Gradient Projection Memory for Continual Learning

Gobinda Saha, Isha Garg, and Kaushik Roy
Elmore Family School of Electrical and Computer Engineering, Purdue University, USA

Abstract

Learning multiple task sequentially causes gradient interference between tasks which leads to **Catastrophic Forgetting**. To solve this issue, we propose a novel approach where a neural network learns new tasks by taking gradient steps in the orthogonal direction to the gradient subspaces deemed important for past tasks.

Approach

Find Important Gradient Spaces:

- ❑ Input and Gradient Spaces are related
- ❑ From learned network activations create **Representation Matrix**
- ❑ Find the bases of this matrix using Singular Value Decomposition (SVD)
- ❑ Select top-k bases using k-rank approximation with threshold hyperparameter, ϵ_{th}
- ❑ Save bases in the memory – **Gradient Projection Memory (GPM)**

Algorithm - GPM:

- ❑ Step-1: Learn new task in the space orthogonal to GPM
- ❑ Step-2: Update GPM (after each task)

Input and Gradient Space

Stochastic gradient descent (SGD) updates lie in the span of input data points [1].

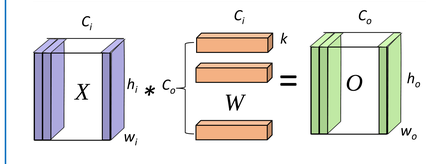
At Fully Connected layer : $\nabla_W L = \delta x^T$

($W \rightarrow$ Weight, $\delta \rightarrow$ Error, $x \rightarrow$ input)

The gradient updates in the fully connected layers lie in the span of input.

At Convolutional Layer :

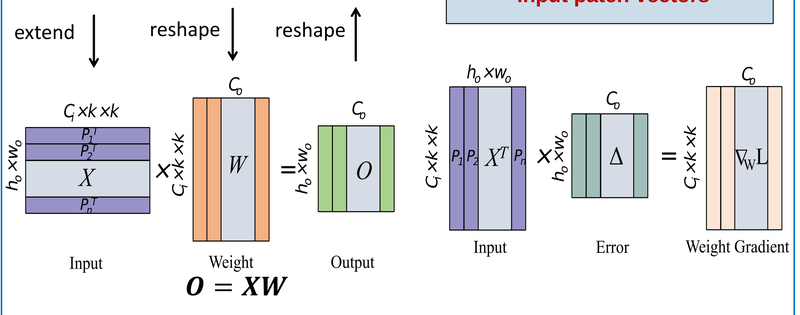
Forward Pass:



Backward Pass:

$\nabla_W L = X^T \Delta$

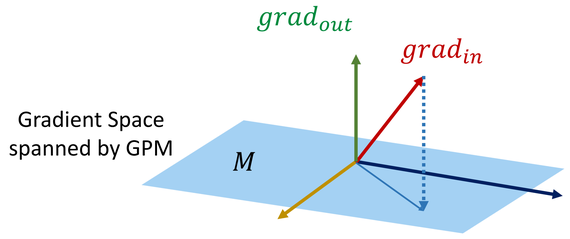
The gradient updates lie in the space spanned by the input patch vectors



In neural network, gradient spaces can be defined in terms of input spaces.

Orthogonal Gradient Descent

Learn new task in the orthogonal direction to space spanned by GPM

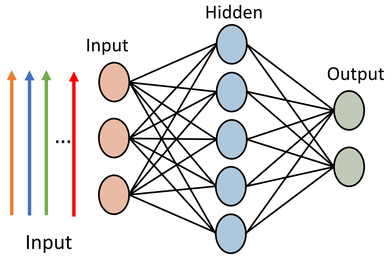


$$grad_{out} = grad_{in} - (MM^T)grad_{in}$$

Projection : $\nabla_w L = \nabla_w L - (MM^T)\nabla_w L$

Weight Update : $W = W - \alpha \nabla_w L$

Gradient Projection Memory

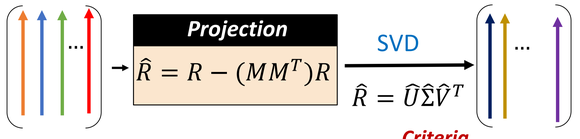


- ✓ Representation Matrix
- ✓ Representation Projection
- ✓ SVD on Representation
- ✓ Select top-k bases
- ✓ Save bases as **GPM**

