

Hypothesis-driven Online Video Stream Learning with Augmented Memory

Mengmi Zhang,^{1,2,*} Rohil Badkundri,^{1,2,3,*} Morgan B. Talbot,^{2,4} Rushikesh Zawar,⁵ and Gabriel Kreiman^{1,2}

* Equal contribution

¹ Children’s Hospital, Harvard Medical School

² Center for Brains, Minds and Machines

³ Harvard University

⁴ Harvard-MIT Health Sciences and Technology, Harvard Medical School

⁵ Birla Institute of Technology and Science, Pilani

Address correspondence to gabriel.kreiman@tch.harvard.edu

Abstract

The ability to continuously acquire new knowledge without forgetting previous tasks remains a challenging problem for computer vision systems. Standard continual learning benchmarks focus on learning from static iid images in an offline setting. Here, we examine a more challenging and realistic online continual learning problem called online stream learning. Like humans, some AI agents have to learn incrementally from a continuous temporal stream of non-repeating data. We propose a novel model, Hypotheses-driven Augmented Memory Network (HAMN), which efficiently consolidates previous knowledge using an augmented memory matrix of "hypotheses" and replays reconstructed image features to avoid catastrophic forgetting. Compared with pixel-level and generative replay approaches, the advantages of HAMN are two-fold. First, hypothesis-based knowledge consolidation avoids redundant information in the image pixel space and makes memory usage far more efficient. Second, hypotheses in the augmented memory can be re-used for learning new tasks, improving generalization and transfer learning ability. Given a lack of online incremental class learning datasets on video streams, we introduce and adapt two additional video datasets, Toybox and iLab, for online stream learning. We also evaluate our method on the CORE50 and online CIFAR100 datasets. Our method performs significantly better than all state-of-the-art methods, while offering much more efficient memory usage. All source code and data are publicly available at <https://github.com/kreimanlab/AugMem>

1. Introduction

The world is not stationary. To thrive in evolving environments, humans are capable of continual acquisition and

transfer of new knowledge, from a *stream of highly temporally correlated visual stimuli over multiple tasks*, while retaining previously learnt experiences [15, 45]. In standard incremental class continual learning problems, neural networks are presented with stationary images that are independent and identically distributed (iid), with multiple presentations of each image within a task [12, 33, 37]. Mimicking human learning experiences, here we tackle a more challenging and realistic variation of incremental class learning, named *online stream learning*, with two salient characteristics: (a) the input is in the form of video streams where the data are temporally highly correlated; and (b) during online learning, data are presented only once and no repeated visits over the old data are allowed within a task.

In online stream learning, AI systems tend to fail to retain good performance across previously learnt tasks [16, 21, 32]. Numerous methods for alleviating catastrophic forgetting have been proposed, the simplest being to jointly train models on both old and new tasks, which demands a large amount of resources to store previous training data and hinders learning of novel data in real time.

Inspired by memory-augmented networks in one-shot image classification [41], memory retrieval [13], and video prediction [14], we propose the Hypothesis-driven Augmented Memory Network (HAMN) for online stream learning in image classification tasks. The network learns a set of hypotheses in the augmented memory, such that the latent representation of an image can be constructed by a linear combination of the hypotheses. Multi-head content-based attention allows HAMN to share the augmented memory over multiple image features, further compressing representations in memory and allowing features to be reconstructed from hypotheses in parallel. Hypothesis-sharing in the augmented memory also enables transfer learning and generalization to new tasks. The hypotheses

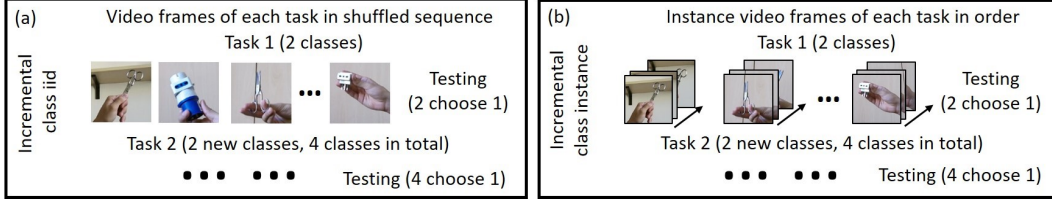


Figure 1. **Schematics of two online stream learning protocols in the incremental class settings: learning with class iid data (a) and class instance data (b).** In the incremental class setting, for each task, the model has to learn to classify 2 new classes while training for a single epoch. During testing, the model has to classify images from all seen classes without knowing task identity. In incremental class iid, within a task, the sequence of images is randomly shuffled, whereas in class instance, the temporal order of images is preserved. See **Online Stream Learning Protocols** for descriptions.

learned in previous tasks are stored in the augmented memory, enabling HAMN to classify new objects.

To prevent catastrophic forgetting, we impose three constraints: (i) To preserve encoded hypotheses in augmented memory, we regularize the network to write new hypotheses into rarely-used locations in memory. (ii) We store the memory indices activated by samples from previous tasks, reconstruct these samples from the corresponding hypotheses, and replay the constructed samples during training. (iii) We apply logit matching [38] during replay to maintain the learnt class distributions from all previous tasks.

We evaluated our method under two typical online stream learning protocols (**Fig. 1**), incremental class iid and incremental class instance on the CORE50 dataset [30]. Because there is a lack of online video stream datasets, we introduce two additional video stream datasets for object recognition, Toybox [48] and iLab [5], by adapting them to the online stream learning setting. In addition, to evaluate the online incremental class setting over long-range tasks, we also include results on the standard online CIFAR100 dataset [24] consisting of 20 tasks. HAMN is better than state-of-the-art (SOTA) methods across all four datasets, ranging from short-range to long-range tasks, and from static images to video streams, while being more memory-efficient than other methods. HAMN demonstrates memory retention and adaptation to new tasks, even while going through continuous presentation of training examples only once.

2. Related Works

Online stream learning approaches can be categorized into (1) weight regularization, and (2) replay methods. Most weight regularization methods store weights trained on previous tasks and impose constraints on weight updates for new tasks [8, 17, 22, 25, 26, 52]. For example, Elastic Weight Consolidation (EWC) [22] stores parameters, estimates their importance for previous tasks, and penalizes changes on new tasks. Selecting “important” parameters for previous tasks complicates implementation by necessitating exhaustive hyper-parameter tuning. Besides, storing the importance of the millions of parameters required by SOTA

recognition models across all previous tasks is costly [49]. Learning without Forgetting (LwF) mitigates this problem by storing only logits from previous tasks, but its performance is inferior to many replay methods. Stable SGD [34] studied the effect of dropout, learning rate decay, and batch sizes on widening the tasks’ local minima in previous tasks and minimizing catastrophic forgetting.

In replay methods, a subset of images or features from a previous task are stored or generated and later shown to the model to prevent forgetting [2, 38, 50]. Compared with weight regularization, replay methods generally lead to enhanced performance [35, 38] and reduced memory storage [8, 31]. Following prototypical contrastive learning [44], subsequent replay methods regularize network parameters with evolving prototypes and constrain their relations with inter and intra class samples [10, 53].

Raw-pixel replay, where images from previous tasks are stored and replayed, involves redundant information and is memory-inefficient. Recent works [2, 4] enhance sample selection in the replay buffer by maximizing diversity or uncertainty and approximating feasible regions in old tasks. Relying on limited sets of replay images can also lead to overfitting. The Bias Correction Method [50] applies a linear model with validation set replays from old tasks to avoid the imbalanced data distribution between old and new tasks. The adaptive aggregation network [28] introduces two types of residual blocks to balance plasticity and stability dynamically. Our method improves generalization by learning discriminative, information-rich hypotheses in a latent space, which can then be re-used to construct multiple new samples for replay and knowledge transfer to new tasks.

To limit storage, generative replay systems complement new tasks with “pseudo-data” that resemble historical data. [29, 39, 43]. Deep Generative Replay [43] is a Generative Adversarial Network that synthesizes training data over all previously learnt tasks. Generative approaches have succeeded with simple and artificial datasets but have failed with more complex inputs [3]. Later work uses bi-level optimization to generate mnemonics for replay [29]. All of these generative models work at the pixel-level. Subsequent works propose to generate examples at the feature lev-

els [27, 42, 46]. However, the generative models needed to create adequate synthetic data tend to be large and memory-intensive [49].

Previous work (REMIND) [16] showed that replay of latent features using memory indexing is efficient at preventing catastrophic forgetting. However, REMIND relies on Product Quantization [20], where a codebook for constructing the memory is pre-computed and fixed. This hinders adaptability to new tasks. Instead of fixed codebooks, our method learns a set of discrete hypotheses on the fly and replays constructed samples from the learnt hypotheses based on a memory indexing mechanism learnt through new tasks.

3. Online Stream Learning Setups

[Protocols] We consider two incremental class settings for the online stream learning protocols [16] (Fig. 1):

Class-iid: Images are randomly shuffled within each task but not interspersed among tasks, and are shown only once.

