Unsupervised Progressive Learning and the STAM Architecture

James Smith [1] **Seth Baer*** [1] **Cameron Taylor*** [1] **Constantine Dovrolis** [1]

Abstract

We first pose the Unsupervised Progressive Learning (UPL) problem: an online representation learning problem in which the learner observes a non-stationary and unlabeled data stream, and identifies a growing number of features that persist over time even though the data is not stored or replayed. To solve the UPL problem we propose the Self-Taught Associative Memory (STAM) architecture. Layered hierarchies of STAM modules learn based on a combination of online clustering, novelty detection, forgetting outliers, and storing only prototypical features rather than specific examples. We evaluate STAM representations using classification and clustering tasks. Even though there are no prior approaches that are directly applicable to the UPL problem, we evaluate the STAM architecture in comparison to some unsupervised and self-supervised deep learning approaches adapted in the UPL context.

1. Introduction

The *Continual Learning (CL)* problem is predominantly addressed in the supervised context with the goal being to learn a sequence of tasks without "catastrophic forgetting" (Goodfellow et al., 2013; Parisi et al., 2019; van de Ven & Tolias, 2019). There are several CL variations but a common formulation is that the learner observes a set of examples $\{(x_i, t_i, y_i)\}$, where x_i is a feature vector, t_i is a task identifier, and y_i is the target vector associated with (x_i, t_i) (Chaudhry et al., 2019a;b; Lopez-Paz & Ranzato, 2017). Other CL variations replace task identifiers with task boundaries that are either given (Hsu et al., 2018) or inferred (Zeno et al., 2018). Typically, CL requires that the learner either stores and replays some previously seen examples (Aljundi et al., 2019a;b; Gepperth & Karaoguz, 2017; Hayes et al., 2019; Kemker et al., 2018; Rebuffi et al., 2017) or generates examples of earlier learned tasks (Kemker & Kanan, 2018; Liu et al., 2020; Shin et al., 2017).

The *Feature (or Representation) Learning (FL)* problem, on the other hand, is unsupervised but mostly studied in the *offline context*: given a set of examples $\{x_i\}$, the goal is to learn a *feature vector* (of a given, fixed dimensionality) $h_i = f(x_i)$ that, ideally, makes it easier to identify the explanatory factors of variation behind the data (Bengio et al., 2013), leading to better performance in tasks such as classification or clustering. FL methods differ in the prior $P(h)$ and the loss function. Autoencoders, for instance, aim to learn features of a lower dimensionality than the input that enable a sufficiently good reconstruction at the output (Bengio, 2014; Kingma & Welling, 2013; Tschannen et al., 2018; Zhou et al., 2012). A similar approach is self-supervised methods, which learn representations by optimizing an auxiliary task (Berthelot et al., 2019; Doersch et al., 2015; Gidaris et al., 2018; Kuo et al., 2019; Oord et al., 2018; Sohn et al., 2020).

In this work, we focus on a new and pragmatic problem that adopts some elements of CL and FL but is also different than them – we refer to this problem as *Unsupervised Progressive Learning (UPL)*. UPL can be described as follows:

1. the data is observed as a non-IID stream (e.g., different portions of the stream may follow different distributions and there may be strong temporal correlations between successive examples),
2. the features should be learned exclusively from unlabeled data,
3. each example is "seen" only once and the unlabeled data are not stored for iterative processing,
4. the number of learned features may need to increase over time, in response to new tasks and/or changes in the data distribution,
5. to avoid catastrophic forgetting, previously learned features need to persist over time, even when the corresponding data are no longer observed in the stream.

The UPL problem is encountered in important AI applications, such as a robot learning new visual features as it

*Equal contribution [1]College of Computing, Georgia Institute of Technology, Atlanta, GA. Correspondence to: James Smith <jamessealesmith@gatech.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

explores a time-varying environment. Additionally, we argue that UPL is closer to how animals learn, at least in the case of *perceptual learning* (Goldstone, 1998). We believe that in order to mimic that, ML methods should be able to learn in a streaming manner and in the absence of supervision. Animals do not "save off" labeled examples to train in parallel with unlabeled data, they do not know how many "classes" exist in their environment, and they do not have to replay/dream periodically all their past experiences to avoid forgetting them.

To the extent of our knowledge, the UPL problem has not been addressed before. The closest prior work is CURL ("Continual Unsupervised Representation Learning") by Rao et al. (Rao et al., 2019). CURL however does not impose the requirement that the data is presented to the learner as a stream that should be processed online, and so CURL requires iterative processing through gradient minimization methods (additional differences with CURL are discussed in Section 6).

To address the UPL problem, we describe an architecture referred to as STAM ("Self-Taught Associative Memory"). STAM learns features through *online clustering* at a hierarchy of increasing receptive field sizes. Online clustering can be performed through a single pass over the data stream. Further, despite its simplicity, clustering can generate representations that enable better classification performance than more complex FL methods such as sparse-coding or some deep learning methods (Coates et al., 2011; Coates & Ng, 2012). STAM allows the number of clusters to increase over time, driven by a *novelty detection* mechanism. Additionally, STAM includes a brain-inspired *dual-memory hierarchy* (short-term versus long-term) that enables the conservation of previously learned features (to avoid catastrophic forgetting) that have been seen multiple times at the data stream, while forgetting outliers.

2. STAM Architecture

In the following, we describe the STAM architecture as a sequence of its major components: a hierarchy of increasing receptive fields, online clustering (centroid learning), novelty detection, and a dual-memory hierarchy that stores prototypical features rather than specific examples. The notation is summarized in SM-A in the extended verison of the paper found under the same title on arXiv.

