Continual Learning on CLVision-Challenge

Haoran Zhu, Hua Sun
Department of Electrical and Computer Engineering
New York University
`{hz1922, hs4062}@nyu.edu`

Abstract

Continual learning is a crucial ability for artificial intelligence. It aims at allowing an agent to learn new tasks without forgetting the knowledge of previous tasks. In real life applications(e.g., autonomous driving cars, robotic applications), due to limited computational and storage resources, it is almost impossible to fully retrain models every time new tasks and new data become available. In this work, we take part in CVPR 2020 Workshop on Continual Learning in Computer Vision Challenge. We reproduce several important continual learning algorithms and apply them on a real life dataset, CORe50, which is a new dataset and benchmark specially designed for continuous object recognition scenarios.

1 Introduction

In today's Deep Learning scheme, the presumption of the training is that all instances of the classes are available, the model will iterate the dataset per epoch to learn the knowledge. This means if there are new instances added to the dataset, we have to retrain the whole model on the whole dataset plus the new instances. This has caused trouble in some scenarios. Specifically, in image classification problems, the pretrained model often encounters new objects or the dataset can be expanded. However, the existence of *Catastrophic Forgetting*(5), i.e. the newly learned parameters will shift the old parameters and weaken its performance on old task, has brought challenge to this problem. This phenomenon can greatly degrade the performance of the model or even totally rewrite the model's parameters, causing the old knowledge being totally forgot. In before, in face of such case, one have to choose either retrain the model on entire dataset or just tolerate such degrading issues. To overcome the dilemma, the concept of continual learning(Lifelong Learning(7)) has emerged. In continual learning, it is required that the model must have not only the ability to acquire new knowledge, but also prevent the novel input to overwhelm the original data.

Continual learning is close to the concept of online learning. But in online learning setting, there will be some dependency on the previous data, while in continual learning, we do not want to access the original dataset because it is costly and time consuming. The focus of continual learning is not only on maintaining the accuracy, but also on training efficiently. Currently, there are three main approaches to apply continual learning,

1. **Retraining** Totally retraining the old parameters θ_o on the new task to obtain the new model θ_n, and then apply regularization to prevent the degrading problem on old task. The typical work in this approach are *Learning without Forgetting*(LwF)(2) and Elastic Weight Consolidation(EWC)(1).
2. **Expansion** Non-Retraining but instead expanding the network. Whenever encounter the new task, just freeze the old weight and expand the network. One of the typical work is *Progressive Network*(6). Another variation of this work is masking, i.e. they focus not only on the weigth itsely, but also considered masking some of the unimportant weights for the new task t. Packenet(4) and Piggyback(3) adopt this approach.

3. **Partial Retraining with Expansion** Another approach is selectively retraining the old network, expanding the capacity when necessary and train dynamically on the optimal solution. One of the outstanding work in this approach is *Dynamically Expandable Networks*(DEN)(8).

In this project, we focus on the *CLVision Challenge*[1], i.e. Continual Learning in Computer Vision, a problem set published by CVPR 2020 Workshop to solve the image classification problem in continual learning. We have reproduced and refined several famous approaches, including *EWC*, *Piggyback* and *DEN*(?), and tried out them on the CLVision dataset, then we evaluate the performance and analyze the potential improvements that can be done.

2 Related Work

Continual Learning, also known as Lifelong Learning, is defined to be an algorithm which is capable of extracting knowledge from a continuous stream of data without forgetting the old knowledge, i.e. catastrophic forgetting(**?**).

3 Dataset and Experiment Design

3.1 Dataset and Tasks

CORe50 is a new dataset specially designed for *(C)ontinual (O)bject (Re)cognition*. Unlike *permuted MNIST* where new tasks to learn are obtained by simply scrambling the pixel positions, *CORe50* is much more complex and a real life dataset. Datasets such as *ImageNet* and *Pascal VOC* provide a good playground for image classification and detection, but they have been designed with static evaluation and lack of multiple views of the same objects taken into different sessions, *CORe50* solves the above problems and meets the requirement for continuous learning scenarios on computer vision.

It consists 50 domestic objects belonging to 10 categories. The dataset is separated into 11 distinct sessions (8 indoors and 3 outdoors) with different background and lightning. For each session and for each object, a 15 seconds video (at 20 fps) has been recorded with a Kinect 2.0 sensor delivering 300 RGB-D frames.

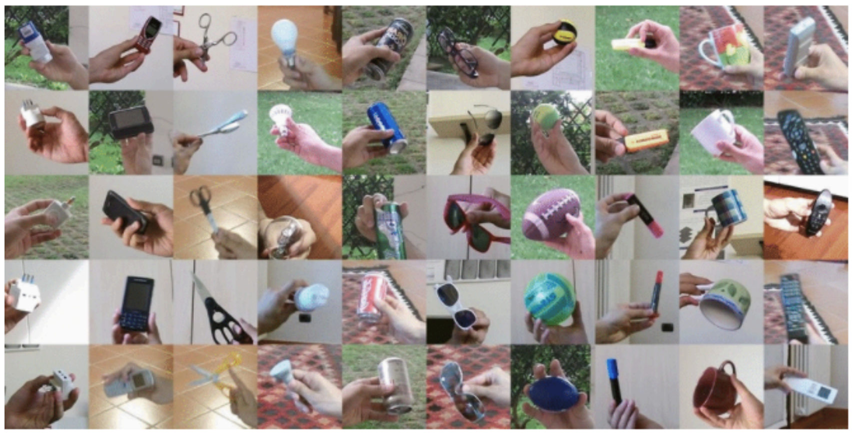

Figure 1: Example images of the 50 objects in CORe50. Each column denotes one of the 10 categories.

