

SUPPLEMENTARY MATERIAL

A STAM NOTATION AND HYPERPARAMETERS

All STAM notation and parameters are listed in Tables 1-5.

Table 1: STAM Notation

Symbol	Description
\mathbf{x}	input vector.
n	dimensionality of input data
M_l	number of patches at layer l (index: $m = 1 \dots M_l$)
$\mathbf{x}_{l,m}$	m 'th input patch at layer l
C_l	set of centroids at layer l
$\mathbf{c}_{l,j}$	centroid j at layer l
$d(\mathbf{x}, c)$	distance between an input vector \mathbf{x} and a centroid c
$\hat{c}(\mathbf{x})$	index of nearest centroid for input x
\tilde{d}_l	novelty detection distance threshold at layer l
$U(t)$	the set of classes seen in the unlabeled data stream up to time t
$L(t)$	the set of classes seen in the labeled data up to time t
k	index for representing a class
$g_{l,j}(k)$	association between centroid j at layer l and class k .
\bar{D}_l	average distance between a patch and its nearest neighbor centroid at layer l .
$v_{l,m}(k)$	vote of patch m at layer l for class k
$v_l(k)$	vote of layer l for class k
$k(\mathbf{x})$	true class label of input x
$\hat{k}(\mathbf{x})$	inferred class label of input x
$\Phi(\mathbf{x})$	embedding vector of input x

Table 2: STAM Hyperparameters

Symbol	Default	Description
Λ	3	number of layers (index: $l = 1 \dots \Lambda$)
α	0.1	centroid learning rate
β	0.95	percentile for novelty detection distance threshold
γ	0.15	used in definition of class informative centroids
Δ	see below	STM capacity
θ	30	number of updates for memory consolidation
ρ_l	see below	patch dimension

Table 3: MNIST/EMNIST Architecture

Layer	ρ_l	Δ (incremental)	Δ (uniform)
1	8	400	2000
2	13	400	2000
3	20	400	2000

Table 4: SVHN Architecture

Layer	ρ_l	Δ (incremental)	Δ (uniform)
1	10	2000	10000
2	14	2000	10000
3	18	2000	10000

Table 5: CIFAR Architecture

Layer	ρ_l	Δ (incremental)	Δ (uniform)
1	12	2500	12500
2	18	2500	12500
3	22	2500	12500

B BASELINE MODELS

The first baseline is based on the Gradient Episodic Memories (GEM) model (Lopez-Paz & Ranzato, 2017) for continual learning. We adapt GEM in the UPL context using the rotation-prediction self-supervised loss (Gidaris et al., 2018). We also adopt the Network-In-Network architecture of (Gidaris et al., 2018). The model is trained with the Adam optimizer with a learning rate of 10^{-4} , batch size of 4 (the four rotations from each example image), and only one epoch (to be consistent with the streaming requirement of UPL). GEM requires knowledge of task boundaries: at the end of each phase (time period with stationary data distribution), the model stores the M_n most recent examples from the training data – see (Lopez-Paz & Ranzato, 2017) for more details. We set the size M_n of the “episodic memories buffer” to the same size with STAM’s STM, as described in SM-C.

The second baseline is based on the Memory Aware Synapse (MAS) model (Aljundi et al., 2018) for continual learning. As in the case of GEM, we adapt MAS in the UPL context using a rotation-prediction self-supervised loss (Gidaris et al., 2018), and the Network-In-Network architecture. At the end of each Phase, MAS calculates the importance of each parameter on the last task. These values are used in a regularization term for future tasks so that important parameters are not forgotten. Importantly, this calculation requires additional data. To make sure that MAS utilizes the same data with STAM and GEM, we train MAS on the first 90% of the examples during each Phase, and then calculate the importance values on the last 10% of the data.

C MEMORY CALCULATIONS

The memory requirement of the STAM model can be calculated as:

$$M = \sum_{l=1}^{\Lambda} \rho_l^2 \cdot \Delta + \sum_{l=1}^{\Lambda} \rho_l^2 \cdot |C_l| \quad (8)$$

where the first sum term is equivalent to the STM size and the second sum term is the LTM size.

