.

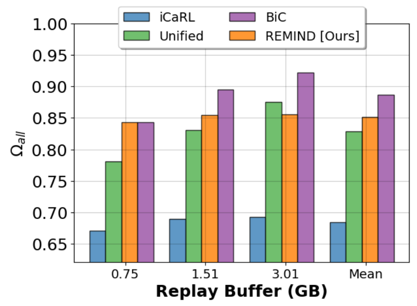
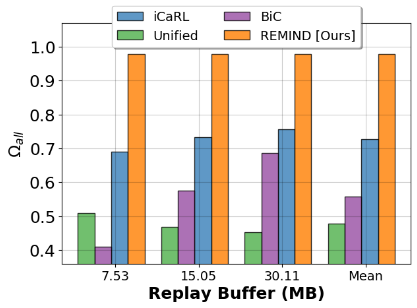


Fig. S2. Performance as a function of buffer size for various batch comparison models on ImageNet.



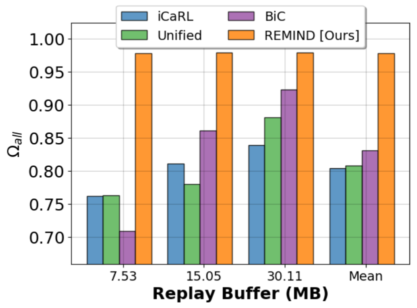


Fig. S3. Performance as a function of buffer size for various streaming (left) and batch (right) comparison models on CORe50. Each bar is the average over 10 permutations.

In Fig. S3, we study the performance of the same models in both streaming and incremental batch mode on the CORe50 dataset. We study the performance of iCaRL, Unified, and BiC with buffer sizes of 50 exemplars = 7.53 MB, 100 exemplars = 15.05 MB, and 200 exemplars = 30.11 MB, which are equivalent to storing 4465 compressed samples = 7.53 MB and 6000 compressed samples (full dataset) = 9.93 MB respectively for REMIND. Our model is run in the streaming paradigm for both plots and outperforms all comparison models, regardless of the training paradigm, across all buffer sizes. This is remarkable since REMIND uses only ⅓ the amount of memory as compared to comparison models at 200 exemplars. Moreover, all of these comparison models use large amounts of additional memory to cache the information needed for distillation before learning a batch, which REMIND does not require.

S3.2 Updating Only θ_F

Since REMIND only updates θ_F, it begs the question: is REMIND's superior performance a result of keeping θ_G fixed during incremental training? To answer this, we explore how other models perform when only θ_F is updated. On ImageNet, iCaRL, Unified, and BiC experience an absolute drop in Ω_{all} performance by 10.6%, 2.7%, and 2.8%, respectively when only θ_F is plastic. This performance degradation indicates that this architectural choice actually harms competitors and does not provide REMIND with an unfair advantage.

S3.3 Changing $F(\cdot)$ and $G(\cdot)$

One of the novelties of REMIND is the use of mid-level CNN features for training θ_F. However, choosing where to extract features to train the PQ is an open question. In Fig, S4, we find that adding more trainable layers to θ_F improves accuracy on CORe50, but it has diminishing returns and there is a greater memory burden since features earlier in the network have larger spatial dimensions.

