REMIND Your Neural Network to Prevent Catastrophic Forgetting

Tyler L. Hayes[1,⋆], Kushal Kafle[2,⋆], Robik Shrestha[1,⋆], Manoj Acharya[1], and Christopher Kanan[1,3,4]

[1] Rochester Institute of Technology, Rochester NY 14623, USA
{tlh6792,rss9369,ma7583,kanan}@rit.edu
[2] Adobe Research, San Jose CA 95110, USA
kkafle@adobe.com
[3] Paige, New York NY 10036, USA
[4] Cornell Tech, New York NY 10044, USA

Abstract. People learn throughout life. However, incrementally updating conventional neural networks leads to catastrophic forgetting. A common remedy is replay, which is inspired by how the brain consolidates memory. Replay involves fine-tuning a network on a mixture of new and old instances. While there is neuroscientific evidence that the brain replays compressed memories, existing methods for convolutional networks replay raw images. Here, we propose REMIND, a brain-inspired approach that enables efficient replay with compressed representations. REMIND is trained in an online manner, meaning it learns one example at a time, which is closer to how humans learn. Under the same constraints, REMIND outperforms other methods for incremental class learning on the ImageNet ILSVRC-2012 dataset. We probe REMIND's robustness to data ordering schemes known to induce catastrophic forgetting. We demonstrate REMIND's generality by pioneering online learning for Visual Question Answering (VQA)[5].

Keywords: Online Learning, Brain-inspired, Deep Learning

1 Introduction

The mammalian brain engages in continuous online learning of new skills, objects, threats, and environments. The world provides the brain a temporally structured stream of inputs, which is not independent and identically distributed (iid). Enabling online learning in artificial neural networks from non-iid data is known as lifelong learning. While conventional networks suffer from catastrophic forgetting [1,57], with new learning overwriting existing representations, a wide variety of methods have recently been explored for overcoming this problem [13,15,27,47,53,58,64,77]. Some of the most successful methods for mitigating catastrophic forgetting use variants of replay [13,22,27,45,64,77], which involves

⋆ Equal Contribution.
[5] https://github.com/tyler-hayes/REMIND

mixing new instances with old ones and fine-tuning the network with this mixture. Replay is motivated by how the brain works: new experiences are encoded in the hippocampus and then these compressed memories are re-activated along with other memories so that the neocortex can learn them [51,60,72]. Without the hippocampus, people lose the ability to learn new semantic categories [48]. Replay occurs both during sleep [31] and when awake [41,74].

For lifelong learning in convolutional neural networks (CNNs), there are two major gaps between existing methods and how animals learn. The first is that replay is implemented by storing and replaying raw pixels, which is not biologically plausible. Based on hippocampal indexing theory [75], the hippocampus stores *compressed* representations of neocortical activity patterns while awake. To consolidate memories, these patterns are replayed and then the corresponding neocortical neurons are re-activated via reciprocal connectivity [51,60,72]. The representations stored in the hippocampus for replay are not veridical (e.g., raw pixels) [31,56], and its visual inputs are high in the visual processing hierarchy [29] rather than from primary visual cortex or retina.

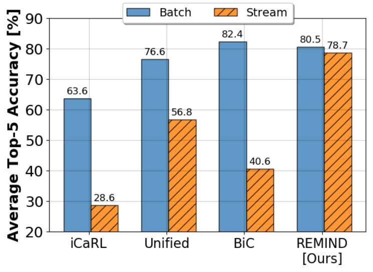


Fig. 1. Average top-5 accuracy results for streaming and incremental batch versions of state-of-the-art models on ImageNet.

The second major gap with existing approaches is that animals engage in *streaming learning* [20,21], or resource constrained online learning from non-iid (temporally correlated) experiences throughout life. In contrast, the most common paradigm for incremental training of CNNs is to break the training dataset into M distinct batches, where for ImageNet each batch typically has about 100000 instances from 100 classes that are not seen in later batches, and then the algorithm sequentially loops over each batch many times. This paradigm is not biologically plausible. There are many applications requiring online learning of non-iid data streams, where batched learning will not suffice, such as immediate on-device learning. Batched systems also take longer to train, further limiting their utility on resource constrained devices, such as smart appliances, robots, and toys. For example, BiC, a state-of-the-art incremental batch method, requires 65 hours to train in that paradigm whereas our proposed streaming model trains in under 12 hours. The incremental batch setting can be transformed into the streaming learning scenario by using very small batches and performing only a single pass through the dataset; however, this results in a large decrease in performance. As shown in Fig. 1, state-of-the-art methods perform poorly on ImageNet in the streaming setting, with the best method suffering an over 19% drop in performance. In contrast, our model outperforms the best streaming model by 21.9% and is only 1.9% below the best batch model.

Here, we propose REMIND, or **re**play using **m**emory **ind**exing, a novel method that is heavily influenced by biological replay and hippocampal indexing theory. **Our main contributions are:**

1. We introduce REMIND, a streaming learning model that implements hippocampal indexing theory using tensor quantization to efficiently store hidden representations (e.g., CNN feature maps) for later replay. REMIND implements this compression using Product Quantization (PQ) [30]. We are the first to test if forgetting in CNNs can be mitigated by replaying hidden representations rather than raw pixels.
2. REMIND outperforms existing models on the ImageNet ILSVRC-2012 [68] and CORe50 [52] datasets, while using the same amount of memory.
3. We demonstrate REMIND's robustness by pioneering streaming Visual Question Answering (VQA), in which an agent must answer questions about images and cannot be readily done with existing models. We establish new experimental paradigms, baselines, and metrics and subsequently achieve strong results on the CLEVR [33] and TDIUC [35] datasets.

2 Problem Formulation

There are multiple paradigms in which incremental learning has been studied [61]. In *incremental batch learning*, at each time step t an agent learns a data batch B_t containing N_t instances and their corresponding labels, where N_t is often 1000 to 100000. While much recent work has focused on incremental batch learning [13,14,18,27,44,45,64,77,81], *streaming learning*, or online learning from non-iid data streams with memory and/or compute constraints, more closely resembles animal learning and has many applications [20,21,49]. In streaming learning, a model learns online in a single pass, i.e., $N_t = 1$ for all t. It cannot loop over any portion of the (possibly infinite) dataset, and it can be evaluated at any point rather than only between large batches. Streaming learning can be approximated by having a system queue up small, temporally contiguous, mini-batches for learning, but as shown in Fig. 1, batch methods cannot easily adapt to this setting.

3 Related Work

Parisi et al. [61] identify three main mechanisms for mitigating forgetting in neural networks, namely 1) replay of previous knowledge, 2) regularization mechanisms to constrain parameter updates, and 3) expanding the network as more data becomes available. Replay has been shown to be one of the most effective methods for mitigating catastrophic forgetting [4,5,13,22,27,44,45,50,59,64,77]. For ImageNet, all recent state-of-the-art methods for incremental class learning use replay of raw pixels with distillation loss. The earliest was iCaRL [64], which stored 20 images per class for replay. iCaRL used a nearest class prototype classifier to mitigate forgetting. The End-to-End incremental learning model [13] extended iCaRL to use the outputs of the CNN directly for classification, instead of a nearest class mean classifier. Additionally, End-to-End used more data augmentation and a balanced fine-tuning stage during training to improve performance. The Unified classifier [27] extended End-to-End by using a cosine normalization layer, a new loss constraint, and a margin ranking loss. The Bias Correction (BiC) [77] method extended End-to-End by training two additional parameters to correct bias in the output layer due to class imbalance. iCaRL, End-to-End, the Unified classifier, and BiC all: 1) store the same number of raw replay images per class, 2) use the same herding procedure for prototype selection, and 3) use distillation loss to prevent forgetting. REMIND, however, is the first model to demonstrate that storing and replaying quantized mid-level CNN features is an effective strategy to mitigate forgetting.

