CONTINUAL LEARNING IN VISION TRANSFORMER

Mana Takeda *Keiji Yanai*

The University of Electro-Communications, Tokyo, Japan

ABSTRACT

Continual learning aims to continuously learn new tasks from new data while retaining the knowledge of tasks learned in the past. Recently, the Vision Transformer, which utilizes the Transformer initially proposed in natural language processing for computer vision, has shown higher accuracy than Convolutional Neural Networks (CNN) in image recognition tasks. However, there are few methods that have achieved continual learning with Vision Transformer. In this paper, we compare and improve continual learning methods that can be applied to both CNN and Vision Transformers. In our experiments, we compare several continual learning methods and their combinations to show the differences in accuracy and the number of parameters.

Index Terms— Continual Learning, Vision Transformer

1. INTRODUCTION

In a deep learning model, when multiple tasks are given sequentially, the previously learned tasks are overwritten by the new tasks and forgotten. This is called catastrophic forgetting. Continual learning deals with catastrophic forgetting by continuously learning new tasks from new data while retaining the knowledge of previously learned tasks.

Recently, Vision Transformer [1, 2], which utilizes the Transformer architecture initially proposed in natural language processing for computer vision, has shown higher accuracy than CNN in image recognition tasks. However, conventional continual learning methods are generally considered to be applied to the convolutional layer of CNN, and therefore, the methods that can be applied to all the combined layers of Vision Transformer are limited. Furthermore, there is no method that has been shown to be effective for both CNN and Vision Transformer.

Therefore, the purpose of this paper is to investigate methods that can be applied to both CNN and Vision Transformer, and can suppress catastrophic forgetting with a small number of additional parameters. By comparing the accuracy and the number of extra parameters of each method, we examine whether the methods are effective not only for CNN but also for Vision Transformer. Furthermore, we aim to propose a method with higher performance by combining conventional methods.

2. RELATED WORKS

Continual learning aims to balance two trade-offs: rigidity to change and plasticity to adapt so that new data is learned but past data is not forgotten [3, 4, 5]. Singh et al. [6] proposed Recognition-based Knowledge Retention (RKR), a method for modifying the weights and intermediate activations of a network for each task in continual learning. The weight modification is done by adding parameters generated by a generator called Rectification Generator (RG) to the weights of the convolutional layer. The intermediate activation is modified by multiplying the output of the convolutional layer by a parameter generated by a generator called Scaling Factor Generator (SFG). Piggyback [7] is a method for learning a large number of tasks with high accuracy by transforming the output by applying a learned weight mask to the weights of the base model. RKR and Piggyback can be applied to the fully connected layer, and can be applied to Vision Transformer.

Transformer was first introduced in machine translation in natural language processing [8] and is now a common method. ViT [1] proposed to apply the Transformer to computer vision by using a patch of pixels as a token. Swin Transformer [2] is a method that solves the problem of adapting the Transformer from language to vision by computing it hierarchically using a shifted window. Specifically, they dealt with large changes in the scale of visual entities and the high resolution of pixels in images compared to words in texts. In this paper, we use ViT and Swin Transformer to verify the performance of the continual learning method in Vision Transformer.

Recently, several continuous learning methods specific to Vision Transformer have been proposed. DyTox [9] is a method where the initial layer is shared by all tasks and task-specific tokens are used to generate task-specific embeddings. Learning to Prompt for Continual Learning (L2P) [10] is a method inspired by prompt learning [11], a new continuous learning method in the field of natural language processing. These methods do not apply to CNN as they are specialized in Vision Transformer. Furthermore, these methods are methods for class incremental learning, which is a different purpose from the purpose of this paper, which is task incremental learning.

3. METHOD

In this paper, we present a comparative study of continual learning methods applicable to CNN and Vision Transformer, which are RKR [6] and Piggyback [7]. Furthermore, we propose a new method, Mask-RKR, which is a combination of these two methods. By applying Piggyback [7] to the base RKR [6], Mask-RKR retains the characteristics of RKR while reducing the number of additional parameters due to the increase in the number of tasks.

