ICIP2022

CONTINUAL LEARNING IN VISION TRANSFORMER

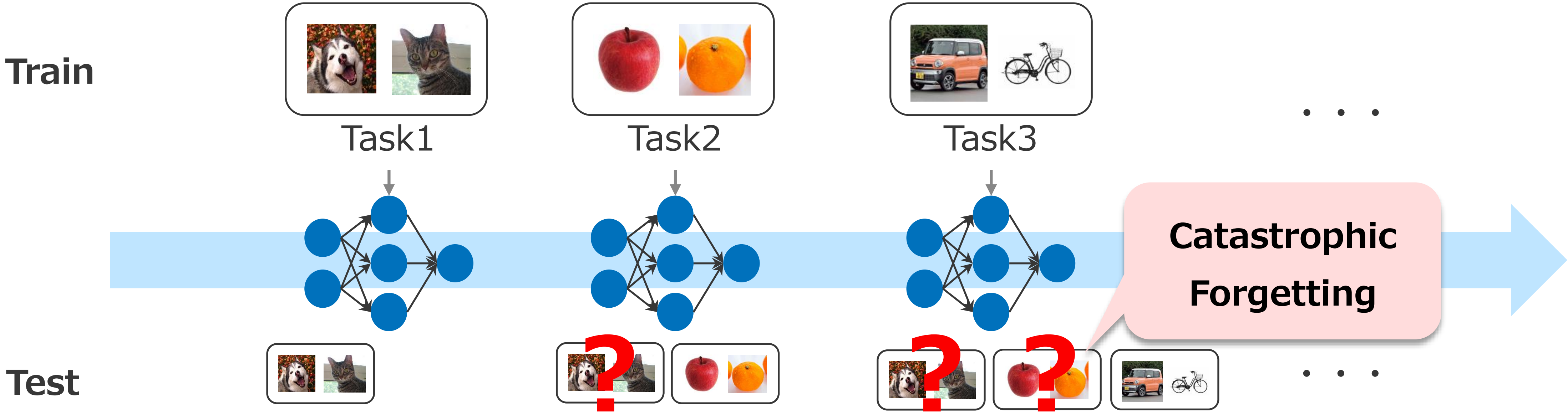
The University of Electro-Communications Department of Informatics
Tokyo, Japan

Mana Takedea, Keiji Yanai

1. INTRODUCTON

- Deep learning models forget previously learned tasks when given a new task (catastrophic forgetting)

- Continual Learning addresses this problem by allowing users to continuously learn new tasks while retaining knowledge of previously learned tasks



1. INTRODUCTON

- Recently, the Vision Transformer, which utilizes the Transformer architecture used in natural language processing for computer vision, has shown accuracy that exceeds that of CNN

- Conventional Continual Leaning methods are generally designed to be applied to CNNs, so **methods that can be applied to Vision Transformer are limited**

- Vision Transformer, which has a larger model size than CNN, requires a larger additional model size when applying Continual Learning methods

→ **Need to suppress catastrophic forgetting with fewer parameters** than conventional methods for application to CNN

1. INTRODUCTON

- Recently, the Vision Transformer, which utilizes the Transformer architecture used in natural language processing for computer vision, has shown accuracy that exceeds that of CNN

- Conventional Continual Leaning methods are generally designed to be applied to CNNs, so **methods that can be applied to Vision Transformer are limited**

- Vision Transformer, which has a larger model size than CNN, requires a larger additional model size when applying Continual Learning methods

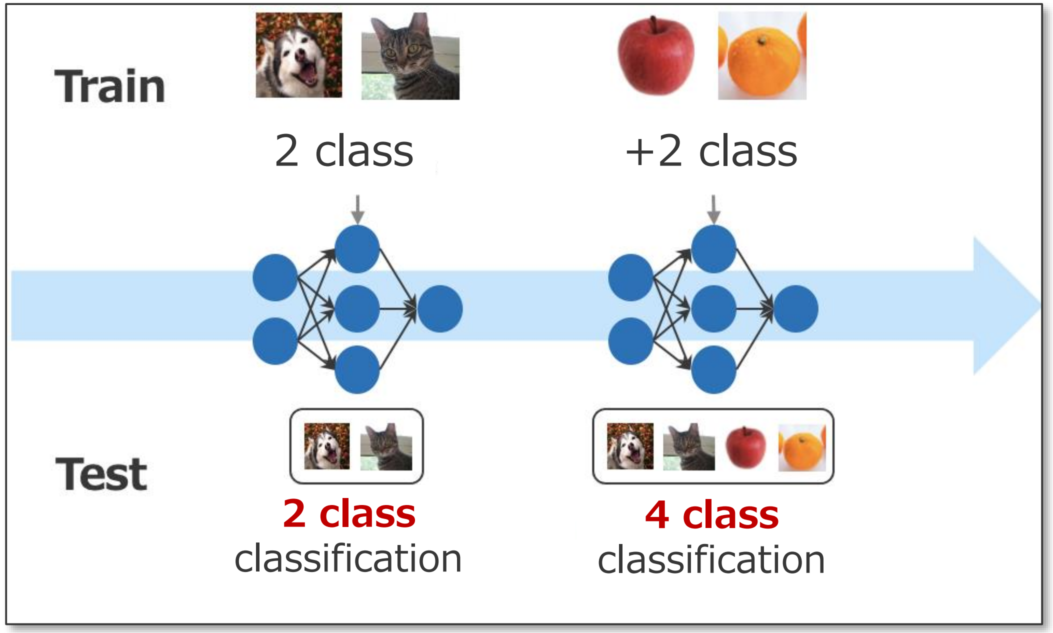
→ **Need to suppress catastrophic forgetting with fewer parameters** than conventional methods for application to CNN

Objective

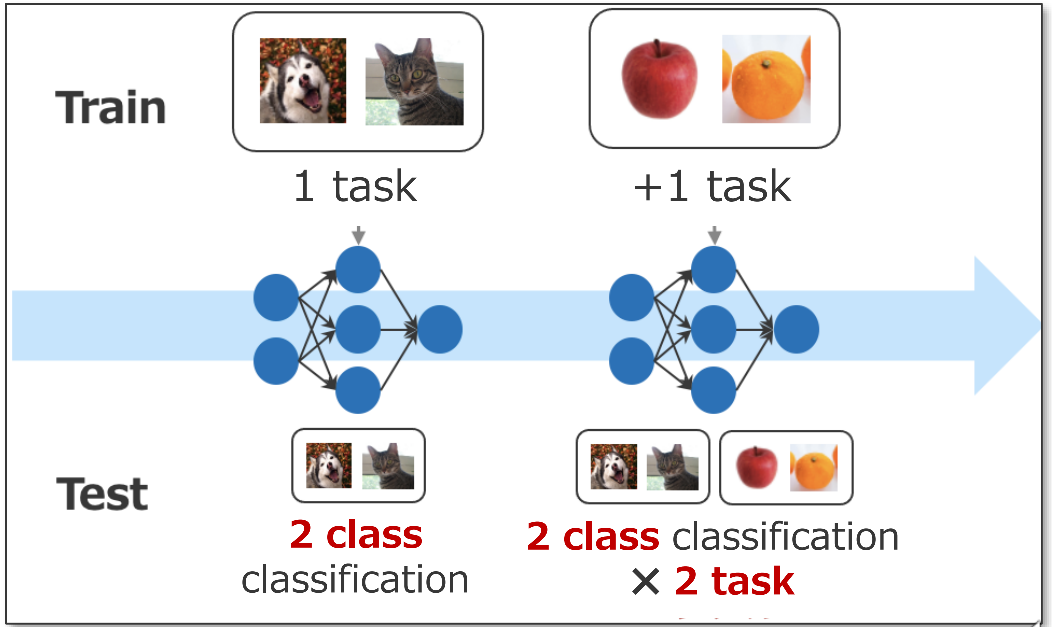
Method to suppress catastrophic forgetting with few parameters applicable to Vision Transformer