Regularization methods vary a weight's plasticity based on how important it is to previous tasks. These methods include Elastic Weight Consolidation (EWC) [47], Memory Aware Synapses (MAS) [3], Synaptic Intelligence (SI) [81], Riemannian Walk (RWALK) [14], Online Laplace Approximator [66], Hard Attention to the Task [70], and Learning without Memorizing [16]. The Averaged Gradient Episodic Memory (A-GEM) [15] model extends Gradient Episodic Memory [53], which uses replay with regularization. Variational Continual Learning [58] combines Bayesian inference with replay, while the Meta-Experience Replay model [65] combines replay with meta-learning. All of these regularization methods are typically used for incremental *task* learning, where batches of data are labeled as different tasks and the model must be told which task (batch) a sample came from during inference. When task labels are not available at test time, which is often true for agents operating in real-time, many methods cannot be used or they will fail [14,17,45]. While our main experiments focus on comparisons against state-of-the-art ImageNet models, we compare REMIND against several regularization models in Sec. 7, both with and without task labels. Some regularization methods also utilize cached data, e.g., GEM and A-GEM.

Another approach to mitigating forgetting is to expand the network as new tasks are observed, e.g., Progressive Neural Networks [69], Dynamically Expandable Networks [79], Adaptation by Distillation [26], and Dynamic Generative Memory [59]. However, these approaches also use task labels at test time, have growing memory requirements, and may not scale to thousand-category datasets.

4 REMIND: Replay using Memory Indexing

REMIND is a novel brain-inspired method for training the parameters of a CNN in the streaming setting using replay. Learning involves two steps: 1) compressing the current input and 2) reconstructing a subset of previously compressed representations, mixing them with the current input, and updating the *plastic* weights of the network with this mixture (see Fig. 2). While earlier work for incremental batch learning with CNNs stored raw images for replay [13,27,64,77], by storing compressed mid-level CNN features, REMIND is able to store far more instances with a smaller memory budget. For example, iCaRL [64] uses a

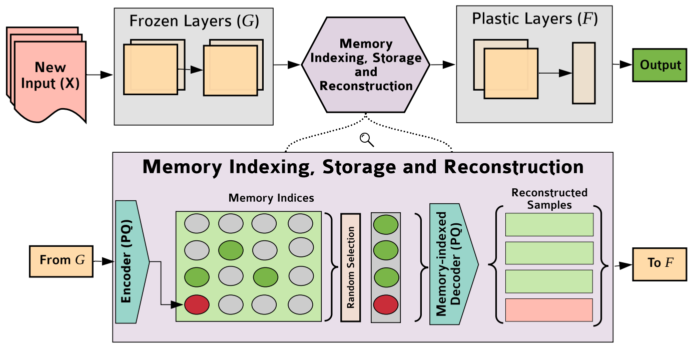


Fig. 2. REMIND takes in an input image and passes it through frozen layers of the network (G) to obtain tensor representations (feature maps). It then quantizes the tensors via product quantization and stores the indices in memory for future replay. The decoder reconstructs tensors from the stored indices to train the plastic layers (F) of the network before a final prediction is made.

default memory budget of 20K examples for ImageNet, but REMIND can store over 1M compressed instances using the same budget. This more closely resembles how replay occurs in the brain, with high-level visual representations being sent to the hippocampus for storage and re-activation, rather than early visual representations [29]. REMIND does not have an explicit sleep phase, with replay more closely resembling that during waking hours [41,74].

Formally, our CNN $y_i = F\left(G\left(\mathbf{X}_i\right)\right)$ is trained in a streaming paradigm, where $\mathbf{X}_i$ is the input image and y_i is the predicted output category. The network is composed of two nested functions: $G\left(\cdot\right)$, parameterized by θ_G, consists of the first J layers of the CNN and $F\left(\cdot\right)$, parameterized by θ_F, consists of the last L layers. REMIND keeps θ_G fixed since early layers of CNNs have been shown to be highly transferable [80]. The later layers, $F\left(\cdot\right)$, are trained in the streaming paradigm using REMIND. We discuss how $G\left(\cdot\right)$ is initialized in Sec. 4.2.

The output of $G\left(\mathbf{X}_i\right)$ is a tensor $\mathbf{Z}_i \in \mathbb{R}^{m \times m \times d}$, where m is the dimension of the feature map and d is the number of channels. Using the outputs of $G\left(\cdot\right)$, we train a vector quantization model for the $\mathbf{Z}_i$ tensors. As training examples are observed, the quantization model is used to store the $\mathbf{Z}_i$ features and their labels in a replay buffer as an $m \times m \times s$ array of integers using as few bits as necessary, where s is the number of indices that will be stored. For replay, we uniformly select r instances from the replay buffer, which was shown to work well in [14], and reconstruct them. Each of the reconstructed instances, $\hat{\mathbf{Z}}_i$, are mixed with the current input, and then θ_F is updated using backpropagation on this set of $r+1$ instances. Other selection strategies are discussed in Sec. 8. During inference, we pass an image through $G\left(\cdot\right)$, and then the output, $\mathbf{Z}_i$, is quantized and reconstructed before being passed to $F\left(\cdot\right)$.

Our main version of REMIND uses PQ [30] to compress and store $\mathbf{Z}_i$. For high-dimensional data, PQ tends to have much lower reconstruction error than models that use only k-means. The tensor $\mathbf{Z}_i$ consists of $m \times m$ d-dimensional tensor elements, and PQ partitions each d-dimensional tensor element into s sub-vectors, each of size d/s. PQ then creates a separate codebook for each partition by using k-means, where the codes within each codebook correspond to the centroids learned for that partition. Since the quantization is done independently for each partition, each sub-vector of the d-dimensional tensor element is assigned a separate integer, so the element is represented with s integers. If s is equal to one, then this approach is identical to using k-means for vector quantization, which we compare against. For our experiments, we set $s = 32$ and $c = 256$, so that each integer can be stored with 1 byte. We explore alternative values of s and c in supplemental materials (Fig. S4) and use the Faiss PQ implementation [32].

Since lifelong learning systems must be capable of learning from infinitely long data streams, we subject REMIND's replay buffer to a maximum memory restriction. That is, REMIND stores quantization indices in its buffer until this maximum capacity has been reached. Once the buffer is full and a new example comes in, we insert the new sample and randomly remove an example from the class with the most examples, which was shown to work well in [14,77]. We discuss other strategies for maintaining the replay buffer in Sec. 8.

