# Deep generative view on continual learning - Assignment

Daniel Mauricio Jimenez Gutierrez, Francesco Verdini,
Marco Ferilli, Gianluca De Carlo and Gaetano Saurio

This file provides a solution for the proposed assignment of the Deep generative view on continual learning Ph.D. course. The assignment aims to create a generative replay method for continual learning under the class-incremental scenario [1]. The code can be found in the correspondent Github repository.

## 1 Question 1

*Measure the memory and (estimate) computational requirements for generative and vanilla replay with raw samples. Is it comparable in memory requirements?*

**Answer:** As demonstrated in the code implementation, the memory required for the VAE replay was 1.031 GB. On the other hand, the vanilla replay needed 0.993 GB. The difference is that vanilla replay involves storing raw data samples from previous tasks. In contrast, VAE replay stores the generated replay data and needs memory to train the VAE model. Thus, the approaches are comparable regarding memory requirements since the vanilla required just a bit less memory. They become comparable since the number of generated samples per task remained constant between both methods [2].

## 2 Question 2

*How would that solution scale with n tasks? Is it linear, quadratic, or something else?* **Answer:** Let us analyze the scalability of the solution from the memory usage and computation time separately:

1. **Memory usage**: The solution depends on the following aspects:
   (a) *Training VAE*: The complexity here is $\mathcal{O}(T)$ since each VAE training session takes a constant amount of time (fixing the $E$).
   (b) *Replaying and storing data while training*: The complexity here is $\mathcal{O}(T \cdot N)$ since the replay data from all previous tasks increases linearly with the number of tasks.

   Therefore, the **memory usage** scales **linearly** on the amount of $\mathcal{O}(T \cdot N)$.

2. **Computation time**: The solution depends on the following aspects:
   (a) *Training VAE*: The complexity here is $\mathcal{O}(T)$ since each VAE training session takes a constant amount of time (fixing the $E$).
   (b) *Replaying and storing data*: The complexity here is $\mathcal{O}(T \cdot N)$ since the replay data from all previous tasks increases linearly with the number of tasks.

(c) *Training classifier*: It is proportional to the amount of data. Considering $E$ for each $T$, the complexity is $\mathcal{O}(T \cdot E \cdot N)$. Then, summing this over all the tasks, we get that the complexity is $\mathcal{O}(T^2 \cdot N)$.

Therefore, the **computation time** scales **quadratically** on the amount of $\mathcal{O}(T^2 \cdot N)$.

In the previous, $T$ represents the number of tasks, $N$ the number of generated samples, and $E$ the number of epochs.

# 3 Question 3

*What are the downsides of your approach?.* **Answer:** The downsides of our approach are:

1. *VAE performance dependently*: The accuracy of the classifier depends directly on the quality of the images generated from the VAE. If the VAE performs poorly, the classification process will be suboptimal.

2. *Increased computation time*: Our solution scales quadratically when increasing the number of tasks. It can generate a bottleneck under systems since it requires longer training times.

3. *Overfitting*: The classifier may overfit to the generated data, leading to suboptimal performance on real-world data.

# 4 Question 4

*What are your ideas for making this solution more memory and computationally efficient?* Some ideas that may improve the solution are:

1. *Alternative generative model*: Using VAE or lighter GANs may help to decrease the computation time needed for the training process.

2. *Reduced sample generation*: Reducing the number of samples generated for each task leads to a more memory-efficient solution.

3. *Data reduction*: Using summarizing or dimensionality reduction techniques (i.e., quantization, PCA, autoencoders) over the generated data stored may help to reduce the memory needed in the training process.

4. *Federated Learning*: Instead of training and generating the samples (with the VAE) in one device, it would be beneficial to distribute the training of the VAEs in multiple nodes and then share the weights generated, aggregate them, and employ them in each desired device.

| Hyperparameter | Values |
|---|---|
| Learning rate classifier | 0.01, **0.001** |
| Learning VAE | 0.02, **0.002**, 0.0002, 0.00002 |
| Number epochs classifier | **2**, 10 |
| Number epochs VAE | 2, **15**, 50 |
| Latent dimensions | **100**, 500 |

Table 1: Hyperparameter's ranges employed for fine-tuning.

# 5 Question 5

*Justifying the choice of hyperparameters.* **Answer:** The **number of generated samples** per task was set equal to the number of samples of each task to keep it comparable to the vanilla replay. The remaining **hyperparameters** were fine-tuned [3] with a random grid search based on the ranges depicted in Table 1.

The final selected hyperparameters providing the best results (i.e., accuracy, backward transfer, forward transfer, and computational time) [4, 5] can be found as **bold** text in Table 1.

# References


[1] Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

[2] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.

[3] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65, 2018.

[4] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

[5] Natalia Díaz-Rodríguez, Vincenzo Lomonaco, David Filliat, and Davide Maltoni. Don't forget, there is more than forgetting: new metrics for continual learning. *arXiv preprint arXiv:1810.13166*, 2018.