Knowledge Capture and Replay for Continual Learning

Saisubramaniam Gopalakrishnan [1,4], Pranshu Ranjan Singh [1,4], Haytham Fayek [3],
Savitha Ramasamy [1,2,4], ArulMurugan Ambikapathi [1,2,4]
[1] Institute for Infocomm Research (I^{2}R), A*STAR, Singapore [*]
[2] Artificial Intelligence, Analytics And Informatics (AI^3), A*STAR, Singapore
[3] RMIT University, Australia
[4] CNRS@CREATE LTD, 1 Create Way, 08-01 CREATE Tower, Singapore 138602 [†]
{g_saisubramaniam, pranshurs, ramasamysa, arul}@i2r.a-star.edu.sg, haytham.fayek@ieee.org

Abstract

Deep neural networks model data for a task or a sequence of tasks, where the knowledge extracted from the data is encoded in the parameters and representations of the network. Extraction and utilization of these representations is vital when data is no longer available in the future, especially in a continual learning scenario. We introduce flashcards*, which are visual representations that* capture *the encoded knowledge of a network as a recursive function of some predefined random image patterns. In a continual learning scenario, flashcards help to prevent catastrophic forgetting by consolidating the knowledge of all the previous tasks. Flashcards are required to be constructed only before learning the subsequent task, hence, they are independent of the number of tasks trained before, making them task agnostic. We demonstrate the efficacy of flashcards in capturing learned knowledge representation (as an alternative to the original data), and empirically validate on a variety of continual learning tasks: reconstruction, denoising, and task-incremental classification, using several heterogeneous (varying background and complexity) benchmark datasets. Experimental evidence indicates that: (i) flashcards as a replay strategy is* task agnostic*, (ii) performs better than generative replay, and (iii) is on par with episodic replay without additional memory overhead.*

1. Introduction

Despite the success of deep neural networks in numerous domains [5, 11, 15], it remains a challenge to extract and reuse the knowledge embedded in the representations

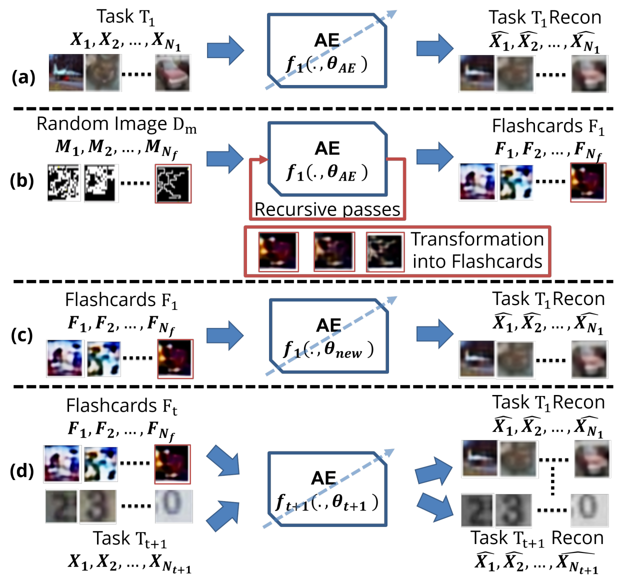

Task T$_1$
$X_1, X_2, \dots, X_{N_1}$
(a)
AE
$f_1(\cdot, \theta_{AE})$
Task T$_1$ Recon
$\widehat{X_1}, \widehat{X_2}, \dots, \widehat{X_{N_1}}$
Random Image D$_m$
$M_1, M_2, \dots, M_{N_f}$
(b)
AE
$f_1(\cdot, \theta_{AE})$
Recursive passes
Flashcards F$_1$
$F_1, F_2, \dots, F_{N_f}$
Transformation into Flashcards
Flashcards F$_1$
$F_1, F_2, \dots, F_{N_f}$
(c)
AE
$f_1(\cdot, \theta_{new})$
Task T$_1$ Recon
$\widehat{X_1}, \widehat{X_2}, \dots, \widehat{X_{N_1}}$
Flashcards F$_t$
$F_1, F_2, \dots, F_{N_f}$
(d)
AE
$f_{t+1}(\cdot, \theta_{t+1})$
Task T$_1$ Recon
$\widehat{X_1}, \widehat{X_2}, \dots, \widehat{X_{N_1}}$
Task T$_{t+1}$
$X_1, X_2, \dots, X_{N_{t+1}}$
Task T$_{t+1}$ Recon
$\widehat{X_1}, \widehat{X_2}, \dots, \widehat{X_{N_{t+1}}}$

Figure 1. Flashcards for knowledge capture and replay. Step (a): Trained autoencoder (AE) for Task T_1; Step (b): Construct flashcards from frozen AE using maze patterns via recursive passes; Step (c): Replay using flashcards on a new network to remember Task T_1; Step (d) Replay using flashcards F_t (containing consolidated knowledge from task T_1 to T_t) while training for Task T_{t+1}.

of trained models for similar or other downstream scenarios [2]. Approaches such as transfer learning [25] and knowledge distillation [7] enable representational or functional translation [9] from one model to the other. However, they cannot maintain a unified representation that evolves with the characteristics of data. Knowledge captured from the past should be consolidated and preserved as some representation with low computational and memory expense, especially, when the past data is no longer available.

In particular, learning new tasks affect retention of previous knowledge [19]. Continual learning (CL) [21] was

[*]This research was partially funded through SC20/19-128310-CORE.
[†]This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme.

proposed as a candidate for such retention, and is gaining attention [18]. Essentially, CL aims to make deep neural networks learn continually from a sequence of tasks, without catastrophically forgetting the past sequences/tasks. There is a need to develop a reliable method that can effectively capture and re-purpose knowledge from a learned model, which can then be exploited in the CL setting. While CL approaches are predominantly studied on classification (class/task incremental learning), its potential remains relatively less explored in other scenarios such as task agnostic unsupervised reconstruction, denoising etc.

