Continual Learning of Image Translation Networks Using Task-dependent Weight Selection Masks

Asato Matsumoto[1] and Keiji Yanai[1]

The University of Electro-Communications, Tokyo
{matsumo-a, yanai}@mm.cs.uec.ac.jp

Abstract. Continual learning is training of a single identical network with multiple tasks sequentially. In general, naive continual learning brings severe catastrophic forgetting. To prevent it, several methods of continual learning for Deep Convolutional Neural Networks (CNN) have been proposed so far, most of which aim at image classification tasks. In this paper, we explore continual learning for the task of image translation. We apply Piggyback [1], which is a method of continual learning using task-dependent masks to select model weights, to an Encoder-Decoder CNN so that it can perform different kinds of image translation tasks with only a single network. By the experiments on continual learning of semantic segmentation, image coloring, and neural style transfer, we show that the performance of the continuously trained network is comparable to the networks trained on each of the tasks individually.

Keywords: continual learning · lifelong learning · image translation · catastrophic forgetting

1 Introduction

Humans and animals can learn and fine-tune their knowledge continually throughout their lives. This ability is realized by the rich neurocognitive function of the brain. As a result, humans and animals can learn new knowledge through many experiences over a long period, and never forget old knowledge. In this way, learning that adapts to new knowledge while retaining previously learned knowledge is called continual learning. Since a CNN operated in the real world is given continual information and tasks sequentially, it is important to learn knowledge over a long period and repeat fine-tuning in such a situation. In addition, continual learning is considered to contribute to the achievement of general-purpose artificial intelligence, which needs to perform various tasks given in large quantities, because it does not forget tasks learned previously. Besides, since multiple tasks can be executed by one CNN, the size of the learned model can be reduced from a practical point of view, and it is thought that CNN applications also contribute to the implementation of smartphones and devices. Although continual learning is related to general-purpose artificial intelligence, continual learning in

CNNs is an unsolved problem because of the property of learning only in specific situations of machine learning. When humans do continual learning, they can learn new tasks without forgetting tasks they have learned in the past. On the other hand, the knowledge CNNs acquired depends on training datasets, and in order to adapt to changes in data distribution, it is necessary to re-train the parameters of CNN for the entire data set. As we learn about new tasks given over time, the accuracy of old tasks decreases. In this way, continual learning in CNNs brings catastrophic forgetting, which results in forgetting the learning results of old tasks while learning new tasks. Catastrophic forgetting causes CNN to suffer from performance degradation and that new knowledge overwrites old knowledge. For this reason, it is difficult for CNNs to perform continual learning like humans and animals. Until now, pseudo-rehearsal [2], EWC [3], network expansion [4], pruning [5], and selection of weights [1] have been proposed as methods to avoid fatal oblivion. However, most of these methods relate to image classification, object detection, and reinforcement learning, and continual studies on image translation tasks have hardly been conducted. Therefore, we propose a method of continual learning in the image translation tasks including semantic region segmentation, neural style transfer, and coloring using an Encoder-Decoder CNN.

This paper treats with continual learning for image translation tasks where both input and output are images. Specifically, as shown in Figure 1, applying "Piggyback" [1], which is a continual learning method using binary masks that select the weights of the trained Encoder-Decoder CNN model. In the original paper of "Piggyback", it was applied to only image classification tasks, and its effectiveness of the other task such as image translation tasks has not confirmed so far. Then, the purpose of this paper is to explore the effectiveness of "Piggyback"on image transformation tasks with a single Conv-Deconv CNN and small task-dependent binary masks.

To summarize it, the contributions of this work are follows: (1) We applied the "Piggyback" method to continual learning of multiple image translation tasks. (2) We confirmed that continual learning of a Conv-Deconv CNN was possible and its performance of each of the trained tasks was comparable to the performance of individually trained model. (3) We analyzed the trained binary masks and found that the distributions of masked-out weights for image translation tasks were totally different from ones for image classification tasks.

2 Related Work

A common way to learn new tasks with already learned CNN is fine-tuning. Fine-tuning is a method to re-train weights of pre-trained CNN for new tasks. Because of re-training, changing the value of CNN weights causes catastrophic forgetting that the accuracy of the old task decrease. To overcome it, many works have been proposed so far. Pseudo-rehearsal [2], optimization [3], network expansion [4], pruning [5], weight selection [1], etc. are representative methods. Here, we will particularly describe optimization [3] and weight selection [1] methods.