There are three tasks in CVPR 2020 Workshop *CLVision Challenge*:

- **New Classes(NC)**: New training patterns belonging to different classes become available in subsequent batches. In this case the model should be able to deal with the new classes without losing accuracy on the previous ones.

[1]https://sites.google.com/view/clvision2020/challenge?authuser=0

- **New Instances(NI)**: New training patterns of the same classes become available in subsequent batches with new poses and conditions (illumination, background, occlusion, etc.). A good model is expected to incrementally consolidate its knowledge about the known classes without compromising what it has learned before.

- **New Instances and Classes(NIC)**: New training patterns belonging both to known and new classes become available in subsequent training batches. A good model is expected to consolidate its knowledge about the known classes and to learn the new ones.

3.2 Piggyback

3.2.1 Theorem

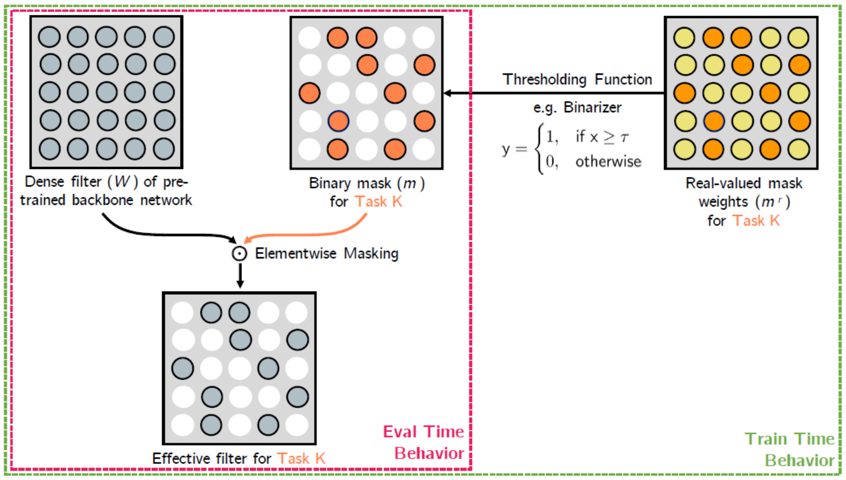
Dense filter (W) of pre-trained backbone network
Binary mask (m) for Task K
Thresholding Function
e.g. Binarizer
$y = \begin{cases} 1, & \text{if } x \geq \tau \\ 0, & \text{otherwise} \end{cases}$
Real-valued mask weights (m^r) for Task K
$\odot$ Elementwise Masking
Effective filter for Task K
Eval Time Behavior
Train Time Behavior

Figure 2: Piggyback Overview

Piggyback adopts masking weight for new task to overcome catastrophic forgetting. The core idea is to selectively mask the fixed weights of a base network, and then use this mask to predict the new task. That is, given model $\mathbf{W}$, the optimizer will genrate and optimize a binary mask weight $\mathbf{W}^{\star}$. The gradient that backpropagates to the layer will not affect $\mathbf{W}$ but $\mathbf{W}^{\star}$. In this way, for a given task, we can obtain a filter vecotor that is consist of 0/1. For example, a weight dense vector [0.1, 0.2, 0.3, 0.4] can be filtered to [0, 0.2, 0.3, 0] after the binary masking. In short, we does not learn what are the ‘right‘ parameters but learn what is not the ‘right‘ parameters. Therefore, the choice of backbone network is crucial to the performance of piggyback because if the original weights are malfunctioning, pure masking cannot greatly improve the accuracy.

The structure of *Piggyback* is shown in figure 2. To explain the procedure for training, we consider an end to end fully-connected layer case. Denote $\mathbf{x} = [x_1, x_2, ..., x_m]$ as input, and $\mathbf{y} = [y_1, y_2, ..., y_n]$ as output. Therefore the weight matrix is $\mathbf{W}^{n\times m}$. Without loss of genrality, we can simply assume $\mathbf{y} = \mathbf{W}\mathbf{x}$ by ignoring the bias term. Suppose the loss function is L, the backpropagation equation for $\mathbf{W}$ is,

$$\begin{aligned} \delta w_{ji} &= \frac{\partial L}{\partial w_{ji}} = (\frac{\partial L}{\partial y_j}) \cdot (\frac{\partial y_j}{\partial w_{ji}}) \\ &= \delta y_j \cdot x_j \\ \therefore \delta \mathbf{W} &= \delta \mathbf{y} \cdot \mathbf{x}^{\mathbf{T}} \end{aligned} \tag{1}$$

In *Piggyback*, the author has introduced a real value matrix $\mathbf{M_r}^{n\times m}$ and a manually set threshold τ. We denote the mask matrices as $\mathbf{M} = [m]_{ji}$, and we can obtain m_{ji} by,

$$m_{ji} = \begin{cases} 1, & \text{if } m^r_{ji} > \tau \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Figure 3: Sample figure caption.