In this work, we introduce *flashcards* for capturing consolidated learned knowledge representations of all the previous tasks. A schematic illustration is shown in Figure 1. Random image patterns, when passed recursively through a trained autoencoder (obtained at end of step (a)), capture its representations with each pass, and are transformed into flashcards (step (b)). Step (c) uses the obtained flashcards to train a new network (single task), and step (d) addresses the multi-task / continual setting. We first evaluate flashcards' potential for capturing representations of the data (as seen by the network) and later extend their usefulness for replay while learning continually from several heterogeneous datasets, such as MNIST, Fashion MNIST, Cifar10, SVHN, Omniglot, ImageNet, and UCMerced for a variety of applications such as reconstruction, denoising, and task-incremental classification. To the best of our knowledge, we are the first to report continual reconstruction and denoising. In task-incremental classification, we show that flashcards replay sustains stability in the absence of task identifier. Our experimental comparisons indicate superiority over standard baselines in terms of performance and memory efficiency.

2. Knowledge Capture and Replay

We first introduce the notion of knowledge capture using flashcards and describe their construction from a trained autoencoder [1]. Consider training an autoencoder for reconstruction task T_t using dataset D_t containing N_t samples, where $D_t = \{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_{N_t}\} \subset \Re^{k\times l\times c}$, is the set of training image samples with k rows, l columns, and c channels. For the task T_t, the autoencoder is trained to maximize the likelihood $P(\mathbf{X}|\boldsymbol{\theta}_t)$, $\forall \mathbf{X} \in D_t$ using the conventional mean absolute error (MAE; Eqn. (1)) between the original and reconstructed images:

$$\min_{\boldsymbol{\theta}_t} \frac{1}{N_t}\sum_{n=1}^{N_t} |\mathbf{X}_n - \widehat{\mathbf{X}}_n|, \tag{1}$$

where $\boldsymbol{\theta}_t$ is the autoencoder network parameters (weights, biases, and batch norm parameters) for task T_t, $\mathbf{X}_n$ is a sample from the empirical data distribution of $P(X)$, and $\widehat{\mathbf{X}}_n = f_t(\mathbf{X}_n, \boldsymbol{\theta}_t)$ is the reconstructed sample. $|\mathbf{X}_n - \widehat{\mathbf{X}}_n|$ denotes the pixel-wise mean absolute error between $\mathbf{X}_n$ and $\widehat{\mathbf{X}}_n$, and $f_t(\cdot)$ is the function approximated by the autoencoder to learn (reconstruct) task T_t. In the above conventional learning setup, the parameters θ_t of an autoencoder network aim to model the knowledge in the data D_t, such as *shape, texture, color* of the images. Let the reconstruction error of the trained autoencoder be bounded by [ϵ_1,ϵ_2], i.e.,

$$\epsilon_1 \le |\mathbf{X}_n - \widehat{\mathbf{X}}_n| \le \epsilon_2,\ \forall \mathbf{X}_n \in D_t, \mathbf{X}_n \sim P(\mathbf{X}) \tag{2}$$

Let $P(\mathbf{M})$ be a different, but well-defined distribution in the same dimensional space as $P(\mathbf{X})$ (i.e., $\Re^{k\times l\times c}$), and $D_m = \{\mathbf{M}_1, \mathbf{M}_2, \ldots \mathbf{M}_{N_f}\}$, where $\mathbf{M}_i \in \Re^{k\times l\times c}$, $\mathbf{M}_i \sim P(\mathbf{M}), i = 1, \ldots, N_f$. The output of the autoencoder for any $\mathbf{M}_i \in D_m$ is

$$\widehat{\mathbf{M}_i} = f_t(\mathbf{M}_i, \boldsymbol{\theta}_t). \tag{3}$$

Since the autoencoder is trained for task T_t and as $\mathbf{M}_i \in D_m$ is sampled from another distribution, $\widehat{\mathbf{M}_i}$ will be a meaningless reconstruction of $\mathbf{M}_i$. Alternatively, it can be said that the activations obtained on passing $\mathbf{M}_i \sim P(\mathbf{M})$ through the trained $f_t(\cdot, \boldsymbol{\theta}_t)$ does not align with T_t. This can be observed from the increased MAE between $\mathbf{M}_i$ and $\widehat{\mathbf{M}_i}$, and from the t-SNE representations of the bottleneck layer (latent space) for several samples drawn from $P(\mathbf{M})$ (first two iterations of Figure 2).

Let $\mathbf{F}_{ri}$ be the output after a series of r *recursive iterations/passes* of $\mathbf{M}_i$ through the trained autoencoder ($f_t(\cdot, \boldsymbol{\theta}_t)$), i.e.,

$$\mathbf{F}_{ri} = f_t^r(f_t^{r-1}(\cdots f_t^1(\mathbf{M}_i, \boldsymbol{\theta}_t)\cdots)). \tag{4}$$

Such recursive passing (output $\rightarrow$ input $\rightarrow$ output) gradually attunes the input M_i towards F_{ri} to produce activations that are accustomed to $\boldsymbol{\theta}_t$. This can be observed in Figure 2 where the MAE reduces over recursive iterations, and the latent space representations of samples drawn from $P(\mathbf{M})$ begin to overlap with those from $P(\mathbf{X})$. Let

$$\gamma_1 \le |\mathbf{F}_{ri} - \mathbf{F}_{(r-1)i}| \le \gamma_2,\ \forall \mathbf{M}_i \in D_m, \mathbf{M}_i \sim P(\mathbf{M}), \tag{5}$$

where γ_1, and γ_2 are the lower and upper bounds of reconstruction error for any $\mathbf{M}_i \in D_m$. The aim now is to have a collection set $F_t = \{\mathbf{F}_{ri}\}_{i=1}^{N_f}$ for task T_t such that the following properties are satisfied:

- P1: $\gamma_2 < \epsilon_1$ for $\forall \mathbf{M}_i \in D_m$,
- P2: Respective latent space distribution of $F_t \sim D_t$.

Such a set of constructions (F_t) obtained through recursive iterations of random inputs (drawn from $P(\mathbf{M})$) are defined

[1]Autoencoder is used *only* to construct flashcards which can be used for different CL applications, including classification (see Section 4).