We compare the LTM size of STAM with the learnable parameters of the deep learning baselines. STAM’s STM, on the other hand, is similar GEM’s a temporary buffer, and so we set the episodic memory storage of GEM to have the same size with STM.

Learnable Parameters and LTM: For the 3-layer SVHN architecture with $|C_l| \approx 3000$ LTM centroids, the LTM memory size is ≈ 1860000 pixels. This is equivalent to ≈ 1800 gray-scale SVHN images. In contrast, the Network-In-Network architecture has 1401540 trainable parameters, which would also be stored at floating-point precision. Again, with four bytes per weight, the STAM model would require $\frac{1860000}{1401540 \times 4} \approx 33\%$ of both GEM’s and MAS’s memory footprint in terms of learnable parameters. Future work can decrease the STAM memory requirement further by merging similar LTM centroids. Figure 9(f) shows that the accuracy remains almost the same when $\Delta = 500$ and $|C_l| \approx 1000$. Using these values we get an LTM memory size of 620000, resulting in $\frac{620000}{1401540 \times 4} \approx 11\%$ of GEM’s and MAS’s memory footprint.

Temporary Storage and STM: We provide GEM with the same amount of memory as STAM’s STM. We set $\Delta = 400$ for MNIST, that is equivalent to $8^2 * 400 + 13^2 * 400 + 18^2 * 400 = 222800$ floating point values. Since the memory in GEM does not store patches but entire images, we need to convert this number into images. The size of an MNIST image is $28^2 = 784$, so the memory for GEM on MNIST contains $222800/784 \approx 285$ images. We divide this number over the total number of Phases – 5 in the case of MNIST – resulting in $M_t = 285/5 = 57$ images per task. Similarly for SVHN and CIFAR the Δ values are 2000 and 2500 respectively, resulting in $M_t \approx 1210/5 = 242$, $1515/5 = 303$, and $285/23 \approx 13$ images for SVHN, CIFAR-10, and EMNIST respectively.

D GENERALIZATION ABILITY OF LTM CENTROIDS

To analyze the quality of the LTM centroids learned by STAM, we assess the discriminative and generalization capability of these features. For centroid c and for class k , the term $g_c(k)$ (defined in Equation 4) is the association between centroid c and class- k , a number between 0 and 1. The closer that metric is to 1, the better that centroid is in terms of its ability to generalize across examples of class- k and to discriminate examples of that class from other classes.

For each STAM centroid, we calculate the maximum value of $g_c(k)$ across all classes. This gives us a distribution of “max-g” values for the STAM centroids. We compare that distribution with a null model in which we have the same number of LTM centroids, but those centroids are randomly chosen patches from the training dataset. These results are shown Figure 7. We also compare the two distributions (STAM versus “random examples”) using the Kolmogorov-Smirnov test. We observe that the distributions are significantly different and the STAM centroids have higher max-g values than the random examples. While there is still room for improvement (particularly with CIFAR-10), these results confirm that STAM learns better features than a model that simply remembers some examples from each class.