3. RELATED WORK - Continual Learning -

- Continual Learning is a method of continuously learning new tasks while retaining knowledge of tasks learned in the past
  - **Class incremental**: a new class is added
  - **Task incremental**: a new task is added



▲ Class incremental

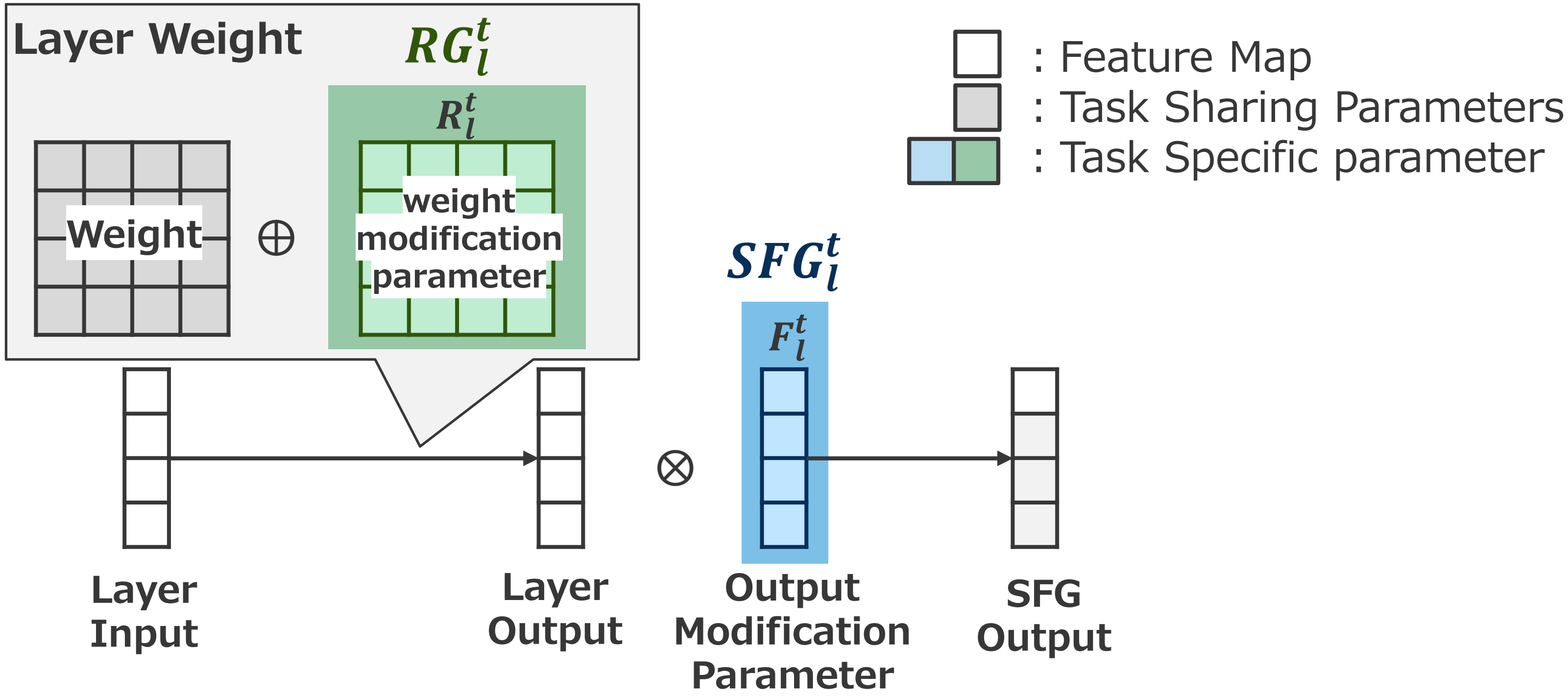


▲ **Task incremental**

- **Rectification-based Knowledge Retention (RKR)**

  [1] Singh et al. Rectification-based Knowledge Retention for Continual Learning. CVPR 2021

  – Apply task-specific modification parameters to the base parameters
    - **R**ectification **G**enerator (**RG**) : Parameters to modify weights
    - **S**caling **F**actor **G**enerator (**SFG**) : Parameters to modify intermediate outputs



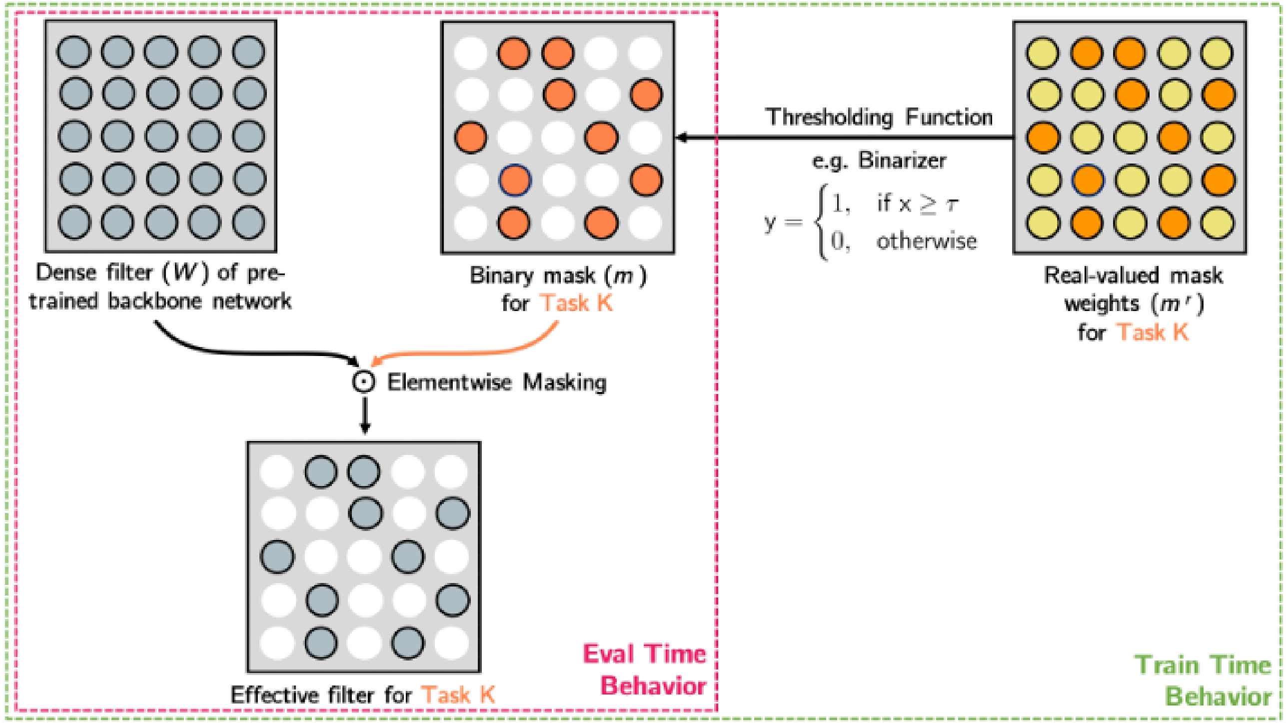
3. RELATED WORK - Continual Learning -

- **Piggyback**

  [3] Arun et al. Piggyback: Adding multiple tasks to a single, fixed network by learning to mask. ECCV 2018

