Generative Knowledge Distillation for General Purpose Function Compression

Matthew Riemer, Michele Franceschini, Djallel Bouneffouf & Tim Klinger
IBM Research AI, Yorktown Heights, NY, USA
{mdriemer,franceschini,djallel.bouneffouf,tklinger}@us.ibm.com

Abstract

Deep lifelong learning systems need to efficiently manage resources to scale to large numbers of experiences and non-stationary goals. In this paper, we explore the relationship between lossy compression and the resource constrained lifelong learning problem of function transferability. We demonstrate that lossy episodic experience storage can enable efficient function transferability between different architectures and algorithms at a fraction of the storage cost of lossless storage. This is achieved by introducing a generative knowledge distillation strategy that does not store any full training examples.

1 Introduction

In this work we focus on developing a biologically motivated deep neural network architecture to accomplish the task of general purpose function transferrability. Our goal is to build a network that can efficiently transfer the function it learns to networks with a wide variety of a different architectures. This is a critical component of an ideal lifelong learning system. Optimal neural network architectures for learning have often been found to be problem specific and by enabling function transferrability we can more readily apply background knowledge using an architecture focused on the current goal. We achieve this by extending knowledge distillation techniques [4, 10] focused on transferring knowledge from a *teacher neural network* to a *student neural network*. Generally, the *student neural network* learns to imitate the output (sometimes modulated by a temperature) created by the *teacher neural network*. While most techniques explore knowledge distillation leveraging real labelled or unlabelled inputs, with scalability to lifelong many task learning in mind, we focus on learning to generate lossy recollections of inputs so that we can that compress the storage needed to maintain general purpose transferability of the teacher's function. Our proposed architecture based on recent discrete latent variable variational autonencoder models [11, 22] generates recollections of comparable effectiveness in training student models to real data, but with multiple orders of magnitude less memory resources consumed.

In our work, we focus on designing a way to scale experience replay solutions by reducing the footprint of storing an experience for recollection later in a non-stationary continual learning environment. This idea actually has biologically inspired motivation relating back to the pioneering work of [23], describing the potential complementary dynamics of the hippocampus and neocortex. In this theory, the hippocampus is responsible for fast learning, providing a very plastic representation for storing short term memories. Because the neocortex responsible for reasoning would otherwise suffer from catastrophic forgetting, the hippocampus also plays a key role in generating approximate recollections to interleave with incoming experiences, stabilizing the learning of the neocortex. To model the hippocampus as a recollection generation engine as suggested by [23], we develop a modern artificial neural network model of the *hippocampal memory index theory* in light of recent advancements in Deep Learning. *Hippocampal memory index theory* was first proposed by [28] as a theory for brain function in the hippocampus and later slightly revised in [29] after twenty years of related research. The theory primarily involves the hippocampus, believed to be involved in recalling previous

experiences, and the neocortex, believed to be responsible for reasoning. The crux of the theory is the idea that the hippocampus does not literally store previous experiences, but rather efficiently stores light weight *indexes* corresponding to information that can then be retrieved from a complimentary *association cortex*. The *association cortex* is considered to be part of the human neocortex, however, its location varies by species. As a result, in our work we model it as a separate component of our system.

Our artificial neural network implementation of this model consists of three primary modules. For clarity, we provide an illustration of our proposed three module architecture in Figure 1. We model a reasoning module with a standard supervised deep neural network that has an architecture suited to the current goal of the system. We also have an association module, which we model as a variational autoencoder [13] with discrete latent variables as in [11, 22]. By leveraging discrete latent variables, we can store codes with significantly smaller storage footprint at the same average distortion achieved by more traditional autoencoder models. The final component of our system is a recollection buffer that stores latent codes affiliated with prior experiences. In conjunction with the decoder of the association module, we can sample from the buffer to create approximate recollections. These recollections are provided as input to the reasoning module. We will demonstrate using this capability to transfer knowledge to a randomly initialized machine learning model without fully storing any data.

2 Related Work

Our proposed model is related to popular generative models such as variational autoencoders [13] and generative adversarial networks [7]. However, our work is unique in its use of an approximate replay buffer for faster generative knowledge transfer. Recent work also looks at the problem of generative lifelong learning [24] with a variational autoencoder, introducing a modified objective that would potentially be complementary to our contribution.

Even the first work on the topic of knowledge distillation [4] introduced a strategy for producing synthetic data to amplify real data. Additionally, unlabelled data has been widely used [26, 16, 1, 15] for knowledge distillation. Generative models have also been used as a sole source for distillation before in the context of language models [27], but not in the more general case where there is a separate input and output to generate for each example. By achieving high quality purely generative distillation, our goal is to obtain a form of general purpose knowledge transfer. As a result, our work is related in motivation to techniques that look to preserve knowledge after transforming the network architecture [5, 30].