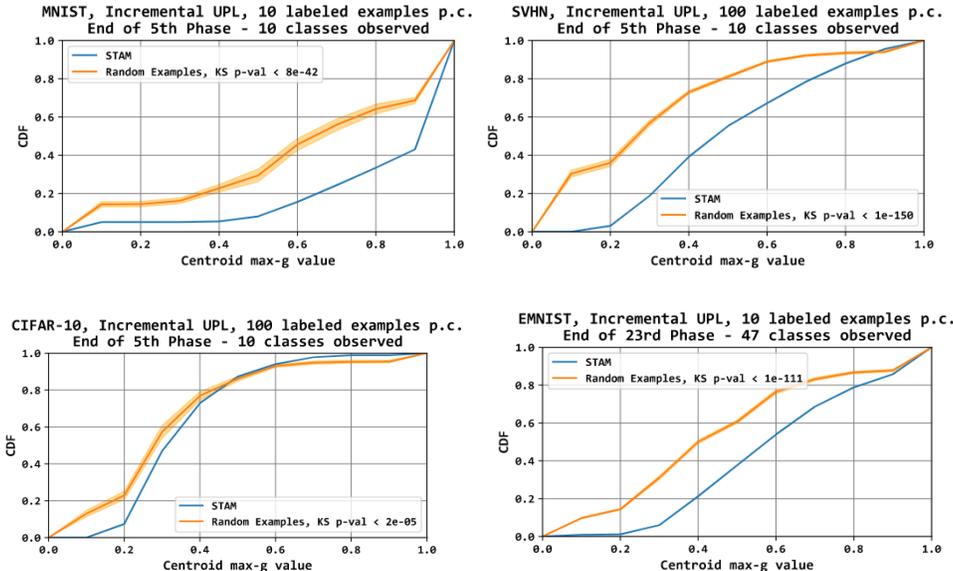


Figure 7: Comparison between the distribution of max-g values with STAM and random patches extracted from the training data.

E EFFECT OF UNLABELED AND LABELED DATA ON STAM

We next examine the effects of unlabeled and labeled data on the STAM architecture (Figure 8). As we vary the length of the unlabeled data stream (left), we see that STAMs can actually perform well even with much less unlabeled data. This suggests that the STAM architecture may be applicable even where the datastream is much shorter than in the experiments of this paper. A longer stream would be needed however if there are many classes and some of them are infrequent. The accuracy “saturation” observed by increasing the unlabeled data from 20000 to 60000 can be explained based on the memory mechanism, which does not update centroids after they move to LTM. As showed in the ablation studies, this is necessary to avoid forgetting classes that no longer appear in the stream. The effect of varying the number of labeled examples per class (right) is much more pronounced. We see that the STAM architecture can perform well above chance even in the extreme case of only a single (or small handful of) labeled examples per class.

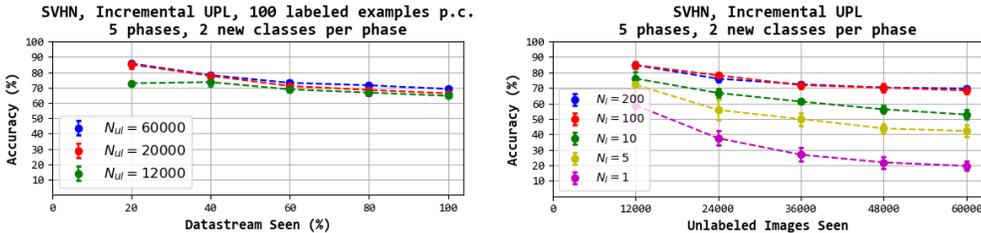
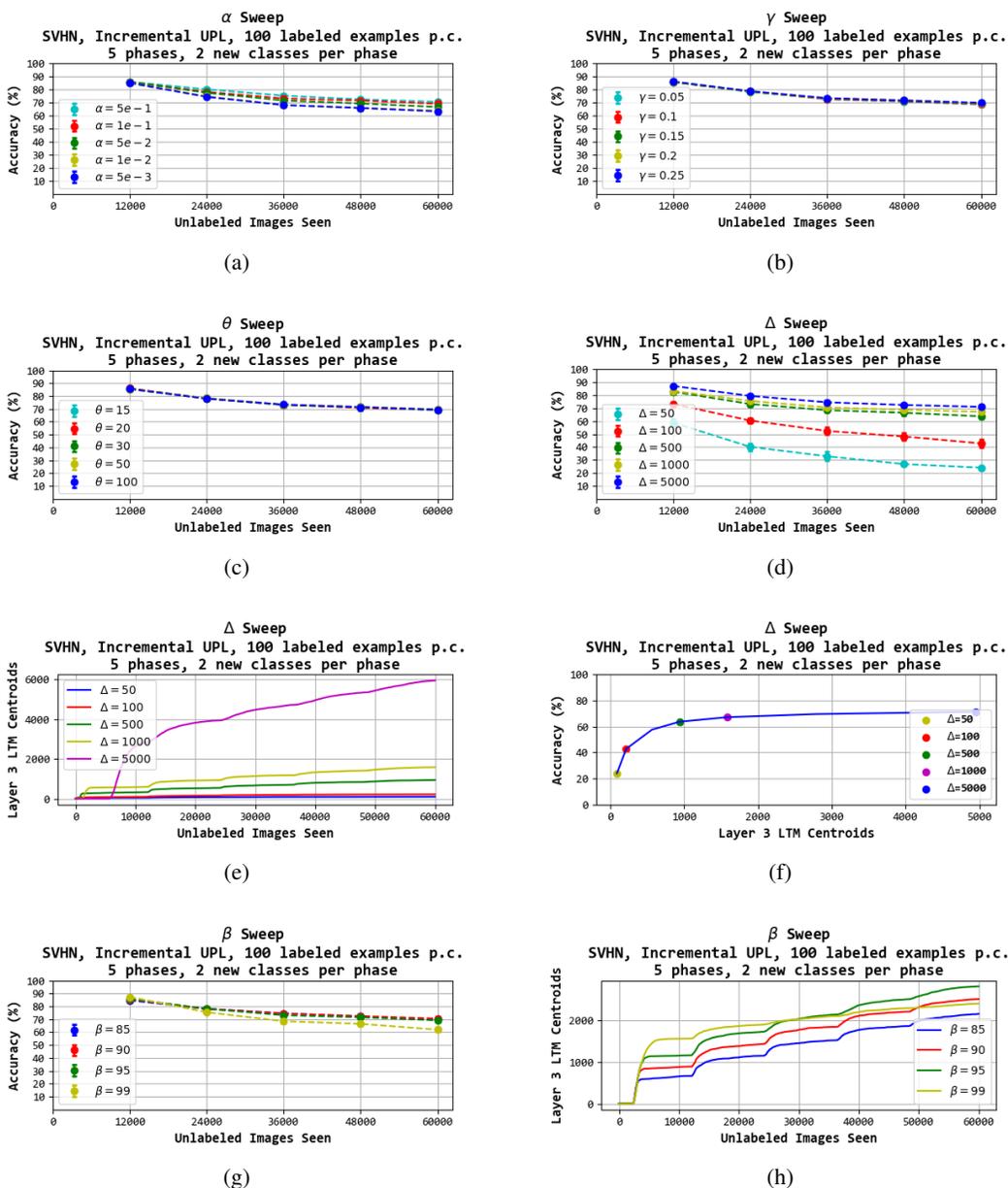


