# Technical Appendix for Decoupling Learning and Remembering: a Bilevel Memory Framework with Knowledge Projection for Task-Incremental Learning

Wenju Sun[1] Qingyong Li[1,2,3] Jing Zhang[1] Wen Wang[1] Yangli-ao Geng[1,2,3,*]
[1]Beijing Key Lab of Traffic Data Analysis and Mining, Beijing Jiaotong University
[2]Frontiers Science Center for Smart High-speed Railway System, Beijing Jiaotong University
[3]The Key Laboratory of Cognitive Computing and Intelligent Information Processing of Fujian Education Institutions, Wuyi University
{SunWenJu,liqy,j_zhang,wangwen,gengyla}@bjtu.edu.cn

## 1. Human Memory Mechanism

The classic Atkinson-Shiffrin human memory model [22] assumes that human memory consists of three separate memory stores: sensory memory, short-term memory, and long-term memory. They play different roles in human learning. The sensory memory is responsible for caching the environmental stimuli captured by sense organs, such as visual signals inputted by eyes. Most of the cached signals decay rapidly unless they attract the attention of the brain. Such signals are then transferred to the short-term memory. The short-term memory has a larger capacity and a longer storage duration, which allows it to process the signals into information to facilitate human decision-making and behavior. Then, the brain utilizes a rehearsal process to re-organize important information into compact representations, which can be viewed as knowledge and will be carried to the long-term memory for storing. The long-term memory has the largest capacity and the longest storage duration, where the stored knowledge can be recalled back to the short-term memory as the brain need.

Remarkably, recent studies [7, 17] suggest that the rehearsal process can be viewed as a self-organization process which is detailed as follows. The brain first tries to represent raw information (from the short-term memory) by utilizing old patterns (stored in the long-term memory) which have been built previously. If the old patterns can represent the raw information well, then the "representation coefficients" instead of the raw information are transferred to the long-term memory for storing. Otherwise, some new patterns will be built from the raw information; the new patterns together with the old patterns are utilized to represent the raw information; then the "representation coefficients" along with the new patterns are transferred to the long-term memory for storing. All the above steps follow the principle of minimum energy consumption.

The memory mechanism and rehearsal process above make human beings good at learning new knowledge incrementally.

## 2. Experimental Details

### 2.1. Experimental Environment

All experiments reported in the manuscript and appendix are conducted on a workstation running OS Ubuntu 16.04 with 18 Intel Xeon 2.60GHz CPUs, 256 GB memory, and 6 NVIDIA RTX3090 GPUs.

### 2.2. List of Hyperparameters

We summarize the hyperparameters for each method used in experiments in Table 1.

### 2.3. Dataset Statistics

In this work, we select three classification datasets to evaluate the proposed BMKP method. We list the statistical information of these three datasets in Table 2.

## 3. Implementation Details

### 3.1. Knowledge Projection for Convolution layer

In the paper, we introduce the process of knowledge projection on linear layers, which can be easily generalized to the convolution layers. Specifically, the convolution layer can be transformed into a special case of the fully connected layer [19]. Let $c_i \times h_i \times w_i$, $c_o \times c_i \times k \times k$ and $c_o \times h_o \times w_o$ be the sizes of the input image (or feature map), the convolution kernel and the output feature map, respectively ($w_i$, $h_i$, and $c_i$ donate the width, height, and channel of the input tensor $X$, respectively; $w_o$, $h_o$, and $c_o$ donate the width, height,

*Corresponding author: Yangli-ao Geng (gengyla@bjtu.edu.cn).

Table 1. List of hyperparameters for the baselines and our BMKP. Here, 'KD' represents "Knowledge distillation".