[12] recently also looked at the problem of memory management with respect to lifelong learning. That work was primarily concerned with mechanisms for combining memories, and for incorporating

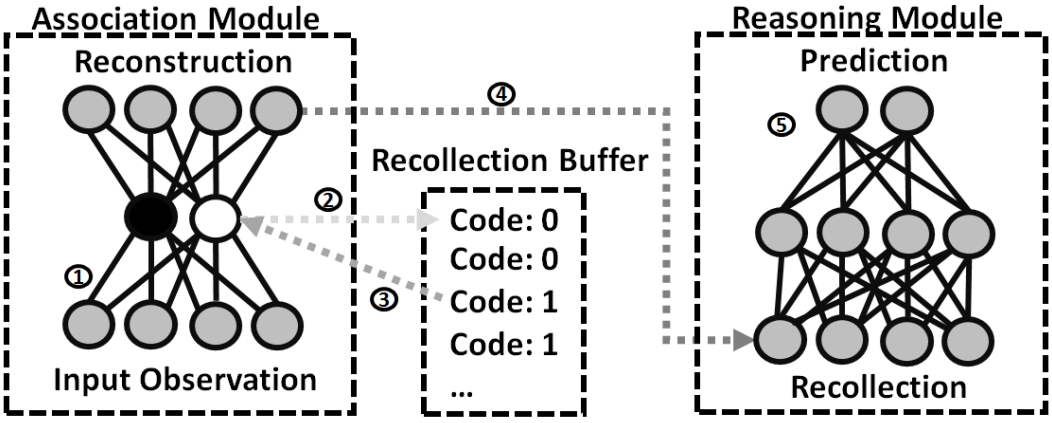

Figure 1: An example illustration of our proposed three module architecture. First, the input observation is encoded to a latent code that is stored in a recollection buffer. Later, a code is selected from the buffer and sent through the decoder to provide a recollection for which the reasoning module makes a prediction.

context dependent look-up with reasoning. Our main contribution instead lies in the compression of memory footprint, which has not been addressed by recent work on deep lifelong learning.

3 Lifelong Episodic Compression

Recent work on lifelong continual learning in deep neural networks [25], [20] 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 distortion of the prior experiences stored memory. We achieve this by considering the resource constraint in scaling lifelong learning not in terms of the number of full examples stored, but instead in terms of bits of storage. This is very practical because it relates back to real footprints on computer hardware. We would like to scale to an unbounded number of examples, but in practice we must ground our work in finite time horizons.

In this paper we explore experiments on the MNIST and Omniglot. For MNIST and Omniglot we follow prior work and consider 28x28 images with 1 channel and 8-bits per pixel. MNIST and Omniglot images were originally larger, but others have found the down sampling to 28x28 does not effect performance of models using it to learn. Multiplying out, we find that storing full images from MNIST and Omniglot will have a cost of 8x1x28x28 = 6,272 bits per image. By the same logic, storing 32x32 CIFAR images with 3 color channels and 8-bits per pixel per channel will have a cost of 8x3x32x32 = 24,576 bits. Deep non-linear autoencoders are considered a non-linear generalization of PCA and are a natural choice for compression problems. Theoretically, an autoencoder with a representation the same size as its input should be able to copy any input by simply learning an identity transformation. An autoencoder with a continuous latent variable of size h, assuming standard 32-bit representations used in modern GPU hardware, will have a storage cost of $32h$ bits for each latent representation.

One hurdle when using a standard continuous variable autoencoder for compressing input observations is that the 32 bits used for the model parameters, that also govern the representation size, most likely exceed the required precision. We propose a principled approach to guarding against this issue by leveraging the recently proposed variational autoencoder model with categorical latent variables [11, 22] to enable the model to learn a representation while explicitly considering a certain degree of storage precision. For a categorical latent variable autoencoder, we consider a bottleneck representation between the encoder and decoder with c categorical latent variables each containing l dimensions representing a *one hot* encoding of the categorical variable. Whereas this representation requires $S_{1\mathrm{h}} = lc$ bits to store the latent variable, with simple binary encoding of each categorical latent variable, we can store this representation with the following number of bits [6]:

$$S_{\mathrm{sbe}} = c \cdot \lceil \log_2(l) \rceil \,. \tag{1}$$

In Figure 2 we back up our theoretical intuition and 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. More detail is provided about this experiment in Appendix A.1.