Class-instance: Each class contains short video clips of different objects. In the CORE50 dataset, the clips are presented one after the other in random order. This introduces a strong biasing effect of seeing large numbers of images from the same class in sequence, making stream learning highly challenging. Thus, in the Toybox and iLab datasets, we provide a complementary test of stream learning capabilities, finding a mid-point of difficulty between class-iid and class-instance. On these datasets, the image order alternates on each frame between two clips from different classes, while preserving the ordering of each clip.

[Datasets] In each video dataset, we used different data/task orderings for each training run. A global holdout test set of frame sequences was used for all training runs.

The CORE50 dataset [30] contains images of 50 objects in 10 classes. Each object has 11 instances, which are 15 second video clips of the object under particular conditions and poses. We followed [16] in the ordering, training and testing data splits.

Due to a lack of video datasets for object recognition in online stream learning, we introduce 2 extra video datasets, originally used for studying object transformations.

The Toybox dataset [48] contains videos of toy objects from 12 classes. We used a subset of the dataset containing 348 toy objects with 10 instances per object, each containing a spatial transformation of that object’s pose. We sampled each instance at 1 frame per second resulting in 15 images per instance per object. We chose 3 of the 10 instances for our test set, leaving 7 instances for training.

The iLab dataset [5] contains videos of toy vehicles. We used a subset of the dataset containing 392 vehicle objects in 14 classes, with 8 backgrounds (instances) per object and 15 images per instance. We chose 2 of the 8 instances per object for our test set.

To increase number of classes in online incremental class

learning, we included the standard image dataset **Online-CIFAR100 [24]**. The dataset is split up into 20 tasks with 5 classes each. All training images within a task are randomly shuffled and presented only once to all models.

4. Our proposed model (HAMN)

4.1. Overview

HAMN consists of a 2D convolutional neural network (2D-CNN) coupled with an augmented memory bank (Fig. 2). The memory matrix stores a fixed number of hypotheses. HAMN extracts feature maps from an image at an intermediate layer of the 2D-CNN. The reading attention mechanism guides the heads to select a linear combination of hypotheses, with higher attention assigned to memory slots with content more similar to the feature maps. To further compress the hypotheses in the augmented memory while maintaining rich representations, we use multi-head reading attention to interact with the memory bank. The number of plausible feature maps increases exponentially with the number of reading heads and hypotheses, enabling a diverse representation space. The latter part of the 2D-CNN combines the constructed feature maps with the hypotheses in the memory for classification. To ensure the memory bank learns useful hypotheses for replay in later tasks, in addition to a classification loss, we imposed a logistic loss, commonly used in knowledge distillation [18], based on the soft class distribution predicted from the constructed and the extracted feature maps, respectively.

4.2. Feature Extraction and Classification

The 2D-CNN contains two nested functions: $F(\cdot)$ for feature extraction, with parameters θ_F , consisting of the first few convolution layers; and $P_t(\cdot)$ for classification, with parameters θ_{P_t} at task t , consisting of the last few convolution layers. Since early convolution layers in 2D-CNNs are highly transferable [51], θ_F is trained for object recognition using ImageNet [11] and then fixed over all tasks. The parameters θ_{P_t} in $P_t(\cdot)$ depend on task t . After pre-training on ImageNet, we fine-tune $P_t(\cdot)$ such that HAMN learns new sets of hypotheses and decision boundaries incrementally to classify new object classes.

Up to task t , HAMN has seen a total of C_t classes. Given an image I_t shown during task t , we define the output tensor Z_t from the feature extraction step as $Z_t = F(I_t)$. Z_t is of size $S \times W \times H$ for channels, width, and height respectively. The output feature maps Z_t can then be used to produce a logit vector $q_t = P_t(Z_t)$. The logit vector $q_t \in \mathbb{R}^{C_t}$ contains the activation values over all C_t seen classes and is passed as input to the *softmax* function, which outputs the predicted probability vector $p_t \in \mathbb{R}^{C_t}$ over all C_t classes.

We used $p_t(c)$ and $q_t(c)$ to denote the predicted probability and logit value for class c respectively. We define

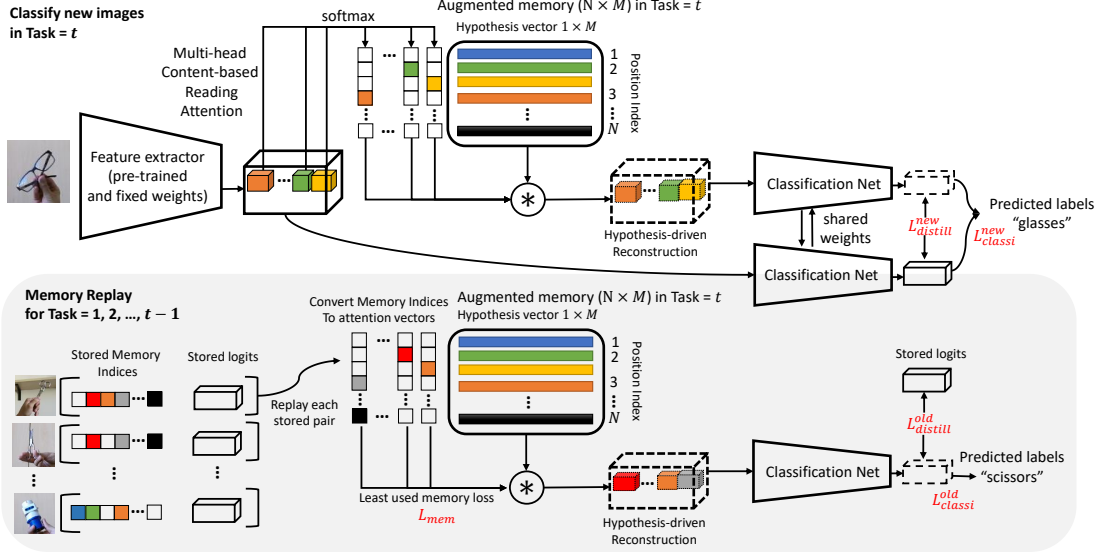


Figure 2. **Schematic illustration of the proposed hypotheses-driven Augmented Memory Network (HAMN) for online stream learning tasks.** HAMN model consists of a 2D-CNN and an augmented memory containing a set of hypotheses. The feature extractor in the 2D-CNN acts as a controller and decides which hypotheses to retrieve from the memory based on a content-similarity addressing mechanism. To further compress the memory and increase representation variety, multi-head reading attention is used. The re-constructed feature maps from the augmented memory (dashed line box) can be used for classification loss L_{classi}^{new} . To ensure these hypothesis-driven feature maps are as discriminative as the output of the CNN feature extractor, a logit distillation loss $L_{distill}^{new}$ is used between the two. To avoid catastrophic forgetting, we stored the memory indices of retrieved hypotheses along with the logits from previous tasks. During replay (shaded area), we used the stored memory indices to re-construct feature maps for classification loss L_{classi}^{old} and stored logits for regularization based on $L_{distill}^{old}$. Least used memory loss L_{mem} is imposed to prevent HAMN from over-writing the frequently used hypotheses from previous tasks.

$L_{classi}(\cdot)$ as the cross-entropy loss between p_t and its corresponding ground truth class label y_c :

$$L_{classi}(p_t, y_c) = - \sum_{i=1}^{C_t} \delta(i - y_c) \log(p_t(i))$$

where $\delta(x) = 1$ if $x = 0$ and $\delta(x) = 0$ otherwise.

4.3. Augmented Memory

We define M_t as the contents of the $N \times M$ memory bank at task t . There are N memory slots, each storing a hypothesis vector of dimension M . Thus, we have $M_t = [M_t(1), M_t(2), \dots, M_t(i), \dots, M_t(N)]$ where $M_t(i)$ is a hypothesis vector indexed by i .

Multi-head content-based reading attention: The feature extractor $F(\cdot)$ serves as a controller and outputs the tensor Z_t as the reading key of image I_t . Based on the similarity between Z_t and each hypothesis in the memory M_t , a content-based memory addressing mechanism draws a hypothesis from M_t . One could store the individual Z_t of each image I_t as a hypothesis in the memory bank and replay each hypothesis to prevent catastrophic forgetting, but this requires extensive memory resources. To further compress and remove redundancies in Z_t , we used a multi-head reading attention mechanism. For each image, the feature tensor Z_t of size $S \times W \times H$ is split into groups, each of which interacts with the shared M_t . We partition Z_t along

the S channels into D groups with each group d of size $S/D \times W \times H$. A hypothesis should represent a local concept and be location-invariant. Thus, Z_t consists of multiple reading keys $z_{t,d,l}$ indexed by group d and spatial location l on Z_t and $l \in \{1, 2, \dots, L = W \times H\}$:

$$Z_t = \{z_{t,1,1}, \dots, z_{t,d,l}, \dots, z_{t,D,L}\}, \quad z_{t,d,l} \in \mathbb{R}^{S/D} \quad (1)$$

For content-based addressing, each reading head compares its reading key $z_{t,d,l}$ with each hypothesis $M_t(i) \in \mathbb{R}^{S/D}$ by a similarity measure $K[\cdot, \cdot]$. To discourage attention blurring over all hypotheses, each reading head bundles with a hyperparameter of a positive constant value β . β denotes the attention sharpening strength and it can amplify or attenuate the precision of hypothesis selection. This mechanism produces a reading attention vector $w_{t,d,l}$ over N locations based on the content similarity, defined as the inner product $K[u, v] = \langle u, v \rangle$.

$$w_{t,d,l}(i) = \frac{e^{\beta K[z_{t,d,l}, M_t(i)]}}{\sum_j e^{\beta K[z_{t,d,l}, M_t(j)]}} \quad (2)$$

The overall reading attention w_t for all reading heads can then be written as $w_t = \{w_{t,1,1}, \dots, w_{t,d,l}, \dots, w_{t,D,L}\}$.

