CSN-300: Lab-based Project

Indian Institute of Technology Roorkee

Department of Computer Science and Engineering

End-Term Report

Gradient Projection Memory for Continual Learning

Under supervision of:
Prof. Pravendra Singh

[illegible]

Group Members:
Pranaw Raj[†] (19114064)
Divyansh Agarwal[†] (19115055)
Md Junaid Mahmood[†] (19116040)

May 3, 2022

[†]All members contributed equally to the project. All members worked under the supervision of Professor Pravendra Singh at the Department of Computer Science and Engineering, IIT Roorkee.
Our code is available at https://github.com/div794/Lab-based-Project.

1 Overview

For our Lab based project, we have used a research paper published at ICLR-2021, as reference. **Gradient Projection Memory for Continual Learning** authored by *Gobinda Saha*, *Isha Garg* and *Kaushik Roy* proposes a novel approach for continual learning to mitigate catastrophic forgetting. We had already prepared a detailed report on the research paper as part of our Mid Term Report.

In the subsequent sections, we present a brief introduction to the research paper and related works in the field of **catastrophic forgetting**. This is followed by the formulation of our problem statement. Next, we describe in detail the originally proposed algorithm by the researchers along with the modifications done by us in the algorithm. Then, we try to explain the reasoning behind the modified algorithm and describe our experimental findings. Finally, we give a sound conclusion to the work done by us.

2 Introduction

Continual Learning is a concept to learn a model for a large number of tasks sequentially without forgetting the knowledge obtained from the preceding tasks. One of the popular research areas in the field of continual learning is the solution to the problem of Catastrophic Forgetting. **In neural networks, catastrophic forgetting refers to the phenomenon of forgetting the information learned in the past tasks upon learning new ones**. Thus, catastrophic forgetting drastically impacts the accuracy of a model as accuracy in the newly learnt tasks comes at the cost of deterioration in performance over previous tasks.

The chosen research paper proposes a novel approach called the **GPM approach** where a neural network learns new tasks by taking gradient steps in the orthogonal direction to the gradient subspaces deemed important for the past tasks. The research paper justify theoretically as well as practically that such orthogonal gradient descent induces minimum to no interference with the past tasks, thereby mitigates catastrophic forgetting.

3 Related Works

Numerous research works have already been carried out in the field of mitigating catastrophic forgetting, each having the approach of its own kind. However, their approaches can be broadly divided into three categories. Description about those categories are:

- **Expansion-based methods:** One of the approach to mitigate catastrophic forgetting is to increase the size of the neural network. Methods in this category overcome catastrophic forgetting by dedicating different subsets of network parameters to each task. However, network-growth is a computationally expensive option and is not scalable.

  Progressive Neural Network (Rusu et al., 2016), Dynamically Expandable Networks (Yoon et al., 2018) and Neural Architecture Search (Li et al., 2019) are some of the popular works under this approach.

- **Memory-based methods:** Another approach to mitigate catastrophic forgetting is to store episodic memories of old data. The idea is that the performance of old tasks is retained by taking gradient steps in the average gradient direction obtained from the new data and memory samples. However, these methods either compromise data privacy by storing raw data or utilize resources poorly, which again limits their scalability.

  Gradient Episodic Memory (Lopez-Paz Ranzato, 2017), Experience Replay (Chaudhry et al., 2019b) and Meta-Experience Replay (Riemer et al., 2019) are some of the popular works under this approach.

- **Regularization-based methods:** Another approach to mitigate catastrophic forgetting is to put constraints on the gradient updates so that task specific knowledge can be preserved. Adding the penalty term acts as a structural regularizer and dictates the degree of stability-plasticity of individual weight. However, due to such restrictions, performance suffers while learning longer task sequences. Elastic Weight Consolidation (Kirkpatrick et al., 2017), PackNet (Mallya Lazebnik, 2018) and HAT (Serra et al., 2018) are some of the popular works under this approach.

