Continual Learning with Deep Generative Replay

Hanul Shin
Massachusetts Institute of Technology
SK T-Brain
skyshin@mit.edu

Jung Kwon Lee,* **Jaehong Kim**,* **Jiwon Kim**
SK T-Brain
{xhark,jklee,jk}@sktbrain.com

Abstract

Attempts to train a comprehensive artificial intelligence capable of solving multiple tasks have been impeded by a chronic problem called catastrophic forgetting. Although simply replaying all previous data alleviates the problem, it requires large memory and even worse, often infeasible in real world applications where the access to past data is limited. Inspired by the generative nature of hippocampus as a short-term memory system in primate brain, we propose the Deep Generative Replay, a novel framework with a cooperative dual model architecture consisting of a deep generative model ("generator") and a task solving model ("solver"). With only these two models, training data for previous tasks can easily be sampled and interleaved with those for a new task. We test our methods in several sequential learning settings involving image classification tasks.

1 Introduction

One distinctive ability of humans and large primates is to continually learn new skills and accumulate knowledge throughout the lifetime [6]. Even in small vertebrates such as rodents, established connections between neurons seem to last more than an year [13]. Besides, primates incorporate new information and expand their cognitive abilities without seriously perturbing past memories. The flexible memory system results from a good balance between synaptic plasticity and stability [1].

Continual learning in deep neural networks, however, suffers from a phenomenon called *catastrophic forgetting* [21], in which the performance of a model on previously learned tasks abruptly degrades when trained for a new task. In artificial neural networks, inputs are coincided with the outputs by implicit parametric representation. Therefore training them towards a new objective can cause almost complete forgetting of former knowledge. Such problem has been a key obstacle to continual learning for deep neural network through sequential training on multiple tasks.

Previous attempts to alleviate catastrophic forgetting often relied on episodic memory system that stores past data [29]. In particular, recorded examples are regularly replayed with real samples drawn from the new task, and the network parameters are jointly optimized. While a network trained in this manner performs as well as separate networks trained solely on each task [27], a major drawback of memory-based approach is that it requires large working memory to store and replay past inputs. Moreover, such data storage and replay may not be viable in some real-world situations.

Notably, humans and large primates learn new knowledge even from limited experience and still retain past memories. While several biological mechanisms contribute to this at different levels, primate brain's the most apparent distinction from artificial neural networks is the existence of separate, interacting memory systems [24]. The Complementary Learning Systems (CLS) theory illustrates the significance of dual memory systems involving the hippocampus and the neocortex. The hippocampal system acts as a short-term memory that encodes recent experiences and passes them to the neocortex, a lifelong memory storage. Here, the memory is consolidated through multiple replays of earlier

---

*Equal Contribution

experiences stored in the hippocampus [25]–a mechanism which inspired the use of experience replay [22] in training deep reinforcement learning agents.

Recent evidences suggest that the hippocampus is more than a simple experience replay buffer. Rather, it regenerates earlier inputs by synchronized reactivations that are induced during unconscious or conscious recall or specific phase of sleep [8]. Stimulation of certain memory traces in the hippocampus can even create false memory that was never experienced [26]. These properties suggest that the hippocampus is better paralleled with a generative model than a replay buffer. Specifically, deep generative models such as deep Boltzmann machines [30] or a variational autoencoder [17] can generate high-dimensional samples that closely match observed inputs.

We now propose an alternative approach to sequentially train deep neural networks without referring to past data. In our deep generative replay framework, the model retains previously acquired knowledge by the concurrent replay of generated pseudo-data. In particular, we train a deep generative model in the generative adversarial networks (GANs) framework [10] to mimic past data. Generated data are then paired with corresponding response from a copy of the past task solver to represent old tasks. Called the scholar model, the generator-solver pair can produce fake data and desired target pairs as much as needed, and when presented with a new task, these produced pairs are interleaved with new data to update the generator and solver networks. Thus, a scholar model can both learn the new task without forgetting its own knowledge and teach other models with generated input-target pairs, even when the network configuration is different.