Construction from multiple hypotheses: The retrieved content vector $r_{t,d,l}$ for its reading key $z_{t,d,l}$, of size M , is computed as the expectation of sampled hypotheses modulated by the reading attention vector $w_{t,d,l}$:

$$r_{t,d,l} = \sum_i w_{t,d,l}(i) M_t(i) \quad (3)$$

We define the final feature tensor \tilde{Z}_t , from a set of hypotheses in the memory bank M_t , as the content retrieved using each reading key $\tilde{Z}_t = \{r_{t,1,1}, \dots, r_{t,d,l}, \dots, r_{t,D,L}\}$.

To enforce that the hypothesis-driven tensor \tilde{Z}_t has discriminative representations as Z_t , we perform classification on \tilde{Z}_t using the classifier $P_t(\cdot)$ attached with softmax. In addition, we introduce the distillation loss $L_{distill}$ on the logits q_t and \tilde{q}_t of Z_t and \tilde{Z}_t respectively:

$$L_{distill}(q_t, \tilde{q}_t) = \sum_{i=1}^{C_t} q_t(i) \log(\tilde{q}_t(i)) + (1 - q_t(i)) \log(1 - \tilde{q}_t(i)) \quad (4)$$

Note that although HAMN makes class predictions based on both \tilde{Z}_t and Z_t , we use the predictions from \tilde{Z}_t as the final predicted result during testing.

4.4. Avoiding catastrophic forgetting

In the incremental class setting (see **Protocols**), HAMN is presented with images I_t from new classes c^{new} in task t where c^{new} belongs to the complement set of $\{1, \dots, c^{old}, \dots, C_{t-1}\}$ in $\{1, \dots, c^{old}, \dots, c^{new}, \dots, C_t\}$. We take three measures below to avoid catastrophic forgetting.

Rarely-used memory locations: Every component in HAMN is differentiable, enabling hypothesis learning via gradient descent. For each new image I_t in task t , HAMN learns a new set of hypotheses in M_t . The update of hypotheses could undesirably occur in recently-used memory locations from previous tasks. To emphasize accurate encoding of new hypotheses and preserve the old hypotheses in most-used memory locations learnt from the previous task $t - 1$, we keep track of the top k most-used memory locations for each reading attention vector and aggregate those locations over all reading heads:

$$A_{t-1} = \bigcup_{I_{t-1}} \bigcup_D \bigcup_L m(w_{t-1,d,l}, k) \quad (5)$$

where $m(\cdot)$ returns the top k indices with largest attention values in $w_{t-1,d,l}$. Thus, we can define the memory usage loss $L_{mem} = \delta(A_{t-1})(M_{t-1} - M_t)^2$. An indicator function $\delta(\cdot)$ returns a binary vector of size N where the vector element at index i is 1 if $i \in A_{t-1}$ and 0 otherwise.

Hypothesis replay: Replay using memory indexing is more efficient in preventing catastrophic forgetting than using pixels [16]. We store the reading attention w_{t-1} from the previous task $t - 1$ and re-construct the feature tensor \tilde{Z}_{t-1} from M_t , which is replayed in the current task t . To further compress memory usage, we store the index with the largest attention value in $w_{t-1,d,l}$ for each reading head:

Algorithm 1 HAMN at task t

Input: stored $X = 200$ or $X = 2000$ logit vectors q_{t-1}^s , corresponding top-1 attention index tensors w_{t-1}^s and their class labels, a binary vector $\delta(A_{t-1})$ of dim N tracking the mostly used memory locations over previous tasks, new training images I_t from new classes

Training:

for batch **in** training images **do**

Train with I_t based on $L_{distill}$ & L_{classi} on Z_t & \tilde{Z}_t

if $t > 1$ **then**

Randomly sample x out of X and replay:

Re-construct hypothesis-driven \tilde{Z}_{t-1}^s using w_{t-1}^s

Train $P_t(\cdot)$ based on $L_{distill}$ using q_{t-1}^s and L_{classi}

Regularize M_t using L_{mem} on $\delta(A_{t-1})$

end if

end for

Testing:

for batch **in** testing images **do**

Compute p_t using $P_t(M_t(F(\cdot)))$ on test image based on \tilde{Z}_t

end for

$$w_{t-1}^s = \{m(w_{t-1,1,1}, 1), \dots, m(w_{t-1,d,l}, 1), \dots, m(w_{t-1,D,L}, 1)\} \quad (6)$$

We can approximate w_{t-1} with a one-hot vector generated using $\delta(w_{t-1}^s)$ and construct \tilde{Z}_{t-1}^s using M_t for replay. Storing only the top-1 index in each reading head greatly reduces its memory usage from $N \times D \times W \times H$ to $D \times W \times H$.

Distillation across tasks: Similar to the work [38], we use a distillation loss to transfer knowledge from the same neural network between different tasks to ensure that the discriminative information learnt previously is not lost in the new task t . Thus, given the constructed feature tensor \tilde{Z}_{t-1}^s and its stored logits vector q_{t-1}^s , we compute its distillation loss $L_{distill}(q_{t-1}^s, P_t(\tilde{Z}_{t-1}^s))$ using **Eq. 4**.

Sampling strategy for replays: Some algorithms select representative image examples to store and replay based on different scoring functions [6, 9, 23]. However, random sampling uniformly across classes yields outstanding performance in continual learning tasks [49]. Hence, we adopt a random sampling strategy and store X logits and top-1 reading attention tensor pairs (q_{t-1}^s, w_{t-1}^s) corresponding to images I_{t-1}^{old} from old classes c^{old} of previous tasks. Depending on the number of seen C_{t-1} classes, the storage for each old class contains X/C_{t-1} pairs. To prevent catastrophic forgetting, we define the total loss:

$$L_{total} = \gamma L_{mem} + \alpha \sum_{I_{t-1}^{old}} (L_{classi}(p_t^{old}, y_c^{old}) + L_{distill}(q_{t-1}^s, P_t(\tilde{Z}_{t-1}^s))) + \sum_{I_t^{new}} (L_{classi}(p_t^{new}, y_c^{new}) + L_{distill}(q_t, \tilde{q}_t)) \quad (7)$$

where α and γ are regularization hyperparameters.

4.5. Implementation and training details

We used the SqueezeNet [19] architecture as the backbone. Following PyTorch [36] conventions for SqueezeNet,

the feature extractor $F(\cdot)$, as the controller, includes convolution blocks up to layer 12 of SqueezeNet and the classification network $P_t(\cdot)$ includes the remaining layers. The tensor Z_t after layer 12 is of size $S = 512 \times (W = 13) \times (H = 13)$. We split Z_t along the S axis into $D = 512/M$ groups, resulting in $13 \times 13 \times (512/M)$ reading keys of length M . Depending on the number of tasks to learn in each dataset, we defined $N = 100, 100, 1000, 1000$ and $M = 8, 8, 128, 128$ respectively for CORE50, iLab, Toybox, and CIFAR100. We initialized the memory bank randomly before training. We tried initializing the memory bank with normalized k-means clustered centers trained on CIFAR100 [24] but there were no improvements. In video datasets, given that each reading head has 100 hypotheses to read from, HAMN could construct $100^{13 \times 13 \times (512/M)}$ feature maps, creating a rich latent space. We chose $k = 1$, taking the top-1 attention index in each reading head, to compute the aggregated attention vector A_{t-1} for regularizing memory using L_{mem} . HAMN stores $X = 200$ old samples for CORE50, iLab, and Toybox, and $X = 2000$ for CIFAR100, which has a higher number of classes. Empirically, we set the following hyperparameter values: $\beta = 5$, $\gamma = 1000$, $\alpha = 5$. Pseudocode is shown in **Algorithm 1**.

5. Experimental Details

5.1. Baselines

We compared HAMN with continual learning methods in several categories. To control for the effect of network architecture on performance, we used SqueezeNet [19] pre-trained on ImageNet [11] as the backbone for the majority of the stream learning algorithms that we used as baselines, except in cases where algorithm designs caused the substitution of SqueezeNet to require extensive modifications. Unlike the other methods, HAMN introduces an augmented memory bank between intermediate layers of SqueezeNet. Due to the additional, randomly-initialized parameters introduced by the augmented memory, we trained HAMN for multiple epochs on only the first task to allow HAMN to achieve similar performance to other methods on the first task, enabling a fair comparison on subsequent tasks.