I. Hierarchy of increasing receptive fields: An input vector $\mathbf{x_t} \in \mathbb{R}^{\mathbf{n}}$ (an image in all subsequent examples) is analyzed through a hierarchy of Λ layers. Instead of neurons or hidden-layer units, each layer consists of STAM units – in its simplest form a STAM unit functions as an online clustering module. Each STAM processes one $\rho_l \times \rho_l$ *patch* (subvector) of the input at that layer. The patches are overlapping, with a small stride (set to one pixel in our experiments) to accomplish translation invariance (similar to CNNs). The patch dimension ρ_l increases in higher layers – the idea is that the first layer learns the smallest and most elementary features while the top layer learns the largest and most complex features.

II. Centroid Learning: Every patch of each layer is clustered, in an online manner, to a set of centroids. These time-varying centroids form the *features* that the STAM architecture gradually learns at that layer. All STAM units of layer l share the same set of centroids $C_l(t)$ – again for translation invariance.[1] Given the m'th input patch $\mathbf{x_{l,m}}$ at layer l, the nearest centroid of C_l selected for $\mathbf{x_{l,m}}$ is

$$\mathbf{c_{l,j}} = \arg\min_{c \in C_l} d(\mathbf{x_{l,m}}, \mathbf{c}) \tag{1}$$

where $d(\mathbf{x_{l,m}}, \mathbf{c})$ is the Euclidean distance between the patch $\mathbf{x_{l,m}}$ and centroid $\mathbf{c}$.[2] The selected centroid is updated based on a learning rate parameter α, as follows:

$$\mathbf{c_{l,j}} = \alpha\, \mathbf{x_{l,m}} + (1-\alpha)\mathbf{c_{l,j}}, \quad 0 < \alpha < 1 \tag{2}$$

A higher α value makes the learning process faster but less predictable. We do not use a decreasing value of α because the goal is to keep learning in a non-stationary environment rather than convergence to a stable centroid.

III. Novelty detection: When an input patch $\mathbf{x_{l,m}}$ at layer l is significantly different than all centroids at that layer (i.e., its distance to the nearest centroid is a statistical outlier), a new centroid is created in C_l based on $\mathbf{x_{l,m}}$. We refer to this event as *Novelty Detection (ND)*. This function is necessary so that the architecture can learn novel features when the data distribution changes.

To do so, we estimate in an online manner the distance distribution between input patches and their nearest centroid (separately for each layer). The novelty detection threshold at layer l is denoted by $\hat{D}_l$ and it is defined as the 95-th percentile (β = 0.95) of this distance distribution.

IV. Dual-memory organization: New centroids are stored temporarily in a *Short-Term Memory (STM)* of limited capacity Δ, separately for each layer. Every time a centroid is selected as the nearest neighbor of an input patch, it is updated based on (2). If an STM centroid $\mathbf{c_{l,j}}$ is selected more than θ times, it is copied to the *Long-Term Memory (LTM)* for that layer. We refer to this event as *memory consolidation*. The LTM has (practically) unlimited capacity and the learning rate is much smaller (in our experiments the LTM learning rate is set to zero).

[1] We drop the time index t from this point on but it is still implied that the centroids are dynamically learned over time.

[2] We have also experimented with the L1 metric with only minimal differences. Different distance metrics may be more appropriate for other types of data.

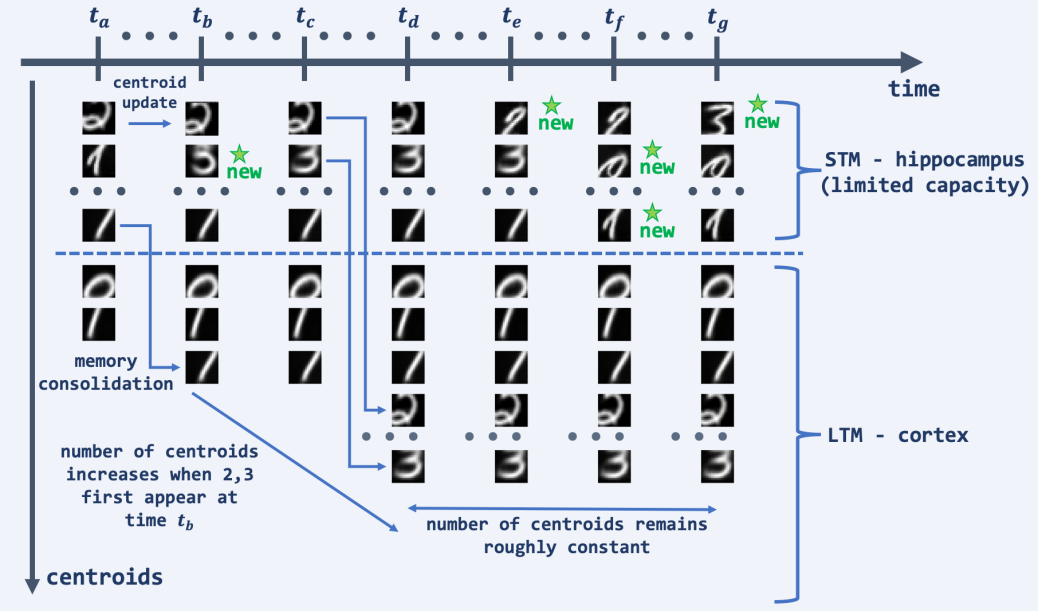

Figure 1. A hypothetical pool of STM and LTM centroids visualized at seven time instants. From t_a to t_b, a centroid is moved from STM to LTM after it has been selected θ times. At time t_b, unlabeled examples from classes '2' and '3' first appear, triggering novelty detection and new centroids are created in STM. These centroids are moved into LTM by t_d. From t_d to t_g, the pool of LTM centroids remains the same because no new classes are seen. The pool of STM centroids keeps changing when we receive "outlier" inputs of previously seen classes. Those centroids are later replaced (Least-Recently-Used policy) due to the limited capacity of the STM pool.