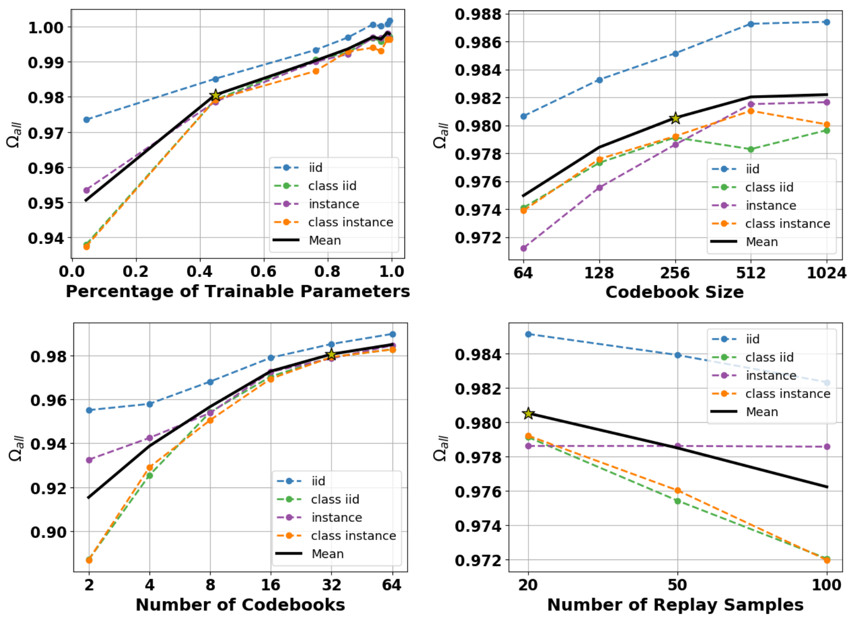


Fig. S4. Additional experiments with REMIND on CORe50. From left to right, top to bottom, performance as a function of: 1) trainable parameters, 2) codebook size, 3) number of codebooks, and 4) number of replay samples (r). The values used for our main experiments are denoted with a yellow star and each dashed line is the average of 10 runs.

S3.4 Varying PQ Settings

REMIND's performance is dependent on the quality of tensor reconstructions used for training $F(\cdot)$. Since we use PQ to reconstruct samples from the replay buffer for REMIND, the performance is dependent on: 1) the number of codebooks used and 2) the size of the codebooks. We study performance on CORe50 as a function of the number of codebooks and codebook size in Fig. S4. We find that the performance improves as the number of codebooks and codebook size increase. However, memory efficiency decreases when these values are increased, so, we choose the number of codebooks to be 32 and codebook size to be 256 for our main experiments, making a trade-off between accuracy and memory efficiency. Since REMIND's performance is nearly unaffected by storing only 4465 samples compared to 6000, i.e., the entire CORe50 dataset, (see Fig. S3), we store the entire training set in the replay buffer for these additional studies.

S3.5 Altering Replay

In our main experiments, each replay set contained 20 and 50 reconstructed samples for CORe50 and ImageNet, respectively. In Fig. S4, we study performance

Table S2. Average accuracy (μ_{all}) results for each dataset and ordering. For CORe50, we report the average over 10 runs. The best *streaming* model for each dataset and ordering is highlighted in **bold**.

		ImageNet	**CORe50**			
Model Type	Model	cls iid	iid	cls iid	inst	cls inst
Streaming	Fine-Tune (θ_F)	26.80	88.72	11.95	76.27	11.95
	ExStream	52.65	87.97	48.01	83.72	46.91
	SLDA	69.28	90.16	53.87	86.52	53.99
	iCaRL	28.61	-	37.88	-	35.46
	Unified	56.77	-	23.18	-	24.00
	BiC	40.64	-	16.08	-	16.68
	REMIND	**78.68**	**91.00**	**55.35**	**88.08**	**55.42**
Incremental Batch	iCaRL	63.59	-	41.94	-	42.10
	Unified	76.56	-	40.03	-	41.19
	BiC	82.38	-	35.08	-	39.24
	REMIND	80.55	-	-	-	-
Upper Bounds	Offline (θ_F)	85.52	91.32	55.80	88.56	55.88
	Offline	91.95	92.35	56.99	89.93	56.94

on CORe50 as a function of the number of replay samples. We found that performance degrades on CORe50 when we use more than 20 samples for replay. We hypothesize that since CORe50 has fewer samples, larger replay sizes cause overfitting, thereby degrading the performance. However, performance increases by 0.6% for ImageNet (in terms of Ω_{all}), when the number of replay samples is increased from 20 to 50, which is the reason for using 50 samples in our main ImageNet experiments. Similar to the study of various PQ settings with REMIND on CORe50, we again store the entire training set in the replay buffer for this study on CORe50 due to the negligible performance difference (see Fig. S3).

