# Continual Learning for an Encoder-Decoder CNN Using "Piggyback"

**Asato Matsumoto** **Keiji Yanai**
Department of Informatics,
The University of Electro-Communications, Tokyo
1-5-1 Chofugaoka, Chofu-shi, Tokyo 182-8585 JAPAN
{matsumo-a,yanai}@mm.inf.uec.ac.jp

## Abstract

Although continual learning has been paid much attention recently, most of the works focused on classification task. In this paper, we explore continual learning on an encoder-decoder CNN which has ability for various kinds of image-to-image transformation tasks such as semantic segmentation, colorization, style transfer and image domain transfer. To realize continual learning on an encoder-decoder CNN, we adopt "piggyback" which was proposed recently as a very efficient continual learning method. In "piggyback", per-task binary masks applied to the trained base network are learned for multiple tasks. By experiments, we show "piggyback" enabled continual learning for three heterogeneous tasks including semantic segmentation and colorization, which achieved comparable performance to independent models.

## 1 Introduction

Continual learning is learning of a single model continously with different tasks, the ability of which a human has but a computer does not have. This ability is really needed to realize artificial generic intelligence which is human-like AI. In addition, from practical point of view, saving the size of a trained model is crucial for implementation of AI applications on smartphones and IoT devices. In continual learning on a computer, "catastrophic forgetting" is always problematic. To overcome it, many works have been proposed so far. Rehearsal [1], distillation [2], optimization [3], network extension [4], and weight selection [5] are representative methods.

So far, most of the works on continual learning have focused on classification task, deep reinforcement learning [3] and image generation [6]. Then, in this paper, we explore continual learning on an encoder-decoder CNN which has ability for various kinds of image-to-image transformation such as semantic segmentation, colorization, depth estimation, style transfer and image domain transfer.

In case of simultaneous training, a multi-task encoder-decoder CNN consisting of a shared encoder and task-specific decoder branches has been shown to be possible. UberNet [7] is a multi-task convolutional encoder-decoder network which can learn multiple image-to-image tasks such as boundary detection, saliency estimation, and semantic segmentation. However, in UberNet, only an encoder part is shared among the different tasks, and each task has an independent decoder branch. This means the UberNet paper did not answer a question whether decoder parts can be shared as well in case of training of multiple tasks. In addition, in UberNet, they assumed simultaneous training of multiple tasks, and did not assume continual learning.

"One model to learn them all" [8] is also a universal encoder-decoder model which can learn not only image classification tasks but also natural language translation, and image captioning tasks. However, only intermediate parts were shared, and it assumed simultaneous training as well.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

Continual learning on an encoder-decoder CNN has not been explored yet. If it is possible that one model can carry out image-to-image translation in various kinds of the ways, it would be very helpful to implement mobile apps for image translation.

In our work, to realize continual learning on an encoder-decoder CNN, we use "Piggyback" [5]. In the "Piggyback" method, an backbone network is pre-trained with a relatively large-scale dataset for the first task, and binary masks that takes value in $\{0,1\}$ are learned for each of the tasks after the first task with the weights of the backbone network being kept fixed. In the original paper of "Piggyback" [5], the authors confirmed that it could learn multiple image classification tasks well by using the ImageNet pre-trained network as a backbone network. However, they did not examine its applicability to the other types of networks than image classification networks which contain only encoder parts.

Then, in this paper, we examine if an encoder-decoder CNN can be trained for multiple tasks with "Piggyback" in the setting of continual learning with almost all the layers shared among all the tasks. Especially we examine the case that the target multiple tasks contain heterogeneous image-to-image translation tasks such as semantic segmentation and colorization. By the experiments, we prove that "Piggyback" can train heterogeneous image-to-image tasks with per-task binary masks and one base model where only the last layer branches for the output of each task.

## 2 Approach

As a method for continual learning, we use Piggyback [5], in which the initial backbone network is fixed and a task-specific binary mask is learned per task. The key idea is to learn to selectively mask the fixed weight of the backbone network to adopt new tasks. In the paper, in addition to new masks, per-task final output layers were prepared. We follow this, and we also prepare a new mask and a per-task final deconvolutional layer for continual learning of an encoder-decoder CNN regarding each image-to-image task. Therefore, binary weights the size of which is the number of the trained weights of the backbone network and an independent final deconvolutional layer are needed to be added for incremental training of a new task.