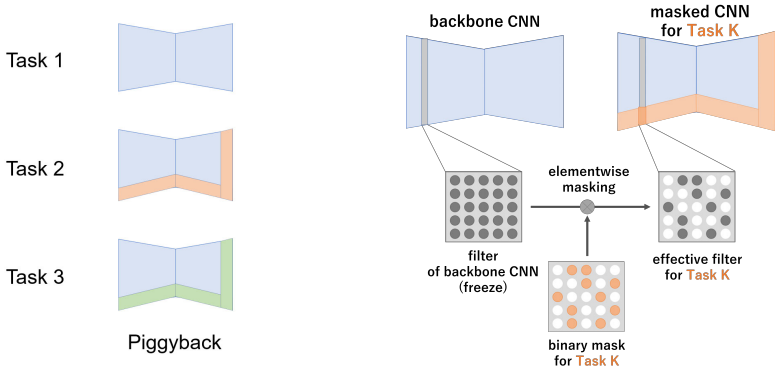

Fig. 1. Outline of the proposed method.

2.1 EWC

EWC [3], standing for Elastic Weight Consolidation, makes it suppress to change important weights for previous tasks. This method reduces the disruption of the learning result in the previous task. Thus the performance degradation of the previous task by learning a new task is expecting to be prevented. The importance of the weights is determined based on a Fisher information matrix of the trained CNN, and the learning rate of each parameter is adjusted in proportion to the importance of the weights. By adjusting learning rate on each weight adaptively, it becomes possible to learn new tasks while maintaining the important weights of the previously learned tasks. However, if the previous task and the new task are very different or if many tasks are added repeatedly, the performance of the previous task will be degraded, because the value of the weights with high importance are expected to be changed greatly even in EWC.

2.2 Piggyback

On the other hand, Piggyback [1] solved the performance degradation of the previous task which is the problem of EWC and succeeded in learning many tasks by only a single CNN with high accuracy. In Piggyback, first we train a general-purpose backbone CNN with a large data set such as ImageNet, and next train a binary mask that selects high importance weights for each of the tasks when learning additional new tasks. Since only the binary mask is learned, the weight value of the backbone CNN does not change. That is why the performance of the previous task is never degraded. While EWC trains less importance weights for additional tasks, Piggyback freezes the backbone CNN and select important weights from it for additional tasks. However, in the original paper, Piggyback was applied to only classification CNN. Then, in this paper, we explore the effectiveness of Piggyback for an Encoder-Decoder CNN.

3 Method

In this paper, continual learning of multiple different image translation tasks is performed with a single CNN. We applied "Piggyback" [1] to an Encoder-Decoder CNN. Piggyback is a method of fixing the parameters of a backbone CNN which is trained first, and learning a task-specific binary mask for each of the additional tasks. In the original paper of Piggyback, when adding a new task, we prepare a task-specific final output layer for each of the tasks in addition to the binary masks. Following this, in this paper, we prepare an Encoder-Decoder CNN, task-dependent binary masks and task-dependent final output layers for continual learning of image translation tasks. That is, "PiggyBack" shares all the backbone weights and needs to prepare binary masks which have 1 bit per backbone CNN parameter and final output layers independently for each of the additional tasks. The learning procedure of Piggyback is shown in Figure 2.