  – Apply the learned weight masks to the weights of the base model to transform the output

  – The weight mask is represented by a binary mask, so the number of additional parameters is small



3. RELATED WORK - Vision Transformer -

- **ViT**

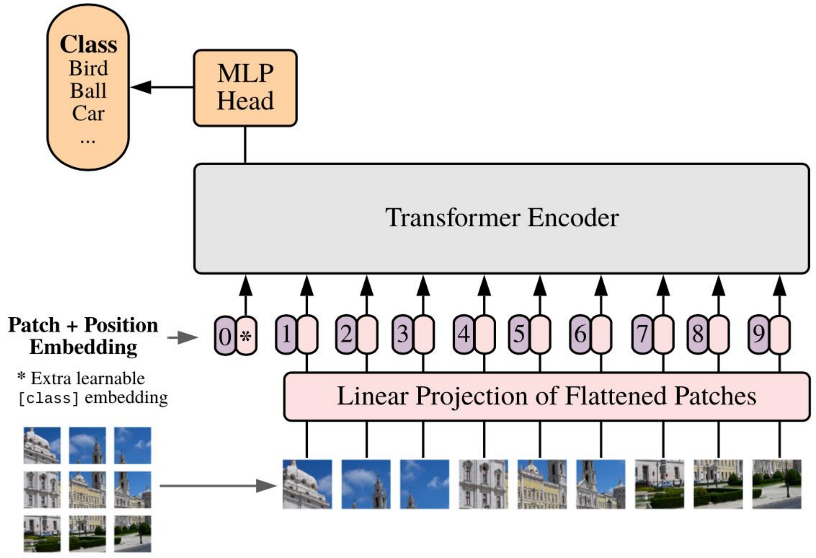
  [2] Dosovitskiy et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. ICLR 2021.

  – Method directly applying the standard Transformer to a sequence of image patches

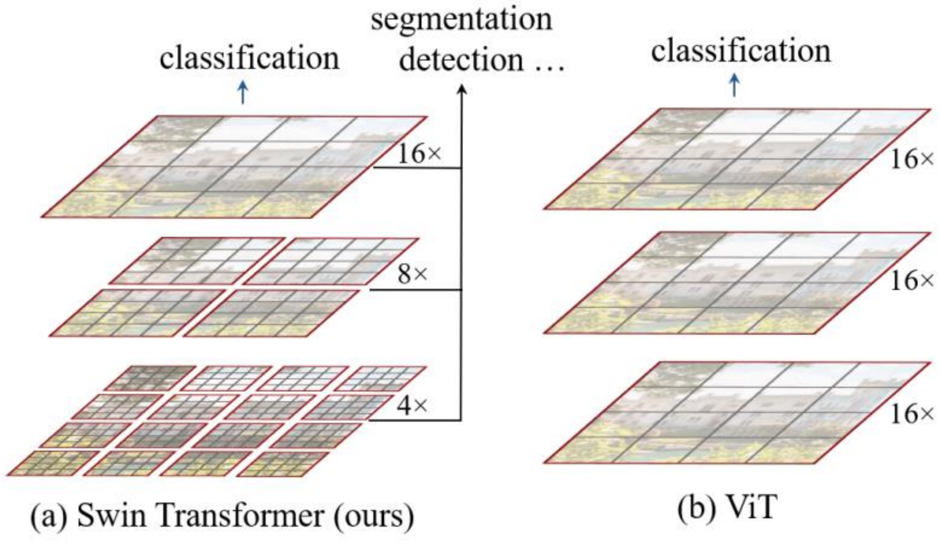
- **Swin Transformer**

  [3] Liu et al. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. CVPR 2021.

  – A method that solves the problems of ViT, such as limited resolution of object detection and a large number of input patches



▲ ViT



▲ Swin Transformer

3. RELATED WORK - Continual Learning in Vision Transformer -

- **DyTox**

  [19] Arthur et al. Dytox: Transformers for continual learning with dynamic token expansion. CVPR 2022.

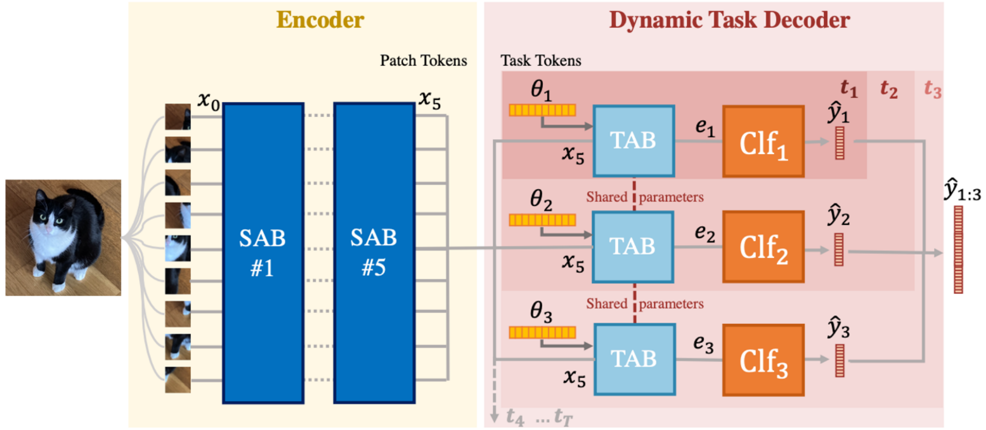
  – Use task-specific tokens to generate task-specific embedding

- **Learning to Prompt for Continual Learning (L2P)**

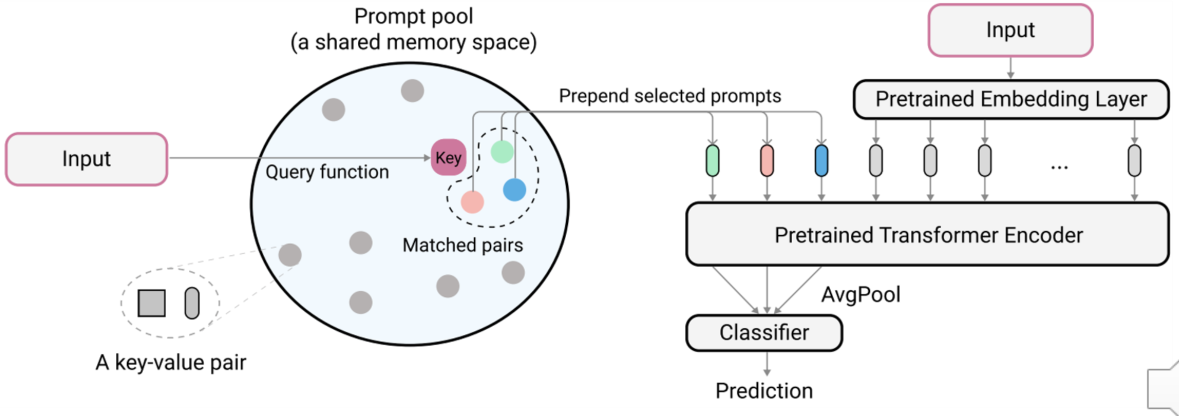
  [20] Zifeng et al. Learning to prompt for continual learning. arXiv:2112.08654, 2021.