As deep generative replay supported by the scholar network retains the knowledge without revisiting actual past data, this framework can be employed to various practical situation involving privacy issues. Recent advances on training generative adversarial networks suggest that the trained models can reconstruct real data distribution in a wide range of domains. Although we tested our models on image classification tasks, our model can be applied to any task as long as the trained generator reliably reproduces the input space.

2   Related Works

The term *catastrophic forgetting* or *catastrophic interference* was first introduced by McCloskey and Cohen in 1980's [21]. They claimed that catastrophic interference is a fundamental limitation of neural networks and a downside of its high generalization ability. While the cause of catastrophic forgetting has not been studied analytically, it is known that the neural networks parameterize the internal features of inputs, and training the networks on new samples causes alteration in already established representations. Several works illustrate empirical consequences in sequential learning settings [7, 27], and provide a few primitive solutions [16, 28] such as replaying all previous data.

2.1   Comparable methods

A branch of works assumes a particular situation where access to data is limited to the current task. These works focus on optimizing network parameters while minimizing alterations to already consolidated weights. It is suggested that regularization methods such as dropout [31] and L2 regularization help reduce interference of new learning [12]. Furthermore, elastic weight consolidation (EWC) proposed in [18] demonstrates that protecting certain weights based on their importance to the previous tasks tempers the performance loss.

Other attempts to sequentially train a deep neural network capable of solving multiple tasks reduce catastrophic interference by augmenting the networks with task-specific parameters. In general, layers close to inputs are shared to capture universal features, and independent output layers produce task-specific outputs. Although separate output layers are free of interference, alteration on earlier layers still cause some performance loss on older tasks. Lowering learning rates on some parameters is also known to reduce forgetting [9]. A recently proposed method called Learning without Forgetting (LwF) [20] addresses the problem of sequential learning in image classification tasks while minimizing alteration on shared network parameters. In this framework, the network's response to new task input prior to fine-tuning indirectly represents knowledge about old tasks and is maintained throughout the learning process.

2.2 Complementary Learning System(CLS) theory

A handful of works are devoted to designing a complementary networks architecture to resist catastrophic forgetting. When the training data for previous tasks are not accessible, only pseudo-inputs and pseudo-targets produced by a memory network can be fed into the task network. Called a pseudorehearsal technique, this method is claimed to maintain old input-output patterns without accessing real data [29]. When the tasks are as elementary as coupling two binary patterns, simply feeding random noises and corresponding responses suffices [2]. A more recent work proposes an architecture that resembles the structure of the hippocampus to facilitate continual learning for more complex data such as small binary pixel images [15]. However, none of them demonstrates scalability to high-dimensional inputs similar to those appear in real world due to the difficulty of generating meaningful high-dimensional pseudoinputs without further supervision.

Our generative replay framework differs from aforementioned pseudorehearsal techniques in that the fake inputs are generated from learned past input distribution. Generative replay has several advantages over other approaches because the network is jointly optimized using an ensemble of generated past data and real current data. The performance is therefore equivalent to joint training on accumulated real data as long as the generator recovers the input distribution.

2.3 Deep Generative Models

Generative model refers to any model that generates observable samples. Specifically, we consider deep generative models based on deep neural networks that maximize the likelihood of generated samples being in given real distribution [11]. Some deep generative models such as variational autoencoders [17] and the GANs [10] are able to mimic complex samples like images.

The GANs framework defines a zero-sum game between a generator G and a discriminator D. While the discriminator learns to distinguish between the generated samples from real samples by comparing two data distributions, the generator learns to mimic the real distribution as closely as possible. The objective of two networks is thereby defined as:

$$\min_{G} \max_{D} V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Theoretically, the Nash equilibrium of this zero-sum game is when the generator perfectly mimics p_{data} and the discriminator cannot distinguish the fake from the real and outputs $\frac{1}{2}$ everywhere.

3 Generative Replay

We first define several terminologies. In our continual learning framework, we define the sequence of tasks to be solved as a *task sequence* $\mathbf{T} = (T_1, T_2, \cdots, T_N)$ of N tasks.

