Oleksiy Ostapenko
576257

Proposal for Master's Thesis

Learning to remember: Dynamic generative memory for continual learning

October 22, 2018

1 Introduction

As opposed to normal training, in which data for all classes is available simultaneously, continual learning (CL) demands the ability to learn continually from a stream of data, in which new data batches are introduced incrementally over time.

Notation: adopting the notation of [2], let $D_k = \{(x_i^k, y_i^k)\}_{i=1}^{n^k}$ denote a collection of data belonging to the task $k \in \mathcal{K}$, where $x_i^k \in \mathcal{X}$ is the input data and $y_i^k \in y^k$ are the ground truth labels. While in the (standard) non-incremental multi-task setup the entire data $D = \cup_{k=0}^{|K|} D_k$ is available at once, in a task-incremental setup the incoming dataset D_k becomes available to the model continuously and specifically only during the learning of task k. Thereby, D_k can be composed of a collection of class items from the same task (can also contain a single class). Furthermore, during test time the output space covers all the labels observed so-far during the training, independently from the task they belong to: $\mathcal{Y}^k = \cup_{j=1}^{k} y^j$. Such an evaluation scenario is often referred to as a single-head evaluation [2, 16]. In CL setup at any time the model should be able to classify test samples from the tasks observed so-far.

Artificial Neural Networks (ANN) fail to learn incrementally about new classes, while maintaining good classification performance on the classes of the previously seen tasks $\mathcal{Y}^{k-1}$, not to mention reusing the knowledge about old tasks in a new context. Generally, there are two fundamental obstacles on the way to a continuously trainable AI system: the problem of forgetting the old knowledge when learning from new data (catastrophic forgetting) as well as lack of model scalability, e.g. inability to scale up the model size with continuously growing amount of data to learn from.

Catastrophic forgetting is believed to occur due to a lack of a plastic component in the neuron connections [8, 25]. In analogy with biological systems, neural or synaptic plasticity is the capability of a neuron or synapse to "lock" itself to a state or a connection, thus retaining previously attained knowledge. Indeed, it has been shown that this plasticity is responsible for maintaining previously acquired structure in the neo-cortical circuits of brains [3, 6].

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 equitable resource constraints, it is inevitable that with a growing number of tasks to learn, the model capacity is exhausted at some point in time. Without an adaptive strategy for network expansion, network size can swiftly go beyond existing hardware limits, or unnecessarily waste resources by parameter reservation.

The goal of this Master's thesis is to address the problem of catastrophic forgetting and lacking model scalability in the continual learning setup. We demand a classification system to be able to continuously learn about new tasks without storing raw samples of previously seen

data, whereas at any time the system should be able to classify test samples from the tasks observed so-far.

2 Related Work

Several recent approaches try to mitigate forgetting by simulating neuroplasticity in ANNs. To that end, one possibility is to identify critically important parameters sections of a network w.r.t. a given task, and imposing a penalty for changing them when learning another task [8, 2, 25, 1]. Common to these methods is that they seek to find a point in the model parameter space that minimizes the loss jointly for all previously learned tasks. However, the existence of such point is not always guaranteed, hence the hard attention to the task (HAT) mechanism was proposed [18]. HAT finds a parameter subspace for each task that can overlap with other tasks' parameter subspaces. The optimal solution is then found in the corresponding parameter subspace of each task. These solutions can be seen as reserving sub-spaces of the neural network's parameter space for each task at hand and driving the subsequent network updates into the free-capacity space.

Singe- vs. multi-head output: while the aforementioned methods successfully reduce the performance gap between incremental and non-incremental training, they all make use of the multi-head output evaluation. As can be seen in the Figure 1, in a multi-head evaluation setup each new task k has its own classification layer containing only the classes y^k belonging to the task k. Thus, at the test time of task k, the model has to pick the corresponding classification layer of this task. To accomplish this the task label k should be available at the test time. In other words, in such evaluation setup the task label k is assumed to be available at the test time in order to reduce the output space of the model $\mathcal{Y}^k = \cup_{j=1}^{k} y^j$ to the output space of the task y^k, to which the current test-sample belongs. If the current task only contains single class, e.g. $|y^k| = 1$, availability of the task label k directly infers the ground truth label leading to zero error of the model.

