Reduced Memory size A-GEM With CGAN

Yihong Wang, Yuanrui Huang
Department of Electrical and Computer Engineering
New York University, Tandon of Engineering
New York,11201
yw3408@nyu.edu, yh2910@nyu.edu

Abstract

Continue learning has always been a challenge in both the industrial and academic worlds because of catastrophic forgetting. The model tends to perform worse on previous tasks while training on a new task. Several approaches have been provided to ease the catastrophic forgetting. In this work, we introduce a method called A-GEM(Averaged GEM) with CGAN(Conditional Generative Adversarial Networks), in which we investigate the efficiency and memory cost of this method. [1]

1 Introduction

Continue Learning(CL), also named Lifelong Learning(LLL), is a human-like learner, which is required to train on a sequence of tasks. However, traditional approaches for model training could suffer from catastrophic forgetting (French, 1999), which means the artificial neural network lost the knowledge related to the previous tasks when it finished the learning of current tasks. For alleviating the catastrophic forgetting, several training approaches have been proposed. In general, we could classify these approaches into three classes.

The first kind of method regulates intrinsic levels of synaptic plasticity to protect consolidated knowledge, like (Elastic Weight Consolidation) (Kirkpatrick et al., 2016). The second kind of method builds a relatively complementary learning system for memory consolidation and experience replay, like GEM, which uses the episodic memory to search idea gradient that brings loss descent among tasks. The third kind of method considers the capability of learning for a fixed-size model, leaner allocate additional resources(typically model size) to learn new information, like PNN(Progressive Neural Networks) (Rusu et al., 2016).

In this work, we explore the limitation of A-GEM in its fixed-size of memory and utilize CGAN to compensate A-GEM's intensive memory cost while maintaining the performance of A-GEM. Data generated from the CGAN would serve as input to the model and calculate the average episodic memory gradient for previous tasks. Therefore, during the training phase, there's no need to store a subset of training data from the previous task and minimize this part of memory cost to batch-size level.

2 Related Work

2.1 Average Gradient Episodic Memory

In 2017, Gradient Episodic Memory (GEM) (Lopez-Paz and Ranzato, 2017), a method that alleviates the model's forgetting, was proposed. GEM maintains episodic memory$\mathcal{M}_k$ by storing the subset of training data of task t. Assuming the lifelong learner was allocated M memory units and total T tasks

[1] Code Implementation: https://github.com/ElectronicTomato/continue_leanrning_agem

need to be learned, the leaner would allocate $m = M/T$ memories units for each task. If the total task number T is unknown, the leaner need to reduce the value of m with the growth of T (Rebuffi et al., 2017). For the GEM model, we use f_θ to represent the predictor parameterized by $\theta \in \mathbb{R}^p$, and denote the loss at the memories from the k-th task as

$$\ell\left(f_\theta, \mathcal{M}_k\right) = \frac{1}{|\mathcal{M}_k|} \sum_{(x_i,k,y_i)\in\mathcal{M}_k} \ell\left(f_\theta\left(x_i, k\right), y_i\right) \tag{1}$$

Therefore, we could use follow objective function to represent GEM, with $\mathcal{D}_t$ represents the training data of task t:

$$minimize_\theta \quad \ell(f_\theta, \mathcal{D}_t) \quad \text{s.t.} \quad \ell(f_\theta, \mathcal{M}_k) \leq \ell(f_\theta^{t-1}, \mathcal{M}_k) \qquad \forall k < t \tag{2}$$

where f_θ^{t-1} represents the predictor f_θ trained till task t-1. By compute the loss of model, GEM could get loss gradient vectors of previous tasks g_k , loss gradient vectors of current tasks g,and the angle between previous gradient vectors and current gradient vector. When the angle is greater than 90° with any of the $g_k's$, GEM projects the current gradient to the closets in L2 norm gradient $\tilde{g}$ which makes sure that the angle wouldn't be larger than 90°. Therefore the optimization problem GEM solves is given by:

$$minimize_{\tilde{g}} \quad \frac{1}{2}||g - \tilde{g}||_2^2 \quad \text{s.t.} \quad \langle \tilde{g}, g_k \rangle \geq 0 \quad \forall k < t \tag{3}$$