This memory organization is inspired by the Complementary Learning Systems framework (Kumaran et al., 2016), where the STM role is played by the hippocampus and the LTM role by the cortex. This dual-memory scheme is necessary to distinguish between infrequently seen patterns that can be forgotten ("outliers"), and new patterns that are frequently seen after they first appear ("novelty").

We initialize the pool of STM centroids at each layer using randomly sampled patches from the first few images of the unlabeled stream. When the STM pool of centroids at a layer is full, the introduction of a new centroid (created through novelty detection) causes the removal of an earlier centroid. We use the Least-Recently Used (LRU) policy to remove atypical centroids that have not been recently selected by any input. Figure 1 illustrates this dual-memory organization.

3. Classification using STAM

Given a small amount of labeled data, STAM can be used in classification tasks. We emphasize that the labeled data is not used for representation learning – it is only used to associate previously learned features with a given set of classes.

I. Associating centroids with classes: Suppose we are given some labeled examples $X_L(t)$ from a set of classes $L(t)$ at time t. We can use these labeled examples to associate existing LTM centroids at time t (learned strictly from unlabeled data) with the set of classes in $L(t)$.

Given a labeled example of class k, suppose that there is a patch $\mathbf{x}$ in that example for which the nearest centroid is $\mathbf{c}$. That patch contributes the following association between centroid $\mathbf{c}$ and class k:

$$f_{\mathbf{x},\mathbf{c}}(k) = e^{-d(\mathbf{x},\mathbf{c})/\bar{D}_l} \tag{3}$$

where $\bar{D}_l$ is a normalization constant (calculated as the average distance between input patches and centroids).

The *class-association vector* $\mathbf{g_c}$ between centroid $\mathbf{c}$ and any class k is computed aggregating all such associations, across all labeled examples in X_L:

$$g_{\mathbf{c}}(k) = \frac{\sum_{\mathbf{x}\in X_L(k)} f_{\mathbf{x},\mathbf{c}}(k)}{\sum_{k'\in L(t)} \sum_{\mathbf{x}\in X_L(k')} f_{\mathbf{x},\mathbf{c}}(k')}, \quad k = 1 \ldots L(t) \tag{4}$$

where $X_L(k)$ refers to labeled examples belonging to class k. Note that $\sum_k g_{\mathbf{c}}(k)$=1.

II. Class informative centroids: If a centroid is associated with only one class k ($g_{\mathbf{c}}(k) = 1$), only labeled examples of that class select that centroid. At the other extreme, if a centroid is equally likely to be selected by examples of any labeled class, ($g_{\mathbf{c}}(k) \approx 1/|L(t)|$), the selection of that centroid does not provide any significant information for the class of the corresponding input. We identify the cen-

troids that are *Class INformative (CIN)* as those that are associated with at least one class significantly more than expected by chance. Specifically, a centroid $\mathbf{c}$ is CIN if

$$\max_{k \in L(t)} g_{\mathbf{c}}(k) > \frac{1}{|L(t)|} + \gamma \tag{5}$$

where $\frac{1}{|L(t)|}$ is the chance term and γ is the significance term.

III. Classification using a hierarchy of centroids: At test time, we are given an input $\mathbf{x}$ of class $k(\mathbf{x})$ and infer its class as $\hat{k}(\mathbf{x})$. The classification task is a "biased voting" process in which every patch of $\mathbf{x}$, at any layer, votes for a single class as long as that patch selects a CIN centroid.

Specifically, if a patch $\mathbf{x_{l,m}}$ of layer l selects a CIN centroid $\mathbf{c}$, then that patch votes $v_{l,m}(k) = \max_{k \in L(t)} g_{\mathbf{c}}(k)$ for the class k that has the highest association with $\mathbf{c}$, and zero for all other classes. If c is *not* a CIN centroid, the vote of that patch is $v_{l,m}(k) = 0$ for all classes.

The vote of layer l for class k is the average vote across all patches in layer l (as illustrated in Figure 2):

$$v_l(k) = \frac{\sum_{m \in M_l} v_{l,m}(k)}{|M_l|} \tag{6}$$

where M_l is the set of patches in layer l. The final inference for input $\mathbf{x}$ is the class with the highest cumulative vote across all layers:

$$\hat{k}(\mathbf{x}) = \arg\max_{k'} \sum_{l=1}^{\Lambda} v_l(k) \tag{7}$$

4. Clustering using STAM

We can also use STAM representations in unsupervised tasks, such as offline clustering. To do this, we first define an embedding function that maps a given vector $\mathbf{x}$ into the space defined by STAM LTM centroids. In particular, the embedding is defined as $\Phi(\mathbf{x}) : \mathbb{R}^{\mathbf{n}} \rightarrow \mathbb{R}^{|\mathbf{C}|}$, where the element $j = 1...|C|$ of $\Phi(\mathbf{x})$ is the normalized distance (Equation equation 3) between the LTM centroid c_l and its *closest patch* in $\mathbf{x}$. The embedding vector represents how strongly each feature (LTM centroid) is present anywhere in the given input. The embedding vectors of a given dataset are then clustered offline using k-means for a given value of k. Any other clustering algorithm could be used instead.

5. Evaluation

To evaluate the STAM architecture in the UPL context, we consider a data stream in which small groups of classes appear in successive *phases*, referred to as **Incremental UPL**. New classes are introduced two at a time in each phase, and they are only seen in that phase. STAM must be able to both recognize new classes when they are first seen in the stream, and to also remember all previously learned classes without catastrophic forgetting.

Another evaluation scenario is **Uniform UPL**, where all classes appear with equal probability throughout the stream. The results for Uniform UPL are shown in SM-F in the extended verison of the paper found under the same title on arXiv.

