Learning to Remember what to Remember: A Synaptic Plasticity Driven Framework

Oleksiy Ostapenko[1,2], **Mihai Puscas**[1,3], **Tassilo Klein**[1], **Patrick Jähnichen**[2], **Moin Nabi**[1]

[1]SAP Machine Learning Research
[2]Humboldt-Universität zu Berlin
[3]University of Trento
{oleksiy.ostapenko, mihai.puscas, tassilo.klein, m.nabi}@sap.com,
{oleksiy.ostapenko, patrick.jaehnichen}@hu-berlin.de

Abstract

Continually trainable models should be able to learn from a stream of data over an undefined period of time. The main challenges of continual learning (CL) are the ability to maintain old knowledge while benefiting from it when learning new tasks, as well as guaranteeing model scalability with growing amount of data to learn from. In this work we introduce Dynamic Generative Memory (DGM) for continual learning - a memory based approach that relies on conditional generative adversarial networks with learnable connection plasticity. Here, connection plasticity is realized with neural masking. We evaluate two variants of neural masking: applied to (i) layer activations and (ii) to connection weights directly. We compare the performance of DGM with state-of-the-art methods.

1 Introduction

Artificial Neural Networks (ANN) fail to continually learn from a stream of data, while maintaining knowledge, not to mention reusing old knowledge in new contexts. Generally, the fundamental obstacles on the way to a continually trainable AI are the problem of catastrophic forgetting when learning from new data, the missing knowledge transfer across the tasks and lack of model scalability, i.e. the inability to scale up model size with a continuously growing amount of training data.

Several recent approaches try to mitigate forgetting by simulating synaptic plasticity in ANNs [5, 18, 2, 1]. Common to all these methods is the idea of discouraging updates of the network parameters that keep old knowledge when learning new tasks. [14] propose to rely on a hard attention to the task (HAT) mechanism. HAT finds parameter subspaces for all tasks while allowing them to mutually overlap. The optimal solution is then found in the corresponding parameter subspace of each task. It is noteworthy that all methods above tackle the task-incremental scenario, i.e. a separate classification layer is trained to make predictions about each task. This further implies the availability of oracle knowledge of task label at inference time. Such evaluation is often refereed to as multi-head evaluation, in which the task label is associated with a dedicated output head. Alternatively, other approaches rely on single-head evaluation [12, 2]. Here, the model is evaluated on all classes observed during the training, no matter which task they belong to. While single-head evaluation does not require oracle knowledge of the task label, it also does not reduce the output space of the model and thus represents a harder, yet more realistic evaluation setup. Single-head evaluation is predominantly used in class-incremental setup.

As opposed to task-incremental setup, models in a class incremental situation typically require replaying samples of previous categories when learning new one. [12, 2, 9] show that such a replay based on real samples of previous tasks significantly alleviates the problem of catastrophic forgetting

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

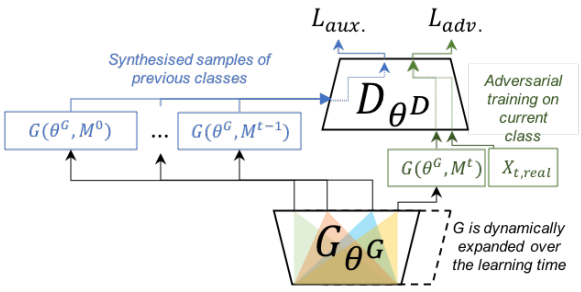

Figure 1: Dynamic generative memory: The AC-GAN[10] architecture allows simultaneous adversarial and auxiliary training, in which auxiliary output is trained on actual samples of the current task and synthesized samples of previous ones. During adversarial training with real and fake samples of the current task, we jointly learn the connection plasticity mask and the weights of the base network in the generator. Connection plasticity masks are task specific and are represented by binary matrices.

in a class-incremental scenario. Yet, retaining samples is very much against the notion of bio inspired design as the brain by no means features the retrieval of raw information identical to originally exposed impressions [8]. Additionally, as pointed out by [17] and [12] storing raw samples of previous data can violate data privacy and memory restrictions of real world applications.

A separate line of work relies on *generative* replay [15, 4, 16]. Here, a generative model is trained incrementally to synthesize samples of previous tasks that can be replayed during subsequent learning. These approaches require repetitive retraining of the generator, as the training set of synthesized samples (of previous tasks) and new samples changes every time a new task is introduced. Apart from being inefficient, this approach is severely prone to "semantic drifting", i.e. the quality of images generated at each memory replay highly depends on the images generated during previous replays. This causes susceptibility to error propagation and can result in a loss of quality over time. We propose to avoid replaying samples to the generator by utilizing learnable parameter masking.