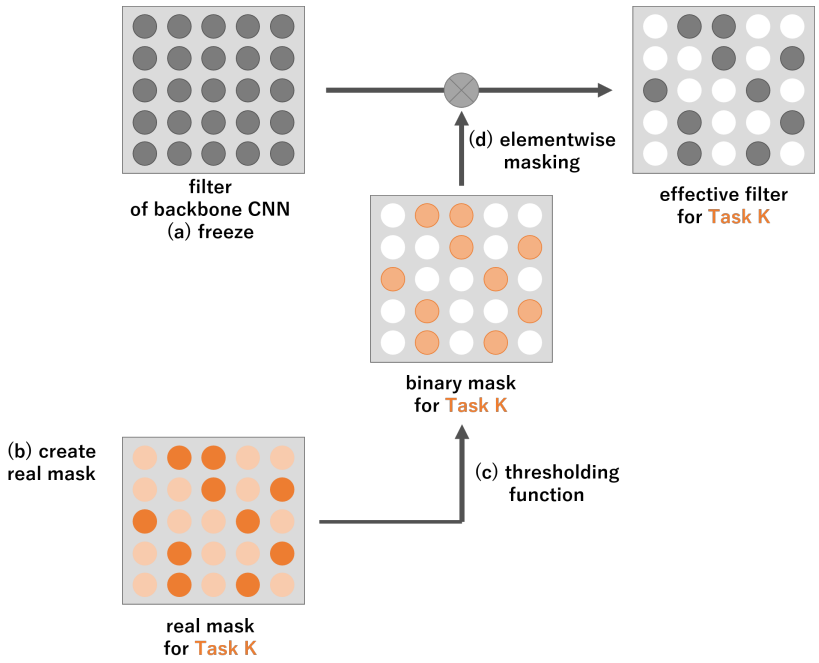

Fig. 2. Learning Masks in Piggyback

In this paper, we apply PiggyBack to image translation tasks. First we train a CNN for one of the image translation tasks to build a backbone network. The task for a backbone network should be trained with a large-scale dataset. The learning of the mask is performed by the following procedure. Note that real masks are the real weights the number of which are the same as one of the backbone CNN.

1. Train and fix the weight of backbone CNN for the first task ((a) in Figure 2)
2. Train real mask weights for the additional tasks ((b) in Figure 2)
3. Binarize real weights with a threshold and create binary masks ((c) in Figure 2)
4. Select weights with masking the weight of backbone CNN with the binary masks ((d) in Figure 2)
5. Calculate forward gradients and back propagation using (4)
6. Update of real mask weights.
7. Repeat (3) to (6)

In the experiment, the value of the real mask is initialized to 1e-2 for all parameters, as in the paper of Piggyback [1]. Also, the threshold used for creating the binary mask was 5e-3 in the same way as [1].

In this experiment, we expected that a highly functional backbone CNN was necessary to be trained with the largest datasets within all the tasks to be trained with a single Conv-Deconv network. Therefore, we utilized semantic segmentation with a large dataset as the first image translation task. In particular, the first image translation task was semantic segmentation with MS COCO where approximately 200,000 images were annotated with 80 objects. The second and subsequent tasks learned binary masks that selects valid weights for the task newly added from the backbone CNN. Furthermore, in this experiment, we have used U-Net [6] which has skip connections between the Encoder and the Decoder CNN and is used for various image translation tasks. Note that instead of Batch Normalization before the activation function ReLU other than the output layer, we used Instance Normalization, which has the effect of accelerating the convergence of learning.

4 Experiments

4.1 Evaluation on continual learning of image translation tasks

In the experiment, we performed continual learning of five different image translation tasks, and evaluated the performance of the proposed method and baselines. The tasks used in the experiments were semantic segmentation with two different dataset, gray image coloring, and neural style transfer with two different styles [7] as shown in Table 1. In training, Tasks 1, 2, 3, 4, 5, and 6 were sequentially trained with a single Conv-Deconv network. Exceptionally, for Task 5 with "fine-tune" which is one of the baselines, the model was trained after not Task 4 but Task 3.

Both MS COCO and Pascal VOC are image datasets containing general images such as people, vehicles and animals. MS COCO consists of about 330,000 images including 80 categories with pixel-wise segmentation masks, and Pascal VOC consists of about 10,000 images including 20 categories included in the eighty categories of MS COCO. Edges2handbags [8] is a dataset consisting of about 137,000 handbags images and binary edges extracted from the images. The reason for performing the same semantic segmentation in Task 1 and Task 2 is to confirm whether continual learning is possible for the same task with

Table 1. Five tasks for continual learning.

	task category	dataset
Task 1	semantic segmentation	MS COCO
Task 2	semantic segmentation	Pascal VOC 2012
Task 3	gray image coloring	MS COCO
Task 4	style transfer (Gogh)	MS COCO
Task 5	style transfer (Munk)	MS COCO
Task 6	edge image coloring	edges2handbags