In the paper, the authors concluded that a backbone network was needed to be well initialized for good performance. Therefore, we set semantic segmentation trained with the MSCOCO dataset [9] where about two hundred thousand images are annotated with 80 kinds of objects in the pixel level as the first image-to-image task. For the second task and more, we train binary masks which select the activating weights for newly added tasks.

## 3 Experiments

In the experiments, we utilize three tasks: MSCOCO semantic segmentation [9], PASCAL VOC semantic segmentation and colorization which converts gray-scale images into RGB color images. For training of the third task, we used pairs of original images and gray-scaled ones in the MSCOCO imaged dataset. Note that training of Task 1, 2 and 3 were carried out in that order continuously.

The first and second task are both semantic segmentation tasks but the datasets and the target categories are different. The third one is completely a different task from semantic segmentation, which can be regarded as a heterogeneous task to the first and second task. Therefore, the biggest objective of this work is examining the applicability of the backbone network trained with the MSCOCO semantic segmentation dataset [9] to the third task with binary masks for "piggybacking".

For comparison, we prepared three baselines: the model trained from scratch independently, the model fine-tuned from the model of the previous task, the model consisting of encoder pre-trained with the first task and per-task decoder branches. Note that in the third baseline, only decoder parts are trained for the second and the third tasks. We represent each of the baselines as "scratch", "fine-tune" and "decoder", respectively. Figure 1 shows the overview of three baselines and "Piggyback".

As an encoder-decoder network, we used a U-Net [10] which has skip connections between each of the corresponding layers of the encoder and the decoder. Instead of batch normalization, we used instance normalization after all the activation functions, ReLU, except the final layer. As loss function for training, pixel-wise softmax loss was used for Task 1 and Task 2, while L2 loss was used for Task 3. An input for Task 1 and 2 is a RGB color image, while an input for Task 3 is a

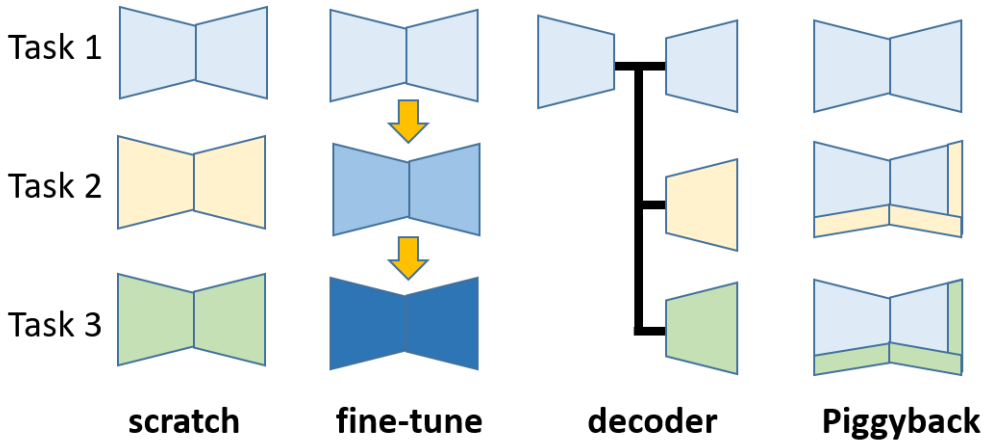


Figure 1: Overview of three baselines and "Piggyback".

gray-scale image. An output for Task 1 and Task 2 is a 81-channel and 21-channel segmentation map including a background map where the size of the feature maps is the same as the input image, respectively, while an output for Task 3 is a 2-channel feature maps consisting of Cb and Cr elements in the YCbCr color space. To obtain a color image, the input gray-scaled image corresponding to a Y channel feature map and the estimated CbCr channels are integrated and converted into a RGB color image. For evaluation, we used test sets of each dataset, and as metric, we used mean intersection over union (mIoU) for Task 1 and 2 and mean square loss (MSE) for Task 3.