Another important factor in the continual learning setting is the ability to scale, i.e. to maintain sufficient capacity to accommodate for a continuously growing number of tasks. Given invariant resource constraints, it is inevitable that with a growing number of tasks to learn, the model capacity is depleted at some point in time. This issue is again exacerbated when simulating neural plasticity with hard attention mechanisms such as parameter level attention masking. Weights blocked for previous tasks can be reused but not changed during subsequent learning, continuously reducing the degree of freedom of the network. We ensure sufficient degrees of freedom for every new task by keeping the number of "free" weights (i.e. weights that can be changed) constant through expanding the network by the number of parameters that were blocked for the previous task.

Our contribution is twofold: **(a)** we introduce **Dynamic Generative Memory** (DGM) - an adversarially trainable generative network that features neuronal plasticity through efficient learning of a sparse attention mask for the network weights (DGMw) or layer activations (DGMa); a *single* generator is able to incrementally learn new information during normal adversarial training *without the need to replay previous knowledge*; **(b)** We propose an **adaptive network expansion mechanism**, facilitating resource efficient continual learning. We compare the proposed method to state-of-the-art approaches for continual learning. Finally, we demonstrate that DGMw accommodates for higher efficiency, better parameter re-usability and slower network growth then DGMa.

2 Dynamic generative memory with learnable connection plasticity

Learning the DGM includes synthesizing samples from previous tasks and replaying them to a single-head classification model at each step of continual training. As visualized in Fig. 1, we learn a Generative Adversarial Network [3] *and* a sparse mask for the weights of its generator simultaneously. In doing so, we repeatedly assign certain network parameters to tasks, prohibiting overwriting of these parameters in the subsequent learning, at the same time promoting their re-usability across tasks insuring knowledge transfer. To this end, we largely rely on the idea of learning a binary mask from a real valued embedding matrix as exploited by [14, 6, 7]. In particular, [6, 7] propose to learn a

binary mask filter on top of a pretrained base network without changing the base network's weights. In contrast, [14] propose to simultaneously learn the base network and a binary mask for the layer *activations*. We pursue a similar strategy, with a key difference that binary masks are applied to the weight matrices directly (DGMw), instead of to the layer activations. The learned masks model connection plasticity of neurons, thus avoiding overwriting of important units by restricting SGD updates to network parameter segments that correspond to free capacity. Dynamic network expansion facilitates the addition of neurons in order to provide sufficient capacity to deal with new tasks.

General idea. We consider a generator network G_{θ^G} consisting of L layers, and a discriminator network D_{θ^D}. The system has to continually learn T tasks. During the SGD based training of task $t \in \{1, \ldots, T\}$, we learn a set of binary masks $M^t = [m_1^t, ..., m_L^t]$ for the weights of each layer.Output of layer l is obtained by combining the binary mask m_l^t with the layer weights:

$$y_l^t = \sigma_{act}[(m_l^t \circ W_l)^\top x], \quad W_l \in \mathbb{R}^{p \times n}, \tag{1}$$

for l fully connected and σ_{act} some activation function. W_l is the weights matrix for connections between layer l and $l-1$ and $\cdot \circ \cdot$ corresponds to the Hadamard product of matrices. Extension to more complex models as e.g. CNNs is straight forward.

In our approach, D_{θ^D} serves as both a discriminator for newly synthesized samples and as a classifier for the original learning problem. D_{θ^D} is reinitialized for every task t and retrained using t's training data and synthesized samples of previously seen ones.

Learning a binary mask. A single binary mask for the generator's layer l and task t is given by

$$m_l^t = \sigma(s \cdot e_l^t), \tag{2}$$

where e_l^t is a real-valued mask embeddings matrix, s is a positive scaling parameter $s \in \mathbb{R}_+$, and σ a thresholding function $\sigma : \mathbb{R} \to [0, 1]$. We use the sigmoid function as a pseudo step-function in order to ensure gradient flow to the embeddings e. In training, we anneal the scaling parameter s incrementally during epoch i from 0 to s_{max}^i. s_{max}^i is similarly adjusted over the course of I epochs from 0 to s_{max} (s_{max} is a meta-parameter) following the scheme largely adopted from [14]:

$$s_{max}^i = \frac{1}{s_{max}} + (s_{max} - \frac{1}{s_{max}})\frac{i-1}{I-1}, \quad s = \frac{1}{s_{max}^i} + (s_{max}^i - \frac{1}{s_{max}^i})\frac{b-1}{B-1}. \tag{3}$$

Here $b \in \{1, \ldots, B\}$ is the batch index and B the number of batches in each epoch of SGD training.