4.1 Augmentation During Replay

To augment data during replay, REMIND uses random resized crops and a variant of manifold mixup [76] on the quantized tensors directly. For random crop augmentation, the tensors are randomly resized, then cropped and bilinearly interpolated to match the original tensor dimensions. To produce more robust representations, REMIND mixes features from multiple classes using manifold mixup. That is, REMIND uses its replay buffer to reconstruct two randomly chosen sets, $\mathcal{A}$ and $\mathcal{B}$, of r instances each ($|\mathcal{A}| = |\mathcal{B}| = r$), which are linearly combined to obtain a set $\mathcal{C}$ of r mixed instances ($|\mathcal{C}| = r$), i.e., a newly mixed instance, $(\mathbf{Z}_{\text{mix}}, y_{\text{mix}}) \in \mathcal{C}$, is formed as:

$$(\mathbf{Z}_{\text{mix}}, y_{\text{mix}}) = \left(\lambda \mathbf{Z}_a + (1-\lambda)\, \mathbf{Z}_b, \lambda y_a + (1-\lambda)\, y_b\right), \tag{1}$$

where $(\mathbf{Z}_a, y_a)$ and $(\mathbf{Z}_b, y_b)$ denote instances from $\mathcal{A}$ and $\mathcal{B}$ respectively and $\lambda \sim \beta(\alpha, \alpha)$ is the mixing coefficient drawn from a β-distribution parameterized by hyperparameter α. We use $\alpha = 0.1$, which we found to work best in preliminary experiments. The current input is then combined with the set $\mathcal{C}$ of r mixed samples, and θ_F is updated using this new set of $r + 1$ instances.

4.2 Initializing REMIND

During learning, REMIND only updates $F(\cdot)$, i.e., the top of the CNN. It assumes that $G(\cdot)$, the lower level features of the CNN, are fixed. This implies that the low-level visual representations must be highly transferable across image datasets, which is supported empirically [80]. There are multiple methods for training $G(\cdot)$, including supervised pre-training on a portion of the dataset, supervised pre-training on a different dataset, or unsupervised self-taught learning using a convolutional auto-encoder. Here, we follow the common practice of doing a 'base initialization' of the CNN [13,27,64,77]. This is done by training both θ_F and θ_G jointly on an initial subset of data offline, e.g., for class incremental learning on ImageNet we use the first 100 classes. After base initialization, θ_G is no longer plastic. All of the examples $\mathbf{X}_i$ in the base initialization are pushed through the model to obtain $\mathbf{Z}_i = G(\mathbf{X}_i)$, and all of these $\mathbf{Z}_i$ instances are used to learn the quantization model for $G(\mathbf{X}_i)$, which is kept fixed once acquired.

Following [13,27,64,77], we use ResNet-18 [25] for image classification, where we set $G(\cdot)$ to be the first 15 convolutional and 3 downsampling layers, which have 6,455,872 parameters, and $F(\cdot)$ to be the remaining 3 layers (2 convolutional and 1 fully connected), which have 5,233,640 parameters. These layers were chosen for memory efficiency in the quantization model with ResNet-18, and we show the memory efficiency trade-off in supplemental materials (Fig. S1).

5 Experiments: Image Classification

5.1 Comparison Models

While REMIND learns on a per sample basis, most methods for incremental learning in CNNs do multiple loops through a batch. For fair comparison, we train these methods in the streaming setting to fairly compare against REMIND. Results for the incremental batch setting for these models are included in Fig. 1 and supplemental materials (Table S2 and Fig. S2-S3). We evaluate the following:

- **REMIND** – Our main REMIND version uses PQ and replay augmentation. We also explore a version that omits data augmentation and a version that uses k-means rather than PQ.
- **Fine-Tuning (No Buffer)** – Fine-Tuning is a baseline that fine-tunes θ_F of a CNN one sample at a time with a single epoch through the dataset. This approach does not use a buffer and suffers from catastrophic forgetting [45].
- **ExStream** – Like REMIND, ExStream is a streaming learning method, however, it can only train fully connected layers of the network [22]. ExStream uses rehearsal by maintaining buffers of prototypes. It stores the input vector and combines the two nearest vectors in the buffer. After the buffer gets updated, all samples from its buffer are used to train the fully connected layers of a network. We use ExStream to train the final layer of the network, which is the only fully connected layer in ResNet-18.
- **SLDA** – Streaming Linear Discriminant Analysis (SLDA) is a well-known streaming method that was shown to work well on deep CNN features [23]. It maintains running means for each class and a running tied covariance matrix. Given a new input, it assigns the label of the closest Gaussian in feature space. It can be used to compute the output layer of a CNN.
- **iCaRL** – iCaRL is an incremental batch learning algorithm for CNNs [64]. iCaRL stores images from earlier classes for replay, uses a distillation loss to preserve weights, and uses a nearest class mean classifier in feature space.

- **Unified** – The Unified Classifier builds on iCaRL by using the outputs from the network for classification and introducing a cosine normalization layer, a constraint to preserve class geometry, and a margin ranking loss to maximize inter-class separation [27]. Unified also uses replay and distillation.
- **BiC** – The Bias Correction (BiC) method builds on iCaRL by using the output layer of the network for classification and correcting the bias from class imbalance during training, i.e., more new samples than replay samples [77]. The method trains two additional bias correction parameters on the output layer, resulting in improved performance over distillation and replay alone.
- **Offline** – The offline model is trained in a traditional, non-streaming setting and serves as an upper-bound on performance. We train two variants: one with only θ_F plastic and one with both θ_F and θ_G plastic.

Our main experiments focus on comparing state-of-the-art methods on ImageNet and we provide additional comparisons in Sec. 7. Although iCaRL, Unified, and BiC are traditionally trained in the incremental batch paradigm, we conduct experiments with these models in the streaming paradigm for fair comparison against REMIND. To train these streaming variants, we set the number of epochs to 1 and the batch size to $r + 1$ instances to match REMIND.

5.2 Model Configurations

In our setup, all models are trained instance-by-instance and have no batch requirements, unless otherwise noted. Because methods can be sensitive to the order in which new data are encountered, all models receive examples in the same order. The same base CNN initialization procedure is used by all models. For ExStream and SLDA, after base initialization, the streaming learning phase is re-started from the beginning of the data stream. All of the parameters except the output layer are kept frozen for ExStream and SLDA, whereas only $G(\cdot)$ is kept frozen for REMIND. All other comparison models do not freeze any layers and incremental training commences with the first new data sample. All models, except SLDA, are trained using cross-entropy loss with stochastic gradient descent and momentum. More parameter settings are in supplemental materials.

5.3 Datasets, Data Orderings, & Metrics

We conduct experiments with ImageNet and CORe50 by dividing both datasets into batches. The first batch is used for base initialization. Subsequently, all models use the same batch orderings, but they are sequentially fed individual samples and they cannot revisit any instances in a batch, unless otherwise noted. For ImageNet, the models are evaluated after each batch on all trained classes. For CORe50, models are evaluated on all test data after each batch.

ImageNet ILSVRC-2012 [68] has 1000 categories each with 732-1300 training samples and 50 validation samples, which we use for testing. During the base initialization phase, the model is trained offline on a set of 100 randomly selected classes. Following [13,27,64,77], each incremental batch then contains 100 random classes, which are not contained within any other batch. We study class incremental (class iid) learning with ImageNet.

