

**HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY**

# **MASTER THESIS**

## **Continual Learning with Knowledge Distillation and Representation Learning**

**Tran Tung Lam**

lam.tt212393m@sis.hust.edu.vn

**Major: Data Science (Elitech)**

**Thesis advisor:** Prof. Than Quang Khoat

**Department:** Department of Computer Science

**School:** School of Information and Communications Technology

**HANOI, 10/2023**

**HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY**

# **MASTER THESIS**

## **Continual Learning with Knowledge Distillation and Representation Learning**

**Tran Tung Lam**

lam.tt212393m@sis.hust.edu.vn

**Major: Data Science (Elitech)**

**Thesis advisor:** Prof. Than Quang Khoat

\_\_\_\_\_

Signature

**Department:** Department of Computer Science

**School:** School of Information and Communications Technology

**HANOI, 10/2023**

## THESIS ASSIGNMENT

### 1. Student's information :

Name : Tran Tung Lam

Phone : 0968609009

Email: lam.tt212393m@sis.hust.edu.vn

Class : Data Science (Elitech)

Affiliation : Hanoi University of Science and Technology

Duration : 10/2021 - 10/2023

### 2. Thesis title : Continual Learning with Knowledge Distillation and Representation Learning

### 3. Declarations/Disclosures :

I – Tran Tung Lam – certify that this Master Thesis is my own research work under the guidance of Prof. Than Quang Khoat. The results stated in Master Thesis are truthful and are not copied from any other works. All references in the Master Thesis — including images, tables, figures, and quotations — are clearly documented or acknowledged. This thesis has not been previously submitted for evaluation or recognition by any other academic institution. I am fully accountable for any potential infractions contained within this thesis.

*Hanoi, date   month   year 2023*  
Author

*Tran Tung Lam*

### 4. Attestation of thesis advisor :

I certify that I have read this thesis and that, in my opinion, it is fully adequate in scope and quality as a thesis for the degree of Master of Science.

*Hanoi, date   month   year 2023*  
Thesis Advisor

*Prof. Than Quang Khoat*

# ACKNOWLEDGMENT

Words alone can never capture the profound sense of gratitude that resides within me, a gratitude that has grown stronger with each step of my academic journey.

I would like to express my deepest appreciation to my thesis advisors, Thầy Khoát, and my mentors at VinAI, anh Toàn Trần and anh Trung Lê. Their guidance and unwavering support have been the guiding stars of my academic pursuits. Their mentorship not only enabled me to successfully complete this thesis but also lit the path to a promising research career. I often wonder where I'd be without their invaluable supervision and insightful suggestions. Their belief in my abilities has been an unshakable source of motivation.

My parents, your enduring encouragement has been a beacon of hope and strength in my life. Your unwavering faith in my abilities is the bedrock upon which I built my dreams. Every achievement I've made is in no small part due to your support and love.

To my cherished friends, though I regret not mentioning your names here, know that I carry each of you in my heart. Our shared moments, laughter, and support have been integral to my journey. I eagerly await the opportunity to express my appreciation in person, to clink glasses in celebration of our friendship and shared experiences.

Finally, but by no means less significant, I extend my heartfelt thanks to VinIF for their generous financial support throughout my Master's journey. This thesis is funded by Vingroup JSC and supported by the Master, PhD Scholarship Programme of Vingroup Innovation Foundation(VINIF), Institute of Big Data, code VINIF.2021.ThS.BK.03.

In closing, as I reflect upon my academic path, I am reminded that none of my achievements have been solitary endeavors. It is with your collective guidance, support, and belief in me that I have reached this point in my journey. My heart is filled with profound gratitude, and I am eager to carry this gratitude forward as I continue to grow and contribute to the world of knowledge and discovery.

# ABSTRACT

The memory-based approach has emerged as an effective solver for the Continual Learning problems (CL) by storing a limited-size buffer of examples representing past tasks to replay with data of the current one. In this thesis, we study two forms in which these examples are saved: Raw data and Bases of the subspace spanned by past tasks' representation features. Both these forms are used as inputs to encourage feature representations across different layers of past data to remain unchanged, so previously learned knowledge could be preserved.

Methods using the former form typically add distillation losses to the current task's loss to distill the knowledge of previous tasks while learning the new one. However, they usually consider these losses in their empirical forms, and thus can easily be overfitting to training data, especially to the limited buffer. This means that the distilled model has less generalization ability on data of old tasks and, hence can not reduce forgetting effectively. To overcome this problem, we leverage Sharpness Aware Minimization (SAM) [1] to improve models' generalization by seeking flat minima whose neighbors' loss values are also low. However, we conjecture that directly applying SAM to memory-based CL methods whose loss function contains multiple objectives may cause gradient conflicts among them. We then propose to manipulate each objective's SAM gradient such that their potential conflicts are minimized by adopting one gradient aggregation strategy from Multi-task Learning. Extensive experiments empirically verify our hypothesis and show consistent improvements over strong memory-replay baselines.

The latter form is the ingredient for gradient-based methods which constrain gradient updates of the current task to be orthogonal to the subspace of old tasks, hence can guarantee no change in old data' feature representations. Inspired by this benefit, and also by the advantages of flat minima in CL, a recent work built on a gradient-based method proposes to flatten the sharpness of the loss landscape. However, we argue that their method could provoke gradient conflicts between current and old tasks, and could hinder SAM in finding flat minima. We thus propose a simple solution to remove such potential gradient conflicts and allow SAM to freely find flat minima for all tasks learned so far. Furthermore, we introduce a novel objective function that explicitly aligns those gradients, hence leading to better knowledge transfer in CL. Experiments confirm our claim and the effectiveness of our method.

The proposed methods in this thesis show a comprehensive picture of how

memory-based CL methods can benefit from sharpness minimization. In particular, despite initial improvements, we analyze potential conflicts caused by the application of SAM on two representative methods, representational knowledge distillation-based and gradient-based, and then propose solutions to further enhance CL performance. Our contributions are valuable to the CL community because we focus on the generic memory-based method which many sophisticated works are built on.

# TABLE OF CONTENTS

<b>Chapter 1. Introduction.....</b>	<b>1</b>
1.1 Motivations.....	1
1.2 Contributions .....	3
1.3 Outline .....	4
<b>Chapter 2. Background .....</b>	<b>5</b>
2.1 Neural Networks.....	5
2.2 Continual Learning .....	6
2.2.1 Problem Formulation.....	6
2.2.2 Desiderata and Metrics of Continual Learning .....	7
2.2.3 Three Scenarios for Continual Learning .....	7
2.2.4 Dataset Construction for Continual Learning .....	9
2.3 Knowledge Distillation.....	10
2.4 The Core Objective of Memory-based Methods.....	11
2.5 Buffer Construction and Buffer Sampling in Memory-based Methods .....	12
2.6 Regularization-based Continual Learning Approach .....	13
2.7 Sharpness Aware Minimization (SAM).....	13
2.8 Gradient Conflict and Projecting Conflicting Gradients (PCGrad) .....	14
<b>Chapter 3. Related work .....</b>	<b>16</b>
3.1 Three main Continual Learning approaches .....	16
3.2 Flat Minima in Continual Learning .....	17
3.3 Two existing Flat-seeking Minimizers for Continual Learning .....	18
3.3.1 Classifier-Projection Regularization for Continual Learning (CPR)	
18	
3.3.2 Finding Flat Minima for Incremental Few-Shot Learning (F2M) ....	19

3.4 Gradient Projection Memory (GPM) .....	20
3.5 Flattening Sharpness for GPM (FS-GPM) .....	22
<b>Chapter 4. Sharpness and Gradient Aware for Memory-based Con-</b>	
<b>tinual Learning .....</b>	<b>24</b>
4.1 Introduction .....	24
4.2 Methodology .....	25
4.2.1 Sharpness Aware Minimization for Memory-based Methods .....	25
4.2.2 Potential Conflicts of Worst-case Gradients .....	26
4.2.3 Sharpness and Gradient Aware Minimization for Memory-based Methods .....	27
4.2.4 Discussion .....	27
4.2.5 Theoretical Analysis .....	29
4.3 Experiments .....	29
4.3.1 Experimental Analysis .....	30
4.3.2 Ablation Study .....	32
4.4 Summary .....	34
<b>Chapter 5. Projected Worst-case Perturbation may put Old and New</b>	
<b>tasks at Odds .....</b>	<b>36</b>
5.1 Introduction .....	36
5.2 Methodology .....	37
5.2.1 Two potential problems of FS-GPM .....	37
5.2.2 Our simple solution .....	39
5.2.3 Discussion .....	41
5.3 Experiments .....	42
5.3.1 Experimental Analysis .....	44
5.3.2 Ablation Study .....	45
5.4 Summary .....	48



<b>Chapter 6. Conclusion .....</b>	<b>49</b>
<b>REFERENCE .....</b>	<b>55</b>
<b>APPENDIX .....</b>	<b>57</b>
A Theorem's Proof .....	57
B Network Architecture .....	61
C Hyper-parameters .....	62

# LIST OF FIGURES

Figure 2.1	A 3-layer simple neural network. . . . .	5
Figure 2.2	A 3-layer simple neural network under CL. . . . .	6
Figure 2.3	Examples of CIFAR-10, CIFAR-100, Tiny ImageNet and MNIST. . . . .	9
Figure 4.1	The ratio of conflict per epoch for Baseline, SAM-CL and SGAM-CL. . . . .	34
Figure 4.2	Loss (left) and Accuracy (right) changes w.r.t weight per- turbation of SGAM-CL and SAM-CL. . . . .	35
Figure 4.3	Loss (left) and Accuracy (right) changes w.r.t weight per- turbation of SAM-CL and several baselines. . . . .	35
Figure 5.1	Visualization for problems of FS-GPM. . . . .	39
Figure 5.2	The ratio of conflict per epoch of FS-GPM and Ours. . . . .	46
Figure 5.3	Loss (left) and Accuracy (right) changes w.r.t weight per- turbation of FS-GPM and Our method. . . . .	47
Figure 5.4	Hyper-parameter analysis of $\rho, \rho_2, \alpha$ . . . . .	47

# LIST OF TABLES

Table 2.1	Examples of three CL scenarios. . . . .	8
Table 4.1	Classification results for standard CL benchmarks between SGAM-CL and baselines . . . . .	31
Table 4.2	Ablation studies of SGAM-CL on S-CIFAR-10, 200 buffer size. (-) means ablated, (+) means used. . . . .	32
Table 4.3	Comparison between SGAM-CL and SAM-CL on S-CIFAR- 10 and S-Tiny-Imagenet. . . . .	33
Table 5.1	Experimental results on task incremental learning. . . . .	44
Table 5.2	Experimental results on PMNIST in single-epoch with 3 runs. . . . .	45
Table 5.3	Ablation study on CIFAR-100 Split and Superclass . . . . .	45
Table 1	Hyper-parameters for Table 4.1 . . . . .	62
Table 2	Hyper-parameters for Tables 5.1 and 5.2 . . . . .	62

# LIST OF ABBREVIATIONS

Abbreviation	Definition
ACC	Accuracy
A-GEM	Averaged Gradient Episodic Memory
BWT	Backward Transfer
CE	Cross Entropy
CIL	Class Incremental Learning
CL	Continual Learning
CPR	Classifier Projection Regularization
DER	Dark Experience Replay
DIL	Domain Incremental Learning
ER	Episodic Replay
EWC	Elastic Weight Consolidation
F2M	Finding Flat Minma
FS-GPM	Flattening Sharpness for Gradient Projection Memory
GEM	Gradient Episodic Memory
GPM	Gradient Projection Memory
HAL	Hindsight to Anchor Past Knowledge in Continual Learning
iCaRL	Incremental Classifier and Representation Learning
KD	Knowledge Distillation
KL	Kullback-Leibler divergence
LwF	Learning without Forgetting
MAS	Memory Aware Synapses
MeRGAN	Memory Replay Generative Adversarial Networks
oEWC	Online Elastic Weight Consolidation
P-MNIST	Permuted MNIST
PCGrad	Projecting Conflicting Gradients
PNN	Progressive Neural Network
R-MNIST	Rotated MNIST
S-Tiny-Imagenet	Sequential Tiny-ImageNet
S-CIFAR-10	Sequential CIFAR-10
SAM	Sharpness Aware Minimization
SGAM	Sharpness Gradient Aware Minimization
SGD	Stochastic Gradient Descent

Abbreviation	Definition
SI	Synaptic Intelligence
SVD	Singular Value Decomposition
TIL	Task Incremental Learning

# Chapter 1. Introduction

## 1.1 Motivations

Our world is changing and evolving rapidly, and for this reason, the human brain needs to constantly change its neural connections to adapt to new knowledge. This process happens continuously during the whole lifetime of humans, by which they can learn new expertise based on what they have acquired, and utilize this new knowledge to improve one learned in the past.

Thus, this ability to acquire knowledge continually and endlessly without forgetting previously learned one has been a desired goal for many deep learning models. This is necessary because when being deployed in real-world scenarios, a model tends to face changes in a dynamic environment in which data distribution may be different from what the model has been previously trained on. In many cases, they are required to deal with tasks coming in a sequential manner. Take a robot whose function is classifying images as an example, it is very likely that new categories will appear as it moves to different areas, and the model should be able to classify these new categories as well as previously trained ones. One naive solution is to gather data on all tasks seen so far and then re-train the whole model whenever a new task appears. However, this creates a great challenge for the model as it will encounter heavy computations or long delays for enough data to be available. Ideally, the model should be updated with only data from the current distribution. However, this inevitably causes significant degradation in the performance of the model on previous tasks because a deep model generally only achieves impressive performance compared to humans when trained with static data, following identical independent distribution (iid). This phenomenon is called Catastrophic Forgetting [2], and to address this problem, Continual Learning (CL) was introduced and has received a great deal of attention from the community.

Existing CL methods are often divided into three main groups: architecture-based [3], [4], regularization-based [5]–[7], and memory-based approaches [8]–[11]. Methods in the first group often either allocate a new part or spare a subset of the network for each new task, resulting in the inevitable network expansion or network insufficiency as more tasks arrive. Meanwhile, the other two groups can deal with CL using a fixed-size network and thus are more applicable in real-life settings, such as deployment on edge devices. In particular, the second group deals with forgetting by penalizing changes in parameters during learning new tasks

based on their importance to old ones. The third group stores a limited size of examples representing old tasks to replay with data from the current task.

Among these two approaches, memory-based, or memory replay, methods become notable because they often achieve state-of-the-art under many CL settings. Their saved examples can be in the original space [8], in hidden space [12], or in gradient space [13], [14]. The replay strategies for these examples can be designed in many different sophisticated ways [15]–[18]. Nevertheless, the majority of this group follows one simple principle when designing the objective function. Specifically, methods saving buffer in the original or hidden space combine the current task’s loss with a specific loss computed on the buffer, e.g. cross-entropy or  $l_2$  regularization on hidden feature layers. The purpose of this additional loss is to **distill knowledge** of previous tasks by constraining **representation layers**, e.g., the output or penultimate layer, of saved examples to remain stable while learning new tasks. Methods working on gradient space also aim to achieve this representational stability, although through a different constrain strategy based on gradient orthogonality that allows them to use the current task’s loss only.

Typically, these losses are considered in the empirical form, i.e. the average over training data points, so memory-based methods tend to share one common drawback: *overfitting* on training data [1], notably to the limited size buffer. In general, the overfitting problem of neural networks is a result of a high-dimensional non-convex loss function whose landscape is complex and contains many local optima [19], [20]. Therefore, to achieve a more generalized model, seeking flat local minima has been one of the most effective approaches [1], [21], [22]. This idea of a flat-seeking optimizer has been applied to CL [13], [23], [24]. Broadly speaking, they observe that a model converging to a flat local minimum tends to find common optimal regions for different tasks more easily than one that converges to sharp minima, thus it obtains better CL performance. However, their empirical results have not yet shown significant improvement for CL, especially under difficult settings, because of the ineffective flat-seeking method [23], or the improper approximation of a model’s flatness [24]. Recently, Deng et al. [13] studied SAM [1], a recently introduced Sharpness Aware Minimization technique, on a gradient-based CL method under task-incremental learning setting, i.e. we know which task a test sample belongs to (known task identity). Therefore, despite obtaining promising results, their conclusion about the benefits of SAM on CL is not yet ready to be applied to a wider range of memory-based methods, and under a class-incremental learning scenario, a more realistic and challenging CL setting in which we do not know in advance task identity of the test sample. More importantly, after thorough

investigation, we further find that their method may hinder the ability of SAM to find flat minima, and may incur conflicts between the learning of new tasks and previously learned ones.

## 1.2 Contributions

In this thesis, on a high level, we aim to investigate the benefit of sharpness aware minimization [1] to memory-based CL methods, or representational knowledge distillation-based to be more specific under our scope. Furthermore, we also study the mechanism of a close baseline, named FS-GPM [13], and reveal their hidden problems, from which we propose a simple solution to enhance their performance. Our contributions can be divided into two main works as follows.

*First*, we propose to incorporate SAM, an effective flat-seeking optimizer, into CL (SAM-CL) to boost model performance. Different from FS-GPM, we apply SAM to a simple yet general memory-based baseline, and additionally conduct experiments under class-incremental learning setting (CIL), a more practical and difficult scenario when no task identity is given during inference [25]. Thus, our work not only can be easily extended to other complex memory-based methods whose core objective function is similar to our chosen baseline but also can perform under a popular CL setting, CIL. Nevertheless, combined with FS-GPM, ours can provide a comprehensive analysis of the benefit of SAM on CL. Furthermore, after analyzing the SAM gradients of each term in the objective function, we conjecture and empirically verify that there may exist conflicts between these gradients, which can hinder the capability of the model to converge to flat minima. This drives us to our final proposed method: Sharpness Gradient Aware Minimization in CL, (SGAM-CL), which first finds the SAM gradient of each loss term, and then combines them using a gradient aggregation strategy to reduce potential conflicts. Through extensive experiments on four common benchmarks, we show that SGAM-CL brings a significant improvement for the baseline, and verify the further enhancement of SGAM-CL over SAM-CL. This work was accepted as a paper at the International Symposium on Information and Communication Technology conference, SOICT 2023.

*Second*, we analyze the worst-case perturbation vector that is projected to the subspace representing old tasks' features and find that at the corresponding perturbed weight, the loss function computed on the combination of current task and buffer data minimizes the cosine similarity between gradients of these two types of data. This can lead to interference between learning new tasks and keeping old



knowledge. Additionally, we also show the infeasibility of SAM in finding flat minima under some extreme cases: when a new task has little interference with the old subspace, i.e., learning this new task does not cause forgetting. From these investigations, we propose a simple solution to allow SAM to freely seek flat minima corresponding to both buffer and current data and to remove the potential gradient conflict. We also propose a novel objective function that explicitly aligns these gradients, thus further improving the performance. Extensive experimental analyses confirm our method’s effectiveness and its rationales.

## 1.3 Outline

With these contributions, the thesis contains the following chapters:

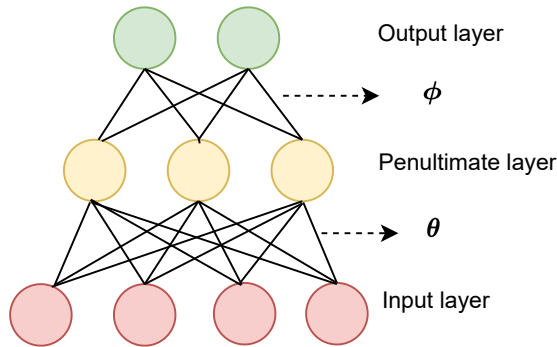
- Chapter 2 first provides needed preliminaries regarding continual learning: its formulation, desiderata, metrics, common scenarios, and dataset benchmarks. Then Knowledge Distillation is briefly introduced, which is followed by memory-based and regularization-based methods. Finally, two techniques our methods leverage are described: Sharpness Aware Minimization and Projecting Conflicting Gradients.
- Chapter 3 presents the overview of three main approaches to solving continual learning problems, as well as how flat minima have been exploited in CL. Then details about two flat-seeking optimizers that have been introduced to CL are given. Finally, two gradient projection-based methods are demonstrated.
- Chapter 4 presents our first contribution with one simple solution incorporating flat minima to CL, and its potential drawbacks in terms of gradient conflict. Then we describe our method to tackle such issues, supported by extensive experiments.
- Chapter 5 presents our second contribution which starts with an in-depth analysis of an existing work also employing sharpness minimization in CL. We then show their hidden problems and propose a simple solution to solve them
- Chapter 6 summarizes our work in this thesis, and gives some thoughts about future research directions which are based on our methods’ current limitations.

## Chapter 2. Background

This chapter provides basic concepts of Neural Networks, Continual Learning, and its commonly used metrics, scenarios, and datasets. Additionally, we introduce Knowledge Distillation, which is followed by a brief description of Memory-based methods that use knowledge distillation on representational layers to retain knowledge, and the construction and usage of the buffer that contains replay data for these methods. We then introduce Regularization-based methods. Next, we present Sharpness Aware Minimization [1], which is the main focus of our two contributions. Finally, we shortly present the concept of gradient conflict, which originates from Multi-task learning [26], [27], and one simple gradient aggregation strategy to mitigate this problem.

### 2.1 Neural Networks

A typical deep neural network consists of one input layer, several hidden layers, and one output layer. The second-last layer is called the penultimate layer. The network, or model, is parameterized by  $w = (\theta, \phi)$ , where  $\theta$  represents the parameter set of the model's feature extractor module  $f$ , and  $\phi$  is the parameters of the classifier module  $g$ , in the case of a classification model. See Figure 2.1 for a simple demonstration of a 3-layer neural network. Please refer to [28] for a more detailed description of neural networks.



**Figure 2.1:** A 3-layer simple neural network.

Given a training set  $D_{\text{tr}} = \{(x_i, y_i)\}_{i=1}^N$ , parameters, or weights, of the model are updated through back-propagation to minimize a loss function, e.g., cross-entropy for classification tasks, computed on this training set. Due to the non-linearity of the neural networks, most objective functions are non-convex and contain many local optima.

Traditional training assumes that  $D$  contains independent and identically distributed, i.i.d, data points sampled from an underlying distribution  $\mathcal{D}_{x,y}$ . After learning the model, it will be tested on an unseen test set  $D_{\text{test}}$ . The impressive performance of deep neural networks is often obtained when a sufficient amount of data is given, and the distribution of training and test data are similar [29].