| **Methods** | **Hyperparameters** |
|---|---|
| LwF [12] | learning rate: 0.01<br>batch size: 32<br>tempture: 2<br>KD weight: 5e-4 |
| SI [31] | learning rate: 0.03 (CIFAR-10), 0.1 (CIFAR-100), 0.05 (Tiny-ImageNet)<br>batch size: 16<br>regularizer weight: 0.5 |
| DGR [21] | learning rate: 0.1 (CIFAR-10), 0.05 (CIFAR-100)<br>batch size: 64 (CIFAR-10), 128 (CIFAR-100)<br>KD weight: 0.1 (CIFAR-10), 1 (CIFAR-100) |
| GEM [14] | learning rate: 0.03 (CIFAR-10, CIFAR-100), 0.05 (Tiny-ImageNet)<br>batch size: 32 (CIFAR-10, CIFAR-100), 64 (Tiny-ImageNet) |
| oEWC [20] | learning rate: 0.03 (CIFAR-10, CIFAR-100), 0.2 (Tiny-ImageNet)<br>batch size: 32 (CIFAR-10, CIFAR-100), 16 (Tiny-ImageNet)<br>regularizer weight: 10 (CIFAR-10), 5 (CIFAR-100), 25 (Tiny-ImageNet) |
| LwM [8] | learning rate: 0.03 (CIFAR-10), 0.01 (CIFAR-100), 0.5 (Tiny-ImageNet)<br>batch size: 32 (CIFAR-10, CIFAR-100), 16 (Tiny-ImageNet)<br>KD weight: 1 (CIFAR-10, CIFAR-100), 0.5 (Tiny-ImageNet)<br>grad cam distillation weight: 1 |
| DI [27] | learning rate: 0.005 (CIFAR-10), 0.05 (CIFAR-100), 0.1 (Tiny-ImageNet)<br>batch size: 32 (CIFAR-10, CIFAR-100), 16 (Tiny-ImageNet)<br>KD weight: 0.5 (CIFAR-10, CIFAR-100), 5 (Tiny-ImageNet) |
| DER [3] | learning rate: 0.03<br>batch size: 32<br>KD weight: 0.3 (CIFAR-10), 0.1 (CIFAR-100, Tiny-ImageNet) |
| HAL [4] | learning rate: 0.03 (CIFAR-10), 0.05 (CIFAR-100)<br>batch size: 32 |
| PASS [32] | learning rate: 1e-4 (CIFAR-10, Tiny-ImageNet), 5e-4 (CIFAR-100)<br>batch size: 64 (CIFAR-10, CIFAR-100), 32 (Tiny-ImageNet)<br>KD weight: 10 (CIFAR-10, Tiny-ImageNet), 0.2 (CIFAR-100)<br>prototype weight: 0.2 (CIFAR-10), 0.05 (CIFAR-100), 0.5 (Tiny-ImageNet) |
| GPM [19] | learning rate: 0.02 (CIFAR-10), 0.01 (CIFAR-100), 0.02 (Tiny-ImageNet)<br>batch size: 64 (CIFAR-10, CIFAR-100), 32 (Tiny-ImageNet) |
| Adam-NSCL [23] | learning rate: 1e-5 (CIFAR-10), 1e-4 (CIFAR-100), 5e-5 (Tiny-ImageNet)<br>batch size: 32 (CIFAR-10, CIFAR-100), 16 (Tiny-ImageNet) |
| CLS-ER [2] | learning rate: 0.1 (CIFAR-10, CIFAR-100), 0.05 (Tiny-ImageNet)<br>batch size: 32 |
| BMKP (ours) | learning rate: 0.05 (CIFAR-10, CIFAR-100), 0.03 (Tiny-ImageNet)<br>batch size: 128<br>regularizer weight: 1 (CIFAR-100), 10 (CIFAR-10, Tiny-ImageNet) |

and channel of the output tensor, respectively, and $k$ denotes the size of convolution kernel). As shown in Figure 1, the convolution process can be reformulated as a product of two matrices with the sizes of $(h_o \cdot w_o) \times (c_i \cdot k \cdot k)$ and

Table 2. Dataset statistics.

| Dataset | Split CIFAR-10 | Split CIFAR-100 | Split Tiny-ImageNet |
|---|---|---|---|
| num. of tasks | 5 | 10 | 10 |
| input size | $3 \times 32 \times 32$ | $3 \times 32 \times 32$ | $3 \times 64 \times 64$ |
| # Classes/task | 2 | 10 | 20 |
| # Training samples/task | 9,000 | 4,500 | 9000 |
| # Validation samples/task | 1,000 | 500 | 1000 |
| # Test samples/task | 2,000 | 1,000 | 1000 |