Definition 1 *A task T_i is to optimize a model towards an objective on data distribution D_i, from which the training examples $(\boldsymbol{x}_i, \boldsymbol{y}_i)$'s are drawn.*

Next, we call our model a *scholar*, as it is capable of learning a new task and teaching its knowledge to other networks. Note that the term scholar differs from standard notion of teacher-student framework of ensemble models [5], in which the networks either teach or learn only.

Definition 2 *A scholar H is a tuple $\langle G, S \rangle$, where a generator G is a generative model that produces real-like samples and a solver S is a task solving model parameterized by θ.*

The solver has to perform all tasks in the task sequence $\mathbf{T}$. The full objective is thereby given as to minimize the unbiased sum of loss among all tasks in the task sequence $\mathbb{E}_{(\boldsymbol{x}, \boldsymbol{y}) \sim D}[L(S(\boldsymbol{x}; \theta), \boldsymbol{y})]$, where D is the entire data distribution and L is a loss function. While being trained under the task T_i, the model is fed with samples drawn from D_i.

3.1 Proposed Method

We consider sequential training on our scholar model. However, training a single scholar model while referring to the recent copy of the network is equivalent to training a sequence of scholar models

$(H_i)_{i=1}^N$ where the n-th scholar H_n $(n > 1)$ learns the current task T_n and the knowledge of previous scholar H_{n-1}. Therefore, we describe our full training procedure as in Figure 1(a).

Training the scholar model from another scholar involves two independent procedures of training the generator and the solver. First, the new generator receives current task input x and replayed inputs x' from previous tasks. Real and replayed samples are mixed at a ratio that depends on the desired importance of a new task compared to the older tasks. The generator learns to reconstruct cumulative input space, and the new solver is trained to couple the inputs and targets drawn from the same mix of real and replayed data. Here, the replayed target is past solver's response to replayed input. Formally, the loss function of the i-th solver is given as

$$L_{train}(\theta_i) = r\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D_i}[L(S(\boldsymbol{x};\theta_i),\boldsymbol{y})] + (1-r)\,\mathbb{E}_{\boldsymbol{x}'\sim G_{i-1}}[L(S(\boldsymbol{x}';\theta_i), S(\boldsymbol{x}';\theta_{i-1}))] \quad (1)$$

where θ_i are network parameters of the i-th scholar and r is a ratio of mixing real data. As we aim to evaluate the model on original tasks, test loss differs from the training loss:

$$L_{test}(\theta_i) = r\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D_i}[L(S(\boldsymbol{x};\theta_i),\boldsymbol{y})] + (1-r)\,\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D_{past}}[L(S(\boldsymbol{x};\theta_i),\boldsymbol{y})] \quad (2)$$

where D_{past} is a cumulative distribution of past data. Second loss term is ignored in both function when $i = 1$ because there is no replayed data to refer to for the first solver.

We build our scholar model with a solver that has suitable architecture for solving a task sequence and a generator trained in the generative adversarial networks framework. However, our framework can employ any deep generative model as a generator.
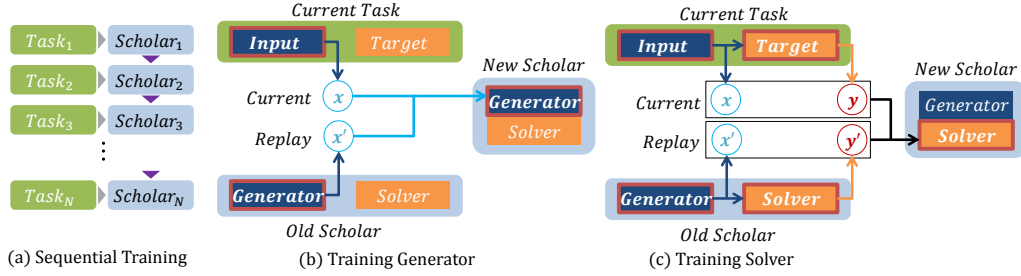

Figure 1: Sequential training of scholar models. (a) Training a sequence of scholar models is equivalent to continuous training of a single scholar while referring to past copy of a self. (b) A new generator is trained to mimic a mixed data distribution of real samples $\boldsymbol{x}$ and replayed inputs $\boldsymbol{x}'$ from previous generator. (c) A new solver learns from real input-target pairs $(\boldsymbol{x}, \boldsymbol{y})$ and replayed input-target pairs $(\boldsymbol{x}', \boldsymbol{y}')$, where replayed response $\boldsymbol{y}'$ is obtained by feeding generated inputs into previous solver.