Table 1 shows the results including the evaluations of the task after training the next and later tasks. All the baselines and "Piggyback" were able to learn their models for all the tasks successfully. Some example results of Task 2 and Task 3 by four methods are shown in Figure 2 and Figure 3.

Since the identical network was used for training of the first task, Task 1, the same mIoU, 21.47, was achieved for all the models. For Task 2, "fine-tune" achieved the best results, since all the weights were re-trained. On the other hand, for Task 2, only the encoder part was trained for "decoder" and binary masks were trained for "Piggyback". Instead, the performance of Task 1 after training of Task 2 remained unchanged for "decoder" and "Piggyback", while "catastrophic forgetting" happened for "fine-tune" and the mIoU dropped down greatly from 21.47 to 0.70.

Regarding Task 3 which is colorization of gray-scale images and can be regarded as a heterogeneous task to semantic segmentation, "decoder" and "Piggyback" achieved almost comparable performance to "fine-tune" with a small drop. Surprisingly, both still outperformed "scratch" regarding the MSE. In the same way as after training of Task 2, performance of "fine-tune" on Task 1 and 2 dropped greatly, which meant that fine-tuning of a normal model made a trained model forget the previous tasks.

Table 1: The experimental results of continual learning of three tasks: Task 1 (COCO segmentation), Task 2 (PASCAL VOC segmentation) and Task 3 (colorization).

| | scratch | fine-tune | decoder | Piggyback |
|---|---|---|---|---|
| Task 1 (mIoU(%)) | | **21.47** | | |
| Task 2 (mIoU(%)) | 58.59 | **64.87** | 61.63 | 61.45 |
| Task 3 (MSE) | 244.00 | **237.92** | 241.66 | 242.49 |
| Task 1 after Task2 (mIoU(%)) | – | 0.70 | **21.47**(*) | **21.47**(*) |
| Task 1 after Task3 (mIoU(%)) | – | 0.41 | **21.47**(*) | **21.47**(*) |
| Task 2 after Task3 (mIoU(%)) | – | 1.87 | **61.63**(*) | **61.45**(*) |
| Model Size (MB) | 169.2 $(56.4 \times 3)$ | 169.2 $(56.4 \times 3)$ | 97.4 $(56.4 + 20.5 \times 2)$ | **60.0** $(56.4 + 1.8 \times 2)$ |

From these results, both "fine-tune" and "Piggyback" have ability to overcome "catastrophic forgetting" in case of continual training of an encoder-decoder CNN. However, the model size is so different, since "fine-tune" has per-task decoder branches each of which consume 20.5MB while "Piggyback" needs only 1.8MB for each combination of a per-task binary mask and a task-specific last deconvolutaional layer. As results, "Piggyback" needs 60.0MB, and "fine-tune" needs 97.4MB, although the performance and ability for overcoming forgetting are comparable.

## 4 Conclusions

In this paper, we made experiments on continual learning of an encoder-decoder CNN with "Piggyback" and three baselines. By the experiments, the "Piggyback" method can successfully train an encoder-decoder CNN under the setting of continual learning, and achieved the best performance regarding both the model size and the ability of overcoming "catastrophic forgetting".

For future work, we plan to made experiments on more heterogeneous tasks such as fast neural style transfer and unsupervised domain transfer such as "horse to zebra" and "apple to orange".

## References


[1] A. V. Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

[2] K. Shmelkov, C. Schmid, and K. Alahari. Incremental learning of object detectors without catastrophic forgetting. In *Proc. of IEEE International Conference on Computer Vision*, 2017.

[3] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, et al. Overcoming catastrophic forgetting in neural networks. *Proc. of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[4] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv:1606.04671*, 2016.

[5] A. Mallya, D. Davis, and S. Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proc. of European Conference on Computer Vision*, 2018.

[6] A. Seff, A. Beatson, D. Suo, and H. Liu. Continual learning in generative adversarial nets. *arXiv:1705.08395*, 2017.

[7] 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*, 2017.

[8] L. Kaiser, A. N. Gomez, N. Shazeer, A. Vaswani, N. Parmar, L. Jones, and J. Uszkoreit. One model to learn them all. *arXiv:1706.05137*, 2017.

[9] T. Y. Lin, M. Maire, S. J. Belongie, L. d. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, P Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context, 2014.

[10] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Proc. of the Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241, 2015.