CORe50 [52] contains sequences of video frames, with one object in each frame. It has 10 classes, and each sequence is acquired with varied environmental conditions. CORe50 is ideal for evaluating streaming learners since it is naturally non-iid and requires agents to learn from temporally correlated video streams. For CORe50, we follow [22] and sample at 1 frame per second, obtaining 600 training images and 225 test images per class. We use the bounding box crops and splits from [52]. Following [22], we use four training orderings to test the robustness of each algorithm under different conditions: 1) iid, where each batch has a random subset of training images, 2) class iid, where each batch has all of the images from two classes, which are randomly shuffled, 3) instance, where each batch has temporally ordered images from 80 unique object instances, and 4) class instance, where each batch has all of the temporally ordered instances from two classes. All batches have 1200 images across all orderings. Since CORe50 is small, CNNs are first initialized with pre-trained ImageNet weights and then fine-tuned on a subset of 1200 samples for base initialization.

We use the Ω_{all} metric [22,24,45] for evaluation, which normalizes incremental learning performance by offline performance: $\Omega_{\text{all}} = \frac{1}{T}\sum_{t=1}^{T}\frac{\alpha_t}{\alpha_{\text{offline},t}}$, where T is the total number of testing events, α_t is the accuracy of the model for test t, and $\alpha_{\text{offline},t}$ is the accuracy of the optimized offline learner for test t. If $\Omega_{\text{all}} = 1$, then the incremental learner's performance matched the offline model. We use top-5 and top-1 accuracies for ImageNet and CORe50, respectively. Average accuracy results are in supplemental materials (Table S2-S3).

5.4 Results: ImageNet

For ImageNet, we use the pre-trained PyTorch offline model with 89.08% top-5 accuracy to normalize Ω_{all}. We allow the iCaRL, Unified, and BiC models to store 10,000 (224×224 uint8) raw pixel image prototypes in a replay buffer, which is equivalent to 1.51 GB in memory. This allows REMIND to store indices for 959665 examples in its replay buffer. We set $r = 50$ samples. We study additional buffer sizes in Sec. 7. Results for incremental class learning on ImageNet are shown in Table 1 and a learning curve for all models is shown in Fig. 3. REMIND outperforms all other comparison models, with SLDA achieving the second best performance. This is remarkable since REMIND only updates θ_F, whereas iCaRL, Unified, and BiC all update θ_F and θ_G.

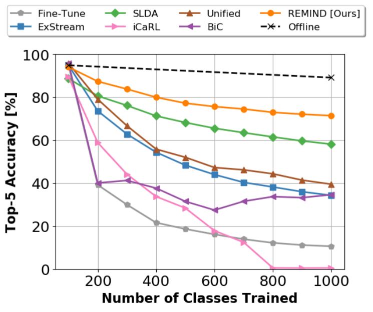


Fig. 3. Performance of streaming ImageNet models.

REMIND is intended to be used for online streaming learning; however, we also created a variant suitable for incremental batch learning which is described

Table 1. ResNet-18 streaming classification results on ImageNet and CORe50 using Ω_{all}. For CORe50, we explore performance across four ordering schemes and report the average of 10 permutations. Upper bounds are at the bottom.

	ImageNet	**CORe50**			
MODEL	CLS IID	IID	CLS IID	INST	CLS INST
Fine-Tune (θ_F)	0.288	0.961	0.334	0.851	0.334
ExStream	0.569	0.953	0.873	0.933	0.854
SLDA	0.752	0.976	0.958	0.963	0.959
iCaRL	0.306	-	0.690	-	0.644
Unified	0.614	-	0.510	-	0.527
BiC	0.440	-	0.410	-	0.415
REMIND	**0.855**	**0.985**	**0.978**	**0.980**	**0.979**
Offline (θ_F)	0.929	0.989	0.984	0.985	0.985
Offline	1.000	1.000	1.000	1.000	1.000

in supplemental materials. Incremental batch results for REMIND and recent methods are given in Fig. 1 and supplemental materials (Table S2 and Fig. S2). While incremental batch methods train much more slowly, REMIND achieves comparable performance to the best methods.

5.5 Results: CORe50

We use the CoRe50 dataset to study models under more realistic data orderings. Existing methods including iCaRL, Unified, and BiC assume that classes from one batch do not appear in other batches, making it difficult for them to learn the iid and instance orderings without modifications. To compute Ω_{all}, we use an offline model that obtains 93.11% top-1 accuracy. The iCaRL, Unified, and BiC models use replay budgets of 50 images, which is equivalent to 7.3 MB. This allows REMIND to store replay indices for 4465 examples. Results for other buffer sizes are in supplemental materials (Fig. S3). REMIND replays $r = 20$ samples. Ω_{all} results for CORe50 are provided in Table 1. For CORe50, REMIND outperforms all models for all orderings. In fact, REMIND is only 2.2% below the full offline model in the worst case, in terms of Ω_{all}. Methods that only trained the output layer performed well on CORe50 and poorly on ImageNet. This is likely because the CNNs used for CORe50 experiments are initialized with ImageNet weights, resulting in more robust representations. REMIND's remarkable performance on these various orderings demonstrate its versatility.

6 Experiments: Incremental VQA

In VQA, a system must produce an answer to a natural language question about an image [8,36,54], which requires capabilities such as object detection, scene understanding, and logical reasoning. Here, we use REMIND to pioneer streaming

VQA. During training, a streaming VQA model receives a sequence of temporally ordered triplets $\mathcal{D} = \{(X_t, Q_t, A_t)\}_{t=1}^{T}$, where X_t is an image, Q_t is the question (string), and A_t is the answer. If an answer is not provided at time t, then the agent must use knowledge from time 1 to $t-1$ to predict A_t. To use REMIND for streaming VQA, we store each quantized feature along with a question string and answer, which can later be used for replay. REMIND can be used with almost any existing VQA system (e.g., attention-based [6,46,78], compositional [7,28], bi-modal fusion [10,19,71]) and it can be applied to similar tasks like image captioning [12] and referring expression recognition [43,63,67].

6.1 Experimental Setup

For our experiments, we use the TDIUC [35] and CLEVR [33] VQA datasets. TDIUC is composed of natural images and has over 1.7 million QA pairs organized into 12 question types including simple object recognition, complex counting, positional reasoning, and attribute classification. TDIUC tests for generalization across different underlying tasks required for VQA. CLEVR consists of over 700000 QA pairs for 70000 synthetically generated images and is organized into 5 question types. CLEVR specifically tests for multi-step compositional reasoning that is very rarely encountered in natural image VQA datasets. We combine REMIND with two popular VQA algorithms, using a modified version of the stacked attention network (SAN) [42,78] for TDIUC, and a simplified version of the Memory Attention and Control (MAC) [28,55] network for CLEVR. A ResNet-101 model pre-trained on ImageNet is used to extract features for both TDIUC and CLEVR. REMIND's PQ model is trained with 32 codebooks each of size 256. The final offline mean per-type accuracy with SAN on TDIUC is 67.59% and the final offline accuracy with MAC on CLEVR is 94.00%. Our main results with REMIND use a buffer consisting of 50% of the dataset and $r = 50$. Results for other buffer sizes are in supplemental materials (Table S4).

For both datasets, we explore two orderings of the training data: iid and question type (q-type). For iid, the dataset is randomly shuffled and the model is evaluated on all test data when multiples of 10% of the total training set are seen. The q-type ordering reflects a more interesting scenario where QA pairs for different VQA 'skills' are grouped together. Models are evaluated on all test data at the end of each q-type. We perform base initialization by training on the first 10% of the data for the iid ordering and on QA pairs belonging to the first q-type for the q-type ordering. Then, the remaining data is streamed into the model one sample at a time. The buffer is then incrementally updated with PQ encoded features and raw question strings. We use simple accuracy for CLEVR and mean-per-type accuracy for TDIUC.

