Homework for PhD course: Deep Generative View on Continual Learning

Alessio Devoto

June 2024

1 Introduction

The model implemented is a simple Multi-Layer Perceptron (MLP) for the classification of the MNIST dataset. A standard Variational Autoencoder (VAE) is used for generative replay. When a new task is introduced, both the VAE and the classifier are trained using samples from the new dataset and replay samples generated by the VAE. The number of replay samples generated increases with each task, with the size of each replay set determined by a hyperparameter.

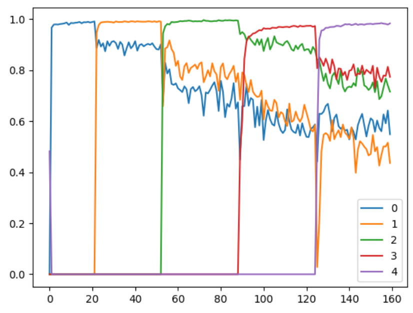

Figure 1: Accuracy for various tasks during training.

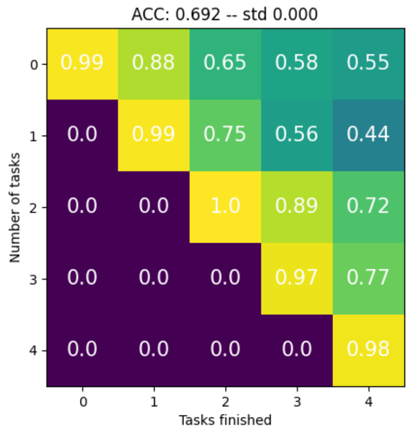

Figure 2: Final accuracy for all tasks as a function of tasks completed.

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

2 Hyperparameters

The model hyperparameters were chosen through manual investigation. Both the number of epochs for training the classifier and the VAE were selected based on the training loss, ensuring the model trained sufficiently. The primary hyperparameter of interest is the number of samples to be generated. Typically, the number of generated samples increases linearly with the number of tasks. However, after some investigation, I chose to manually fix the number of samples for each task, achieving slightly better results.

3 Memory Requirements

Adding the VAE to the MLP significantly increases the model's size, as shown in Table 1.

	# Parameters	Weight (MB)
MLP	44,860	0.17
VAE	1,068,820	4.08
Total	1,113,680	4.25

Table 1: Model parameters and weight

This substantial increase is due to the VAE being over-complicated for the task at hand, outweighing the MLP by more than 20 times. However, the images are quite small (0.78 KB each), so storing a buffer of 5,000 images only amounts to 3.9 MB. Therefore, in terms of memory, our model is roughly equivalent to a buffer of 5,000 images.

The main advantage of this method is that the model size does not scale with buffer size, and it is likely capable of handling more complex 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 generated per task.

This implies that the dataset size scales linearly with the number of tasks, and thus the total complexity of the training scales quadratically with the number of tasks.

In my implementation, I manually adjusted this hyperparameter for each task, observing better performance with a less-than-linear scaling of the replay dataset. This adjustment leads to less-than-quadratic scaling, but still more than linear.

4 Downsides

The primary downside of this method is that adding a generative model to the classifier significantly increases the training time. The VAE requires the majority of the training time in these experiments, whereas saving samples in a buffer would avoid this issue entirely.

Another downside is the addition of a trainable model to the pipeline, which introduces another potential point of failure. If the VAE training collapses, it would adversely affect the classifier training.

5 Improvements

Some possible improvements include:

- Using a simpler generative model to reduce overhead on the base model.
- Applying learning rate scheduling to mitigate forgetting.
- Further investigating the number of generated samples per task.