Parameter Regularization Methods: We compared against Elastic Weight Consolidation (EWC) [22], Synaptic Intelligence (SI) [52], Memory Aware Synapses (MAS) [1], Learning without Forgetting (LwF) [26], Stable SGD [34], and naive L2 regularization (L2) where the L2 distance indicating parameter changes between tasks is added to the loss [22]. Due to varying source code availability, some algorithms were re-adapted for online stream learning.

Memory Distillation and Replay Methods: We compared against Gradient Episodic Memory (GEM) [31], Averaged GEM (AGEM) [8], Incremental Classifier and Representation Learner (iCARL) [38], Bias Correction method (BIC) [50], Gradient sample selection (GSS) [2], Continual Pro-

totype Evolution (CoPE) [10], Adaptive Aggregation Network (AAN) [28], REMIND (REM) [16], and Rainbow Memory (RM) [4].

Lower bound is trained sequentially over all tasks without any measures to avoid catastrophic forgetting.

Upper bound is trained on shuffled images from both the current task and all previous tasks over multiple epochs.

Chance predicts class labels by randomly choosing 1 out of the total of C_t classes up to current task t .

We report Top-1 classification accuracy on the current task for all seen C_t classes after 10 runs per protocol. See average top-1 accuracy over all tasks in Supp. material.

5.2. Memory Footprint Comparison

SqueezeNet has $J \approx 1.2 \times 10^6$ parameters. Weight regularization methods other than LwF and StableSGD have to store importance values for J parameters for each task, causing memory requirements to grow linearly with the number of tasks. In more challenging classification tasks, the network size and memory usage tend to increase. To provide a fair comparison with the weight regularization methods, we allocated a comparable amount of memory to the replay methods for storing examples to replay in subsequent tasks. Since image replay methods require storing at least one image from each previous class, we store 15 images for all classes for all raw pixel replay methods over all 3 video stream datasets and 100 images for CIFAR100.

To illustrate memory usage, consider CORE50 in the class instance protocol. Weight regularization methods require memory of size $5J$, i.e., about 6 million parameters for the 5 tasks. The dimension of the logit vector changes over tasks (i.e., 2, 4, etc.). For simplicity, we treat any HAMN logit vector q_{t-1}^s of constant dimension 10 for all classes in the 5 tasks. The top-1 attention index tensor w_{t-1}^s is of constant size $D \times W \times H = 10, 816$ parameters. We store $X = 200$ old pairs of logit and attention indices, resulting in 2 million parameters - equivalent to storing ≈ 13 images of size $3 \times 224 \times 224$ - which is 2 images fewer than raw pixel replay methods. In CIFAR100, our memory usage is even more dramatically reduced to **9** times smaller than the replay methods. Compared with parameter regularization methods, HAMN uses only one third as much memory in CORE50 and **11** times less memory in CIFAR100.

6. Results and Discussion

6.1. Online Stream Learning Performance

An ideal online stream learning method should avoid catastrophic forgetting while adapting to new tasks. A trivial algorithm that learns the first task and stops learning thereafter could have perfect memory for Task 1, but fail to classify new objects in subsequent tasks. **Fig. S2** shows the results of continual learning methods on the CORE50 dataset over 5 tasks across two stream learning protocols (See Supp.

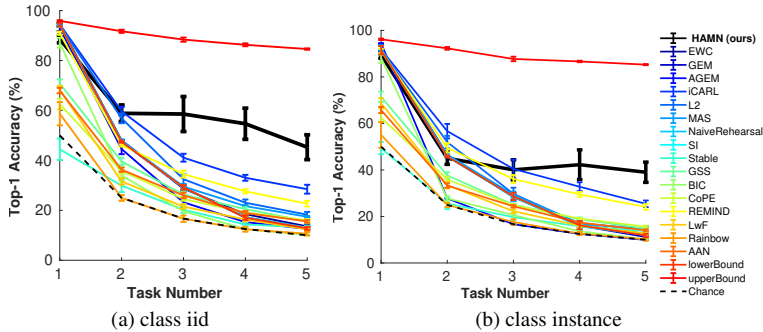


Figure 3. **Top-1 classification accuracy over all tasks in class iid and instance stream learning protocols on CORE50.** See **Protocols** for experimental protocols and datasets, **Baselines** for baselines, and Supp. material for results on Toybox and iLab. HAMN is the bolded black line. Error bars denote root mean squared error (RMSE) over 10 runs.

material for results in individual tasks on the Toybox and iLab datasets). We also report the averaged top-1 classification accuracy over all learnt classes across all tasks on Toybox and iLab datasets in the summary **Table 2**. HAMN generally performed better than other continual learning methods on CORE50 (**Fig. S2**). From **Table 2**, HAMN outperforms next best method (L2) by 4.8% and 1.7% on average in class iid and instance scenarios, respectively, on iLab (also see Supp. Fig S1c, S2c, and S3c). HAMN also surpasses the next best baseline (iCARL) with an improvement of 6% and 3.2% on class iid and instance, respectively, on the Toybox dataset (also see Supp. Fig S1b, S2b, and S3b).

Methods generally performed similarly on task 1 after one epoch, with the upper bound performing slightly better due to training on multiple epochs. As more tasks are added, task 1 performance decreased due to forgetting (Supp. Fig S5). For example, on task 5 of CoRE50, many baseline methods were barely above chance and were comparable to the lower bound. The best baseline methods were iCARL, L2, and REMIND. HAMN achieved the highest accuracy overall (see also Supp. Fig S3a). HAMN consistently outperformed the best previous method, iCARL, across all tasks by 7% on average. Thus, given comparable memory usage with iCARL, HAMN can re-use learnt hypotheses and transfer knowledge from previous tasks more effectively. Similarly, the feature maps reconstructed from learnt hypotheses used for replay carry discriminative information. REMIND heavily relies on a pre-computed and fixed codebook for constructing the memory based on first tasks. This hinders adaptability to new tasks. REMIND underperformed when there is limited feature diversity with only 2 classes in the first task in all 3 video datasets. Its performance remained inferior even in the online CIFAR100 image dataset. Finally, HAMN was three-fold more efficient in memory usage than parameter regularization methods. HAMN’s improvement in memory usage efficiency was even more pronounced for CIFAR100 (**Sec 5.2**).

Accuracy (%)	class_iid	instance
HAMN (SqueezeNet)	36.8 ± 2	29.6 ± 2
HAMN (MobileNet)	49.9 ± 2	39.1 ± 3
NumMemSlot	27.7 ± 1	27.6 ± 4
NumIndexReplays	33.3 ± 2	33.9 ± 6
MemSparseness	35.8 ± 3	33.0 ± 5
DistillationLoss	36.8 ± 4	25.1 ± 3
MemUsageLoss	33.2 ± 5	26.8 ± 5
NoReplay	16.0 ± 1	17.2 ± 1
Replay with Herding	32.0 ± 1	28.2 ± 4

Table 1. **Top-1 classification accuracy in the 5th task for the ablated HAMN models after 5 runs on CORE50 in the class iid and instance protocols.** See **Sec 6.2** for each ablated method. \pm value denotes root mean squared error (RMSE).

The class instance protocol is more challenging than class iid, as shown by the performance differences of each method between the two protocols (compare **Fig. S2a** and **Fig. S2b**). Compared with its own performance in the class iid setting, one reason HAMN is worse in the class instance setting is overfitting of the learnt hypotheses to particular tasks. One way to eliminate hypothesis overfitting is to decrease the hyperparameter controlling the attention strength β , encouraging the network to use many hypotheses drawn from different tasks simultaneously. In an ablation study described below, we show the importance of controlling the attention strength β in the two protocols.

Figure S4 provides visualizations of the predicted logit vectors after hypothesis replays during the class instance protocol. We randomly choose 50 logit vectors from each class and plot them using t-sne [47]. In this example, Task 1 involves separating two classes: cups and scissors. As expected, the representation of those two classes is quite distinct, which leads to a high classification accuracy. In Tasks 3 and 5, the plots show the representation of the added classes. Remarkably, the two initial classes remain distinctly separated despite no training on those classes after the first task. This shows that HAMN accommodates new classes while maintaining the clustering of previous ones. We also provide visualization of the learnt augmented memory (see Supp.Sec.S2), demonstrating that HAMN learns meaningful and consistent object representations over tasks.

Moreover, we tested all models in the standard benchmark online-CIFAR100 dataset with 20 tasks. **Table 2** shows that HAMN consistently outperforms other competitive models and leads the second best model by a huge margin of 11.5% (see also Supp. Fig S1d, S2d, and S3d).