For brevity, we include results for three datasets: MNIST (Lecun et al., 1998), EMNIST (balanced split with 47 classes) (Cohen et al., 2017), and SVHN (Netzer et al., 2011) (we have also experimented with CIFAR). For each dataset we utilize the standard training and test splits. We preprocess the images by applying per-patch normalization (instead of image normalization), and color images are transformed to grayscale. More information about the image preprocessing can be found in SM-G in the extended verison of the paper found under the same title on arXiv.

We create the training stream by randomly selecting, with equal probability, N_p data examples from the classes seen during each phase. N_p is set to 8000, 8000, and 2000 for MNIST, SVHN, and EMNIST respectively. More information about the impact of the stream size can be found in SM-D in the extended verison of the paper found under the same title on arXiv.

In the classification task, we select a small portion of the training dataset as the labeled examples that are available only to the classifier.

In each task, we average results over three different unlabeled data streams. During testing, we select 100 random examples of each class from the test dataset. This process is repeated five times for each training stream (i.e., a total of fifteen results per experiment). The following plots show mean$\pm$standard deviation ranges.

We utilize one F72s V2 and one NC6s V2 virtual machine from Microsoft's Azure cloud computing service to perform all experiments.

For all datasets, we use a 3-layer STAM hierarchy. The hyperparameter values are tabulated in SM-A in the extended verison of the paper found under the same title on arXiv. The robustness of the results with respect to these values is shown in SM-E in the extended verison of the paper found under the same title on arXiv.

Baseline Methods: We evaluate the STAM architecture comparing its performance to few basic unsupervised and self-supervised models that we have adapted in the UPL context. We emphasize that there are no prior approaches

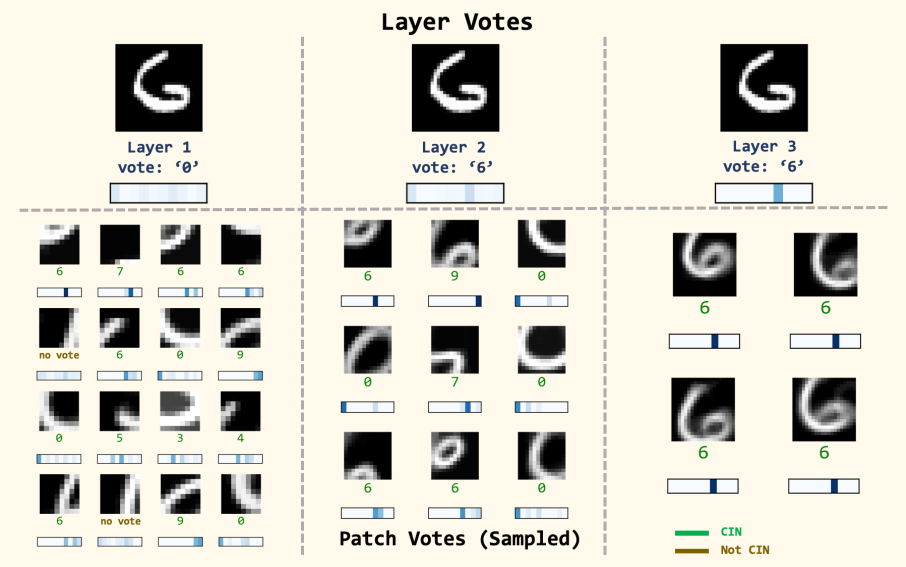

Figure 2. An example of the classification process. Every patch (at any layer) that selects a CIN centroid votes for the single class that has the highest association with. These patch votes are first averaged at each layer. The final inference is the class with the highest cumulative vote across all layers.

that are directly applicable to the UPL problem, and so we cannot perform direct comparisons between STAM and "competing" models. In follow-up work, we plan to modify existing methods that were designed to address different problems (such as CURL, GEM or iCARL) in the UPL context and compare them with STAM. The baselines we consider here are:

(I) a Convolutional AutoEncoder (CAE) trained to minimize Euclidean reconstruction error,

(II) a rotation-based self-supervised method which learns the auxiliary task of predicting image rotations based on *RotNet* (Gidaris et al., 2018), and

(III) offline Principal Component Analysis (PCA), just for reference purposes, as it is probably the simplest baseline.

A detailed description of these baseline models is presented in SM-B in the extended verison of the paper found under the same title on arXiv.

To satisfy the stream requirement of UPL, the number of training epochs for the CAE and RotNet models is set to one. This is necessary so that each unlabeled example is processed only once. Deep learning methods become weaker in this streaming scenario because they cannot train iteratively over several epochs on the same dataset. For all baselines, the classification task is performed using a K nearest-neighbor (KNN) classifier – we have experimented with various values of K and other single-pass classifiers, and report only the best performing results here.

We have also compared the memory requirement of STAM (storing centroids at STM and LTM) with the memory requirement of the CAE and RotNet baselines (storing neural network weights). The results of that comparison appear in SM-H in the extended verison of the paper found under the same title on arXiv. For instance, STAM has 17% of RotNet's memory footprint.

Classification Task: We focus on an *expanding classification task*, meaning that in each phase we need to classify *all* classes seen so far. The results for the classification task are given in Figure 3. Note that we use only 10 labeled examples per class for MNIST and EMNIST, and 100 examples per class for SVHN.

As we introduce new classes in the training stream, the average accuracy per phase decreases for all methods in each dataset. This is expected, as the task gets *more difficult* after each phase. We focus on which method performs best in each task, and which methods see a smaller decrease in accuracy per phase. In the first dataset (MNIST), we observe that STAM performs consistently better than RotNet and CAE, and STAM is less vulnerable to catastrophic forgetting. For SVHN, the trend is similar after the first phase but the difference between STAM and RotNet is much smaller. Finally, in EMNIST, we see a consistently higher accuracy with STAM compared to the deep learning baselines. For additional analysis and discussion of these results, please also refer to SM-C in the extended verison of the paper found under the same title on arXiv.