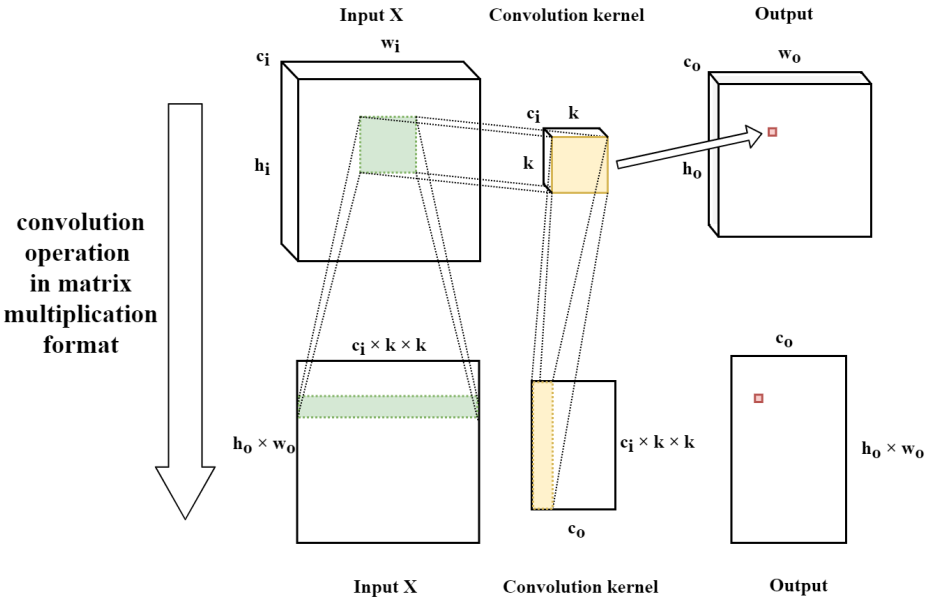


Figure 1. Diagram of convolution operation in matrix multiplication format. $w_i$, $h_i$, and $c_i$ donate the width, height, and channel of the input tensor $X$, respectively; $w_o$, $h_o$, and $c_o$ donate the width, height, and channel of the output tensor, respectively, and $k$ denotes the size of convolution kernel.

$(c_i \cdot k \cdot k) \times c_o$. With the above transformation, we can perform the knowledge projection on the generated $(h_o \cdot w_o) \times (c_i \cdot k \cdot k)$ matrix.

### 3.2. Network Parameter Composing

During incremental learning, the dimension of pattern basis $B$ increases as the number of learned tasks. However, the previously learned $A_t$ will not change, which can incur the dimension mismatching problem between $B$ and $A_t$. To handle this issue, the expansion of $B$ is constrained to be strictly incremental, meaning that the previously learned basis in $B$ keeps unchanging when adding new basis into $B$. To be more specific, as shown in Figure 2, take layer $l$ as an example, the first $m'$ row of $B^l$ are exactly the basis after learning the first $t$ tasks. Thus, during working memory parameter composing, BMKP clip the superfluous basis in $B^l$ and then multiplies $A_t^l$ with it to obtain the composed parameter $W_t^l$. With this design, the previously learned $A_t$ is still applicable to the expanded $B$.

## 4. The Influence of Basis on Each Task

We next analyze how each basis affects each task. To this end, we calculated the weight of last layer knowledge representations of all tasks $\{A_1^L, A_2^L, ..., A_T^L\}$ on all basis and presented it in a heat map in Figure 3. As can be seen, all basis contribute to all subsequent tasks. Moreover, weights on the new basis are smaller than on the old, showing that BMKP tends to reuse the shared knowledge.

## 5. Additional Experiments

### 5.1. The Computational Efficiency of BMKP

Compared with the baseline methods, BMKP has two additional steps: the knowledge projection and the long-term memory updating, which may incur more computation consumption. Nevertheless, we will show the computation