  – Methods for applying prompt learning in the field of natural language processing

- These methods are not comparable because they are class incremental methods



▲ DyTox



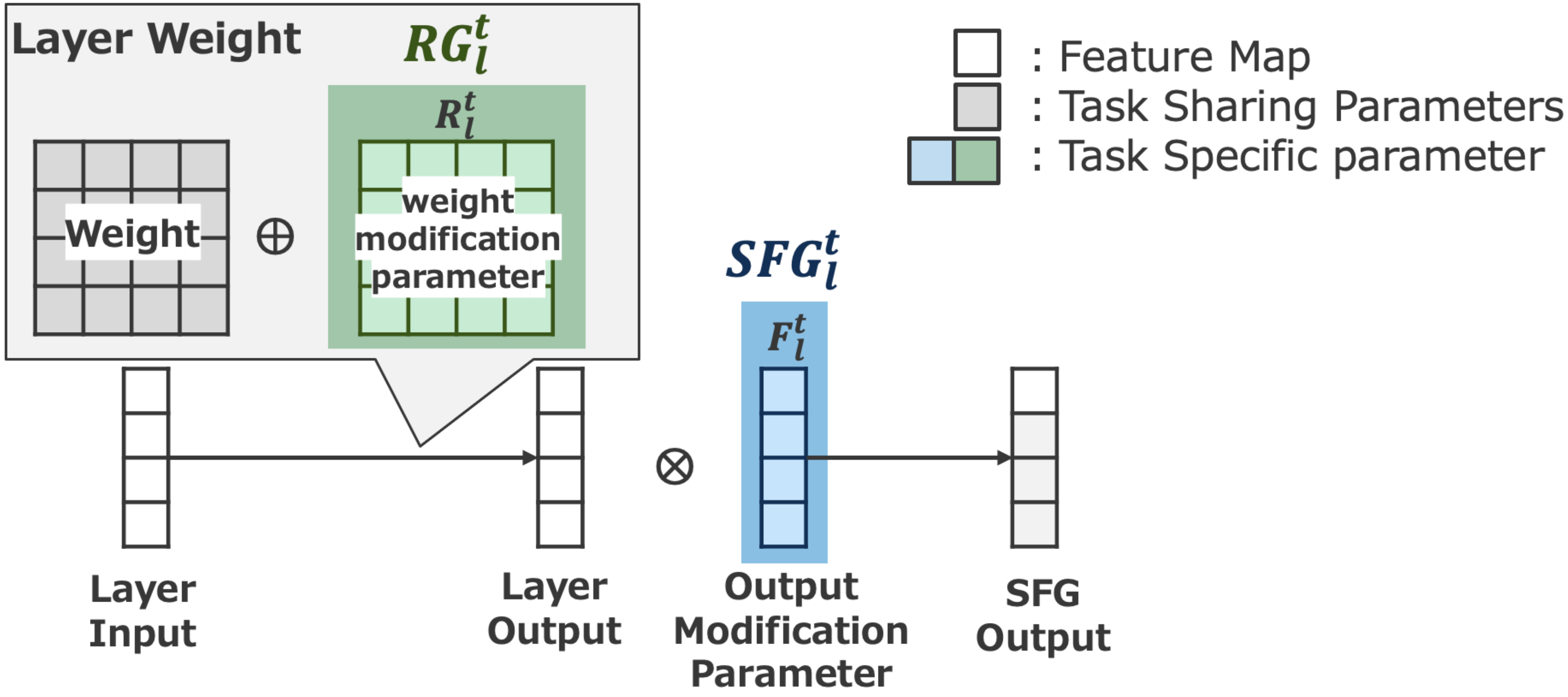
▲ L2P

4. METHOD - Method Overview -

- In this work, we propose **Mask-RKR** as a method to perform task incremental Continual Learning
- Mask-RKR is a method that applies Piggyback to the base RKR
- Main features of Mask-RKR
  - Adaptation to task by RKR
  - Parameter reduction by Piggyback

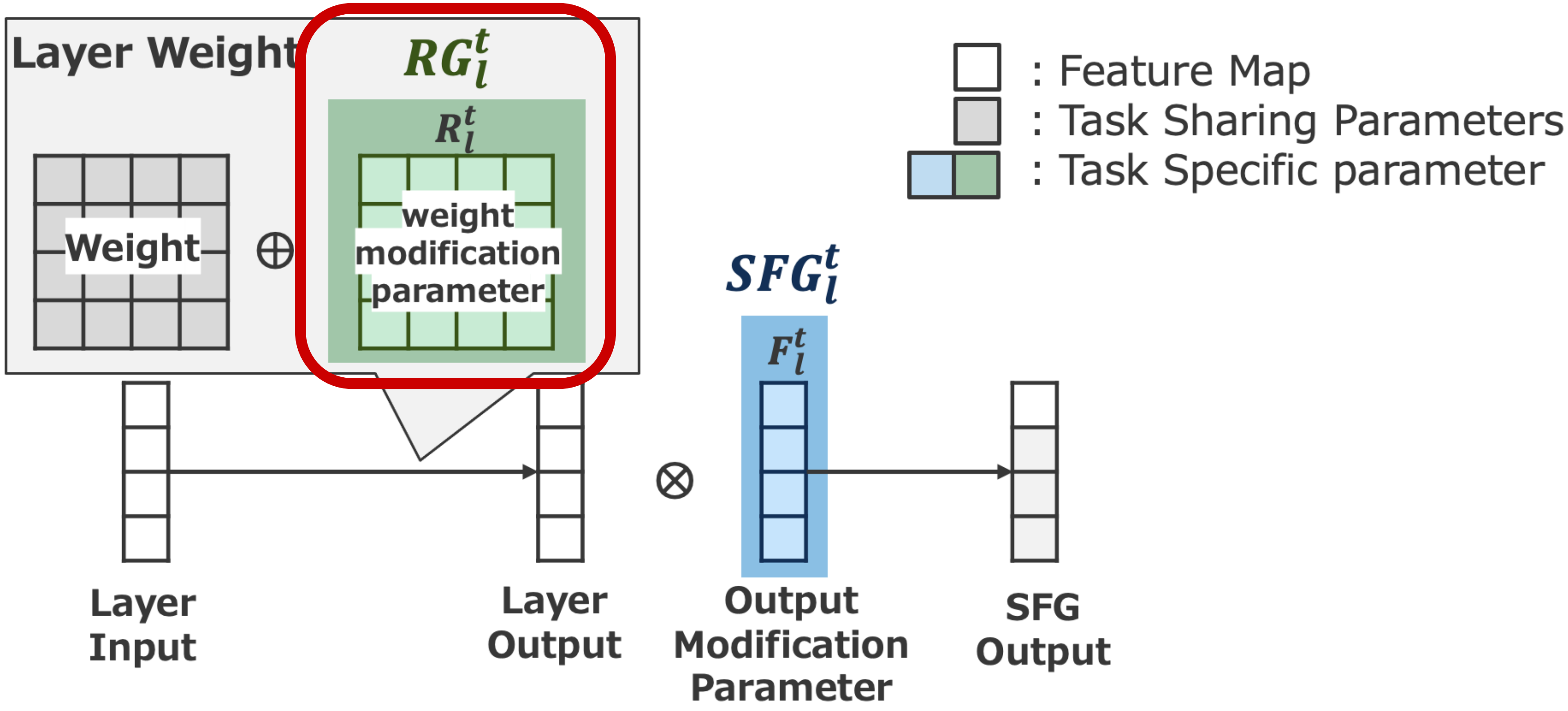
4. METHOD - Adaptation to task by RKR -

- Mask-RKR adapts the network to each task by using RKR as the base.
- RKR uses two generators, the **R**ectification **G**enerator **(RG)** and the **S**caling **F**actor **G**enerator **(SFG)**, to modify the weights and intermediate outputs of the network



RG Overview(1/2)