Clustering Task: Given that we have the same number of test vectors per class, we associate each cluster with the

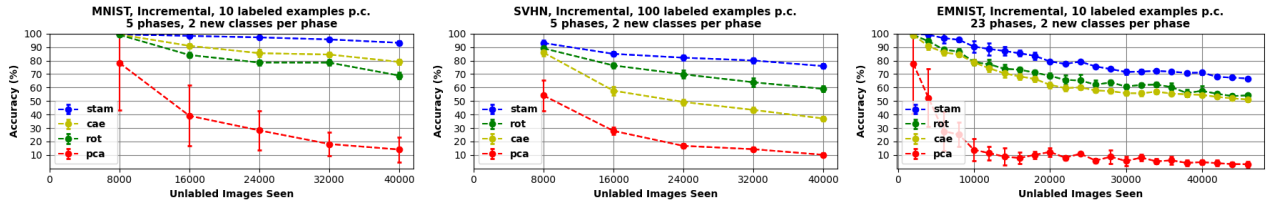

Figure 3. Classification accuracy for MNIST (left), SVHN (center), and EMNIST (right). The task is expanding classification for incremental UPL, i.e., recognize all classes seen so far. Note that the number of labeled examples is 10 per class for MNIST and EMNIST and 100 per class for SVHN.

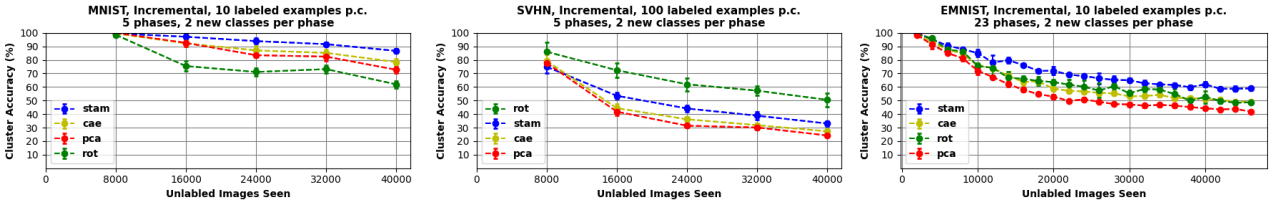

Figure 4. Clustering accuracy for MNIST (left), SVHN (center), and EMNIST (right). The task is expanding clustering for incremental UPL. The number of clusters is equal to the number of classes in the data stream seen up to that point in time.

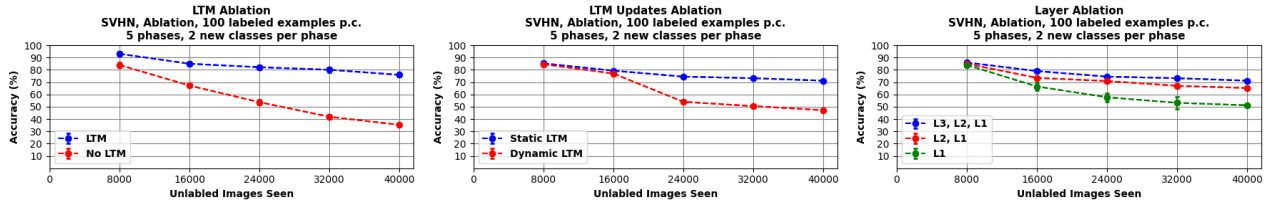

Figure 5. Ablation study: A STAM architecture without LTM (left), a STAM architecture in which the LTM centroids are adjusted with the same learning rate α as in STM (center), and a STAM architecture with removal of layers (right)

most-represented class in that cluster. Any instances of another class in that cluster are counted as errors. The number of clusters k is equal to twice the number of classes seen up to that phase in the unlabeled datastream. The results of the clustering task are given in Figure 4.

For MNIST, STAM still performs consistently better than the two other models, and its accuracy stays almost constant going from 4 classes to 10 classes. For SVHN, RotNet performs significantly better. Finally, for EMNIST, STAM outperforms the two deep learning methods without experiencing significant loss of accuracy after the first 10 phases (20 classes).

Ablation studies: Several STAM ablations are presented in Figure 5. On the left, we remove the LTM capability and only use STM centroids for classification. During the first two phases, there is little (if any) difference in classification accuracy. However, we see a clear dropoff during phases 3-5. This suggests that, without the LTM mechanisms, features from classes that are no longer seen in the stream are forgotten over time, and STAM can only successfully classify classes that have been recently seen.

We also investigate the importance of having static LTM centroids rather than dynamic centroids (Fig. 5-middle). Specifically, we replace the static LTM with a dynamic LTM in which the centroids are adjusted with the same learning rate parameter α, as in STM. The accuracy suffers drastically because the introduction of new classes "takes over" LTM centroids of previously learned classes, after the latter are removed from the stream. Similar to the removal of LTM, we do not see the effects of "forgetting" until phases 3-5. Note that the degradation due to a dynamic LTM is less severe than that from removing LTM completely.

Finally, we look at the effects of removing layers from the STAM hierarchy (Fig. 5-right). We see a small drop in accuracy after removing layer 3, and a large drop in accuracy after also removing layer 2. The importance of having a deeper hierarchy would be more pronounced in datasets with higher-resolution images or videos, potentially show-

ing multiple objects in the same frame. In such cases, CIN centroids can appear at any layer, starting from the lowest to the highest.

Additional results: The reader can find additional experimental results in the Supp-Material section that focus on the questions: how does the number of LTM centroids increase with time and what fraction of them are "Class Informative" (in classification tasks), how does the accuracy of STAM vary with the number of labeled examples per class, and how do the various hyperparameters of the STAM architecture affect classification accuracy?

6. Related Work

The UPL problem has some similarities with several recent approaches in the machine learning literature but it is also different in important aspects we describe in this section. Each paragraph highlights the most relevant prior work and explains how it is different from UPL.

