Scalable Recollections for Continual Lifelong Learning

Matthew Riemer, Tim Klinger, Djallel Bouneffouf, Michele Franceschini
IBM Research
T.J. Watson Research Center, Yorktown Heights, NY
{mdriemer, tklinger, djallel.bouneffouf, franceschini}@us.ibm.com

Abstract

Given the recent success of Deep Learning applied to a variety of single tasks, it is natural to consider more human-realistic settings. Perhaps the most difficult of these settings is that of continual lifelong learning, where the model must learn online over a continuous stream of non-stationary data. A successful continual lifelong learning system must have three key capabilities: it must *learn and adapt* over time, it must *not forget* what it has learned, and it must be *efficient* in both training time and memory. Recent techniques have focused their efforts primarily on the first two capabilities while questions of efficiency remain largely unexplored. In this paper, we consider the problem of efficient and effective storage of experiences over very large time-frames. In particular we consider the case where typical experiences are $O(n)$ bits and memories are limited to $O(k)$ bits for $k << n$. We present a novel scalable architecture and training algorithm in this challenging domain and provide an extensive evaluation of its performance. Our results show that we can achieve considerable gains on top of state-of-the-art methods such as GEM. [1]

Introduction

A long-held dream of the AI community is to build a machine capable of operating autonomously for long periods or even indefinitely. Such a machine must necessarily learn and adapt to a changing environment and, crucially, manage memories of what it has learned for the future tasks it will encounter. A spectrum of learning scenarios are available depending on problem requirements. In lifelong learning (Thrun 1996) the machine is presented a sequence of tasks and must use knowledge learned from the previous tasks to perform better on the next. In the resource-constrained lifelong learning setting the machine is constrained to a small buffer of previous experiences. Some approaches to lifelong learning assume that a task is a set of examples chosen from the same distribution (Rusu et al. 2016; Fernando et al. 2017; Shin et al. 2017; Ramapuram, Gregorova, and Kalousis 2017; Al-Shedivat et al. 2017; Lee et al. 2018). If instead the machine is given a sequence of examples without any batching, then this is called continual learning. In this paper we focus on this more challenging continual learning scenario.

[1] See an extended version of this paper including the appendix at https://arxiv.org/pdf/1711.06761.pdf.

Continual learning (Thrun 1994; Ring 1994; Thrun 1996) has three main requirements: (1) continually adapt in a non-stationary environment, (2) retain memories which are useful, (3) manage compute and memory resources over a long period of time. Most neural network research has focused on methods to improve (1) and (2). In this paper we consider (3) as well and further investigate the role of efficient experience storage in avoiding the catastrophic forgetting (McCloskey and Cohen 1989) problem that makes (2) so challenging.

Experience memory has been influential in many recent approaches. For example, experience replay (Lin 1992) was integral in helping to stabilize the training of Deep Q Learning on Atari games (Mnih et al. 2015). Episodic storage mechanisms (Schaul et al. 2015; Blundell et al. 2016; Pritzel et al. 2017; Rebuffi, Kolesnikov, and Lampert 2017; Lopez-Paz and Ranzato 2017) were also some of the earliest solutions to the catastrophic forgetting problem in the supervised learning setting (Murre 1992; Robins 1995). Unlike approaches which simply focus on remembering representations of old tasks (Li and Hoiem 2016; Riemer, Khabiri, and Goodwin 2016; Kirkpatrick et al. 2017), episodic storage techniques achieve superior performance because of their ability to continually improve on old tasks over time as useful information is learned later (Lopez-Paz and Ranzato 2017).

All of these techniques try to use stored experiences to stabilize learning. However, they do not consider agents which must operate independently in the world for a long time. In this scenario, assuming the kind of high-dimensional data which make up human experiences, the efficient storage of experiences becomes an important factor. Storing full experiences in memory, as these methods do, causes storage costs to scale linearly with the number of experiences. To truly learn over a massive number of experiences in a non-stationary environment, the incremental cost of adding experiences to memory must be sub-linear in the number experiences.

In this paper we propose a scalable experience memory module which learns to adapt to a non-stationary [2] environment and improve itself over time. The memory module is implemented using a variational autoencoder which learns to compress high-dimensional experiences to a compact latent code for storage. This code can then be used to reconstruct

[2] We assume the environment is non-stationary, but not adversarial. So past experiences can be helpful for learning future tasks.

realistic recollections for both experience replay training and improvement of the memory module itself. We demonstrate empirically that the module achieves sub-linear scaling with the number of experiences and provides a useful basis for a realistic continual learning system. Our experiments show superior performance over state-of-the-art approaches for lifelong learning with a very small incremental storage footprint.