4 Problem Statement

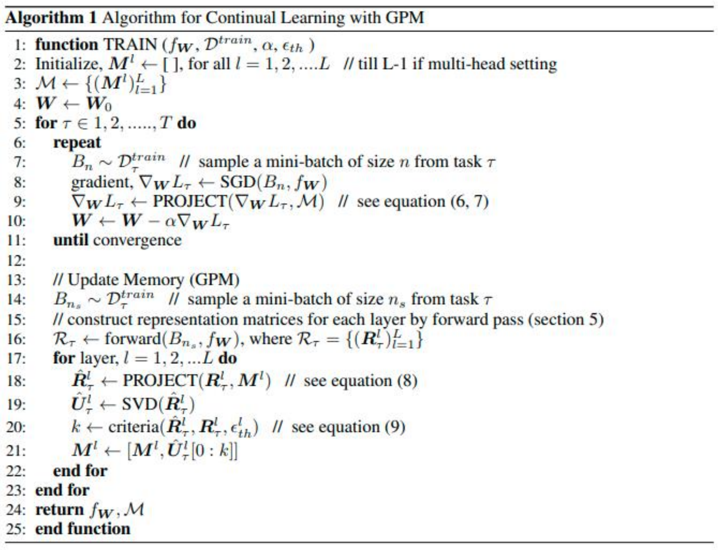
The main objective of the original research is to find an optimal algorithm in terms of both time complexity and space complexity that could effectively mitigate catastrophic forgetting in an online Continual Learning Setup. The proposed algorithm should be generic enough such that it can be incorporated with any reasonable dataset easily.

The objective of our group under the course CSN-300 Lab Based Project is to **understand the research work thoroughly and reproduce the claimed results**. In addition, we also aim to **suggest as well as implement some suitable modifications in the proposed algorithm** to further improve the accuracy over the benchmark datasets. The improvement must ensure learnability over new tasks as well as retention of learnt knowledge on previous tasks.

5 Algorithm

In the following section, we aim to describe both the algorithms in details. The first subsection corresponds to the original algorithm that was proposed by the researchers as a generic solution to the problem of catastrophic forgetting. The second subsection corresponds to the modified version of algorithm, where we describe the changes we have introduced in the algorithm.

5.1 Originally Proposed Algorithm



Algorithm 1 Algorithm for Continual Learning with GPM

```
function TRAIN ($f_W$, $\mathcal{D}^{train}$, $\alpha$, $\epsilon_{th}$ )
Initialize, $M^l \leftarrow [\,]$, for all $l = 1, 2, ....L$   // till L-1 if multi-head setting
$\mathcal{M} \leftarrow \{(M^l)_{l=1}^{L}\}$
$W \leftarrow W_0$
for $\tau \in 1, 2, ....., T$ do
    repeat
        $B_n \sim \mathcal{D}_\tau^{train}$   // sample a mini-batch of size $n$ from task $\tau$
        gradient, $\nabla_W L_\tau \leftarrow$ SGD($B_n, f_W$)
        $\nabla_W L_\tau \leftarrow$ PROJECT($\nabla_W L_\tau, \mathcal{M}$)   // see equation (6, 7)
        $W \leftarrow W - \alpha \nabla_W L_\tau$
    until convergence

    // Update Memory (GPM)
    $B_{n_s} \sim \mathcal{D}_\tau^{train}$   // sample a mini-batch of size $n_s$ from task $\tau$
    // construct representation matrices for each layer by forward pass (section 5)
    $\mathcal{R}_\tau \leftarrow$ forward($B_{n_s}, f_W$), where $\mathcal{R}_\tau = \{(R_\tau^l)_{l=1}^{L}\}$
    for layer, $l = 1, 2, ...L$ do
        $\hat{R}_\tau^l \leftarrow$ PROJECT($R_\tau^l, M^l$)   // see equation (8)
        $\hat{U}_\tau^l \leftarrow$ SVD($\hat{R}_\tau^l$)
        $k \leftarrow$ criteria($\hat{R}_\tau^l, R_\tau^l, \epsilon_{th}^l$)   // see equation (9)
        $M^l \leftarrow [M^l, \hat{U}_\tau^l[0:k]]$
    end for
end for
return $f_W, \mathcal{M}$
end function
```

Figure 1: The proposed algorithm by the researchers for the Continual Learning using **Gradient Projection Memory**.