Figure 8: The effect of varying the amount of unlabeled data in the entire stream (left) and labeled data per class (right).

F STAM HYPERPARAMETER SWEEPS

We examine the effects of STAM hyperparameters in Figure 9. (a) As we decrease the rate of α , we see a degradation in performance. This is likely due to the static nature of the LTM centroids - with low α values, the LTM centroids will primarily represent the patch they were initialized as. (b) As we vary the rates of γ , there is little difference in our final classification rates. This suggests that the maximum $g_{i,j}(k)$ values are quite high, which may not be the case in other datasets besides SVHN. (c) We observe that STAM is robust to changes in θ . (d,e) The STM size Δ has a major effect on the number of learned LTM centroids and on classification accuracy. (e) The accuracy shows diminishing returns after we have about 1000 LTM centroids at layer-3. (g,h) As β increases the number of LTM centroids increases (due to a lower rate of novelty detection); if $\beta \geq 0.9$ the classification accuracy is about the same.

Figure 9: Hyperparameter sweeps for α , γ , θ , β , and Δ .

G UNIFORM UPL

In order to examine if the STAM architecture can learn all classes simultaneously, but without knowing how many classes exist, we also evaluate the STAM architecture in a uniform UPL scenario (Figure 10). Note that LTM centroids converge to a constant value, at least at the top layer. Each class is recognized at a different level of accuracy, depending on the similarity between that class and others.

H IMAGE PREPROCESSING

Given that each STAM operates on individual image patches, we perform *patch normalization* rather than image normalization. We chose a normalization operation that helps to identify similar patterns despite variations in the brightness and contrast: every patch is transformed to zero-mean, unit variance before clustering. At least for the datasets we consider in this paper, grayscale images result in higher classification accuracy than color.

We have also experimented with ZCA whitening and Sobel filtering. ZCA whitening did not work well because it requires estimating a transformation from an entire image dataset (and so it is not compatible with the online nature of the UPL problem). Sobel filtering did not work well because STAM clustering works better with filled shapes rather than the fine edges produced by Sobel filters.