However, GEM suffers from low efficiency as every new task brings an increasingly growing memory cost and computation burden. To improve the Continue Learning computation's efficiency, a new method, Averaged gradient episodic memory(A-GEM) (Chaudhry et al., 2018) is proposed. Rather than makes sure the angle of all gradient vectors is not large than 90°, A-GEM use the average gradient to represent all previous tasks' loss gradient vectors. Therefore, A-GEM has an objective function as:

$$minimize_\theta \quad \ell\left(f_\theta, \mathcal{D}_t\right) \quad s.t. \quad \ell\left(f_\theta, \mathcal{M}\right) \leq \ell\left(f_\theta^{t-1}, \mathcal{M}\right) \quad where \mathcal{M} = \cup_{k<t}\mathcal{M}_k \tag{4}$$

The corresponding optimization problems with gradient vectors is:

$$minimize_{\tilde{g}} \quad \frac{1}{2}\|g - \tilde{g}\|_2^2 \quad s.t. \quad \tilde{g}^\top g_{ref} \geq 0 \tag{5}$$

where g_{ref} is a gradient computed using a batch randomly sampled from the episodic memory of all the past tasks. We could think that g_{ref} is the average of the gradients from the previous tasks. Once the constraint is violated, the gradient of the current task is projected via:

$$\tilde{g} = g - \frac{g^\top g_{ref}}{g_{ref}^\top g_{ref}} g_{ref} \tag{6}$$

2.2 Conditional Generative Adversarial Nets(CGAN)

Generative adversarial nets(GAN) (Goodfellow et al., 2014) has shown the robust capability to generate data that has similar probability distribution to original data. It consists of two 'adversarial' models: a generative model G that simulates the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The loss function of GAN could be represented as:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{7}$$

After that, Conditional Generative Adversarial Nets(CGAN) (Mirza and Osindero, 2014), shows how the generator and discriminator work with extra information y. In our study, we assume it represents the class of training data. And we could get objective function as:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x|y)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z|y)))] \tag{8}$$

3 A-GEM with CGAN

Based on the experiment result of A-GEM (Chaudhry et al., 2018), we could see that the A-GEM has similar performance with GEM and also less computation. However, it still suffers from the limitation of memory size. With fixed-length of memory and increasingly growing task number, the model will inevitably experience performance loss. Because the mechanism of A-GEM recalls episodic memory by a subset of training set, and each insertion of new task correspond to less memory space per previous task and fewer data to represent the probability distribution of previous tasks and therefore average gradient of previous tasks can be less representative or even wrongly representative. Ideally, in the A-GEM experiments, each stored dataset ought to be a miniature of the previous task. The abnormal samples in the training set enable model better generalization during the training stage; however, if the same anomalous data appears in the episodic memory stage, it could deviate the average gradient from the optimum direction.

In this work, we propose a method called A-GEM with CGAN, which aims to optimize A-GEM's memory problem. This method utilizes CGAN to learn the probability distribution of the task's training data and replace A-GEM's stored datasets with CGAN's imitated data to calculate the previous tasks' loss gradient. There are two advantages to this approach. First, we lower the memory utilization of A-GEM. In particular, every training phase when the leaner require the previous tasks' data, it could use the CGAN to supply data. Once the learner finishes updating, it neither needs to save data from CGAN nor the previous training subset. Second, CGAN is capable of capturing the main features of each task and reproduce these features. So even for the task with some abnormal training samples, CGAN could still generate data with the main feature and reduce the noise's influence. Since anomalous data may again present while training current task, so, the model can still acquire generalization to abnormal data to some extend.