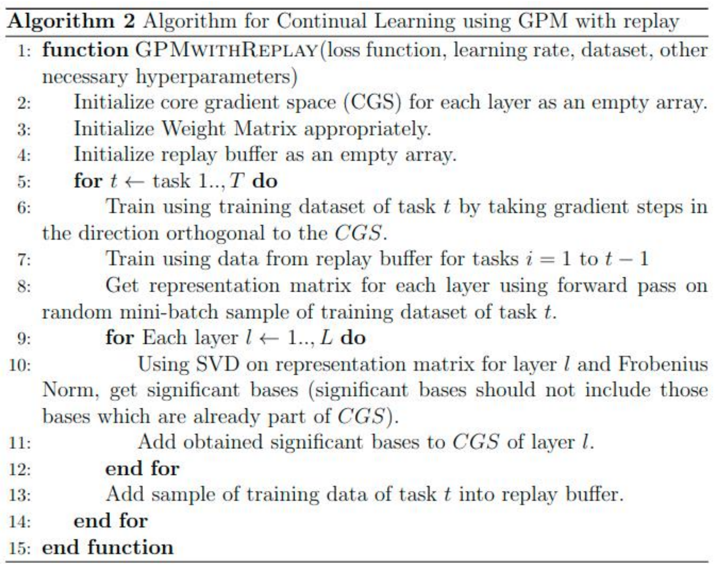
In the following subsection, we describe the original proposed algorithm. As shown in figure 1, the algorithm behind continual learning using Gradient Projection Memory(GPM) takes input in the form of a loss function, training datasets for all the tasks from 1 to T, learning rate α and threshold hyperparameter ϵ_{th}. Since the network has not been trained upon any task initially, Core Gradient Space (CGS) for each layer is set to be an empty array. Set of Core Gradient Space for all the layers in the neural network forms the Gradient Projection Memory (GPM). The weight matrix of the network is initialized to be W_0.

After all necessary initialisation, the network is trained for each task. At each iteration, first of all a mini-batch of training data from task *t* is sampled. Here *t* is a loop variable and its value will vary from 1 to T. Then, gradient descent is calculated considering this batch of training data. However, gradient descent update is done only along those directions that are orthogonal to the GPM (*CGS for each layer*). The process repeats until convergence is obtained.

After model has been trained over the task *t*, GPM needs to be updated in order to ensure preservation of past knowledge and learnability over new tasks. For this, a mini batch of training data is again sampled. Representation matrix or activation matrix is then calculated at each layer using forward pass of this mini-batch of training data. Singular Value Decomposition (SVD) is then used to identify significant bases for the matrix. Those bases that are already present in the GPM are not included in this set of significant bases. The obtained important bases are then added into the Core Gradient Space (CGS) of that layer. The process repeats for each and every layer in the neural network and in this manner GPM gets updated.

Above mention iteration is done for each and every task, starting from Task 1 up till Task T. The algorithm terminates after learning final task. Core Gradient Space for each layer after learning the final task gives the final Gradient Projection Memory (GPM) for the model.

5.2 Modified Algorithm



Algorithm 2 Algorithm for Continual Learning using GPM with replay

```
 1: function GPMwithReplay(loss function, learning rate, dataset, other necessary hyperparameters)
 2:     Initialize core gradient space (CGS) for each layer as an empty array.
 3:     Initialize Weight Matrix appropriately.
 4:     Initialize replay buffer as an empty array.
 5:     for $t \leftarrow$ task 1.., $T$ do
 6:         Train using training dataset of task $t$ by taking gradient steps in the direction orthogonal to the $CGS$.
 7:         Train using data from replay buffer for tasks $i = 1$ to $t - 1$
 8:         Get representation matrix for each layer using forward pass on random mini-batch sample of training dataset of task $t$.
 9:         for Each layer $l \leftarrow 1..,L$ do
10:             Using SVD on representation matrix for layer $l$ and Frobenius Norm, get significant bases (significant bases should not include those bases which are already part of $CGS$).
11:             Add obtained significant bases to $CGS$ of layer $l$.
12:         end for
13:         Add sample of training data of task $t$ into replay buffer.
14:     end for
15: end function
```

Figure 2: The modified **Gradient Projection Memory** algorithm by our group.