We compare REMIND to ExStream [22], SLDA [23], an offline baseline, and a simple baseline where models are fine-tuned without a buffer, which causes catastrophic forgetting. To adapt ExStream and SLDA for VQA, we use a variant of the linear VQA model in [34], which concatenates ResNet-101 image features to question features extracted from a universal sentence encoder [73] and then trains a linear classifier. Parameter settings are in supplemental materials.

6.2 Results: VQA

Streaming VQA results for REMIND with a 50% buffer size are given in Table 2. Variants of REMIND with other buffer sizes are in supplemental materials (Table S4). REMIND outperforms the streaming baselines for both datasets, with strong performance on both TDIUC using the SAN model and CLEVR using the MAC model. Interestingly, for CLEVR the results are much greater for q-type than for iid. We hypothesize that the q-type ordering may be acting as a natural curriculum [11], allowing our streaming model to train more efficiently. Our results demonstrate that it is possible to train complex, multi-modal agents capable of attention and compositional reasoning in a streaming manner. Learning curves and qualitative examples are in supplemental materials (Fig. S5-S6).

Table 2. Ω_{all} results for streaming VQA.

	TDIUC		CLEVR	
Ordering	iid	q-type	iid	q-type
Fine-Tune	0.716	0.273	0.494	0.260
ExStream	0.676	0.701	0.477	0.375
SLDA	0.624	0.644	0.518	0.496
REMIND	**0.917**	**0.919**	**0.720**	**0.985**
Offline	1.000	1.000	1.000	1.000

7 Additional Classification Experiments

In this section, we study several of REMIND's components. In supplemental materials, we study other factors that influence REMIND's performance (Fig. S4), e.g., where to quantize, number of codebooks, codebook size, and replay samples (r). In supplemental materials, we also explore the performance of iCaRL, Unified, and BiC when only θ_F is updated (Sec. S3.2).

REMIND Components. REMIND is impacted by the size of its overall buffer, using augmentation, and the features used to train $F(\cdot)$. We study these on ImageNet and results are given in Table 3. REMIND (Main) denotes the variant of REMIND from our main experiments that uses augmentation with a buffer size of 959665 and 32 codebooks of size 256. PQ is critical to performance, with PQ (32 codebooks) outperforming k-means (1 codebook) by 7.7% in terms of Ω_{all}. Augmentation is the next most helpful component and improves performance by 3.7%. Storing the entire dataset (100% Buffer) does not yield significant improvements. Using real features yields marginal improvements (1.3%) while requiring nearly 16 times more memory.

Table 3. REMIND variations on ImageNet with their memory (GB).

Variant	Ω_{all}	Memory
REMIND (Main)	0.855	1.51
100% Buffer	0.856	2.01
No Augmentation	0.818	1.51
k-Means	0.778	0.12
Real Features	0.868	24.08

Replay Buffer Size. Since REMIND and several other models rely on a replay buffer to mitigate forgetting, we studied performance on ImageNet as a function of buffer size. We compared the performance of iCaRL, Unified, and BiC on ImageNet at three different buffer sizes (5K exemplars=0.75GB, 10K exemplars=1.51GB, and 20K exemplars=3.01GB). To make the experiment fair, we compared REMIND to these models at equivalent buffer sizes, i.e., 479665 compressed samples=0.75GB, 959665 compressed samples=1.51GB, and 1281167 compressed samples (full dataset)=2.01 GB. In Fig. 4, we see that more memory generally results in better performance. Overall, REMIND has the best performance and is nearly unaffected by buffer size. A plot with incremental batch models is in supplemental materials (Fig. S2), and follows the same trend: larger buffers yield better performance.

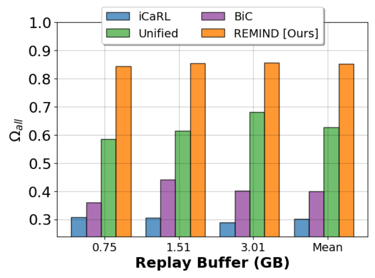


Fig. 4. Ω_{all} as a function of buffer size for streaming ImageNet models.

Regularization Comparisons. In Table 4, we show the results of REMIND and regularization methods for combating catastrophic forgetting on CORe50 class orderings. These regularization methods constrain weight updates to remain close to their previous values and are trained on batches of data, where each batch resembles a *task*. At test time, these models are provided with task labels, denoting which task an unseen sample came from. In our experiments, a *task* consists of several classes, and providing task labels makes classification easier. We analyze performance when task labels are provided and when they are withheld. To evaluate REMIND and Offline with task labels, we mask off probabilities during test time for classes not included in the specific task. Consistent with [14,17,45], we find that regularization methods perform poorly when no task labels are provided. Regardless, REMIND outperforms all comparisons, both with and without task labels.

Table 4. Ω_{all} for regularization models averaged over 10 runs on CORe50 with and without Task Labels (TL).

	CLS IID		CLS INST	
MODEL	TL	NO TL	TL	NO TL
SI	0.895	0.417	0.905	0.416
EWC	0.893	0.413	0.903	0.413
MAS	0.897	0.415	0.905	0.421
RWALK	0.903	0.410	0.912	0.417
A-GEM	0.925	0.417	0.916	0.421
REMIND	**0.995**	**0.978**	**0.995**	**0.979**
Offline	1.000	1.000	1.000	1.000

8 Discussion & Conclusion

We proposed REMIND, a brain-inspired replay-based approach to online learning in a streaming setting. REMIND achieved state-of-the-art results for object classification. Unlike iCaRL, Unified, and BiC, REMIND can be applied to iid and instance ordered data streams without modification. Moreover, we showed that REMIND is general enough for tasks like VQA with almost no changes.

REMIND replays compressed (lossy) representations that it stores, rather than veridical (raw pixel) experience, which is more consistent with memory consolidation in the brain. REMIND's replay is more consistent with how replay occurs in the brain during waking hours. Replay also occurs in the brain during slow wave sleep [9,31], and it would be interesting to explore how to effectively create a variant that utilizes sleep/wake cycles for replay. This could be especially beneficial for a deployed agent that is primarily engaged in online learning during certain hours, and is engaged in offline consolidation in other hours.

Several algorithmic improvements could be made to REMIND. We initialized REMIND's quantization model during the base initialization phase. For deployed, on-device learning this could instead be done by pre-training the codebook on a large dataset, or it could be initialized with large amounts of unlabeled data, potentially leading to improved representations. Another potential improvement is using selective replay. REMIND randomly chooses replay instances with uniform probability. In early experiments, we also tried choosing replay samples based on distance from current example, number of times a sample has been replayed, and the time since it was last replayed. While none performed better than uniform selection, we believe that selective replay still holds the potential to lead to better generalization with less computation. Because several comparison models used ResNet-18, we also used ResNet-18 for image classification so that we could compare against these models directly. The ResNet-18 layer used for quantization was chosen to ensure REMIND's memory efficiency, but co-designing the CNN architecture with REMIND could lead to considerably better results. Using less memory, REMIND stores far more compressed representations than competitors. For updating the replay buffer, we used random replacement, which worked well in [14,77]. We tried a queue and a distance-based strategy, but both performed nearly equivalent to random selection with higher computational costs. Furthermore, future variants of REMIND could incorporate mechanisms similar to [62] to explicitly account for the temporal nature of incoming data. To demonstrate REMIND's versatility, we pioneered streaming VQA and established strong baselines. It would be interesting to extend this to streaming chart question answering [37,38,40], object detection, visual query detection [2], and other problems in vision and language [39].