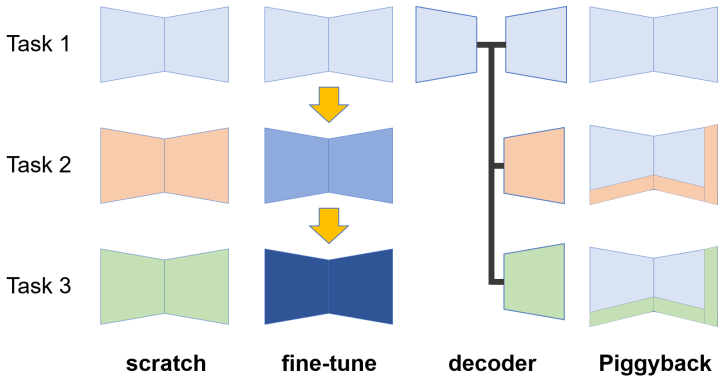

Fig. 3. The overview of three baselines and "Piggyback'.

different datasets. After Task 3, experiments were performed on the tasks that differ from semantic segmentation, and we verified whether continual learning across different tasks was possible. In the original "Piggyback" paper [1], only classification tasks are performed with different datasets. On the other hand, in this experiments, we perform continual learning of heterogeneous tasks with different loss functions and different datasets. This point is completely different from the original work.

In the framework of Piggyback, the first task is importance since the network trained in the first task is used as a backbone CNN. It should be comprehensive and trained with as large dataset as possible. This is why we used MSCOCO for training of Task 1.

In this experiment, three types of baselines were prepared for comparison. The outline of the three baselines and Piggyback is shown in Figure 3. Three types of the baselines are "scratch" in which we learn task-dependent models from scratch separately for each task, "fine-tune" in which we fine-tune a single identical model with five tasks continuously, and "decoder" in which we train both the encoder and decoder part when training of Task 1, fix the encoder part and train only task-dependent decoder parts for all the other tasks independently. "Piggyback" is the method we explore its effectiveness for image translation tasks in this paper.

Table 2. Results of continual learning (See the Table 1 for the contents of each task)

	scratch	**fine-tune**	**decoder**	**Piggyback**
Task 1 (mIoU(%))		**21.47**		
Task 2 (mIoU(%))	58.59	**64.87**	61.63	61.45
Task 3 (MSE)	244.000	**237.92**	241.66	242.49
(SSIM)	0.9138	**0.9148**	0.9121	0.9058
Task 4 (SSIM)	**0.3678**	0.3555	0.3595	0.3501
(total loss)	413,833	**405,893**	473,723	528,587
Task 5 (loss)	**447,480**	490,490	544,348	521,476
Task 6 (MSE)	211.96	**207.76**	237.53	232.02
Task 1 after Task 2 (mIoU)	-	0.70	21.47	21.47
Task 2 after Task 3 (mIoU)	-	1.87	61.63	61.45
Task 3 after Task 4 (MSE)	-	870.18	241.66	242.49
(SSIM)	-	0.5321	0.9121	0.9058
Model Size (MB)	338.4 (56.4*6)	338.4 (56.4*6)	158.9 (56.4+20.5*5)	**65.4** (56.4+1.8*5)

For training for each of the tasks, the loss functions are as follows: In Task 1 and Task 2, we use Cross-Entropy Loss, in Task 3 we use L2 loss, Tasks 4 and Task 5 we used Content Loss and Style Loss used in the fast style transfer proposed by Johnson et al. [7], and in Task 6 we used GAN Loss and L1 used in the pix2pix by Isola et al. [8].

The input image are RGB images in Task 1, Task 2, Task 4, and Task 5, gray images are used for Task 3, and binary edges in Task 6. The outputs were as follows: In Task 1 and Task 2 they are segmentation masks of 81 channels and 21 channels, respectively. In Task 3, they are the CbCr components which need to be combined with input grayscale images to obtain RGB images. In Task 4, Task 5, and Task 6, they were RGB images.

For evaluation, the test dataset is used, and the performance was evaluated as follows: For Task 1 and Task 2 we use mean intersections over union (mIoU), for Task 3 we use mean square error (MSE) and structural similarity (SSIM), for Task 4 we use SSIM and total loss values, for Task 5 we use only total loss values, and for Task 6 we use MSE. As described in Gatys's style translation paper [9] we use the total loss that combines Content Loss and Style Loss for Task 4 and 5.

