Supplementary Material: Continual Learning with Lifelong Vision Transformer

Zhen Wang[1], Liu Liu[1], Yiqun Duan[3], Yajing Kong[1], Dacheng Tao[2,1]
[1]The University of Sydney, Australia, [2]JD Explore Academy, China, [3]University of Technology Sydney, Australia
{zwan4121,liu.liu1,ykon9947}@sydney.edu.au, yiqun.duan@student.uts.edu.au, dacheng.tao@gmail.com

Appendix A provides detailed experimental settings and implementation of LVT. In Appendix B, we present additional empirical results of Backward Transfer and discuss the limitation of vision transformers for continual learning.

A. Experimental Settings and Implementation

A.1. Experimental Setup

Pytorch[1] is used to implement the proposed algorithm and conduct all the experiments. All the computations are performed on a 64-Bit Linux GPU group with a 36-core Intel Core CPU i7-6850K 3.60GHz processor, 256 GB memory, and 4 Nvidia Tesla V100 GPUs. To provide a fair comparison among continual learning methods, we use the Stochastic Gradient Descent (SGD) optimizer for training all the networks. For data extracting, we apply random cropping and horizontal flipping to both stream and previous examples following [10, 87, 89]. We propagate this choice to competitors for fairness.

A.2. Architecture Details

Patch embedding. Convolutions block with small strides is applied on input to LVT, which preserves local spatial relationships and allows for efficient tokenization to reduce the computational cost of the transformers. In LVT, we only apply 3×3 convolutions (stride 2) with the skip connection and relu activation. The convolutions perform the resolution reduction and the number of channels goes $C = 3, 64, 128$. Using convolutions block for patch embedding makes up for the lack of inductive biases in transformers and improves the stability of LVT for continual learning. The preliminary experiments show that the performance of continual learning can be improved by using the small convnet.

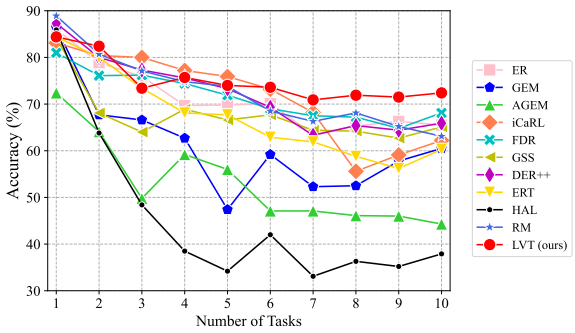
Lifelong Transformer block. LVT is composed of stacked lifelong transformer blocks after a simple convolutional block. We adopt multi-resolution pyramid architectures used in CNNs to built transformer block. Between the LVT stages, we perform a downsampling (Shrink) to reduce the resolution of the activation maps and increase their number of channels between LVT stages. For CIFAR100, we use two LVT stages; For ImageNet, we use three LVT stages. Lifelong transformer block includes a residual structure with proposed inter-task attention and feed forward. We use 1×1 convolution to control the the number of channel, followed by the batch normalization. The implementation of transformer block is based on ViT [21] and LeViT [25]. LVT uses GELU activation and dropout in transformer blocks and applies a global average pooling to the last activation map.

Normalization. Most vision transformers use layer normalization (LN) [6] before each attention. Under layer normalization, all the hidden units in a layer share the same normalization terms, mean and standard deviation, but different training samples have different layer normalization terms. However, in continual learning settings, the data stream is evolving in a non-i.i.d. manner over time, where one mini-batch contains many different data distributions, causing difficulty in converging to the optimal. We claim that batch normalization (BN) [38] is more appropriate for vision transformers in continual learning than LN. Batch normalization uses the distribution of the summed input to a neuron over a mini-batch composed of different task data, improving robustness of continual learning. In LVT, we adopt BN after attention computing.

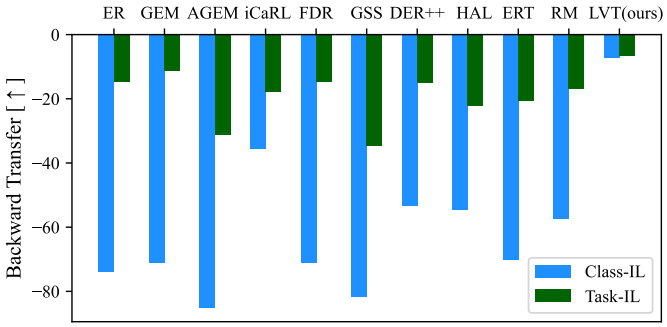
Pooling. Instead of adding an additional token as in vision transformers [21, 83, 90] to perform classification, we apply global average pooling to the last activation map, which produces an embedding used in the classifiers.