3.1. Adaptation to the task by RKR

In Mask-RKR, RKR is applied to all the fully connected layers of the Vision Transformer except the final output layer. For each layer, the weights are modified by RG and the intermediate activations are modified by SFG. The RKR for all

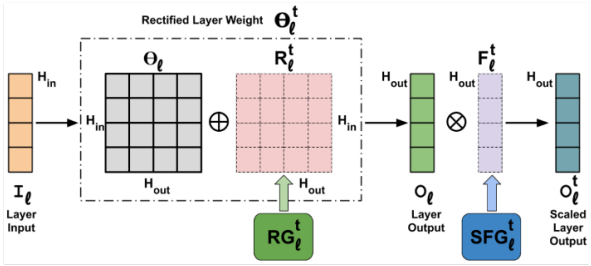


Fig. 1: Overview of adaptation to tasks with RKR in Mask-RKR(Quote from [6])

the fully connected layers in the Vision Transformer is shown in the Figure 1. In the case of the lth layer of task t, the RG_l^t generates the weights θ_l^t of all the coupled layers adapted to each task. θ_l^t is then applied to input I_l to produce the output O_l. SFG_l^t generates a scaling factor F_l^t, which is applied to O_l to produce the scaled output O_l^t.

The modification of the weights by RG is shown in Equation (1).

$$\Theta_l^t = \Theta_l \oplus R_l^t \quad (1)$$

$$R_l^t = MATMUL(LM_l^t, RM_l^t) \quad (2)$$

In RG, a parameter R is learned to modify the weights, and R_l^t is added to the weights of layer l of task t, respectively. Here, Θ_l is the weight of layer l, Θ_l^t is the modified weight of layer l in task t, and $\oplus$ is the sum of each element. To reduce the number of parameters in RG, we use a low-rank approximation of R_l^t. This is shown in Equation (2). In RG, two matrices of small size, LM_l^t and RM_l^t, are learned, and the product of these two matrices generates the parameter R_l^t for weight modification. In the fully connected layer, a filter R_l^t of size $H_{in} \times H_{out}$ is generated from a matrix LM_l^t of size $H_{in} \times K$and a matrix RM_l^t of size $K \times H_{out}$, Where H_{in} and H_{out} are the input and output sizes of fully connected layers, respectively. MATMUL is the matrix product. K is the rank of the low-rank approximation. Here, $K << H_{in}$ and $K << H_{out}$.

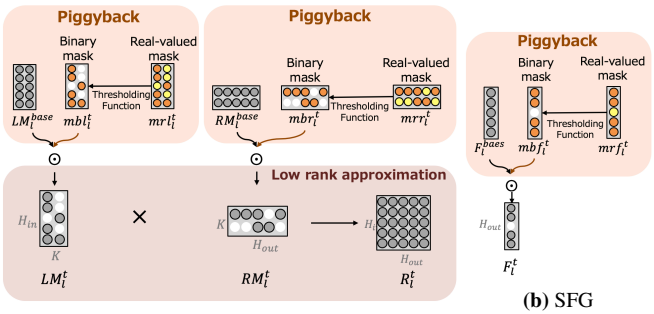
The modification of the intermediate activation by SFG is shown in Equation Equation (3).

$$O_l^t = O_l \odot F_l^t \quad (3)$$

In SFG, a parameter F is learned to modify the intermediate activations, and F_l^t is multiplied by the intermediate activation of layer l in task t. Here, O_l is the output of layer l, O_l^t is the modified output of layer l in task t, and $\odot$ is the per-element product. In SFG, F_l^t is learned. In the fully connected layer, the size of the F_l^t is H_{out}. Compared to RG, SFG does not require a low-rank approximation of matrices because it generates fewer parameters.

3.2. Parameter reduction by Piggyback

Piggyback is a method of transforming the output by applying a learned weight mask to the weights of the base model. Mask-RKR does not apply Piggyback directly to the weights, but the parameters of RKR. Specifically, it is applied to the two low-rank approximated parameters LM and RM of RKR



(a) RG **(b)** SFG

Fig. 2: Parameter Reduction in Mask-RKR

and the parameter F of SFG. Mask learning is accomplished by keeping a set of real-valued weights, passing them through a deterministic threshold function to obtain a binary mask, and applying it to the existing weights. By updating the real-valued weights using back propagation, a binary mask suitable for the task is learned. By learning different binary masks for each task and applying them to the RG and SFG parameters element by element, the same base network can be reused for multiple tasks with a minimum of additional parameters.