Table 2 shows the evaluation values of each task, the evaluation values of the previous task after training of the next task represented as "Task n after Task $(n+1)$", and the total size of trained models over five tasks. Note that the most important part of this table is the part of "Task n after Task $(n+1)$", since the values of this part are expected to be degraded greatly if catastrophic forgetting happens.

This table indicates continual learning by "decoder" and "Piggyback" never brought catastrophic forgetting, while it happens and the performance was degraded greatly in case of "fine-tuning". Regarding the size of the total models,

the "Piggyback" model is about half of the "decoder" model. "Decoder" has independent decoder parts and shares only the encoder part across tasks, while "Piggyback" shares the backbone model which is trained with the first task and has independent binary masks for four other tasks than the first task. In general, the size of one binary masks is 1/32 of the size of the backbone model, since each element of the binary masks is represented in one bit and each element of the backbone model is represented in a 32-bit floating value. From this results, we can confirm the effectiveness of "Piggyback" for image translation tasks.

Some examples of generated images of Task 2, 3, 4, 5, and 6 are shown in Figure 4, Figure 5, Figure 6, Figure 7, and Figure 8, respectively. Compared between the results by "Piggyback" and the results by the other methods, the quality of output images are almost comparable and not degraded at all.

4.2 Analysis on trained binary masks

Here, we analyze the binary masks trained by Piggyback in the same way as Mallya et al. [1]. We examined the ratio of 0 in the trained binary masks. The ratio of 0 in the binary mask represents how many parameters are required to changed in the backbone CNN for performing each of the tasks or how effective the backbone CNN was initialized in the MS COCO semantic segmentation task for each task. The ratio of 0 in the binary mask learned for each layer of U-Net is shown from Figure 9 to Figure 13 for Task 2, 3, 4, 5, and 6, respectively. In addition, for comparison on the ratio of 0, Figure 14 shows the ratio of 0 in the binary mask for the Wiki Art classification task used in Mallya et al. [1]. The horizontal axis of the graph of Figure 9 to Figure 13 represents each layer of U-Net, and "conv" and "up-conv" in the graph correspond to "conv" and "up-conv" in the U-Net

The ratio of 0 in the binary masks of classification task for the VGG16 network tended to be low in the low layer and to increase with depth of the layer gradually as shown in Figure 14. From this, for continual learning of classification tasks with Piggyback, it is considered that the rate of re-using backbone CNN is high in the lower layer because the lower layer is a part to extract low-level local features. Regarding the uppper layers, the ratio of 0 was higher than one of the lower layers, since task-dependent semantics information are extracted.

On the other hand, in the case of image translation with U-Net, the abovementioned tendency is only slightly seen in the Encoder part of Task 2. Conversely, the common tendency in classification tasks are not seen in the Encoder of other tasks and in the Decoder of all tasks. In addition, the ratio of 0 in both Encoders and Decoders of all tasks was about 40% to 60%. Moreover, the ratio of 0 to the first layer of the U-Net in Task 2 is about 30 points which is much higher than the first layer of VGG16, and the weight re-use ratio of the backbone CNN becomes lower.

Furthermore, in the tasks after Task 3, other than semantic segmentation, the ratio of 0 in the lower layer also increased. The reason why the ratio of 0 is higher in all layers of U-Net with image translation compared to VGG16 with classification is expected to be attributed to the following three reasons.