In order to prevent the overwriting of the knowledge related to previously seen classes in the generator network, gradients g_l w.r.t. the weights of each layer l are multiplied by the reverse of the accumulated binary masks for all previous tasks:

$$g_l' = [1 - m_l^{\leq t}]g_l, \quad m_l^{\leq t} = \max(m_l^t, m_l^{t-1}), \tag{4}$$

where g_l' corresponds to the new weights gradient and $m_l^{\leq t}$ is the accumulated mask.

Similarly to [14], we promote sparsity of the binary mask by adding a regularization term R^t to the loss function L_G of the AC-GAN[10] generator:

$$R^t(M^t, M^{t-1}) = \frac{\sum_{l=1}^{L-1}\sum_{i=1}^{N_l} m_{l,i}^t(1 - m_{l,i}^{<t})}{\sum_{l=1}^{L-1}\sum_{i=1}^{N_l} 1 - m_{l,i}^{<t}}, \tag{5}$$

where N_l is the number of parameters of layer l. Here, parameters that were reserved previously are not penalized, promoting reuse of weight units over reserving new ones as new tasks are learned.

Dynamic network expansion. As discussed by [7], significant domain shift between tasks leads to rapid network capacity exhaustion, ultimately manifesting in catastrophic forgetting. This can be explained by decreasing sparsity of the accumulated mask $m_l^{\leq t}$ over the course of training. To avoid this effect, we take measures to ensure stationary sparsity of the masks in each training cycle t.

Consider network layer l with input vector of size p and output vector of size n. At the beginning of the initial training cycle, the binary mask $m_l^1 \in [0, 1]^{p \times n}$ is initialized with zero sparsity. Thus, all neurons of the layer will be used, with all values of the mask m_l^1 set to 0.5 (real-valued embeddings e_l^1 are initialized with 0).

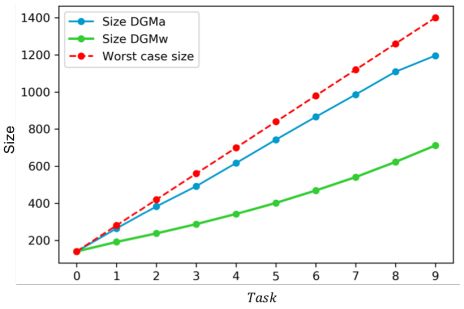

Figure 2: Network growth when learning a plasticity mask for connections versus activations in the incremental MNIST setup. Worst case scenario refers to growing the network by adding the initial size after each task. Size is reported as number of activations (neurons) of the layer, initial size is 140 ($\sim 50\%$ of the size used by [16])

Method	MNIST (%) A_5	MNIST (%) A_{10}	SVHN(%) A_5	SVHN(%) A_{10}
EWC-M [13]	70.62	77.03	39.84	33.02
DGR [15]	90.39	85.4	61.29	47.28
MeRGAN [16]	98.19	97.0	80.90	66.78
JT	97.66	98.1	85.30	84.82
DGMw	98.26	96.33	80.37	67.05
DGMa	**99.17**	**98.14**	**84.18**	**68.36**

Table 1: Comparison to the benchmark presented by [16] of approaches that make use of generative memory. Joint training (JT) represents the the performance of the discriminator trained in non-incremental fashion. Both, performance of DGM with binary mask for activations (DGMa) and connections (DGMw) is presented after incremental learning of 5 and 10 classes for MNIST and SVHN datasets.

After the initial training cycle with the sparsity regularizer R^1, the number of free weight parameters not reserved by the mask will decrease to $np - \delta_1$, where, in general, δ_t is the number of parameters reserved for the generation task t ($t = 1$ here). After training cycle t we expand layer l's number of output weights n by δ_t/p. The free capacity of the layer is kept constant: $(n + \delta_t/p)p - \delta_t = np$. In practice we extend the number of output units n by $\lceil \delta_t/p \rceil$. The number of free parameters is thus either np, if $\delta_t/p \in \mathbb{Z}$, or $np + p$ otherwise.

