Homework CL

Alessandro Baiocchi

June 2024

1 Introduction

The model implemented is a simple MLP for the classification of the MNIST dataset. For generative replay a standard VAE is used. When a new task is added to the problem both the VAE and the classifier are trained using samples both from the new dataset and some replay samples generated from the VAE. The number of replay samples to be generated is incremented each task, the size of each replay set is fixed by a hyper parameter.

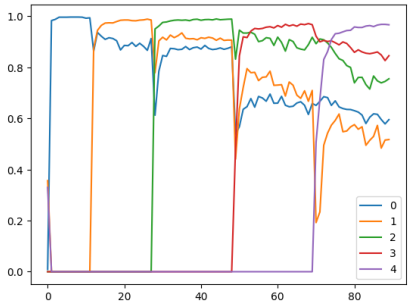

Figure 1: Accuracy for the various tasks during the training.

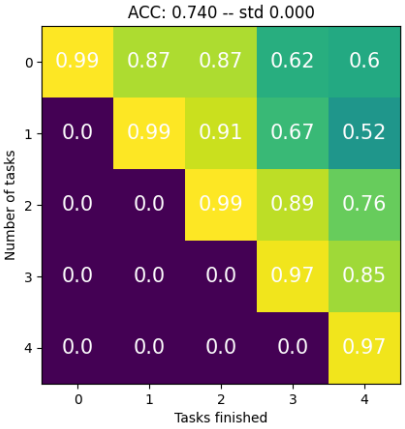

Figure 2: Final accuracy for all the tasks in function of tasks finished.

The results obtained by the model on MNIST split in 5 tasks are shown in figure 1 and 2.

2 Hyper parameters

The model hyper parameters were chosen by manual investigation. Both the number of epochs of the classifier and the VAE training were chosen by looking

at the loss during training, in such a way that the model would train sufficiently. The only peculiar hyper parameter is the number of samples to be generated. In a standard approach the number of samples generated each task would increase linearly with the number of tasks. After some investigation i chose to fix the number of samples manually for each task obtaining slightly better results.

3 Memory requirements

Adding the VAE to the MLP adds considerable weight to the model, as shown in the following table.

	#Params	Weight (MB)
MLP	44860	0.17
VAE	1068820	4.08
Total	1113680	4.25

This considerable increase in weight is due to the fact that the VAE model is probably over-complicated for the task at hand, and outsizes by more than 20 times the MLP used for classification. On the other hand the images are really light (0.78 KB each), so storing a buffer of, for instance, 5000 images only amounts to 3.9 MB. So in terms of memory our model is roughly equivalent to a buffer of 5000 images.

The main advantage of this method is that the size of the model does not scale if we want a larger buffer, and is probably capable of facing more difficult tasks without enlarging the VAE model.

3.1 Task scaling

Since we generate more samples depending on the number of completed tasks, the size of the dataset we train our model on scales as:

$$S = S_0 + T \cdot S_t \tag{1}$$

Where S_0 is the size of the original dataset for the task, T is the number of completed tasks and S_t is the number of samples per task to be generated.

This means that the size of the dataset scales linearly with the number of tasks, so the total complexity of the training scales quadratically with the number of tasks.

This is assuming the number of samples to be added for each task is fixed, in my implementation i hand picked values of this hyperparameter for each task, as i observed a better behaviour when the scaling of the replay dataset was less-than-linear. This change leads to a less-than-quadratic scaling, but still more than linear.

4 Downsides

The main downside of this method is the fact that adding a generative model to the classifier increases significantly the training time. In fact, the VAE takes the vast majority of the training time in these experiments, while saving samples in a buffer would completely avoid this problem.

Another downside is that it adds a trainable model to the pipeline, being another model that could fail and affect the whole training process. If the VAE training should collapse, the training of the classifier would get affected too.

5 Improvements

Some possible improvements are:

- Using a generative model as simple as possible, in order to reduce the overhead on the base model.
- Applying a learning rate scheduling to reduce forgetting.
- Further investigate the number of generated samples per task.