3.2 Preliminary Experiment

Prior to our main experiments, we show that the trained scholar model alone suffices to train an empty network. We tested our model on classifying MNIST handwritten digit database [19]. Sequence of scholar models were trained from scratch through generative replay from previous scholar. The accuracy on classifying full test data is shown in Table 1. We observed that the scholar model transfers knowledge without losing information.

Table 1: Test accuracy of sequentially learned solver measured on full test data from MNIST database. The first solver learned from real data, and subsequent solvers learned from previous scholar networks.

	$Solver_1$ $\rightarrow$	$Solver_2$ $\rightarrow$	$Solver_3$ $\rightarrow$	$Solver_4$ $\rightarrow$	$Solver_5$
Accuracy(%)	98.81%	98.64%	98.58%	98.53%	98.56%

4 Experiments

In this section, we show the applicability of generative replay framework on various sequential learning settings. Generative replay based on a trained scholar network is superior than other continual learning approaches in that the quality of the generative model is the only constraint of the task

performance. In other words, training the networks with generative replay is equivalent to joint training on entire data when the generative model is optimal. To draw the best possible result, we used WGAN-GP [14] technique in training the generator.

As a base experiment, we test if generative replay enables sequential learning while compromising performance on neither the old tasks nor a new task. In section 4.1, we sequentially train the networks on independent tasks to examine the extent of forgetting. In section 4.2, we train the networks on two different yet relevant domains. We demonstrate that generative replay not only enables continual learning on our design of the scholar network but also compatible with other known structures. In section 4.3, we show that our scholar network can gather knowledge from different tasks to perform a meta-task, by training the network on disjoint subsets of training data.

We compare the performance of the solver trained with variants of replay methods. Our model with generative replay is notated in the figure as *GR*. We specify the upper bound by assuming a situation when the generator is perfect. Therefore, we replayed actual past data paired with the predicted targets from the old solver network. We denote this case as *ER* for exact replay. We also consider the opposite case when the generated samples do not resemble the real distribution at all. Such case is denoted as *Noise*. A baseline of naively trained solver network is denoted as *None*. We use the same notation throughout this section.

4.1 Learning independent tasks

The most common experimental formulation used in continual learning literature [32, 18] is a simple image classification problem where the inputs are images from MNIST handwritten digit database [19], but pixel values of inputs are shuffled by a random permutation sequence unique to each task. The solver has to classify permuted inputs into the original classes. Since the most, if not all pixels are switched between the tasks, the tasks are technically independent from each other, being a good measure of memory retention strength of a network.
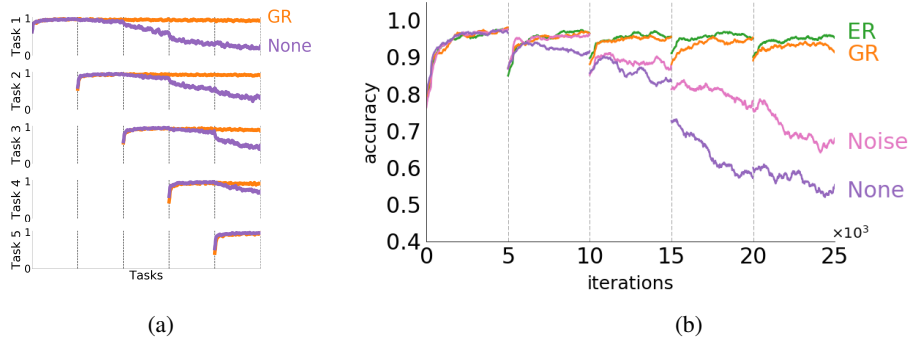

Figure 2: Results on MNIST pixel permutation tasks. (a) Test performances on each task during sequential training. Performances for previous tasks dropped without replaying real or meaningful fake data. (b) Average test accuracy on learnt tasks. The achieved higher accuracy when the replayed inputs better resembled real data.