Related Work

Storing Parameters Instead of Experiences. Our method is complementary to recent work leveraging episodic storage to stabilize learning (Mnih et al. 2015; Blundell et al. 2016; Pritzel et al. 2017; Rebuffi, Kolesnikov, and Lampert 2017; Lopez-Paz and Ranzato 2017). Some recently proposed methods for lifelong learning don't store experiences at all, instead recording the parameters of a network model for each task (Rusu et al. 2016; Kirkpatrick et al. 2017; Fernando et al. 2017). This yields linear (or sometimes worse) storage cost scaling with respect to the number of tasks. For our experiments, and in most settings of long-term interest for continual learning, the storage cost of these extra model parameters per task significantly exceeds the per task size of a corresponding experience buffer.

Generative Models to Support Lifelong Learning. Pseudorehearsals (Robins 1995) is a related approach for preventing catastrophic forgetting that unlike our recollection module does not require explicit storage of codes. Instead it learns a generative experience model alongside the main model. For simple learning problems, very crude approximations of the real data such as randomly generated data from an appropriate distribution can be sufficient. However, for complex problems like those found in NLP and computer vision with highly structured high dimensional inputs, more refined approximations are needed to stimulate the network with relevant old representations. To the best of our knowledge, we are the first to consider variational autoencoders (Kingma and Welling 2014) as a method of creating pseudo-experiences to support supervised learning. Some recent work (Ramapuram, Gregorova, and Kalousis 2017) considers the problem of generative lifelong learning for a variational autoencoder, introducing a modified training objective. This is potentially complementary to our contributions in this paper.

Biological Inspiration and Comparisons. Interestingly, the idea of scalable experience storage has a biologically inspired motivation relating back to the pioneering work of McClelland, McNaughton, and O'Reilly (1995), who hypothesized complementary dynamics for the hippocampus and neocortex. In this theory, updated in (Kumaran, Hassabis, and McClelland 2016), the hippocampus is responsible for fast learning, providing a very plastic representation for retaining short term memories. Because the neocortex, responsible for reasoning, would otherwise suffer as a result of catastrophic forgetting, the hippocampus also plays a key role in generating approximate *recollections* (experience memories) to interleave with incoming experiences, stabilizing the learning of the neocortex. Our approach follows *hippocampal memory index theory* (Teyler and DiScenna 1986; Teyler and Rudy 2007), approximating this role of the hippocampus, with a modern deep neural network model. As this theory suggests, lifelong systems need both a mechanism of *pattern completion* (providing a partial experience as a query for a full stored experience) and *pattern separation* (maintaining separate indexable storage for each experience). As in the theory, we do not literally store previous experiences, but rather compact *indexes* which can be used to retrieve the experience from an *association cortex*, modeled as an auto-encoder.

Storing Few Experiences. Recent work on continual lifelong learning in deep neural networks (Rebuffi, Kolesnikov, and Lampert 2017; Lopez-Paz and Ranzato 2017) has focused on the resource constrained lifelong learning problem and how to promote stable learning with a relatively small diversity of prior experiences stored in memory. In this work, we complete the picture by also considering the relationship to the *fidelity* of the prior experiences stored memory. We achieve this by considering an additional resource constraint on the number of bits of storage allowed for each experience.

The Scalable Recollection Module (SRM)

The core of our approach is an architecture which supports scalable storage and retrieval of experiences as shown in Figure 1. There are three primary components: an *encoder*, an *index buffer*, and a *decoder*. When a new experience is received (in the figure, an image of the numeral "6"), the encoder compresses it to a sequence of discrete latent codes (one hot vectors). These codes are concantenated and further compressed to a k bit binary code or "index" shown in decimal in the figure. This compressed code is then stored in the index buffer. This path is shown in blue. Experiences are retrieved from the index buffer by sampling a code from the index buffer and passing it through the decoder to create an approximate reconstruction of the original input. This path is shown in red in the figure.

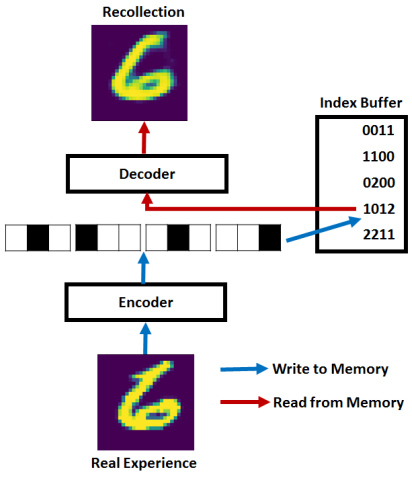

Figure 1: The scalable recollection module.

The recollection buffer is implemented using a discrete variational auto-encoder (Jang, Gu, and Poole 2017)(Maddison, Mnih, and Teh 2017). A discrete variational autoencoder

is a generative neural model with two components: an encoder and a decoder. The encoder is trained to take the input (say an image) and produce a discrete distribution over a set of latent categories which describe that input. To generate data, the discrete distribution is sampled. This can be done in a differentiable way using the so-called "reparameterization trick" which pushes the (non-differentiable) sampling operation to the front of the network. The decoder takes the sample and decodes it back into the original form of the input (in our example, an image). The VAE can then be used to encode experiences into a compact discrete code and later to generate experiences by sampling codes in the latent space and running them through the decoder.