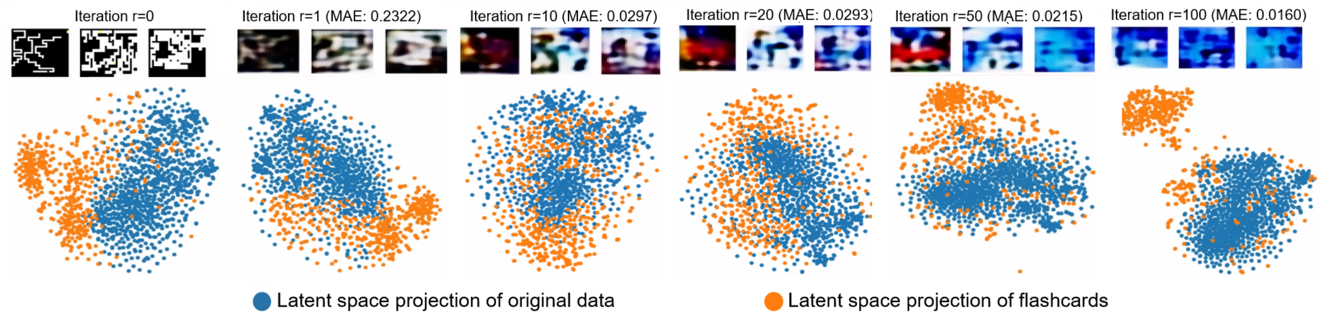


Figure 2. A sample of three Flashcards (F_t) constructions over recursive iterations, with Cifar10 as an example, difference in MAE between successive iterations with iteration number and t-SNE of latent space in 2D. It can be observed that the initial (raw) input maze patterns adapt to the texture captured by the model, as the number of recursive iterations increase (Iteration $r = 10, 20$). Also, the corresponding latent space clusters get closer. However, on further repeated passes ($r \geq 50$), the reconstruction deteriorates / smooth out (as MAE keeps reducing) and the two clusters drift apart again (P1 is satisfied but P2 is not). Best viewed in color.

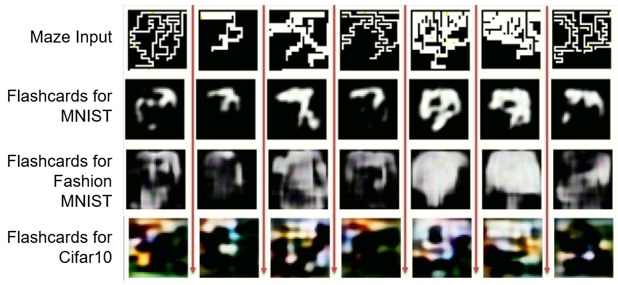


Figure 3. Transformation from maze patterns (D_m) into Flashcards (F_t) for tasks: MNIST, Fashion MNIST and Cifar10.

as *flashcards*. These flashcards are expected to capture the knowledge from the trained network and serve as a potential alternative to D_t. Figure 3 illustrates a few flashcards constructed with same set of random images, from AEs trained with different datasets. Although these flashcards are intended to capture the knowledge representations in $\boldsymbol{\theta}_t$ as a function of $\mathbf{M}_i \in D_m$, it can be observed from Figure 3 that they do not bear direct shape similarities with their respective datasets. In fact, during the recursive process, the raw input image patterns are modified/transformed to textures that are suited to the trained autoencoder model (colored textures in flashcards are observed for Cifar10). This is in-line with [6] which shows that neural networks learn predominantly texture than shape by default. Now, since an autoencoder is trained to maximize $\log P(\mathbf{X}|\boldsymbol{\theta}_t)$, the following fact holds:

Fact 1: *Let D_A and D_B be two i.i.d. datasets whose elements are drawn from $P(\mathbf{X})$. If there exists a trained autoencoder $f_A(\cdot, \boldsymbol{\theta}_A)$, to reconstruct D_A with error $[\beta_1, \beta_2]$, $\forall \mathbf{X} \in D_A$, then the autoencoder $f_A(\cdot, \boldsymbol{\theta}_A)$ can reconstruct D_B with the same error range.*

The converse is not true because any image dataset D_B with images that have few pixels with non-zero values (other pixels as zeros), can still yield smaller reconstruction error and not necessary to be drawn from $P(\mathbf{X})$ (satisfying P1). Hence, both properties P1 and P2 for F_t need to be satisfied. As P1 and P2 are dependent on the initial input distribution $P(\mathbf{M})$, it is important to have a suitable $P(\mathbf{M})$ to get F_t ($P(\mathbf{M}) = P(\mathbf{X})$ is one trivial and uninteresting solution). As finding a perfect $P(\mathbf{M})$ is an ideal research problem by itself, in this work, we use random maze pattern images as a potential candidate D_m to get F_t. The above leads to the following hypothesis:

Hypothesis 1: *Since the autoencoder trained on D_t also reconstructs elements from F_t (Property P1) and has matching latent space distribution (Property P2), flashcards (F_t) can be used as a set of pseudo-samples for D_t to learn a new autoencoder $f_t(\cdot, \boldsymbol{\theta}_{new})$, when trained from scratch using only F_t.*

2.1. Flashcard Construction

Algorithm 1 Flashcard construction

```
Require: $f_t(\cdot, \boldsymbol{\theta}_t)$: Autoencoder model for task $T_t$
Require: $N_f$: Number of flashcards
Require: $r$: Number of recursive iterations through $f_t(\cdot, \boldsymbol{\theta}_t)$
1: Let $D_m \leftarrow \{\mathbf{M}_1, \dots \mathbf{M}_{N_f}\}$, where $\mathbf{M}_i \sim P(\mathbf{M}), \forall i$,
   $P(\mathbf{M})$ corresponds to the distribution of initial input patterns
2: $F_t \leftarrow \{\}$, where $F_t$ denotes the set of flashcards
3: for $i \in \{1, \dots, N_f\}$ do
4:     $\mathbf{F}_{ri} = f_t^r(f_t^{r-1}(\cdots f_t^1(\mathbf{M}_i, \boldsymbol{\theta}_t) \cdots))$
5:     $F_t = F_t \cup \{\mathbf{F}_{ri}\}$
6: end for
7: return $F_t$
```