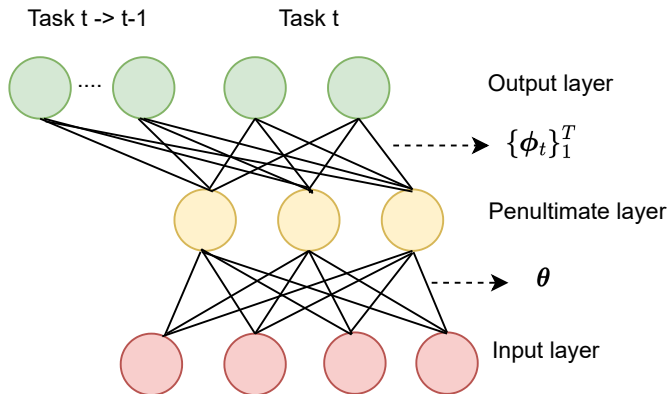
## 2.2 Continual Learning

Continual Learning or Lifelong Learning, Incremental Learning, studies the problem of training a neural network using a sequence of tasks each of which contains a different training dataset. We focus on the continual classification problem to be in line with the setting of many existing CL methods.

### 2.2.1 Problem Formulation

Formally, denote  $1, 2, \dots, T$  as the indices of  $T$  sequential incremental tasks where each task includes a set of disjoint classes. For the  $t^{\text{th}}$  task, we denote  $D_t = \{(x_{t,i}, y_{t,i})\}_{i=1}^{N_t}$  to represent the dataset corresponding to this task, where  $N_t$  is the number of samples in the dataset.

In addition, we have a model parameterized by  $w = (\theta, \{\phi_t\}_{t=1}^T)$ , where  $\theta$  represents the parameter set of the model's feature extractor module  $f$ , and  $\phi_t$  is the parameters of the classifier module  $g$  for the classes belonging to task  $t$ . At the beginning of the  $t^{\text{th}}$  task, given the model  $\{\theta_{t-1}^*, \phi_{1:t-1}^*\}$  previously trained on past tasks, and only data  $D_t$ , we need to update the model to obtain a new one  $\{\theta_t^*, \phi_{1:t}^*\}$ . Figure 2.2 gives an example of the incrementally added classifier heads.



**Figure 2.2:** A 3-layer simple neural network under CL.

## 2.2.2 Desiderata and Metrics of Continual Learning

To build an effective CL model, several important points to aim for are:

1) **Stability-Plasticity**: The new model should remain good performance on the previous  $t - 1$  tasks while having the ability to adapt to the  $t^{th}$  task. This is called the *Stability-Plasticity dilemma*.

2) **Constant memory w.r.t tasks**: To prevent linearly increasing memory for saving models w.r.t the number of tasks, we need to ensure the total memory consumption should not too large.

3) **Forward transfer**: This means that the model should be able to utilize learned knowledge to better adapt to new ones.

4) **Backward transfer**: Ideally, the model should aim to improve learned knowledge by using current data, not limited to retaining it.

To evaluate the CL performance of a method, two common metrics are used:

$$\text{Average Accuracy: } ACC = \frac{1}{T} \sum_{i=1}^T A_{T,i}, \quad (2.1)$$












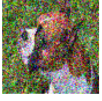
$$\text{Backward Transfer: } BWT = \frac{1}{T-1} \sum_{i=1}^T (A_{T,i} - A_{i,i}), \quad (2.2)$$

and where  $T$  is the total number of tasks and  $A_{j,i}$  is the accuracy of task  $i$  after the model has learned task  $j$ .  $BWT$  denotes the interference of new task on old task performance, i.e., a lower value of  $BWT$  means more severe forgetting and a positive value means some tasks' performance has been improved (positive backward transfer).

Intuitively, a good CL model should maintain high overall results on all tasks while minimizing performance drop of previous tasks, high  $ACC$  and high  $BWT$ . Between the two, the former is more informative in terms of balancing plasticity and stability, because a low value of accuracy  $ACC$  can still come with a high value of backward transfer  $BWT$ , e.g. when the model underfits and thus forgets less, or even improves old tasks.

## 2.2.3 Three Scenarios for Continual Learning

Following [31], we introduce the concept of three common scenarios in CL: Task-incremental learning (TIL), Domain-incremental learning (DIL), and Class-incremental learning (CIL). In all three scenarios, the model is essentially learned

Task	TIL		CIL		DIL	
$D_{i-1}$	x: 		x: 		x: 	
	y: Bird	Dog	y: Bird	Dog	y: Bird	Dog
task-ID(test)	<b>i-1</b>		<b>Unknown</b>		<b>Unknown</b>	
$D_i$	x: 		x: 		x: 	
	y: Ship	Guitar	y: Ship	Guitar	y: Bird	Dog
task-ID(test)	<b>i</b>		<b>Unknown</b>		<b>Unknown</b>	

**Table 2.1:** Source is from [30]. Examples of the three CL scenarios. (x, y, task-ID) represents (input images, target label, and task identity). The main distinction between TIL and CIL is the availability of task-ID. The main difference between CIL and DIL is that in CIL, a new task contains completely new classes, whereas, in DIL, a new task consists of new instances from a different distribution (e.g., noise) of all the seen classes.

with all the data of one task at a time. This is considered offline CL.

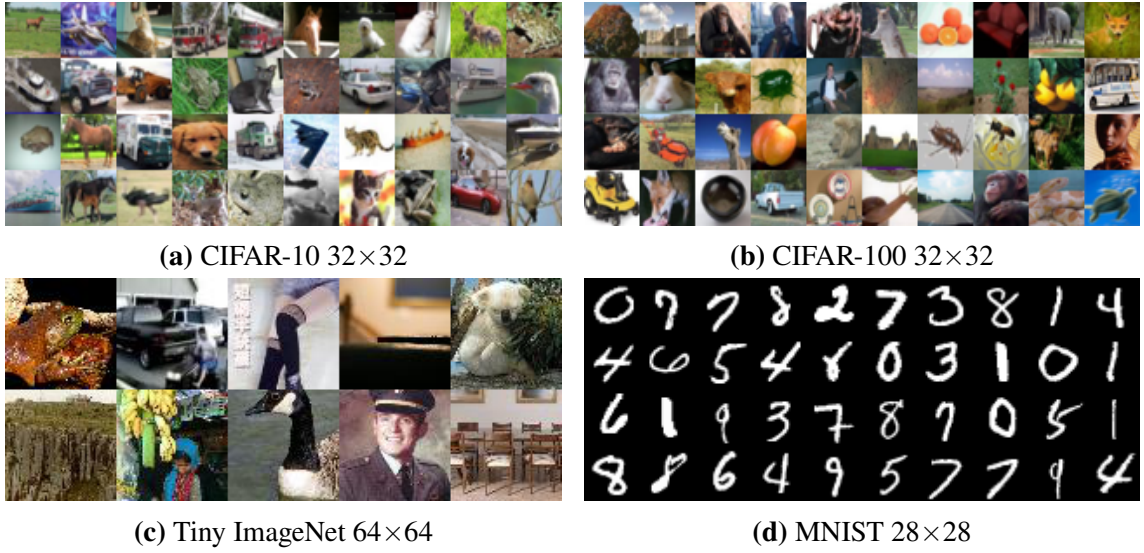
(1) In TIL: the model always has the identity of the task in both the training and inference phases. Taking advantage of this, models are designed to have a separate output layer for each task, or, in short, a multi-head CL.

(2) In DIL: classes of tasks are the same, but their distribution is different. A real-world example of this scenario is a self-driving car agent that needs to learn to make decisions regarding three directions: left, right, or straight in different environments, without necessarily needing to know the specific environment it is interacting with. Normally, the model only needs one classifier head in this scenario, i.e., single-head CL.

(3) In CIL: the model does not know which task the data is being inferred belongs to, and it needs to classify the images into all classes seen from the beginning of the learning process.

The main difference between CIL and TIL is the information about the task identity of a test sample, which leads to severe degradation in the performance of many good TIL methods [12]. CIL is also known as a realistic and challenging CL setting. Please refer to Table 2.1 for an example of the three scenarios.

Now look back to the Problem Formulation above, we have defined  $\{\phi_t\}_{t=1}^T$  is the set of classifier parameters, or 'heads' from task 1 to task  $t$ . Under the evaluation of TIL, knowing task identity  $t$  of a sample means that the model only needs to make a prediction by feeding it through the head  $\phi_t$ . Meanwhile, under CIL, the model needs to feed the sample through the whole classifier  $\phi_{1:T}$ .



**Figure 2.3:** Dataset (a) CIFAR-10 (32x32). (b) CIFAR-100 (32x32). (c) Tiny ImageNet (64x64). (d) MNIST (28x28).

## 2.2.4 Dataset Construction for Continual Learning

We provide a brief description of datasets that we use in the experiments of this thesis and how to construct them into sequences of tasks for CL. Note that we here only present the general idea for dataset construction, details are left to the experiment section of each chapter. A few examples of each dataset are given in Figure 2.3.

- CIFAR-10 [32] consists of 50,000 training images and 10,000 testing images evenly split into 10 classes representing different objects, all with a resolution of 32x32 pixels.
- CIFAR-100 [32] is similar to CIFAR-10, which has 100 classes that are grouped into 20 superclasses. Each class has 500 and 100 images for training and testing, respectively.
- Tiny-ImageNet [33] evenly splits 100000 colored images into 200 classes which are of size 64x64. Each class has 500 training, 50 validation, and 50 images. We will use the validation set for the reported results.
- MNIST [34] contains 70000 images, 60000 for training, and 10000 for testing, of hand-written digits from 0-9. Each image has a size of 28x28 and has a white digit on the black ground.

Following many CL methods, to make these datasets suitable for CL experiments, we split them into groups of classes to form a sequence of tasks. This is applied for CIFAR-10, CIFAR-100, and Tiny-ImageNet. For example, a 10-split CIFAR-100 is a sequence of 10 tasks, each of which is a 10-way classification

problem. This splitting strategy allows us to stimulate TIL and CIL. In addition, to create a benchmark for DIL, we use pixel permutation and image rotation on MNIST. In specific, a Permuted MNIST uses one common pixel permutation for all images, and a 20-split Permuted MNIST contains 20 such permuted versions. A rotated MNIST is created by rotating each image at the same angle, and similarly, a 20-split Rotated MNIST chooses 20 angles and applies each of them to the original MNIST. These create a 20-task benchmark under DIL.

To sum up, our experiments focus on continual image classification whose datasets are created by randomly splitting existing benchmarks into multiple non-overlapping groups of classes, and by permuting or rotating images several times. Below is specific datasets we use for three CL scenarios.

- For TIL and CIL: Split CIFAR100, Split CIFAR10 and Split Tiny-ImageNet.
- For DIL: Rotated MNIST and Permuted MNIST.

## 2.3 Knowledge Distillation

Knowledge Distillation (KD) [35] is originally a network compression technique that can transfer knowledge from a pretrained teacher, normally a big model having been trained on a large dataset, to a student, a much smaller network. This is important as in many cases, small models are more favored due to their practicality, such as edge device deployments. This teacher-guidance mechanism also facilitates the learning of the student model as it receives more supervision from a decent network in addition to the one-hot labels. Formally, suppose  $y_s$  and  $y_t$  are the outputs of the student  $w_s$  and teacher  $w_t$  networks for one data point, respectively. Then the knowledge distillation loss of this sample is defined as:

$$KD(w_s; w_t, x) = KL(\text{softmax}(y_s/\tau) || \text{softmax}(y_t/\tau)), \quad (2.3)$$

where  $\text{softmax}(y)$  is a vector whose entry  $i$ th is  $\frac{e^{y_i}}{\sum_{j=1}^K e^{y_j}}$  with  $K$  is the number of neurons in the output layer;  $\tau$  is the temperature to softens the probability over classes and KL is the Kullback–Leibler divergence that measures distance between two probabilities:  $KL(p||q) = \sum_{i=1}^K p_i \log \frac{p_i}{q_i}$ . Note that  $w_t$  is kept frozen and only  $w_s$  is updated. Later on, other types of distillation loss were proposed, such as the  $L_2$  regularization between other layers of the two networks.

In continual learning, Knowledge Distillation is the right recipe when the model trained from the previous task can be treated as the frozen teacher, and the stu-

dent one is the model learning the current task which distills knowledge from the teacher, i.e., retains old knowledge. Learning without Forgetting [6] is a typical example of using KD to mitigate forgetting. It is worth noting that if old data is not observed, one workaround is to use the current task’s data to feed to the distillation loss. However this might be not ideal as the old model has not trained with this new data yet, hence the supervision may be noisy [36]. This problem, however, is less serious for memory-based methods as they allow a small number of past data to be stored. More detail about these methods is given in the next section.

## 2.4 The Core Objective of Memory-based Methods

Methods in this group save data points of previous tasks in a small buffer  $B$ , i.e.  $|B| \ll |D_{1:t-1}|$ , to replay with data of the current task. Recall the *Stability-Plasticity* dilemma, it can be seen that The latter property can be achieved by optimizing the model with the commonly used classification loss function on  $D_t$ :

$$\mathcal{L}_{D_t}(\mathbf{w}) := \mathbb{E}_{(x,y) \sim D_t} \ell(g(f(x, \boldsymbol{\theta}), \boldsymbol{\phi}_t), y) \quad (2.4)$$

where  $\ell(.,.)$  is the cross-entropy loss function (CE). However, the former property is challenging as we have no access to previous data, and if the model is trained with Eq. (2.4) only, it will face catastrophic forgetting, i.e., its classification performance on past classes degrading significantly. To overcome this, a small subset of past data is stored in a buffer  $B = \{M_i\}_{i=1}^{t-1}$ . In this scenario, many sophisticated methods to retain the previously acquired knowledge encapsulated in  $\{\boldsymbol{\theta}_{t-1}^*, \boldsymbol{\phi}_{1:t-1}^*\}$  and  $B$  have been developed [16], [18]. Despite this, their objective functions mostly follow the same core design: combining the CE loss on the current task’s data with CE loss on buffer data and a knowledge distillation loss between the current model and the model trained after task  $t - 1$ , represented as follows:

$$\mathcal{L}_{total} := \mathcal{L}_{D_t}(\mathbf{w}) + \mathcal{L}_B(\mathbf{w}) + KD_B(\mathbf{w}, \mathbf{w}_{t-1}^*), \quad (2.5)$$

where  $KD_B(\mathbf{w}, \mathbf{w}_{t-1}^*)$  can be the  $L_2$  regularization on the feature space, i.e.  $\|f(x, \boldsymbol{\theta}) - f(x, \boldsymbol{\theta}_{t-1}^*)\|_2^2$ , or the KL divergence on the output space, i.e.  $\mathcal{D}_{KL}(\text{softmax}(o) \parallel \text{softmax}(o_{t-1}^*))$ ,  $o = g(f(x, \boldsymbol{\theta}), \boldsymbol{\phi}_{1:t-1})$ . In addition to using CE loss only (the second term), being inspired by knowledge distillation, the third loss term is introduced to force the current model to mimic the knowledge of its teacher, i.e. the previous model [12], [37], thus further reducing forgetting. It should be noted that the design choices for this term can be varied. For instance, DER [12]



uses the  $L_2$  regularization on the logit space and additionally saves samples' logits along the training trajectory to the buffer. These logits are then treated as targets for the current model's logits to match, instead of using logits generated from the previous model as in conventional KD. It is worth noting that this simple replay strategy has shown its great benefit in the CIL scenario while only needing a fixed-size network. This is because when the model has access to old data, the classification heads of previous tasks can be retrained with those of new ones, thus a more calibrated classifier head for all classes is obtained [30]. Also, as we can view the CE loss as the regularization between the model's outputs and the one-hot probability, we will refer to CE loss, together with the regularization terms mentioned above, as distillation loss whose input is old data and functionality to preserve old knowledge.

## 2.5 Buffer Construction and Buffer Sampling in Memory-based Methods

As aforementioned, these methods rely on a small buffer that stores data of past tasks. This section presents one way to construct such buffer, Reservoir Sampling [38], and how to sample from this buffer to train with current task, Random Sampling.

Reservoir Sampling is commonly adopted by many memory-based methods due to its simplicity and applicability under many CL settings [12]. Reservoir sampling ensures every coming data point has the same probability to be stored in the buffer. In specific, this probability is the ratio between the fixed size of the buffer and the number of data points observed so far. Algorithm 1 presents the pseudo-code for updating the buffer using reservoir sampling with one coming data point. It is worth noting that while there are other buffer construction strategies choosing data points based on specific criteria, such as the herding selection [10], we choose reservoir sampling in this thesis for simplicity since our scope does not involve how to update the buffer.

Random Sampling the the easiest way to random past data from the buffer to replay with current one. This is done at every mini-batch update when looping over data of the current task. It should be noted that these past samples can also be chosen and modified before training using many other principles, such as maintaining class-balance [10] or maximizing conflicting with current data [18]. However, we employ random sampling as our contributions are not about buffer retrieval.

---

**Algorithm 1** Reservoir Sampling

---

**Input:** A fixed-size buffer  $B$ , number of seen data points  $N$ , one data point  $(\mathbf{x}, y)$

```
1: if  $|B| > N$  then
2:    $B[N] = (\mathbf{x}, y)$ 
3: else
4:    $i = \text{RandomInteger}(0, N)$            {Random an integer in range  $[0, N]$ }
5:   if  $i < |B|$  then
6:      $B[i] = (\mathbf{x}, y)$ 
7:   end if
8: end if
9: return  $B$ 
```

---

## 2.6 Regularization-based Continual Learning Approach

As its name suggests, to preserve old knowledge, methods in this group add a regularization term to the current task’s loss function. Typically, this term penalizes large updates of parameters that are important for previous tasks. Formally, suppose the main loss is cross entropy, cf. Eq. (2.4), the new loss can be defined as:

$$\mathcal{L}_{total}(\mathbf{w}) := \mathcal{L}_{D_t}(\mathbf{w}) + \lambda \sum_j \Omega_j^t(\mathbf{w}_j - \mathbf{w}_{t-1,j}^*), \quad (2.6)$$

where  $\mathbf{w}_j$  and  $\mathbf{w}_{t-1,j}$  are the  $j^{th}$  parameter of current and previous optimal models, respectively;  $\lambda$  is the controlling weight for the regularization term.

A few notable examples of this approach are Elastic Weight Consolidation (EWC) [5], Synaptic Intelligence (SI) [7], Memory Aware Synapses (MAS) [39]. EWC estimates  $\Omega$  using the Diagonal Fisher Information Matrix after learning each task. SI and MAS compute the gradient of the loss function and the learned function, i.e., the sum of squares of logits, to approximate  $\Omega$ .

## 2.7 Sharpness Aware Minimization (SAM)

Now we present the mechanism of the flat optimizer that we use for our contributions.

Traditional training techniques, which concentrate on reducing the empirical loss, are susceptible to overfitting issues. This occurs when the validation error plateaus even though the training loss continues to drop, leading to a limitation

in the model's ability to generalize. To address this, Foret et al. [40] suggested a method that involves minimizing the worst-case loss within a neighborhood of the current model parameters. Specifically, assume our model has parameter set  $\mathbf{w}$ , then the worst-case loss is:

$$\min_{\mathbf{w}} \max_{\|\epsilon\|_p \leq \rho} \mathcal{L}_D(\mathbf{w} + \epsilon), \quad (2.7)$$

where  $\rho$  is the radius perturbation of the current parameter  $\mathbf{w}$ .

The optimal solution of the inner optimization in Eq.(2.7) can be approximated using first-order Taylor expansion as:

$$\begin{aligned} \epsilon^*(\mathbf{w}) &:= \arg \max_{\|\epsilon\|_p \leq \rho} \mathcal{L}_D(\mathbf{w} + \epsilon) \approx \arg \max_{\|\epsilon\|_p \leq \rho} \mathcal{L}_D(\mathbf{w}) + \epsilon^T \nabla_{\mathbf{w}} \mathcal{L}_D(\mathbf{w}) \\ &= \arg \max_{\|\epsilon\|_p \leq \rho} \epsilon^T \nabla_{\mathbf{w}} \mathcal{L}_D(\mathbf{w}) = \rho \cdot s(\nabla_{\mathbf{w}} \mathcal{L}_D(\mathbf{w})) \frac{|\nabla_{\mathbf{w}} \mathcal{L}_D(\mathbf{w})|^{q-1}}{(|\nabla_{\mathbf{w}} \mathcal{L}_D(\mathbf{w})|_q^q)^{1/p}}, \end{aligned}$$

with  $p, q$  satisfy:  $\frac{1}{p} + \frac{1}{q} = 1$ , and  $s(\cdot)$  is the sign function. If we set  $p = 2$ , the worst-case perturbation  $\epsilon^*$  is:

$$\epsilon^* = \rho \cdot \frac{\nabla_{\mathbf{w}} \mathcal{L}_D(\mathbf{w})}{\|\nabla_{\mathbf{w}} \mathcal{L}_D(\mathbf{w})\|_2}. \quad (2.8)$$

This leads to the gradient update of the model:

$$\mathbf{g}^{SAM} := \nabla_{\mathbf{w}} \max_{\|\epsilon\|_p \leq \rho} \mathcal{L}_D(\mathbf{w} + \epsilon) \approx \nabla_{\mathbf{w}} \mathcal{L}_D(\mathbf{w})|_{\mathbf{w} + \epsilon^*}. \quad (2.9)$$

## 2.8 Gradient Conflict and Projecting Conflicting Gradients (PCGrad)

We finish this chapter by demonstrating the problem of gradient conflict, which is a tool for our analyses in this thesis, and one simple solution for it, PCGrad [26].

In multi-task learning, we aim to concurrently learn  $n$  tasks with the corresponding objective functions:  $\mathcal{L}_1(\mathbf{w}), \dots, \mathcal{L}_n(\mathbf{w})$ , which is normally done by minimizing the loss:  $\min_{\mathbf{w}} \sum_{i=1}^n \mathcal{L}_i(\mathbf{w})$ . However, in practice, this minimization does not guarantee to decrease losses of all tasks due to the possible conflicts between tasks. Particularly, some tasks may dominate others, leading to the improvement of some tasks while the rest suffer [26]. Formally, two tasks conflict when the cosine sim-

ilarity between their gradients is less than 0:  $\langle \mathbf{g}_i, \mathbf{g}_j \rangle < 0$ , where  $\langle \cdot, \cdot \rangle$  is the dot product. This means task gradients might cancel out each other or point in a direction where the performance of one task will be hurt. It is worth noting that in many MTL settings, the model's parameter  $\mathbf{w}$  often consists of two components: the shared parameters between all tasks  $\mathbf{w}_{sh}$  and the non-shared classifier parameters of individual tasks  $\mathbf{w}_{ns,i}$  ( $\forall i = 1, \dots, n$ ), and gradient aggregation strategies work on these shared  $\mathbf{w}_{sh}$  while the non-shared parts are updated normally using gradient signals from their corresponding objective functions. To ease notation, we have omitted the non-shared classifiers and used  $\mathbf{w}$  to refer  $\mathbf{w}_{sh}$ .