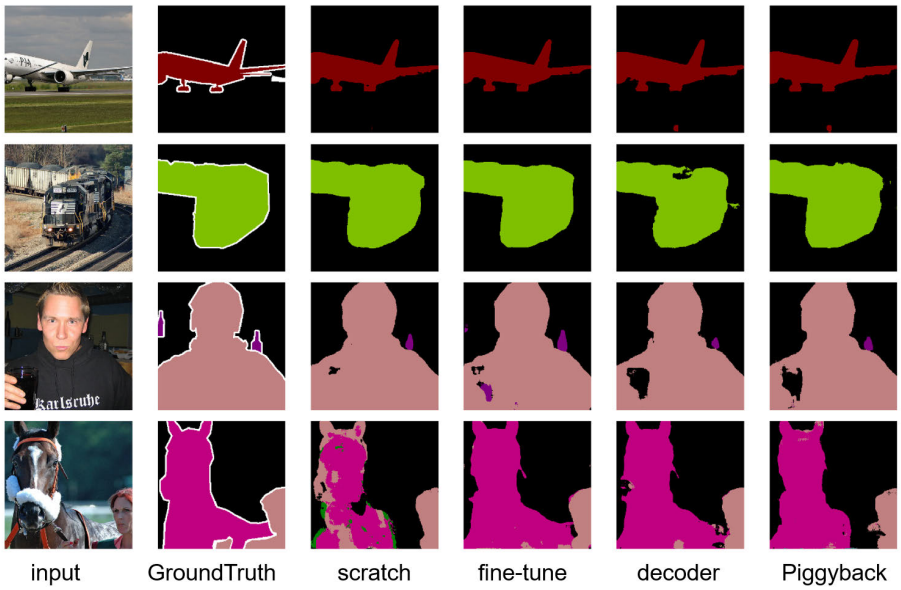

Fig. 4. Results of Task 2 (semantic segmentation for Pascal VOC)

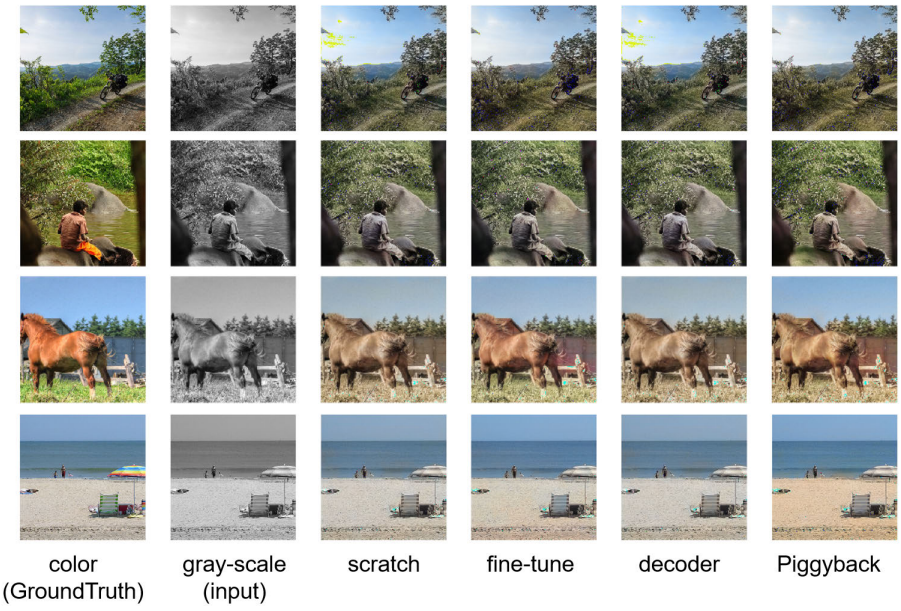

Fig. 5. Results of Task 3 (gray image coloring)

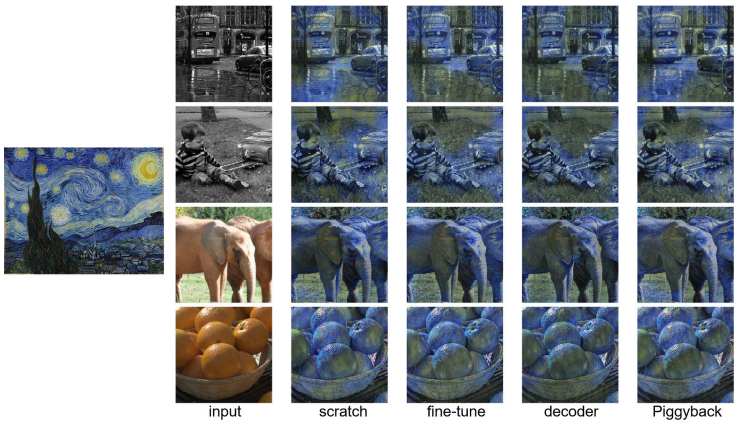

Fig. 6. Results of Task 4 (style translation with "Gogh" style)