The procedure of mask learning in RG of Mask-RKR is shown in Figure 2a. In the RG, the weight-modifying parameter R_l^t is added to the weights of layer l of task t, respectively. The R_l^t is approximated by a low-rank approximation for parameter reduction and decomposed into two matrices LM_l^t and RM_l^t for training. where $R_l^t \in \mathbb{R}^{H_{in} \times H_{out}}$ and $RM_l^t \in \mathbb{R}^{K \times H_{out}}$ are used for the full coupling layer. First, we use ImageNet-1k for training only the base parameters LM_l^{base} and RM_l^{base}. Next, in task t, we learn only real-valued masks while keeping LM_l^{base} and RM_l^{base} fixed. Real-valued masks are learned for both LM and RM, with the real-valued mask mrl_l^t for LM having the same size as LM_l^1 and the real-valued mask mrr_l^t for RM having the same size as RM_l^1. Each real-valued mask is passed through the hard binary thresholding function given in Equation (4), as in [7], to obtain the thresholded mask matrices mbl_l^t and mbr_l^t. Where mr is the real-valued mask, mb is the binary mask, and τ is the chosen threshold. In the binary mask, the base parameter is activated or deactivated depending on whether a particular value mb_{ji} of the thresholded mask matrix is 0 or 1.

$$mb_{ji} = \begin{cases} 1, & \text{if } mr_{ji} \geq \tau \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

Using the binary mask generated by the thresholding process, the modification of the weights by RG expressed in Equation (2) is replaced by Equation (5) in the task t. Here, $\odot$ denotes the per-element product or masking.

$$\begin{aligned} LM_l^t &= LM_l^{base} \odot mbl_l^t \\ RM_l^t &= RM_l^{base} \odot mbr_l^t \\ R_l^t &= MATMUL((LM_l^{base} \odot mbl_l^t) \\ &\quad , (RM_l^{base} \odot mbr_l^t)) \end{aligned} \quad (5)$$

Next, the procedure of mask learning in SFG of Mask-RKR is shown in Figure 2b. In SFG, the intermediate activation of layer l of task t is multiplied by F_l^t. Here, $F_l^t \in \mathbb{R}^{H_{out}}$

Table 1: Dataset used in Experiment 3

Dataset	Description	No. of Classes	Training Data	Test Data
D. Textures	Textures and Patterns	47	1,880	1,880
GTSRB	Road signs in Germany	4	31,367	7,842
SVHN	Digit Image	10	47,217	26,040
UCF101	Action	101	7,629	1,908
VGG-Flowers	Flower	102	1,020	1,020

for the fully connected layer. First, in SFG, as in RG, the base network is trained in ImageNet-1k. We use ImageNet-1k for training not the mask, but only F_l^{base}. Next, in task t, we learn only the real-valued mask while keeping the base parameter F_l^{base} learned in ImageNet-1k fixed. The real-valued mask of F_l^t, mrf_l^t, is the same size as F_l^1. Each real-valued mask is passed through the hard binary thresholding function given in Equation (4), as in [7], to obtain the thresholded mask matrix mbf_l^t.

After learning the mask for a given task, the weights of the real-valued mask are no longer needed. The real-valued mask is discarded and only the binary mask is stored.

4. EXPERIMENTS

4.1. Experiment Summary

In this paper, we conducted a comparison experiment of continual learning methods applicable to CNN and Vision Transformer, and an ablation experiment of the proposed method, Mask-RKR. In the comparison experiments, we used three models and datasets with different numbers of classes and domains to compare the performance in three task incremental continual learning settings. In the ablation experiments, we examined the effects of the components of Mask-RKR.

In Experiment 1 and the ablation experiments, we used CIFAR-100, which is a dataset containing 100 classes of animals, plants, devices, and vehicles. To run the experiments in the continual learning setting, CIFAR-100 was divided into 10 tasks with 10 classes each. The image size is 32×32. We used 50,000 images for the training data and 10,000 images for the test data. In Experiment 2, we used ImageNet-1k, which is a large dataset containing 1,000 classes. To run the experiment in a continual learning setting, we divided ImageNet-1k into 10 tasks, each with 100 classes. The image size is 224×224. We used 1,232,167 images for training data and 49,000 images for test data. In Experiment 3, we used five datasets of different domains from the Visual Decathlon (VD) benchmark [12]. The VD benchmark is a benchmark that evaluates the ability to solve 10 different visual domains simultaneously. The datasets used are shown in Table 1. The image size is 72×72 for all datasets.