PCGrad resolves the disagreement between tasks by projecting gradients that conflict with each other, i.e.  $\langle \mathbf{g}_i, \mathbf{g}_j \rangle < 0$ , to the orthogonal direction of each other. Specifically,  $\mathbf{g}_i$  is replaced by its projection on the normal plane of  $\mathbf{g}_j$ :

$$\bar{\mathbf{g}}_i = \mathbf{g}_i - \frac{\langle \mathbf{g}_i, \mathbf{g}_j \rangle}{\|\mathbf{g}_j\|^2} \mathbf{g}_j \quad (2.10)$$

Then the aggregated gradient is computed based on these deconflicted vectors  $\mathbf{g} = \sum_i^n \bar{\mathbf{g}}_i$ . Below is the algorithm for PCGrad.

---

**Algorithm 2** PCGrad

---

**Input:** Shared parameters  $\mathbf{w}$ , individual task losses  $\mathcal{L}_1(\mathbf{w}), \dots, \mathcal{L}_n(\mathbf{w})$ , learning rate  $\eta$ .

**Output:** Updated shared parameter  $\mathbf{w}^*$

```

1: for  $i \in \{1, \dots, n\}$  do
2:   Compute gradients  $\mathbf{g}_i$ 
3: end for
4: for  $i \in \{1, \dots, n\}$  do
5:   for  $j \in \{1, \dots, n\}$  and  $j \neq i$  do
6:     if  $\langle \mathbf{g}_i, \mathbf{g}_j \rangle < 0$  then
7:        $\bar{\mathbf{g}}_i = \mathbf{g}_i - \frac{\langle \mathbf{g}_i, \mathbf{g}_j \rangle}{\|\mathbf{g}_j\|^2} \mathbf{g}_j$ ;  $\bar{\mathbf{g}}_j = \mathbf{g}_j - \frac{\langle \mathbf{g}_i, \mathbf{g}_j \rangle}{\|\mathbf{g}_i\|^2} \mathbf{g}_i$ 
8:     end if
9:   end for
10: end for
11: Gradient aggregation:  $\mathbf{g} = \sum_i^n \bar{\mathbf{g}}_i$ 
12: Update shared parameter:  $\mathbf{w}^* = \mathbf{w} - \eta \mathbf{g}$ 
13: return  $\mathbf{w}^*$ 

```

---

## Chapter 3. Related work

In this chapter, we give an overview of three main approaches to solving Continual Learning. Then we will present in detail works that are closely related to ours. First, in terms of our first contribution, in Chapter 4, we present Classifier-Projection Regularization for Continual Learning (CPR) [23] and Finding Flat Minima for Incremental Few-Shot Learning (F2M) [24]. After that, we introduce the mechanism Gradient Projection Memory (GPM) [14] because our second main contribution in this thesis, Chapter 5, builds upon this method. Finally, we summarize Flattening Sharpness for GPM (FS-GPM) [13], a method that incorporates sharpness minimization into GPM.

### 3.1 Three main Continual Learning approaches

Continual learning, also known as lifelong learning, aims to enable machine learning models to learn new tasks sequentially without forgetting previously learned tasks. A key challenge is catastrophic forgetting, where learning a new task interferes with and overwrites knowledge about previous tasks. There are three main approaches to mitigate forgetting in CL.

**Expansion-based methods.** assign separate subsets of parameters to each task to avoid interference. For example, Progressive Neural Networks [3] learn lateral connections to prior tasks to leverage existing features while allocating new parameters to each task. Dynamically expandable networks like Progressive Neural Networks [3] and Dynamically Expandable Network [41] grow the network capacity by allocating new layers tailored to each incremental task. Reinforced Continual Learning [42] and Compacting, Picking and Growing [43] combine architecture search methods to find the optimal configuration for new tasks. Alternatively, with a fixed-size network, Hard attention to the tasks [4] learns a task-specific mask over neurons and restricts the update of gradients connecting two important neurons. PathNet [44] uses agents to select pathways through a neural network to optimize parameter usage for new tasks. While these methods can reduce interference to a great extent, they normally require expanding the model complexity and knowing task identity during testing, thus not applicable under CIL. Some works, e.g., [45] tackle the latter by training a good task predictor to choose the correct subnetwork for an evaluated sample, which may not always guarantee to be effective and may cause a computation burden.

**Regularization-based methods.** adds regularization terms to the loss function to restrict changes in weights that are important for old tasks. Elastic Weight Consolidation [5] is a seminal method that penalizes weight updates based on their Fisher information to preserve performance on old tasks. Another line of works, inspired by Knowledge Distillation, forces outputs of the current network to be similar to those of the previous one, with Learning without Forgetting [6] as a notable example. More recent methods like Memory Aware Synapses [39] and Hard Attention to Task [4] learn to regulate weight importance and plasticity. One of the limitations is determining weight importance can itself be challenging. This group of methods has been reported to perform poorly under the CIL setting when saving no past data [12].

**Memory-based methods.** store data from previous tasks and replay them during new task learning. Exemplar-replay [9] uses reservoir sampling to select incoming data for the buffer. iCaRL [10] saves data closest to the feature mean of each class. Gradient Episodic Memory (GEM) [46] and Averaged GEM [8] store a subset of data and constrain gradient updates to not increase loss on replayed data. When access to raw data is not allowed or more old data is required, generative replay approaches like Deep Generative Replay [47], [48] use generative models to produce synthetic samples from previous tasks; Or MeRGAN [49] leverages generative adversarial networks to produce condensed memory for efficient replay. Recently, Dark Experience Replay [12], whose objective is a mixture of distillation, regularization, and replay, obtains a strong yet simple baseline. Based on the fact that the majority of CIL competitive methods share this same loss, such as [18], [50], [51], a further boost to this simple baseline would benefit the community.

## 3.2 Flat Minima in Continual Learning

Flat-seeking minimizers have been extensively studied theoretically and empirically due to their ability to improve model generalization in single-task learning [1], [19], [21]. Inspired by this, several works attempt to investigate the merits of flat minima in CL [12], [13], [23], [52]. For instance, in [52], forgetting of the first task is approximated using Taylor expansion suggests from which a wider minimum is proven to forget less. It then proposes to control batch size, learning rate, and dropout during training as these elements have been proven to promote wider local minima [19]. Additionally, CPR [23] enforces high entropy of softmax outputs to reduce forgetting of regularization-based methods under the TIL scenario. Later, in few-shot CIL, F2M [24] introduces flatness to the first task training

of base classes so that good solutions for novel tasks can be found in such flat regions. Recently, FS-GPM [13] studies the dilemma between learning new tasks and memorizing old ones from the weight loss landscape viewpoint. They similarly find that solutions with low loss values and flat landscapes can lead to better CL performance, then apply SAM to Gradient Projection Memory, a method requiring task-id to achieve good results.

Compared with our method, CPR [23] adds an entropy penalty to the objective of existing regularization-based methods which may make the model sensitive to hyper-parameters as a higher weight for the entropy term can hinder learning of the current task, while a lower value may not guarantee a flat minimum. Although F2M [24] also promotes flatness by penalizing the loss value of current parameters and those of close neighbors, it achieves this by randomly adding noises to the current parameters and jointly minimizing losses of obtained models. This approximation in practice is costly or may not find flat minima if the number of random samplings is small to maintain feasible training time. Lastly, as mentioned earlier, the investigation in FS-GPM [13] does not cover the CIL scenario, and mainly shows the benefit of SAM to gradient-based methods. In Chapter chapter 4 of this thesis, we aim to present the superiority of SAM over previous flat minimizers when applied to a strong CL approach, replay-based, and under a more realistic scenario, CIL. Not only that, we find that the incorporation of SAM into the gradient-based framework of FS-GPM may raise two hidden issues that actually can hinder the learning of the new task and the power of SAM. As a side note, in combination with works done in FS-GPM, we can provide a comprehensive study of the weight loss landscape on the performance of memory-based CL methods.

### **3.3 Two existing Flat-seeking Minimizers for Continual Learning**

#### **3.3.1 Classifier-Projection Regularization for Continual Learning (CPR)**

CPR [23] originally is a plug-in method for the regularization-based approach by finding wide local minima for the current task. It regularizes the entropy of the model’s softmax output to be close to the uniform distribution, i.e. maximizing

entropy. Formally, derived from Eq. 2.6, CPR’s loss is defined as:

$$\mathcal{L}_{CPR}(\mathbf{w}) := \mathcal{L}_{D_t}(\mathbf{w}) + \lambda \sum_j \Omega_j^t(\mathbf{w}_j - \mathbf{w}_{t-1,j}^*) + \frac{\beta}{|D_t|} \sum_{i=1}^{|D_t|} KL(g(f(x, \boldsymbol{\theta}), \boldsymbol{\phi}_t) || \mathbf{P}_U),$$

where  $\mathbf{P}_U$  is the uniform distribution over  $C_t$ , the number of classes in task  $t$ , and  $\beta$  is a hyper-parameter to control this entropy maximization term.

Though simple, CPR may be sensitive to the hyper-parameter  $\beta$  as the last term forces the outputs of the model to be uniform over classes, which contradicts the purpose of CE loss, making the model produce confident outputs for the correct class. It should be noted that in our experiments, the model has access to the smaller buffer of old data. Thus we allow CPR to maximize the entropy of all classes seen so far. We hypothesize that this is necessary since at evaluation, the model has to predict from all learned classes, so during training, it should be able to produce ‘proper’ confidence over all these classes. Later through experimental results, we will show that this flat-seeking technique is not as beneficial for CL as SAM is, especially under the difficult setting of CIL.

### 3.3.2 Finding Flat Minima for Incremental Few-Shot Learning (F2M)

As a method specified for Incremental Few-Shot Learning, i.e., all tasks except the first one have only a few examples provided, Shi et al. [24] propose F2M to seek for flat minima of the base classes of the first task, which can better generalize to later tasks. They approximate the flat solutions by jointly optimizing some models, each of which is constructed by adding small random noise to the original model. The intuition is that if they find a local minimum such that its close neighbors also have low losses, that minimum is flat. Formally, suppose  $P(\epsilon)$  is the truncated random noise, ideally, given a training set  $D_t$ , the following empirical loss should be minimized:

$$\mathcal{L}_{F2M}(\mathbf{w}) := \int_{\mathbb{R}^{d_\epsilon}} \mathcal{L}_{D_t}(\mathbf{w} + \boldsymbol{\epsilon}) dP(\boldsymbol{\epsilon}), \quad (3.1)$$

where  $d_\epsilon$  is the dimension of  $\epsilon$ . However, computing this integral is intractable, they hence approximate it by randomizing noise  $M$  times and taking the average



loss:

$$\mathcal{L}_{F2M}(\mathbf{w}) := \frac{1}{M} \sum_{i=1}^M \mathcal{L}_{D_t}(\mathbf{w} + \epsilon_i). \quad (3.2)$$

They further add another term to regularize the changing of class prototypes after perturbing the network, as follows:

$$\mathcal{L}_{F2M}(\mathbf{w}) := \frac{1}{M} \sum_{i=1}^M \mathcal{L}_{D_t}(\mathbf{w} + \epsilon_i) + \frac{\lambda}{|C_t|} \sum_{c \in C_t} \|p_c - p_c^*\|_2^2, \quad (3.3)$$

where  $|C_t|$  is the number of classes in task  $t$ ;  $p_c$  and  $p_c^*$  are class prototypes of class  $c$  before and after injecting noise, respectively. The class  $c$ 's prototype is the mean of features of data points belonging to that class:  $p_c = \frac{1}{|N_c|} \sum_{(x,y):y=c} f(x, \theta)$ , with  $|N_c|$  is the number of samples of class  $c$ .

In practice, to ensure feasible training time, they only inject noise to the few last layers of the model, e.g., 4 layers, and sample noise for a small number of times, e.g., 2 to 4 times. We experimentally find that this results in a poor approximation of the perturbed neighbors, thus the improvement for CL is rather marginal. Note that in our settings, each task has roughly the same number of samples, thus we employ their loss for all of them, instead of only the first tasks as they do in their continual few-shot learning scenario.

### 3.4 Gradient Projection Memory (GPM)

Consider a simple case of two-task sequential learning. Denote the weight matrix at the layer  $l$  of the model after learning task 1 as  $\mathbf{w}_l^1$  which receives  $\mathbf{x}_{l-1}$  as input and outputs  $\mathbf{x}_l$ . In order to keep  $\mathbf{x}_l$  of a sample  $\mathbf{x}_0^1$  from task 1 unchanged when learning Task 2, the following condition should be satisfied:  $\mathbf{w}_l' \mathbf{x}_{l-1}^1 = (\mathbf{w}_l^1 + \Delta \mathbf{w}_l^1) \mathbf{x}_{l-1}^1 = \mathbf{w}_l^1 \mathbf{x}_{l-1}^1$ , where  $\mathbf{w}_l'$  is the model weight after one update under the training process of task 2. This means that the update  $\Delta \mathbf{w}_l^1$  should be orthogonal to the space spanned by samples from Task 1.

Inspired by the simple observation above, GPM, proposed in [14], aims to find the representation subspace of all learned tasks in the past and then constrains the model's weight update to be orthogonal to this constructed subspace. In this way, the knowledge learned from the previous task would not significantly interfere when learning the current task. More specifically, denote  $\mathbf{S}$  be the subspace of the learned previous tasks, and to constrain the updated weight, we seek an orthogonal

direction with the subspace  $\mathbf{S}$  from the gradient vector after the back-propagation operation:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta (\mathbf{I} - \mathbf{S} \cdot \mathbf{S}^T) \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}) \quad (3.4)$$

The subspace  $\mathbf{S}$  is gradually constructed over  $t$  tasks. Suppose the model has finished learning task 1, we first construct the representation matrix from a batch of input  $\mathbf{R}_1^\ell = [\mathbf{x}_1^\ell, \dots, \mathbf{x}_n^\ell] \in \mathbb{R}^{n \times d}$  with  $\mathbf{x}_i^\ell \in \mathbb{R}^d$  is the corresponding output of  $\mathbf{x}_i$  after forwarding through the  $\ell^{th}$  layer of the deep network. Then we carry out the SVD (Singular Value Decomposition) on  $\mathbf{R}_1^\ell$ :

$$\mathbf{R}_1^\ell = \mathbf{U}_1^\ell \Sigma_1^\ell (\mathbf{V}_1^\ell)^T \quad (3.5)$$

with  $\mathbf{U}_1^\ell \in \mathbb{R}^{n \times n}$ ,  $\mathbf{V}_1^\ell \in \mathbb{R}^{d \times d}$  are unitary matrices,  $\Sigma_1^\ell \in \mathbb{R}^{n \times d}$  is rectangular diagonal matrix with non-negative singular values in a descending order. Then we perform  $k$ -rank approximation to select the  $k$  first columns in matrix  $\mathbf{U}_1^\ell$ :  $\mathcal{B}_1^\ell = [\mathbf{u}_{1,1}^\ell, \mathbf{u}_{1,2}^\ell, \dots, \mathbf{u}_{1,k}^\ell]$  as the basis vectors of the subspace of current task and push these to the set of basis vectors  $\mathbf{S} = \mathbf{S} \cup \mathcal{B}_1^\ell$  as follow:

$$\|(\mathbf{R}_1^\ell)_k\|_2^2 \geq \epsilon \cdot \|\mathbf{R}_t^\ell\|_2^2 \quad (3.6)$$

with  $(\mathbf{R}_t^\ell)_k = \sum_{i=1}^k \sigma_{1,i}^\ell \mathbf{u}_{1,i}^\ell (\mathbf{v}_{1,i}^\ell)^T$ .

For each task  $t > 1$ , we first repeatedly construct the representation matrix  $\mathbf{R}_t^\ell$  and the basis set  $\mathcal{B}_t^\ell$  after learning the model for task  $t$ . However, some basis vectors in  $\mathcal{B}_t^\ell$  are duplicated in obtained set  $\mathbf{S}$ . Therefore, we aim to obtain the set  $\mathbf{S}$  including the most important basis of old tasks and newly constructed bases. First for each distinct basis  $\mathbf{u}_i^\ell$  in  $\mathbf{S}$ , we compute the corresponding eigenvalue as  $\delta_i^\ell = (\mathbf{u}_i^\ell)^T \cdot \mathbf{R}_t^\ell \cdot (\mathbf{R}_t^\ell)^T \cdot \mathbf{u}_i^\ell$

Then we perform SVD on  $\hat{\mathbf{R}}_t^\ell = \mathbf{R}_t^\ell - \mathbf{R}_t^\ell \cdot \mathbf{S} \cdot \mathbf{S}^T$  and gain the set of eigenvalue  $\{\hat{\sigma}_i^\ell\}$ . We then concatenate two sets  $\{\hat{\sigma}_i^\ell\}$  and  $\{\delta_i^\ell\}$  in a vector  $\boldsymbol{\delta}$  and sort in a descending order. We continue perform  $k_t$ -rank approximation on  $\mathbf{R}_t^\ell$  to collect  $k_t$  first elements whose summation is greater than  $\epsilon \cdot \|\mathbf{R}_t^\ell\|_2^2$ . These elements form  $\mathbf{S}$ .

To sum up, GPM is a data-free CL method that can guarantee minimal forgetting of previous tasks, i.e., **great stability**. By forcing the gradient update of the current task to be orthogonal to the feature subspace learned so far, it learns subsequent tasks such that the representation of data from old tasks remains unchanged, i.e., **invariant feature representation across tasks**. However, this is ensured for all hidden layers except the last linear classification layer, thus GPM typically requires

task-id during testing, or in other words, works well under the TIL scenario. In addition, it can be seen that GPM can face the challenge of lacking plasticity. This happens when the subspace  $\mathbf{S}$  of all tasks so far has spanned the whole feature space, meaning the remained gradient after subtracting out the projection is zero, thus the model can not learn new tasks.

### 3.5 Flattening Sharpness for GPM (FS-GPM)

By observing that a CL method forgets an old task less and learns a new task better when the loss landscape of that task is flatter, FS-GPM further improves the stability of GPM by promoting flatness of the new task’s loss landscape with respect to the subspace of old tasks. Additionally, to facilitate GPM’s plasticity, a soft learnable weight accounting for the importance of each basis of the subspace is introduced to free less important bases for later task learning. Formally, each basis  $\{\mathbf{u}_i\}_{i=1}^K$  of the subspace  $\mathbf{S}$  now will be assign with an importance weight  $\{\lambda_i\}_{i=1}^K$ . The diagonal matrix of these weights is denoted as  $\mathbf{\Lambda} = \text{diag}[\lambda_1, \dots, \lambda_K]$ . Note that FS-GPM stores a small subset of old task data  $B$ , and reconstructs the subspace  $\mathbf{S}$  after learning each task.

To encourage a flat loss landscape, FS-GPM relies on the intuition of SAM [1] to optimize the worst-case loss of the new task. However, instead of finding the worst-case perturbation in the entire parameter space, they restrict it to be in the old parameter space, i.e., in the subspace spanned by learned tasks, as follows:

$$\min_{\mathbf{w}} \left( \max_{\boldsymbol{\epsilon} \in \mathbf{S}} \mathcal{L}_{D_t \cup B_t}(\mathbf{w} + \boldsymbol{\epsilon}) \right) \quad (3.7)$$

Because of the constraint of the inner maximization, we can not approximate its solution as in Eq. 2.8. However, we can iteratively gradient-ascend update in order to obtain the solution  $\boldsymbol{\epsilon}$ , which is initialized as  $\mathbf{0}$ , after some steps as follows:

$$\boldsymbol{\epsilon} \leftarrow \boldsymbol{\epsilon} + \eta_1 \left( \mathbf{S} \mathbf{\Lambda} \mathbf{S}^T \right) \nabla_{\mathbf{w}} \mathcal{L}_{D_t}(\mathbf{w})|_{\mathbf{w}+\boldsymbol{\epsilon}}, \quad (3.8)$$

in which  $\mathbf{\Lambda} = \{\lambda_i\}_{i=1}^K$  indicates the importance of each basis of the subspace  $\mathbf{S}$  and each value  $\lambda_i$  is also updated iteratively:

$$\lambda_i \leftarrow \lambda_i - \eta_2 \nabla_{\lambda_i} \mathcal{L}_{D_t \cup B_t}(\mathbf{w} + \boldsymbol{\epsilon}). \quad (3.9)$$

Finally, finding the updated value  $\mathbf{w}$  of the outer minimization problem in Eq.

(3.7) is conducted similarly to GPM with the inclusion of  $\Lambda$  to adjust the importance of each basis:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta_3 \left( \boldsymbol{I} - \boldsymbol{S}\boldsymbol{\Lambda}\boldsymbol{S}^T \right) \nabla_{\boldsymbol{w}} \mathcal{L}_{D_t \cup B_t}(\boldsymbol{w})|_{\boldsymbol{w}+\boldsymbol{\epsilon}}. \quad (3.10)$$

# Chapter 4. Sharpness and Gradient Aware for Memory-based Continual Learning

## 4.1 Introduction

Continual Learning is an evolving field in machine learning, that aims to enable models to learn continuously from a sequence of tasks with varying data distributions. A challenging CL scenario is Class Incremental Learning, where a model sequentially learns new categories and must classify all seen classes without task identity information, leading to a fundamental issue in CL known as Catastrophic Forgetting [2], where performance on earlier tasks degrades due to the absence of old task data and differences in data distributions.

In CIL, models are required to classify test samples without prior knowledge of their task identities. Replay-based methods have shown promise in addressing this challenge [10], [12], [18] by using a buffer for old data. However, due to overfitting to the small buffer, their performance still falls behind joint training, which combines all data for training from scratch. Storing more old data can boost performance, but becomes computationally expensive with a large number of tasks and raises privacy concerns. Therefore, there is a workaround to increase the model’s generalization on old task data. As one of the most effective methods to achieve this [1], [21], [22], flat minimizers which seek for flat low loss landscapes have been introduced to CL [13], [23], [24]. Despite their evidence for improvement, we argue that such improvement can be further enhanced because their methods still have drawbacks in terms of potentially hurting the performance of the task at hand [23], or poorly approximating flatness of a model [24].