The ability of a discrete variational autoencoder to memorize inputs should be strongly related to the effective bottleneck capacity C_{ve}, which we define, for discrete latent variables, as:

$$C_{\mathrm{ve}} = \log_2 l^c \,. \tag{2}$$

3.1 Incremental Storage Resource Constraints

First, let us consider the dynamics of balancing resources in a simple setting where we have an incremental storage constraint for new incoming data without regard for the size of the model used to compress and decompress recollections. We refer to the total storage constraint over all N incoming examples as γ and the average storage rate limit as γ/N. We can then define ρ as the probability that an incoming example is stored in memory. Thus, the expected number of bits required per example stored is ρS_{sbe}, assuming simple binary encoding. If we treat ρ as fixed, we can then define the following optimization procedure to search for a combination of c and l that maximizes capacity while fulfilling an incremental resource storage constraint:

$$\begin{aligned} \underset{c,l}{\text{maximize}} \quad & C_{\text{ve}} \\ \text{subject to} \quad & \rho S_{\text{sbe}} \leq \frac{\gamma}{N}, \end{aligned} \tag{3}$$

which yields the approximate solution $C_{\text{ve}} \simeq \frac{\gamma}{N\rho}$. As seen in equation 3, there is an inherent tradeoff between the diversity of experiences we store governed by ρ and the distortion achieved that is related to the capacity. The optimal tradeoff is likely problem dependent. However, it is often possible to achieve significant compression at only a small degree of distortion.

3.2 Total Storage Resource Constraints

In some ways, the incremental storage constraint setting described in the previous section is not the most rigorous setting when comparing lossy compression to lossless compression where a subset of full inputs are selected. Another important factor is the number of parameters in the model $|\theta|$ used for compression and decompression. $|\theta|$ generally is also to some degree a function of c and l. For example, in most of our experiments, we use the same number of hidden units cl at each layer as used in the bottleneck layer. With fully connected layers, this yields $|\theta|(c, l) \propto (cl)^2$. As such, we can revise equation 3 to handle a more rigorous constraint for optimizing a discrete latent variable autoencoder architecture:

$$\begin{aligned} \underset{c,l}{\text{maximize}} \quad & C_{\text{ve}} \\ \text{subject to} \quad & \rho S_{\text{sbe}} + |\theta|(c, l) \leq \gamma / N, \end{aligned} \tag{4}$$

While this setting is more rigorous when comparing to lossless inputs, it is a somewhat harsh restriction with which to measure lifelong learning systems. This is because it is assumed that the compression model's parameters should be largely transferrable across tasks. To some degree, these parameters can be viewed as a sunk cost from the standpoint of continual learning.

4 Lifelong Learning with a Recollection Buffer for Function Transfer

In *hippocampal memory index theory* there are two central capabilities to model: pattern completion and pattern separation. In this section we will first discuss the architecture of our proposed recollection generator used for pattern completion and then explain how it is combined with a buffer used for pattern separation.

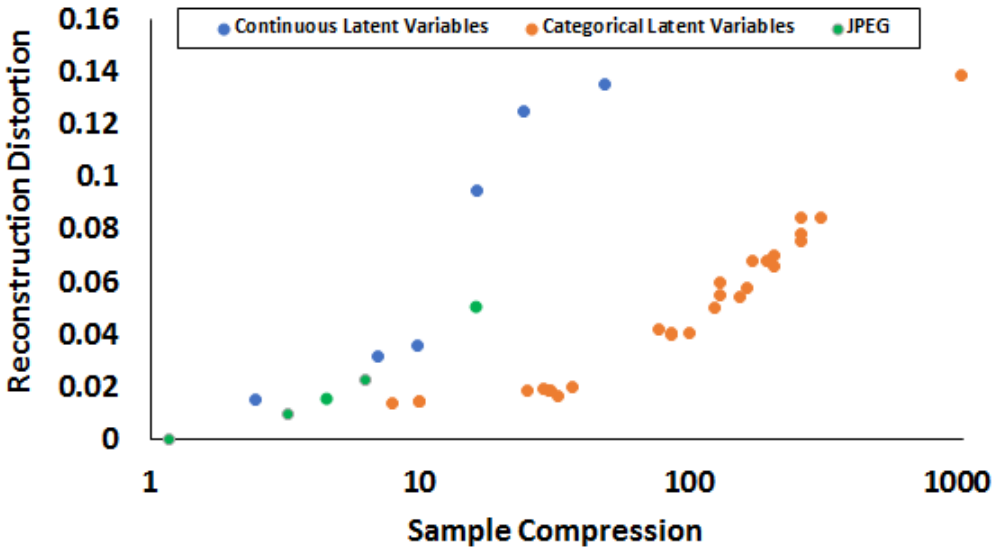

Figure 2: A comparison of the relationship between average reconstruction L1 distance on the MNIST training set and sample compression for both continuous latent variable and categorical latent variable autoencoders.