Algorithm 1 provides the steps to construct flashcards using a trained autoencoder model. For flashcards construction, the following hyperparameters are useful - (i) number of recursive iterations (r), (ii) number of flashcards to construct (N_f), (iii) choice of initial input distribution ($P(\mathbf{M})$). Figure 4 portrays empirical analysis of selecting the optimal

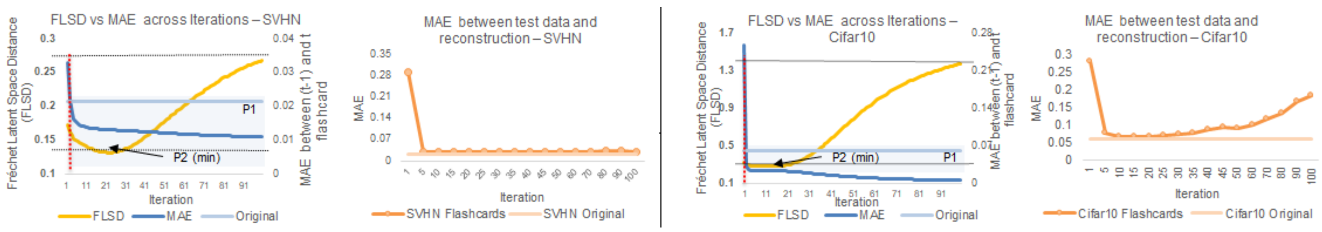


Figure 4. Analysis on iterations r for flashcards construction, demonstrated using SVHN and Cifar10. Columns 1 and 3 portray metrics corresponding to change on passing multiple random maze inputs in iterative fashion through an autoencoder. Orange curve depicts the trend of Fréchet latent space distance (FLSD) between intermediate input at (t) iteration and original data samples, dark blue curve shows constant decrease in reconstruction error between (t-1) and (t), across different iterations for each dataset. Light blue line serves as reference MAE on original data. Property P1 is satisfied when the blue curve goes below ϵ_1 (marked by dotted red line) and P2 is satisfied at the first minima of FLSD, which occurs around $10 \leq r \leq 20$, irrespective of dataset. Columns 2 and 4 compare autoencoder error (MAE) between test data and its reconstruction, when trained on flashcards constructed from different iterations. This again confirms the acceptable range of r and that flashcards construction is *not critically sensitive* to the choice of r within the range.

iteration, considering two datasets - SVHN and Cifar10. Figure 4 (columns 1,3) shows Frechet Latent Space Distance (FLSD is inspired from FID [8] and calculated as the Frechet distance between latent space activations of flashcards versus original samples (smaller the better), from the eyes of the trained autoencoder) (Orange) and MAE (Blue), as a function of r. Initially, increase in r leads to decrease in FLSD, up to certain iterations, this indicates that flashcards distribution is getting closer to learned distribution. As r increases further, though MAE keeps going down, FLSD increases, because propagation of reconstruction error in autoencoder causes drift in the features. Based on the empirical analysis $10 \leq r \leq 20$ is a sensible choice that worked well across all of our experiments. Also, reconstruction error (Figure 4, columns 2,4) is observed to be low in this iteration range. It further indicates that the flashcards are not critically sensitive to r. In all experiments r is set to 10.

Finding the optimal $P(\mathbf{M})$ is a research question in itself. For our experiments, the choice of $P(\mathbf{M})$ is maze-like random image patterns since there is resemblance to edges and shapes. Gaussian random noise was also considered but there was lack of diversity among the constructed flashcards. Using an additional dataset (such as next task images) did not help at all. A comparative study among the different choices is provided at the end of Section 4.1.

Generally, performance of a task is dependent on the number of flashcards constructed with sufficient diversity, and having more flashcards indicate better performance, similar to the trend with coreset exemplars. Note from [17], that memory requirement for experiments is calculated based on network parameters and is independent of generated samples.

2.2. Flashcards for Capturing Representations

We first verify Hypothesis 1 by demonstrating the efficacy of flashcards to capture representations for single-task.
Alternative to Original Dataset: Consider three different datasets: MNIST, Fashion MNIST, and Cifar10. Let 3 different autoencoders (Network1), all with same architecture, be trained for each of these 3 datasets separately using Eqn. (1) as loss function. Flashcards (F_t) constructed from each of these (Network1) autoencoders, for the same set of random input maze pattern images (D_m) (few samples shown in Figure 3) are used respectively to train new autoencoder (Network2) models from scratch (with the same architecture). Results in Table 1 show reconstructions are very close to the autoencoders trained on originals.

Flashcards also work for large-scale resolution, as seen from Figure 5. An autoencoder is trained on ImageNet 256x256px resolution (1M images) and has a reconstruction error of 0.10. A separate autoencoder is trained using flashcards only of varying percentages. The error when the network is randomly initialized without any data is high (0.42), but with increase in number of flashcards, it drops between 0.17 to 0.13, matching closely with the original benchmark.

Results from Table 1 and Figure 5 confirm the hypothesis that flashcards indeed capture network parameters as a function of D_m, therefore be used as pseudo-samples / training data for both small and large-scale resolutions (as network parameters involved in flashcard construction are initially learned by training with the original dataset D_t).
Data-Free Knowledge Distillation: Consider training an autoencoder teacher network AE1 with data D_{T_1}. Once

Dataset	Network 1's MAE	Network 2's MAE	Alpha Original	Alpha Flashcards	Alpha Untrained
MNIST	0.0184	0.0491	1.9185	2.0531	1.4343
Fashion	0.0259	0.0440	2.0044	2.1158	1.4343
Cifar10	0.0564	0.0686	2.0266	2.1531	1.4343

Table 1. Reconstructions from two separate autoencoders, one trained on original data (Network1), and the other trained on flashcards (Network2).The metrics indicate that flashcards are sufficient alternatives towards learning the original data. Weighted Alpha [16] is based on HT-SR Theory. Alpha values are closer for the trained networks which indicates similarity between the two network weights. Reported values are averaged over 5 runs.