Therefore, in this chapter, a method that can effectively exploit the benefits of flat minima in CL is our goal, which drives us to the following contributions:

- Investigating SAM on replay-based approach, the commonly used ingredient of a competitive CIL solver, under various scenarios including TIL, CIL, and DIL.
- Showing that there may exist conflicts between SAM gradients of two loss terms representing new task learning and old task replaying, respectively. Then we propose to align them using a simple gradient aggregation strategy, which subsequently improves CIL performance consistently and significantly.

## 4.2 Methodology

In this section, we will first present the direct application of SAM on a simple memory-based CL method to enable the flatness of the learned model. Next, we will show our conjecture about the potential conflicts between SAM gradients of the loss terms, which might hinder the model from finding a region where all losses converge to a flat local minimum. Finally, our proposed solution, SAM-CL, which aims to reduce these conflicts, will be presented.

### 4.2.1 Sharpness Aware Minimization for Memory-based Methods

As aforementioned, our focus is the benefit of SAM on a memory replay method whose loss function consists of the core terms on which many complex and sophisticated methods are based. In particular, our choice for the objective is:

$$\mathcal{L}_{total} := \mathcal{L}_{D_t}(\mathbf{w}) + \alpha \mathcal{L}_B(\mathbf{w}) + \beta \|f(x, \boldsymbol{\theta}) - f(x, \boldsymbol{\theta}_{t-1}^*)\|_2^2, \quad (4.1)$$

where  $\alpha, \beta$  are two weighting terms. Note that here we use the  $l_2$  regularization between penultimate layers of two models  $\mathbf{w}$  and  $\mathbf{w}_{t-1}^*$ , but the idea can be applied flexibly for other types of loss, such as their cosine similarity, or KL divergence between two output layers.

To apply SAM, the most direct way is to treat  $\mathcal{L}_{total}$  as the loss function  $\mathcal{L}_D$  in Eq. (2.9) to find the SAM gradient, then gradient descent update the model:

$$\begin{aligned} \boldsymbol{\epsilon}^* &= \rho \cdot \frac{\nabla_{\mathbf{w}} \mathcal{L}_{total}(\mathbf{w})}{\|\nabla_{\mathbf{w}} \mathcal{L}_{total}(\mathbf{w})\|_2}, \\ \mathbf{g}^{SAM} &= \nabla_{\mathbf{w}} \mathcal{L}_{total}(\mathbf{w})|_{\mathbf{w}+\boldsymbol{\epsilon}^*}, \\ \mathbf{w} &= \mathbf{w} - \eta \mathbf{g}^{SAM}, \end{aligned} \quad (4.2)$$

where  $\eta > 0$  is the learning rate. As a result, the objective value  $\mathcal{L}_{total}$  of the obtained model will not change significantly against small local changes in the parameter space, i.e. flat minima, thus enhancing CL performance, which has been proven theoretically and empirically in previous works [23], [52]. However, we argue that this enhancement, brought by the direct approach, can still be further improved if we take into account the directions of gradients of each loss term (currently only their magnitudes are considered through  $\alpha$  and  $\beta$ ). This is because these gradients define where the model will update to reach flat minima. Supposing the gradient of one term dominates or cancels out those of the others, its flat minima

may be favored over other terms'. Thus, the obtained minima may not be evenly flat for each loss term, or for current and buffer data in our case. Our formal conjecture for this hidden risk of the direct approach will be presented below.

## 4.2.2 Potential Conflicts of Worst-case Gradients

To begin with, let us re-write  $\mathbf{g}^{SAM}$  in a more intuitive way using first-order Taylor expansion:

$$\begin{aligned}\mathbf{g}^{SAM} &= \nabla_{\mathbf{w}} \mathcal{L}_{total}(\mathbf{w})|_{\mathbf{w}+\epsilon^*} \\ &\approx \nabla_{\mathbf{w}} [\mathcal{L}_{total}(\mathbf{w}) + \langle \epsilon^*, \nabla_{\mathbf{w}} \mathcal{L}_{total}(\mathbf{w}) \rangle] \\ &= \nabla_{\mathbf{w}} \left[ \mathcal{L}_{total}(\mathbf{w}) + \rho \left\langle \frac{\nabla_{\mathbf{w}} \mathcal{L}_{total}(\mathbf{w})}{\|\nabla_{\mathbf{w}} \mathcal{L}_{total}(\mathbf{w})\|_2}, \nabla_{\mathbf{w}} \mathcal{L}_{total}(\mathbf{w}) \right\rangle \right] \\ &= \nabla_{\mathbf{w}} [\mathcal{L}_{total}(\mathbf{w}) + \rho \|\nabla_{\mathbf{w}} \mathcal{L}_{total}(\mathbf{w})\|_2] \tag{4.3}\end{aligned}$$

$$= \nabla_{\mathbf{w}} [\mathcal{L}_{D_t}(\mathbf{w}) + \mathcal{L}_B(\mathbf{w}) + \rho \|\nabla_{\mathbf{w}} \mathcal{L}_{D_t}(\mathbf{w}) + \nabla_{\mathbf{w}} \mathcal{L}_B(\mathbf{w})\|_2]. \tag{4.4}$$

Equation (4.3) shows that the SAM gradient will follow two directions to optimize the model: (i) direction that minimizes the original loss  $\mathcal{L}(\mathbf{w})$ , and (ii) direction that minimizes norm of the gradient. The first direction accounts for finding regions with low loss value, and the second one finds flat regions, thus together leading the model to the flat minima.

Considering our case where the total loss contains two loss terms: one aims to find local flat minima w.r.t current data, and the other seeks for those minima w.r.t buffer data (the second and third terms are combined as they are all computed on buffer). Combined with the above analysis and from Equation (4.4), it can be seen that each loss term has two directions, low-loss and flat regions, to follow, making the total objective a four-target function whose flat minima may be hard to find. Therefore, we hypothesize, and will empirically validate in the experiment section, that there may exist gradient conflict between the two terms so that the process in Eq. (4.2) might not guarantee finding the common flat minima for both current and buffer data. Such a case can happen when there exists at least one gradient that interferes or cancels out the others. For example, if the current task is about classifying dogs and cars, then the model may take the four-legged feature to recognize a dog. However, that feature may fail to classify two past classes: cats and deer as they are both four-legged animals. Thus if the model follows directions that benefit current data, it may harm the performance of previous tasks.

### 4.2.3 Sharpness and Gradient Aware Minimization for Memory-based Methods

To circumvent the aforementioned pitfall, we propose to separately compute SAM gradients of each loss term, then aggregate them such that their conflicts are minimized, and finally update the model with the aggregated gradient. By doing this, the model is encouraged to learn features that are useful for all tasks. We name our method as SGAM-CL (SGAM short for Sharpness and Gradient Aware).

In this way, our method’s objective is:

$$\min_{\mathbf{w}} \left\{ \max_{\|\epsilon_1\|_2 \leq \rho} \mathcal{L}_D(\mathbf{w} + \epsilon_1) + \alpha \max_{\|\epsilon_2\|_2 \leq \rho} \mathcal{L}_B(\mathbf{w} + \epsilon_2) + \beta \max_{\|\epsilon_3\|_2 \leq \rho} KD_B(\mathbf{w} + \epsilon_3) \right\} \quad (4.5)$$

where  $\rho$  is the radius perturbation of the current parameter  $\mathbf{w}$ .

With a symbol abuse  $\mathcal{L}$  for  $KD$ , our method can be done with the steps described as follows: For each loss term  $\mathcal{L}_i$ :

$$\epsilon_i^* = \rho \cdot \frac{\nabla_{\mathbf{w}} \mathcal{L}_i(\mathbf{w})}{\|\nabla_{\mathbf{w}} \mathcal{L}_i(\mathbf{w})\|_2}, \quad \mathbf{g}_i^{SAM} = \nabla_{\mathbf{w}} \mathcal{L}_i(\mathbf{w})|_{\mathbf{w}+\epsilon_i^*}. \quad (4.6)$$

Then apply PCGrad on these SAM gradients to get the final gradient for the model update:

$$\begin{aligned} \bar{\mathbf{g}} &= PCGrad(\mathbf{g}_1^{SAM}, \mathbf{g}_2^{SAM}, \mathbf{g}_3^{SAM}), \\ \mathbf{w} &= \mathbf{w} - \eta \bar{\mathbf{g}}. \end{aligned} \quad (4.7)$$

In our method, we choose PCGrad as a representative gradient aggregation. We note that our method is agnostic to the application of other aggregation strategies. We leave further exploration on the impact of different aggregation strategies on CL for future work.

Our method, SGAM-CL, is summarized in Algorithm 3.

### 4.2.4 Discussion

We have proposed a direct application of SAM on a simple memory-based baseline which can boost CL performance due to the flatness of the loss landscape. We have further hypothesized the potential risk of SAM gradient conflicts and proposed to apply one gradient aggregation strategy to tackle this risk. Regarding the direct approach, it is worth noting that Deng et al. [13] also investigated the impact of the loss landscape’s flatness in the context of CL, and applied SAM to their



---

**Algorithm 3** SGAM-CL

---

**Input:** Model parameter  $\mathbf{w} = \{\boldsymbol{\theta}, \phi_{1:t}\}$ , perturbation radius  $\rho$ , step size  $\eta$ , current dataset  $D_t$ , buffer  $B = \{M_i\}_{i=1}^{t-1}$ .

**Output:** Updated parameter  $\boldsymbol{\theta}^*, \phi^*$

- 1: **for**  $bx \in D_t$  **do**
  - 2:   - Sample  $obx$  samples from  $B$
  - 3:   - Objective function:  
     $\mathcal{L}_{total} = \mathcal{L}_{bx}(\boldsymbol{\theta}, \phi_t) + \mathcal{L}_{obx}(\boldsymbol{\theta}, \phi_{1:t}) + \text{KD}_{obx}(\boldsymbol{\theta}, \boldsymbol{\theta}_{t-1}^*)$ .
  - 4:   - For each term in the loss, calculate the SAM gradient:  $\mathbf{g}_1^{SAM}, \mathbf{g}_2^{SAM}, \mathbf{g}_3^{SAM}$ .
  - 5:   - Gradient aggregation:  $\bar{\mathbf{g}} = \text{PCGrad}(\mathbf{g}_1^{SAM}, \mathbf{g}_2^{SAM}, \mathbf{g}_3^{SAM})$
  - 6:   - Update model:  $\mathbf{w} = \mathbf{w} - \eta \bar{\mathbf{g}}$
  - 7:   - Add data  $D_t$  to buffer  $B$  with reservoir sampling
  - 8: **end for**
  - 9: **return**
- 

method. However, they focused on GPM, a gradient-based method, and reported accuracy in the task-incremental setting. Therefore, from their work, it is not clear how beneficial SAM is for a more general memory-based method under a more realistic evaluation scenario, class-incremental learning. Meanwhile, we provide experimental results for both mentioned scenarios.

Regarding the application of PCGrad, we assume that the model from the most recent task,  $\mathbf{w}_{t-1}^*$ , has been well-trained to effectively resolve SAM gradient conflicts among past classes. In other words,  $\mathbf{w}_{t-1}^*$  is an optimal flat solution for all tasks so far, not favoring any particular task. Under this assumption, we only need to mitigate gradient disagreement between the loss of current data and the loss(es) of buffer data. This is practically beneficial as if we individually consider all  $t - 1$  old tasks, represented by  $t - 1$  terms in the total loss, the calculation of each task's gradient will cause great computational and memory complexity.

In terms of time limitations, our method takes around 3.5 times as much as a memory-replay baseline does (no SAM and no PCGrad). In particular, similar to SAM, our method first performs one forward-backward pass for each term in the loss function, then another forward-backward pass to compute their SAM gradients, and finally aggregates them. Regarding space complexity, for each mini-batch update, our method requires three more times spaces to store gradients of the three loss terms.

### 4.2.5 Theoretical Analysis

In this section, we present two bounds that are derived from [1] to show that by minimizing the sharpness of the empirical buffer losses in the objective function 4.1, SGAM minimizes the upper bounds of such losses on the original training datasets. For simplicity, we consider learning a sequence of two tasks whose training datasets are  $D_1$  and  $D_2$ , respectively. After learning task 1, a memory buffer  $B$  is randomly sampled from  $D_1$ :  $B \sim D_1$ . Then the theorem is informally stated as follows.

**Theorem 1.** *For any perturbation radius  $\rho > 0$ , under some mild conditions, with probability  $1 - \delta$  over the choice of drawing  $B$ , we have:*

$$\mathcal{L}_{D_1}(\mathbf{w}) \leq \max_{\|\epsilon_2\|_p \leq \rho} \mathcal{L}_B(\mathbf{w}) + h(\|\mathbf{w}\|_2^2), \quad (4.8)$$

$$KD_{D_1}(\mathbf{w}) \leq \max_{\|\epsilon_3\|_p \leq \rho} KD_B(\mathbf{w}) + h(\|\mathbf{w}\|_2^2), \quad (4.9)$$

where  $h : \mathbb{R}_+ \rightarrow \mathbb{R}_+$  is a strictly increasing function.

Theorem 1 shows the relationship between the cross-entropy and knowledge distillation losses computed on the full training set  $D_1$  and their counterparts that are computed on the buffer  $B$ . It states that minimizing the worst-case of each loss on the buffer can minimize that loss on the full data. This justifies our method’s objective 4.5 can mitigate catastrophic forgetting. Proof for the theorem can be found in the Appendix.

## 4.3 Experiments

**Datasets.** In our experiments, we apply the proposed method to a range of widely used continual learning datasets, including sequential CIFAR-10, sequential Tiny ImageNet, Permuted MNIST, and Rotated MNIST. We construct sequences of tasks for them as follows: Sequential-CIFAR-10 (S-CIFAR-10) contains 5 sequential tasks, each involving 2 classes. Sequential-Tiny-ImageNet (S-Tiny-ImageNet) divides the 100 classes of Tiny ImageNet into 20 tasks, each consisting of 5 classes. Permuted MNIST (P-MNIST) and Rotated MNIST (R-MNIST) consist of 20 subsequent tasks; the former shuffles pixel positions using random permutations, while the latter rotates input images by random angles within the  $[0; \pi)$  range.

**Architectures.** We utilize a standard ResNet18 architecture for CIFAR-10 and Tiny ImageNet without pretraining; A fully connected network with two hidden

layers for the Permuted and Rotated MNIST. This is consistent with [12]. All networks use the ReLU activation function, and the classification loss is cross-entropy loss on the softmax layer.

**Experimental settings.** S-CIFAR-10 and S-Tiny-ImageNet are used in CIL and TIL settings, while P-MNIST and R-MNIST are for DIL. All experiments use Stochastic Gradient Descent (SGD) to optimize the model. We set the number of epochs to 50 and 100 for the first two datasets, respectively, and 1 for the other two. In each mini-batch update, we randomly sample from the buffer with the number of samples equal to that of the current mini-batch. The buffer is updated using Reservoir Sampling [9] at each of each task for S-CIFAR-10 and S-Tiny-ImageNet, and at each of each mini-batch update for the two MNIST datasets.

**Evaluation protocol.** We adopt the commonly used metric to measure the overall performance of a CL method, Average Accuracy:  $ACC$  as defined in (2.2).

**Baselines.** We compare SGAM-CL with several baselines in three main CL approaches, including: Regularization-based: EWC [5], SI [7], LwF [6]; Architecture-based: PNN [3]; and Memory-based: ER [9], GEM [46], A-GEM [8], iCaRL [10], HAL [11], DER [12]. We include sequential training without any CL techniques (SGD) and joint training with data from all tasks (JOINT) as lower-bound and upper-bound, respectively. We also compare SGAM-CL with F2M and CPR, two flat-seeking methods used in CL. For a fair comparison, we also incorporate them into the same baseline that SGAM-CL uses (Eq.(4.1)), namely ER-F2M and ER-CPR.

**Hyper-parameters.** We use the reported results from [12] as we follow their experimental settings. For ER-F2M [24] and ER-CPR [23], we tune  $\alpha$  and  $\beta$  in  $\{0, 0.1, 0.3, 0.5, 1.0\}$ . We also tune the number of layers to which F2M injects noises in  $\{2, 3, 4\}$ , and the bound of these noises in  $\{0.001, 0.01\}$ , and the weight term enforcing flatness used by CPR in  $\{0.02, 0.05, 0.1, 0.2, 0.5\}$ . For SGAM-CL, we tune the radius perturbation  $\rho$  in  $\{0.01, 0.05, 0.1, 0.5, 1.0\}$ . Details for the chosen hyper-parameters can be found in Table 1 in Appendix C.

### 4.3.1 Experimental Analysis

**Main results.** Table 4.1 shows the average accuracy of our proposed method and baselines on standard CL benchmarks. For the different values of buffer size, our algorithm SGAM-CL which aims to find the common flat optimal regions for the proposed terms in the objective function obtains the best results. Notably, on S-CIFAR-10 and S-Tiny-ImageNet, SGAM-CL outperforms the best baseline, by a large margin with a buffer size of 200 and 500 under the CIL setting. With S-

**Table 4.1:** Classification results for standard CL benchmarks. ‘-’ means the experiments were not feasible due to: intractable training time (GEM, HAL on S-Tiny-ImageNet), no access to task-id when testing (PNN under CIL), incompatibility under DIL (ICaRL, PNN, LwF). Bold font denotes best results; Italic font denotes results of CPR and F2M.

Buffer	Method	S-CIFAR-10		S-Tiny-ImageNet		P-MNIST	R-MNIST
		CIL( $\uparrow$ )	TIL( $\uparrow$ )	CIL( $\uparrow$ )	TIL( $\uparrow$ )	DIL( $\uparrow$ )	DIL( $\uparrow$ )
-	JOINT	92.20 $\pm$ 0.15	98.31 $\pm$ 0.12	59.99 $\pm$ 0.19	82.04 $\pm$ 0.10	94.33 $\pm$ 0.17	95.76 $\pm$ 0.04
	SGD	19.62 $\pm$ 0.05	61.02 $\pm$ 3.33	7.92 $\pm$ 0.26	18.31 $\pm$ 0.68	40.70 $\pm$ 2.33	67.66 $\pm$ 8.53
	oEWC [5]	19.49 $\pm$ 0.12	68.29 $\pm$ 3.92	7.58 $\pm$ 0.10	19.20 $\pm$ 0.31	<b>75.79<math>\pm</math>2.25</b>	<b>77.35<math>\pm</math>5.77</b>
	SI [7]	19.48 $\pm$ 0.17	68.05 $\pm$ 5.91	6.58 $\pm$ 0.31	36.32 $\pm$ 0.13	65.86 $\pm$ 1.57	71.91 $\pm$ 8.3
	LwF [6]	<b>19.61<math>\pm</math>0.05</b>	63.29 $\pm$ 2.35	<b>8.46<math>\pm</math>0.22</b>	15.85 $\pm$ 0.58	-	-
-	PNN [3]	-	<b>95.13<math>\pm</math>0.72</b>	-	<b>67.84<math>\pm</math>0.29</b>	-	-
200	ER [9]	44.79 $\pm$ 1.86	91.19 $\pm$ 0.94	8.49 $\pm$ 0.16	38.17 $\pm$ 2.00	72.37 $\pm$ 0.87	85.01 $\pm$ 1.90
	GEM [46]	25.54 $\pm$ 0.76	90.44 $\pm$ 0.94	-	-	66.93 $\pm$ 1.25	80.80 $\pm$ 1.15
	A-GEM [8]	20.04 $\pm$ 0.34	83.88 $\pm$ 1.49	8.07 $\pm$ 0.08	22.77 $\pm$ 0.03	66.42 $\pm$ 4.00	81.91 $\pm$ 0.76
	iCaRL [10]	49.02 $\pm$ 3.20	88.99 $\pm$ 2.13	7.53 $\pm$ 0.79	28.19 $\pm$ 1.47	-	-
	HAL [11]	32.36 $\pm$ 2.70	82.51 $\pm$ 3.20	-	-	74.15 $\pm$ 1.65	84.02 $\pm$ 0.98
	DER [12]	61.93 $\pm$ 1.79	91.40 $\pm$ 0.92	11.87 $\pm$ 0.78	40.22 $\pm$ 0.67	81.74 $\pm$ 1.07	90.04 $\pm$ 2.61
	DER++ [12]	64.88 $\pm$ 1.17	91.92 $\pm$ 0.60	10.96 $\pm$ 1.17	40.87 $\pm$ 1.16	83.58 $\pm$ 0.59	90.43 $\pm$ 1.87
	F2M-DER++	<u>65.36<math>\pm</math>1.76</u>	<u>92.25<math>\pm</math>0.58</u>	<u>11.66<math>\pm</math>0.93</u>	<u>40.30<math>\pm</math>1.56</u>	<u>83.76<math>\pm</math>0.27</u>	<u>90.66<math>\pm</math>0.70</u>
	CPR-DER++	<u>66.17<math>\pm</math>1.12</u>	<u>91.78<math>\pm</math>0.25</u>	<u>11.54<math>\pm</math>0.47</u>	<u>41.00<math>\pm</math>0.21</u>	<u>83.69<math>\pm</math>0.79</u>	<u>90.49<math>\pm</math>1.40</u>
	<b>SGAM-CL (Ours)</b>	<b>70.44<math>\pm</math>0.18</b>	<b>93.95<math>\pm</math>0.15</b>	<b>21.02<math>\pm</math>0.40</b>	<b>56.41<math>\pm</math>1.75</b>	<b>84.70<math>\pm</math>0.96</b>	<b>90.78<math>\pm</math>0.27</b>
500	ER[9]	57.74 $\pm$ 0.27	93.61 $\pm$ 0.27	9.99 $\pm$ 0.29	48.64 $\pm$ 0.46	80.60 $\pm$ 0.86	88.91 $\pm$ 1.44
	GEM [46]	26.20 $\pm$ 1.26	92.16 $\pm$ 0.69	-	-	76.88 $\pm$ 0.52	81.15 $\pm$ 1.98
	A-GEM [8]	22.67 $\pm$ 0.57	89.48 $\pm$ 1.45	8.06 $\pm$ 0.04	25.33 $\pm$ 0.49	67.56 $\pm$ 1.28	80.31 $\pm$ 6.29
	iCaRL [10]	47.55 $\pm$ 3.95	88.22 $\pm$ 2.62	9.38 $\pm$ 1.53	31.55 $\pm$ 3.27	-	-
	HAL [11]	41.79 $\pm$ 4.46	84.54 $\pm$ 2.36	-	-	80.13 $\pm$ 0.49	85.00 $\pm$ 0.96
	DER [12]	70.51 $\pm$ 1.67	93.40 $\pm$ 0.39	17.75 $\pm$ 1.14	51.78 $\pm$ 0.88	87.29 $\pm$ 0.46	92.24 $\pm$ 1.12
	DER++ [12]	72.70 $\pm$ 1.36	93.88 $\pm$ 0.50	19.38 $\pm$ 1.41	51.91 $\pm$ 0.68	88.21 $\pm$ 0.39	92.77 $\pm$ 1.05
	F2M-DER++	<u>72.85<math>\pm</math>1.32</u>	<u>93.95<math>\pm</math>0.72</u>	<u>19.40<math>\pm</math>0.82</u>	<u>51.48<math>\pm</math>0.77</u>	<u>88.46<math>\pm</math>0.36</u>	<u>92.90<math>\pm</math>0.17</u>
	CPR-DER++	<u>73.21<math>\pm</math>1.43</u>	<u>94.11<math>\pm</math>0.65</u>	<u>19.04<math>\pm</math>0.74</u>	<u>53.46<math>\pm</math>0.42</u>	<u>88.65<math>\pm</math>0.25</u>	<u>92.81<math>\pm</math>1.03</u>
	<b>SGAM-CL (Ours)</b>	<b>76.26<math>\pm</math>1.02</b>	<b>95.25<math>\pm</math>0.41</b>	<b>26.38<math>\pm</math>0.12</b>	<b>62.59<math>\pm</math>0.50</b>	<b>88.94<math>\pm</math>0.69</b>	<b>93.03<math>\pm</math>0.17</b>
5120	ER [9]	82.47 $\pm$ 0.52	96.98 $\pm$ 0.17	27.40 $\pm$ 0.31	67.29 $\pm$ 0.23	89.90 $\pm$ 0.13	93.45 $\pm$ 0.56
	GEM [46]	25.26 $\pm$ 3.46	95.55 $\pm$ 0.02	-	-	87.42 $\pm$ 0.95	88.57 $\pm$ 0.40
	A-GEM [8]	21.99 $\pm$ 2.29	90.10 $\pm$ 2.09	7.96 $\pm$ 0.13	26.22 $\pm$ 0.65	73.32 $\pm$ 1.12	80.18 $\pm$ 5.52
	iCaRL [10]	55.07 $\pm$ 1.55	92.23 $\pm$ 0.84	14.08 $\pm$ 1.92	40.83 $\pm$ 3.11	-	-
	HAL [11]	59.12 $\pm$ 4.41	88.51 $\pm$ 3.32	-	-	89.20 $\pm$ 0.14	91.17 $\pm$ 0.31
	DER [12]	83.81 $\pm$ 0.33	95.43 $\pm$ 0.33	36.73 $\pm$ 0.64	69.50 $\pm$ 0.26	91.66 $\pm$ 0.11	94.14 $\pm$ 0.31
	DER++ [12]	85.24 $\pm$ 0.49	96.12 $\pm$ 0.21	39.02 $\pm$ 0.97	69.84 $\pm$ 0.63	92.26 $\pm$ 0.17	94.65 $\pm$ 0.33
	F2M-DER++	<u>85.38<math>\pm</math>0.53</u>	<u>96.19<math>\pm</math>0.75</u>	<u>39.34<math>\pm</math>0.38</u>	<u>69.41<math>\pm</math>0.99</u>	<u>92.19<math>\pm</math>0.10</u>	<u>94.43<math>\pm</math>0.15</u>
	CPR-DER++	<u>86.12<math>\pm</math>0.42</u>	<u>96.11<math>\pm</math>0.38</u>	<u>39.45<math>\pm</math>0.87</u>	<u>69.88<math>\pm</math>0.54</u>	<u>92.11<math>\pm</math>0.11</u>	<u>94.53<math>\pm</math>0.11</u>
	<b>SGAM-CL (Ours)</b>	<b>86.58<math>\pm</math>0.46</b>	<b>97.30<math>\pm</math>0.20</b>	<b>42.19<math>\pm</math>0.52</b>	<b>73.70<math>\pm</math>0.44</b>	<b>92.26<math>\pm</math>0.10</b>	<b>94.93<math>\pm</math>0.27</b>