In the following subsection, we describe the modification done by us in the original algorithm. We basically followed the memory based approach. In other words, we experimented with data replay routine during training along with the original GPM approach.

In our modified algorithm, we introduced a replay buffer whose responsibility is to store training data corresponding to the learnt tasks. The size of training data to store is a research field in its own way. However, we tried to describe the size of the training data that must be stored for two datasets that we used for our experiments in Sec. 6.

Training on task 1 is quite similar to the training done in the original GPM approach (*for Task 1*). The only change introduced is that we store a batch of training data corresponding to Task 1 in the replay buffer. From Task 2 onward, we start training on replay data as well. Let us say we are currently training over task *t* (*we are training considering all layer from layer 1 to layer L*). First of all, we train our model upon corresponding training data for task *t* using gradient descent along the directions that are orthogonal to the Core Gradient Space. Upon training using task-specific data for task *t*, we again train our model using the stored training data (*from Task 1 to Task t-1*) from the replay buffer. In other words, at every iteration, we train majorly over task-specific training data and minorly over stored training data from all the preceding learnt tasks.

After training over task-specific as well as stored training data, we follow the normal GPM approach. We sample a batch of training data from the corresponding tasks, calculate representation or activation matrix using forward pass and then obtain significant bases using Singular Value Decomposition

(SVD). The significant bases, thus obtained, becomes part of the Core Gradient Space. In addition to this, we also sample a batch of task-specific data for task *t* that we add into the replay buffer for use in training of newer tasks.

Above mention approach is used for every iteration for training from Task 2 to Task T. The main motivation behind using data replay mechanism is to somehow introduce contributions from the less significant bases as well in the output that was left behind due to Singular Value Decomposition (SVD). The idea is further elaborated in Sec. 7.

6 Experiments

Dataset Used: For our experiments on modified algorithm, we used two benchmarks datasets for training and evaluating our model. These datasets are **PMNIST dataset** and **10-Split CIFAR-100 dataset**.

Training Details: All models are trained with plain Stochastic Gradient Descent (SGD).

Performance Metrics: The model is evaluated using two methods: **ACC** and **BWT** (Backward Transfer). The *ACC* metric is used to evaluate the classification performance. It is the average test classification of all tasks. Whereas, *BWT* is used to measure forgetting i.e., it indicates the influence of newer learning on past knowledge. Negative *BWT* indicates catastrophic forgetting.

$$ACC = \frac{1}{T}\sum_{i=1}^{T} R_{T,i}, \quad BWT = \frac{1}{T-1}\sum_{i=1}^{T-1} R_{T,i} - R_{i,i} \tag{1}$$

where T is the total number of tasks learnt up-to current point of evaluation and $R_{T,i}$ is the accuracy of model on i_{th} task after learning T_{th} task sequentially.

	PMNIST	10-Split CIFAR-100
No. of Tasks	10	10
Input size	1 x 28 x 28	3 x 32 x 32
No. of classes/tasks	10	10
No. of Training samples/tasks	54,000	4,750
No. of Validation samples/tasks	6,000	250
No. of Test samples/tasks	10,000	1,000

Table 1: Different statistics related to two datasets - PMNIST and 10-Split CIFAR-100

In the following subsections, we provide a brief description about the datasets along with the experimental setup and obtained results for both datasets - PMNIST and 10-Split CIFAR-100.

6.1 PMNIST

PMNIST dataset is a variant of MNIST (which contains around 70000 gray scale images of digits from 0 to 9). Only difference here is that 10 sequential tasks are created using different permutations where each task contains 10 classes (each class correspond to a digit from 0 to 9). Each task is considered as a random permutations of the original MNIST pixels.

