PhD Course CL Report

Filippo Betello, Antonio Purificato

June 2024

1 Introduction

The implemented approach consists in the use of a Generative Replay Buffer for class-incremental continual learning, as proposed by Shin et al. (2017). We train a Generative Adversarial Networks (GAN) Goodfellow et al. (2014) to mimic past data. Generated data are then paired with corresponding response from the past task solver to represent old tasks. The selected GAN is composed by a Generator with 4 layers and a Discriminator with 4 layers, and is trained in an adversarial manner, as usually done in GANs.

2 Hyperparameter Tuning

After some hyperparameters tuning, the parameters which provided the best results are presented in Table 1.

Hyperparameter	Value
Classifier Learning Rate	10^{-3}
Generator Learning Rate	3×10^{-4}
Discriminator Learning Rate	3×10^{-4}
GAN Training Epochs	12
Solver Training Epochs	10
Batch Size	128

Table 1: Hyperparameters for training

The goal of hyperparameter tuning was to maximize the accuracy, avoiding to let the number of parameters of the architecture explode. Another hard constraint that we had to consider was the GPU usage for Colab, since it is limited for each day.

3 Number of parameters

With the desired choice of the hyperparameters, the number of total parameters of the two networks (the generator and the solver) is close to 1.2×10^6, where the majority of parameters is given by the generator network. The solver is the part of the architecture with the lowest number of parameters, close to 5×10^5, an order of magnitude less than the other two networks.

The number of parameters required by the first approach is much lower, since only the ones of the solver architecture are considered. However, the number of parameters does not take into account the impact of storing original training samples in the replay buffer.

4 Computational Complexity

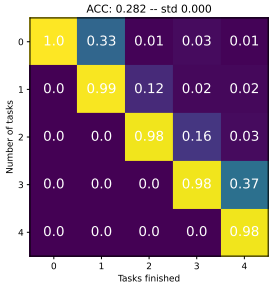
The memory complexity of a generative replay buffer approach is primarily determined by the size of the generative model used to generate the synthetic data samples. This is because the buffer needs to store one generative model per task or a fixed number of generative models that can generate samples for all tasks. In contrast, a traditional replay buffer that stores original data samples would have a memory complexity that is quadratic in the number of tasks, as it needs to store an increasing number of data samples as more tasks are learned. However, it should be noticed that the GAN plays an important role in this approach. The size of the architecture should not increase too much, otherwise the method will be influenced by the computational cost of the generative architecture.

5 Results

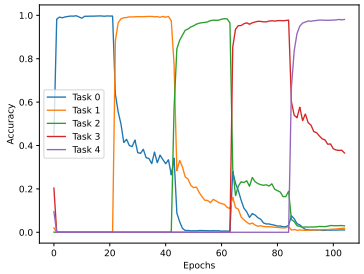
The proposed approach was tested on the MNIST dataset in a class-incremental setting with 10 classes in total and 2 classes for each task. The results can be seen in Figure 1. In Table 2 can be seen the results in terms of forward and backward transfer and also number of parameters.

Method	Forward Transfer	Backward Transfer	#Parameters
Standard RB	-0.042	-0.132	59210
Generative RB	-0.082	-0.882	1210331

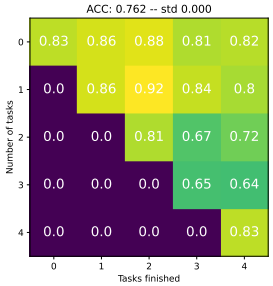
Table 2: Results in terms of forward transfer, backward transfer and number of parameters.



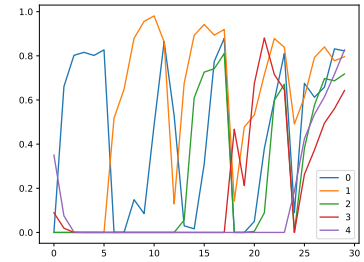
(a) Confusion Matrix - Generative RB.



(b) Accuracy of Generative RB.



(c) Confusion Matrix - Standard RB.



(d) Accuracy of Standard RB.

Figure 1: Confusion Matrix and accuracy values over time for each task in a class-incremental scenario with 5 tasks.

6 Limitations and Possible Improvements

The main downside of the proposed approach is that the generator architecture is not created ad-hoc for this task. Probably, the creation of the GAN with a dedicated structure for continual learning would improve the performance in terms of accuracy, forward and backward transfer and also reduce the computational cost, avoiding not useful operations.

One possible solution to improve the efficiency in terms of memory could consist in the development of an "early-exit technique" Scardapane et al. (2020) in the generation part. We trained the GAN for 12 epochs, but probably this network is able to generates sample of acceptable quality after less than 12 epochs. For this reason, one idea could be to exit after the quality of the generated samples for the desired class satisfies a certain property. This will transform the GAN training in a dynamic training.

References

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks.

Scardapane, S., Scarpiniti, M., Baccarelli, E., and Uncini, A. (2020). Why should we add early exits to neural networks? *Cognitive Computation*, 12(5):954–966.

Shin, H., Lee, J. K., Kim, J., and Kim, J. (2017). Continual learning with deep generative replay. *Advances in neural information processing systems*, 30.