I: Continual learning: In addition to CL models cited in the introduction, other *supervised* CL methods include regularization-based approaches (Aljundi et al., 2018; Golkar et al., 2019; Hayes & Kanan, 2019; Kirkpatrick et al., 2017; Yoon et al., 2018; Zenke et al., 2017), expanding architectures (Lomonaco & Maltoni, 2017; Maltoni & Lomonaco, 2019; Rusu et al., 2016), and distillation-based methods (Lee et al., 2019; 2020; Li & Hoiem, 2017). Their main difference with UPL and STAM is that they are designed for supervised learning, and it is not clear how to adapt them for non-stationary and unlabeled data streams.

II. Offline unsupervised learning: Additional *offline* representation learning methods include clustering (Caron et al., 2018; Jiang et al., 2017a; Xie et al., 2016; Yang et al., 2016), generative models (Eslami et al., 2016; Jiang et al., 2017b; Kosiorek et al., 2018; 2019), information theory (Hjelm et al., 2019; Ji et al., 2019), among others. These methods require prior information about the number of classes present in a given dataset (to set the number of cluster centroids or class outputs) and iterative training (i.e. data replay), and therefore cannot be directly applied in the UPL setting.

III. Semi-supervised learning (SSL): SSSL methods require labeled data during the representation learning stage and so they are not compatible with UPL (Kingma et al., 2014; Lee, 2013; Miyato et al., 2018; Oliver et al., 2018; Rasmus et al., 2015; Springenberg, 2015; Tarvainen & Valpola, 2017).

IV. Few-shot learning (FSL) and Meta-learning: These methods recognize object classes not seen in the training set with only a single (or handful) of labeled examples (Fei-Fei et al., 2006; Finn et al., 2017; Ren et al., 2018; Snell et al., 2017). Similar to SSL, FSL methods require labeled data to learn representations and therefore are not applicable in the UPL context. Centroid networks (Huang et al., 2019) do not require labeled examples at inference time but require labeled examples for training.

V. Multi-Task Learning (MTL): Any MTL method that involves separate heads for different tasks is not compatible with UPL because task boundaries are not known a priori in UPL (Ruder, 2017). MTL methods that require pre-training on a large labeled dataset are also not applicable to UPL (Pan & Yang, 2010; Yosinski et al., 2014).

VI. Online and Progressive Learning: Many earlier methods learn in an online manner, meaning that data is processed in fixed batches and discarded afterwards. This includes progressive learning (Venkatesan & Er, 2016) and streaming with limited supervision (Chiotellis et al., 2018; Li et al., 2018; Loo & Marsono, 2015), both of which require labeled data in the training stream.

VII. Continual Unsupervised Representation Learning (CURL): Similar to STAM, CURL also focuses on the problem of continual unsupervised learning from non-stationary data with unknown task boundaries (Rao et al., 2019). Its main difference with STAM however is that it is not a streaming method, and so it does not require that each example is seen only once. The CURL model requires gradient-based optimization, going through the same data multiple times. Another difference with STAM is that catastrophic forgetting in CURL is addressed through a generative model that also needs to be learned.

VIII. Data dimensionality and clustering-based representation learning: As mentioned earlier in this section, clustering has been used successfully in the past for offline representation learning (e.g., (Coates et al., 2011; Coates & Ng, 2012)). Its effectiveness, however, gradually drops as the input dimensionality increases (Beyer et al., 1999; Hinneburg et al., 2000). In the STAM architecture, we avoid this issue by clustering smaller subvectors (patches) of the input data. If those subvectors are still of high dimensionality, another approach is to reduce the *intrinsic dimensionality* of the input data at each layer by reconstructing that input using representations (selected centroids) from the previous layer.

VIII. Related work to other STAM components: STAM relies on online clustering. This algorithm can be implemented with a rather simple recurrent neural network of excitatory and inhibitory spiking neurons, as shown recently (Pehlevan et al., 2017). The novelty detection component of STAM is related to the problem of anomaly detection in streaming data (Dasgupta et al., 2018) — and the simple algorithm currently in STAM can be replaced with more sophisticated methods (e.g., (Cui et al., 2016; Yong

et al., 2012)). Finally, brain-inspired dual-memory systems have been proposed before for memory consolidation (e.g., (Kemker & Kanan, 2018; Parisi et al., 2018; Shin et al., 2017)).

7. Discussion

The STAM architecture aims to address the following desiderata that is often associated with Lifelong Learning (Parisi et al., 2019):

I. Online learning: STAMs update the learned features with every observed example. There is no separate training stage for specific tasks, and inference can be performed in parallel with learning.

II. Transfer learning: The features learned by the STAM architecture in earlier phases can be also encountered in the data of future tasks (forward transfer). Additionally, new centroids committed to LTM can also be closer to data of earlier tasks (backward transfer).

III. Resistance to catastrophic forgetting: The STM-LTM memory hierarchy of the STAM architecture mitigates catastrophic forgetting by committing to "permanent storage" (LTM) features that have been often seen in the data during any time period of the training period.

IV. Expanding learning capacity: The unlimited capacity of LTM allows the system to gradually learn more features as it encounters new classes and tasks. The relatively small size of STM, on the other hand, forces the system to forget features that have not been recalled frequently enough after their creation.

V. No direct access to previous experience: STAM only needs to store data centroids in a hierarchy of increasing receptive fields – there is no need to store previous exemplars or to learn a generative model that can produce such examples.

References

Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., and Tuytelaars, T. Memory aware synapses: Learning what (not) to forget. In *ECCV*, 2018.

Aljundi, R., Belilovsky, E., Tuytelaars, T., Charlin, L., Caccia, M., Lin, M., and Page-Caccia, L. Online continual learning with maximal interfered retrieval. In *Advances in Neural Information Processing Systems*, pp. 11849–11860, 2019a.

Aljundi, R., Lin, M., Goujaud, B., and Bengio, Y. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems*, pp. 11816–11825, 2019b.