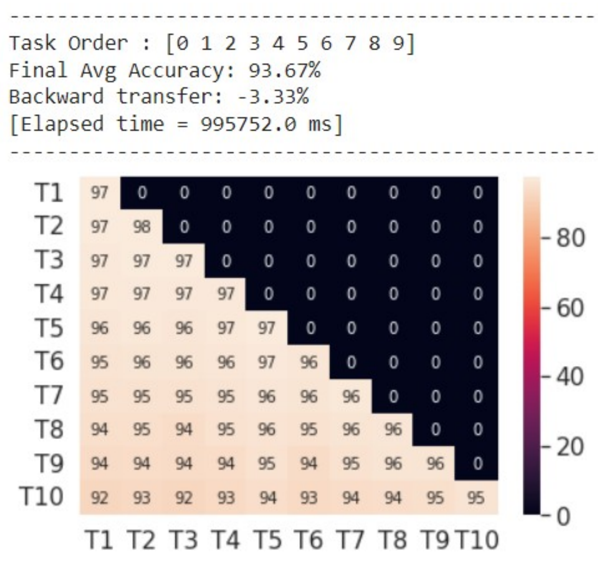
6.1.1 Setup

Following are salient features of the experimental setup used for PMNIST dataset.

- The model architecture consists of Fully Connected layer with 2 hidden layer of 100 units
- For each task we have saved 100 randomly selected training data into Replay buffer
- Number of epochs and batch size used for training and replay dataset is 5 and 10 respectively
- Value of ϵ_{th} (threshold hyperparameter) used is 0.95 for first layer and 0.99 for subsequent layers

6.1.2 Result

Following are the results obtained by us upon using above experimental setup on PMNIST dataset. The code corresponding to the original algorithm and proposed algorithm can be found here.



```
--------------------------------------------------
Task Order : [0 1 2 3 4 5 6 7 8 9]
Final Avg Accuracy: 93.67%
Backward transfer: -3.33%
[Elapsed time = 995752.0 ms]
--------------------------------------------------
```

Figure 3: Results obtained from modified algorithm on PMNIST

	Accuracy (%)	Backward Transfer (%)
Using GPM	93.07	-3.88
Using GPM with Replay buffer	93.67	-3.33

Clearly, GPM with Replay buffer outperforms the original algorithm both in terms of Accuracy as well as Backward Transfer.

6.2 10-Split CIFAR-100

10-Split CIFAR-100 dataset consists of 60000 32x32 colored images in 100 classes, with 600 images per class. These 100 classes of CIFAR-100 are split into 10 tasks, each with 10 classes.

6.2.1 Setup

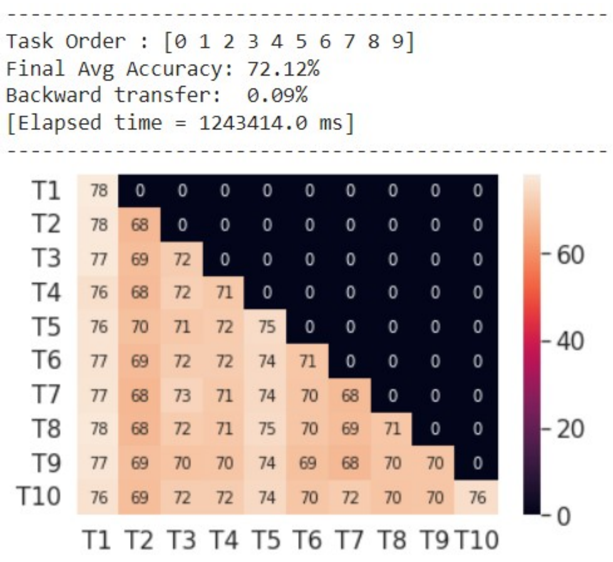
Following are salient features of the experimental setup used for 10-Split CIFAR-100 dataset.