6.2. Ablation Study

We assessed the importance of design choices by evaluating ablated versions of the model from the predictions based on Z_t on CORE50 (**Table S5**). First, the multi-head reading mechanism reduces memory usage by compressing

	HAMN (ours)	EWC [22]	GEM [31]	AGEM [8]	iCARL [38]	L2 [22]	MAS [1]	Naive Rehs	SI [52]	Stable [34]	GSS [2]	BIC [50]	CoPE [10]	LwF [26]	REM [16]	AAN [28]	RM [4]
Toybox dataset [48]																	
iid	56.1	39.4	39.2	40.6	50.1	44.1	41.7	39.2	39.2	34.1	35.5	32.0	42.9	44.9	39.5	37.3	33.3
inst.	53.7	39.4	38.8	37.1	50.5	43.6	42.0	38.9	38.9	35.3	35.1	32.5	41.9	45.3	39.5	31.1	31.6
iLab dataset [5]																	
iid	46.8	35.0	35.1	32.4	41.0	42.0	41.3	35.2	35.1	26.7	34.1	32.3	38.0	34.5	34.1	32.1	28.4
inst.	44.2	35.7	34.5	29.2	42.1	42.5	40.5	34.5	34.8	29.8	34.5	32.0	37.4	32.1	34.5	28.9	33.2
online CIFAR100 dataset [24]																	
iid	45.1	15.7	-	21.1	33.6	13.6	18.4	15.2	15.2	17.9	12.8	16.3	18.4	16.6	16.4	15.4	12.1

Table 2. **Averaged Top-1 classification accuracy over all tasks in class iid and instance stream learning protocols on Toybox, iLab datasets and online class incremental iid protocol on CIFAR100 dataset.** See Sec. 3 for experiments and datasets, Sec. 5.1 for baselines, and Supp. Fig S3 for individual plots over all datasets. The best and second best results are bolded.

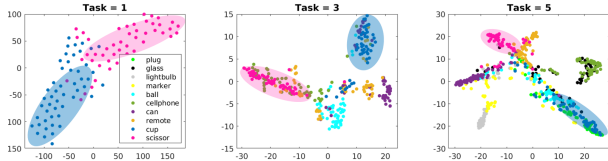


Figure 4. **Clusters of class embeddings learnt in task 1 remain separated in subsequent tasks after hypothesis replays.** We provide t-sne [47] plots of logit vectors from each class after hypothesis replays within a class instance training run on the CORE50 dataset. Task 1 is a binary classification problem. Tasks 3 and 5 are 6-choose-1 and 10-choose-1 classification problems respectively. Colors correspond with the object classes in the legend. We use the shaded blobs to emphasize the clusters of cups and scissors from the 1st task and their corresponding locations in subsequent tasks.

feature maps into a minimal number of hypotheses needed for reconstruction. Instead of using SqueezeNet as a backbone, we tested MobileNet v2 [40] with fewer channels in the feature extractor. Compared with SqueezeNet (with $S = 512$ channels in the feature maps), in MobileNet v2 layer 7, the feature maps are of size $96 \times 14 \times 14$. A hypothesis vector of dim. 8 would result in $12 \times 14 \times 14$ reading keys. Assuming a fixed memory size, the fewer reading keys than SqueezeNet would reduce the reconstruction error onto the original feature maps. As expected, we observed stronger continual learning performance for MobileNet v2.

Second, we tested the importance of the number of memory slots N by reducing it from 100 to 50. There was a drop in accuracy on the final task of almost 10% (class iid) and 2% (class instance). This implies that we need a sufficient number of hypotheses to discriminate between classes.

Third, we reduced the number of stored samples by half (pairs of top-1 attention memory indices and their corresponding logits) from $X = 200$ to $X = 100$. For class iid, accuracy dropped by $\approx 5\%$, indicating that we need to store multiple samples per class to represent class diversity. Reducing X is not as harmful as reducing N , suggesting that the hypotheses in the augmented memory are highly discriminative and that there exist regularities among representations within the same class that reduce the need to store as many replay samples as in pixel-based replay. In the class instance protocol, accuracy improved by 4% in the class instance protocol, suggesting that models more easily

overfit in the class instance setting.

Fourth, in MemSparseness, we decreased the reading attention strength β (Eq. 2) from 5 to 1. A higher β forces the model to attend to fewer memory slots. Setting β to 1 produces a more blurred distribution over all hypotheses, causing an accuracy drop of $\approx 1\%$ in the class iid protocol. Surprisingly, the lower sparsity helped the model in the class instance protocol. One conjecture is that HAMN more easily overfits in the class instance protocol, so a blurred hypothesis distribution could help alleviate the strong preference for a single hypothesis during predictions.

Fifth, distillation loss is important for knowledge transfer between networks [18] and across sequential tasks [38]. We introduced two distillation losses in HAMN, during replay and training. There was no significant effect of removing the distillation loss for the class iid protocol. However, accuracy dropped by 5% in the class instance protocol.

Sixth, in MemUsageLoss, we removed the least used memory loss L_{mem} , which was introduced to prevent useful old hypotheses from being over-written. Removing L_{mem} results in a 3% accuracy drop for both protocols, demonstrating that preventing interference between old and new hypotheses helps prevent catastrophic forgetting.

Seventh, we removed replay from the HAMN model (NoReplay). As expected, L_{mem} alone is not sufficient to avoid catastrophic forgetting. The large decrease in accuracy of 15-20% for both protocols emphasizes that replay is essential and demonstrates that the learnt hypotheses capture representative information for old classes.

Lastly, rather than storing random samples in replay buffers from previous tasks, we tested the herding sampling strategy [38]. We did not observe any performance increase. It is possible that herding only selects essential components in object representations in each class and these representations are already present with the same minimal number of memory indices in HAMN. In contrast, random sampling might increase the chance of introducing more diversity to object representations per object class.

7. Conclusions

We addressed the problem of online stream learning for classification tasks and proposed a novel method of memory index replay using hypotheses stored in augmented mem-

ory. In addition to ameliorating catastrophic forgetting on benchmark datasets, our method uses memory more efficiently than other methods, while generalizing to novel concepts even when trained only once on a continuous stream.

Despite the promising results of HAMN in online stream learning, several future directions should be explored. First, to test long-range online stream learning abilities, there is a lack of large video datasets for object recognition with hundreds of classes. Second, HAMN requires a feature extractor pre-trained on ImageNet, hindering its ability of generalization to other domains (e.g., X-ray scans). For continual learning in new domains, one could first train the feature extractor via unsupervised learning in the new domain, e.g., via generative adversarial networks [7], and then use it with HAMN for online stream learning tasks in the new domain.

References

- [1] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154, 2018. 6, 8
- [2] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *arXiv preprint arXiv:1903.08671*, 2019. 2, 6, 8
- [3] Craig Atkinson, Brendan McCane, Lech Szymanski, and Anthony Robins. Pseudo-recursal: Solving the catastrophic forgetting problem in deep neural networks. *arXiv preprint arXiv:1802.03875*, 2018. 2
- [4] Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8218–8227, 2021. 2, 6, 8
- [5] Ali Borji, Saeed Izadi, and Laurent 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*, pages 2221–2230, 2016. 2, 3, 8
- [6] Pratik Prabhanjan Brahma and Adrienne Othon. Subset replay based continual learning for scalable improvement of autonomous systems. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1179–11798. IEEE, 2018. 5
- [7] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018. 9
- [8] Arslan Chaudhry, Marc Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with agem. *arXiv preprint arXiv:1812.00420*, 2018. 2, 6, 8
- [9] Yutian Chen, Max Welling, and Alex Smola. Super-samples from kernel herding. *arXiv preprint arXiv:1203.3472*, 2012. 5
- [10] Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. *arXiv preprint arXiv:2009.00919*, 2020. 2, 6, 8
- [11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 3, 6
- [12] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999. 1
- [13] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014. 1
- [14] Tengda Han, Weidi Xie, and Andrew Zisserman. Memory-augmented dense predictive coding for video representation learning. *arXiv preprint arXiv:2008.01065*, 2020. 1
- [15] Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, 2017. 1
- [16] Tyler L Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting. In *European Conference on Computer Vision*, pages 466–483. Springer, 2020. 1, 3, 5, 6, 8
- [17] Xu He and Herbert Jaeger. Overcoming catastrophic interference using conceptor-aided backpropagation. 2018. 2
- [18] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 3, 8
- [19] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016. 5, 6
- [20] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010. 3
- [21] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler L Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *Thirty-second AAAI conference on artificial intelligence*, 2018. 1
- [22] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 2, 6, 8
- [23] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1885–1894. JMLR. org, 2017. 5
- [24] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 2, 3, 6, 8
- [25] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *Advances in neural information processing systems*, pages 4652–4662, 2017. 2
- [26] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017. 2, 6, 8