In this experiment, we used three models: ResNet-18 [13], ViT [1], and Swin Transformer [2]. ResNet-18 is based on CNN, while ViT and Swin Transformer are based on the architecture of Vision Transformer. Since Swin Transformer requires a model that corresponds to the size of the input image, we used the Swin-Tiny model with a small model size in Experiment 1, Experiment 3, and the ablation experiment, and the Swin-Base model, which is a general model of Swin Transformer, in Experiment 2. All the models used in the experiments were pre-trained models in ImageNet-1k.

In our experiments, we compared the performance of Mask-RKR with that of conventional continuous learning methods. "Single" is a method in which each task is trained with a separate model. "Multi Head" is a method in which only the final output layer is trained for each task. "RKR" [6] is a method to modify the weights and intermediate activations of the network for each task. In all the experiments, the rank K of the low-rank approximation was $K = 2$, which has the lowest number of parameters. "Piggyback" [7] is a method to transform the output by applying the learned weight mask. The threshold of the hard binary threshold function was set to 5e-3 as in [7]. "Mask-RKR" is a method in which the number of parameters is reduced by applying Piggyback to the base RKR. The threshold τ of the hard binary threshold function of Piggyback is set to 5e-3 as in "Piggyback". The rank K of the low-rank approximation was tested in two cases: the first was "Mask-RKR (K=2)" with $K = 2$ and the lowest number of parameters, and the second was "Mask-RKR K+" with the same number of parameters as "RKR" by increasing the value of K. The base parameters of all Mask-RKRs used in the experiments were those pre-trained with ImageNet-1k.

In all the experiments, we used an SGD as an optimizer with an initial learning rate of 3e-2 and momentum of 0.9. The scheduler was set to linear warmup and cosine decay. For the loss function, we used Cross Entropy Loss. The batch size was 100. For the evaluation metric, we used the average accuracy of all trained tasks.

Table 2: Results of the experiment using CIFAR-100 (Exp. 1)

Method	Ave. Acc			Params.[M]		
	ResNet-18	ViT	Swin	ResNet-18	ViT	Swin
Single	0.833	0.857	0.876	111.72	856.59	11.98
Multi Head	0.727	0.791	0.768	11.22	85.73	1.22
RKR(K=2)	0.794	0.843	0.858	11.74	89.88	1.43
Piggyback	**0.804**	0.838	**0.875**	14.71	112.27	1.56
Mask-RKR(K=2)	0.781	0.840	0.841	11.28	86.26	1.24
Mask-RKR K+	0.796	**0.845**	0.858	11.74	89.87	1.43

Table 3: Results of the experiment using ImageNet-1k(Exp. 2)

Method	Ave. Acc			Params.[M]		
	ResNet-18	ViT	Swin	ResNet-18	ViT	Swin
Single	0.678	0.888	0.902	112.18	858.76	868.46
Multi Head	0.523	0.871	0.887	11.68	86.57	87.77
RKR(K=2)	0.545	**0.885**	0.892	12.20	90.71	92.34
Piggyback	0.440	0.881	0.805	15.17	113.11	113.94
Mask-RKR(K=2)	0.557	0.879	0.870	11.75	87.10	88.35
Mask-RKR K+	**0.582**	**0.885**	**0.894**	12.43	90.71	92.30

Table 4: Results of experiments on data sets from different domains(Exp. 3)

Method	Ave. Acc			Params.[M]		
	ResNet-18	ViT	Swin	ResNet-18	ViT	Swin
Single	0.776	0.816	0.842	111.91	857.39	594.62
Multi Head	0.567	0.625	0.682	11.32	85.89	59.59
RKR(K=2)	0.714	0.791	**0.840**	11.58	87.97	61.49
Piggyback	**0.723**	**0.809**	0.839	13.07	99.16	68.75
Mask-RKR(K=2)	0.695	0.775	0.824	11.38	86.36	60.02
Mask-RKR K+	0.720	0.778	0.831	11.52	87.67	61.39

4.2. Comparative experiments

In the comparative experiment, we examined the performance of the continual learning methods on Vision Transformer.