CIFAR-10 under TIL, the gap is less significant as the model only needs to infer the correct class between two classes (this dataset has two classes per task), but this trend is different with the more challenging dataset S-Tiny-ImageNet: SGAM-CL still boosts TIL performance considerably. Under the DIL scenario, we still obtain better results across all settings than DER, DER++, and two flat versions of DER++, and surpass other baselines significantly. This large can be attributed to the special buffer update scheme of DER that saves a sample’s logits (pre-softmax) with its hard label to the buffer along the training trajectory. By seeking flat minima, we further improve the performance.

As a side note, regularization-based methods, oEWC, SI, and LwF, which do not replay old data, perform poorly, especially under CIL, compared to exemplar-based ones. This suggests that preserving knowledge of a past task through constraining its important parameters is not sufficient.

To sum up, being aware of sharpness for each term in the loss function, SGAM-

CL improves DER++, the exemplar replay baseline using the same objective (Eq. (4.1)), by nearly 6% and 10% on S-CIFAR-10 and S-Tiny-ImageNet under a limited buffer size, respectively.

**SGAM - A better flat seeking optimizer for CL.** Additionally, in comparison with CPR-DER++ and F2M-DER++, whose results are underlined in Table 4.1, SGAM-CL clearly shows its superiority over the other two across all settings, especially under the challenging S-Tiny-ImageNet dataset. Meanwhile, the improvement F2M and CPR bring to DER++ is relatively minor compared to that of SGAM, and in some cases, they even degrade DER++, e.g. P-MNIST and R-MNIST with buffer 5120. Regarding F2M, we follow their experiments to consider 2 random times to ensure feasible training time. This probably results in poor performance of F2M-DER++ compared to SGAM-CL in practice. Concerning CPR, it generally can work better than F2M under CIL but still falls behind SGAM, especially with a limited buffer.

### 4.3.2 Ablation Study

**SAM and Gradient Aggregation.** Table 4.2 presents the effect of different components in our proposed method, SAM and Gradient Aggregation (PCGrad), using S-CIFAR-10 with 200 buffer size. First, it can be observed that the improvement brought to the baseline by SAM is higher by PCGrad. This could be because gradient conflict is not a big problem in the loss terms of the baseline, which seems in line with existing CL methods as they often neglect gradient conflict when minimizing the empirical loss. Nevertheless, when combining both SAM and PCGrad, the model performance is boosted the most.

**Table 4.2:** Ablation studies of SGAM-CL on S-CIFAR-10, 200 buffer size. (-) means ablated, (+) means used.

SAM	Grad Aggr	CIL( $\uparrow$ )	TIL( $\uparrow$ )
-	-	65.13 $\pm$ 0.86	92.23 $\pm$ 0.70
-	+	66.82 $\pm$ 2.85	92.78 $\pm$ 0.41
+	-	69.64 $\pm$ 0.06	93.79 $\pm$ 0.02
+	+	<b>70.44<math>\pm</math>0.18</b>	<b>93.95<math>\pm</math>0.15</b>

**SGAM-CL and SAM-CL.** In Section 4.2.2, we have conjectured that there might exist conflicts between the SAM gradient of each loss term, and thus the solution found by SAM-CL may not be flat for both current and old data. Therefore, a step of resolving gradient conflicts is added before the final model update step can improve the solution. We will empirically verify this conjecture and show that SGAM-CL does improve SAM-CL.

**Table 4.3:** Comparison between SGAM-CL and SAM-CL on S-CIFAR-10 and S-Tiny-Imagenet.

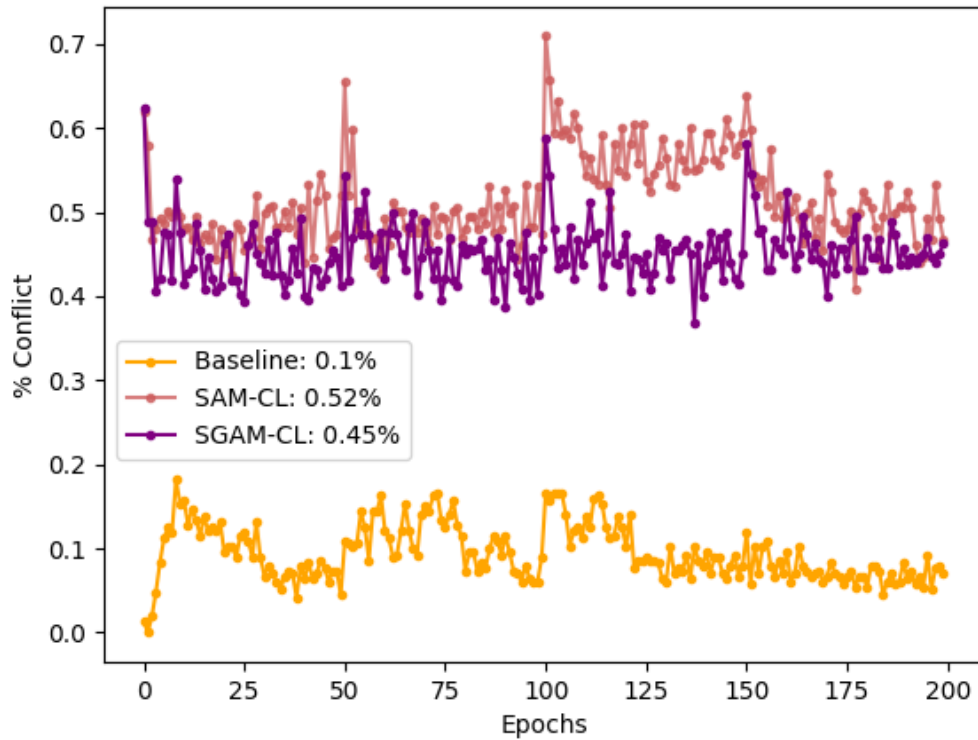
Dataset	Buffer size	SAM-CL		SGAM-CL	
		CIL( $\uparrow$ )	TIL( $\uparrow$ )	CIL( $\uparrow$ )	TIL( $\uparrow$ )
S-CIFAR-10	200	69.64	93.79	<b>70.44</b>	<b>93.95</b>
	500	74.46	95.12	<b>76.26</b>	<b>95.25</b>
	5120	85.57	96.94	<b>86.58</b>	<b>97.30</b>
S-Tiny-ImageNet	200	20.26	53.09	<b>21.02</b>	<b>56.41</b>
	500	25.27	60.30	<b>26.38</b>	<b>62.59</b>
	5120	40.63	72.56	<b>42.19</b>	<b>73.70</b>

First, let us define the *conflict number* in one training epoch: the number of mini-batch updates that have negative cosine similarity between the loss term on current data and one on buffer data. Then *conflict ratio* equals the ratio between *conflict number* and the total number of updates in one epoch. We plot this ratio along the 5-task training on S-CIFAR-10 of the baseline, SAM-CL, and SGAM-CL in Figure 4.1. It can be seen that the *conflict ratio* of SAM-CL is consistently higher than that of the baseline, showing that there is more conflict in SAM-CL than in Baseline (0.52% vs 0.1%). After aggregating gradients, SGAM-CL does decrease the conflict (0.45%), but in comparison with Baseline, there still exists a large gap. Despite this, SGAM-CL still outperforms Baseline, as was shown in Table 4.2. Therefore, it is interesting to investigate whether further closing this gap will result in a better SGAM-CL, which we leave for future work.

Second, we show that the solution obtained by SAM-CL is less robust against small parameter perturbations than SGAM-CL. Following [12], we add zero-mean Gaussian noise with increasing variance to the optimal parameters learned by the two methods, then visualize the average training loss and accuracy of all tasks, as can be seen from Figure 4.2. Clearly, SGAM-CL shows stronger robustness as its training loss is always smaller than that of SAM-CL, and similarly with the training accuracy, except for the final perturbation.

Third, we present the full average accuracy of SAM-CL and SGAM-CL for S-CIFAR-10 and S-Tiny-ImageNet with various buffer sizes in Table 4.3. These consistent improvements of SGAM-CL over SAM-CL further prove that being aware of gradient congruence does benefit sharpness-aware minimization in memory-based CL methods.

**SGAM-CL’s robustness vs other baselines.** Finally, we visualize model sensitivity to local perturbations of the proposed method and other baselines in Figure 4.3. It can be seen that among methods that explicitly seek flat minima (F2M-ER, CPR-ER, F2M-DER++, CPR-DER++), SGAM-CL possesses a higher tolerance to perturbations, showing that it has converged to a flatter local minima than the oth-

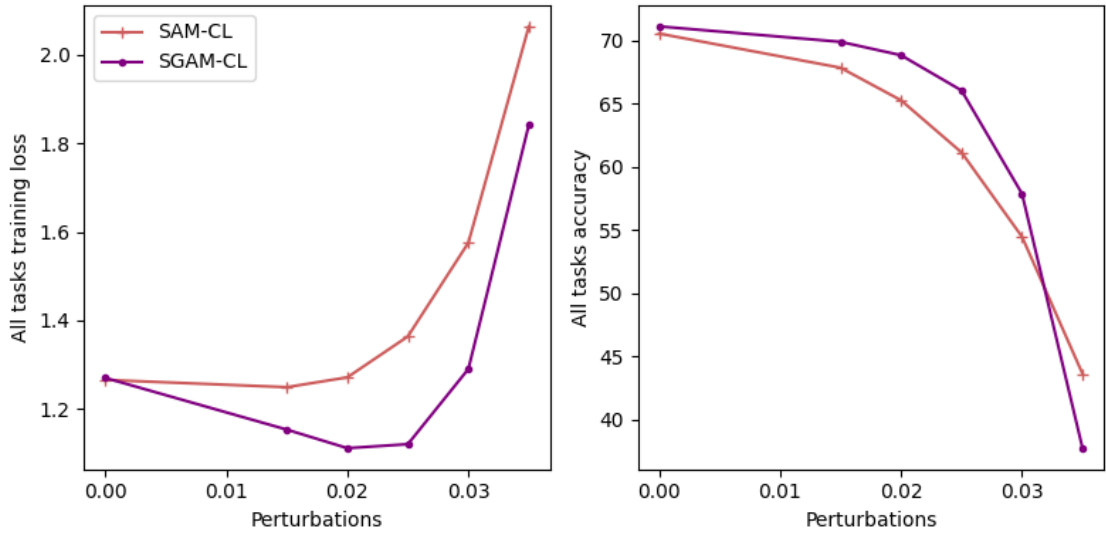


**Figure 4.1:** The ratio of conflict in one epoch along 5-task-training of S-CIFAR-10, and the average ratio of Baseline, SAM-CL and SGAM-CL.

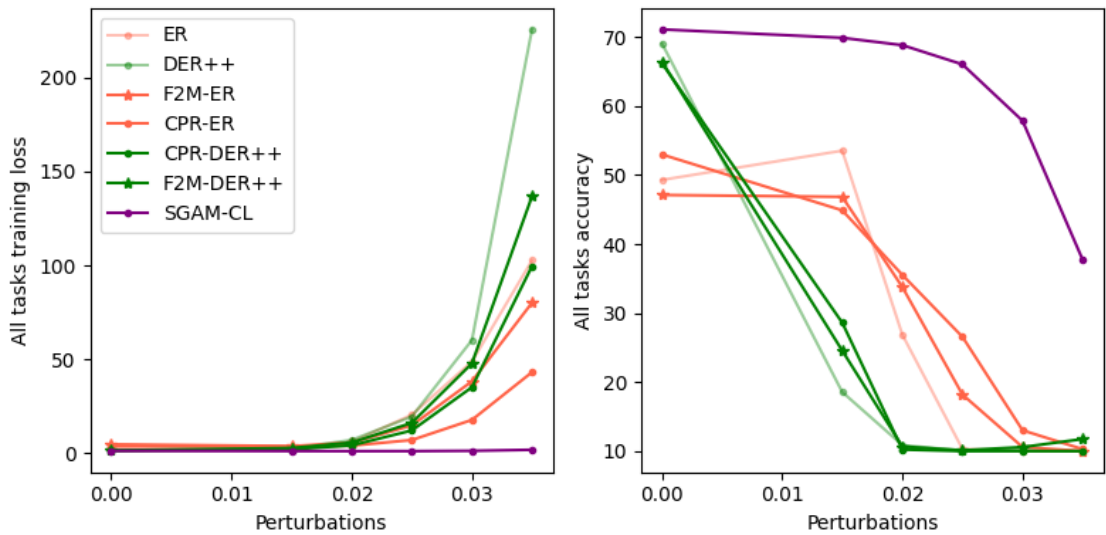
ers have. This positive correlation between flat minima and CL performance is in line with previous observations in [12], [23].

## 4.4 Summary

We have proposed a method that can significantly boost the performance of a simple memory-based baseline by exploiting a well-known approach: flatness enforcement. In particular, we leverage the effectiveness of SAM to make the model less overfitting, especially on a limited buffer. Moreover, we present a hypothesis on potential gradient conflict between loss terms corresponding to new and old data, from which a gradient-aware version of SAM is proposed to further improve CL performance. Extensive experiments and analyses on benchmarks confirm our hypothesis and the effectiveness of our method. Despite these promising results, better ones could be obtained for example by considering variations of SAM and gradient aggregation methods, or further studying the conflict gap between SGAM-CL and the baseline counterpart. The latter could be an interesting direction for future work.



**Figure 4.2:** Loss (left) and Accuracy (right) changes w.r.t weight perturbation of SGAM-CL and SAM-CL.



**Figure 4.3:** Loss (left) and Accuracy (right) changes w.r.t weight perturbation of SAM-CL and several baselines.



## Chapter 5. Projected Worst-case Perturbation may put Old and New tasks at Odds

In the previous chapter, we have studied the benefits and the potential gradient conflict when applying SAM to a memory-based continual learning method under class incremental learning scenarios. In this chapter, we will turn our focus to SAM’s application in gradient-projection memory methods, which has shown to be an effective task incremental learner when it can guarantee the least forgetting compared to its regularization-based counterpart. Specifically, methods in this group generally modify the gradient update of the current task such that minimal interference with previous task knowledge is obtained. This is done by forcing the direction of the current gradient to be orthogonal to the subspace spanned by previous tasks’ features, thus obtaining *invariant feature across tasks*. To further improve GPMs, one recent work, FS-GPM [13], aims to incorporate sharpness minimization in the process of projecting gradient by finding the worst case perturbation for the current task data in the subspace of old task features, i.e., *projected worst-case perturbation*. We will show mathematically that this scheme may cause conflict between gradients of old and current tasks. In addition, this worst-case projection can also hinder SAM from freely finding flat minima for the current task in case it has no interference with previous ones, i.e., its feature subspace is orthogonal to that of the previous. We then overcome these drawbacks of FS-GPM with a simple solution to remove the gradient conflict and to ensure the feasibility of SAM in finding the worst-case perturbation.

### 5.1 Introduction

The connection between wide local minima of a neural network and its continual learning performance has been established in several works [13], [23], [24], which we call as the flatness-based approach. As a pioneering work, Cha et al. [23] show that a model that converges to flatter minima tends to find common good regions for all tasks more easily and one that converges to sharp minima. In other words, this could benefit plasticity while maintaining stability, i.e., the ability to learn new tasks and retain old ones, respectively, of a CL model. The intuition is that the search space for the next task becomes larger when the model has learned the previous task with wide minima. However, converging to flat minima truly benefits only when knowledge of previous tasks is kept unchanged to a great extent. This is

because if the performance of the first task has significantly dropped after several tasks, then finding flat minima for later tasks will not be enough to recover the first one. For this reason, one could apply flat-seeking minimizers for architecture-based methods, which can guarantee no forgetting by expanding the network for learning new tasks when needed. However, this option will come with the cost of a large memory burden, thus limiting its practicality.

Recently, Deng et al. [13] also follow this line of flat-based works by incorporating sharpness minimization into gradient projection memory [14], a gradient-based CL method that can effectively retain learned knowledge while keeping the network’s size fixed. They observe that after learning a sequence of tasks, a CL method favoring stability tends to make the loss landscape of old tasks flatter than that favoring plasticity; and one favoring the latter will have a flatter landscape of the newly learnt task than one favoring the former. Therefore, they suggest that if one could make GPM converge to flatter minima after learning each task, better CL performance could be achieved, and propose FS-GPM.

However, we find two hidden problems in FS-GPM that in fact can weaken the stability-plasticity balance they are trying to obtain. First, by projecting the adversarial perturbation corresponding to current task data onto the subspace of old tasks’ features, then computing the loss at the perturbed weights using both current and buffer data, FS-GPM inadvertently creates a loss term that provokes an obtuse angle between gradients of the two data. This will make learning a new task interfere remaining old one, hence knowledge transfer may be hindered. Second, we also point out that FS-GPM may fail to find flat minima for the current task when this task does not incur forgetting of learned one.

From the aforementioned analyses, we propose a simple solution that: (i) allows SAM to flat minima for buffer data; (ii) removes the hidden gradient conflict between old and new data; (iii) explicitly aligns these gradients; and (iv) allows SAM to reach its full potential in finding flat minima. Empirical results further verify our findings and our method’s effectiveness in reducing forgetting.

## 5.2 Methodology

### 5.2.1 Two potential problems of FS-GPM

This section provides our in-depth analysis of two potential problems of FS-GPM [13] in terms of gradient conflict and a failure case of SAM, which consequently can be opposite to its initial goals: to better balance stability and plasti-

city of GPM [14].