Variational autoencoders have been used in the past to learn generative models of experience (called "pseudo-rehearsals" in the literature) (Robins 1995). Our model differs in two respects: first we are using a discrete VAE which can produce compact codes; and second we are maintaining a buffer of those codes. Typical applications for VAEs focus on a fixed distribution to be learned and require that the VAE reproduce samples from that distribution with high fidelity. For continual learning it is also important that the VAE adapt to the non-stationary data distribution. Without an index buffer, the VAE's parameters would quickly adapt to the current input and forget its past experiences. Additionally, as we will show later, the buffer leads to greater efficiency in generating samples that capture the variation of the distribution seen.

Improving Experience Replay Training

The recollection module can be used in many ways in a continual learning setting. In Algorithm 1 we show one approach which we will use later in our experiments.

In this setting the model must learn T tasks sequentially from dataset D. At every step it receives a triplet (x, t, y) representing the input, task label, and correct output. There are two models to be trained: the recollection module which consists of a memory index buffer M, an encoder ENC_ϕ and decoder DEC_ψ; and a predictive task model F_θ. The training proceeds in two phases: in the first phase we ensure that the recollection module itself is stabilized against forgetting; in the second phase we stabilize the predictive model.

For the recollection module, we achieve stabilization through a novel extension of experience replay (Lin 1992). When an incoming example is received, we first sample multiple batches of recollections from the index buffer and decode them into experiences using the current decoder. We then perform N steps of optimization on the encoder/decoder parameters ϕ and ψ by interleaving the current input example with a different batch of past recollections at each of the N optimization steps. For each optimization step, the error for each experience in a batch is computed by encoding that experience into a latent code using the encoder and then decoding back to an experience to compute the reconstruction error. On the first optimization step, the the reconstruction error is computed using the same decoder parameters that were used in the creation of that input experience in the batch. In subsequent steps, those parameters change as the recollection module is stabilized, learning parameters to successfully reconstruct both the old experiences in the buffer as well as the new experience. In this way the recollection module continues to *remember* the relevant past experiences in the buffer while integrating new experiences.

After the recollection module is trained with loss function ℓ_{REC} and learning rate β, the predictive model F_θ is trained on just one of the recollection sample sets (we arbitrarily chose the first) using loss function ℓ and learning rate α. Finally, the new sample is written to the index buffer. Perhaps surprisingly, this strategy of reconstructing experiences from codes and then performing experience replay training using them can be as effective for enabling continual learning as replaying real, uncompressed inputs in some cases.

In the resource constrained setting we study in this paper, there is an upper limit on the number of allowable episodic memories L which governs the memory update $M \leftarrow M \cup \{(z, y)\}$. Our approach is fairly robust to changes in the update rule. For experience replay, we maintain the buffer using reservoir sampling (Vitter 1985). In constrast, for the recently proposed Gradient Episodic Memory (GEM) algorithm which modulates gradients on incoming examples by solving a quadratic program with respect to past examples, we follow prior work and keep an equal number of the most recent examples for each task. As detailed in the appendix, integration of Scalable Recollections across algorithms is quite easy and the other differences between the experience replay and GEM algorithms with Scalable Recollections are contained to the different ways of utilizing episodic memory inherent to each lifelong learning algorithm. We note that the PROJECT function within GEM solves a quadratic program explained in (Lopez-Paz and Ranzato 2017).

Recollection Efficiency

In this section we argue that it is more efficient to employ an index buffer rather than sampling directly from the latent VAE code space. This results from the ability of the model to recreate the input distribution. A typical strategy is to randomly sample from old experiences in episodic storage and interleave them with new ones to stabilize learning. So to the extent that the recollections are representative of the full distribution of prior experiences, learning proceeds exactly as if samples were drawn from a stationary i.i.d. distribution.

However, when sampling from the code space of the VAE without a buffer, the sample distribution will be unlikely to match that of the experiences from training. If the number of examples is larger than the number of possible codes (the capacity of the VAE) then the VAE will be unable to differentiate truly different images and hence have poor reconstruction. This scenario must be avoided for performance reasons. On the other hand, if the VAE capacity is considerably larger than the number of training examples, then sampling it at random is highly unlikely to reproduce the input distribution.