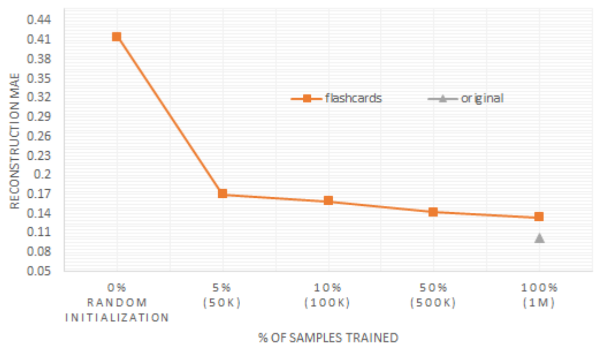


Figure 5. Reconstruction error of two networks, one trained on original ImageNet 256x256 resolution 1M images (grey), and the second trained only on flashcards 256x256 resolution (orange). X-axis denotes the % of samples added, and as more flashcards are introduced, error of the second network approaches closer to first.

Train Arch. Type	Params	Test Original	Test AE2
AE1 - Variant I	$94,243$	0.0640	–
AE2 - Smaller arch.	$24,083$	0.0787	0.0963
AE2 - Larger arch.	$372,803$	0.0512	0.0570
AE1 - Variant II	$298,947$	0.0437	–
AE2 - Smaller arch.	$24,083$	0.0358	0.0389
AE2 - Larger arch.	$372,803$	0.0512	0.0570

Table 2. Building autoencoder AE2 (smaller and larger arch.) using flashcards obtained from AE1 (Variant I and II, each with different parameters) trained on Cifar10 dataset. Column Test Original shows test MAE when these architectures are trained using original Cifar 10 dataset. Column Test AE2 refers to test MAE on Cifar10 when flashcards obtained from AE1 are used to train AE2 for reconstruction. Better performance on both AE2-Smaller and AE2-Larger architectures showcase transfer using flashcards.

trained, D_{T_1} may become unavailable due to confidentiality/privacy, storage requirements, etc. Regular Knowledge Distillation [7] is no longer possible because of the absence of original data while distilling to the student network. If in the future, there is a newer and better architecture AE2, migration is possible by training on flashcards constructed from AE1 to AE2 (Data-Free Knowledge Distillation). We show feasibility by distilling knowledge from teacher AE1 to student AE2, using two different architecture variants. Under each variant, the student AE2 is selected as one smaller and one larger modified architecture to the teacher. From Table 2 we observe, (i) distillation and transfer is still achievable without original data, (ii) improvement in performance when selecting larger architectures indicate that flashcards enable migration to better models in future.

2.3. Flashcards for Replay in Continual Learning

Consider a sequence of T tasks $\{T_1, \ldots, T_t, \ldots, T_T\}$. In the Continual Learning (CL) scenario, the network for task T_{t+1} is required to be trained on top of previous learned tasks $T_1, \ldots, T_t$. In other words, $\boldsymbol{\theta}_{t+1}$ is adapted from the

Algorithm 2 Flashcard for Replay in Continual Learning

```
Require: $f_t(\cdot, \boldsymbol{\theta}_t)$: Autoencoder model trained till task $T_t$
Require: $N_f$: Number of flashcards
Require: $r$: Number of iterations through $f_t(\cdot, \boldsymbol{\theta}_t)$
Require: $D_{t+1}$: Data for task $T_{t+1}$
Require: $\lambda$: Scaling parameter, by default set to 1 (in all Exp.)
1: $F_t \leftarrow$ Algorithm 1 $(f_t(\cdot, \boldsymbol{\theta}_t), N_f, r)$, where $F_t$ is the set of flashcards for task $T_t$
2: Train the autoencoder model $f_{t+1}(., \boldsymbol{\theta}_{t+1})$ on flashcards $F_t$ and data $D_{t+1}$
3: Initialize $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t$
4: Optimize $\boldsymbol{\theta}_{t+1}$ using given loss
   $$\boldsymbol{\theta}^*_{t+1} \leftarrow \arg\min_{\boldsymbol{\theta}_{t+1}} \Big( \frac{1}{|D_{t+1}|} \sum_{n=1}^{|D_{t+1}|} |D^{(n)}_{t+1} - f_{t+1}(D^{(n)}_{t+1}, \boldsymbol{\theta}_{t+1})| + \frac{\lambda}{N_f} \sum_{n=1}^{N_f} |F^{(n)}_t - f_{t+1}(F^{(n)}_t, \boldsymbol{\theta}_{t+1})| \Big)$$
   where $D^{(n)}_{t+1}$ and $F^{(n)}_t$ are the $n^{th}$ data sample and flashcard sample, respectively
5: return $f_{t+1}(., \boldsymbol{\theta}^*_{t+1})$
```

previously trained network parameters $\boldsymbol{\theta}_t$. Training for task T_{t+1} may result in the network forgetting the representations learned till the previous task T_t. Unlike other CL based approaches which aims to preserve $\boldsymbol{\theta}_t$ through regularization, data replay (either episodic or using external generative networks), architectural strategies, or their combinations, we use the flashcards constructed on $\boldsymbol{\theta}_t$ along with data for task T_{t+1}, while training for task T_{t+1}. Flashcards are required to be constructed only at the end of task T_t *irrespective* of the number of preceding tasks, so that knowledge representations for tasks $\{T_1, \ldots, T_t\}$ can be captured and trained with the next task T_{t+1}. Thus, the proposed method avoids storing of flashcards for each successive task, thereby significantly reducing the memory overhead, while ensuring robust performance (experiments in Section 4). Algorithm 2 provides the steps for using flashcards as a replay strategy for CL. Flashcards are constructed on-the-fly before training a new task (irrespective of number of previous tasks).