- [27] Xialei Liu, Chenshen Wu, Mikel Menta, Luis Herranz, Bogdan Raducanu, Andrew D Bagdanov, Shangling Jui, and Joost van de Weijer. Generative feature replay for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 226–227, 2020. [3](#)
- [28] Yaoyao Liu, Bernt Schiele, and Qianru Sun. Adaptive aggregation networks for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2544–2553, 2021. [2](#), [6](#), [8](#)
- [29] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 12245–12254, 2020. [2](#)
- [30] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *Conference on Robot Learning*, pages 17–26. PMLR, 2017. [2](#), [3](#)
- [31] David Lopez-Paz et al. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6467–6476, 2017. [2](#), [6](#), [8](#)
- [32] Davide Maltoni and Vincenzo Lomonaco. Continuous learning in single-incremental-task scenarios. *Neural Networks*, 2019. [1](#)
- [33] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989. [1](#)
- [34] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Razvan Pascanu, and Hassan Ghasemzadeh. Understanding the role of training regimes in continual learning. *arXiv preprint arXiv:2006.06958*, 2020. [2](#), [6](#), [8](#)
- [35] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017. [2](#)
- [36] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019. [5](#)
- [37] Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285, 1990. [1](#)
- [38] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. Icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017. [2](#), [5](#), [6](#), [8](#)
- [39] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995. [2](#)
- [40] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. [8](#)
- [41] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. One-shot learning with memory-augmented neural networks. *arXiv preprint arXiv:1605.06065*, 2016. [1](#)
- [42] Gehui Shen, Song Zhang, Xiang Chen, and Zhi-Hong Deng. Generative feature replay with orthogonal weight modification for continual learning. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021. [3](#)
- [43] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999, 2017. [2](#)
- [44] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017. [2](#)
- [45] Sebastian Thrun and Tom M Mitchell. Lifelong robot learning. *Robotics and autonomous systems*, 15(1-2):25–46, 1995. [1](#)
- [46] Guido M van de Ven, Zhe Li, and Andreas S Tolias. Class-incremental learning with generative classifiers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3611–3620, 2021. [3](#)
- [47] Laurens Van Der Maaten. Accelerating t-sne using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014. [7](#), [8](#)
- [48] Xiaohan Wang, Tengyu Ma, James Ainooson, Seunghwan Cha, Xiaotian Wang, Azhar Molla, and Maithilee Kunda. The toybox dataset of egocentric visual object transformations. *arXiv preprint arXiv:1806.06034*, 2018. [2](#), [3](#), [8](#)
- [49] Junfeng Wen, Yanshuai Cao, and Ruitong Huang. Few-shot self reminder to overcome catastrophic forgetting. *arXiv preprint arXiv:1812.00543*, 2018. [2](#), [3](#), [5](#)
- [50] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 374–382, 2019. [2](#), [6](#), [8](#)
- [51] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *arXiv preprint arXiv:1411.1792*, 2014. [3](#)
- [52] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3987–3995. JMLR. org, 2017. [2](#), [6](#), [8](#)
- [53] Mengmi Zhang, Tao Wang, Joo Hwee Lim, Gabriel Kreiman, and Jiashi Feng. Variational prototype replays for continual learning. *arXiv preprint arXiv:1905.09447*, 2019. [2](#)

S8. Appendix

List of supplementary figures

1. Top-1 classification accuracy in each task in the class iid stream learning setting on the CORE50, Toybox, iLab and CIFAR100 datasets, **Fig S1**
2. Top-1 classification accuracy in each task in the class instance stream learning setting on the CORE50, Toybox and iLab datasets, **Fig S2**
3. Average top-1 classification accuracy over all tasks in the class iid and class instance stream learning settings on the CORE50, Toybox, iLab and CIFAR100 datasets, **Fig S3**
4. Embedding clusters of classes learned by upper bound over 5 tasks (a) and confusion matrix between actual class labels and predicted class labels after the 5th task in class instance setting on CORE50 dataset, **Fig S4**
5. Average top-1 classification accuracy on Task 1 over all tasks in class iid and class instance stream learning settings on CORE50, Toybox, iLab and CIFAR100 datasets, **Fig S5**
6. Each hypothesis represents a concept and each learnt concept remains approximately the same across tasks, **Fig S6**

S9. Preventing catastrophic forgetting

A proficient stream learning method should not only show good memory retention and avoid catastrophic forgetting, but should also be able to adapt to new tasks. In the main text, we described the average classification accuracy over *all* previous tasks. To evaluate the ability to prevent catastrophic forgetting, here we report the classification accuracy only on the first task (first two classes seen by the model) after each continual learning algorithm learns a new task. If the algorithm completely forgets what has been learnt in the first task and overfits to the current task (i.e. current 2 new classes in CORE50, iLab, and Toybox, or current 5 new classes in CIFAR100), the classification accuracy would be 0% for the older classes. For example, on CORE50, an algorithm with catastrophic forgetting would achieve high accuracy (e.g. 95%) in the current task, and 0% accuracy on each previous task due to overfitting.

Conversely, if an algorithm learns the first task perfectly (100%) but completely fails to learn additional new tasks, the algorithm would achieve 100% accuracy on all images in Task 1 after training on all subsequent tasks; but 0% of accuracy on all images from other tasks.

Figure S5a shows the average classification accuracy on the first two classes in Task 1 throughout all the tasks of

CORE50. Most of the algorithms have an average classification accuracy of around 20% on Task 1 while our method achieves 76.9% in class iid and 75.1% in class instance settings. Although REMIND achieves the highest accuracy on Task 1, the average classification accuracy over all tasks is lower than that of our method across all four datasets. This indicates that REMIND does not adapt to new tasks as well as our method, possibly due to its reliance on a fixed, pre-computed codebook for encoding and decoding memories.

Overall, after comparison with all baselines in terms of both classification accuracy on all images from all tasks or only from the first task, our HAMN model not only generalizes to learn new tasks, but also demonstrates excellent memory retention capabilities. Similar conclusions can be drawn from the Toybox, iLab, and CIFAR100 datasets (**Fig S3b, S3c, S3d**). Our method achieves the highest average classification accuracy over all tasks on all four datasets in both class iid and class instance settings.

S10. Memory Interpretation

To interpret what the learnt hypotheses in the augmented memory represent, we provide a visualization of 2D probabilistic hypothesis activation maps for three hypotheses on an image from each class within a class instance training run on CORE50 (**Fig S6**). The brighter regions denote locations where the hypothesis of interest is more likely to get activated. We find consistent activation patterns over all classes. For example, some hypotheses focus on object parts while others focus on the background and hand features. We also provide a side-by-side comparison of the hypotheses from the same memory index between tasks 1 and 5. We do not observe a change of hypothesis patterns with new tasks, implying that the learnt hypotheses are not overwritten by new ones thanks to the least used memory loss L_{mem} . This is useful to avoid catastrophic forgetting and to assist transfer learning on new tasks. We also verified the importance of L_{mem} in the ablation study.

S11. Ablation Study using \tilde{Z}_t

In the main paper, we assessed the importance of design choices by evaluating ablated versions of our model from the predictions based on Z_t on CORE50 (**Table 1**). In contrast, here we provide the ablation results based on the predictions from \tilde{Z}_t .

Accuracy (%)	class_iid	instance
HAMN (SqueezeNet)	45.0 \pm 5	39.7 \pm 6
HAMN (MobileNet)	59.6 \pm 1	48.8 \pm 3
NumMemSlot	46.7 \pm 3	34.2 \pm 2
NumIndexReplays	41.0 \pm 4	34.5 \pm 5
MemSparseness	47.2 \pm 4	39.2 \pm 3
DistillationLoss	50.6 \pm 4	42.8 \pm 5
MemUsageLoss	28.1 \pm 3	30.2 \pm 4
NoReplay	15.0 \pm 1	15.4 \pm 1
Replay with Herding	31.3 \pm 3	33.7 \pm 3

Table S3. **Top-1 classification accuracy in the 5th task for the ablated HAMN models after 10 runs on CORE50 in the class iid and instance protocols.** Only runs with accuracy above 80% on the first task were used in the calculations. NumMemSlot class instance required 20 initial runs to yield more than one run with task 1 accuracy above 80%. See **Sec 6.2** for each ablated method. \pm values denote root mean squared error (RMSE).