To alleviate this problem, a buffer can be added whose distribution of codes will match the distribution of the input data codes, if it is sufficiently large. Intuitively, setting the buffer size to be less than the capacity of the VAE will ensure that sampling from the buffer is more efficient than sampling directly from the VAE code space. Our experiments empirically support this hypothesis (Question 6):

Algorithm 1 Experience Replay Training for Continual Learning with a Scalable Recollection Module

procedure TRAIN($D, F_\theta, ENC_\phi, DEC_\psi, \alpha, \beta$)
  $M \leftarrow \{\}$
  for $t = 1, ..., T$ **do**
    for x, y `in` D_t **do**
      Scalable Recollection Module Training:
      # create N recollection sample sets
      for `s = 1,...,N` **do**
        # sample latent codes and labels
        $z_s, y_s \leftarrow Sample(M)$
        # decode the latent codes into recollections
        $x_s \leftarrow DEC_\psi(z_s)$
        # save the current label
        $Y_s \leftarrow y_s \cup y$
        # save the current recollection
        $X_s \leftarrow x_s \cup x$
      end for
      # train the recollection module
      for $s = 1, ..., N$ **do**
        # compute recollection module gradients
        $u \leftarrow \nabla_{\phi,\psi} \ell_{REC}(DEC_\psi(ENC_\phi(X_s))), X_s)$
        # update the encoder parameters
        $\phi \leftarrow \phi - \beta u_\phi$
        # update the decoder parameters
        $\psi \leftarrow \psi - \beta u_\psi$
      end for
      Experience Replay Training:
      # compute main model gradients
      $g \leftarrow \nabla_\theta \ell(F_\theta(X_1, t), Y_1)$
      # update the main model parameters
      $\theta \leftarrow \theta - \alpha g$
      # encode the recollection sample set
      $z \leftarrow ENC_\phi(x)$
      # store it in the index buffer
      $M \leftarrow M \cup \{(z, y)\}$
    end for
  end for
  return $F_\theta, ENC_\phi, DEC_\psi, M$
end procedure

Hypothesis 1 *An index buffer is a more efficient parameterization of the input space seen by a variational autoencoder than its latent code if:*

$$\ell^c \geq L$$

ℓ is the autoencoder latent variable size (i.e. 32 for typical continuous latent variables), c is the number of latent variables in the autoencoders (i.e. the number of hidden units for continuous latent variables), and L is the index buffer size.

In the vast majority of settings relevant to the study of continual lifelong learning today it is safe to assume that this inequality will hold. For example, even if we store an index for every example in a dataset like MNIST or CIFAR, this inequality will hold unless a continuous latent variable autoencoder has 3 or fewer latent variables. To put it another way, the per sample compression would have to be over 400X for popular datasets like Omniglot and MNIST and well over 1500X for CIFAR. These levels of compression with good reconstruction are far beyond what is possible with any known tools today. Moreover, given, as an example, the blurry nature of CIFAR images, it is possible that this compression quality is completely unfeasible. In practice, do to natural redundancy in the space of samples, it is more likely that L will also only need to be significantly less than the number of examples seen. In our work we find that selecting subsets works well although a buffer may be even more efficient with online clustering strategies as in (Kaiser et al. 2017).

Evaluation

Datasets

Our experiments will primarily focus on three public datasets commonly used for deep lifelong and multi-task learning.

MNIST-Rotations: (Lopez-Paz and Ranzato 2017) A dataset with 20 tasks including 1,000 training examples for each task. The tasks are random rotations between 0 degrees and 180 degrees of the input space for each digit in MNIST.

Incremental CIFAR-100: (Lopez-Paz and Ranzato 2017) A continual learning split of the CIFAR-100 image classification dataset considering each of the 20 course grained labels to be a task with 2,500 examples each.

Omniglot: A character recognition dataset (Lake et al. 2011) in which we consider each of the 50 alphabets to be a task. This is an even more challenging setting than explored in prior work on continual lifelong learning, containing more tasks and fewer examples of each class.

Evaluation for Continual Lifelong Learning

In this section we evaluate the benefits of the Scalable Recollection Module in enabling continual lifelong learning.

Metric: As in prior work, we measure *retention* as our key metric. It is defined as the test set accuracy on all tasks after sequential training has been completed over each task.

Architecture: We model our experiments after (Lopez-Paz and Ranzato 2017) and use a Resnet-18 model as F_θ for CIFAR-100 and Omniglot as well as a two layer MLP with 200 hidden units for MNIST-Rotations. Across all of our experiments, our autoencoder models include three convolutional layers in the encoder and three deconvolutional layers in the decoder. Each convolutional layer has a kernel size of 5. As we vary the size of our categorical latent variable across experiments, we in turn model the number of filters in each convolutional layer to keep the number of hidden variables consistent at all intermediate layers of the network.

Module hyperparameters: In our experiments we used a binary cross entropy loss for both ℓ and ℓ_{REC}. In the appendix we outline a constrained optimization procedure to find the optimal discrete autoencoder latent code design for a given resource footprint constraint. We follow this procedure to derive architectures that can be directly compared to various episodic storage baselines in our experiments.