4.1 Categorical Latent Variable Autoencoders with the Gumbel-Softmax

Deep autoencoders, sometimes referred to as autoassociators, are trained to first compress and then reconstruct approximations of the input. By providing autoassociation capabilities, autoencoders excel at pattern completion. We can think of every autoencoder as having two major components called the encoder and decoder, which should have specialized architectures tuned to the problem of interest. In the standard formulation, the representation learned by the encoder z is generally modeled with continuous variables. In order to model an autoencoder with discrete latent variables, we follow the success of recent work [11, 22] and leverage the Gumbel-Softmax function. The Gumbel-Softmax function leverages the Gumbel-Max trick [9, 21] which provides an efficient way to draw samples z from a categorical distribution with class probabilities p_i:

$$z = \text{one\_hot}(\underset{i}{\text{argmax}}[g_i + \log(p_i)]) \tag{5}$$

In equation 5, g_i,...,g_d are samples drawn from Gumbel(0,1), which is calculated by drawing u_i from Uniform(0,1) and computing g_i=-log(-log(u_i)). The one_hot function quantizes its input into a one hot vector. The softmax function is used as a differentiable approximation to argmax, and we generate d-dimensional sample vectors y with temperature τ in which:

$$y_i = \frac{\exp((g_i + \log(p_i))/\tau)}{\sum_{j=1}^{d} \exp((g_j + \log(p_j))/\tau)} \tag{6}$$

The Gumbel-Softmax distribution is smooth for $\tau > 0$, and therefore has a well-defined gradient with respect to the parameters p. During forward propagation of our categorical autoencoder, we send the output of the encoder through the sampling procedure of equation 5 to create a categorical variable. However, during backpropagation we replace non-differentiable categorical samples with a differentiable approximation during training as the Gumbel-Softmax estimator in equation 6. Although past work [11, 22] has found value in varying τ over training, we still were able to get strong results keeping τ fixed at 1.0 across our experiments.

4.2 Recollection Buffer Design

Now that we have discussed our method for achieving pattern completion with a variational autoencoder, we will explain how we also preserve pattern separation by storing discrete latent codes to be used for recalling specific past experiences. Figure 3 provides an illustration of our proposed approach for distilling the function of a teacher neural network to a student neural network. A recollection buffer is maintained that stores latent codes corresponding to the autoencoder's compressed representation of the input. We either randomly or adaptively select codes from the recollection buffer and pass them through the decoder of the autoencoder to produce approximate recollections. The recollections are used as input to activate both a teacher neural network which produces an output vector for the student to imitate and a student neural network. The student neural network computes a loss function with respect to the teacher's output and learns using a variant of gradient descent. An alternative to consider is keeping a memory of the class index associated with each input as opposed to a teacher network as they are already in a very compact representation. In our experiments it was more effective to train using continuous signal of the teacher's output than the real discrete output classes. However, using the real class is more robust to imperfections in the teacher network.

5 Evaluation Settings

In this section we will empirically support the methods proposed in the previous sections by applying them to the popular MNIST digit recognition dataset [19] and the Omniglot character recognition dataset [17] considering each of the 50 alphabets to be a task. Across all of our experiments, our generator model is a discrete latent variable convolutional variational autoencoder including three convolutional layers in the encoder and three deconvolutional layers in the decoder. In all of our knowledge distillation experiments, we report an average result over 5 runs. More details can be found for all of our experiments in Appendix A.