As opposed to a single-head evaluation, in which the model is evaluated on all classes seen so far during the training, multi-head evaluation simplifies the problem relying on the assumption that a separate task predicting model (TP) can predict he task affiliation k of a test batch D_{test} with a high accuracy. Indeed, such an assumption is legitimate in a non-incremental setup, where the data for all tasks $D = \cup_{t=0}^{|K|} D_k$ is available at once. In the continual learning setup the TP also needs to be trained incrementally and thus would also suffer from catastrophic forgetting. Consequently, a continuously trainable system can not rely on the assumption of the oracle knowledge of the task label at the test time and therefor should be evaluated in a single-head output setup. As shown in Figure 2 in a single-head evaluation the network has a single classification layer for all incrementally learned classes.

In order to address catastrophic forgetting in a continual learning setting with a single-head architecture some approaches rely on storing raw samples of previously seen data and making use of replay strategies during the training of subsequent tasks [16, 14]. However, this starkly contrasts with the natural learning mechanisms of the brain, which does not feature the retrieval of raw information identical to originally exposed impressions [12]. What is more, storing raw samples of previous data may violate data privacy and memory restrictions in the real world applications [22].

Generative Memory: instead of storing raw samples, we propose to rely on the idea of employing deep generative model for memorizing previously seen data distributions. [19, 21] follow a similar idea relying on generative replay, which requires retraining the generator from scratch at each time step on a mixture of synthesized images of previous classes and new real samples. Apart from being inefficient for training, it is severely prone to "semantic drifting". Namely, the quality of data generated at every memory replay point highly depends on the images generated during previous replays, which can result in loss of quality over time.

Dynamic network expansion: only few existing works explore ANNs with dynamically extendable capacity. [26] proposes to dynamically add new neurons to an auto-encoder for groups of difficult examples that yield in high loss. Another line of work explores dynamic adjustment of networks structure in the context of network compression and pruning [5, 4, 20, 24] - this involves dynamically removing units rather then adding new once. [23] considers network expansion in a continual learning setting, relying on a process called "selective-retraining". This involves diverse heuristics and hyperparameters in order to identify units that causes semantic drift, which are then duplicated and the network is further retrained.

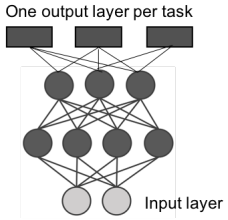

Figure 1: Multi-head output architecture with task specific output (classification) layers. Each output layer contains only classes of the corresponding task.

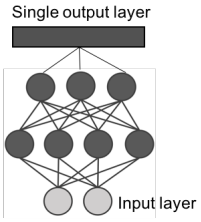

Figure 2: Single-head output architecture with single output layer for all classes seen so-far in the training independently from which task they belong to.

3 Dynamic generative memory network with learnable plasticity

In contrast to the methods described above, we propose to utilize a single generator that is able to incrementally learn new information during the normal adversarial training without the need to replay previous knowledge. Simultaneous with the adversarial training on each incoming data batch, we learn a sparse mask for the layer activations of the generator network. This serves the purpose of encouraging parameter re-usability across tasks as well as preventing important parameters for past tasks from changing when learning subsequent tasks. The values of the learned mask correspond to the plasticity of the connections between the layers. Thus units with low plasticity (values of mask ~ 1) are reserved for previous tasks and can be reused but not changed during the subsequent training. We make use of the learned binary mask as an efficient mechanism for indicating the necessary amount of additional capacity to be added after learning a task k. We propose to add exactly the amount of neurons after learning the task k that were blocked for this task , in this way keeping the amount of "free" capacity constant for each task ("free" capacity are the neurons that are not blocked for any of the previous tasks).

As visualized in the Figure 3, the proposed method is composed of the following components:

- A discriminator network D_{θ^D} with two output layers: multi-class classifier $L_{aux.}$ and binary classifier $L_{adv.}$. Here θ^D are the parameters of the discriminator network.
- A generator network G_{θ^G} with binary masks M^k learned for each data batch D_k seen so far in the training. Here θ^G are the parameters of the generator network. We utilize techniques proposed by [18] and [11] to learn the binary mask from a real valued embedding vector. The values of the learned mask correspond to the plasticity of the connections. Thus, units with low plasticity (masked with values 1) are reserved for previous tasks and can be reused but not changed during the subsequent training. Free parameters (masked with 0) can be changed during subsequent training.
- G_{θ^G} can generate samples of previously seen classes, e.g.: $X_{k-1,synt.} = G(\theta^G, M^{k-1})$. We utilize multi-head architecture in the generator with one output layer per each seen class.