3. Related Works

In general, continual learning algorithms are based on architectural strategies, regularization, memory replay, and their combinations [18]. By the intrinsic nature of flashcards (as discussed in Sections 2.2 and 2.3), they implicitly exhibit the characteristics of both regularization (refer Algorithm 2 point 4) and replay, and will therefore be compared with the respective strategies. Regularization strategies such as [10, 24] learn new tasks while imposing constraints on the network parameters to avoid deviating too

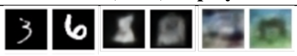
Trait	Flashcards for Replay	Episodic Memory Replay	Generative (Data) Replay
Examples			
Single-Task Scenario			
Alternative to original data?	Yes	No, subset of original	Yes
Distinction	Postprocess knowledge capture from trained network	Stores a subset of the original data in memory	Network is trained to generate samples per task
Visually similar to dataset?	No	Yes	Depends on data complexity
Performance on task	Close to original network	Very close to original network	Close to original network
Continual Learning Scenario			
Store sample across tasks?	No	Yes, need to store per task	Depends on nw. complexity
Nw. used to obtain alt. data	Simple autoencoder	N/A	VAE / GAN etc.
Store between tasks?	No, just in time creation	Yes, either memory or disk	Depends on support for CL
Independent of no. of tasks?	Yes, need current snapshot only	No, need to store per task	Depends on support for CL
Scaling up tasks (complexity)	Results shows feasibility	Straightforward - linear storage	Challenging with more tasks

Table 3. Comparison between different continual learning replay strategies.

much from those learned from the previous tasks. [13] uses current task samples to regularize the soft labels of past tasks, thus avoiding explicit data storage. However, it is important to note that these regularization methods perform well in homogeneous (sharing similar characteristics) task environments, where it is possible to find mutually suboptimal points in the solution space. However, such methods suffer when subsequent tasks are from different domains / heterogeneous datasets [13].

In a rehearsal mechanism, subset of samples from previous tasks, referred to as coreset samples / exemplars, serve as memory replay [3, 4, 14]. However, they suffer from excessive storage requirements that grows with the number of tasks. On the other hand, generative replay approaches [1, 12, 22, 20] face challenges in scalability to complex datasets with many tasks, since they involve heavy training and computation associated with an auxiliary network to generate visually meaningful images for each task.

Approaches discussed above are either suitable only for homogeneous data, or are memory intensive as they involve *preserving* samples, or computationally intensive in "generating" samples, or require task identifiers. The proposed flashcards are constructed on the fly, requires only simple autoencoder, independent of the number of tasks, and helpful *both* as regularization and replay mechanism. Flashcards can be used as *pseudo samples* constructed with low computational and memory expense. Furthermore, flashcards can scale across tasks without considerable drop in performance. Table 3 lists overall comparison on various traits between flashcards, episodic, and generative replay.

4. Experiments in Continual Learning

We provide experimental validation of using flashcards in CL scenario for different applications such as: (i) continual reconstruction (Section 4.1), (ii) continual denoising (Section 4.2), and new task incremental classification (Section 4.3). For all experiments, we compare with upperbound (Joint Training, JT), lowerbound (Sequential Fine-tuning, SFT), and standard methods for regularization, generative and episodic replay as baseline. To the best of our knowledge, we are the first to report continual reconstruction and denoising. Our results indicate that using flashcards indeed helps under heterogeneous conditions spanning different domains, and where the task identifier is not available.

Datasets: For the continual reconstruction and denoising, 5 heterogeneous public datasets, referred as Sequence5 was employed. The datasets used in Sequence5 are in the order - MNIST, Fashion MNIST, Cifar10, SVHN, and Omniglot. Each dataset was considered as a task, with class labels omitted. MNIST, Fashion MNIST and Omniglot were resized to 32x32x3 (bilinear rescale and channel copied twice) to maintain the same scale as the other two datasets. Continual reconstruction is also compared on UC Merced Land Use [23], a real-world large-scale remote sensing dataset with 256x256 resolution and 21 classes. For task incremental classification, Sequence3 comprising of the order - Cifar10, MNIST and Fashion MNIST was employed.

Architecture and Training Details: For reconstruction and denoising, flashcards are constructed using an autoencoder architecture with 4 layers of down/upsampling and 64 convolutional filters per layer. The bottleneck dimension 256 offers 12x reduction from image space, serving to demonstrate the effectiveness of forgetting. The architecture is a simplified variant of the VGG , adapted with fewer layers and filters for proper reconstruction. All hidden layers employ *tanh* activation.

For task incremental classification, the architecture of classifier is based on specifications from [22]. For the architecture of autoencoder, the encoder weights are pre-initialized with classifier encoder weights, decoder is attached after latent space, and the autoencoder is trained by passing current task samples along with flashcards upto current task. For the next task, classifier uses new task samples along with constructed flashcards of previous task. Soft labels for flashcards are obtained by passing them through the classifier, before the start of next task.

Training details: For reconstruction and denoising, the

Method	Type	N/w	Mem	Recon MAE ↓	Recon BWT ↑	Denoise MAE ↓	Denoise BWT ↑
JT	-	1.5	798.7	0.0348 ± 0.000	-	0.0552 ± 0.000	-
SFT	-	1.5	-	0.5518 ± 0.001	-0.6385 ± 0.002	0.5477 ± 0.002	-0.6275 ± 0.002
Coreset 500	ER	1.5	1.5	0.0580 ± 0.005	-0.0246 ± 0.006	0.0748 ± 0.004	-0.0219 ± 0.005
Coreset 5000	ER	1.5	15.3	0.0495 ± 0.002	-0.0185 ± 0.002	0.0595 ± 0.002	-0.0172 ± 0.002
LwF	Reg+R	1.5	-	0.5495 ± 0.002	-0.5241 ± 0.002	0.4370 ± 0.002	-0.3338 ± 0.002
VAE 5000 + AE	GR	2.9	-	0.0751 ± 0.004	-0.0575 ± 0.006	0.0711 ± 0.005	-0.0438 ± 0.005
CL VAE 5000	GR	1.4	-	0.3524 ± 0.041	-0.1658 ± 0.038	0.5460 ± 0.042	-0.2839 ± 0.033
Flashcards 5000	FR	**1.5**	-	$\mathbf{0.0536} \pm \mathbf{0.002}$	$\mathbf{-0.0277} \pm \mathbf{0.002}$	$\mathbf{0.0588} \pm \mathbf{0.002}$	$\mathbf{-0.0262} \pm \mathbf{0.002}$