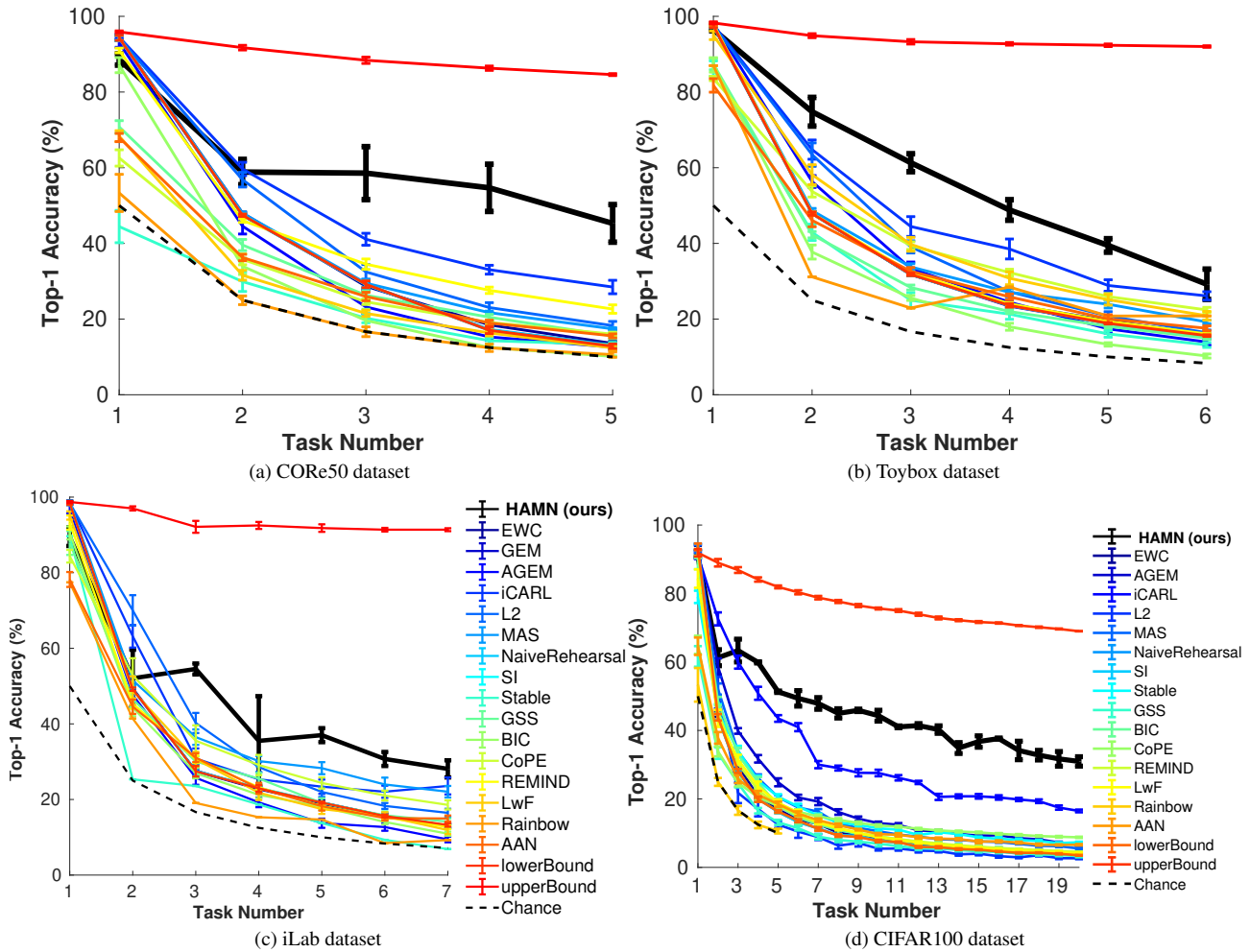


Figure S1. **Top-1 classification accuracy over all tasks in class iid stream learning settings on CORE50 (a), Toybox (b), iLab (c), and CIFAR100 (d) datasets.** Each color shows a different model (see main text for abbreviations and model definitions). The dashed line denotes chance level. See main paper for experimental protocols and datasets as well as baselines. The error bars denote root mean standard deviation (RMSD) over total 10 runs. Part a is the same as Fig.3a in the main text and is reproduced here for comparison purposes.

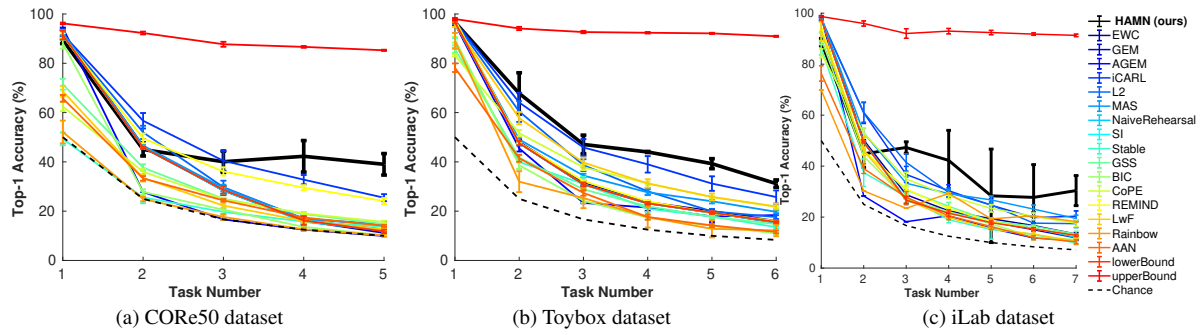


Figure S2. **Top-1 classification accuracy over all tasks in class instance stream learning settings on CORE50 (a), Toybox (b) and iLab (c) datasets.** The figure format and conventions follow **Figure S1**. Part a is the same as Fig.3b in the main text and is reproduced here for comparison purposes.

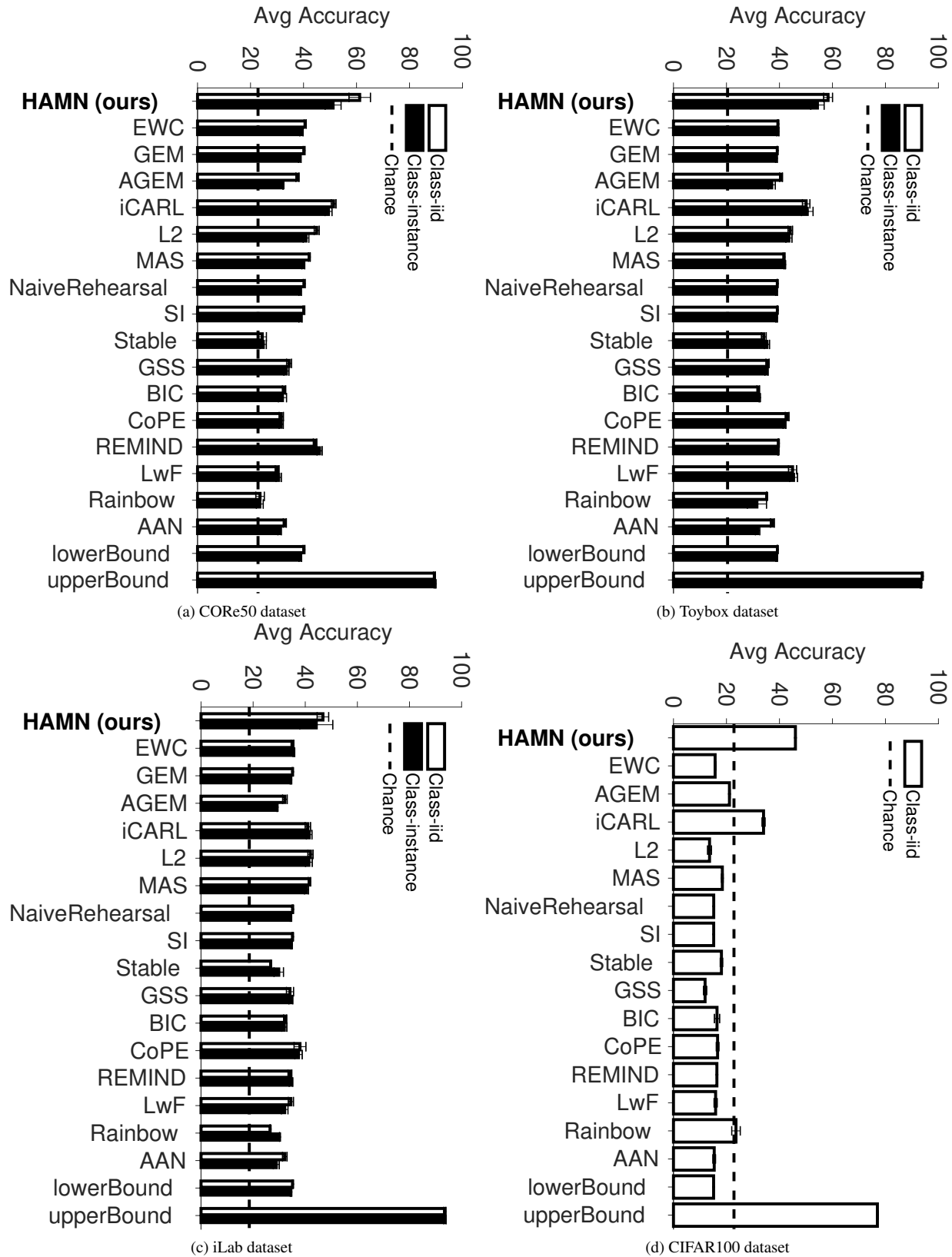


Figure S3. Average top-1 classification accuracy over all tasks in class iid (white) and instance stream learning (black) settings on CORe50 (a), Toybox (b), iLab (c), and CIFAR100 (d) datasets. Dashed line is chance.