- Network architecture used is 5 Layer AlexNet model
- Since each task contains 10 unique classes in the dataset, we needed to add examples from each and every class in the dataset. Thus, for every class we added 2 examples into Replay buffer

- Number of epochs used for the replay dataset was 5 and batch size was 10. Due to constraints in the structure of the model it was necessary to ensure that the batch size is a factor of the number of examples from each task, that is, 20. For training dataset number of epochs and batch size was 200 and 64 respectively
- Value of ϵ_{th} (threshold hyperparameter) used is 0.97 for all layers and is increased by 0.003 for each new tasks.

6.2.2 Result

Following are the results obtained by us upon using above experimental setup on 10-Split CIFAR-100 dataset. The code corresponding to the original algorithm and proposed algorithm can be found here. Clearly, GPM with Replay buffer outperforms the original algorithm both in terms of Accuracy as



```
--------------------------------------------------
Task Order : [0 1 2 3 4 5 6 7 8 9]
Final Avg Accuracy: 72.12%
Backward transfer:  0.09%
[Elapsed time = 1243414.0 ms]
--------------------------------------------------
```

Figure 4: Results obtained from modified algorithm on 10-Split CIFAR-100

	Accuracy (%)	Backward Transfer (%)
Using GPM	71.92	-0.12
Using GPM with Replay buffer	72.12	0.09

well as Backward Transfer. In addition, we can observe from the table that backward transfer in both the cases are very small (nearly 0), because the dataset is too small to get a difference. However, we think that if we would have used it on larger datasets, then we could have achieved better results.

7 Motivation & Novelty

As mentioned in Sec. 5.2, we have basically used a data replay approach in the original GPM algorithm. Under this approach, we save a sample of training data for each task. In practice, after learning each new task, we again train our model on the stored training data of the all the preceding tasks.

The motivation behind the approach can be explained using the following observation. Representation matrix is the matrix calculated using forward pass of a random batch of task-specific training data after updating the weight of the network in an iteration. The GPM approach finds significant bases of the representation matrix for a given task and then freezes it, that is, it cannot be modified during training on future tasks. Now, it is arguable that to what extent neglecting less significant bases for a task hampers performance over it. It is quite intuitive to see that with **successive training over past training data along with the future tasks, we can also add the slightly modified contributions from those bases in the final output**. This intuition was the main motivation behind our approach.

The approach proposed by us is unique in its own way. **To the best of our knowledge, we are the first to integrate the concept of data replay with the original GPM algorithm**, which makes our work novel. However, our approach comes with a limitation. Our approach is applicable only for those applications where data privacy is not an issue. This is because we are also storing a part of the training data even after learning that task completely.

8 Conclusion

In our project, we used research paper as reference. We developed thorough understanding of the algorithm proposed in the paper for mitigating Catastrophic Forgetting in an online Continual Learning Setup. We reproduced the results using the proposed algorithm on two benchmark datasets- PMNIST and 10-Split CIFAR-100 and found the claims made in the research paper to be consistent.

Using the statistics presented in the paper about the performance of other contemporary methods on same datasets, we infer that the Gradient Projection Memory (GPM) approach achieves higher performance with minimal forgetting compared to the other contemporary solutions.

We then tried to work on an approach which uses GPM approach as a backbone such that accuracy can be improved further. We developed a solution which involves use of Replay buffer to enable minor training on preceding tasks along with the major training on current tasks that is being learnt. We found that our approach increases accuracy on two benchmark datasets- PMNIST and 10-Split CIFAR-100.

To the best of our knowledge, we are the first to incorporate the concept of data replay with the original GPM algorithm. However, our solution comes with a limitation. It will work only in those scenarios where data privacy is not an issue, as we are storing training data for a task even after training is accomplished.

9 Acknowledgment

We thank Prof. Pravendra Singh for providing us with an opportunity to work on the project.
References used for the work done till now:

- Saha, Gobinda Garg, Isha Roy, Kaushik. (2021). Gradient Projection Memory for Continual Learning.
- Reza Bagheri. (2020). Understanding Singular Value Decomposition and its Application in Data Science.
- Paul Hand. (2020). Continual Learning and Catastrophic Forgetting

Pravendra

Approved: ____

Professor Pravendra Singh

Department of Computer Science and Engineering

IIT Roorkee