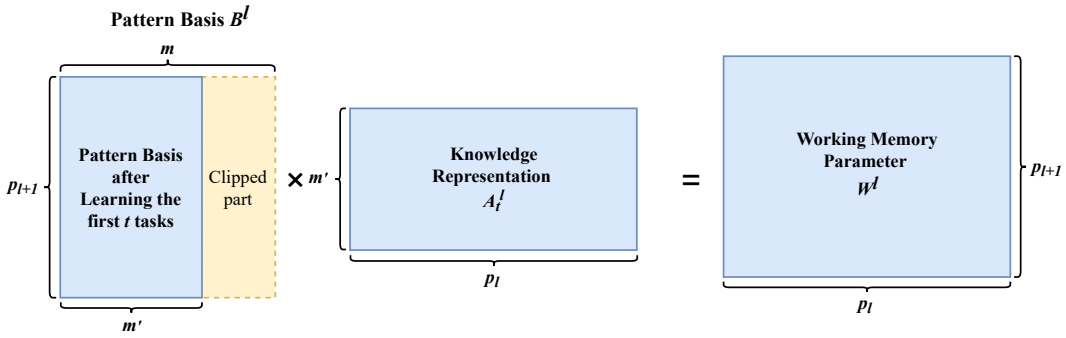


Figure 2. Diagram about how to "multiply" the pattern basis $B^l \in \mathbb{R}^{p_{l+1} \times m'}$ by the knowledge representation $A_t^l \in \mathbb{R}^{m \times p_l}$ to get the network parameter $W_t^l \in \mathbb{R}^{p_{l+1} \times p_l}$. $m'$ and $m$ ($m' \leq m$) denote the number of basis after learning the first $t$ tasks and learning all the tasks, respectively. $p_l$ and $p_{l+1}$ represent the number of neurons in the network layer $l$ and $l+1$, respectively.

Table 3. Time consumption (second) in last incremental task.

| **Methods** | **Split CIFAR-10** | **Split CIFAR-100** | **Split Tiny-ImageNet** |
|---|---|---|---|
| GPM [19] training | 389 | 390 | 1155 |
| DER++ [3] training | 1450 | 1457 | 6073 |
| Working memory learning | 373 | 402 | 1192 |
| Knowledge projection | 38 | 39 | 180 |
| Long-term memory updating | 72 | 72 | 108 |
| BMKP (ours) training | 483 | 513 | 1480 |
| GPM [19] testing | 4.1 | 4.1 | 4.9 |
| DER++ [3] testing | 4.5 | 4.5 | 4.7 |
| BMKP (ours) testing | 4.3 | 4.5 | 5.2 |

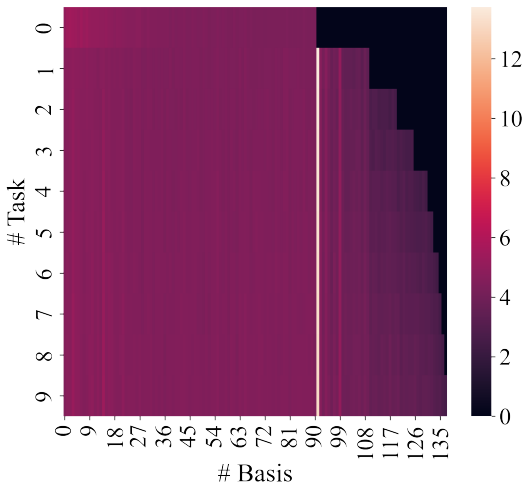


Figure 3. Diagram of the contribution of each basis for the last layer knowledge representation of all tasks.

complexity of these two steps is acceptable.

During the knowledge projection step, the major computational overhead stems from the singular value decomposition (SVD), which incurs a $\mathcal{O}(HW^2)$ computational cost for a $H \times W$ matrix (assuming $H > W$ without loss of generality) [19]. Specifically for our convolution scenario, let $c_i \times h_i \times w_i$, $c_o \times c_i \times k \times k$ and $c_o \times h_o \times w_o$ be the sizes of the input image (or feature map), the convolution kernel and the output feature map, respectively. As shown in Figure 1, the convolution process can be reformulated as a product of two matrices with the sizes of $(h_o \cdot w_o) \times (c_i \cdot k \cdot k)$ and $(c_i \cdot k \cdot k) \times c_o$. With this transformation, what the SVD applying on is actually a $(h_o \cdot w_o) \times (c_i \cdot k \cdot k)$ matrix (i.e., $H = h_o \cdot w_o$ and $W = c_i \cdot k \cdot k$,), which cause a $\mathcal{O}\left((h_o \cdot w_o) \times (c_i \cdot k \cdot k)^2\right)$ computational cost. Therefore the total computational overhead of the knowledge projection step is $\sum \mathcal{O}\left((h_o \cdot w_o) \times (c_i \cdot k \cdot k)^2\right)$, where the summation is taken over all the network layers. As the kernel side $k$ is usually small enough (e.g., 3 in our case), the computational consumption of the knowledge projection step is generally acceptable.