Table 4. Continual learning for Sequence5 reconstruction and denoising. We provide a comparison with different methods in terms of error observed MAE and BWT on test set. Additionally, we compare the Network Capacity (N/w (MB)) and External Memory (Mem (MB)) required by each method. Our method performs better than generative method based replay, and is on-par with the error on using episodic memory based coreset replay. Further, our memory requirement is very less, and an external memory buffer to store past tasks is not required. Lower Avg MAE (↓) and higher BWT (↑) are better. Reg=Regularization, Reg+R=Regularization and Replay, ER=Episodic Replay, GR=Generative Replay, FR=Flashcard Replay. Reported scores are averaged over 3 independent runs.

network is trained for 100 epochs per task, and optimized using Adam and learning rate of 0.001. Each minibatch update is based on equal number of flashcards and the current task samples. 10% of the training data is allocated for validation. Early stopping is employed if there is no improvement for over 20 epochs. For task incremental classification, mini-batch size of 256 and Adam optimizer with learning rate=0.0001 is used. Both classifier and autoencoder models are trained for 5K iterations. 5K flashcards with $\lambda = 1$ (refer Algorithm 2) was employed for replay.

4.1. Heterogeneous Continual Reconstruction

Continual reconstruction is an experiment to study the ability of different continual learning methods to effectively remember the characteristics of data learned thus far to reconstruct samples of both past and present. A single network is trained on multiple heterogeneous datasets incrementally, and during inference, it is expected to remember the task of reconstruction without an explicit task identifier. We measure individual task MAE (averaged) and BWT, customized for reconstruction. Results for reconstruction using Sequence5 (details in Section 4) are available under columns 5 and 6 in Table 4. Since continual reconstruction for homogeneous cases is trivial and does not require extra effort (as observed from Table 5), we focus on the more challenging heterogeneous scenario where forgetting is inevitable. Comparison with baseline methods is summarized in the captions of Table 4. We compare flashcards against different continual learning based regularization and replay techniques and observe that it outperforms most baselines user fewer network capacity. It performs on-par with episodic replay methods without occupying external memory for past samples.

Continual reconstruction results over 21 tasks on UC Merced land use remote sensing dataset (256px) is illustrated in Figure 6. Comparison is made per task (on x-axis) against SFT, Coreset (max 100 samples in buffer), and VAE+AE, with JT as reference line. All autoencoders require only 2.9MB (for 256x256 dim) in terms of network parameters. Coreset requires extra 75MB for storing 100 images. VAE occupies an extra 4.7MB. Both Flashcards and VAE+AE generate 100 samples towards next task and have similar performance. VAE+AE is slightly better than Flashcards at the expense of more storage. Other methods experience spike in transition to next task, whereas the performance is stable when using flashcards.

Method	Cifar10			Caltech101		
	T_1	T_5	T_{10}	T_1	T_{50}	T_{101}
JT	—	—	0.0596	—	—	0.0831
SFT	0.0872	0.0748	0.0719	0.1088	0.1002	0.0906

Table 5. Continual reconstruction for homogeneous tasks (each task T_k specifies upto k classes from the same dataset have been observed) shows minimal / no forgetting, examples showcasing reconstruction error for Cifar10 and Caltech101. On the contrary there is strong indication that knowledge from past improves learning newer tasks (forward transfer).

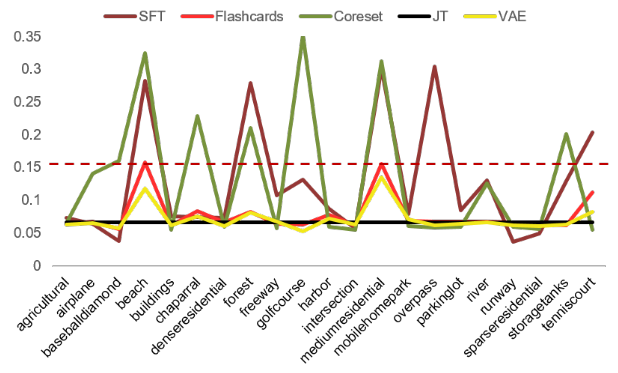


Figure 6. Continual reconstruction on UC Merced land use 256px dataset. Line plot shows reconstruction error averaged from first till current task, as new tasks are introduced. SFT and Coreset oscillate occasionally as new tasks are encountered. E.g. beach (blue texture) after agriculture and baseball ground (green), forest after series of buildings (grey), golfcourse after freeway, overpass after park etc. Despite minimal capacity among other methods, Flashcards has relatively stable transition (dotted horizontal line shows peak error) and low error for all tasks.

Method	Type	With Task Identifier			Without Task Identifier		
		T1	T1→T2	T1→T2→T3	T1	T1→T2	T1→T2→T3
SFT	-	61.55 ± 0.8	60.20 ± 0.1	43.55 ± 0.4	61.55 ± 0.8	49.41 ± 0.1	30.21 ± 0.2
Coreset 100%	ER	61.55 ± 0.8	79.26 ± 0.6	83.13 ± 0.8	61.55 ± 0.8	81.15 ± 1.1	84.38 ± 1.0
EWC	Reg	61.55 ± 0.8	66.50 ± 2.9	72.75 ± 1.9	61.55 ± 0.8	50.27 ± 3.2	28.18 ± 2.2
SI	Reg	61.55 ± 0.8	63.42 ± 1.0	67.22 ± 1.1	61.55 ± 0.8	50.60 ± 1.0	30.08 ± 1.3
LwF	Reg+R	61.55 ± 0.8	72.76 ± 2.9	**74.90 ± 2.9**	61.55 ± 0.8	31.77 ± 3.4	31.52 ± 3.2
CL VAE 5000	GR	61.55 ± 0.8	**73.12 ± 2.5**	66.86 ± 3.2	61.55 ± 0.8	67.97 ± 2.7	50.61 ± 3.1
BI-R	GR	61.55 ± 0.8	52.09 ± 2.4	52.70 ± 2.6	61.55 ± 0.8	22.65 ± 2.7	26.47 ± 2.4
Flashcards 5000	FR	61.55 ± 0.8	70.75 ± 1.0	72.13 ± 0.9	61.55 ± 0.8	**67.64 ± 1.0**	**63.71 ± 0.8**

Table 6. Comparison of different methods for task incremental learning using Sequence3 tasks (Cifar10-MNIST-Fashion MNIST). Reported values are accuracy computed at the end of task T. When task identifier is provided, flashcards performance matches other baseline methods. However, in the absence of task identifier, baselines fail while flashcards helps to retain accuracy. Flashcards is robust to degradation across different domain tasks and outperforms other methods. Reg=Regularization, Reg+R=Regularization and Replay, ER=Episodic Replay, GR=Generative Replay, FR=Flashcard Replay. Reported scores are averaged over 3 independent runs.