In Experiment 1, we divided CIFAR-100 into 10 tasks with 10 classes. We examined the performance in a relatively simple continual learning setting, where the number of classification classes was 10 and each task was in a similar domain. The experimental results are shown in Table 2. "Ave.

Acc" indicates the average accuracy, and "Params." indicates the number of parameters. Bold values in the table are the most accurate values. The K values of "Mask-RKR K+" are 17 for ResNet-18, 19 for ViT, and 18 for Swin Transformer. "Multi Head" only requires a final layer for each task, so the number of additional parameters is quite small. However, the accuracy is inferior to the other methods, indicating that the model cannot be successfully applied to tasks simply by replacing the final layer. The experimental results show that "RKR" and "Piggyback" have higher accuracy while reducing the number of parameters. Both of them are close to each other in performance, but "Piggyback" is better in accuracy, and "RKR" is better in controlling the number of parameters. "Mask-RKR (K=2)" has a lower accuracy than "RKR" and "Piggyback" despite its lower parameter count. However, "Mask-RKR K+" has the same number of parameters as "RKR", but shows higher accuracy than "RKR". Furthermore, Mask-RKR showed the same or better accuracy than "Piggyback" while reducing the number of parameters. These results show that Mask-RKR can achieve high accuracy while minimizing the increase in the number of parameters.

In Experiment 2, we divided ImageNet-1k into 10 tasks with 100 classes. Although each task is in a similar domain, the number of classification classes is 100, and we verify the accuracy of the proposed method in the continual learning setting, which is more difficult than Experiment 1. The K values of "Mask-RKR K+" are 3 for ResNet-18, 18 for ViT, and 19 for Swin Transformer. The results of the experiments are shown in Table 3. As in Experiment 1, "RKR," "Piggyback" and "Mask-RKR" showed the highest accuracy overall for all models. "Mask-RKR (K=2)" has the same accuracy as "RKR" and "Piggyback", but the increase in the number of parameters is smaller. In addition, "Mask-RKR K+" showed higher accuracy than "RKR" despite having the same number of parameters as "RKR". This suggests that Mask-RKR is the most effective even in the continual learning setting, which is a more difficult setting than Experiment 1.

In Experiment 3, the five different domains (Textures, GTSRB, SVHN, UCF101, and VGG-Flower) are trained in order. The K values of "Mask-RKR K+" are 9 for ResNet-18, 11 for ViT, and 10 for Swin Transformer. The experimental results are shown in Table 4. As in Experiments 1 and 2, "RKR," "Piggyback" and "Mask-RKR" showed the highest overall accuracy in all models. Among them, "Piggyback" showed the highest accuracy even though it has the largest number of parameters. Comparing Experiment 1 and Experiment 2, "Mask-RKR K+" was not more accurate than "RKR". Compared to "RKR", which generates task-specific parameters as they are, "Mask-RKR" applies a mask to the base parameters to generate task-specific parameters. Therefore, it is thought that this is because Mask-RKR may not be flexible enough for regions that are far from the base parameters. From this, we can say that Piggyback is the most effective in the continual learning setting for datasets in different domains unless the number of parameters needs to be minimized.

4.3. Ablation studies

In the ablation studies, we compared the accuracy and the number of parameters for different components of Mask-RKR. The experiments were conducted with a threshold value of τ =5e-3 for the hard binary threshold function of Mask-RKR and a rank of $K = 2$ for the low-rank approximation.

Table 5: Verification results of the usefulness of Piggyback

LM w/ PB	RM w/ PB	F w/ PB	Ave. Acc ResNet-18	Ave. Acc ViT	Ave. Acc Swin	Params[M] ResNet-18	Params[M] ViT	Params[M] Swin
×	×	×	0.794	0.843	0.858	11.74	89.88	1.43
✓	×	×	0.792	**0.848**	**0.859**	11.55	88.51	1.37
×	✓	×	0.792	0.842	0.850	11.51	88.51	1.35
✓	✓	×	0.780	0.844	0.846	11.33	87.07	1.28
×	×	✓	0.794	0.845	0.858	11.69	89.15	1.40
✓	×	✓	**0.795**	0.846	0.846	11.51	89.15	1.33
×	✓	✓	0.790	0.836	0.843	11.47	87.71	1.31
✓	✓	✓	0.781	0.840	0.841	11.28	86.26	1.24