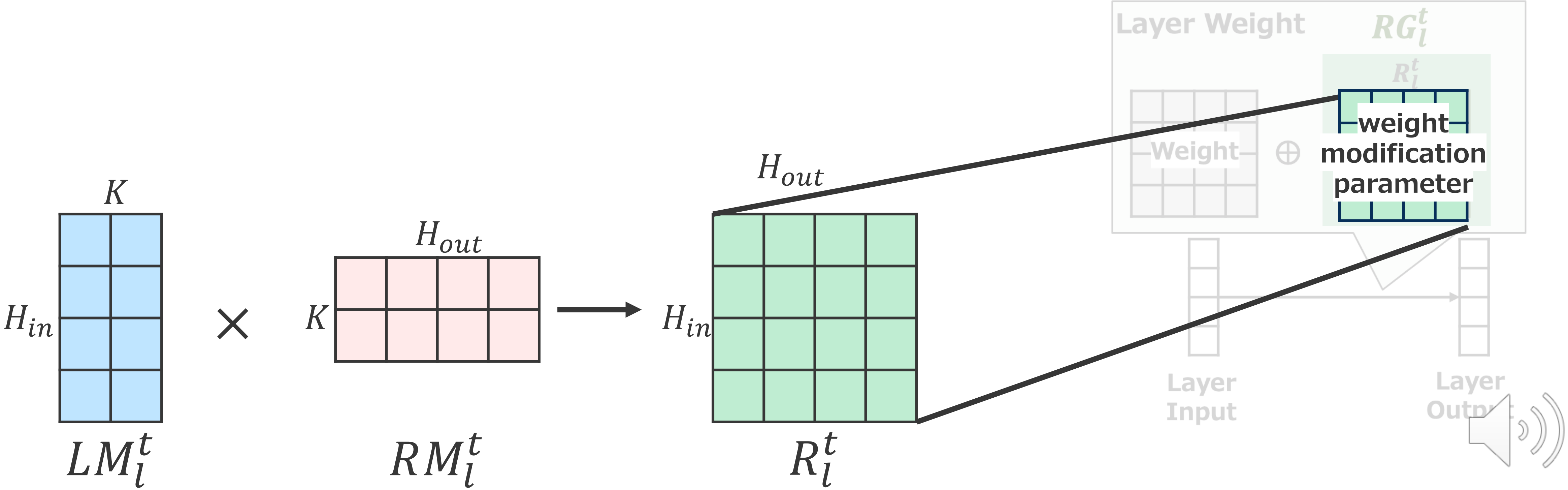
- In RG, task- and layer-specific weight modification parameters are added to the weights of each task and layer that have already been pre-trained on the large data set



RG Overview(2/2)

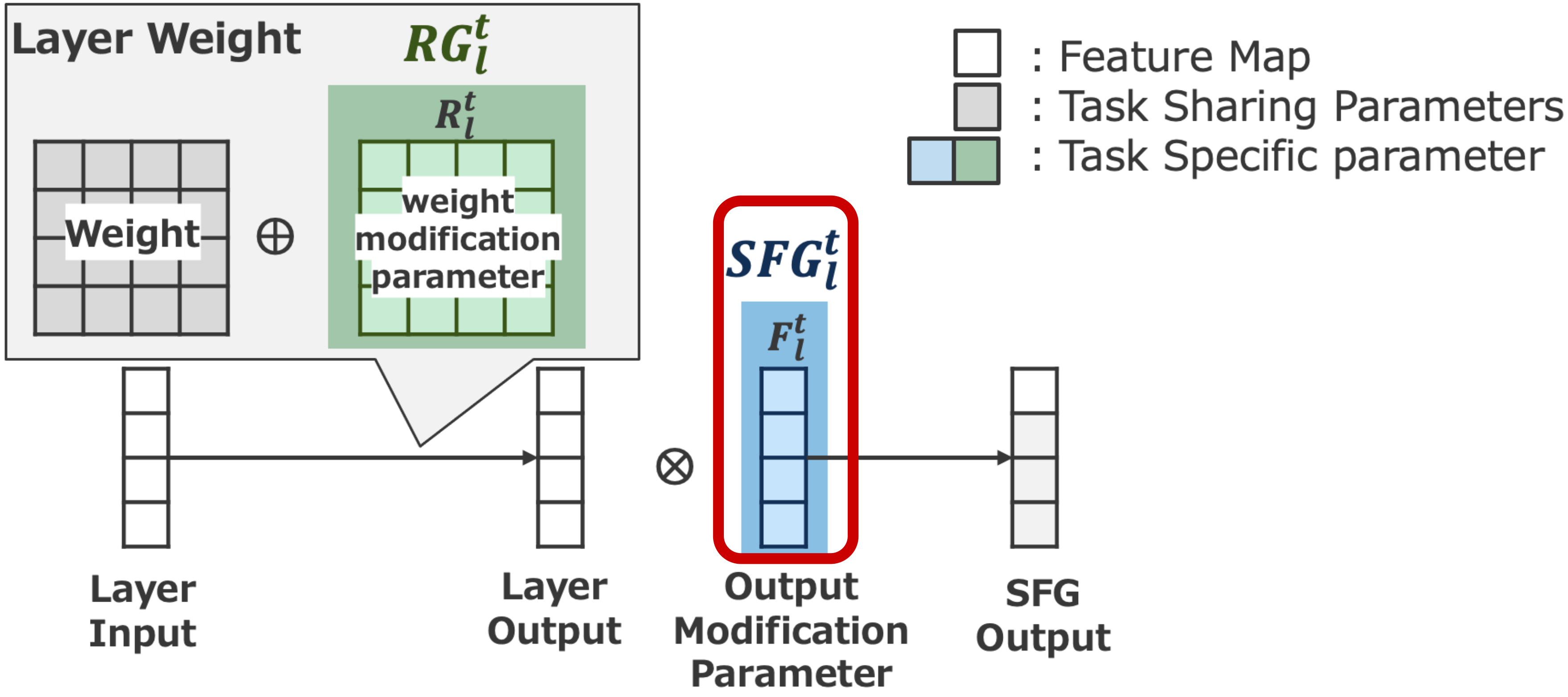
- Parameter reduction with **low-rank approximation**

- Learn two matrices LM and RM of small size and use their product to generate parameters for weight modification