The system structure shows as Figure 1:

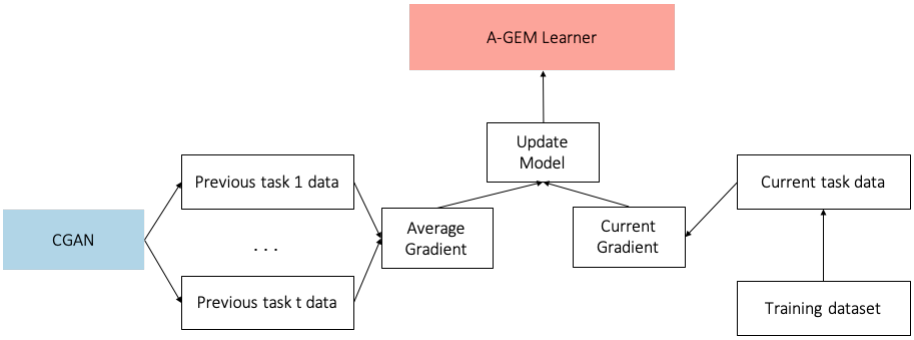

Figure 1: A-GEM with CGAN working structure

4 Experiments

In terms of architecture, for the CGAN training, we adopt a 2-layer CNN network with LeakyReLU and MaxPool as discriminator and 3-layer CNN network with batch normalization and ReLU as the generator. As for A-GEM learner, we use a 3-layer fully-connected Network as A-GEM learner which would be used for report the average accuracy. Due to the computational cost and time limitation, we measure the performance of the model on the magnitude of 10 tasks and therefore setting the output dimension of A-GEM learner to 100. The total number of parameters on A-GEM learner is 2126100.

At first, we train the learner with A-GEM training method, which is implemented with fully-connected Network and episodic memory(4000 data as fixed-memory size). And we control the leaner' learning epochs as 1 and 10. In this way, we get two standard A-GEM learners with different study levels.

Then we train several new A-GEM learners with CGAN. The leaners all have only 1 epoch learning time for each new task. We take the output of CGAN as the backtracking data for the learner to compute the average gradients of the previous tasks. In this way, there is no need for leaner to maintain episodic memory. From the second task on, we keep generating batch-size of input data(in our experiment batch size is 128) for each previous task, and load into A-GEM learner to acquire the

gradient of correspond task, and calculate the average gradient until each the tasks reach 1280 data. In this way, we ensure there are enough data to backtrack previous tasks, and the data size wouldn't drop as task increase. Meanwhile, we control the memory size wouldn't blow.

We consider using the Multi-task model, which is trained on a single pass over shuffled data from all tasks, as the baseline. The Multi-task training method violates the Continue Learning assumption and could be seen as an upper bound performance for average accuracy. Meanwhile, we also take EWC as a comparison model, which is classic regularization-based approaches aiming at avoiding catastrophic forgetting by limiting the learning of parameters critical to the performance of previous tasks.

4.1 Datasets

We consider using the Permuted MNIST (Kirkpatrick et al., 2016), which is a variant of MNIST (LeCun, 1998), and is transformed by permutation of pixels. Each class correspond to a specific permutation generated randomly. In this dataset, due to random permutation, we could make sure that the probability distribution for each task is unrelated.

4.2 Matrics

For a principled evaluation, we provide a test set for each of the T tasks. When the model finishes the learning about task t_i, we use the test sets for previous tasks to evaluate the model's performance. Therefore, we could get the average accuracy for the whole tasks:

$$AverageAccuracy : \mathrm{ACC} = \frac{1}{T}\sum_{i=1}^{T} R_{T,i} \tag{9}$$

where R_T, j represents the test classification accuracy of model on task t_j after observing the last sample from task T.

4.3 Result

In this part, we would discuss the experiment results of CGAN, A-GEM and A-GEM with CGAN.

4.3.1 Result of CGAN

As Figure 2 shows, different epoch will lead to a different performance in CGAN. We could get three results, under-fitting, optimal-fitting, and over-fitting. Figure 2,(a) shows the CGAN model underfitting results(epoch number = 20), and we can tell some numbers' images are not good enough, like '7' and '8'.And Figure 2,(b) shows the CGAN model's optimal fitting results(epoch number = 50), and all numbers' images are clear and recognizable. Figure 2,(c) shows the results of over-fitting CGAN(epoch number = 75), and some images start twisting.