Maze Pattern	Gaussian Noise	Next Task data
$\mathbf{0.0536 \pm 0.002}$	0.0945 ± 0.003	0.3713 ± 0.275

Table 7. Maze patterns provide best reconstruction MAE among different initializations used for construction of flashcards. Experiments run on Sequence5 continual reconstruction.

Flashcards initialization: A study on the choice of initialization is presented in Table 7 and Figure 7 using Sequence5 for continual reconstruction. Among the three choices - Maze, Gaussian Noise and next task data, only Maze based flashcards had diversity in terms of shape and texture, and performed best due to its ability to capture activations at edges. Using Gaussian noise as initialization resulted in repetition of same patterns. Flashcards from new task initialization being dependent on the random subset images' coverage of old tasks' activations led to inconsistent results.

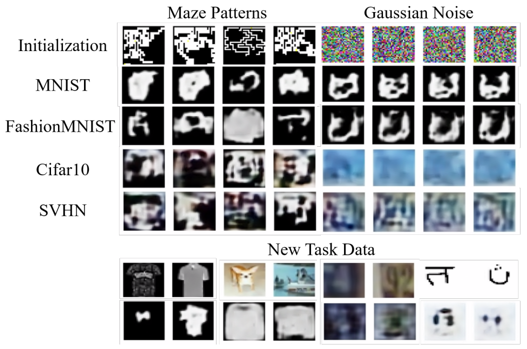


Figure 7. Visual comparison of different initializations used for construction of flashcards - Maze, Gaussian Noise and next task.

4.2. Heterogeneous Continual Denoising

A more challenging extension to reconstruction is denoising in continual learning scenario. Here the objective of the network is to learn, remember and denoise the samples simultaneously. For our experiments, we impose noise sampled from a standard normal distribution factored by a small parameter to the Sequence5 datasets. Results reported for Denoise MAE and BWT under columns 7 and 8 in Table 4 are obtained by adding noise factor of 0.1 to original images. Our approach (Flashcards) performs better than baseline approaches, with lower error compared to regularization and generative replay methods, and even outperforms episodic replay method - Coreset 5000 in terms of MAE.

4.3. Task (Agnostic) Incremental Classification

Task Incremental Learning (Task-IL) is a continual learning setting where new tasks, each comprising of multiple classes ($\geq$ 2) are learned in incremental fashion. The network encoder is shared whereas there exists task-exclusive multihead output. Generally, classification is performed using the specific multihead, with the help of task identifier. We compare performance of flashcards for both cases, with and without task identifier. We follow the same setup as provided in [22] (details in Section 4). Results for Sequence3 (details in Section 4) from Table 6, show that baseline methods designed to make decisions only on seeing the task identifier during inference perform poorly in its absence. Our approach, on the other hand, is designed as task agnostic, relying only on the most recent checkpoint. It is competitive in the presence of task identifier, and in the absence, it outperforms baseline methods by a big margin.

5. Conclusion

We introduced *flashcards* that can capture knowledge representations of a trained autoencoder through recursive passing of random image patterns, and showed that it can be used as alternative to original data. We further demonstrated its efficacy as a task agnostic replay mechanism for various continual learning scenarios, such as reconstruction, denoising, and task incremental learning, with heterogeneous datasets. Flashcard replay outperforms generative replay and regularization methods, without additional memory and training, and also performs on par with episodic replay, without storing exemplars. The intrinsic nature of flashcards allows for data abstraction that can be exploited for potential data privacy applications. Generalization of flashcards to other domains will foster further research.

References

[1] R. Aljundi, L. Caccia, E. Belilovsky, M. Caccia, M. Lin, and L. Charlin. Online continual learning with maximally interfered retrieval. *Proceedings of the Advances in Neural Information Processing Systems 32*, 2019.

[2] Liu Bing. Learning on the job: Online lifelong and continual learning. In *Proceedings of the AAAI*, 2020.

[3] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 233–248, 2018.

[4] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with A-GEM. In *ICLR*, 2019.

[5] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[6] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *ICLR*, 2019.

[7] Jianping Gou, Baosheng Yu, Stephen John Maybank, and Dacheng Tao. Knowledge distillation: A survey, 2020.

[8] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637, 2017.

[9] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[10] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, and Grabska-Barwinska. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114, 2017.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012.

[12] H. Li, W. dong, and B. G. Hu. Incremental concept learning via online generative memory recall. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[13] Z. Li and D. Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[14] David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems 30*, 2017.

[15] M. Längkvist, L. Karlsson, and A. Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 2014.

[16] Charles H Martin and Michael W Mahoney. Heavy-tailed universality predicts trends in test accuracies for very large pre-trained deep neural networks. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, 2020.

[17] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi. Learning to remember: A synaptic plasticity driven framework for continual learning. In *Proceedings of the CVPR*, 2019.

[18] German I. Parisi, Ronald Kemker, Jose L.Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.

[19] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

[20] M. Rostami, S. Kolouri, J. McClelland, and P. Pilly. Generative continual concept learning. *Proceedings of the 34th AAAI*, 2020.

[21] Sebastian Thrun. Is learning the n-th thing any easier than learning the first? In *Advances in neural information processing systems*, pages 640–646, 1996.

[22] G. M. van de Ven, H. T. Siegelmann, and A. S. Tolias. Brain-inspired replay for continual learning with artificial neural networks. *Nature Communications*, 11, 2020.

[23] Yi Yang and Shawn Newsam. Bag-of-visual-words and spatial extensions for land-use classification. In *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*, pages 270–279, 2010.

[24] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th ICML - Volume 70*, 2017.

[25] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2021.