Report for Deep Generative View on Continual Learning

PhD. in Data Science 23/24

Francesco Paolo Nerini

June 14, 2024

Solution adopted: generative replay using a VAE to generate samples for a MLP trained on MNIST

Hyperparameters selection Most of the hyperparameters were chosen by using default or standardised choices. For the VAE, I adapted the implementation from `https://github.com/lyeoni/pytorch-mnist-VAE`, where they used a learning rate scheduler. With respect to the original implementation, I increased the number of training epochs in order to obtain a better fit of the model. The rest of the code has been adapted from the exercises of the course. The VAE loss function computes a binary cross entropy between couple of original and reconstructed images; this required for the original images to have features scaled between 0 and 1. Finally, the choice of the VAE was due to the high difficulty in training an effective GAN on the training set of the first task: the GAN proved to be extremely sensible to hyperparameter selection, making it harder for being used.

Problem 1. Measure the memory and (estimate) computational requirements for generative and vanilla replay with raw samples. Is it comparable in memory requirements?

In terms of model size efficiency, the two methods are equivalent: in both cases the model involved does not increase in size when training on a new task (model size efficiency of 1). However, in terms of sample storage size, the generative replay has the strong advantage of requiring no storage at all, since new samples are generated on the fly. Meanwhile, the vanilla replay method will have a lower sample storage size efficiency as we increase the samples per class parameter (in my case, with 300 samples per class, it was about 0.4) This, in turns, requires a larger raw model memory, since we must also take into account the generative model memory, and higher computational requirements, as more operations are required for each pass. In the solution I adopted, the VAE required around 4.5 times more operation for a pass than the MLP, the common model for both replay methods (about 203776 MACs vs. 44700 MACs). This reflected on longer training times.

Problem 2. How would that solution scale with n tasks? Is it linear, quadratic, or something else?

The generative replay solution I found would need to scale the memory size of final layer of the MLP linearly with the total number of tasks; however, the VAE model would not require to scale as the number of tasks increases. The training at each epoch of the MLP and the VAE would also cause a linear scaling the computational requirements of the solution.

Problem 3. What are the downsides of your approach?

The performance of the generative replay method I adopted are good but not on par with the results obtained through a vanilla replay (Vanilla: ACC= 0.87, BWT= 0.10, FWT= 0.0; VAE replay: ACC= 0.79, BWT= -0.23, FWT= -0.13). More tuning of the parameters may be required, but this would probably worsen the times required for training, which are already longer than the vanilla replay method.

Problem 4. What are your ideas for making this solution more memory- and computationally-efficient?

The key to make the solution more efficient would be to check the generative method employed by choosing a model which use less resources overall. Or, by combining aspects of the vanilla replay method, one could store few, carefully chosen prototypes of each class and a trained generative model which adds small random noise to each sample. In this way, we could keep the sample storage size efficiency high while using a simpler model to perform data augmentation.