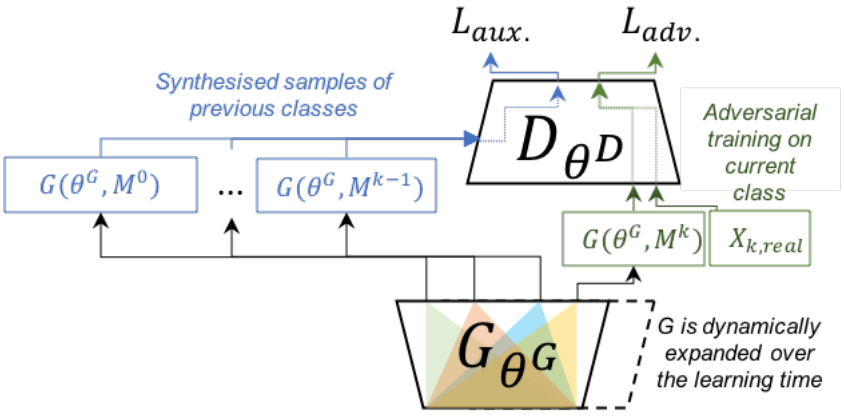

Figure 3: Deep generative memory: The AC-GAN [15] architecture allows simultaneous adversarial and auxiliary training, in which auxiliary output of $L_{aux.}$ is trained on the real samples of the current task $X_{k,real}$ and synthesized sample of previously seen tasks $X_{k-1,synt.} = G(\theta^G, M^{k-1})$. During the adversarial training with real and fake samples of current class k a connections' plasticity in the generator is learned simultaneously with the weights matrices. Connections plasticity is represented by task specific, learnable binary masks. Generator is dinamically expanded over time in a way that keeos number of free neurons constant for each task.

- G_{θ^G} is adversarially trained to generate samples from the data distribution of the current batch D_k using real samples $X_{k,real}$ and fake samples $X_{k,fake} = G(\theta^G, M^k)$
- In order to unsure scalability of the method with the increasing amount of data to learn from, the G_{θ^G} is dynamically expanded at every time step in a way that keeps the number of free parameters (masked with 0) constant for every task.

We utilize the AC-GAN architecture [15] that has auxilary and adversarial outputs. This allows training of the final classification model simultaneously with the adversarial training.

4 Experiments

We define a strictly incremental setup as a setup in which every new task contains a single class and no raw samples of previous tasks can be stored. We perform experiments measuring the classification accuracy of our system in a strictly incremental setup with single-head evaluation on three benchmark datasets: MNIST [10], SVHN [13] and CIFAR-10 [9]. Accuracy after incrementally many learned tasks is reported to provide a high level impression on the criticality of the catastrophic forgetting problem: ideally a system that does not forget will see results that are stable, or even improving as it learns to generalize better from previously seen tasks.

All datasets are used to train a classification network in the strictly incremental setup, and the performance of our method is evaluated quantitatively through comparison with benchmark methods. Note that we compare to both type of methods; (i) where the strictly incremental constraints are relaxed (every task contains more then one class), [16], [8], [2], [25], (ii) methods that strictly adhere to the setup (single class per task), making use of a type of memory: [17], [19], [21]. Further, we plan to present a qualitative evaluation of CIFAR-10 using a perceptual metric (FID score) [7].

Datasets: The MNIST and SVHN datasets are composed of 60000 and 99289 images respectively, containing digits. The main difference is in the complexity and variance of the data used. SVHN's images are cropped photos containing house numbers and as such present varying viewpoints, illuminations, etc. All images are further resized to 32 x 32 before use. Finally,

CIFAR10 contains 60000 32×32 labelled images of objects (e.g. planes, cars, dogs etc.), split in 10 classes, roughly 6k images per class.

5 Conclusion

The contributions of this Master's thesis are two-fold: **(a)** we propose a dynamic generative memory (DGM) endowed with learnable binary masks for layer activations, mitigating forgetting and rendering storing and memory replay unnecessary; **(b)** an adaptive network expansion mechanism, facilitating resource efficient continual learning. We study and discuss the impact and effectiveness of the proposed mask and network extension modules, and evaluate the proposed method in the incremental learning scenario. The goal is to reach state of the art performance on benchmark datasets.