(a) under-fitting (b) optimal-fitting (c) over-fitting

Figure 2: Result of CGAN

Therefore, we could tell the results of CGAN are sensitive to the training epochs. In order to avoid unrecognizable generator's production, we use early stop to identify the optimal CGAN model.

4.4 Result of A-GEM and A-GEM with CGAN

Figure 2 shows the Average Accuracy of Multi-task, EWC and different setting A-GEM model with 10 tasks.

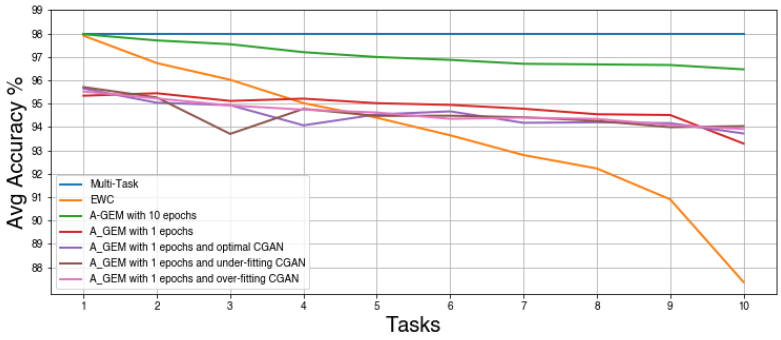

Figure 3: Perform of Model on Continue Learning Tasks

According to Figure 3, first, we could tell that with the same learning epochs(epoch number=10), A-GEM learner has better performance than EWC. Second, when we control the learning level of A-GEM, the low-level learning A-GEM has lower average accuracy than high level learning A-GEM at the beginning of 10 tasks. However, with the process of continued learning, they have a similar decrease trend on the A-ACC(Avg Accuracy). Third, for one-epoch A-GEM learners, the learner using episodic memory has a comparable performance with the leaner with using CGAN's production, which means using CGAN to replace episodic memory is feasible. Forth, we try to combine the A-GEM leaner with different performance CGAN; they all have similar A-ACC. We think this is because the over-fitting CGAN and under-fitting CGAN we use could produce recognizable results, maybe not clear enough, which still keep the main feature of each task. So the CGAN we used to replace the episodic memory wouldn't be perfect, and this means we could save some computation costs to get CGAN.

5 Conclusion

CGAN, A-GEM and two-combined have shown significant performance on the Permuted MNIST. Partly because Permuted MNIST is a relatively more straightforward task, and their performance can be different when task difficulty elasticated.

The ultimate learning ability of the model is still confined by the model structure. The task performance will experience a tremendous decline as the task number exceed 10, which correspond to 100 class number in total(the output dimension is also 100). Comparing to A-GEM, this method can save memory that initially used in the backtracking stage. However, CGAN training process is a time-consuming task that requires substantial computation. Although generating data from trained CGAN takes little time, considering the two method training time combined, we can hardly call it a time-wise approach even comparing to A-GEM.

6 Further work

For now, we use the same CGAN architecture to learn the distribution of per task. And for different tasks, the learner needs to remember the various parameter settings. We hope to introduce the Lifelong GAN (Zhai et al., 2019) to remember more tasks by using one model.

References

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 2018. Efficient lifelong learning with A-GEM. *CoRR*, abs/1812.00420.

Robert M French. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks.

James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2016. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796.

Yann LeCun. 1998. The mnist database of handwritten digits.

David Lopez-Paz and Marc Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6467–6476. Curran Associates, Inc.

Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *CoRR*, abs/1411.1784.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, and Christoph H. Lampert. 2017. icarl: Incremental classifier and representation learning. *CoRR*, abs/1611.07725.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. *CoRR*, abs/1606.04671.

Mengyao Zhai, Lei Chen, Frederick Tung, Jiawei He, Megha Nawhal, and Greg Mori. 2019. Lifelong GAN: continual learning for conditional image generation. *CoRR*, abs/1907.10107.