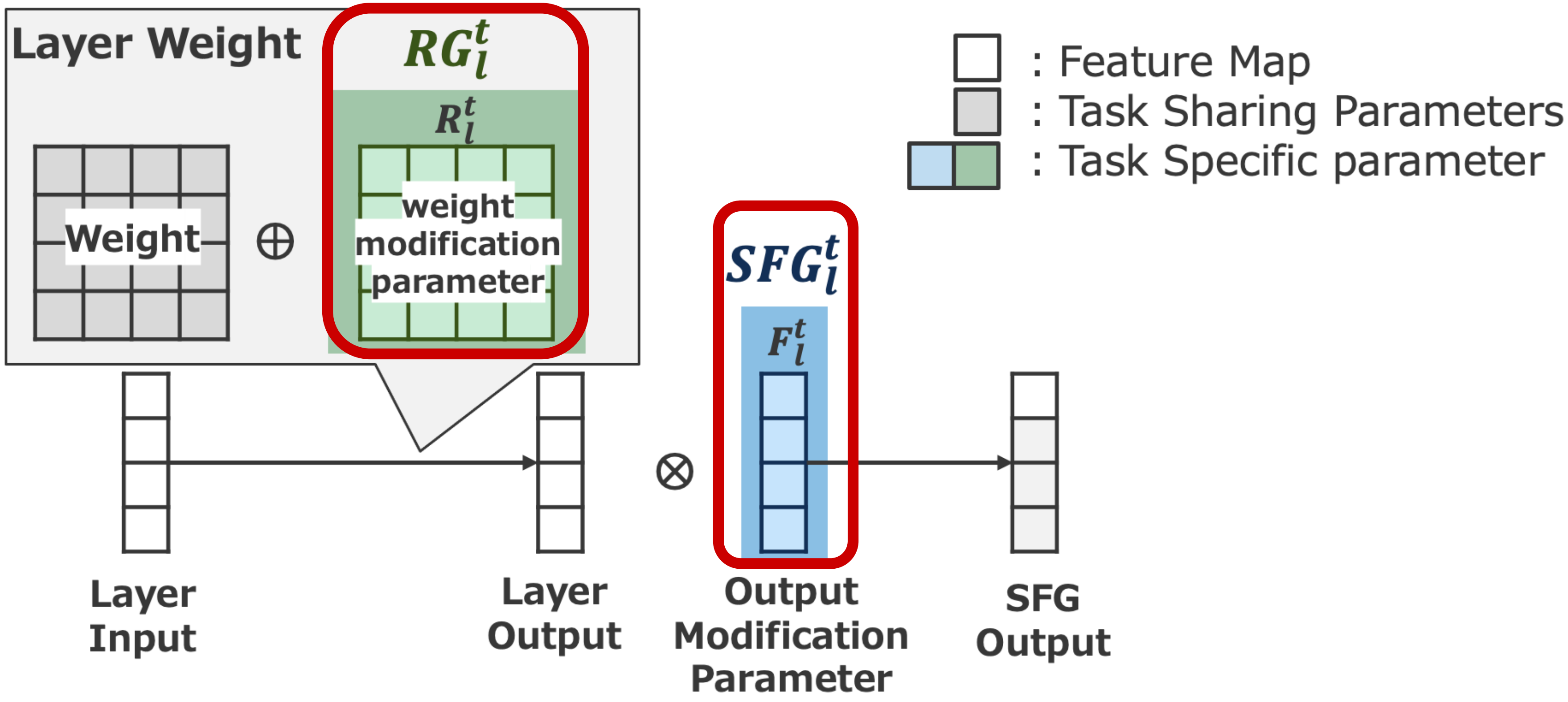
SFG Overview

- In SFG, the intermediate output of each task and layer is multiplied by the intermediate output modification parameters specific to each task and layer



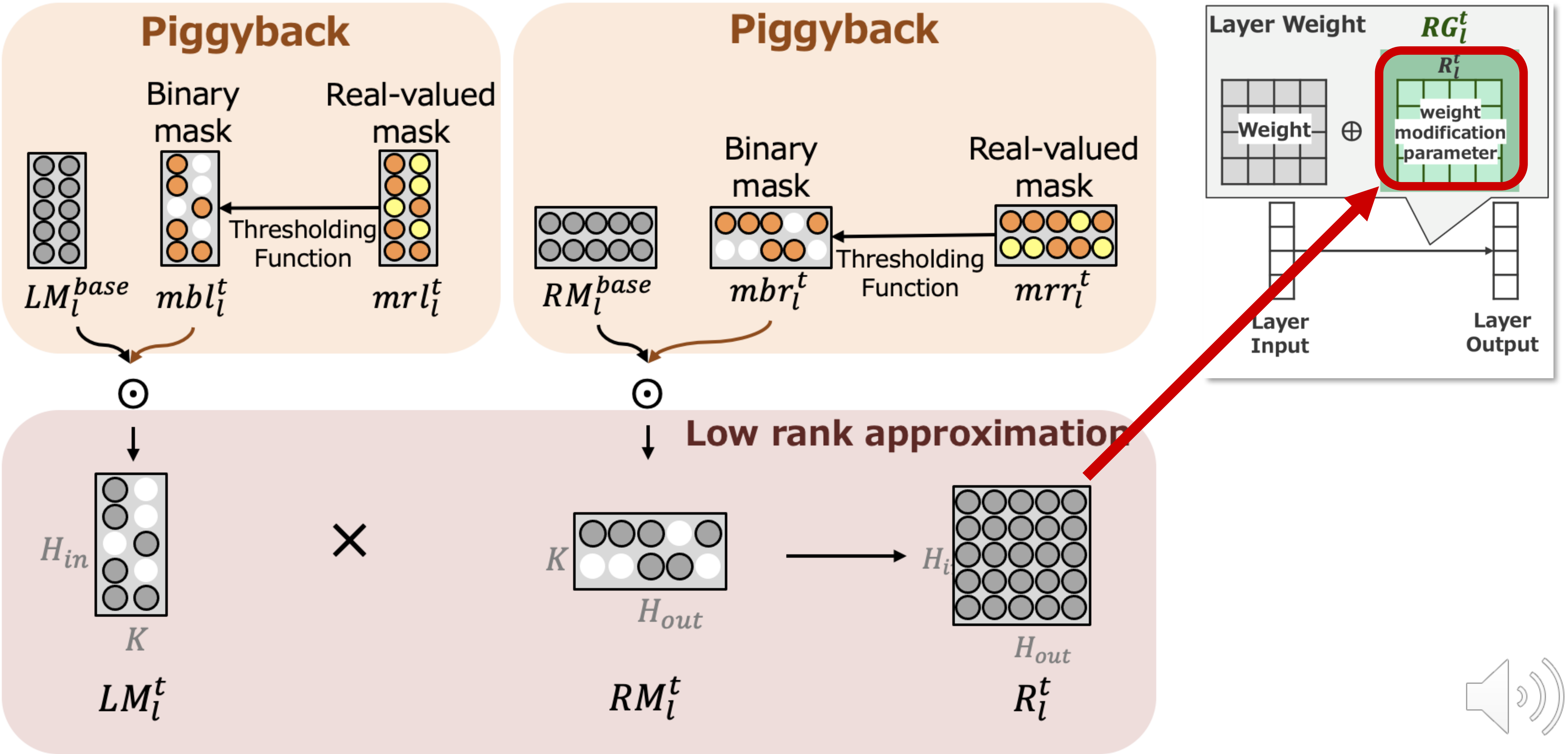
4. METHOD - Parameter reduction by Piggyback -

- Piggyback transforms the output by applying a learned weight mask to the base weights
- Mask-RKR further reduces the number of parameters by applying Piggyback to the RKR parameters