Bengio, Y. How auto-encoders could provide credit assignment in deep networks via target propagation. *arXiv preprint arXiv:1407.7906*, 2014.

Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, August 2013. ISSN 0162-8828. doi: 10.1109/TPAMI.2013.50.

Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., and Raffel, C. A. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*, pp. 5050–5060, 2019.

Beyer, K. S., Goldstein, J., Ramakrishnan, R., and Shaft, U. When is "nearest neighbor" meaningful? In *Proceedings of the 7th International Conference on Database Theory*, ICDT '99, pp. 217–235, London, UK, UK, 1999. Springer-Verlag. ISBN 3-540-65452-6.

Caron, M., Bojanowski, P., Joulin, A., and Douze, M. Deep clustering for unsupervised learning of visual features. In *The European Conference on Computer Vision (ECCV)*, September 2018.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. Efficient lifelong learning with a-GEM. In *International Conference on Learning Representations*, 2019a.

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H., and Ranzato, M. Continual learning with tiny episodic memories. *arXiv preprint arXiv:1902.10486*, 2019b.

Chiotellis, I., Zimmermann, F., Cremers, D., and Triebel, R. Incremental semi-supervised learning from streams for object classification. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5743–5749. IEEE, 2018.

Coates, A. and Ng, A. Y. *Learning feature representations with k-means*, pp. 561–580. Springer, 2012.

Coates, A., Ng, A., and Lee, H. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223, 2011.

Cohen, G., Afshar, S., Tapson, J., and van Schaik, A. Emnist: an extension of mnist to handwritten letters. *ArXiv*, abs/1702.05373, 2017.

Cui, Y., Ahmad, S., and Hawkins, J. Continuous online sequence learning with an unsupervised neural network model. *Neural Comput.*, 28(11):2474–2504, November 2016. ISSN 0899-7667. doi: 10.1162/NECO_a_00893.

Dasgupta, S., Sheehan, T. C., Stevens, C. F., and Navlakha, S. A neural data structure for novelty detection. *Proceedings of the National Academy of Sciences*, 115(51): 13093–13098, 2018.

Doersch, C., Gupta, A., and Efros, A. A. Unsupervised visual representation learning by context prediction. *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1422–1430, 2015.

Eslami, S. M. A., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., Kavukcuoglu, K., and Hinton, G. E. Attend, infer, repeat: Fast scene understanding with generative models. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 3233–3241, USA, 2016. Curran Associates Inc. ISBN 978-1-5108-3881-9.

Fei-Fei, L., Fergus, R., and Perona, P. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, April 2006. ISSN 0162-8828. doi: 10.1109/TPAMI.2006.79.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 1126–1135. JMLR.org, 2017.

Gepperth, A. and Karaoguz, C. Incremental learning with self-organizing maps. *2017 12th International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization (WSOM)*, pp. 1–8, 2017.

Gidaris, S., Singh, P., and Komodakis, N. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations*, 2018.

Goldstone, R. L. Perceptual learning. *Annual review of psychology*, 49(1):585–612, 1998.

Golkar, S., Kagan, M., and Cho, K. Continual learning via neural pruning. *arXiv preprint arXiv:1903.04476*, 2019.

Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., and Bengio, Y. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

Hayes, T. L. and Kanan, C. Lifelong machine learning with deep streaming linear discriminant analysis. *arXiv preprint arXiv:1909.01520*, 2019.

Hayes, T. L., Cahill, N. D., and Kanan, C. Memory efficient experience replay for streaming learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9769–9776. IEEE, 2019.

Hinneburg, A., Aggarwal, C. C., and Keim, D. A. What is the nearest neighbor in high dimensional spaces? In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, pp. 506–515, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-715-3.

Hjelm, D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. Learning deep representations by mutual information estimation and maximization. In *ICLR 2019*. ICLR, April 2019.

Hsu, Y.-C., Liu, Y.-C., Ramasamy, A., and Kira, Z. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. In *NeurIPS Continual learning Workshop*, 2018.

Huang, G., Larochelle, H., and Lacoste-Julien, S. Centroid networks for few-shot clustering and unsupervised few-shot classification. *arXiv preprint arXiv:1902.08605*, 2019.

Ji, X., Henriques, J., and Vedaldi, A. Invariant infromation clustering for unsupervised image classification and segmentation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

Jiang, Z., Zheng, Y., Tan, H., Tang, B., and Zhou, H. Variational deep embedding: An unsupervised and generative approach to clustering. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pp. 1965–1972. AAAI Press, 2017a. ISBN 9780999241103.

Jiang, Z., Zheng, Y., Tan, H., Tang, B., and Zhou, H. Variational deep embedding: An unsupervised and generative approach to clustering. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pp. 1965–1972. AAAI Press, 2017b. ISBN 978-0-9992411-0-3.

Kemker, R. and Kanan, C. Fearnet: Brain-inspired model for incremental learning. *International Conference on Learning Representations (ICLR)*, 2018.

Kemker, R., McClure, M., Abitino, A., Hayes, T., and Kanan, C. Measuring catastrophic forgetting in neural networks. *AAAI Conference on Artificial Intelligence*, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kingma, D. P., Rezende, D. J., Mohamed, S., and Welling, M. Semi-supervised learning with deep generative models. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pp. 3581–3589, Cambridge, MA, USA, 2014. MIT Press.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 2017.

Kosiorek, A., Kim, H., Teh, Y. W., and Posner, I. Sequential attend, infer, repeat: Generative modelling of moving objects. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 8606–8616. Curran Associates, Inc., 2018.

Kosiorek, A., Sabour, S., Teh, Y. W., and Hinton, G. E. Stacked capsule autoencoders. In *Advances in Neural Information Processing Systems*, pp. 15486–15496, 2019.