S3.6 Average Accuracy for ImageNet and CORe50

In the main paper, we present Ω_{all}, which makes it easy to compare across datasets, orderings, and paradigms. However, it can hide the raw performance of the models. Following others [13, 27, 64, 77], we provide the average accuracy metric over all testing intervals, i.e.,

$$\mu_{\text{all}} = \frac{1}{T}\sum_{t=1}^{T} \alpha_t \quad , \tag{2}$$

where T is the total number of testing events and α_t is the accuracy of the model for test t. We provide μ_{all} results in Table S2, which shows the top-5 accuracy for ImageNet and top-1 accuracy for CORe50. When using these metrics, REMIND is still the top streaming performer and competitive in the incremental batch

Table S3. Average accuracy (μ_{all}) results for CORe50 with their associated standard deviations over 10 runs with different permutations of the data. The streaming models are at the top of the table, while the upper bounds are at the bottom. The best model for each ordering is highlighted in **bold**.

Model	iid	cls iid	inst	cls inst
Fine-Tune (θ_F)	88.72±1.57	11.95±0.02	76.27±4.44	11.95±0.03
ExStream	87.97±0.83	48.01±2.17	83.72±1.78	46.91±2.35
SLDA	90.16±0.63	53.87±0.79	86.52±1.12	53.99±0.82
iCaRL	-	37.88±3.41	-	35.46±2.89
Unified	-	23.18±5.47	-	24.00±5.69
BiC	-	16.08±1.93	-	16.68±2.00
REMIND	**91.00±0.58**	**55.35±0.95**	**88.08±1.33**	**55.42±0.86**
Offline (θ_F)	91.32±0.42	55.80±0.61	88.56±1.04	55.88±0.60
Offline	92.35±0.40	56.99±0.48	89.93±0.78	56.94±0.46

setting on ImageNet. CORe50 results for the class orderings are lower because we test on all test data at every interval, which includes classes that are yet to be seen. This leads to low accuracies for the unseen classes, which affects μ_{all}.

On CORe50 we also report the average accuracy and associated standard deviation values over 10 runs with different permutations of the dataset in Table S3. Overall, the iid and instance orderings yielded the highest model performances, making them easiest, while the class orderings resulted in much worse performance, making them hardest. REMIND's results are statistically significantly different from each of the comparison models for all four data orderings according to a Student's t-test at a 99% confidence interval.

S4 Additional VQA Experiments

S4.1 REMIND Performance for Various Buffer Sizes

Table S4. Ω_{all} results for REMIND with various buffer sizes on streaming VQA.

	TDIUC		CLEVR	
Buf. Size	iid	q-type	iid	q-type
25%	0.914	**0.936**	**0.724**	0.960
50%	0.917	0.919	0.720	0.979
75%	**0.919**	0.914	0.722	0.984
100%	0.914	0.931	0.723	**0.985**
Offline	1.000	1.000	1.000	1.000

In Table S4, we provide additional results for REMIND on TDIUC and CLEVR with buffer sizes that consist of 25%, 50%, 75%, and 100% of the samples from the entire training set. Overall, we see that REMIND performs remarkably well with a limited buffer size. For example, the model trained with only a 25%