Acknowledgements. This work was supported in part by the DARPA/MTO Lifelong Learning Machines program [W911NF-18-2-0263], AFOSR grant [FA9550-18-1-0121], NSF award #1909696, and a gift from Adobe Research. We thank NVIDIA for the GPU donation. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies or endorsements of any sponsor. We thank Michael Mozer, Ryne Roady, and Zhongchao Qian for feedback on early drafts of this paper.

References

1. Abraham, W.C., Robins, A.: Memory retention–the synaptic stability versus plasticity dilemma. Trends in Neurosciences (2005)
2. Acharya, M., Jariwala, K., Kanan, C.: Vqd: Visual query detection in natural scenes. In: NAACL (2019)
3. Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., Tuytelaars, T.: Memory aware synapses: Learning what (not) to forget. In: ECCV. pp. 139–154 (2018)
4. Aljundi, R., Belilovsky, E., Tuytelaars, T., Charlin, L., Caccia, M., Lin, M., Page-Caccia, L.: Online continual learning with maximal interfered retrieval. In: NeurIPS. pp. 11849–11860 (2019)
5. Aljundi, R., Lin, M., Goujaud, B., Bengio, Y.: Gradient based sample selection for online continual learning. In: NeurIPS. pp. 11816–11825 (2019)
6. Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., Zhang, L.: Bottom-up and top-down attention for image captioning and visual question answering. In: CVPR (2018)
7. Andreas, J., Rohrbach, M., Darrell, T., Klein, D.: Neural module networks. In: CVPR. pp. 39–48 (2016)
8. Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C.L., Parikh, D.: VQA: Visual question answering. In: ICCV (2015)
9. Barnes, D.C., Wilson, D.A.: Slow-wave sleep-imposed replay modulates both strength and precision of memory. Journal of Neuroscience **34**(15), 5134–5142 (2014)
10. Ben-Younes, H., Cadene, R., Cord, M., Thome, N.: Mutan: Multimodal tucker fusion for visual question answering. In: ICCV (2017)
11. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum Learning. In: ICML. pp. 41–48 (2009)
12. Bernardi, R., Cakici, R., Elliott, D., Erdem, A., Erdem, E., Ikizler-Cinbis, N., Keller, F., Muscat, A., Plank, B.: Automatic description generation from images: A survey of models, datasets, and evaluation measures. Journal of Artificial Intelligence Research **55**, 409–442 (2016)
13. Castro, F.M., Marín-Jiménez, M.J., Guil, N., Schmid, C., Alahari, K.: End-to-end incremental learning. In: ECCV. pp. 233–248 (2018)
14. Chaudhry, A., Dokania, P.K., Ajanthan, T., Torr, P.H.: Riemannian walk for incremental learning: Understanding forgetting and intransigence. In: ECCV. pp. 532–547 (2018)
15. Chaudhry, A., Ranzato, M., Rohrbach, M., Elhoseiny, M.: Efficient lifelong learning with A-GEM. In: ICLR (2019)
16. Dhar, P., Singh, R.V., Peng, K.C., Wu, Z., Chellappa, R.: Learning without memorizing. In: CVPR. pp. 5138–5146 (2019)
17. Farquhar, S., Gal, Y.: Towards robust evaluations of continual learning. arXiv:1805.09733 (2018)
18. Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A.A., Pritzel, A., Wierstra, D.: Pathnet: Evolution channels gradient descent in super neural networks. arXiv:1701.08734 (2017)
19. Fukui, A., Park, D.H., Yang, D., Rohrbach, A., Darrell, T., Rohrbach, M.: Multimodal compact bilinear pooling for visual question answering and visual grounding. In: EMNLP (2016)
20. Gama, J.: Knowledge discovery from data streams. Chapman and Hall/CRC (2010)

21. Gama, J., Sebastião, R., Rodrigues, P.P.: On evaluating stream learning algorithms. Machine learning **90**(3), 317–346 (2013)
22. Hayes, T.L., Cahill, N.D., Kanan, C.: Memory efficient experience replay for streaming learning. In: ICRA (2019)
23. Hayes, T.L., Kanan, C.: Lifelong machine learning with deep streaming linear discriminant analysis. In: CVPRW (2020)
24. Hayes, T.L., Kemker, R., Cahill, N.D., Kanan, C.: New metrics and experimental paradigms for continual learning. In: CVPRW. pp. 2031–2034 (2018)
25. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
26. Hou, S., Pan, X., Change Loy, C., Wang, Z., Lin, D.: Lifelong learning via progressive distillation and retrospection. In: ECCV. pp. 437–452 (2018)
27. Hou, S., Pan, X., Wang, Z., Change Loy, C., Lin, D.: Learning a unified classifier incrementally via rebalancing. In: CVPR (2019)
28. Hudson, D.A., Manning, C.D.: Compositional attention networks for machine reasoning. In: ICLR (2018)
29. Insausti, R., Marcos, M., Mohedano-Moriano, A., Arroyo-Jiménez, M., Córcoles-Parada, M., Artacho-Pérula, E., Ubero-Martinez, M., Munoz-Lopez, M.: The non-human primate hippocampus: neuroanatomy and patterns of cortical connectivity. In: The hippocampus from cells to systems, pp. 3–36. Springer (2017)
30. Jegou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. TPAMI **33**(1), 117–128 (2010)
31. Ji, D., Wilson, M.A.: Coordinated memory replay in the visual cortex and hippocampus during sleep. Nature Neuroscience **10**(1), 100–107 (2007)
32. Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with gpus. IEEE Transactions on Big Data (2019)
33. Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C.L., Girshick, R.: Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In: CVPR (2017)
34. Kafle, K., Kanan, C.: Answer-type prediction for visual question answering. In: CVPR. pp. 4976–4984 (2016)
35. Kafle, K., Kanan, C.: An analysis of visual question answering algorithms. In: ICCV. pp. 1983–1991 (2017)
36. Kafle, K., Kanan, C.: Visual question answering: Datasets, algorithms, and future challenges. Computer Vision and Image Understanding (2017)
37. Kafle, K., Price, B., Cohen, S., Kanan, C.: Dvqa: Understanding data visualizations via question answering. In: CVPR. pp. 5648–5656 (2018)
38. Kafle, K., Shrestha, R., Cohen, S., Price, B., Kanan, C.: Answering questions about data visualizations using efficient bimodal fusion. In: WACV. pp. 1498–1507 (2020)
39. Kafle, K., Shrestha, R., Kanan, C.: Challenges and prospects in vision and language research. Frontiers in Artificial Intelligence **2**, 28 (2019)
40. Kahou, S.E., Michalski, V., Atkinson, A., Kádár, Á., Trischler, A., Bengio, Y.: Figureqa: An annotated figure dataset for visual reasoning. arXiv preprint arXiv:1710.07300 (2017)
41. Karlsson, M.P., Frank, L.M.: Awake replay of remote experiences in the hippocampus. Nature Neuroscience **12**(7), 913 (2009)
42. Kazemi, V., Elqursh, A.: Show, ask, attend, and answer: A strong baseline for visual question answering. arXiv:1704.03162 (2017)
43. Kazemzadeh, S., Ordonez, V., Matten, M., Berg, T.: Referitgame: Referring to objects in photographs of natural scenes. In: EMNLP. pp. 787–798 (2014)