Recall that FS-GPM first finds the worst-case perturbation in the subspace spanned by old tasks, then updates the model from this with the adversarial weights corresponding to this perturbation using both data from the current task and buffer.

$$\epsilon \leftarrow \epsilon + \eta_1 (\mathbf{S}\mathbf{\Lambda}\mathbf{S}^T) \nabla_{\mathbf{w}} \mathcal{L}_{D_t}(\mathbf{w})|_{\mathbf{w}+\epsilon}, \quad (5.1)$$

$$\mathbf{w} \leftarrow \mathbf{w} - \eta_3 (\mathbf{I} - \mathbf{S}\mathbf{\Lambda}\mathbf{S}^T) \nabla_{\mathbf{w}} \mathcal{L}_{D_t \cup B_t}(\mathbf{w})|_{\mathbf{w}+\epsilon}. \quad (5.2)$$

Without loss of generality, consider one inner step of FS-GPM, cf. Eq. 5.1, with  $\mathbf{w}$  is the current model parameters, we have the adversarial update projected onto  $\mathbf{S}$  as:

$$\epsilon_{\parallel} := \eta_1 \mathbf{S}\mathbf{\Lambda}\mathbf{S}^T \nabla_{\mathbf{w}} \mathcal{L}_{D_t}(\mathbf{w}) \quad (5.3)$$

Then FS-GPM's final gradient, which is computed on both current and buffer data, can be expanded using First-order Taylor expansion as:

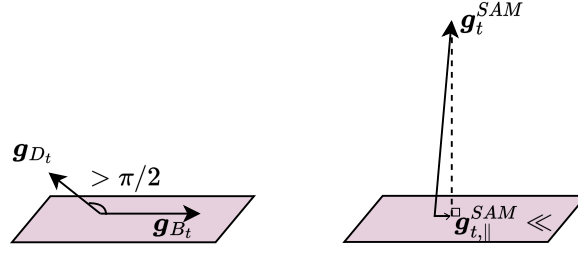
$$\begin{aligned} \nabla_{\mathbf{w}} \mathcal{L}_{D_t \cup B_t}(\mathbf{w})|_{\mathbf{w}+\epsilon_{\parallel}} &\approx \nabla_{\mathbf{w}} \left[ \mathcal{L}_{D_t \cup B_t}(\mathbf{w}) + \epsilon_{\parallel}^T \nabla_{\mathbf{w}} \mathcal{L}_{D_t \cup B_t}(\mathbf{w}) \right] \\ &= \frac{1}{2} \nabla_{\mathbf{w}} \left[ \mathcal{L}_{D_t}(\mathbf{w}) + \mathcal{L}_{B_t}(\mathbf{w}) + \epsilon_{\parallel}^T \nabla_{\mathbf{w}} \mathcal{L}_{D_t}(\mathbf{w}) + \epsilon_{\parallel}^T \nabla_{\mathbf{w}} \mathcal{L}_{B_t}(\mathbf{w}) \right] \end{aligned} \quad (5.4)$$

First, substitute Eq. (5.3) to the last term in Eq. (5.4), we have

$$\begin{aligned} \epsilon_{\parallel}^T \nabla_{\mathbf{w}} \mathcal{L}_{B_t}(\mathbf{w}) &= \eta_1 (\mathbf{S}\mathbf{\Lambda}\mathbf{S}^T \nabla_{\mathbf{w}} \mathcal{L}_{D_t}(\mathbf{w}))^T \nabla_{\mathbf{w}} \mathcal{L}_{B_t}(\mathbf{w}) \\ &= \eta_1 (\nabla_{\mathbf{w}} \mathcal{L}_{D_t}(\mathbf{w}))^T \mathbf{S}\mathbf{\Lambda}\mathbf{S}^T \nabla_{\mathbf{w}} \mathcal{L}_{B_t}(\mathbf{w}) \\ &\approx \eta_1 (\nabla_{\mathbf{w}} \mathcal{L}_{D_t}(\mathbf{w}))^T \nabla_{\mathbf{w}} \mathcal{L}_{B_t}(\mathbf{w}), \end{aligned} \quad (5.5)$$

where the last approximation in Eq. (5.5) is based on the property from [14]: the gradient update,  $\nabla_{\mathbf{w}} \mathcal{L}_{B_t}(\mathbf{w})$ , lies in the span of inputs,  $\mathbf{S}$ .

Therefore, following the opposite direction of FS-GPM's final gradient, Eq. (5.4), means that the last term, which is the dot product between the gradient of old tasks and that of the new one, is minimized. In such cases, these gradient vectors are forced to create an obtuse angle, or in other words, a gradient conflict. This is undesirable because it constrains the model to learn a new task at the expense of forgetting old ones. Although after that, the model is still updated following the orthogonal direction w.r.t  $\mathbf{S}$  of FS-GPM's final gradient as Eq. (5.2), i.e., minimal forgetting is ensured, this conflict enforcement may hinder knowledge transfer from



**Figure 5.1:** Two problems of FS-GPM. On the left, an obtuse angle is forced between the gradient of current and buffer data. On the right, the SAM gradient of current data nearly collapses to vector  $\mathbf{0}$ .

old tasks to new one. However, since FS-GPM approximates the subspace in practice using a buffer that is updated after each task, forgetting will gradually worsen as more tasks arrive. Hence, FS-GPM may suffer from potential gradient conflict which prevents stability-plasticity balancing

Second, the third term in Eq. (5.4) accounts for the direction leading towards flat regions of the current task  $t$ , i.e.,  $\eta_1(\nabla_{\mathbf{w}}\mathcal{L}_{D_t}(\mathbf{w}))^T \mathbf{S} \mathbf{A} \mathbf{S}^T \nabla_{\mathbf{w}}\mathcal{L}_{D_t}(\mathbf{w})$ . However, considering the following extreme case in which  $\nabla_{\mathbf{w}}\mathcal{L}_{D_t}(\mathbf{w}) \perp \mathbf{S}$ , its projection onto  $\mathbf{S}$  is the zero vector, resulting in the zero value for the third term. This means that, when the current task has no or little interference with old tasks, i.e., its gradient update is (almost) orthogonal to the subspace of the previous, although it can be learned freely without causing forgetting, SAM fails to find its flat minima, thereby limiting its generalization, which is the goal FS-GPM aims to achieve.

These two problems of the potential gradient conflict between new-old tasks, and the failure case of SAM are demonstrated in Fig. 5.1.

## 5.2.2 Our simple solution

To tackle the first problem, we simply separate  $B_t$  from the worst-case loss of the current task because their dot product will not appear when the loss is first-order Taylor expanded. Specifically, we substitute the original loss of FS-GPM with:  $\mathcal{L}_{D_t}(\mathbf{w} + \epsilon_{\parallel}) + \mathcal{L}_{B_t}(\mathbf{w})$ . We further apply SAM to the latter to encourage flat minima for old tasks, which shares the same benefits with SGAM in Chapter 4.

Then, our objective has the form:

$$\mathcal{L}_{D_t}(\mathbf{w} + \rho \cdot \epsilon_{\text{new}}) + \alpha \mathcal{L}_{B_t}(\mathbf{w} + \rho \cdot \epsilon_{\text{old}}), \quad (5.6)$$

$$\epsilon_{\text{new}} = \frac{\nabla_{\mathbf{w}}\mathcal{L}_{D_t}(\mathbf{w})}{\|\nabla_{\mathbf{w}}\mathcal{L}_{D_t}(\mathbf{w})\|_2}, \quad \epsilon_{\text{old}} = \frac{\nabla_{\mathbf{w}}\mathcal{L}_{B_t}(\mathbf{w})}{\|\nabla_{\mathbf{w}}\mathcal{L}_{B_t}(\mathbf{w})\|_2} \quad (5.7)$$

where  $\alpha$  is the hyper-parameter adjusting the impact of data in the buffer.  $\epsilon_{\text{new}}, \epsilon_{\text{old}}$

are the worst-case perturbations found in a  $\rho$ -radius ball, corresponding to new and old tasks, respectively.

Moreover, we posit that by encouraging congruence between new and old tasks, the model could learn invariant features between tasks, and thus can better transfer knowledge. This is obtained by our novel objective:

$$\mathcal{L}_{\text{final}} = \mathcal{L}_{D_t} \left( \mathbf{w} + \rho \cdot \epsilon_{\text{new}} - \frac{\rho_2 \cdot \epsilon_{\text{old}}}{\|\nabla_{\mathbf{w}} \mathcal{L}_{D_t}(\mathbf{w})\|_2} \right) \quad (5.8)$$

$$+ \alpha \mathcal{L}_{B_t} \left( \mathbf{w} + \rho \cdot \epsilon_{\text{old}} - \frac{\rho_2 \cdot \epsilon_{\text{new}}}{\|\nabla_{\mathbf{w}} \mathcal{L}_{B_t}(\mathbf{w})\|_2} \right), \quad (5.9)$$

where  $\rho_2$  is a hyper-parameter to control the degree to which gradient agreement is enforced. We will explain how this loss can help reduce conflict between gradients using first-order Taylor expansion as follows:

$$\begin{aligned} & \nabla_{\mathbf{w}} \mathcal{L}_{D_t} \left( \mathbf{w} + \rho \cdot \epsilon_{\text{new}} - \frac{\rho_2 \cdot \epsilon_{\text{old}}}{\|\nabla_{\mathbf{w}} \mathcal{L}_{D_t}(\mathbf{w})\|_2} \right) \\ &= \frac{d \left( \mathbf{w} + \rho \cdot \epsilon_{\text{new}} - \frac{\rho_2 \cdot \epsilon_{\text{old}}}{\|\nabla_{\mathbf{w}} \mathcal{L}_{D_t}(\mathbf{w})\|_2} \right)}{d\mathbf{w}} \nabla_{\mathbf{w}} \mathcal{L}_{D_t}(\mathbf{w})|_{\mathbf{w} + \rho \cdot \epsilon_{\text{new}} - \rho_2 \cdot \epsilon_{\text{old}} / \|\nabla_{\mathbf{w}} \mathcal{L}_{D_t}(\mathbf{w})\|_2} \\ &\stackrel{\Delta}{\approx} \nabla_{\mathbf{w}} \mathcal{L}_{D_t}(\mathbf{w})|_{\mathbf{w} + \rho \cdot \epsilon_{\text{new}} - \rho_2 \cdot \epsilon_{\text{old}} / \|\nabla_{\mathbf{w}} \mathcal{L}_{D_t}(\mathbf{w})\|_2} \end{aligned} \quad (5.10)$$

$$\begin{aligned} &\approx \nabla_{\mathbf{w}} \left[ \mathcal{L}_{D_t}(\mathbf{w}) + \rho \epsilon_{\text{new}}^T \nabla_{\mathbf{w}} \mathcal{L}_{D_t}(\mathbf{w}) - \frac{\rho_2 \epsilon_{\text{old}}^T \nabla_{\mathbf{w}} \mathcal{L}_{D_t}(\mathbf{w})}{\|\nabla_{\mathbf{w}} \mathcal{L}_{D_t}(\mathbf{w})\|_2} \right] \\ &= \nabla_{\mathbf{w}} [\mathcal{L}_{D_t}(\mathbf{w}) + \rho \|\nabla_{\mathbf{w}} \mathcal{L}_{D_t}(\mathbf{w})\|_2 - \rho_2 \cos(\nabla_{\mathbf{w}} \mathcal{L}_{D_t}(\mathbf{w}), \nabla_{\mathbf{w}} \mathcal{L}_{B_t}(\mathbf{w}))], \end{aligned} \quad (5.11)$$

where in  $\stackrel{\Delta}{\approx}$ , we have dropped the derivatives of  $\epsilon_{\text{old}}$  and  $\epsilon_{\text{new}}$  w.r.t  $\mathbf{w}$  to omit the computation of Hessian, and  $\cos(\cdot, \cdot)$  is the cosine similarity between two vectors. From Eq. (5.11), it can be seen that when the loss corresponding to new data,  $\mathcal{L}_{D_t}(\cdot)$ , is minimized, the model is not only guided to low-loss and flat regions, or SAM direction, of the new task  $t$ , cf. the first and the second term, but it is also led to the low-loss regions of previous tasks, cf. the third term. This is because the cosine similarity between the gradient of task  $t$  and that of data from the buffer is maximized, hence encouraging the model to find a good solution for both tasks. Similarly, the second loss term in Eq. (5.9) helps the model find flat minima corresponding to buffer data while not interfering with the learning of task  $t$ .

In terms of the second problem, our formulation for the objective 5.9 is already a solution because we do not constrain the worst-case perturbation to lie in  $\mathbf{S}$ , instead we only restrict it to be within a ball with radius  $\rho$ . Thus SAM can reach its full potential in finding flat minima.

Overall, our novel objective in Eq. (5.9) has the merits of (i) fully removing the adverse effect of SAM in terms of provoking conflicts between gradients of old and new tasks, (ii) allowing SAM to find flat minima of old tasks, (iii) further explicitly aligning such gradients to aim for common useful knowledge of all tasks, and (iv) allowing SAM to reach its full potential in finding flat minima. The pseudo-code for our method, named Gradient alignment for FS-GPM, is given in Algorithm 4.

---

**Algorithm 4** Gradient alignment for FS-GPM

---

**Input:** Model parameters  $w$ , learning rate  $\eta$ , SAM radius  $\rho$ , Gradient alignment radius  $\rho_2$ , current task data  $D_t$ , buffer  $B$ .

```

1: Initialize  $S \leftarrow I, \Lambda \leftarrow I$ 
2: for  $t \in \{1, \dots, T\}$  do
3:   for  $e \in \{1, \dots, E\}$  do
4:     for  $D_t \sim D_t$  do
5:        $B_t \sim B$ 
6:       Compute worst-case perturbations:
            $\epsilon_{\text{new}} = \nabla_w \mathcal{L}_{D_t}(w) / \|\nabla_w \mathcal{L}_{D_t}(w)\|_2,$ 
            $\epsilon_{\text{old}} = \nabla_w \mathcal{L}_{B_t}(w) / \|\nabla_w \mathcal{L}_{B_t}(w)\|_2$ 
7:       Compute  $\mathcal{L}_{\text{final}}$  using Eq. (5.9)
8:       if  $t \geq 2$  then
            $\Lambda \leftarrow \Lambda - \eta \nabla_{\Lambda} \mathcal{L}_{\text{final}}$ 
9:       end if
10:      Update parameters
            $w \leftarrow w - \eta_3 (I - S \Lambda S^T) \nabla_w \mathcal{L}_{\text{final}}$ 
11:     end for
12:   end for
13:   Update  $B$  with  $D_t$  using reservoir sampling
14:   Recompute bases of  $S$  with  $B$ 
15: end for
```

---

### 5.2.3 Discussion

In this chapter, we have analyzed two problems that FS-GPM may face, which are the hidden conflict between gradients of old and new tasks, and the failure of SAM in promoting flat minima. We simply circumvent these by first realizing that separating the old task’s perturbation from the loss of the current task, and vice versa, can remove such conflict. Furthermore, we not only find the perturbation corresponding to new tasks, but for old tasks as well. This could prevent the model from overfitting to the limited buffer. Finally, we explicitly seek to align the gradient directions of data from the new task and buffer by proposing a novel objective function. This could encourage the model to learn useful knowledge that can be transferred across tasks, thereby improving both its plasticity and stability.

In terms of space complexity, compared with FS-GPM, we need to store one more perturbation vector for the buffer data, whose size is equal to that of the network. In terms of time computation, during each mini-batch update, we have to sequentially forward data from the current task and then data from the buffer to separately compute those perturbation vectors. Additionally, our novel objective has two more hyper-parameters, weight terms for gradient alignment and for the old loss, compared to FS-GPM. It is also worth noting that under the data-free setting, where no buffer is stored, our method can not apply SAM on old data and thus the explicit gradient alignment is impossible since there is no perturbation for old data. However, in this case, our method still overcomes the second problem of FS-GPM, SAM’s infeasibility, as we do not project the perturbation onto the subspace. And we leave a more sophisticated constraint for  $\epsilon$  for future work.

## 5.3 Experiments

**Datasets.** We assess the performance of our algorithm across four distinct image classification datasets: Permuted MNIST (PMNIST), CIFAR-100 Split, CIFAR-100 Superclass, and TinyImageNet. PMNIST benchmark dataset comprises 20 tasks, each containing only 1000 samples drawn from 10 different classes. The CIFAR-100 Split dataset is created by randomly dividing the 100 classes of CIFAR-100 into 10 tasks, with each task consisting of 10 classes. On the other hand, the CIFAR-100 Superclass dataset is segmented into 20 tasks based on the 20 superclasses found in the CIFAR-100 dataset, with each superclass encompassing 5 distinct but semantically related classes. Lastly, TinyImageNet is constructed by dividing 200 classes into 40 5-way classification tasks. It is important to note that in our experiments, we did not employ any data augmentation techniques.

**Network Architecture.** In the case of PMNIST, we employ a fully connected network comprising two hidden layers, each containing 100 units, as described in [46]. When conducting experiments on CIFAR-100 Split and CIFAR-100 Superclass datasets, we adopt a 5-layer AlexNet architecture and a LeNet architecture, respectively, akin to the approach outlined in [14]. In the context of TinyImageNet, we utilize the identical network architecture as detailed in [53], which encompasses 4 convolutional layers and 3 fully connected layers. Notably, for PMNIST, all tasks share the final classifier layer, whereas, in other experiments, we employ a multi-head incremental setup, meaning that each task has its own dedicated classifier. Their detailed architectures are given in section B in the Appendix.

**Baselines** Our approach is evaluated against various methods as outlined below.

(1) EWC [5], which relies on regularization and utilizes the Fisher information diagonal to identify crucial weights; (2) iCaRL [10], a memory-based approach that leverages knowledge distillation and episodic memory to mitigate forgetting; (3) GEM [46], another memory-based technique that employs episodic memory gradients to constrain optimization and prevent forgetting; (4) ER [9], a straightforward yet competitive approach founded on reservoir sampling; (5) Multitask, serving as an oracle baseline where all tasks are learned jointly using the entire dataset in a single network. These other two close baselines are GPM [14] and FS-GPM [13]. It’s worth noting that multitasking is not a continual learning strategy but represents an upper bound for average test accuracy across all tasks. Since we use the same code base as FS-GPM [13], we adopt their reported results for these baselines. We re-implement FS-GPM with one inner step update to match our analysis above.

**Training Details:** All baseline methods, along with our own approach, employ stochastic gradient descent (SGD) during training. In the case of PMNIST and TinyImageNet, we train the network for 1 and 10 epochs per task, respectively, with a batch size of 10. In the CIFAR-100 Split and CIFAR-100 Superclass experiments, we employ an early termination strategy, training each task for up to 50 epochs based on validation loss, as suggested in [4]. For both datasets, the batch size is set to 64. The size of the replay buffers varies PMNIST, CIFAR-100 Split, CIFAR-100 Superclass, and TinyImageNet with 200, 1000, 1000, and 400, respectively. We push data to the memory buffer using reservoir sampling at the end of the first task to ensure this memory update happens once per task.

**Hyper-parameters.** We have  $(\rho, \rho_2, \alpha)$  as our hyper-parameters. For CIFAR-100 Split, CIFAR-100 Superclass, TinyImageNet, and PMNIST, the set of hyper-parameters are  $(0.01, 0.005, 0.3)$ ,  $(0.05, 0.05, 0.2)$ ,  $(0.001, 0.01, 0.5)$ ,  $(0.15, 0.01, 0.4)$ , respectively. For FS-GPM, we use the same  $\rho$  for fair comparison. Details for the chosen hyper-parameters can be found in Table 2 in Appendix C.

**Evaluation Protocol.** We use two common metrics to evaluate CL performance: Average Accuracy, ACC, and Backward Transfer, BWT as defined in Eq. (2.2).



### 5.3.1 Experimental Analysis

**Table 5.1:** Experimental results on 10-task CIFAR-100, 20-Task CIFAR-100 Superclass and 40-Task TinyImageNet in 50 epochs. Each experiment is run with 3 seeds. Bold and underline denote the best and the second-best results, respectively.

Method	CIFAR-100 Split		CIFAR-100 Superclass		TinyImageNet	
	ACC(%)	BWT(%)	ACC(%)	BWT(%)	ACC(%)	BWT(%)
EWC	72.77 $\pm$ 0.45	-3.59 $\pm$ 0.55	50.26 $\pm$ 1.48	-7.87 $\pm$ 1.63	-	-
GEM	70.15 $\pm$ 0.34	-8.61 $\pm$ 0.42	50.35 $\pm$ 0.80	-9.50 $\pm$ 0.85	50.57 $\pm$ 0.61	-20.50 $\pm$ 0.10
ICARL	53.50 $\pm$ 0.81	-20.44 $\pm$ 0.82	49.05 $\pm$ 0.51	-11.24 $\pm$ 0.27	54.77 $\pm$ 0.32	-3.93 $\pm$ 0.55
ER	70.07 $\pm$ 0.35	-7.70 $\pm$ 0.59	51.64 $\pm$ 1.09	-7.86 $\pm$ 0.89	48.32 $\pm$ 1.51	-19.86 $\pm$ 0.70
GPM	73.18 $\pm$ 0.52	<b>-1.17 <math>\pm</math> 0.27</b>	57.33 $\pm$ 0.37	<b>-0.37 <math>\pm</math> 0.12</b>	67.39 $\pm$ 0.47	<b>1.45 <math>\pm</math> 0.22</b>
FS-GPM	<u>73.94 <math>\pm</math> 0.47</u>	-2.26 $\pm$ 0.35	<u>57.85 <math>\pm</math> 0.40</u>	-4.17 $\pm$ 0.55	<u>70.41 <math>\pm</math> 1.30</u>	-2.11 $\pm$ 0.84
<b>Ours</b>	<b>74.71 <math>\pm</math> 0.35</b>	<u>-1.26 <math>\pm</math> 0.35</u>	<b>59.16 <math>\pm</math> 0.34</b>	<u>-2.98 <math>\pm</math> 0.35</u>	<b>71.30 <math>\pm</math> 0.41</b>	<u>-1.28 <math>\pm</math> 0.30</u>
Multitask	79.75 $\pm$ 0.38	-	61.00 $\pm$ 0.20	-	77.10 $\pm$ 1.06	-

**Task Incremental Learning.** Table 5.1 shows the results our our method and baselines on three multi-head datasets. It can be seen that we obtained the best average accuracy on all settings, 74.71%, 59.16%, and 71.30% on CIFAR-100 Split, CIFAR-100 Superclass, and TinyImageNet, respectively. In terms of backward transfer, we only fall behind GPM, the method with the least forgetting as it compromises the learning of new tasks. This trade-off is further emphasized by the longest datasets with 40 tasks: the accuracy gap between GPM and FS-GPM and our method is bigger than those in the other two datasets, while an improvement on old tasks is observed. To make GPM more plastic towards new tasks, FS-GPM relaxes some old bases that are less important, thus achieving higher average accuracy than GPM with some trade-off in backward transfer. However, due to the potential conflict of gradients in their projected worst-case perturbation, their performance may be limited. By tackling this problem, we simultaneously improve accuracy and mitigate forgetting of FS-GPM, or in other words, better balance plasticity-stability trade-off. Other baselines although utilizing a buffer to re-learn knowledge, are inferior to gradient-based methods.