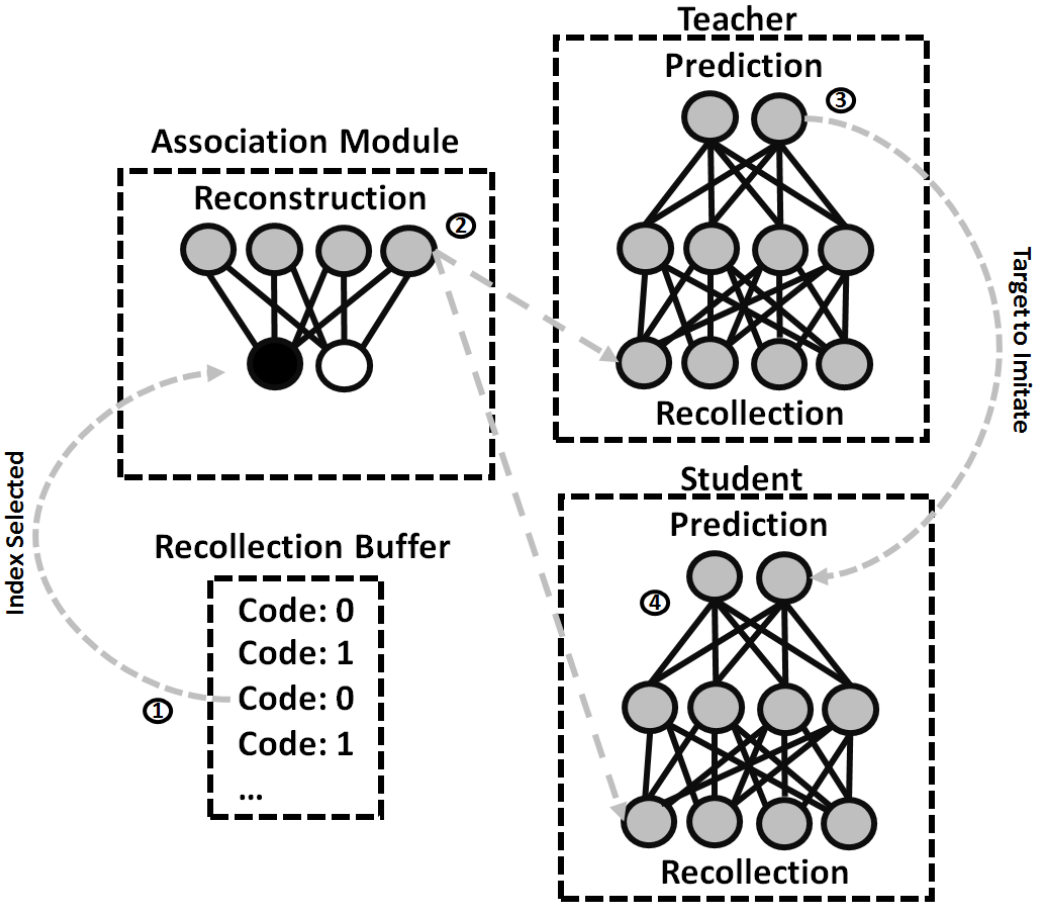

Figure 3: An illustration of how our proposed recollection generator is used to produce recollections to transfer knowledge from a teacher neural network to a student neural network.

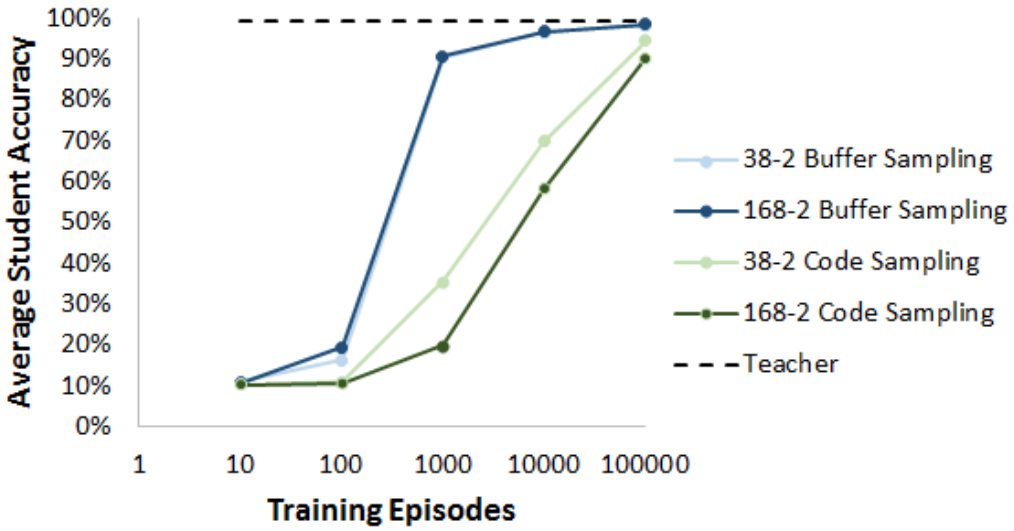

Figure 4: Comparison of generative transfer learning performance using a CNN teacher and student model on MNIST while using code sampling and recollection buffer sampling.

5.1 General Purpose Knowledge Transfer Experiments

We will now empirically demonstrate our proposed recollection generator's ability to transfer knowledge from a teacher neural network to a student model. In our experiments, we train a teacher model with a LeNet [18] convolutional neural network (CNN) architecture on the popular MNIST benchmark, achieving 99.29% accuracy on the test set. Alongside the teacher model, we train a generator model with discrete latent variables. Each model is trained for 500 epochs. During the final pass through the data, we forward propogate through each training example and store the latent code in a recollection buffer. This buffer eventually grows to a size of 50,000. After training is complete, the recollection buffer is used as a statistical basis for sampling diverse recollections to train a student network. A logical and effective strategy for training a student model is to sample randomly from this buffer and thus capture the full distribution.

Validating the utility of the recollection buffer. First, we will empirically make the case for our proposal of using episodic recollections to augment a variational autoencoder model for knowledge transfer. In Figure 4 we look at two discrete latent variable models with 168 2d variables and 38 2d variables, comparing performance for buffer based sampling with standard random code sampling of with no explicit episodic memory. We see that much more efficient knowledge transfer can be achieved by sampling from an episodic buffer. It is important to note that this advantage is not simply a result of increased storage used by the recollection buffer. The smaller model with an episodic buffer achieves better transfer performance than the larger model does using code sampling. This is despite the fact that the larger model's parameter storage is 7.4x more than the total storage of the parameters and buffer combined for the smaller model. Actually, if anything we are seeing the opposite behavior. Interestingly, we are observing decreased transfer ability with the larger model when using code sampling. We hypothesize that although the reconstruction modeling power of a discrete autoencoder increases with capacity, there is a tension between increasing the capacity and as a result increasing the dimensionality of the latent space to search through. Large latent spaces have large regions which they can model that are not statistically representative of the data distribution seen. By providing the variational autoencoder with an episodic buffer we are able to get the best of both worlds: the ability to increase the size of the representation used for modeling and the ability to efficiently transfer from it in a way that is representative of the prior distribution of inputs.