The key question we consider is the following:

Question 1 *Is the recollection module useful in improving retention for continual lifelong learning?*

To answer this question we compare the retention performance of a system equipped with the recollection module to one equipped with a buffer of real experiences. We consider

three datasets: MNIST-Rotations, CIFAR-100, and Omniglot. To account for the fact that real experiences take up more storage, we give both approaches the same storage budget. Specifically, we define two new quantities: *incremental storage* and *items*. The *incremental storage* is the effective size of the incremental storage used after the sunk cost of the initial model parameters. For clarity we express the effective *incremental storage* size in terms of the number of real examples of storage that would have the same footprint of resources used. The number of *items* by contrast refers to the number of items used in the episodic storage buffer whether they are real or approximate recollections. Obviously, by compressing items stored in the buffer we are able to store more *items* at the same effective *incremental storage* size.

In Table 1 we consider learning with a very small incremental resource footprint on MNIST-Rotations where we allow the effective storage of only one full experience or fewer per class. We find that storing real experiences can perform well in this regime, but the Scalable Recollection Module significantly outperforms them. For example, we achieve considerable improvements over results reported for the popular elastic weight consolidation (EwC) algorithm (Kirkpatrick et al. 2017) on this benchmark. We note that EwC stores some items for computing the Fisher information. We also note that the large incremental cost of the parameter and Fisher information storage for each task is equal to that of storing a real buffer of over 18 thousand examples.

In Table 2 we show results for Incremental CIFAR-100. This is a significant challenge for the Scalable Recollection Module since the CIFAR tiny image domain is known to be a particularly difficult setting for VAE performance (Kingma et al. 2016; Chen et al. 2017). We explore settings with a very small incremental resource footprint, including a couple of settings with even less incremental memory allowance than the number of classes. Real storage performs relatively well when the number of examples is greater than the number of classes, but otherwise suffers from a biased sampling towards a subset of classes. This can be seen by the decreased performance for small buffer sizes compared to using no buffer at all, learning online. Consistently we see that tuning the recollection module to approximate recollections with a reasonably sized index buffer results in improvements over real storage at the same incremental resource cost.

To further validate our findings, we tried the Omniglot dataset, attempting to learn continually in the difficult incremental 50 task setting. With an incremental resource constraint of 10 full examples, replay achieves 3.6% final retention accuracy (online learning produces 3.5% accuracy). In contrast, the recollection module achieves 5.0% accuracy. For an incremental resource footprint of 50 full examples, replay achieves 4.3% accuracy which is further improved to 4.8% accuracy by taking three gradient descent steps per new example. The recollection module again achieves better performance with 9.3% accuracy at one step per example and 13.0% accuracy at three steps per example.

Question 2 *How does use of Scalable Recollections influence the long term retention of knowledge?*

Method	Incremental Storage	Items	Retention
GEM Real Storage	100	100	62.5
	200	200	67.4
GEM Recollections	100	3000	**79.0**
	200	3000	**81.5**
Replay Real Storage	100	100	63.4
	200	200	71.3
Replay Recollections	100	3000	**75.6**
	200	3000	**81.1**
EwC	18288	1000	54.6
Online	0	0	51.9

Table 1: Retention results on MNIST-Rotations for low effective buffer sizes with an incremental storage resource constraint.

Model	Incremental Storage	Items	Retention
Online	0	0	33.3
LwF (Li and Hoiem 2016)	0	0	34.5
Replay Real Storage	10	10	29.4
	50	50	33.4
	200	200	43.0
Replay Recollections	10	5000	**39.7**
	50	5000	**47.9**
	200	5000	**51.6**
GEM Real Storage	20	20	23.4
	60	60	40.6
	200	200	48.7
GEM Recollections	20	5000	**52.4**
	60	5000	**56.0**
	200	5000	**59.0**

Table 2: Incremental CIFAR-100 results for low effective incremental buffer sizes. GEM requires sizes that are multiples of $T = 20$.

The value of using recollections becomes even more apparent for long term retention of skills. We demonstrate this in Figure 2 by first training models on Incremental CIFAR-100 and then training them for 1 million training examples on CIFAR-10. The number of training examples seen from CIFAR-100 is only 5% of the examples seen from CIFAR-10. Not only does the recollection module allow experience replay to generalize more effectively than real storage during initial learning, it also retains the knowledge much more gracefully over time. We provide a detailed chart in the appendix that includes learning for larger real storage buffer sizes as a comparison. A six times larger real storage buffer loses knowledge much faster than Scalable Recollections despite better performance during training on CIFAR-100.