Double-classifier Structure. Most rehearsal-based methods use the same classifier for learning new task and replaying previous data in memory $\mathcal{M}$. A key challenge for these methods is the imbalance between old and new classes, as only a small amount of old class data are stored in the memory buffer $\mathcal{M}$. Training on the imbalanced data, the classifier of the network is biased towards classes from the most recent task, causing significant performance degradation for previous tasks, i.e., catastrophic forgetting. To address this problem, LVT proposes to utilize a novel double-classifier structure for independently injecting knowledge without interference, and accumulating knowledge in a balanced manner, then the overall performance could be improved.

[1] https://pytorch.org/



(a) Backward transfer analysis by observing the changes of $a_{t,1}$.



(b) Backward Transfer on CIFAR100 of 10 splits with memory 500.

Figure 6. Backward Transfer Analysis.

Under mild assumptions [34, 45], the optimization of the KL divergence in Eq. (5) is equivalent to minimizing the Euclidean distance between the corresponding pre-softmax responses (*i.e.* logits). We can match the logits using ℓ_2-norm by optimizing the following objective:

$$\mathcal{L}_d = \mathbb{E}_{(x',z')\sim\mathcal{M}}\big[\,\|z' - h_A(x')\|_2^2\,\big]. \tag{9}$$

We approximate the expectation by computing gradients on batches sampled from the memory buffer.

B. Additional Results and Discussion

B.1. Backward Transfer (BWT)

We define classification accuracy $a_{T,t}$ as the testing accuracy on task $\mathcal{T}_t$ when the model completed learning task $\mathcal{T}_T$. By observing the $a_{T,t}$ curve over t, we can see how the stability of the network evolves along the increments. Figure 6a shows the changes of $a_{t,1}$ on CIFAR100 with 10 incremental tasks under Task-IL. The results show that LVT alleviates the performance degradation on previous tasks and mitigates catastrophic forgetting effectively.

Then we compare the Backward Transfer value for different methods in both of Class-IL and Task-IL. Backward Transfer (BWT) [10, 15, 47] is the influence of learning a task on the performance of previous tasks. BWT is also used to evaluate the forgetting of the network where negative BWT indicates forgetting. Different from Average Forgetting, BWT assumes that the highest accuracy value of a task is generated at the end of the task. This is not always true, since rehearsal-based methods can take advantage of memory buffers and even improve their performance in the previous task if they start from low accuracy. In formal, BWT is defined as:

$$\textbf{BWT} = \frac{1}{T-1}\sum_{t=1}^{T-1}(a_{T,t} - a_{t,t}). \tag{10}$$

We analyze BWT for different methods on CIFAR100 of 10 splits with memory 500, as shown in Figure 6b. It is observed that other methods have large negative BWT in the Class-IL setting, which means severe forgetting. In contrast, our method has small negative BWT, which means the learning of new tasks may help previous tasks' performance. This result further proves the superiority of our method.

B.2. The limitation of vision transformers in continual learning

Although vision transformers have been shown to achieve highly competitive performance in a wide range of vision applications [31, 39, 90], they could not directly excel in continual learning. Current vision transformers are "data hungry" models and only fit to i.i.d. large datasets [32]. According to our observation, without sufficient training data data, the Visual Transformers normally could not achieve a good generalized ability on continual learning tasks as tradition CNNs [31] due to the lack of the inductive biases inherent to CNNs, such as *locality* and *translation invariance* [39, 83]. In continual learning settings, the data stream is evolving in a non-i.i.d. manner over time and each task data is not enough to satisfy current Vision Transformers' needs, resulting in the performance of Transformers that cannot outperform their convolutional counterparts. Besides, most transformer architectures require a larger number of parameters than CNNs. However, LVT integrates components that were proven useful for CNNs into vision transformers, which reduces the dependence on a large number of parameters and data.