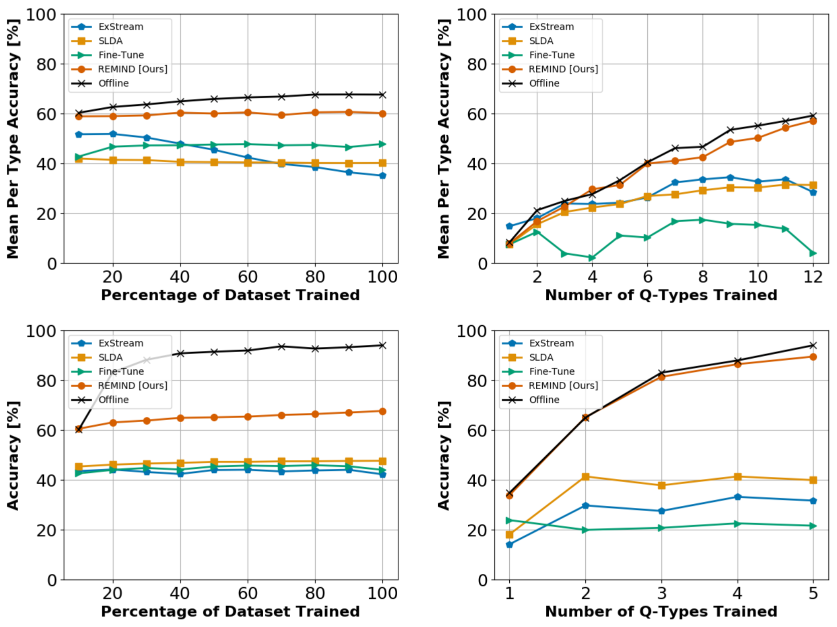
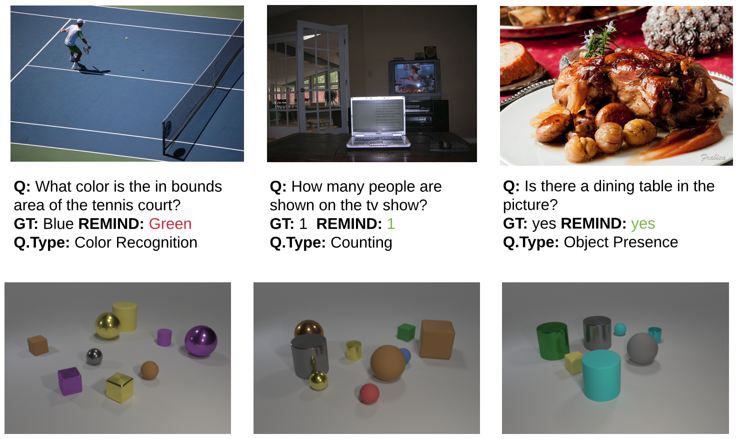


Fig. S5. Learning curves for each ordering of the TDIUC (top row) and CLEVR (bottom row) datasets. We provide curves from the REMIND model trained with 50% buffer size.

buffer size rivals, and in some cases outperforms, the model with a 100% buffer size.

S4.2 Learning Curves and Qualitative Examples

We provide learning curves for each of the main VQA experiments in Fig. S5 and qualitative examples in Fig. S6. REMIND's learning curve closely follows the offline curve for the q-type ordering of both the TDIUC and CLEVR datasets. This indicates that our model is able to learn new q-types without forgetting old q-types. For the iid ordering of TDIUC, the accuracy remains more or less constant after the first increment and for the iid ordering of CLEVR, the accuracy increases at a slower rate than the offline model. We believe that training with samples ordered by q-type may have acted as a natural curriculum for REMIND, providing more benefits to the VQA model.



Q: What color is the in bounds area of the tennis court?
GT: Blue **REMIND:** Green
Q.Type: Color Recognition

Q: How many people are shown on the tv show?
GT: 1 **REMIND:** 1
Q.Type: Counting

Q: Is there a dining table in the picture?
GT: yes **REMIND:** yes
Q.Type: Object Presence

Q: What number of tiny things are both on the left side of the gray shiny sphere and to the right of the brown rubber cube?
GT: 1 **REMIND:** 1
Q.Type: Counting

Q: Are the big brown ball that is on the right side of the big metal cylinder and the sphere on the left side of the small metallic ball made of the same material?
GT: yes **REMIND:** no
Q.Type: Material Comparison

Q: Is the number of tiny yellow matte things on the left side of the small yellow matte object less than the number of cyan metal cylinders that are left of the cyan shiny thing?
GT: no **REMIND:** no
Q.Type: Integer Comparison

Fig. S6. Qualitative VQA examples on the TDIUC (top row) and CLEVR (bottom row) datasets. We provide examples from the REMIND model trained with 50% buffer size on the q-type ordering for both datasets.