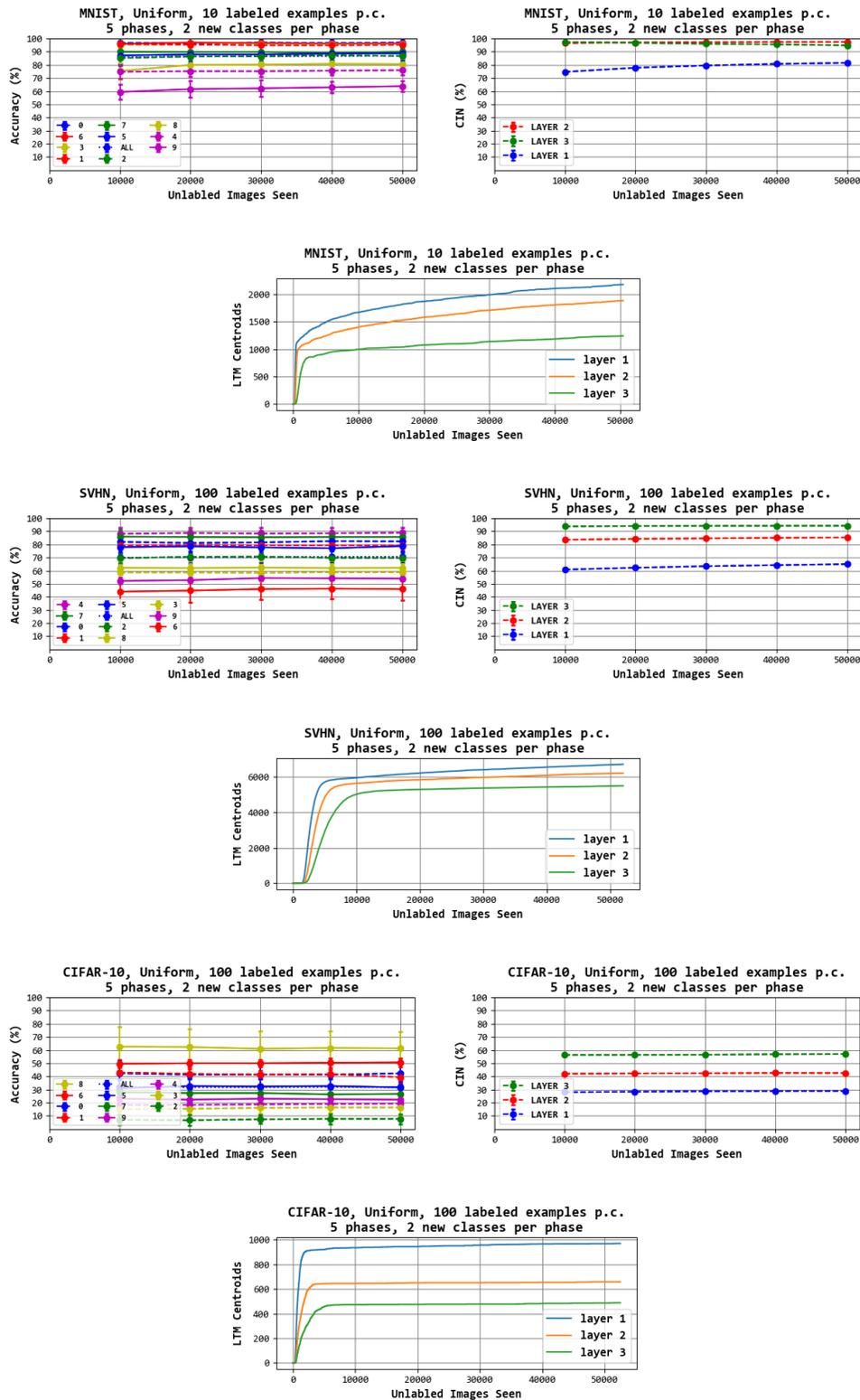


Figure 10: Uniform UPL evaluation for MNIST (row-1) and SVHN (row-2). Per-class/average classification accuracy is given at the left; the number of LTM centroids over time is given at the center; the fraction of CIN centroids over time is given at the right.