Question 3 *Can Scalable Recollections overcome the over-*

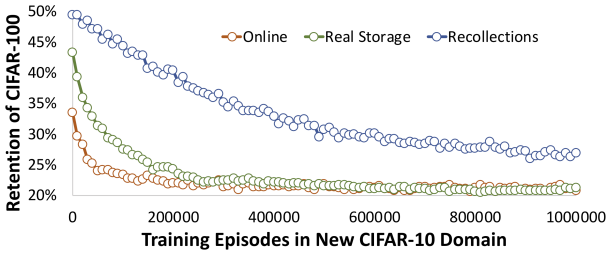

Figure 2: Retained accuracy on CIFAR-100 after prolonged training on CIFAR-10. CIFAR-10 contains images with a similar structure, but is drawn from a disjoint set of labels.

Model	Items	Retention
Replay Real Storage	200	43.0
Replay Recollections - No Transfer	1392	43.7
Replay Recollections - CIFAR-10 Transfer	1392	49.7
GEM Real Storage	200	48.7
GEM Recollections - No Transfer	1392	43.7
GEM Recollections - CIFAR-10 Transfer	1392	**54.2**
iCaRL (Rebuffi et al., 2017)	200	43.6

Table 3: Retention results on Incremental CIFAR-100 with a 200 real episode effective total storage resource footprint.

head of autoencoder model parameters?

To achieve the greater goals of lifelong learning, we are mostly interested in scaling to conditions where the number of examples seen is very large and as a result the incremental storage footprint dominates the overhead of the initial model parameters. However, we would like to empirically demonstrate that we can overcome this overhead in practical problems. Unfortunately, to demonstrate performance with a very small total storage footprint on a single dataset, an incredibly small autoencoder would then be required to learn a function for the very complex input space from scratch. We demonstrate in Table 3 that transfer learning provides a solution to this problem. By employing unlabeled background knowledge we are able to perform much better at the onset with a small autoencoder. We explore a total resource footprint on top of the size of F_θ equivalent to 200 real examples. This equates to the smallest setting explored in (Lopez-Paz and Ranzato 2017) and we demonstrate that we are able to achieve state of the art results by initializing only the autoencoder representation with one learned on CIFAR-10. CIFAR-10 is drawn from the same larger database as CIFAR-100, but is non-overlapping.

Why Scalable Recollections Work

Given the impressive performance of the Scalable Recollections Module for supporting the continual lifelong learning for neural networks, we would like to further explore the proposed system to elucidate why it works so well.

Question 4 *How do discrete latent codes compare with continuous latent codes for compression?*

In Figure 3 we empirically demonstrate that autoencoders with categorical latent variables can achieve significantly more storage compression of input observations at the same average distortion as autoencoders with continuous variables. In this experiment to make the continuous baseline even tougher to beat on the training set, we leverage a standard autoencoder instead of a variational one as it does not add noise to its representation, which would make it harder to reconstruct the original input. See the appendix for details.

Question 5 *How do learned methods of compression compare with static compression algorithms?*

In Figure 3 we also compare the performance of autoencoders with JPEG, which is a static compression algorithm commonly used in industry. We can see that JPEG performs quite well for low degrees of compression, but scales less gracefully than discrete autoencoder based compression for larger degrees of sample compression. This is because learned compression algorithms have the ability to further customize to the regularities seen in the data than a generic one. More detail is provided in the appendix.

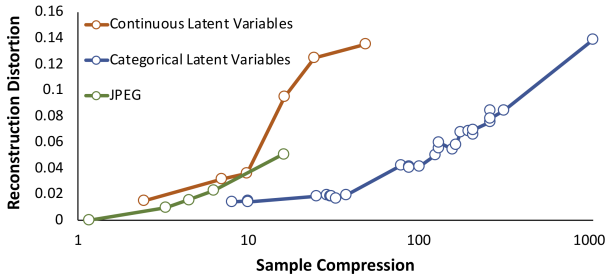

Figure 3: Comparing reconstruction L1 distance on the MNIST training set and sample compression for continuous latent variable and categorical latent variable autoencoders.

Question 6 *Do we see gains in learning efficiency as a result of the index buffer as predicted by Hypothesis 1?*

The recollection module must not only provide a means of compressing the storage of experiences in a scalable way, but also a mechanism for efficiently sampling recollections so that they are truly representative of prior experiences.

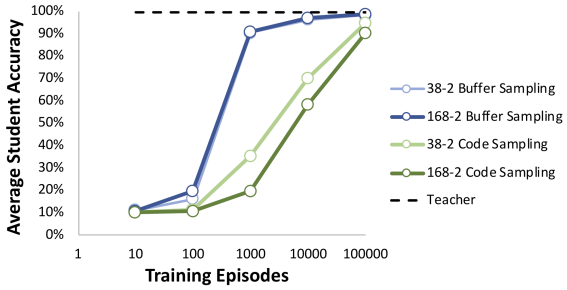

Figure 4: Comparison of generative transfer learning performance using a CNN teacher and student model on MNIST while using code sampling and recollection module sampling.