Gradient Projection Memory

GPM

$M = [\boldsymbol{M}, \hat{u}_1 \dots \hat{u}_k]$



Projection

$\hat{R} = R - (MM^T)R$

SVD

$\hat{R} = \hat{U}\hat{\Sigma}\hat{V}^T$

Representation Matrix, ***R***

$R = [x_1, x_2, \dots x_{ns}]$

Conv layer : $x \rightarrow X^T$

Criteria

$||R_{proj}||_F^2 + ||\hat{R}_k||_F^2 \geq \varepsilon_{th}||R||_F^2$

Threshold hyperparameter, $\varepsilon_{th} \in [0,1]$

Experiments and Results

Dataset: CIFAR-100 Superclass, Architecture: LeNet-5

GPM outperforms expansion-based methods using less parameters

	Methods					
Metric	STL†*	PGN†	DEN†	RCL†	APD†	GPM (ours)
ACC (%)	61.00	50.76	51.10	51.99	56.81	**57.72**
Capacity (%)	2000	271	191	184	130	**100**

Experiments and Results

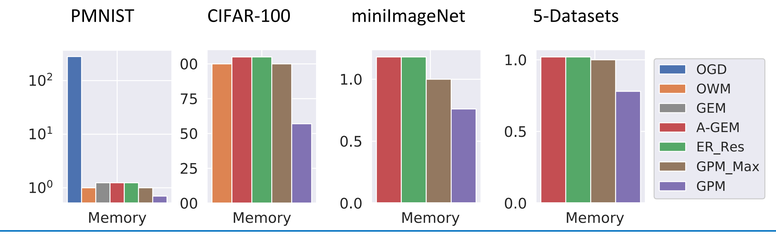
Architecture: MLP, AlexNet, ResNet18

Methods	PMNIST ACC (%)	PMNIST BWT	Methods	CIFAR-100 ACC (%)	CIFAR-100 BWT	miniImageNet ACC (%)	miniImageNet BWT	5-Datasets ACC (%)	5-Datasets BWT
OGD	82.56	- 0.14	OWM	50.94	- 0.30	-	-	-	-
OWM	90.71	- 0.01	EWC	68.80	- 0.02	52.01	- 0.12	88.64	- 0.04
GEM	83.38	- 0.15	HAT	72.06	- 0.00	59.78	- 0.03	**91.32**	- 0.01
A-GEM	83.56	- 0.14	A-GEM	63.98	- 0.15	57.24	- 0.12	84.04	-0.12
ER_Res	87.24	- 0.11	ER_Res	71.73	- 0.06	58.94	- 0.07	88.31	- 0.04
EWC	89.97	- 0.04	GPM (ours)	**72.48**	- 0.00	**60.41**	- 0.00	91.22	- 0.01
GPM (ours)	**93.91**	-0.03							
Multitask*	96.70	-	Multitask*	79.58	-	69.46	-	91.54	-

- ➢ GPM outperforms memory-based methods such as GEM, A-GEM, ER_Res
- ➢ GPM achieves best accuracy (60.41%) with zero forgetting in 20-miniImageNet tasks
- ➢ GPM has comparable performance to SOTA method in 5-dataset tasks

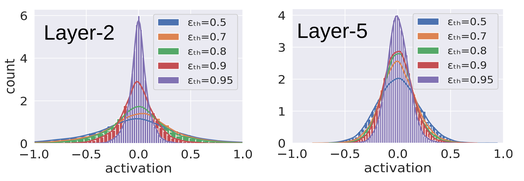
Memory Analysis

GPM outperforms memory-based methods with less memory utilization

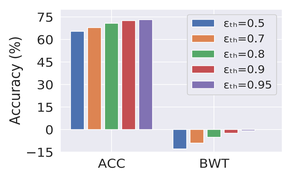


Controlling Forgetting

Distributions of Interference activation



Impact of ϵ_{th} on performance



Weight after task T : $W_T = W_1 + \Delta W_{1 \rightarrow T}$ $\quad x_1 \sim D_1$ (Task 1)

Activation for task 1 : $W_T x_1 = W_1 x_1 + \Delta W_{1 \rightarrow T} x_1$

Zero Interference: $W_T x_1 \approx W_1 x_1$ $\quad \Delta W_{1 \rightarrow T} x_1$ (interference activation) $\rightarrow 0$

- ➢ Threshold (ϵ_{th}) determines number of gradient bases selected per task
- ➢ Threshold (ϵ_{th}) controls the degree of interference between tasks
- ➢ Lower ϵ_{th} has higher interference while higher ϵ_{th} has lower interference
- ➢ Increasing ϵ_{th} reduces forgetting and improves accuracy

References: [1] Zhang et al., ICLR 2017, [2] Liu et al. NeurIPS 2018 [3] Farajtabar et al., AISTATS 2020

https://github.com/sahagobinda/GPM
gsaha@purdue.edu