## Appendix

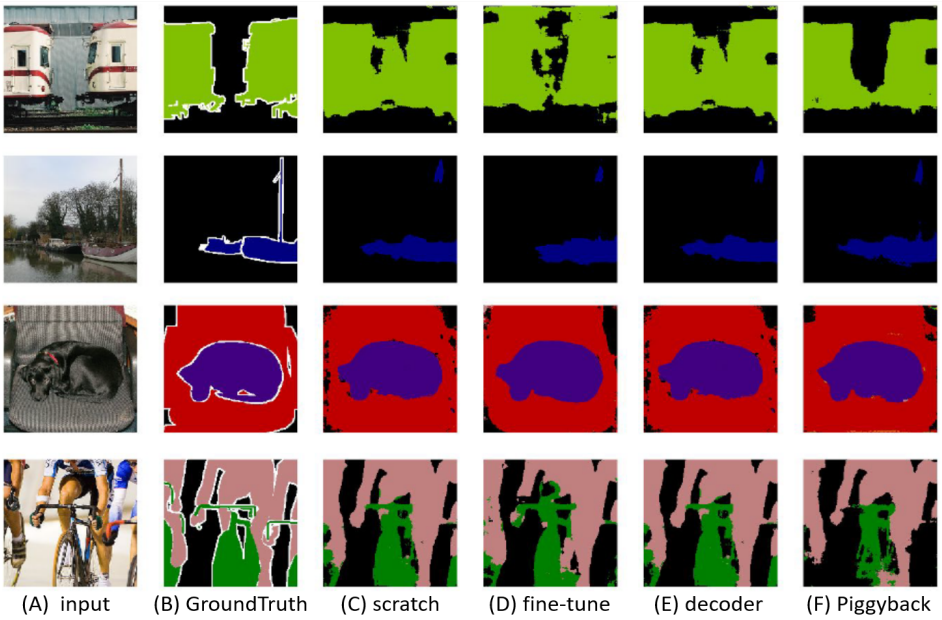


Figure 2: Results of Task 2 (Pascal VOC Segmentation).

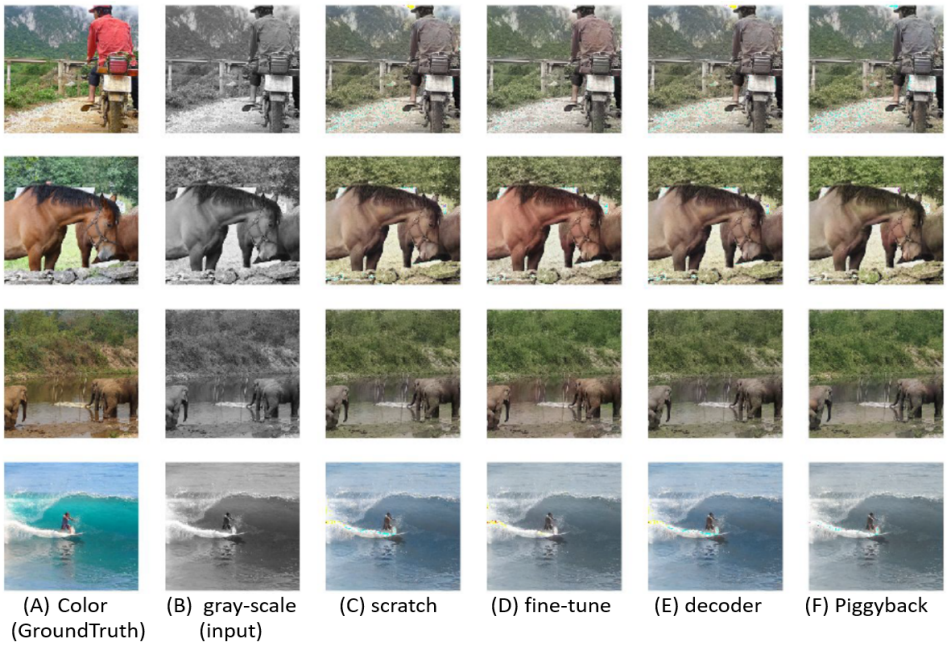


Figure 3: Results of Task 3 (Colorization).