We observed that generative replay maintains past knowledge by recalling former task data. In Figure 2(a), the solver with generative replay (orange) maintained the former task performances throughout sequential training on multiple tasks, in contrast to the naively trained solver (violet). An average accuracy measured on cumulative tasks is illustrated in Figure 2(b). While the solver with generative replay achieved almost full performance on trained tasks, sequential training on a solver alone incurred catastrophic forgetting (violet). Replaying random gaussian noises paired with recorded responses did not help tempering performance loss (pink).

4.2 Learning new domains

Training independent tasks on the same network is inefficient because no information is to be shared. We thus demonstrate the advantage of our model in more reasonable settings where the model aids from solving multiple tasks. In particular, we consider the expanded generalization of classes to new domains which share semantically the same classes.

A model operating in multiple domains has several advantages over that only works in a single domain. First, the knowledge of one domain can help better and faster understanding of other domains if not the domains are completely independent. Second, generalization over multiple domains may result in more universal knowledge that is applicable to unseen domains. Such phenomenon is also observed in infants learning to categorize objects [3, 4]. Encountering similar but diverse objects, young children can infer the properties shared within the category, and can make a guess of which category that the new object may belong to.

We tested if the model can incorporate the knowledge of a new domain with generative replay. In particular, we sequentially trained our model on classifying MNIST and Street View House Number (SVHN) dataset [23], and vice versa. Experimental details are provided in supplementary materials.
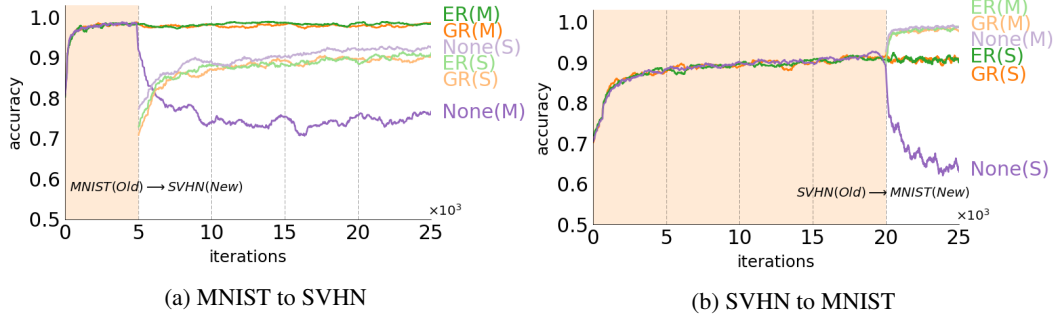

(a) MNIST to SVHN

(b) SVHN to MNIST

Figure 3: Accuracy on classifying samples from two different domains. (a) The models are trained on MNIST then on SVHN dataset or (b) vice versa. When the previous data are recalled by generative replay (orange), knowledge of the first domain is retained as if the real inputs with predicted responses are replayed (green). Sequential training on the solver alone incurs forgetting on the former domain, thereby resulting in low average performance (violet).

1000 iterations   2000 iterations   5000 iterations   10000 iterations   20000 iterations
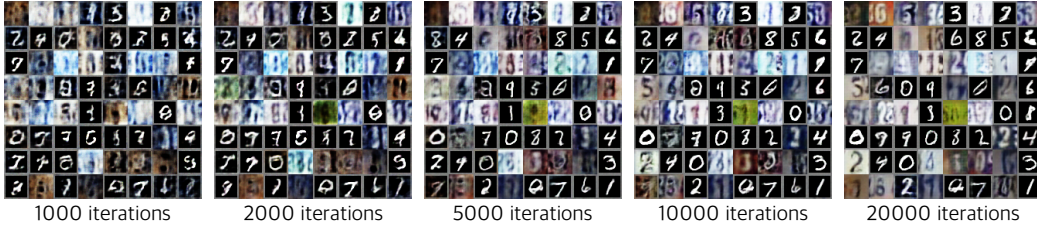
Figure 4: Samples from trained generator in MNIST to SVHN experiment after training on SVHN dataset for 1000, 2000, 5000, 10000, and 20000 iterations. The samples are diverted to mimic either SVHN or MNIST input images.