**Domain Incremental Learning** We then evaluate our algorithm for 20 sequential PMNIST tasks in a single-head setting in Table 5.2. Overall, gradient-based methods still outperform the others with ours achieving the highest in both average accuracy and backward transfer, 77.40% and -6.22%. It should be noted that in a single-head setting, the shared classifier is updated sequentially without having its gradients constrained by GPM. Therefore, if the feature extractor can learn more useful feature representation for all tasks, the classifier can benefit more. By aligning gradients of new and old data, we encourage the model to find such features, thus enhancing domain incremental learning performance.

**Table 5.2:** Experimental results on PMNIST in single-epoch with 3 runs.

Method	PMNIST	
	ACC(%)	BWT(%)
EWC	$62.25 \pm 1.44$	$-15.22 \pm 1.25$
GEM	$61.82 \pm 0.85$	$-15.58 \pm 1.17$
ER	$68.31 \pm 0.56$	$-13.91 \pm 0.67$
GPM	$74.54 \pm 0.36$	$-7.17 \pm 0.51$
FS-DGPM	$76.96 \pm 0.51$	$-7.45 \pm 0.30$
<b>Ours</b>	<b><math>77.40 \pm 0.62</math></b>	<b><math>-6.22 \pm 0.80</math></b>
Multitask	$86.54 \pm 1.74$	-

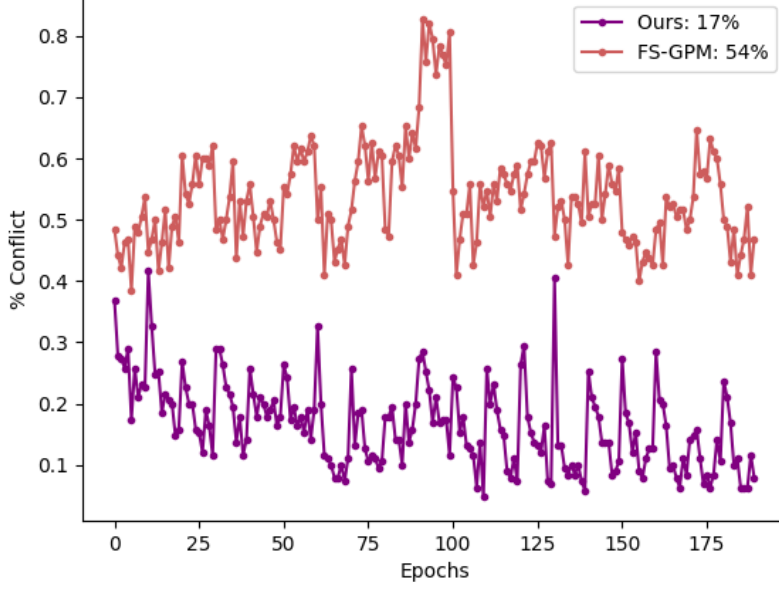
### 5.3.2 Ablation Study

We now investigate the impact of each component in our method, including: (i) applying SAM on buffer, and (ii) aligning gradients. We note that when ablating SAM on the buffer, we still separately computed gradients w.r.t current and past data to align them. The results are shown in Table 5.3. First, when ablating both components, we are left with SAM on current data without projection on subspace. Its improvement over FS-GPM shows that letting SAM find flat minima in the entire space can be beneficial. Second, when combined with gradient alignment, we can boost the performance slightly more than with SAM on the buffer. This can be because, at the final model update, the gradient step is still forced to be orthogonal to the subspace to ensure no forgetting, thus finding flat minima on the buffer might not significantly help old performance. Meanwhile, aligning gradients helps find useful knowledge for both tasks, thus facilitating both new task learning and old task preservation. Finally, when combining them all, our method inherits its merits and outperforms FS-GPM under both metrics.

**Table 5.3:** Ablation study on CIFAR-100 Split and Superclass. (-) means ablated, (+) means used.

SAM on buffer	Gradient alignment	CIFAR-100 Split		CIFAR-100 Superclass	
		ACC(%)	BWT(%)	ACC(%)	BWT(%)
-	-	$74.38 \pm 0.32$	$-1.50 \pm 0.27$	$57.92 \pm 0.45$	$-4.03 \pm 0.41$
-	+	$74.55 \pm 0.34$	$-1.36 \pm 0.30$	$58.72 \pm 0.30$	$-3.90 \pm 0.17$
+	-	$74.46 \pm 0.20$	$-1.48 \pm 0.22$	$58.36 \pm 0.29$	$-3.81 \pm 0.23$
+	+	<b><math>74.71 \pm 0.35</math></b>	<b><math>-1.26 \pm 0.35</math></b>	<b><math>59.16 \pm 0.34</math></b>	<b><math>-2.98 \pm 0.35</math></b>
FS-GPM		$73.94 \pm 0.47$	$-2.26 \pm 0.35$	$57.85 \pm 0.40$	$-4.17 \pm 0.55$

**Gradient Conflict Reduction.** Figure 5.2 further provides evidence for the benefit of our method in resolving potential gradient conflict that FS-GPM may face

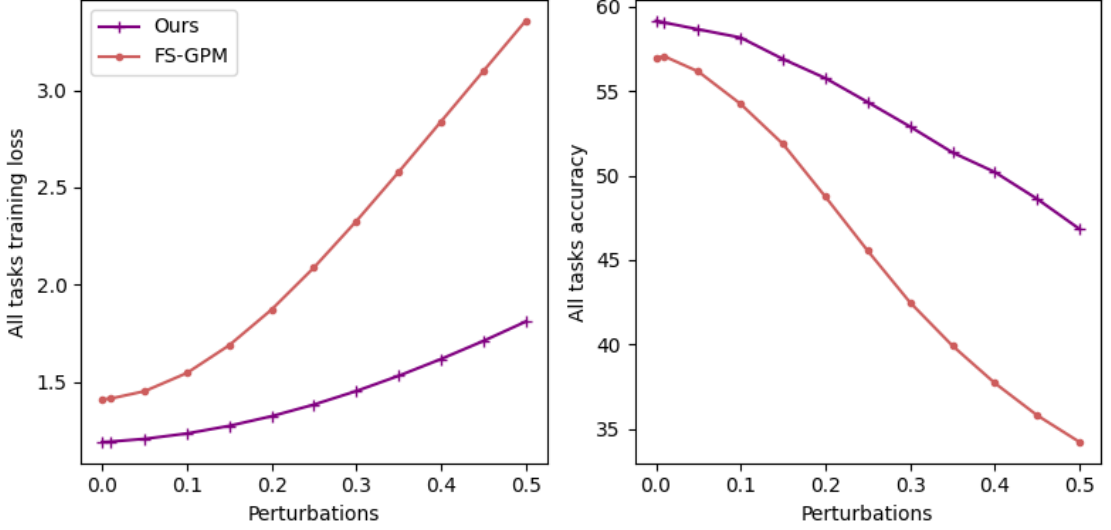


**Figure 5.2:** The ratio of conflict in one epoch along 20-task-training of CIFAR-100 Superclass, and the average ratio of FS-GPM and Our method.

due to the projection of SAM perturbation on old subspace. In detail, we measure the percentage of gradient conflict, i.e., the number of mini-batch updates that have negative cosine similarity between old and current gradients out of the total number of updates in one epoch. The figure shows the % Conflict along the training trajectory of the CIFAR-100 Superclass. Our method significantly and consistently reduces conflict in comparison with FS-GPM, and its benefit on CL has been shown in previous Tables.

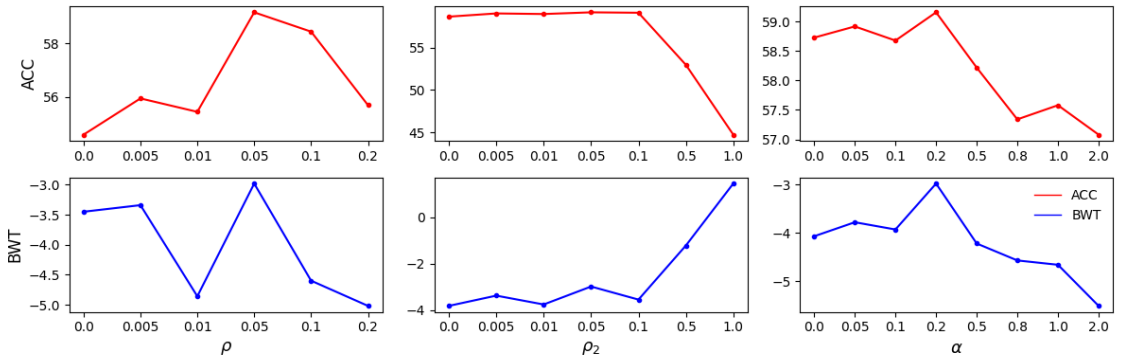
**Flatter Minima.** We also verify whether our method can find flatter minima compared to FS-GPM because we do not project the perturbation of the current task on the subspace spanned by previous features, which could hinder SAM from finding such minima when this task has little interference with old tasks, i.e., being orthogonal. Figure 5.3 presents the robustness of a model at the end of the training sequence. Specifically, we measure the changes in average training loss and accuracy over all tasks w.r.t the changes of weight perturbations. This number is the variance of the zero-mean Gaussian noise added to the model’s parameters to perturb it. The less significant the changes are, the more robust the model is. Thus, from the figure, we see that the minima found by our method are flatter than those of FS-GPM.

**Hyper-parameter Analysis** There are three hyper-parameter in our algorithm 4: the radius  $\rho$  of SAM perturbation, the weight term  $\rho_2$  for gradient alignment and the weight term  $\alpha$  for the buffer loss. Figure 5.4 shows how different values



**Figure 5.3:** Loss (left) and Accuracy (right) changes w.r.t weight perturbation of FS-GPM and Our method.

of them affect average accuracy and backward transfer on the CIFAR-100 Superclass dataset. It can be seen that  $\rho = 0.05$  gives the best results. For  $\rho_2$ , the model is rather stable with values less than 0.1 and degrades significantly for too large values 1.0, which is reasonable as a large value of  $\rho_2$  forces the model to align gradients instead of minimizing classification loss. However, it is worth noting that only  $\rho = 1.0$  gives positive backward transfer, at the expense of a low average accuracy. This suggests that gradient alignment does help transfer knowledge from later tasks to previous ones. This also means that there is room for improvement in backward transfer as our current results only improve that of baselines, but have not yet achieved positive backward transfer. Finally,  $\alpha$  ranging from 0.0 to 0.2 results in satisfying average accuracy and backward transfer, with the highest values at  $\alpha = 0.2$ .



**Figure 5.4:** Impact of SAM’s perturbation radius  $\rho$  (left), Gradient Alignment  $\rho_2$  (middle) and Old loss term  $\alpha$  (right) on ACC (up) and BWT (down) using CIFAR-100 Superclass.

## 5.4 Summary

This chapter covers how gradient projection memory methods can be improved by sharpness-aware minimization. In general, having flat minima assists a model in learning new tasks, i.e., plasticity, and preserving old tasks, i.e., stability. Because GPMs tend to favor the latter, being able to incorporate a flat-seeking optimizer into their optimization process can benefit CL performance. There has been one attempt at this before, but through in-depth investigation, we argue that their method may suffer from tasks conflicting with each other and SAM failing to find flat minima, which consequently can hinder their stability-plasticity balancing. To overcome these issues, we propose a simple solution to effectively remove such conflicts and failure cases of SAM. We empirically verify our findings and our method’s effectiveness with extensive experiments.

## Chapter 6. Conclusion

This thesis’s motivation is to improve the class of memory-based methods to solve Continual Learning, which often achieves state-of-the-art performance, especially under many challenging and realistic settings. We specifically aim to improve the generic form of this class as it can then be expanded to a wide range of complex methods. To do this, we target two subgroups of this approach: representation and knowledge distillation-based, and gradient-projection-based. Based on the observation that the old buffer these methods save is typically limited in size which often leads to a model being overfitted to this buffer, and thus can not capture generalized features representative of old tasks. As a result, their effectiveness in reducing catastrophic forgetting may be restrained. Therefore, the main focus of this thesis is: *Is there an effective way to deal with the overfitting of a model on the small memory buffer?* Inspired by the power of SAM, one recently introduced optimization method that combats overfitting by minimizing the sharpness of a model, we seek to investigate its benefits on memory-based continual learning. With the target of representative subgroups of this approach, we have come up with two solutions.

In terms of methods of the representation and knowledge distillation-based subgroup, despite previous attempts to bring flat minima to CL, we find their improvements are rather marginal under difficult scenarios and datasets. On the other hand, with the help of SAM, we obtain notable gains over these baselines. We further posit potential gradient conflict between SAM gradients of different loss terms representing old and new data. We then propose a gradient-aware version of SAM, SGAM, which significantly improves CL performance.

Regarding the gradient-projection-based subgroup, we notice one existing work aims at better balancing the stability and plasticity of a model by minimizing the sharpness of that model during learning of the current task. Despite their reported improvement, we mathematically show their drawbacks in terms of potentially putting the gradient of old and new tasks at odds, and restricting SAM from finding flat minima, all of which can hinder their initial goal of dealing with the stability-plasticity dilemma. To overcome these, we propose a simple solution to completely remove such hidden conflict between the gradients, to further explicitly make them more congruent, and to allow SAM to reach its full potential in searching for flat minima of both old and new data.

Extensive experiments verify our hypotheses and analyses as well as showcase

the impressive results of our methods. Despite this improvement, we have discussed their limits and set out some possible research directions in the future. First, we notice there is still a gap in the amount of gradient conflict between non-SAM methods and SGAM. One can also further improve SGAM in terms of CL performance and time computations by considering more advanced gradient aggregation strategies. Second, our solution in the second contribution has not yet obtained positive backward transfer, even though an explicit term is used to encourage it. We reckon this might be due to the incompatible nature of some tasks in which case forcing transfer may have a negative impact. Thus an adaptive scheme to decide when to transfer knowledge may help this. Finally, in the future, we also aim to test our methods under different settings of CL such as online CL, or to bring them to other CL approaches.

## REFERENCE

- [1] P. Foret, A. Kleiner, H. Mobahi and B. Neyshabur, “Sharpness-aware minimization for efficiently improving generalization,” in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, OpenReview.net, 2021. [Online]. Available: <https://openreview.net/forum?id=6TmlmposlrM>.
- [2] R. M. French, “Catastrophic forgetting in connectionist networks,” *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.
- [3] A. A. Rusu, N. C. Rabinowitz, G. Desjardins *et al.*, “Progressive neural networks,” *CoRR*, vol. abs/1606.04671, 2016. arXiv: 1606.04671. [Online]. Available: <http://arxiv.org/abs/1606.04671>.
- [4] J. Serra, D. Suris, M. Miron and A. Karatzoglou, “Overcoming catastrophic forgetting with hard attention to the task,” in *International conference on machine learning*, PMLR, 2018, pp. 4548–4557.
- [5] J. Kirkpatrick, R. Pascanu, N. Rabinowitz *et al.*, “Overcoming catastrophic forgetting in neural networks,” *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.
- [6] Z. Li and D. Hoiem, “Learning without forgetting,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.
- [7] F. Zenke, B. Poole and S. Ganguli, “Continual learning through synaptic intelligence,” in *International conference on machine learning*, PMLR, 2017, pp. 3987–3995.
- [8] A. Chaudhry, M. Ranzato, M. Rohrbach and M. Elhoseiny, “Efficient lifelong learning with A-GEM,” 2019. [Online]. Available: [https://openreview.net/forum?id=Hkf2\\\_sC5FX](https://openreview.net/forum?id=Hkf2\_sC5FX).
- [9] A. Chaudhry, M. Rohrbach, M. Elhoseiny *et al.*, “Continual learning with tiny episodic memories,” *CoRR*, vol. abs/1902.10486, 2019. arXiv: 1902.10486. [Online]. Available: <http://arxiv.org/abs/1902.10486>.
- [10] S.-A. Rebuffi, A. Kolesnikov, G. Sperl and C. H. Lampert, “Icarl: Incremental classifier and representation learning,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010.



- [11] A. Chaudhry, A. Gordo, P. Dokania, P. Torr and D. Lopez-Paz, “Using hindsight to anchor past knowledge in continual learning,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, 2021, pp. 6993–7001.
- [12] P. Buzzega, M. Boschini, A. Porrello, D. Abati and S. Calderara, “Dark experience for general continual learning: A strong, simple baseline,” *Advances in neural information processing systems*, vol. 33, pp. 15 920–15 930, 2020.
- [13] D. Deng, G. Chen, J. Hao, Q. Wang and P.-A. Heng, “Flattening sharpness for dynamic gradient projection memory benefits continual learning,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 710–18 721, 2021.
- [14] G. Saha, I. Garg and K. Roy, “Gradient projection memory for continual learning,” 2021. [Online]. Available: <https://openreview.net/forum?id=3AOj0RCNC2>.
- [15] S. Mirzadeh, M. Farajtabar, D. Görür, R. Pascanu and H. Ghasemzadeh, “Linear mode connectivity in multitask and continual learning,” 2021. [Online]. Available: [https://openreview.net/forum?id=Fmg\\\_fQYUejf](https://openreview.net/forum?id=Fmg\_fQYUejf).
- [16] A. Chaudhry, N. Khan, P. Dokania and P. Torr, “Continual learning in low-rank orthogonal subspaces,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 9900–9911, 2020.
- [17] L. Bonicelli, M. Boschini, A. Porrello, C. Spampinato and S. Calderara, “On the effectiveness of lipschitz-driven rehearsal in continual learning,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 31 886–31 901, 2022.
- [18] L. Kumari, S. Wang, T. Zhou and J. A. Bilmes, “Retrospective adversarial replay for continual learning,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 28 530–28 544, 2022.
- [19] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy and P. T. P. Tang, “On large-batch training for deep learning: Generalization gap and sharp minima,” 2017. [Online]. Available: <https://openreview.net/forum?id=H1oyRlYgg>.
- [20] H. Petzka, M. Kamp, L. Adilova, C. Sminchisescu and M. Boley, “Relative flatness and generalization,” *Advances in neural information processing systems*, vol. 34, pp. 18 420–18 432, 2021.
- [21] K. Lyu, Z. Li and S. Arora, “Understanding the generalization benefit of normalization layers: Sharpness reduction,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 34 689–34 708, 2022.

- [22] Y. Zhao, H. Zhang and X. Hu, “Penalizing gradient norm for efficiently improving generalization in deep learning,” in *International Conference on Machine Learning*, PMLR, 2022, pp. 26 982–26 992.
- [23] S. Cha, H. Hsu, T. Hwang, F. P. Calmon and T. Moon, “CPR: classifier-projection regularization for continual learning,” 2021. [Online]. Available: <https://openreview.net/forum?id=F2v4aqEL6ze>.
- [24] G. Shi, J. Chen, W. Zhang, L.-M. Zhan and X.-M. Wu, “Overcoming catastrophic forgetting in incremental few-shot learning by finding flat minima,” *Advances in neural information processing systems*, vol. 34, pp. 6747–6761, 2021.
- [25] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov and J. Van De Weijer, “Class-incremental learning: Survey and performance evaluation on image classification,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5513–5533, 2022.
- [26] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman and C. Finn, “Gradient surgery for multi-task learning,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 5824–5836, 2020.
- [27] B. Liu, X. Liu, X. Jin, P. Stone and Q. Liu, “Conflict-averse gradient descent for multi-task learning,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 878–18 890, 2021.
- [28] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
- [29] Y. LeCun, Y. Bengio and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [30] Z. Mai, R. Li, J. Jeong, D. Quispe, H. Kim and S. Sanner, “Online continual learning in image classification: An empirical survey,” *Neurocomputing*, vol. 469, pp. 28–51, 2022.
- [31] G. M. van de Ven and A. S. Tolias, “Three scenarios for continual learning,” *CoRR*, vol. abs/1904.07734, 2019. arXiv: 1904 . 07734. [Online]. Available: <http://arxiv.org/abs/1904.07734>.
- [32] A. Krizhevsky, G. Hinton *et al.*, “Learning multiple layers of features from tiny images,” 2009.
- [33] Y. Le and X. Yang, “Tiny imagenet visual recognition challenge,” *CS 231N*, vol. 7, no. 7, p. 3, 2015.
- [34] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

- [35] G. E. Hinton, O. Vinyals and J. Dean, “Distilling the knowledge in a neural network,” *CoRR*, vol. abs/1503.02531, 2015. arXiv: 1503 . 02531. [Online]. Available: <http://arxiv.org/abs/1503.02531>.
- [36] J. Smith, Y.-C. Hsu, J. Balloch, Y. Shen, H. Jin and Z. Kira, “Always be dreaming: A new approach for data-free class-incremental learning,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9374–9384.
- [37] S. Hou, X. Pan, C. C. Loy, Z. Wang and D. Lin, “Learning a unified classifier incrementally via rebalancing,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 831–839.
- [38] J. S. Vitter, “Random sampling with a reservoir,” *ACM Trans. Math. Softw.*, vol. 11, no. 1, 37–57, 1985, ISSN: 0098-3500. DOI: 10 . 1145 / 3147 . 3165. [Online]. Available: <https://doi.org/10.1145/3147.3165>.
- [39] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach and T. Tuytelaars, “Memory aware synapses: Learning what (not) to forget,” in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 139–154.
- [40] P. Foret, A. Kleiner, H. Mobahi and B. Neyshabur, “Sharpness-aware minimization for efficiently improving generalization,” in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, OpenReview.net, 2021. [Online]. Available: <https://openreview.net/forum?id=6TmlmposlrM>.
- [41] J. Yoon, E. Yang, J. Lee and S. J. Hwang, “Lifelong learning with dynamically expandable networks,” in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, OpenReview.net, 2018. [Online]. Available: <https://openreview.net/forum?id=Sk7KsfW0->.
- [42] J. Xu and Z. Zhu, “Reinforced continual learning,” *Advances in Neural Information Processing Systems*, vol. 31, 2018.
- [43] C.-Y. Hung, C.-H. Tu, C.-E. Wu, C.-H. Chen, Y.-M. Chan and C.-S. Chen, “Compacting, picking and growing for unforgetting continual learning,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [44] C. Fernando, D. Banarse, C. Blundell *et al.*, “Pathnet: Evolution channels gradient descent in super neural networks,” *CoRR*, vol. abs/1701.08734, 2017. arXiv: 1701 . 08734. [Online]. Available: <http://arxiv.org/abs/1701.08734>.