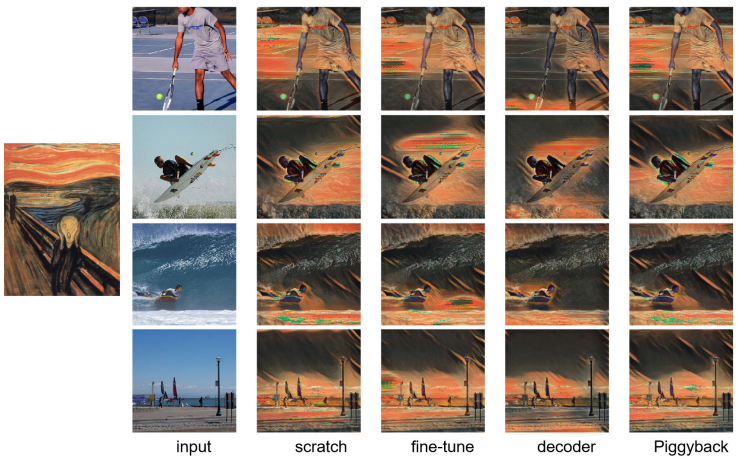

Fig. 7. Results of Task 5 (style translation with "Munk" style)

Fig. 8. Results of Task 6 (edge image coloring)

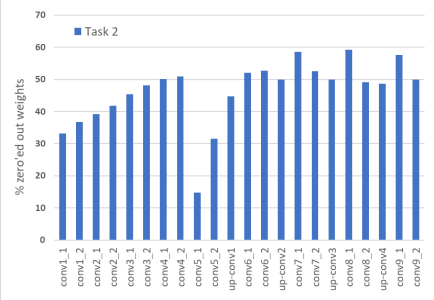

Fig. 9. The ratio of 0 in the trained binary masks for Task 2 (Pascal VOC segmentation)

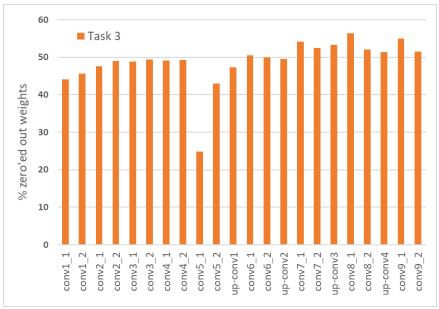

Fig. 10. The ratio of 0 in the trained binary masks for Task 3 (gray image coloring)

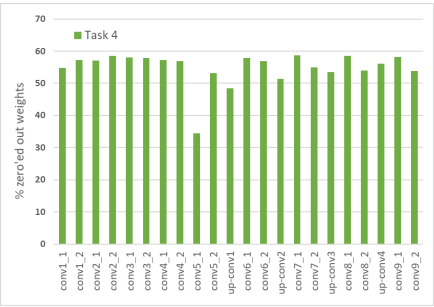

Fig. 11. The ratio of 0 in the trained binary masks for Task 4 (style translation with "Gogh")

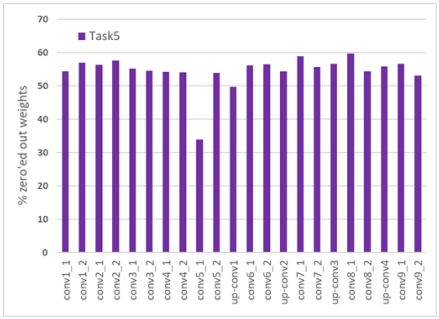

Fig. 12. The ratio of 0 in the trained binary masks for Task 5 (style translation with "Munck" style)

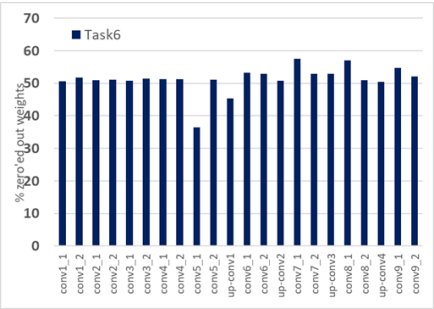

Fig. 13. The ratio of 0 in the trained binary masks for Task 6 (edge image coloring)

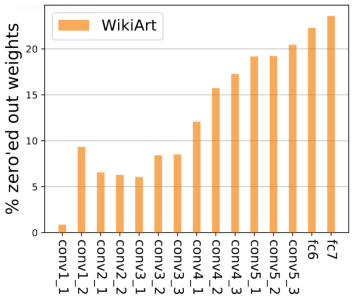

Fig. 14. The ratio of 0 in the trained binary masks for image classification [1].