Figure 3 illustrates the performance on the original task (thick curves) and the new task (dim curves). A solver trained alone lost its performance on the old task when no data are replayed (purple). Since MNIST and SVHN input data share similar spatial structure, the performance on former task did not drop to zero, yet the decline was critical. In contrast, the solver with generative replay (orange) maintained its performance on the first task while accomplishing the second one. The results were no worse than replaying past real inputs paired with predicted responses from the old solver (green). In both cases, the model trained without any replay data achieved slightly better performance on new task, as the network was solely optimized to solve the second task.

Generative replay is compatible with other continual learning models as well. For instance, Learning without Forgetting (LwF), which replays current task inputs to revoke past knowledge, can be augmented with generative models that produce samples similar to former task inputs. Because LwF requires the context information of which task is being performed to use task-specific output layers, we tested the performance separately on each task. Note that our scholar model with generative replay does not need the task context.

In Figure 5, we compare the performance of LwF algorithm with a variant LwF-GR, where the task-specific generated inputs are fed to maintain older network responses. We used the same training
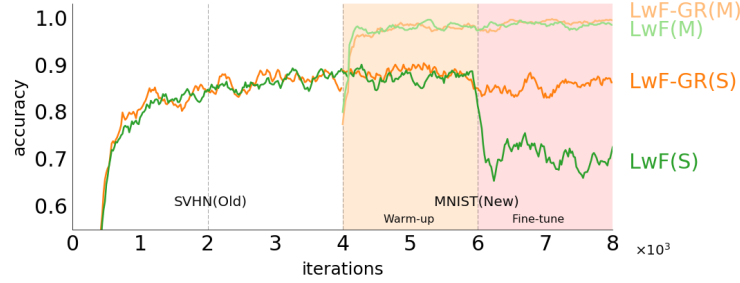
Figure 5: Performance of LwF and LwF augmented with generative replay (LwF-GR) on classifying samples from each domain. The networks were trained on SVHN then on MNIST database. Test accuracy on SVHN classification task (thick curves) dropped when the shared parameters were fine-tuned, but generative replay greatly tempered the loss (orange). Both networks achieved high accuracy on MNIST classification (dim curves).

regime as proposed in the original literature, namely warming up the new network head for some amount of the time and then fine tuning the whole network. The solver trained with original LwF algorithm loses performance on the first task when fine-tuning begins, due to alteration to shared network (green). However, with generative replay, the network maintains most of the past knowledge (orange).

4.3 Learning new classes

To illustrate that generative replay can recollect the past knowledge even when the inputs and targets are highly biased between the tasks, we propose a new experiment in which the network is sequentially trained on disjoint data. In particular, we assume a situation where the agent can access examples of only a few classes at a time. The agent eventually has to correctly classify examples from all classes after being sequentially trained on mutually exclusive subsets of classes. We tested the networks on MNIST handwritten digit database.

Note that training the artificial neural networks independently on classes is difficult in standard settings, as the network responses may change to match the new target distribution. Hence replaying inputs and outputs that represent former input and target distributions is inevitable to train a balanced network. We thus compare the variants described earlier in this section from the perspective of whether the input and target distributions of cumulative real data is recovered. For *ER* and *GR* models, both the input and target distributions represent cumulative distribution. *Noise* model maintains cumulative target distributions, but the input distribution only mirrors current distribution. *None* model has current distribution for both.
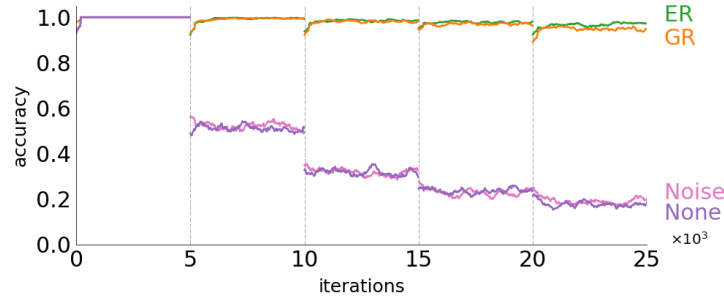