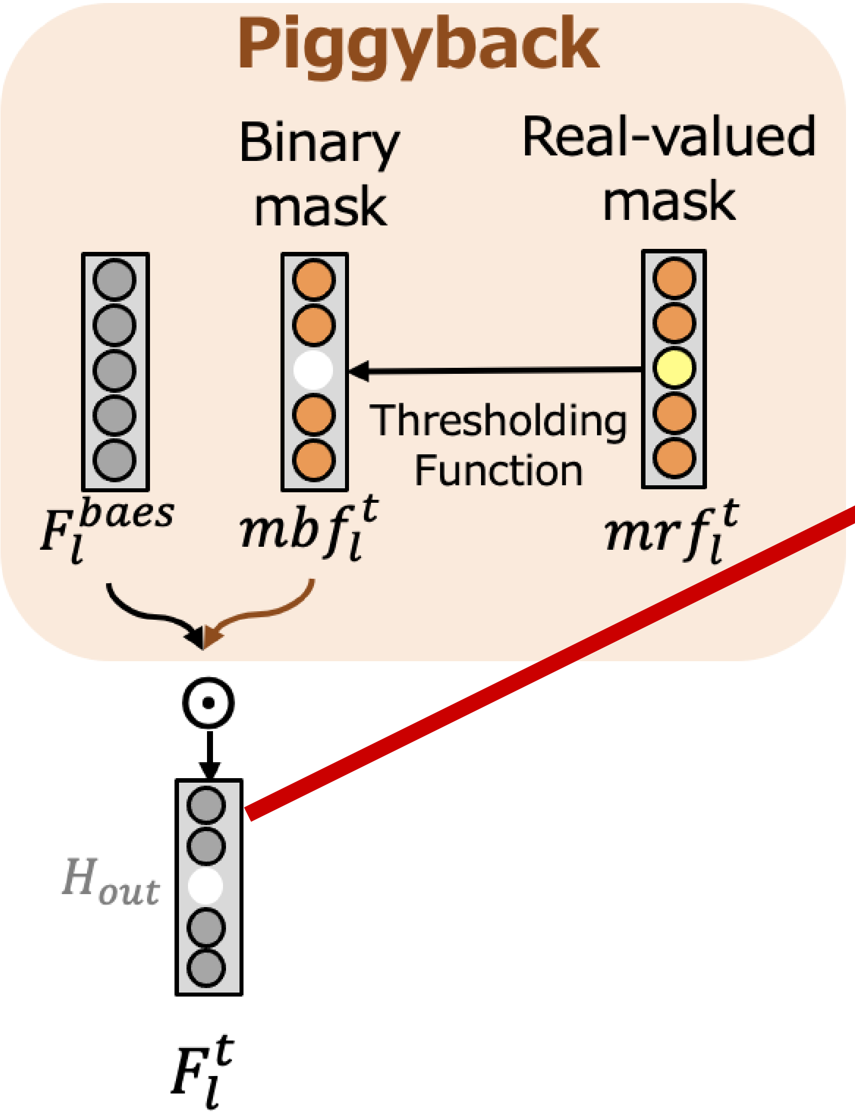


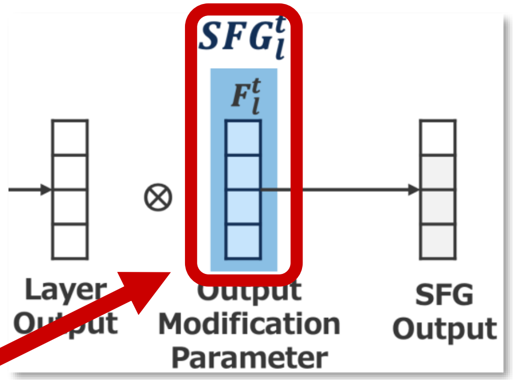
4. METHOD - Parameter reduction by Piggyback -

Parameter reduction in RG



Parameter reduction in SFG





5. COMPARISON WITH BASELINE - Experimental Overview -

- Experiments were conducted in three Continual Learning settings to verify the performance of Mask-RKR

- Model
  - ResNet-18, ViT, Swin Transformer

- Baseline
  - **Single** : Learning each task with a unique model
  - **Multi Head** : Only the final output layer is replaced for each task
  - **RKR(K=2)** : A method to modify network weights and intermediate outputs for each task
  - **Piggyback** : A method of transforming output by applying learned weight masks
  - Ours
    - **Ours(K=2)** : Mask-RKR of the proposed method
    - **Ours K+** : Mask-RKR with the same number of parameters as "RKR" by adjusting the value of K

5. COMPARISON WITH BASELINE - Experimental Overview -

- Experiments were conducted in three Continual Learning settings to verify the performance of Mask-RKR

- Model
  - ResNet-18, ViT, Swin Transformer

- Baseline
  - **Single** : Learning each task with a unique model
  - **Multi Head** : Only the final output layer is replaced for each task
  - **RKR(K=2)** : A method to modify network weights and intermediate outputs for each task
  - **Piggyback** : A method of transforming output by applying learned weight masks
  - Ours
    - **Ours(K=2)** : Mask-RKR of the proposed method
    - **Ours K+** : Mask-RKR with the same number of parameters as "RKR" by adjusting the value of K

5. COMPARISON WITH BASELINE – EX1 : Experiment using CIFAR-100 -

- Using CIFAR-100, which contains 100 classes of plants, animals, equipment, etc.
  - Divided into 10 tasks with 10 classes and studied in sequence
    (Task 1 → Task 2 → … → Task 10)

Method＼Model	Ave. Acc			Params.[M]		
	ResNet-18	ViT	Swin	ResNet-18	ViT	Swin
Single	0.833	0.857	0.876	111.72 (+900.00%)	856.59 (+900.00%)	11.98 (+900.00%)
Multi Head	0.727	0.791	0.768	11.22 (+0.41%)	85.73 (+0.08%)	1.22 (+1.45%)
RKR(K=2)	0.794	0.843	0.858	11.74 (+5.05%)	89.88 (+4.92%)	1.43 (+19.72%)
Piggyback	**0.804**	0.838	**0.875**	14.71 (+31.65%)	112.27 (+31.07%)	1.56 (+30.29%)
Ours(K=2)	0.781	0.840	0.841	11.28 (+1.01%)	86.26 (+0.70%)	1.24 (+3.79%)
Ours K+	0.796	**0.845**	0.858	11.74 (+5.05%)	89.87 (+4.92%)	1.43 (+19.56%)

5. COMPARISON WITH BASELINE – EX1 : Experiment using CIFAR-100 -

- Using CIFAR-100, which contains 100 classes of plants, animals, equipment, etc.
  - Divided into 10 tasks with 10 classes and studied in sequence (Task 1 → Task 2 → … → Task 10)

Method＼Model	Ave. Acc			Params.[M]		
	ResNet-18	ViT	Swin	ResNet-18	ViT	Swin
			0.876	111.72 (+900.00%)	856.59 (+900.00%)	11.98 (+900.00%)
			0.768	11.22 (+0.41%)	85.73 (+0.08%)	1.22 (+1.45%)
RKR(	0.794	0.843	0.858	11.74 (+5.05%)	89.88 (+4.92%)	1.43 (+19.72%)
Piggyback	**0.804**	0.838	**0.875**	14.71 (+31.65%)	112.27 (+31.07%)	1.56 (+30.29%)
Ours(K=2)	0.781	0.840	0.841	11.28 (+1.01%)	86.26 (+0.70%)	1.24 (+3.79%)
Ours K+	0.796	**0.845**	0.858	11.74 (+5.05%)	89.87 (+4.92%)	1.43 (+19.56%)

Reduces parameter increase but decreases accuracy

5. COMPARISON WITH BASELINE – EX1 : Experiment using CIFAR-100 -

- Using CIFAR-100, which contains 100 classes of plants, animals, equipment, etc.
  - Divided into 10 tasks with 10 classes and studied in sequence
    (Task 1 → Task 2 → ... → Task 10)

Achieves high accuracy while minimizing parameter increases

Method＼Model	Ave. Acc			Params.[M]		
	ResNet-18	ViT	Swin	ResNet-18	ViT	Swin
[illegible]	[illegible]	[illegible]	[illegible]76	111.72 (+900.00%)	856.59 (+900.00%)	11.98 (+900.00%)
[illegible]	[illegible]	[illegible]	[illegible]58	11.22 (+0.41%)	85.73 (+0.08%)	1.22 (+1.45%)
RKR(K=2)	0.794	<u>0.843</u>	<u>0.858</u>	11.74 (+5.05%)	89.88 (+4.92%)	1.43 (+19.72%)
Piggyback	**0.804**	0.838	**0.875**	14.71 (+31.65%)	112.27 (+31.07%)	1.56 (+30.29%)
Ours(K=2)	0.781	0.840	0.841	11.28 (+1.01%)	86.26 (+0.70%)	1.24 (+3.79%)
Ours K+	↑ <u>0.796</u>	↑ **0.845**	↑ <u>0.858</u>	↓ 11.74 (+5.05%)	↓ 89.87 (+4.92%)	↓ 1.4[illegible] (+19.56%)