Comparing the recollection buffer to lossless baslines. In Table 1 we further validate the effectiveness of our technique by comparing it to some additional baselines of interest. As baselines we consider training with the the same number of randomly sampled real examples, using real input and the teacher's output vector as a target, and using random sampling to select a subset of lossless memories with equivalent storage footprints. When training with a large number of memories for a more complete knowledge transfer, the recollection compression clearly shows dividens over random sampling baselines. This is impressive particularly because these results are for the stricter resource constraint setting discussed in section 3.2 and on a per sample basis the compression is actually 37x, 101x, and 165x to account for the autoencoder model capacity. We also would like to validate these findings in a more complex setting for which we consider distillation with outputs from a 50 task Resnet-18 teacher model that gets 94.86% accuracy on Omniglot. We test out performance after one million training episodes, which is enough to achieve teacher performance using all of the real training examples. However, sampling diversity restricts learning significantly, for example, achieving 28.87% accuracy with 10% sampling, 8.88% with 2% sampling, and 5.99% with 1% sampling. In contrast lossy compression is much more effective, achieving 87.86% accuracy for 10x total resource compression, 74.03% accuracy for 50x compression, and 51.45% for 100x compression.

Distilling functions to different neural architectures. Our main motivation for enabling general purpose knowledge distillation is for occasions where we would like to change the architectural form of our knowledge over time, not keep it constant. In Table 5 we consider distillation from our LeNet CNN teacher model to a multi-layer perceptron (MLP) student with two hidden layers of 300 hidden units. For MLPs we again see our recollection compression is comparable to the performance of real examples while using much less storage, and lossy compression scales much better than sampling lossless inputs.

5.2 Automated Generative Curriculum Learning Experiments

We would like to maximize the efficiency of distilling knowledge from a teacher model to a student model. This motivates the automated curriculum learning setting [2] as recently explored for multi-

Episodes	Real Data	10% Sample	2% Sample	1% Sample	Real x Teacher y	10x Compress	50x Compress	100x Compress
10	10.43	9.94	11.07	10.70	10.07	10.65	10.99	**13.89**
100	19.63	18.16	22.82	22.35	**25.32**	19.34	16.20	21.06
1000	90.45	88.88	90.71	89.93	**91.01**	90.66	90.52	90.03
10000	97.11	96.83	95.98	94.97	**97.42**	96.77	96.37	95.65
100000	98.51	97.99	96.14	94.92	**98.63**	98.59	98.17	97.75

Table 1: Generative knowledge distillation random sampling experiments with a CNN teacher and student model on MNIST.

Episodes	Real Data	Real x Teacher y	Active 10x Compress	Active 100x Compress	Active & Diverse 10x Compress	Active & Diverse 100x Compress
10	10.43	10.07	9.95	10.19	10.67	**11.51**
100	19.63	25.32	14.80	22.57	27.05	**29.93**
1000	90.45	91.01	93.45	92.97	**94.81**	92.54
10000	97.11	97.42	**98.61**	97.53	98.59	97.66
100000	98.51	98.63	99.18	98.25	**99.20**	98.32

Table 2: Generative knowledge distillation active and diverse sampling experiments with a CNN teacher and student model on MNIST. The real input baselines are randomly sampled.

task learning in [8] or rather automated generative curriculum learning in our case. We tried some simple reinforcement learning solutions with rewards based on [8] but were unsuccessful in our initial experiments because of the difficulty of navigating a complex continuous action space. We also tried an active learning formulation proposed for GANs to learn the best latent code to sample [33] at a given time. We had limited success with this strategy as well as it tends to learn to emphasize regions of the latent space that optimize incorrectness, but no longer capture the distribution of inputs.

Designing generative sampling heuristics. Inspired by these findings, we instead employ simple sampling heuristics to try to design a curriculum with prototypical qualities like responsiveness to the student and depth of coverage. We model responsiveness to the student as *active sampling* by focusing on examples where the student does not have good performance. We randomly sample k latent codes using our recollection buffer and choose the one that is most difficult for the current student for backpropagation by cheaply forward propagating through the student for each. By sampling from the recollection buffer, we are able to ensure our chosen difficult samples are still representative of the training distribution. We set k to 10 in our experiments so the sampling roughly equates to sampling once from the most difficult class for the student model at each point in time. We model depth of coverage by sampling a bigger batch of random examples and adding a filtering step before considering difficulty. We would like to perform *diverse sampling* that promotes subset diversity when we filter from kn examples down to k examples. One approach to achieving this is a Determinantal Point Process (DPP) [14] as recently proposed for selecting diverse neural network mini-batches [32]. We use the dot product of the inputs as a measure of similarity between recollections and found the DPP to achieve effective performance as a diverse sampling step. However, we follow [3] and use a process for sampling based on the sum of the squared similarity matrix as outlined in Appendix A.4. We found the sum of the squared similarity matrix to be equally effective to the determinant and significantly more scalable to large matrices. We also set n to 10 in our experiments. In Table 2 we demonstrate the superior performance of the proposed active and diverse sampling strategies. Consistently we are able to achieve more efficient training of the student networks than is achieved with random real examples.