Kumaran, D., Hassabis, D., and McClelland, J. L. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences*, 20(7):512–534, 2016.

Kuo, C.-W., Ma, C.-Y., Huang, J.-B., and Kira, Z. Manifold graph with learned prototypes for semi-supervised image classification. *arXiv preprint arXiv:1906.05202*, 2019.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. ISSN 0018-9219. doi: 10.1109/5.726791.

Lee, D.-H. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks. *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)*, 07 2013.

Lee, K., Lee, K., Shin, J., and Lee, H. Overcoming catastrophic forgetting with unlabeled data in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 312–321, 2019.

Lee, S., Ha, J., Zhang, D., and Kim, G. A neural dirichlet process mixture model for task-free continual learning. *arXiv preprint arXiv:2001.00689*, 2020.

Li, Y., Wang, Y., Liu, Q., Bi, C., Jiang, X., and Sun, S. Incremental semi-supervised learning on streaming data. *Pattern Recognition*, 88, 11 2018. doi: 10.1016/j.patcog.2018.11.006.

Li, Z. and Hoiem, D. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Liu, X., Wu, C., Menta, M., Herranz, L., Raducanu, B., Bagdanov, A. D., Jui, S., and van de Weijer, J. Generative feature replay for class-incremental learning. *arXiv preprint arXiv:2004.09199*, 2020.

Lomonaco, V. and Maltoni, D. Core50: a new dataset and benchmark for continuous object recognition. *arXiv preprint arXiv:1705.03550*, 2017.

Loo, H. R. and Marsono, M. N. Online data stream classification with incremental semi-supervised learning. In *Proceedings of the Second ACM IKDD Conference on Data Sciences*, CoDS '15, pp. 132–133, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450334365. doi: 10.1145/2732587.2732614.

Lopez-Paz, D. and Ranzato, M. Gradient episodic memory for continual learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6470–6479, USA, 2017. Curran Associates Inc. ISBN 978-1-5108-6096-4.

Maltoni, D. and Lomonaco, V. Continuous learning in single-incremental-task scenarios. *Neural Networks*, 116:56–73, 2019.

Miyato, T., Maeda, S.-i., Ishii, S., and Koyama, M. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

Oliver, A., Odena, A., Raffel, C. A., Cubuk, E. D., and Goodfellow, I. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems*, pp. 3235–3246, 2018.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Pan, S. and Yang, Q. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22 (10):1345–1359, 2010.

Parisi, G. I., Tani, J., Weber, C., and Wermter, S. Lifelong learning of spatiotemporal representations with dual-memory recurrent self-organization. *Frontiers in neurorobotics*, 12:78, 2018.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54 – 71, 2019. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2019.01.012.

Pehlevan, C., Genkin, A., and Chklovskii, D. B. A clustering neural network model of insect olfaction. In *2017 51st Asilomar Conference on Signals, Systems, and Computers*, pp. 593–600. IEEE, 2017.

Rao, D., Visin, F., Rusu, A., Pascanu, R., Teh, Y. W., and Hadsell, R. Continual unsupervised representation learning. In *Advances in Neural Information Processing Systems 32*, pp. 7645–7655. Curran Associates, Inc., 2019.

Rasmus, A., Berglund, M., Honkala, M., Valpola, H., and Raiko, T. Semi-supervised learning with ladder networks. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28*, pp. 3546–3554. Curran Associates, Inc., 2015.

Rebuffi, S., Kolesnikov, A., Sperl, G., and Lampert, C. H. icarl: Incremental classifier and representation learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR'17, pp. 5533–5542, 2017.

Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J. B., Larochelle, H., and Zemel, R. S. Meta-learning for semi-supervised few-shot classification. In *Proceedings of 6th International Conference on Learning Representations ICLR*, 2018.

Ruder, S. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Shin, H., Lee, J. K., Kim, J., and Kim, J. Continual learning with deep generative replay. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R.,

Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 2990–2999. Curran Associates, Inc., 2017.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pp. 4077–4087, 2017.

Sohn, K., Berthelot, D., Li, C.-L., Zhang, Z., Carlini, N., Cubuk, E. D., Kurakin, A., Zhang, H., and Raffel, C. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*, 2020.

Springenberg, J. T. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015.

Tarvainen, A. and Valpola, H. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 1195–1204. Curran Associates, Inc., 2017.

Tschannen, M., Bachem, O., and Lucic, M. Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*, 2018.

van de Ven, G. M. and Tolias, A. S. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

Venkatesan, R. and Er, M. J. A novel progressive learning technique for multi-class classification. *Neurocomput.*, 207(C):310–321, September 2016. ISSN 0925-2312. doi: 10.1016/j.neucom.2016.05.006.

Xie, J., Girshick, R., and Farhadi, A. Unsupervised deep embedding for clustering analysis. In Balcan, M. F. and Weinberger, K. Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 478–487, New York, New York, USA, 20–22 Jun 2016. PMLR.

Yang, J., Parikh, D., and Batra, D. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5147–5156, 2016.

Yong, S.-P., Deng, J. D., and Purvis, M. K. Novelty detection in wildlife scenes through semantic context modelling. *Pattern Recogn.*, 45(9):3439–3450, September 2012. ISSN 0031-3203. doi: 10.1016/j.patcog.2012.02.036.

Yoon, J., Yang, E., Lee, J., and Hwang, S. J. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 3320–3328. Curran Associates, Inc., 2014.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, 2017.

Zeno, C., Golan, I., Hoffer, E., and Soudry, D. Task agnostic continual learning using online variational bayes. *arXiv preprint arXiv:1803.10123*, 2018.

Zhou, G., Sohn, K., and Lee, H. Online incremental feature learning with denoising autoencoders. In Lawrence, N. D. and Girolami, M. (eds.), *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pp. 1453–1461, La Palma, Canary Islands, 21–23 Apr 2012. PMLR.