Table 6: Verification results of where Piggyback is applied

Method	Ave. Acc ResNet-18	Ave. Acc ViT	Ave. Acc Swin	Params[M] ResNet-18	Params[M] ViT	Params[M] Swin
RG w/o PB	0.794	**0.845**	**0.858**	11.69	89.15	1.40
R w/ PB	**0.805**	**0.845**	0.847	14.41	110.05	1.55
LMRM w/ PB	0.781	0.840	0.841	11.28	86.26	1.24

First, we verified the usefulness of Piggyback by comparing the case where Piggyback is applied to each of RG and SFG of Mask-RKR with and without Piggyback. The experimental results are shown in Table 5. "LM w/ PB," "RM w/ PB" and "F w/ PB" indicate whether or not Piggyback is applied to the RG and SFG parameters LM, RM, and F, respectively. Here, the case where Piggyback is not applied to LM, RM, and F is the same as RKR. The experimental results show that regardless of where Piggyback is applied, the higher the parameter reduction effect, the lower the accuracy. In this experiment, we use the model with Piggyback applied to LM, RM and F, which can reduce the number of parameters the most, considering the application to Vision Transformer.

Next, we compared the "R w/ PB" when Piggyback is applied directly to the weight modification parameter R in the RG of Mask-RKR, the "LMRM w/ P" when Piggyback is applied to LM and RM, respectively, and the "RG w/o PB" when Piggyback is not applied to RG. In all the models, Piggyback is applied to SFG. The experimental results are shown in Table 6. The experimental results show that "R w/ PB" increases the number of parameters compared to "RG w/o PB" and "LMRM w/ PB". In addition, the accuracy of "R w/ PB" is better than that of "LMRM w/ PB", but is comparable to that of "RG w/o PB", and there is no improvement in accuracy due to the application of Piggyback. These results indicate that applying Piggyback to LM and RM is more effective in RG to reduce the number of parameters.

5. CONCLUSION

In this paper, we compared continual learning methods that can be applied to both CNN and Vision Transformer. We also proposed Mask-RKR, which combined RKR [6] and Piggyback [7]. From the experiments, we found that Mask-RKR can achieve higher accuracy than the original RKR and Piggyback while reducing the number of parameters. However, in the continuous learning setting for different domains, Piggyback shows the highest accuracy and is more effective when the number of parameters is not limited.

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

Acknowledgment: This work was supported by JSPS KAKENHI Grant Numbers, 21H05812 and 22H00548, 22K19808.

6. REFERENCES

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. of International Conference on Learning Representation*, 2021.

[2] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc.of IEEE Computer Vision and Pattern Recognition*, 2021.

[3] Anthony Robins, "Catastrophic forgetting, rehearsal and pseudorehearsal," *Connection Science*, vol. 7, no. 2, pp. 123–146, 1995.

[4] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, "Distilling the knowledge in a neural network," in *Proc. of Neural Information Processing Systems conference*, 2015.

[5] James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proc.of the National Academy of Sciences*, 2016.

[6] Pravendra Singh, Pratik Mazumder, Piyush Rai, and Vinay P. Namboodiri, "Rectification-based knowledge retention for continual learning," in *Proc.of IEEE Computer Vision and Pattern Recognition*, 2021.

[7] Arun Mallya and Svetlana Lazebnik, "Piggyback: Adding multiple tasks to a single, fixed network by learning to mask," in *Proc.of European Conference on Computer Vision*, 2018.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Proc. of Neural Information Processing Systems conference*, 2017.

[9] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord, "Dytox: Transformers for continual learning with dynamic token expansion," *arXiv:2111.11326*, 2021.

[10] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer G. Dy, and Tomas Pfister, "Learning to prompt for continual learning," *arXiv:2112.08654*, 2021.

[11] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *arXiv:2107.13586*, 2021.

[12] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi, "Learning multiple visual domains with residual adapters," *In Advances in Neural Information Processing Systems*, p. 506–516.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proc.of IEEE Computer Vision and Pattern Recognition*, 2016.