5. COMPARISON WITH BASELINE – EX2 : Experiment using ImageNet-1k -

- Using ImageNet-1k, a large dataset with 1000 classes
  - Split into 10 tasks with 100 classes and train them in sequence (Task 1 → Task 2 → ... → Task 10)

Method＼Model	Ave. Acc			Params.[M]		
	ResNet-18	ViT	Swin	ResNet-18	ViT	Swin
Single	0.678	0.888	0.902	112.18 (+900.00%)	858.76 (+900.00%)	868.46 (+900.00%)
			87	11.68 (+4.12%)	86.57 (+0.81%)	87.77 (+1.06%)
			92	12.20 (+8.73%)	90.71 (+5.64%)	92.34 (+6.33%)
Piggyba	0.440	0.881	0.805	15.17 (+35.22%)	113.11 (+31.71%)	113.94 (+31.20%)
Ours(K=2)	0.557	0.879	0.870	11.75 (+4.71%)	87.10 (+1.42%)	88.35 (+1.74%)
Ours K+	⬆ **0.582**	⬆ **0.885**	⬆ **0.894**	⬇ 12.43 (+10.83%)	⬇ 90.71 (+5.63%)	⬇ 92.3 (+6.28%)

Achieves high accuracy while minimizing parameter increases

- Use datasets from different domains
  - 5 tasks trained in sequence

    (**D. Textures → GTSRB → SVHN → UCF101 → VGG-Flower**)

Method＼Model	Ave. Acc			Params.[M]		
	ResNet-18	ViT	Swin	ResNet-18	ViT	Swin
Single	0.776	0.816	0.842	111.91 (+900.00%)	857.39 (+900.00%)	594.62 (+900.00%)
[illegible]	[illegible]	[illegible]	0.682	11.32 (+1.17%)	85.89 (+0.18%)	59.59 (+0.22%)
[illegible]	[illegible]	[illegible]	**.840**	11.58 (+3.49%)	87.97 (+2.60%)	61.49 (+3.41%)
Piggyba	**0.723**	**0.809**	<u>0.839</u>	13.07 (+16.76%)	99.16 (+15.66%)	68.75 (+15.62%)
Ours(K=2)	0.695	0.775	0.824	11.38 (+1.71%)	86.36 (+0.72%)	60.02 (+0.94%)
Ours(K+)	<u>0.720</u>	0.778	0.831	11.52 (+2.95%)	87.67 (+2.25%)	61.39 (+3.24%)

Reduces parameter increase

but decreases accuracy

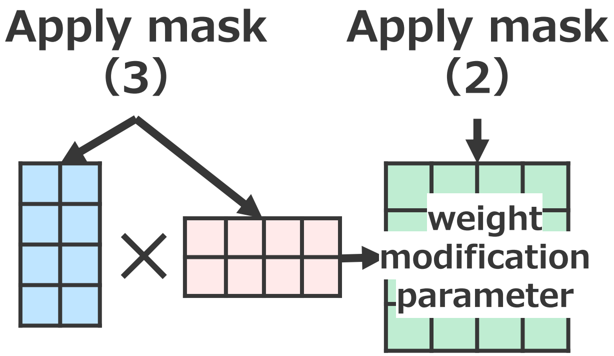
6. ABLATION EXPERIMENT - Verification of the usefulness of the mask -

- The usefulness was verified by comparing RG and SFG w/ and w/o applying masks to each.
  - "RG w/ Mask": Apply mask to RG
  - "SFG w/ Mask": Apply mask to SFG
- In this experiment, the model with **Piggyback applied to RG and SFG** with the lowest number of parameters is used

RG w/ Mask	SFG w/ Mask	Ave. Acc			Params.[M]		
		ResNet-18	ViT	Swin	ResNet-18	ViT	Swin
x	x	**0.794**	0.843	**0.858**	11.74 (+5.05%)	89.88 (+4.92%)	1.43 (+19.72%)
✓	x	0.780	0.844	0.846	11.33 (+1.38%)	87.07 (+1.64%)	1.28 (+6.59%)
x	✓	**0.794**	**0.845**	**0.858**	11.69 (+4.68%)	89.15 (+4.08%)	1.40 (+17.20%)
✓	✓	0.781	0.840	0.841	11.28 (+1.01%)	86.26 (+0.70%)	1.24 (+3.79%)

6. ABLATION EXPERIMENT - Verification of Piggyback application locations -

- Verified where masks are applied in RG
  (1) Not applied
  (2) Applied to weight modified parameters
  (3) Applied to low-rank approximated parameters (Mask-RKR)

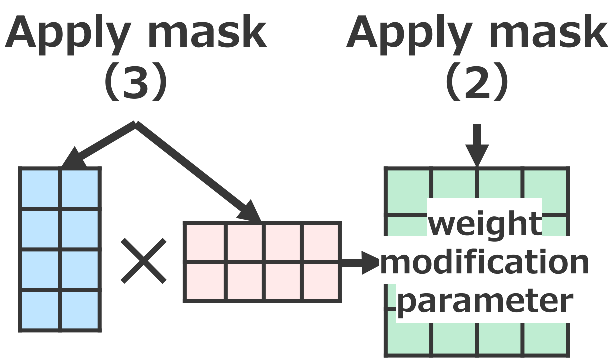


- To reduce the number of parameters, it is more effective to **apply Piggyback to each of LM and RM**

Method	Ave. Acc			Params.[M]		
	ResNet-18	ViT	Swin	ResNet-18	ViT	Swin
(1)	0.794	**0.845**	**0.858**	11.69 (+4.68%)	89.15 (+4.08%)	1.40 (+17.20%)
(2)	**0.805**	**0.845**	0.847	14.41 (+29.00%)	110.05 (+28.48%)	1.55 (+29.32%)
(3)	0.781	0.840	0.841	11.28 (+1.01%)	86.26 (+0.70%)	1.24 (+3.79%)

6. ABLATION EXPERIMENT - Verification of Piggyback application locations -

- Verified where masks are applied in RG
  - (1) Not applied
  - (2) Applied to weight modified parameters
  - (3) Applied to low-rank approximated parameters (Mask-RKR)



- To reduce the number of parameters, it is more effective to **apply Piggyback to each of LM and RM**

Method	Ave. Acc			Params.[M]		
	ResNet-18	ViT	Swin	ResNet-18	ViT	Swin
(1)	0.794	**0.845**	**0.858**	11.69 (+4.68%)	89.15 (+4.08%)	1.40 (+17.20%)
(2)	➡ **0.805**	➡ **0.845**	➡ 0.847	⬆ 14.41 (+29.00%)	⬆ 110.05 (+28.48%)	⬆ 1.55 (+29.32%)
(3)	0.781	0.840	0.841	⬇ 11.28 (+1.01%)	⬇ 86.26 (+0.70%)	⬇ 1.24 (+3.79%)

7. CONCLUSION

- We proposed Mask-RKR, a continual learning method that can be applied to both CNN and Vision Transformer

- Experimental results show that Mask-RKR can achieve higher accuracy than conventional methods while minimizing the increase in the number of parameters

- In the future, we would like to improve Mask-RKR to make it flexible enough to handle continuous learning using datasets from different domains

UEC
TOKYO