3 Experiments

First, we compare DGM with other methods that make use of generative memory. Table 1 shows that both variants of our method perform similarly or slightly better then MeRGAN[16] on the (split-) MNIST benchmark (classes are introduced incrementally with one class per time step). This suggests that for this dataset both methods have largely curbed the effects of catastrophic forgetting. DGMa outperforms joint training after seeing both 5 and 10 tasks whereas MeRGAN is able to do so after learning 5 tasks but not after learning 10 tasks. We conclude that incremental training with generated samples indeed forced the network to learn better generalizations compared to a joint training setting. In the (split-) SVHN benchmarks DGMa and DGMw reach a slightly better performance then MeRGAN[16], however all three approaches are still far from reaching the joint training accuracy. DGM(a and w) features identical architecture as used by [16] - a 3 layer DCGAN [11] with projection and reshape operation being performed with a convolutional layer instead of a fully connected one. We initialize the generator to be $\sim 50\%$ the size of the network used by [16].

Further, we evaluate the efficiency of (generator) network growth for both versions of our method. Figure 2 reports network growth against the number of tasks learned. We find that learning masks directly for the layer weights dramatically boosts the parameter re-usability, slowing down network growth as new tasks are introduced. This can be largely explained by the fact that weight-masking offers a much higher level of detail: activations-based masking takes out a complete neuron, whereas weights-based pruning acts on the level of a neuron's connections.

4 Conclusion

We propose a Dynamic Generative Memory approach for continual learning. Our results suggest that our approach successfully overcomes catastrophic forgetting by making use of a conditional generative adversarial model where the generator is used as a memory module through neural masking. We also show that the proposed dynamic memory expansion mechanism facilitates a resource efficient generator adaptation to successfully accommodate learning new tasks. We find that neural masking works significantly better when applied directly to layers' weights instead of activations.

References

[1] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars. Memory aware synapses: Learning what (not) to forget. *CoRR*, abs/1711.09601, 2017. URL `http://arxiv.org/abs/1711.09601`.

[2] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. *CoRR*, abs/1801.10112, 2018. URL `http://arxiv.org/abs/1801.10112`.

[3] 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.

[4] N. Kamra, U. Gupta, and Y. Liu. Deep generative dual memory network for continual learning. *arXiv preprint arXiv:1710.10368*, 2017.

[5] J. Kirkpatrick, R. Pascanu, N. C. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796, 2016. URL `http://arxiv.org/abs/1612.00796`.

[6] A. Mallya and S. Lazebnik. Piggyback: Adding multiple tasks to a single, fixed network by learning to mask. *arXiv preprint arXiv:1801.06519*, 2018.

[7] M. Mancini, E. Ricci, B. Caputo, and S. R. Bulò. Adding new tasks to a single network with weight trasformations using binary masks. *arXiv preprint arXiv:1805.11119*, 2018.

[8] M. Mayford, S. A. Siegelbaum, and E. R. Kandel. Synapses and memory storage. *Cold Spring Harbor perspectives in biology*, page a005751, 2012.

[9] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

[10] A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585*, 2016.

[11] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[12] S. Rebuffi, A. Kolesnikov, and C. H. Lampert. icarl: Incremental classifier and representation learning. *CoRR*, abs/1611.07725, 2016. URL `http://arxiv.org/abs/1611.07725`.

[13] A. Seff, A. Beatson, D. Suo, and H. Liu. Continual learning in generative adversarial nets. *arXiv preprint arXiv:1705.08395*, 2017.

[14] J. Serrà, D. Surís, M. Miron, and A. Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. *CoRR*, abs/1801.01423, 2018. URL `http://arxiv.org/abs/1801.01423`.

[15] H. Shin, J. K. Lee, J. Kim, and J. Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999, 2017.

[16] C. Wu, L. Herranz, X. Liu, Y. Wang, J. van de Weijer, and B. Raducanu. Memory Replay GANs: learning to generate images from new categories without forgetting. *ArXiv e-prints*, Sept. 2018.

[17] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, Z. Zhang, and Y. Fu. Incremental classifier learning with generative adversarial networks. *CoRR*, abs/1802.00853, 2018. URL `http://arxiv.org/abs/1802.00853`.

[18] F. Zenke, B. Poole, and S. Ganguli. Improved multitask learning through synaptic intelligence. *CoRR*, abs/1703.04200, 2017. URL `http://arxiv.org/abs/1703.04200`.