References

[1] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. *CoRR*, abs/1711.09601, 2017. URL `http://arxiv.org/abs/1711.09601`.

[2] Arslan Chaudhry, Puneet Kumar Dokania, Thalaiyasingam Ajanthan, and Philip 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] Joseph Cichon and Wen-Biao Gan. Branch-specific dendritic ca2+ spikes cause persistent synaptic plasticity. *Nature*, 520(7546):180–185, 2015.

[4] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR*, abs/1510.00149, 2015. URL `http://arxiv.org/abs/1510.00149`.

[5] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. *CoRR*, abs/1506.02626, 2015. URL `http://arxiv.org/abs/1506.02626`.

[6] Akiko Hayashi-Takagi, Sho Yagishita, Mayumi Nakamura, Fukutoshi Shirai, Yi I Wu, Amanda L Loshbaugh, Brian Kuhlman, Klaus M Hahn, and Haruo Kasai. Labelling and optical erasure of synaptic memory traces in the motor cortex. *Nature*, 525(7569):333, 2015.

[7] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a nash equilibrium. *arXiv preprint arXiv:1706.08500*, 2017.

[8] James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796, 2016. URL `http://arxiv.org/abs/1612.00796`.

[9] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2014.

[10] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

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

[12] Mark Mayford, Steven A Siegelbaum, and Eric R Kandel. Synapses and memory storage. *Cold Spring Harbor perspectives in biology*, page a005751, 2012.

[13] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew 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.

[14] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

[15] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585*, 2016.

[16] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. *CoRR*, abs/1611.07725, 2016. URL `http://arxiv.org/abs/1611.07725`.

[17] Ari Seff, Alex Beatson, Daniel Suo, and Han Liu. Continual learning in generative adversarial nets. *arXiv preprint arXiv:1705.08395*, 2017.

[18] Joan Serrà, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. *CoRR*, abs/1801.01423, 2018. URL `http://arxiv.org/abs/1801.01423`.

[19] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999, 2017.

[20] Suraj Srinivas and R. Venkatesh Babu. Data-free parameter pruning for deep neural networks. *CoRR*, abs/1507.06149, 2015. URL `http://arxiv.org/abs/1507.06149`.

[21] 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*, September 2018.

[22] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, Zhengyou Zhang, and Yun Fu. Incremental classifier learning with generative adversarial networks. *CoRR*, abs/1802.00853, 2018. URL `http://arxiv.org/abs/1802.00853`.

[23] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. 2018.

[24] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I. Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S. Davis. NISP: pruning networks using neuron importance score propagation. *CoRR*, abs/1711.05908, 2018. URL `http://arxiv.org/abs/1711.05908`.

[25] Friedemann Zenke, Ben Poole, and Surya Ganguli. Improved multitask learning through synaptic intelligence. *CoRR*, abs/1703.04200, 2017. URL `http://arxiv.org/abs/1703.04200`.

[26] Guanyu Zhou, Kihyuk Sohn, and Honglak Lee. Online incremental feature learning with denoising autoencoders. In *Artificial intelligence and statistics*, pages 1453–1461, 2012.

Appendices

A: Preliminary results

Method	$A_{10}(\%)$
EWS [8]	55.8
PI [25]	57.6
iCarl-S [16]	55.8
EWS-S[8]	79.7
RWalk-S[2]	82.5
PI-S [25]	78.7
DGM (ours)	**94.4**
DGM-strict (ours)	**93.9**

(a)

Method	$A_{10}(\%)$
JT (upper bound, non incremental) [21]	96.9
EWC-M [17]	77.3
DGR-M [19]	85.4
MeRGAN-M [21]	97.0
DGM-strict (ours)	**98.14**

(b)

Table 1: comparison with different baselines on split MNIST benchmark after incrementally learning 10 tasks (10 digits 0-9). **(a)**: comparison to benchmark and architecture presented by [2] in which classes are added in pairs of 2, as well performance of DGM-strict in strict split MNIST benchmark (classes added by one), where only our method is capable of learning in a setup where classes are introduced one by one. Baselines where raw samples or previous data are stored and reused for further training are appended with "-S". **(b)**: comparison to benchmark and architecture utilized by Wu et al. [21]. All approaches make use of generative memory and are evaluated in a strict incremental setup. DGM togather with MeRGAN [21] reach the upper bound performance of the joint training (JT)