Figure 6: The models were sequentially trained on 5 tasks where each task is defined to classify MNIST images belong to 2 out of 10 labels. In this case, the networks are given with examples of 0 and 1 during the first task, 2 and 3 for the second, and in the same manner. Only our networks achieved test performance close to the upper bound.

In Figure 6, we divided MNIST dataset into 5 disjoint subsets, each of which contains samples from only 2 classes. When the networks are sequentially trained on the subsets, we observed that a naively trained classifier completely forgot previous classes and only learned the new subset of data (purple).

7

Table 2: Comparison of EWC, LwF and Generative Replay. Generative replay is favorable than other two methods in that it is equivalent to joint training on accumulated real data as long as the trained generator can recover real input space.

	EWC	LwF	Generative Replay
Performance Guarantee	until saturated	depends on task similarity	depends on generator
Network Size	huge	increasing	ordinary
Task Balancing	hard	moderate	easy
Old Tasks Performance	compromised	compromised	optimized
New Task Performance	compromised	optimized	optimized
Network Flexibility	none	little	high

Recovering only the past output distribution without a meaningful input distribution did not help retaining knowledge, as evidenced by the model with a noise generator (pink). When both the input and output distributions are reconstructed, generative replay evoked previously learnt classes, and the model was able to discriminate all encountered classes (orange).

Figure 7: Generated samples from trained generator after the task 1, 2, 3, 4, and 5. The generator is trained to reproduce cumulative data distribution.

Because we assume that the past data are completely discarded, we trained the generator to mimic both current inputs and the generated samples from the previous generator. The generator thus reproduces cumulative input distribution of all encountered examples so far. As shown in Figure 7, generated samples from trained generator include examples equally from encountered classes.

5 Discussion

We introduce deep generative replay framework, which allows sequential learning on multiple tasks by generating and rehearsing fake data that mimics former training examples. The trained scholar model comprising a generator and a solver serves as a knowledge base of a task. Although we described a cascade of knowledge transfer between a sequence of scholar models, a little change in formulation proposes a solution to other topically relevant problems. For instance, if the previous scholar model is just a past copy of the same network, it can learn multiple tasks without explicitly partitioning the training procedure.

As comparable approaches, regularization methods such as EWC and careful training the shared parameters as in LwF have shown that catastrophic forgetting could be alleviated by protecting former knowledge of the network. However, regularization approaches constrain the network with additional loss terms for protecting weights, so they potentially suffer from the tradeoff between the performances on new and old tasks. To guarantee good performances on both tasks, one should train on a huge network that is much larger than normally needed. Also, the network has to maintain the same structure throughout all tasks when the constraint is given specific to each parameter as in EWC. Drawbacks of LwF framework are also twofold: the performance highly depends on the relevance of the tasks, and the training time for one task linearly increases with the number of former tasks.

In Table 2, we compare our method with Elastic Weight Consolidation and Learning without Forgetting, which are proven to achieve the best results so far in own branches. In marked contrast to other approaches, generative replay maintains the former knowledge solely with produced input-target pairs, so it allows ease of balancing the former and new task performance and flexible knowledge

transfer. Most importantly, the network is jointly optimized towards task objectives, hence guaranteed to achieve the full performance when the former input spaces are recovered by the generator. One defect of generative replay framework is that the efficacy of algorithm heavily depends on the quality of a generator. Indeed, we observed some performance loss while training the model on SVHN dataset with in same setting employed in section 4.3. Detailed analysis is provided in supplementary materials.