Then, for $y_j \in \mathbf{y}$, it gives $y_j = \sum_{i=1}^{m} w_{ji} \cdot m_{ji} \cdot x_i$. The backpropagation equation is used to update the real value matrix M_r. That is, during the whole training procedure, the weight matrix $\mathbf{W}$ is fixed as constant. In *Piggyback*, the mask weigths updates as follows. Here, $A \odot B = C = [c_{ji} = A_{ji} B_{ji}]$.

$$\begin{aligned} \delta m_{ji} = \frac{\partial L}{\partial m_{ji}} &= (\frac{\partial L}{\partial y_j}) \cdot (\frac{\partial y_j}{\partial m_{ji}}) \\ &= \delta y_j \cdot w_{ji} \cdot x_j \\ \therefore \delta \mathbf{m} &= (\delta \mathbf{y} \cdot x^T) \odot \mathbf{W} \end{aligned} \tag{3}$$

3.2.2 Experiment Setting

In practical, it is hard to derive the analytical solution to threshold τ. In the original work, the author set $\tau = 5e - 3$. As for the matrix $\mathbf{m_r}$, the author initialized the value to 0.01. The best results are produced by Adam optimizer. In our experiment, we have inherited the parameter settings in the work. However, as mentioned in the previous statement, the crucial part for *Piggyback* is that the base model have to be fine-tuned, otherwise the model will not work very well. We have also verified such phenomon in our experiment. To improve the performance, we modified the logic of the model. At the first task, we use the pure base model to perform the training, as the pre-trained network(ResNet50(**?**), in our case)cannot perform well on the given dataset. Then we begin performing mask and *Piggyback*. This part will later be discussed in detail in Section 4.

4 Evaluation and Analyze

5 Conclusion

5.1 Footnotes

Footnotes should be used sparingly. If you do require a footnote, indicate footnotes with a number[2] in the text. Place the footnotes at the bottom of the page on which they appear. Precede the footnote with a horizontal rule of 2 inches (12 picas).

Note that footnotes are properly typeset *after* punctuation marks.[3]

5.2 Figures

All artwork must be neat, clean, and legible. Lines should be dark enough for purposes of reproduction. The figure number and caption always appear after the figure. Place one line space before the figure caption and one line space after the figure. The figure caption should be lower case (except for first word and proper nouns); figures are numbered consecutively.

[2]Sample of the first footnote.

[3]As in this example.

Table 1: Sample table title

Part		
Name	Description	Size (μm)
Dendrite	Input terminal	$\sim$100
Axon	Output terminal	$\sim$10
Soma	Cell body	up to 10^6

You may use color figures. However, it is best for the figure captions and the paper body to be legible if the paper is printed in either black/white or in color.

5.3 Tables

All tables must be centered, neat, clean and legible. The table number and title always appear before the table. See Table 1.

Place one line space before the table title, one line space after the table title, and one line space after the table. The table title must be lower case (except for first word and proper nouns); tables are numbered consecutively.

Note that publication-quality tables *do not contain vertical rules.* We strongly suggest the use of the `booktabs` package, which allows for typesetting high-quality, professional tables:

`https://www.ctan.org/pkg/booktabs`

This package was used to typeset Table 1.

References

[1] KIRKPATRICK, J., PASCANU, R., RABINOWITZ, N., VENESS, J., DESJARDINS, G., RUSU, A. A., MILAN, K., QUAN, J., RAMALHO, T., GRABSKA-BARWINSKA, A., ET AL. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences 114*, 13 (2017), 3521–3526.

[2] LI, Z., AND HOIEM, D. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence 40*, 12 (2017), 2935–2947.

[3] MALLYA, A., DAVIS, D., AND LAZEBNIK, S. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 67–82.

[4] MALLYA, A., AND LAZEBNIK, S. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 7765–7773.

[5] MCCLOSKEY, M., AND COHEN, N. J. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, vol. 24. Elsevier, 1989, pp. 109–165.

[6] RUSU, A. A., RABINOWITZ, N. C., DESJARDINS, G., SOYER, H., KIRKPATRICK, J., KAVUKCUOGLU, K., PASCANU, R., AND HADSELL, R. Progressive neural networks. *arXiv preprint arXiv:1606.04671* (2016).

[7] THRUN, S. A lifelong learning perspective for mobile robot control. In *Intelligent Robots and Systems* (1995), Elsevier, pp. 201–214.

[8] YOON, J., YANG, E., LEE, J., AND HWANG, S. J. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547* (2017).