- [45] G. Kim, C. Xiao, T. Konishi, Z. Ke and B. Liu, “A theoretical study on solving continual learning,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 5065–5079, 2022.
- [46] D. Lopez-Paz and M. Ranzato, “Gradient episodic memory for continual learning,” *Advances in neural information processing systems*, vol. 30, 2017.
- [47] H. Shin, J. K. Lee, J. Kim and J. Kim, “Continual learning with deep generative replay,” *Advances in neural information processing systems*, vol. 30, 2017.
- [48] M. van der Ven and A. S. Tolias, “Generative replay with feedback connections as a general strategy for continual learning,” *CoRR*, vol. abs/1809.10635, 2018. arXiv: 1809.10635. [Online]. Available: <http://arxiv.org/abs/1809.10635>.
- [49] C. Wu, L. Herranz, X. Liu, J. Van De Weijer, B. Raducanu *et al.*, “Memory replay gans: Learning to generate new categories without forgetting,” *Advances in Neural Information Processing Systems*, vol. 31, 2018.
- [50] Z. Borsos, M. Mutny and A. Krause, “Coresets via bilevel optimization for continual learning and streaming,” *Advances in neural information processing systems*, vol. 33, pp. 14 879–14 890, 2020.
- [51] R. Aljundi, K. Kelchtermans and T. Tuytelaars, “Task-free continual learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 254–11 263.
- [52] S. I. Mirzadeh, M. Farajtabar, R. Pascanu and H. Ghasemzadeh, “Understanding the role of training regimes in continual learning,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 7308–7320, 2020.
- [53] G. Gupta, K. Yadav and L. Paull, “Look-ahead meta learning for continual learning,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 11 588–11 598, 2020.
- [54] D. A. McAllester, “Pac-bayesian model averaging,” in *Proceedings of the twelfth annual conference on Computational learning theory*, 1999, pp. 164–170.
- [55] P. Alquier, J. Ridgway and N. Chopin, “On the properties of variational approximations of gibbs posteriors,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 8374–8414, 2016.
- [56] B. Laurent and P. Massart, “Adaptive estimation of a quadratic functional by model selection,” *Annals of statistics*, pp. 1302–1338, 2000.

# APPENDIX

## A Theorem's Proof

Before we begin, it is worth noting that the proof in [1] is based on PAC-Bayes bound [54] which defines the bound for 0-1 loss. However, to apply to more general losses used in CL, such as  $l_2$  regularization or cosine similarity, we have exploited a more general PAC-Bayes bound [55] which only requires the loss to be bounded, and this is typically ensured in training a deep model. Also note that when the loss is  $l_2$  regularization between penultimate features of old and current models, the parameters being considered are  $\theta$ . However, for ease of notation, we abuse  $\mathcal{L}$  and  $w$  for the loss and the model's parameters, respectively.

We first start with the following theorem, which is inspired by the general PAC-Bayes in [55].

**Theorem 2.** *Suppose  $\mathcal{D}$  is the underlying data distribution and  $\mathcal{S}$  is the training dataset drawn i.i.d from that distribution. With the assumption that adding Gaussian perturbation will raise the test error:  $\mathcal{L}_{\mathcal{D}}(w) \leq \mathbb{E}_{\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_T)} [\mathcal{L}_{\mathcal{D}}(w + \epsilon)]$ . Let  $T$  be the number of parameter  $w$ , and  $N$  be the cardinality of  $\mathcal{S}$ , then the following inequality is true with the probability  $1 - \delta$ :*

$$\mathcal{L}_{\mathcal{D}}(w) \leq \mathbb{E}_{\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbb{I})} [\mathcal{L}_{\mathcal{S}}(w + \epsilon)] + \frac{1}{\sqrt{N}} \left[ \frac{1}{2} + \frac{T}{2} \log \left( 1 + \frac{\|w\|^2}{T\sigma^2} \right) + \log \frac{1}{\delta} + 6 \log(N + T) + \frac{L^2}{8} \right]$$

where  $L$  is the upper bound of the loss function.

*Proof.* We use the PAC-Bayes theory for  $P = \mathcal{N}(0, \sigma_P^2 \mathbb{I}_T)$  and  $Q = \mathcal{N}(w, \sigma^2 \mathbb{I}_T)$  are the prior and posterior distributions, respectively.

By using the bound in [55], with probability at least  $1 - \delta$  and for all  $\beta > 0$ , we have:

$$\mathbb{E}_{w \sim Q} [\mathcal{L}_{\mathcal{D}}(w)] \leq \mathbb{E}_{w \sim Q} [\mathcal{L}_{\mathcal{S}}(w)] + \frac{1}{\beta} \left[ \text{KL}(Q \| P) + \log \frac{1}{\delta} + \Psi(\beta, N) \right],$$

where we have defined:

$$\Psi(\beta, N) = \log \mathbb{E}_P \mathbb{E}_{\mathcal{S}} \left[ \exp \left\{ \beta (\mathcal{L}_{\mathcal{D}}(w) - \mathcal{L}_{\mathcal{S}}(w)) \right\} \right]$$

Note that the loss function is bounded by  $L$ , according to Hoeffding's lemma,

we have:

$$\Psi(\beta, N) \leq \frac{\beta^2 L^2}{8N}.$$

Consider the case where  $\|\mathbf{w}\|^2 \geq T\sigma^2 \left[ \exp \frac{4N}{T} - 1 \right]$

By Cauchy inequality:

$$\frac{1}{\sqrt{N}} \left[ \frac{T}{2} \log \left( 1 + \frac{\|\mathbf{w}\|^2}{T\sigma^2} \right) + \frac{L^2}{8} \right] \geq \frac{L}{2\sqrt{N}} \sqrt{T \log \left( 1 + \frac{\|\mathbf{w}\|^2}{T\sigma^2} \right)} \geq L,$$

which means that the theorem is proved since the loss function is upper bounded by  $L$ , following assumptions.

Now, we only need to prove the theorem under the case:  $\|\mathbf{w}\|^2 \leq T\sigma^2 \left[ \exp \frac{4N}{T} - 1 \right]$ .

We need to specify  $P$  in advance since it is a prior distribution. However, we do not know in advance the value of  $\mathbf{w}$  that affects the KL divergence term. Hence, we build a family of distribution  $P$  as follows:

$$\mathfrak{P} = \left\{ P_j = \mathcal{N}(\mathbf{0}, \sigma_{P_j}^2 \mathbb{I}_T) : \sigma_{P_j}^2 = c \exp \left( \frac{1-j}{T} \right), c = \sigma^2 \left( 1 + \exp \frac{4N}{T} \right), j = 1, 2, \dots \right\}.$$

Set  $\delta_j = \frac{6\delta}{\pi^2 j^2}$ , the below inequality holds with probability at least  $1 - \delta_j$ :

$$\mathbb{E}_{\mathbf{w} \sim Q} [\mathcal{L}_{\mathcal{D}}(\mathbf{w})] \leq \mathbb{E}_{\mathbf{w} \sim Q} [\mathcal{L}_{\mathcal{S}}(\mathbf{w})] + \frac{1}{\beta} \left[ \text{KL}(Q \| P_j) + \log \frac{1}{\delta_j} + \frac{\beta^2 L^2}{8N} \right].$$

Or it can be written as:

$$\mathbb{E}_{\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbb{I})} [\mathcal{L}_{\mathcal{D}}(\mathbf{w} + \epsilon)] \leq \mathbb{E}_{\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbb{I})} [\mathcal{L}_{\mathcal{S}}(\mathbf{w} + \epsilon)] + \frac{1}{\beta} \left[ \text{KL}(Q \| P_j) + \log \frac{1}{\delta_j} + \frac{\beta^2 L^2}{8N} \right].$$

Thus, with probability  $1 - \delta$  the above inequalities hold for all  $P_j$ . We choose:

$$j^* = \left\lfloor 1 + T \log \left( \frac{\sigma^2 (1 + \exp\{4N/T\})}{\sigma^2 + \|\mathbf{w}\|^2/T} \right) \right\rfloor.$$

Since  $\frac{\|\mathbf{w}\|^2}{T} \leq \sigma^2 \left[ \exp \frac{4N}{T} - 1 \right]$ , we get  $\sigma^2 + \frac{\|\mathbf{w}\|^2}{T} \leq \sigma^2 \exp \frac{4N}{T}$ , thus  $j^*$  is well-defined.

We also have:

$$\begin{aligned}
& T \log \frac{c}{\sigma^2 + \|\mathbf{w}\|^2/T} \leq j^* \leq 1 + T \log \frac{c}{\sigma^2 + \|\mathbf{w}\|^2/T} \\
\Rightarrow \quad & \log \frac{c}{\sigma^2 + \|\mathbf{w}\|^2/T} \leq \frac{j^*}{T} \leq \frac{1}{T} + \log \frac{c}{\sigma^2 + \|\mathbf{w}\|^2/T} \\
\Rightarrow \quad & -\frac{1}{T} + \log \frac{\sigma^2 + \|\mathbf{w}\|^2/T}{c} \leq \frac{-j^*}{T} \leq \log \frac{\sigma^2 + \|\mathbf{w}\|^2/T}{c} \\
\Rightarrow \quad & e^{-1/T} \frac{\sigma^2 + \|\mathbf{w}\|^2/T}{c} \leq e^{-j^*/T} \leq \frac{\sigma^2 + \|\mathbf{w}\|^2/T}{c} \\
\Rightarrow \quad & \sigma^2 + \frac{\|\mathbf{w}\|^2}{T} \leq ce^{\frac{1-j^*}{T}} \leq e^{\frac{1}{T}} \left( \sigma^2 + \frac{\|\mathbf{w}\|^2}{T} \right) \\
\Rightarrow \quad & \sigma^2 + \frac{\|\mathbf{w}\|^2}{T} \leq \sigma_{P_{j^*}}^2 \leq e^{\frac{1}{T}} \left( \sigma^2 + \frac{\|\mathbf{w}\|^2}{T} \right).
\end{aligned}$$

Hence, we have:

$$\begin{aligned}
\text{KL}(Q\|P_{j^*}) &= \frac{1}{2} \left[ \frac{T\sigma^2 + \|\mathbf{w}\|^2}{\sigma_{P_{j^*}}^2} - T + T \log \frac{\sigma_{P_{j^*}}^2}{\sigma^2} \right] \\
&\leq \frac{1}{2} \left[ \frac{T\sigma^2 + \|\mathbf{w}\|^2}{\sigma^2 + \|\mathbf{w}\|^2/T} - T + T \log \frac{e^{1/T} (\sigma^2 + \|\mathbf{w}\|^2/T)}{\sigma^2} \right] \\
&\leq \frac{1}{2} \left[ 1 + T \log \left( 1 + \frac{\|\mathbf{w}\|^2}{T\sigma^2} \right) \right].
\end{aligned}$$

For the term  $\log \frac{1}{\delta_{j^*}}$ , use the inequality  $\log(1 + e^t) \leq 1 + t$  for  $t > 0$ :

$$\begin{aligned}
\log \frac{1}{\delta_{j^*}} &= \log \frac{(j^*)^2 \pi^2}{6\delta} = \log \frac{1}{\delta} + \log \left( \frac{\pi^2}{6} \right) + 2 \log(j^*) \\
&\leq \log \frac{1}{\delta} + \log \frac{\pi^2}{6} + 2 \log \left( 1 + T \log \frac{\sigma^2 (1 + \exp(4N/T))}{\sigma^2 + \|\mathbf{w}\|^2/T} \right) \\
&\leq \log \frac{1}{\delta} + \log \frac{\pi^2}{6} + 2 \log \left( 1 + T \log (1 + \exp(4N/T)) \right) \\
&\leq \log \frac{1}{\delta} + \log \frac{\pi^2}{6} + 2 \log \left( 1 + T \left( 1 + \frac{4N}{T} \right) \right) \\
&\leq \log \frac{1}{\delta} + \log \frac{\pi^2}{6} + \log(1 + T + 4N).
\end{aligned}$$

Choosing  $\beta = \sqrt{N}$ , with probability at least  $1 - \delta$  we get:

$$\begin{aligned}
& \frac{1}{\beta} \left[ \text{KL}(Q\|P_{j^*}) + \log \frac{1}{\delta_{j^*}} + \frac{\beta^2 L^2}{8N} \right] \\
&\leq \frac{1}{\sqrt{N}} \left[ \frac{1}{2} + \frac{T}{2} \log \left( 1 + \frac{\|\mathbf{w}\|^2}{T\sigma^2} \right) + \log \frac{1}{\delta} + 6 \log(N + T) \right] + \frac{L^2}{8\sqrt{N}}.
\end{aligned}$$

Thus the theorem is proved. □



Now, we can follow [1] to finish the proof. To make the thesis self-contained, we will include the details here.

First, let us re-state theorem 1 formally:

**Theorem 3.** *For any perturbation radius  $\rho > 0$ , under some mild conditions, with probability  $1 - \delta$  over the choice of drawing  $B$ , we have:*

$$\mathcal{L}_{D_1}(\mathbf{w}) \leq \max_{\|\epsilon_2\|_p \leq \rho} \mathcal{L}_B(\mathbf{w}) + h(\|\mathbf{w}\|_2^2), \quad (1)$$

$$KD_{D_1}(\mathbf{w}) \leq \max_{\|\epsilon_3\|_p \leq \rho} KD_B(\mathbf{w}) + h(\|\mathbf{w}\|_2^2), \quad (2)$$

where

$$h(\|\mathbf{w}\|_2^2) = \frac{1}{\sqrt{N}} \left[ \frac{1}{2} + \frac{T}{2} \log \left( 1 + \frac{\|\mathbf{w}\|^2}{T\sigma^2} \right) + \log \frac{1}{\delta} + 6 \log(N + T) + \frac{L^2}{8} \right].$$

*Proof.* Without loss of generality, we prove for  $\mathcal{L}_{D_1}$ . Now we can see that theorem 3 can be proven if we replace  $\mathcal{D}$  and  $\mathcal{S}$  in theorem 2 with  $D_1$  and with  $B$ , respectively, which is reasonable, in the following theorem:

**Theorem 4.** *For any perturbation radius  $\rho > 0$ , under some mild conditions, with probability  $1 - \delta$  over the choice of drawing  $B$ , we have:*

$$\mathcal{L}_{\mathcal{D}}(\mathbf{w}) \leq \max_{\|\epsilon\|_p \leq \rho} \mathcal{L}_{\mathcal{S}}(\mathbf{w}) + h(\|\mathbf{w}\|_2^2), \quad (3)$$

where  $h(\|\mathbf{w}\|_2^2)$  is defined similarly as in theorem 3.

*Proof.* From Theorem 2, we have:

$$\mathcal{L}_{\mathcal{D}}(\mathbf{w}) \leq \mathbb{E}_{\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbb{I})} [\mathcal{L}_{\mathcal{S}}(\mathbf{w} + \epsilon)] + h(\|\mathbf{w}\|_2^2)$$

We have  $\epsilon \sim N(0, \sigma^2 \mathbb{I}_T)$  with the dimension  $T$ , therefore  $\|\epsilon\|_2^2$  follows the Chi-square distribution. As proven in [56], we have:

$$P \left( \|\epsilon\|_2^2 \geq T\sigma^2 + 2\sigma^2\sqrt{Tt} + 2t\sigma^2 \right) \leq e^{-t},$$

$$P \left( \|\epsilon\|_2^2 < T\sigma^2 + 2\sigma^2\sqrt{Tt} + 2t\sigma^2 \right) > 1 - e^{-t}, \forall t > 0.$$

Select  $t = \ln(\sqrt{N})$ , we derive the following bound for the noise magnitude in

terms of the perturbation radius  $\rho$ :

$$P \left( \|\epsilon\|_2^2 \leq \sigma^2(2 \ln(\sqrt{N}) + T + 2\sqrt{T \ln(\sqrt{N})}) \right) > 1 - \frac{1}{\sqrt{N}}. \quad (4)$$

By choosing  $\sigma$  less than  $\frac{\rho}{\sqrt{2 \ln N^{1/2} + T + 2\sqrt{T \ln N^{1/2}}}}$ , from, Eq. (4), we achieve:

$$P \left( \|\epsilon\|_2^2 < \rho \right) > 1 - \frac{1}{N^{1/2}}.$$

Now we can finish the proof as:

$$\begin{aligned} \mathcal{L}_{\mathcal{D}}(\mathbf{w}) &\leq \mathbb{E}_{\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbb{I})} [\mathcal{L}_{\mathcal{S}}(\mathbf{w} + \epsilon)] + h(\|\mathbf{w}\|_2^2) \\ &\leq \left(1 - \frac{1}{\sqrt{N}}\right) \int_{\epsilon \in B(\rho)} \mathcal{L}_{\mathcal{S}}(\mathbf{w} + \epsilon) d\epsilon + \frac{L}{\sqrt{N}} + h(\|\mathbf{w}\|_2^2) \\ &\leq \max_{\|\epsilon\|_p \leq \rho} \mathcal{L}_{\mathcal{S}}(\mathbf{w} + \epsilon) + \frac{L}{\sqrt{N}} + h(\|\mathbf{w}\|_2^2), \end{aligned}$$

where  $B(\rho) = \{\epsilon : \|\epsilon\|_p \leq \rho\}$ . By replacing  $h(\|\mathbf{w}\|_2^2) = h(\|\mathbf{w}\|_2^2) + \frac{L}{\sqrt{N}}$ , we finish the proof for Theorem 4.  $\square$

Thus our main Theorem 3 is proven.  $\square$

## B Network Architecture

In Chapter 5, we use the following architectures.

**MLP Architecture:** We use a simple network with 2 fully connected layers with 100 units in each layer, illustrated in Fig. 2.2.

**AlexNet Architecture:** For the **CIFAR** dataset, we employ the **AlexNet** architecture with 3 convolutional layers and 2 fully connected layers. The number of filters in these 3 convolutional layers is 64, 128, 256 respectively, with filter sizes of  $4 \times 4$ ,  $3 \times 3$ ,  $2 \times 2$ . Additionally, the network uses **Dropout** and **Batch Normalization**.

**LeNet-5 Architecture:** We use a modified LeNet-5 architecture with the first 2 layers being convolutional layers with 20 and 50 filters of size  $5 \times 5$ , followed by 2 fully connected layers with 800 and 500 neurons. Additionally, **Batch Normalization** and **Max-Pooling** layers with a  $3 \times 3$  size are used.

**Architecture used for Tiny-ImageNet:** We employ a deep network with 4 convolutional layers, each consisting of 160 filters of size  $3 \times 3$ . The output of the last

convolutional layer is flattened and connected to 2 fully connected layers with 320 and 640 units, respectively.

All the networks used utilize the ReLU activation function, and the final layer represents separate outputs for the classification of each task.

## C Hyper-parameters

We provide details for the hyper-parameters chosen for each of our experiments reported in the main thesis.

**Table 1:** Hyper-parameters for F2M, CPR and SGAM in Table 4.1.  $l$  is the number of layers to inject noise in F2M and  $\epsilon$  is the corresponding noise bound.  $\gamma$  is the weight for CPR’s entropy maximization term.

		F2M-DER++	CPR-DER++	SGAM-CL
S-CIFAR-10	200	$\alpha = 0.1, \beta = 1.0, l = 2, \epsilon = 0.001$	$\alpha = 0.5, \beta = 1.0, \gamma = 0.1$	$\alpha = 1.0, \beta = 2.0, \rho = 0.1$
	500	$\alpha = 0.1, \beta = 1.0, l = 2, \epsilon = 0.001$	$\alpha = 0.2, \beta = 0.5, \gamma = 0.1$	$\alpha = 1.0, \beta = 2.0, \rho = 0.05$
	5120	$\alpha = 0.1, \beta = 1.0, l = 2, \epsilon = 0.001$	$\alpha = 0.1, \beta = 1.0, \gamma = 0.1$	$\alpha = 1.0, \beta = 2.0, \rho = 0.1$
S-Tiny-ImageNet	200	$\alpha = 0.1, \beta = 1.0, l = 3, \epsilon = 0.001$	$\alpha = 0.1, \beta = 1.0, \gamma = 0.1$	$\alpha = 1.0, \beta = 2.0, \rho = 0.1$
	500	$\alpha = 0.2, \beta = 1.0, l = 3, \epsilon = 0.001$	$\alpha = 0.1, \beta = 0.5, \gamma = 0.02$	$\alpha = 2.0, \beta = 1.0, \rho = 0.05$
	5120	$\alpha = 0.1, \beta = 0.5, l = 3, \epsilon = 0.001$	$\alpha = 0.1, \beta = 0.5, \gamma = 0.1$	$\alpha = 1.0, \beta = 2.0, \rho = 0.1$
P-MNIST	200	$\alpha = 1.0, \beta = 1.0, l = 3, \epsilon = 0.001$	$\alpha = 1.0, \beta = 1.0, \gamma = 0.1$	$\alpha = 1.0, \beta = 1.0, \rho = 0.1$
	500	$\alpha = 1.0, \beta = 1.0, l = 3, \epsilon = 0.001$	$\alpha = 1.0, \beta = 1.0, \gamma = 0.1$	$\alpha = 1.0, \beta = 0.5, \rho = 0.1$
	5120	$\alpha = 0.5, \beta = 1.0, l = 3, \epsilon = 0.001$	$\alpha = 1.0, \beta = 1.0, \gamma = 0.1$	$\alpha = 0.5, \beta = 1.0, \rho = 0.01$
R-MNIST	200	$\alpha = 1.0, \beta = 1.0, l = 3, \epsilon = 0.001$	$\alpha = 1.0, \beta = 1.0, \gamma = 0.1$	$\alpha = 1.0, \beta = 1.0, \rho = 0.01$
	500	$\alpha = 1.0, \beta = 1.0, l = 3, \epsilon = 0.001$	$\alpha = 1.0, \beta = 1.0, \gamma = 0.1$	$\alpha = 0.5, \beta = 1.0, \rho = 0.01$
	5120	$\alpha = 0.5, \beta = 0.5, l = 3, \epsilon = 0.001$	$\alpha = 0.5, \beta = 0.5, \gamma = 0.1$	$\alpha = 0.1, \beta = 0.5, \rho = 0.01$

**Table 2:** Hyper-parameters for FS-GPM and Ours in Tables 5.1 and 5.2.

	Ours	FS-GPM
CIFAR-100 Split	$\rho = 0.01, \rho_2 = 0.005, \alpha = 0.5$	$\rho = 0.01$
CIFAR-100 Superclass	$\rho = 0.05, \rho_2 = 0.05, \alpha = 0.2$	$\rho = 0.05$
TinyImageNet	$\rho = 0.001, \rho_2 = 0.01, \alpha = 0.5$	$\rho = 0.001$
PMNIST	$\rho = 0.15, \rho_2 = 0.05, \alpha = 0.4$	$\rho = 0.15$