We first consider the typical method of sampling a variational autoencoder, which we will refer to as *code sampling*, where each latent variable is selected randomly. Obviously, by increasing the capacity of the autoencoder we are able to achieve lower reconstruction distortion. However, interestingly, we find that while increasing the autoencoder capacity increases modeling power, it also increases the chance that a randomly sampled latent code will not be representative of those seen in the training distribution. Instead, we maintain an index buffer of indexes associated with prior experiences. Let us call sampling from the index buffer, *buffer sampling*. Table 4 shows a comparison of code and buffer sampling for two different latent variable representation sizes. The reconstruction

Latent Representation	Sampling Strategy	Reconstruction Distortion	Nearest Neighbor Distortion
38 2d variables	Code Sampling	0.058	0.074
	Buffer Sampling	0.058	0.054
168 2d variables	Code Sampling	0.021	0.081
	Buffer Sampling	0.021	0.021

Table 4: Comparing the nearest training example L1 distance of code and buffer sampling based recollections averaged across 10,000 samples. Reconstruction distortion of the autoencoder is measured on the test set and is not influenced by the strategy.

distortion is the error in reconstructing the recollection using the decoder. The nearest neighbor distortion is the distance from the sampled code to its nearest neighbor in the training set. We can see that for the same reconstruction distortion, the buffer approach yields a significantly smaller nearest neighbor distortion. This means that the buffer sampling produces a more representative sample than code sampling.

How much does this matter in practice? Figure 4 demonstrates its utility through a knowledge distillation experiment. Here we compare the two representation sizes and approaches at the task of distilling a CNN teacher model into a student model of the same architecture through the latent codes. That is the student is trained on the reconstructed data using the teacher output as a label. By far the best learning curve is obtained using buffer sampling. We would like to emphasize that these results are not a byproduct of increased model capacity associated with the buffer: the small representation with the buffer significantly outperforms the big representation with code sampling despite 7.4x fewer total bits of storage including the model parameters and buffer. In the appendix we include comprehensive experiments showing that distillation based on buffer sampling with a discrete latent code VAE is even more efficient than storing real examples.

Question 7 *Does recollection based self-stabilization of the autoencoder lead to effective continual lifelong learning?*

We validate our recollection module training procedure by demonstrating that recollections generated by an autoencoder model can actually be effective in preventing catastrophic forgetting for the very same model. As shown in Figure 5 for continual learning on CIFAR-100 with number of steps $N = 10$ and an effective incremental buffer size of an average of two items per class, the recollection module is very similarly effective to real storage for stabilizing the lifelong autoencoder. The negative effects of the less effective synthetic examples are apparently drowned out by the positive effects of a larger diversity of stored examples.

Question 8 *Can recollection efficiency improve over time?*

We explore this question in Figure 6 where we consider online training of the model with a random initialization and no buffer (Online) and offline training with random initialization and full data storage (Offline) trained over 100 iterations. Predictably, access to unlimited storage and all of the tasks simultaneously means that the performance of Offline is consistently better than Online. To demonstrate the value of transfer from a good representation in the continual learning setting, we additionally plot an online model with no replay buffer and a representation initialized after training for 100 iterations on CIFAR-10 (Transfer Online). Transfer adds significant value, performing comparably to the randomly initialized model with access to all tasks simultaneously and unlimited storage (Offline). In fact, it performs considerably better for the first few tasks where the number of prior experiences is much greater than the number of new experiences. Improvements from transfer learning thus have a substantial effect in stabilizing F_θ as well as demonstrated in Table 3.

Conclusion

We have proposed and experimentally validated a general purpose Scalable Recollection Module that is designed to scale for very long time-frames. We have demonstrated superior performance over other state-of-the-art approaches for lifelong learning using very small incremental storage footprints. These increases can be dramatically boosted with unsupervised recollection module pre-training. We have shown that VAEs with categorical latent variables significantly outperform those with continuous latent variables (and even JPEG) for lossy compression. Finally, we have also shown that maintaining an explicit buffer is key to capturing the distribution of

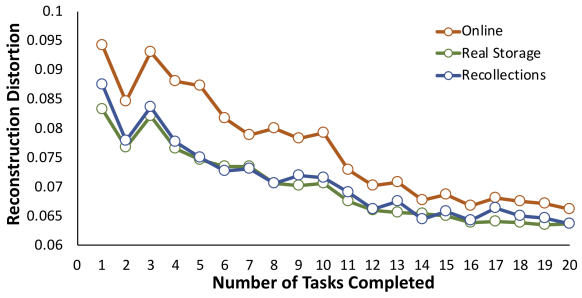

Figure 5: Test set L1 reconstruction distortion on Incremental CIFAR-100 with an effective incremental buffer size of 200.