6 Conclusion

We have proposed a discrete latent variable autoencoder based model for generative knowledge distillation, enabling general purpose function transferrability. Autoencoders with discrete latent variables are capable of far more sample compression than continuous latent variable models. Additionally, variational autoencoders equipped with a lossy recollection buffer are capable of multiple orders of magnitude faster knowledge distillation in comparison to traditional code sampling. We consistently find lossy compression of memories to be more effective than lossless episodic sampling. In exploring the task of automated generative curriculum learning, we find that a teacher with an intelligent generator can outperform random real examples in terms of the efficiency of teaching a function to a student neural network. This promising result demonstrates how general purpose machine to machine knowledge communication can be a useful tool for efficiently training models to perform new skills that have been mastered by other models.

Acknowledgments

We would like to thank Irina Rish, Ignacio Cases, Cicero dos Santos, Kahini Wadhawan, Brian Kingsbury, and Bowen Zhou for engaging discussions that helped make this work possible.

References

[1] Shuang Ao, Xiang Li, and Charles X Ling. Fast generalized distillation for semi-supervised domain adaptation. 2017.

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[3] Djallel Bouneffouf and Inanc Birol. Sampling with minimum sum of squared similarities for nystrom-based large scale spectral clustering. 2015.

[4] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. ACM, 2006.

[5] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

[6] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[8] Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. *arXiv preprint arXiv:1704.03003*, 2017.

[9] Emil Julius Gumbel. *Statistical theory of extreme values and some practical applications: a series of lectures*. Number 33. US Govt. Print. Office, 1954.

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

[11] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *ICLR*, 2017.

[12] Łukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. Learning to remember rare events. *arXiv preprint arXiv:1703.03129*, 2017.

[13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2014.

[14] Alex Kulesza, Ben Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

[15] Mandar Kulkarni, Kalpesh Patil, and Shirish Karande. Knowledge distillation using unlabeled mismatched images. *arXiv preprint arXiv:1703.07131*, 2017.

[16] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *ICLR*, 2017.

[17] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the Cognitive Science Society*, volume 33, 2011.

[18] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[19] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[20] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continuum learning. *NIPS*, 2017.

[21] Chris J Maddison, Daniel Tarlow, and Tom Minka. A* sampling. In *Advances in Neural Information Processing Systems*, pages 3086–3094, 2014.

[22] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *ICLR*, 2017.

[23] James L McClelland, Bruce L McNaughton, and Randall C O'reilly. 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, 1995.

[24] Jason Ramapuram, Magda Gregorova, and Alexandros Kalousis. Lifelong generative modeling. *arXiv preprint arXiv:1705.09847*, 2017.

[25] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, and Christoph H Lampert. icarl: Incremental classifier and representation learning. *CVPR*, 2017.

[26] Matthew Riemer, Elham Khabiri, and Richard Goodwin. Representation stability as a regularizer for improved text analytics transfer learning. *arXiv preprint arXiv:1704.03617*, 2017.

[27] Sungho Shin, Kyuyeon Hwang, and Wonyong Sung. Generative knowledge transfer for neural language models. 2017.

[28] Timothy J Teyler and Pascal DiScenna. The hippocampal memory indexing theory. *Behavioral neuroscience*, 100(2):147, 1986.

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

[30] Tao Wei, Changhu Wang, Yong Rui, and Chang Wen Chen. Network morphism. In *International Conference on Machine Learning*, pages 564–572, 2016.

[31] Yongxin Yang and Timothy Hospedales. Deep multi-task representation learning: A tensor factorisation approach. *ICLR*, 2017.

[32] Cheng Zhang, Hedvig Kjellstrom, and Stephan Mandt. Stochastic learning on imbalanced data: Determinantal point processes for mini-batch diversification. *arXiv preprint arXiv:1705.00607*, 2017.

[33] Jia-Jie Zhu and Jose Bento. Generative adversarial active learning. *arXiv preprint arXiv:1702.07956*, 2017.

A Additional Details on Experimental Protocol

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. In practice, this implies that the number of filters in each layer is equal to $cl/4$. We note that the discrete autoencoder is stochastic, not deterministic and we just report one stochastic pass through the data for each experimental trial.