The first reason is that the number of U-Net layers used in the experiment is larger than VGG16. By increasing the number of parameters in the CNN, it was considered that the number of important parameters possessed by one layer decreased, and the types of features acquired by one layer decreased. The second reason is that the task of learning the backbone CNN is different from the newly learned task. In the experiments by Mallya et al. [1], the dataset was changed for each task and continual learning was performed only with the classification task. On the other hand, in the experiments, after the backbone CNN was learned by the semantic segmentation, continual learning was performed by the tasks different from backbone CNN such as gray image coloring and style translation. It is considered that the important weights are changed by tasks or a different transformation from the original layer is realized by setting 0 to the mask. The third reason is that the parameter that is important in CNN may be about 50% of the whole layer. Even if 50% of the pre-trained CNN parameters are pruned in advance by Packnet [5] of Mallya et al., performance degradation of less than 1% is achieved from the accuracy of CNN before pruning. In addition, one interesting findings in the experiments is that the ratio of 0 for conv5_1 is low for all tasks. From this, it is considered that there may be a common translation even for different tasks.

Next, we analyze the similarity between the masks. Since all masks are binary, we calculated the similarity by taking XOR between the masks. The similarity matrix between the binary masks is shown in Table 3. From this table, all the similarity values except for Task 1 and Task 4,5 are around 0.5. The similarity between Task 1 and Task 4 and between Task 1 and Task 5 both of which are the pair of semantic segmentation and style transfer are relative low the value of which are less than 0.5, while the similarity between Task 1 and Task 2 both of which are the semantic segmentation, and Task 4 and Task 5 both of which are the style translation are relatively high. From this, it is considered that important weights are different in each task, and common weights are used between similar tasks and different weights are used between different task types.

Table 3. Similarity matrix of binary masks between the tasks.

	Task 1	Task 2	Task 3	Task 4	Task 5
Task 2	0.5075	-	-	-	-
Task 3	0.5042	0.5054	-	-	-
Task 4	0.4326	0.5034	0.5020	-	-
Task 5	0.4529	0.5029	0.5025	0.5210	-
Task 6	0.4847	0.5063	0.5026	0.5093	0.5077

5 Conclusions

In this paper, we have explored continual learning of different image translation tasks. From the experimental results, it was found that applying Piggyback to the Encoder-Decoder CNN achieves the same performance as the baseline with minimum overhead in continual learning of semantic segmentation, gray image coloring and neural style transfer.

The experiments were conducted with only three kinds of the tasks, segmentation, coloring and style transfer. Therefore, we plan to conduct additional experiments with other kinds of image translation tasks used in this paper to confirm the versatility of "Piggyback". Hopefully we like to set up various kinds of image translation tasks like Visual Domain Decathlon in which the ten types of classification tasks are trained continuously [1] or the experiments conducted by Kokkinos at Ubernet [10].

Acknowledgements: This work was supported by JSPS KAKENHI Grant Number 15H05915, 17H01745, 17H06100 and 19H04929.

References

1. A. Mallya, S. Lazebnik, and D. Davis. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proc.of European Conference on Computer Vision (ECCV)*, pp. 67–82, 2018.
2. A. Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. In *Jour.of Connection Science*, Vol. 7, pp. 123–146. Citeseer, 1995.
3. J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. In *Proc.of the National Academy of Sciences (PNAS)*, Vol. abs/1612.00796, 2016.
4. A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. In *arXiv preprint arXiv:1606.04671*, 2016.
5. A. Mallya and S. Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proc.of IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 7765–7773, 2018.
6. O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proc.of International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pp. 234–241. Springer, 2015.
7. J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Proc.of European Conference on Computer Vision (ECCV)*, 2016.
8. P Isola, J Zhu, T. Zhou, and A. Efros. Image-to-image translation with conditional adversarial networks. In *Proc.of IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017.

[1] https://www.robots.ox.ac.uk/∼vgg/decathlon/

9. L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proc.of IEEE Computer Vision and Pattern Recognition (CVPR)*, June 2016.
10. I. Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *Proc.of IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 6129–6138, 2017.