We acknowledge that the three branches are not completely exclusive, as they contribute to memory retention at different levels. Nevertheless, each method poses some constraints on training procedure or network configurations, and there is no straightforward mixture of any two frameworks. Still, we believe a good mix of three frameworks would give better solution to the chronic problem in continual learning.

Future works of generative replay may include extension to reinforcement learning domain or developing continuously evolving network that maintains knowledge from previous copy of the self. Also, we expect the improvements in training deep generative models would directly aid the performance of generative replay framework on more complex domains.

References

[1] W. C. Abraham and A. Robins. Memory retention–the synaptic stability versus plasticity dilemma. *Trends in neurosciences*, 28(2):73–78, 2005.

[2] B. Ans and S. Rousset. Avoiding catastrophic forgetting by coupling two reverberating neural networks. *Comptes Rendus de l'Académie des Sciences-Series III-Sciences de la Vie*, 320(12):989–997, 1997.

[3] D. A. Baldwin, E. M. Markman, and R. L. Melartin. Infants' ability to draw inferences about nonobvious object properties: Evidence from exploratory play. *Child development*, 64(3):711–728, 1993.

[4] M. H. Bornstein and M. E. Arterberry. The development of object categorization in young children: Hierarchical inclusiveness, age, perceptual attribute, and group versus individual analyses. *Developmental psychology*, 46(2):350, 2010.

[5] T. G. Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[6] J. Fagot and R. G. Cook. Evidence for large long-term memory capacities in baboons and pigeons and its implications for learning and the evolution of cognition. *Proceedings of the National Academy of Sciences*, 103(46):17564–17567, 2006.

[7] R. M. French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[8] H. Gelbard-Sagiv, R. Mukamel, M. Harel, R. Malach, and I. Fried. Internally generated reactivation of single neurons in human hippocampus during free recall. *Science*, 322(5898):96–101, 2008.

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[11] I. J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160, 2017.

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

[13] J. Grutzendler, N. Kasthuri, and W.-B. Gan. Long-term dendritic spine stability in the adult cortex. *Nature*, 420(6917):812–816, 2002.

[14] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

[15] M. Hattori. A biologically inspired dual-network memory model for reduction of catastrophic forgetting. *Neurocomputing*, 134:262–268, 2014.

[16] G. E. Hinton and D. C. Plaut. Using fast weights to deblur old memories. In *Proceedings of the ninth annual conference of the Cognitive Science Society*, pages 177–186, 1987.

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

[18] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[20] Z. Li and D. Hoiem. Learning without forgetting. In *European Conference on Computer Vision*, pages 614–629. Springer, 2016.

[21] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of learning and motivation*, 24:109–165, 1989.

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

[23] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5, 2011.

[24] R. C. O'Reilly and K. A. Norman. Hippocampal and neocortical contributions to memory: Advances in the complementary learning systems framework. *Trends in cognitive sciences*, 6(12):505–510, 2002.

[25] J. O'Neill, B. Pleydell-Bouverie, D. Dupret, and J. Csicsvari. Play it again: reactivation of waking experience and memory. *Trends in neurosciences*, 33(5):220–229, 2010.

[26] S. Ramirez, X. Liu, P.-A. Lin, J. Suh, M. Pignatelli, R. L. Redondo, T. J. Ryan, and S. Tonegawa. Creating a false memory in the hippocampus. *Science*, 341(6144):387–391, 2013.

[27] R. Ratcliff. Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285–308, 1990.

[28] A. Robins. Catastrophic forgetting in neural networks: the role of rehearsal mechanisms. In *Artificial Neural Networks and Expert Systems, 1993. Proceedings., First New Zealand International Two-Stream Conference on*, pages 65–68. IEEE, 1993.

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

[30] R. Salakhutdinov and G. Hinton. Deep boltzmann machines. In *Artificial Intelligence and Statistics*, pages 448–455, 2009.

[31] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[32] R. K. Srivastava, J. Masci, S. Kazerounian, F. Gomez, and J. Schmidhuber. Compete to compute. In *Advances in neural information processing systems*, pages 2310–2318, 2013.