44. Kemker, R., Kanan, C.: FearNet: Brain-inspired model for incremental learning. In: ICLR (2018)
45. Kemker, R., McClure, M., Abitino, A., Hayes, T.L., Kanan, C.: Measuring catastrophic forgetting in neural networks. In: AAAI (2018)
46. Kim, J.H., Jun, J., Zhang, B.T.: Bilinear attention networks. In: NeurIPS. pp. 1564–1574 (2018)
47. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., Hadsell, R.: Overcoming catastrophic forgetting in neural networks. PNAS (2017)
48. Konkel, A., Warren, D.E., Duff, M.C., Tranel, D., Cohen, N.J.: Hippocampal amnesia impairs all manner of relational memory. Frontiers in Human Neuroscience **2**, 15 (2008)
49. Le, T., Stahl, F., Gaber, M.M., Gomes, J.B., Di Fatta, G.: On expressiveness and uncertainty awareness in rule-based classification for data streams. Neurocomputing **265**, 127–141 (2017)
50. Lee, K., Lee, K., Shin, J., Lee, H.: Overcoming catastrophic forgetting with unlabeled data in the wild. In: ICCV. pp. 312–321 (2019)
51. Lewis, P.A., Durrant, S.J.: Overlapping memory replay during sleep builds cognitive schemata. Trends in Cognitive Sciences **15**(8), 343–351 (2011)
52. Lomonaco, V., Maltoni, D.: Core50: a new dataset and benchmark for continuous object recognition. In: CoRL. pp. 17–26 (2017)
53. Lopez-Paz, D., Ranzato, M.: Gradient episodic memory for continual learning. In: NeurIPS. pp. 6467–6476 (2017)
54. Malinowski, M., Fritz, M.: A multi-world approach to question answering about real-world scenes based on uncertain input. In: NeurIPS (2014)
55. Marois, V., Jayram, T., Albouy, V., Kornuta, T., Bouhadjar, Y., Ozcan, A.S.: On transfer learning using a mac model variant. arXiv:1811.06529 (2018)
56. McClelland, J.L., Goddard, N.H.: Considerations arising from a complementary learning systems perspective on hippocampus and neocortex. Hippocampus **6**(6), 654–665 (1996)
57. McCloskey, M., Cohen, N.J.: Catastrophic interference in connectionist networks: The sequential learning problem. Psychology of Learning and Motivation **24**, 109–165 (1989)
58. Nguyen, C.V., Li, Y., Bui, T.D., Turner, R.E.: Variational continual learning. In: ICLR (2018)
59. Ostapenko, O., Puscas, M., Klein, T., Jähnichen, P., Nabi, M.: Learning to remember: A synaptic plasticity driven framework for continual learning. In: CVPR (2019)
60. ONeill, J., Pleydell-Bouverie, B., Dupret, D., Csicsvari, J.: Play it again: reactivation of waking experience and memory. Trends in Neurosciences **33**(5), 220–229 (2010)
61. Parisi, G.I., Kemker, R., Part, J.L., Kanan, C., Wermter, S.: Continual lifelong learning with neural networks: A review. Neural Networks (2019)
62. Parisi, G.I., Tani, J., Weber, C., Wermter, S.: Lifelong learning of spatiotemporal representations with dual-memory recurrent self-organization. Frontiers in Neurorobotics **12**, 78 (2018)
63. Plummer, B.A., Wang, L., Cervantes, C.M., Caicedo, J.C., Hockenmaier, J., Lazebnik, S.: Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In: ICCV. pp. 2641–2649 (2015)

64. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: CVPR (2017)
65. Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., , Tesauro, G.: Learning to learn without forgetting by maximizing transfer and minimizing interference. In: ICLR (2019)
66. Ritter, H., Botev, A., Barber, D.: Online structured laplace approximations for overcoming catastrophic forgetting. In: NeurIPS. pp. 3738–3748 (2018)
67. Rohrbach, A., Rohrbach, M., Hu, R., Darrell, T., Schiele, B.: Grounding of textual phrases in images by reconstruction. In: ECCV. pp. 817–834. Springer (2016)
68. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. IJCV **115**(3), 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y
69. Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., Hadsell, R.: Progressive neural networks. arXiv:1606.04671 (2016)
70. Serra, J., Suris, D., Miron, M., Karatzoglou, A.: Overcoming catastrophic forgetting with hard attention to the task. In: ICML. pp. 4555–4564 (2018)
71. Shrestha, R., Kafle, K., Kanan, C.: Answer them all! toward universal visual question answering models. In: CVPR (2019)
72. Stickgold, R., Hobson, J.A., Fosse, R., Fosse, M.: Sleep, learning, and dreams: off-line memory reprocessing. Science **294**(5544), 1052–1057 (2001)
73. Subramanian, S., Trischler, A., Bengio, Y., Pal, C.J.: Learning general purpose distributed sentence representations via large scale multi-task learning. In: ICLR (2018)
74. Takahashi, S.: Episodic-like memory trace in awake replay of hippocampal place cell activity sequences. Elife **4**, e08105 (2015)
75. Teyler, T.J., Rudy, J.W.: The hippocampal indexing theory and episodic memory: updating the index. Hippocampus **17**(12), 1158–1169 (2007)
76. Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitliagkas, I., Courville, A., Lopez-Paz, D., Bengio, Y.: Manifold mixup: Better representations by interpolating hidden states. In: ICML (2019)
77. Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., Fu, Y.: Large scale incremental learning. In: CVPR. pp. 374–382 (2019)
78. Yang, Z., He, X., Gao, J., Deng, L., Smola, A.J.: Stacked attention networks for image question answering. In: CVPR (2016)
79. Yoon, J., Yang, E., Lee, J., Hwang, S.J.: Lifelong learning with dynamically expandable networks. In: ICLR (2018)
80. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: NeurIPS. pp. 3320–3328 (2014)
81. Zenke, F., Poole, B., Ganguli, S.: Continual learning through synaptic intelligence. In: ICML. pp. 3987–3995 (2017)

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

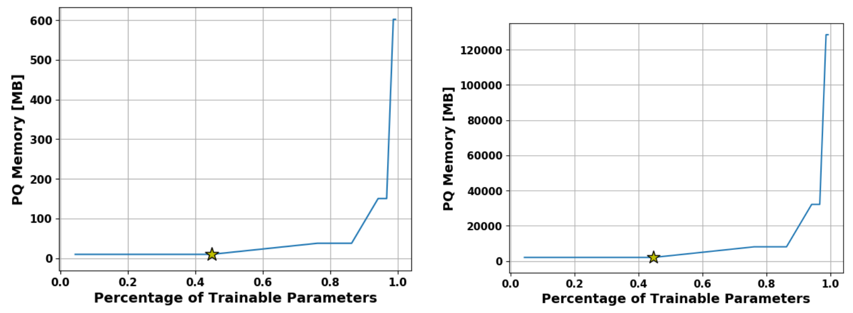


Fig. S1. Auxiliary storage required to store quantized CNN features for the entire dataset as a function of the percentage of ResNet-18 parameters used in the top of the CNN, $F(\cdot)$, which are updated during streaming learning in REMIND. Storage requirements are shown for CORe50 (left) and ImageNet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