A.1 Distortion as a Function of Compression Experiments

More detail about the architecture used in these experiments are provided for categorical latent variables in Table 3 and for continuous latent variables in Table 4. For each architecture we ran with a learning rate of 1e-2, 1e-3, 1e-4, and 1e-5, reporting the option that achieves the best training distortion. For the distortion, the pixels are normalized by dividing by 255.0 and we take the mean over the vector of the absolute value of the reconstruction to real sample difference and then report the mean over the samples in the training set. Compression is the ratio between the size of an 8bpp

c	l	Compression	Distortion
6	20	209.067	0.06609
10	20	125.440	0.04965
6	16	261.333	0.07546
12	10	130.667	0.05497
10	14	156.800	0.05410
24	3	130.667	0.05988
38	2	165.053	0.05785
6	2	1045.333	0.13831
40	3	78.400	0.04158
20	2	313.600	0.08446
8	6	261.333	0.08423
12	6	174.222	0.06756
30	2	209.067	0.06958
24	6	87.111	0.04065
4	37	261.333	0.07795
8	15	196.000	0.06812
48	10	32.667	0.01649
209	8	10.003	0.01455
12	37	87.111	0.03996
313	4	10.019	0.01420
392	3	8.000	0.01348
50	18	25.088	0.01859
168	2	37.333	0.01955
108	3	29.037	0.01894
62	2	101.161	0.04073
208	2	30.154	0.01832
68	5	30.745	0.01849

Table 3: This table provide more specifics about the discrete latent variable architectures involved in Figure 2.

h	Compression	Distortion
1	49	0.135196
2	24.5	0.124725
3	16.33333333	0.0947032
5	9.8	0.0354035
7	7	0.031808
20	2.45	0.0149272

Table 4: This table provide more specifics about the continuous latent variable architectures involved in Figure 2.

MNIST image and the size of the latent variables, assuming 32 bits floating point numbers in the continuous case and the binary representation as in (1) for the categorical variables. The JPEG data points were collected using the Pillow Python package using quality 1, 25, 50, 75, and 100. We subtracted the header size form the JPEG size so it is a relatively fair accounting of the compression for a large data set of images all of the same size. The JPEG compression is computed as an average over the first 10,000 MNIST training images.

A.2 MNIST Generative Distillation Experiments

For all of our distillation experiments we ran the setting with a learning rate of 1e-3 and 1e-4, reporting the best result. We found that the higher learning rate was beneficial in setting with a low number of examples and the lower learning rate was beneficial in setting with a larger number of examples. The categorical latent variable autoencoders explored had the following representation sizes: 168 2d variables for 10x compression, 62 2d variables for 50x compression, and 38 2d variables for 100x compression. For our code sampling baselines, we used the numpy random integer function to generate each discrete latent variable.

Episodes	Real Data	10% Sample	2% Sample	1% Sample	Real x Teacher y	10x Compress	50x Compress	100x Compress
10	13.64	**17.04**	14.57	15.13	15.87	16.70	11.80	14.66
100	36.37	37.04	38.35	34.04	38.56	37.16	40.09	**42.31**
1000	80.54	79.08	78.18	77.76	80.00	**80.72**	80.00	77.75
10000	91.04	90.84	88.38	86.83	90.86	**91.37**	90.60	90.46
100000	96.66	95.02	91.61	88.97	96.60	**96.71**	96.24	95.22

Table 5: Generative knowledge distillation random sampling experiments with a CNN teacher and MLP student model on MNIST.

A.3 Omniglot Generative Distillation Experiment

The learning rate for the Resnet-18 reasoning model was 1e-4 in our experiments. Our trained discrete autoencoder models were of the following representation sizes: 32 variables of size 2 for 100x compression, 50 variables of size 2 for 50x compression, and 134 variables of size 2 for 10x compression. We follow 90% multi-task training and 10% testing splits for Omniglot established in [31].

A.4 Minimum Sum of Squared Similarities

This algorithm is trying to find a new landmark point that maximizes the determinant by finding a point that minimizes the sum of squared similarities (MSSS). The MSSS algorithm initially randomly chooses two points from the dataset X. It then computes the sum of similarities between the sampled points and a subset, T, selected randomly from the remaining data points. The point with the smallest sum of squared similarities is then picked as the next landmark data point. The procedure is repeated until a total of m landmark points are picked.

Algorithm 1 The Minimum Sum of Squared Similarities Algorithm

Input: $X = \{x_1, x_2, ..., x_n\}$: dataset
m: number of landmark data points
γ: size of the subsampled set from the remaining data, in percentage

Output: $\widetilde{S} \in R^{m \times m}$: similarity matrix between landmark points
Initialize $\widetilde{S} = I_0$
For (i=0 to i<2) do
    $\widetilde{x}_i = Random(X)$
    $\widetilde{S} := \widetilde{S}_{\cup x_i}$
    $\widetilde{X} := \widetilde{X} \cup \{\widetilde{x}_i\}$
End For
While $i < m$ **do**
    $T = Random(X \backslash \{\widetilde{X}\}, \gamma)$
    Find $\widetilde{x}_i = argmin_{x \in T} \sum_{j<i-1} sim^2(x, \widetilde{x}_j)$
    $\widetilde{S} := \widetilde{S}_{\cup \widetilde{x}_i}$
    $\widetilde{X} := \widetilde{X} \cup \{\widetilde{x}_i\}$
End While

B CNN to MLP Distillation Results