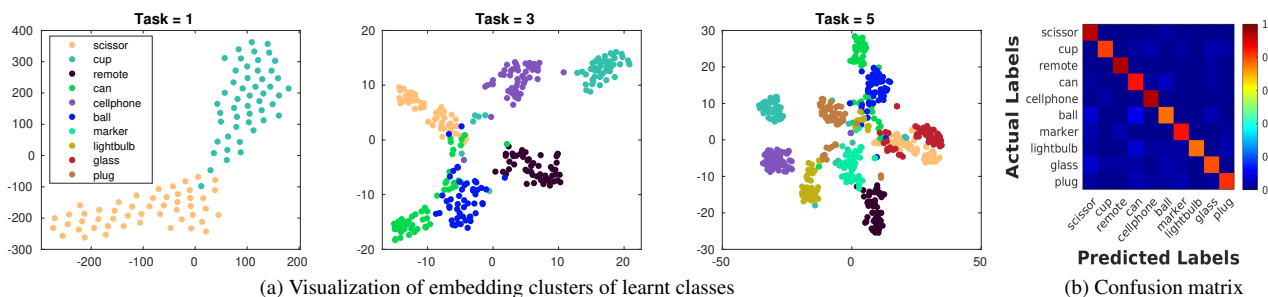


Figure S4. **Embedding clusters of classes learned by upper bound over 5 tasks (a) and confusion matrix between actual class labels and predicted class labels after the 5th task in class instance setting on CORE50 dataset.** (a). t-sne plots of logit vectors from each class. The first task (Task = 1) is a binary classification problem. The 3rd task is a 1-choose-6 classification problem. The 5th task is a 1-choose-10 classification problem. Colors correspond with the object classes in the legend. Note that the representation of the first two classes (scissors and cups) remains distinct throughout the different tasks, showing that the model does not completely forget the initial classes learnt during Task 1. (b). Confusion matrix after the 5th task. The sequence of actual labels (top to bottom) follows the class presentation order across tasks, i.e. scissor and cup in 1st task, remote and can in the second task and so on. See the scale bar on the right for probability values. See Fig 4 in the main paper to compare with our HAMN method.

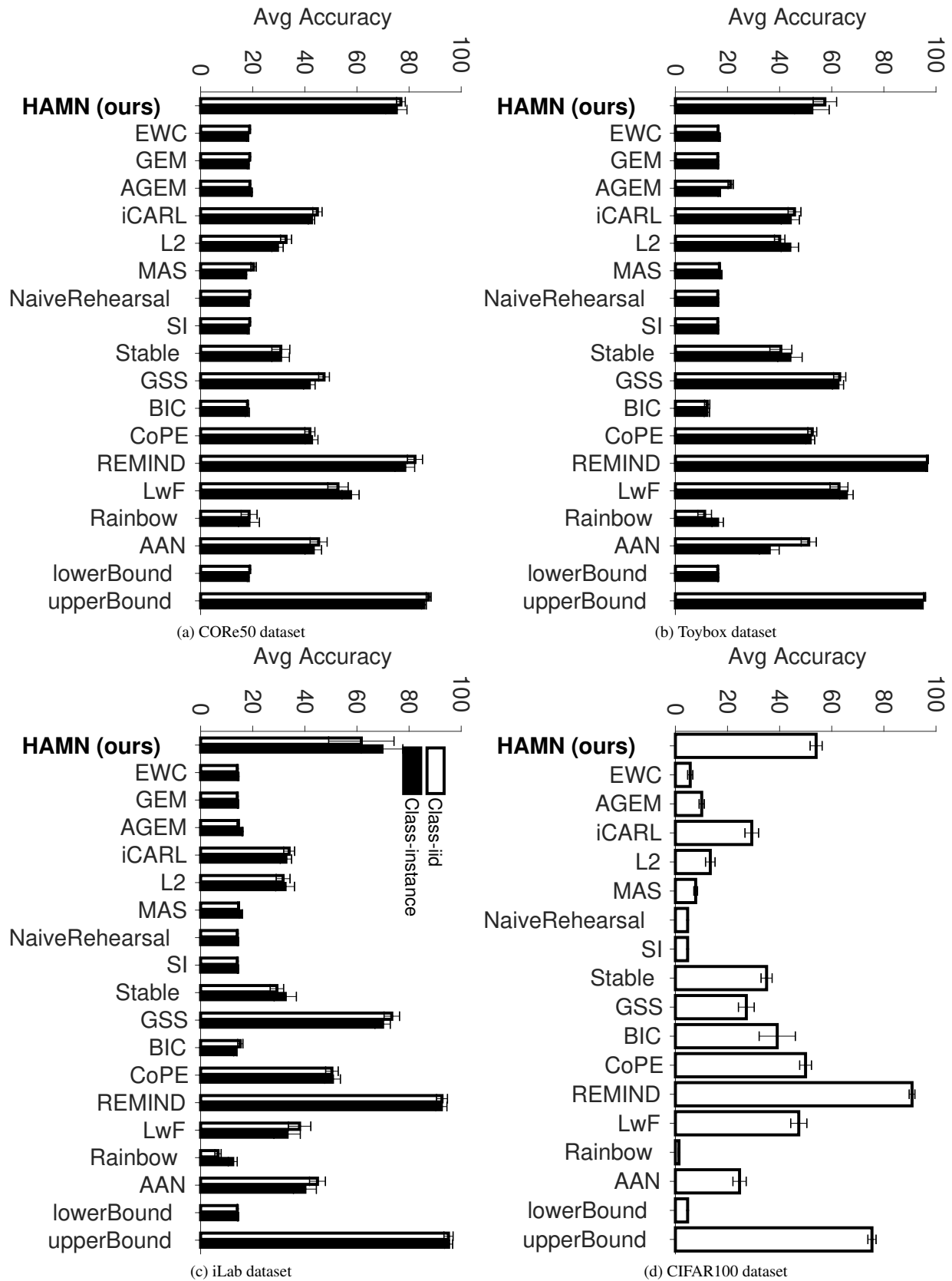


Figure S5. Average top-1 classification accuracy on Task 1 over all tasks in class iid (white) and instance (black) stream learning settings on CORe50 (a), Toybox (b), iLab (c), and CIFAR100 (d) datasets.

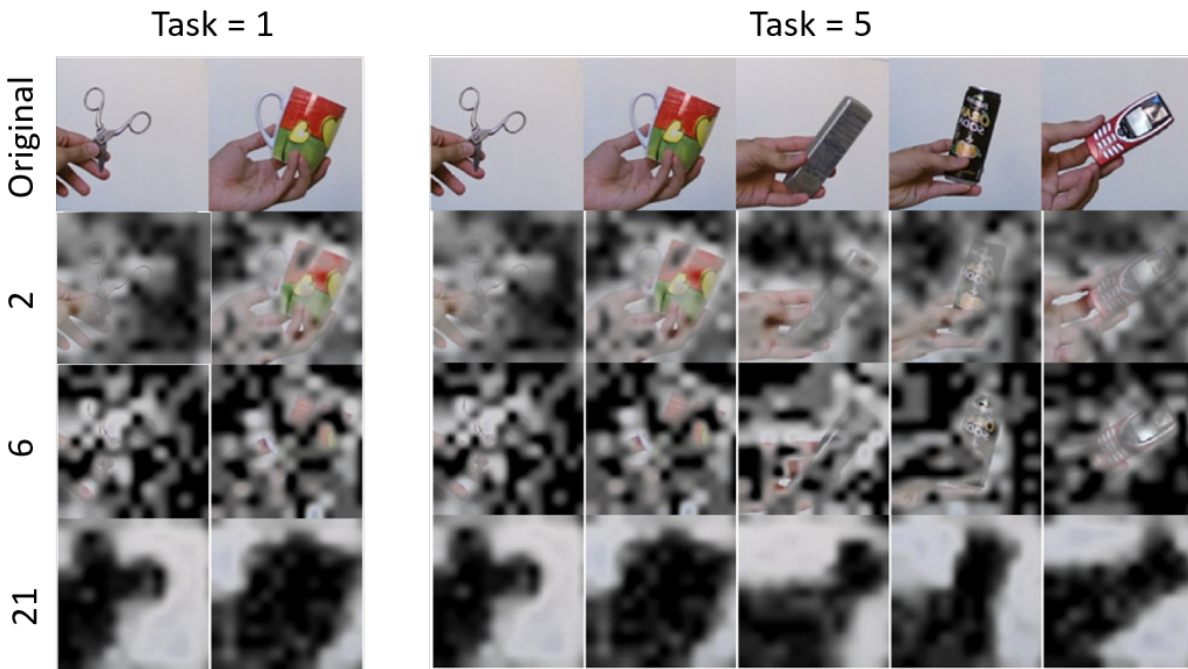


Figure S6. **Each hypothesis represents a concept and each learnt concept remains the same across tasks.** To visualize what activates each hypothesis in the augmented memory, given any image I_t , we used its top-1 attention index tensor w_t^z of size $D \times W \times H$, selected the hypothesis index of interest, and computed its activation probability over all locations, resulting in a 2D probabilistic hypothesis activation map. We overlaid this map onto the original image, with lower activation probabilities corresponding to higher opacity (darker regions). The brighter regions denote locations where the selected hypotheses get activated with higher probability.