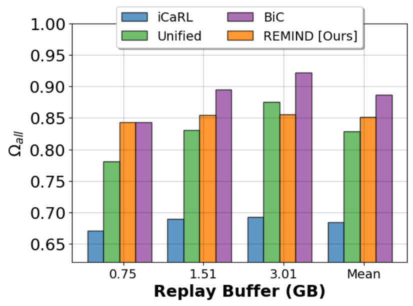
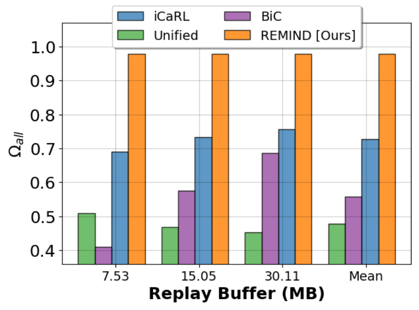


Fig. S2. Performance as a function of buffer size for various batch comparison models on ImageNet.

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.



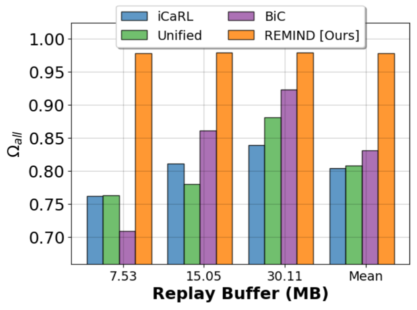


Fig. S3. Performance as a function of buffer size for various streaming (left) and batch (right) comparison models on CORe50. Each bar is the average over 10 permutations.

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.

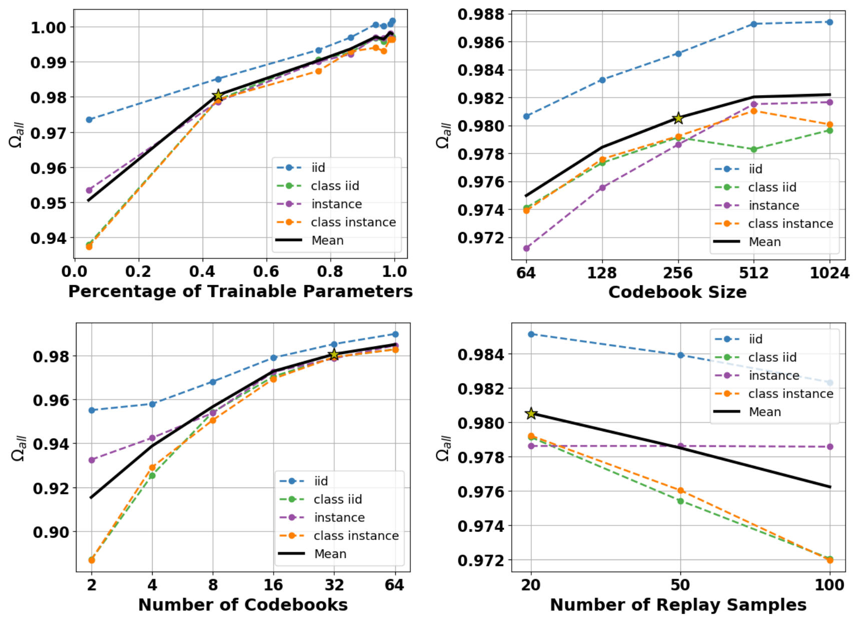


Fig. S4. Additional experiments with REMIND on CORe50. From left to right, top to bottom, performance as a function of: 1) trainable parameters, 2) codebook size, 3) number of codebooks, and 4) number of replay samples (r). The values used for our main experiments are denoted with a yellow star and each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

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

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. REMINDs results are statistically significantly different from each of the comparison models for all four data orderings according to a Students 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%

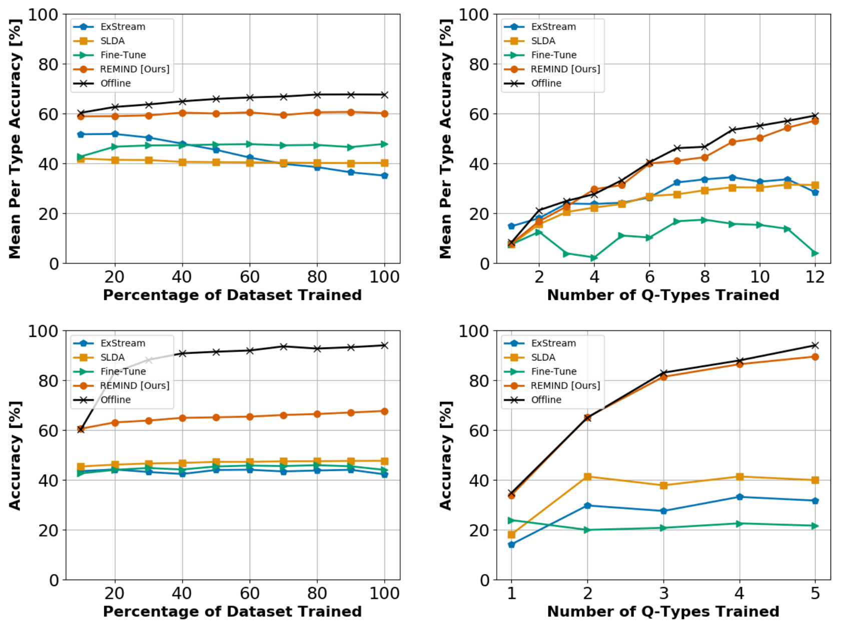
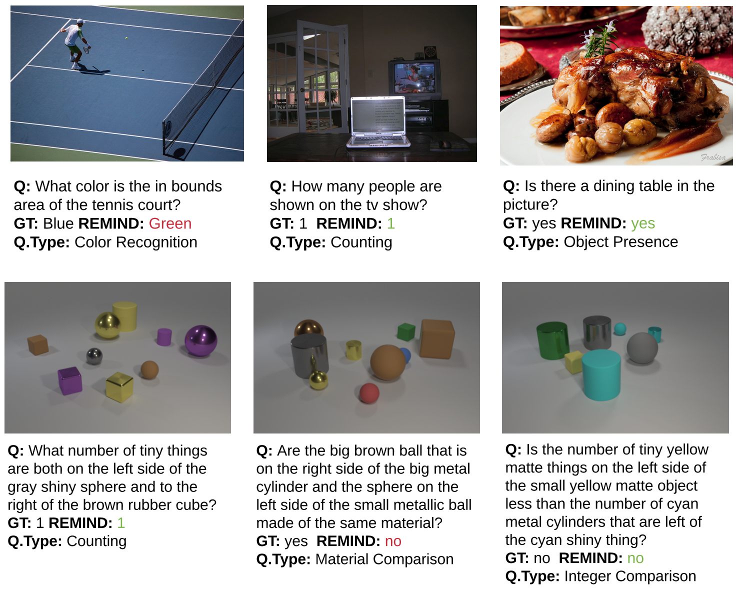


Fig. S5. Learning curves for each ordering of the TDIUC (top row) and CLEVR (bottom row) datasets. We provide curves from the REMIND model trained with 50% buffer size.

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.