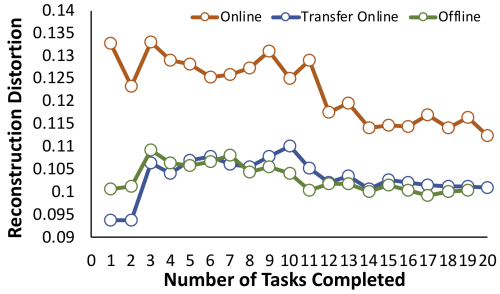

Figure 6: Test set L1 reconstruction distortion on Incremental CIFAR-100 of a 76 2d categorical latent variable autoencoder.

previously seen samples and generating realistic recollections needed to effectively prevent forgetting.

References

Al-Shedivat, M.; Bansal, T.; Burda, Y.; Sutskever, I.; Mordatch, I.; and Abbeel, P. 2017. Continuous adaptation via meta-learning in nonstationary and competitive environments. *arXiv preprint arXiv:1710.03641*.

Blundell, C.; Uria, B.; Pritzel, A.; Li, Y.; Ruderman, A.; Leibo, J. Z.; Rae, J.; Wierstra, D.; and Hassabis, D. 2016. Model-free episodic control. *arXiv preprint arXiv:1606.04460*.

Chen, X.; Kingma, D. P.; Salimans, T.; Duan, Y.; Dhariwal, P.; Schulman, J.; Sutskever, I.; and Abbeel, P. 2017. Variational lossy autoencoder. *ICLR*.

Fernando, C.; Banarse, D.; Blundell, C.; Zwols, Y.; Ha, D.; Rusu, A. A.; Pritzel, A.; and Wierstra, D. 2017. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*.

Jang, E.; Gu, S.; and Poole, B. 2017. Categorical reparameterization with gumbel-softmax. *ICLR*.

Kaiser, Ł.; Nachum, O.; Roy, A.; and Bengio, S. 2017. Learning to remember rare events. *arXiv preprint arXiv:1703.03129*.

Kingma, D. P., and Welling, M. 2014. Auto-encoding variational bayes. *ICLR*.

Kingma, D. P.; Salimans, T.; Jozefowicz, R.; Chen, X.; Sutskever, I.; and Welling, M. 2016. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, 4743–4751.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences* 201611835.

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

Lake, B.; Salakhutdinov, R.; Gross, J.; and Tenenbaum, J. 2011. One shot learning of simple visual concepts. In *Proceedings of the Cognitive Science Society*, volume 33.

Lee, J.; Yun, J.; Hwang, S.; and Yang, E. 2018. Lifelong learning with dynamically expandable networks. *ICLR*.

Li, Z., and Hoiem, D. 2016. Learning without forgetting. In *European Conference on Computer Vision*.

Lin, L.-J. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning* 8(3-4):293–321.

Lopez-Paz, D., and Ranzato, M. 2017. Gradient episodic memory for continuum learning. *NIPS*.

Maddison, C. J.; Mnih, A.; and Teh, Y. W. 2017. The concrete distribution: A continuous relaxation of discrete random variables. *ICLR*.

McClelland, J. L.; McNaughton, B. L.; and O'Reilly, R. C. 1995. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review* 102(3):419.

McCloskey, M., and Cohen, N. J. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of learning and motivation*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Murre, J. M. 1992. Learning and categorization in modular neural networks.

Pritzel, A.; Uria, B.; Srinivasan, S.; Badia, A. P.; Vinyals, O.; Hassabis, D.; Wierstra, D.; and Blundell, C. 2017. Neural episodic control. In *International Conference on Machine Learning*, 2827–2836.

Ramapuram, J.; Gregorova, M.; and Kalousis, A. 2017. Lifelong generative modeling. *arXiv preprint arXiv:1705.09847*.

Rebuffi, S.; Kolesnikov, A.; and Lampert, C. 2017. icarl: Incremental classifier and representation learning. *CVPR*.

Riemer, M.; Khabiri, E.; and Goodwin, R. 2016. Representation stability as a regularizer for improved text analytics transfer learning. *arXiv preprint arXiv:1704.03617*.

Ring, M. B. 1994. *Continual learning in reinforcement environments*. Ph.D. Dissertation, University of Texas at Austin Austin, Texas 78712.

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

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

Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Shin, H.; Lee, J. K.; Kim, J.; and Kim, J. 2017. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, 2994–3003.

Teyler, T. J., and DiScenna, P. 1986. The hippocampal memory indexing theory. *Behavioral neuroscience* 100(2):147.

Teyler, T. J., and Rudy, J. W. 2007. The hippocampal indexing theory and episodic memory: updating the index. *Hippocampus* 17(12):1158–1169.

Thrun, S. 1994. Lifelong learning perspective for mobile robot control. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, volume 1, 23–30.

Thrun, S. 1996. Is learning the n-th thing any easier than learning the first? *Advances in neural information processing systems* 640–646.

Vitter, J. S. 1985. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*.