For the long-term memory updating step, retraining the

knowledge representation $A$ will take up additional time. However, since this step is only used for fine-tuning, we normally take a small number of iteration epochs. Besides, the parameter that requires gradient update is smaller (the size of $A$ is smaller than that of the original network parameter $W$), which also saves some time.

Table 3 reports the time consumption of each stage in BMKP. As can be seen, the knowledge projection and the long-term memory updating step take much lesser time compared with the working memory learning stage. Consequently, the total time consumption of BMKP is acceptable.

As for the inference, BMKP needs to re-build the working memory parameters according to the task label, which indeed incurs extra time costs. Nevertheless, the bottom part of Table 3 empirically suggests that the inference speed of BMKP is still competitive compared with the two baselines.

### 5.2. Results using Larger Backbone

We carried out experiments with ResNet34 [10] as the network backbone, and report the result in Table 4.

Table 4. Result (ACC %) with ResNet34 as the network backbone.

| **Methods** | **Split CIFAR-10** | **Split CIFAR-100** |
|---|---|---|
| GPM | 84.96 | 66.14 |
| BMKP | 93.58 | 77.2 |

### 5.3. Comparison with Mask-based methods

Following SparCL, we carried out experiments on CIFAR-10 and Tiny-ImageNet, and the results are reported in Table 5. The compared mask-based methods include PackNet [15], LPS [24], and SparCL [25]. Notably, SparCL also keeps an exemplar memory with 500 samples. As can be seen, BMKP has a similar performance to SparCl on the easier CIFAR10 tasks. Besides, in the more challenging Tiny-ImageNet, BMKP performs much better than all comparison methods. Notably, the performance of BMKP is much more stable than others, with the lowest variance.

Table 5. Comparison result (ACC %) with with Mask-based methods on CIFAR-10 and Tiny-ImageNet experiments. Results with (†) stem from SparCL [25].

| **Methods** | **Split CIFAR-10** | **Split Tiny-ImageNet** |
|---|---|---|
| PackNet† | 93.73±0.6 | 61.88±1.0 |
| LPS† | 94.50±0.5 | 63.37±0.8 |
| SparCL† | **95.19**±0.3 | 52.19±0.4 |
| BMKP | 94.49±0.2 | **70.36**±0.2 |

Table 6. Comparison with expansion-based methods on 20-split-CIFAR-100 Superclass dataset. (†) and (‡) denote the result reported from APD [28] and GPM [19], respectively. (*) indicates the methods that do not adhere to Task-IL setup. Single-task learning (STL) trains a separate network for each task and serves as an upper bound of Task-IL methods.

| Methods | ACC (%) | Capacity (%) |
|---|---|---|
| STL*† | 61.00 | 2000 |
| PNN† | 50.76 | 271 |
| DEN† | 51.10 | 191 |
| RCL† | 51.99 | 184 |
| APD† | 56.81 | 130 |
| GPM‡ | 57.72 | **100** |
| BMKP(ours) | **57.82** | **100** |

Table 7. Comparison results (ACC %) on 20-split CIFAR-100 and 25-split Tiny-ImageNet. Results with (†) stem from Adam-NSCL [23].

| **Methods** | **20-split CIFAR-100** | **25-split Tiny-ImageNet** |
|---|---|---|
| EWC† | 71.66 | 52.33 |
| MAS† | 63.84 | 47.96 |
| MUC-MAS† | 67.22 | 41.18 |
| SI† | 59.76 | 45.27 |
| LWF† | 74.38 | 56.57 |
| InstAParam† | 51.04 | 34.64 |
| GD-WILD† | 77.16 | 42.74 |
| GEM† | 68.89 | - |
| A-GEM† | 61.91 | 53.32 |
| MEGA† | 64.98 | 57.12 |
| OWM† | 68.47 | 49.98 |
| Adam-NSCL† | 75.95 | 58.28 |
| GPM | 77.55 | 68.62 |
| BMKP (ours) | **85.81** | **80.01** |

### 5.4. Comparison with Expansion-based Methods

Following Additive Parameter Decomposition (APD) [28], we carry out experiments on 20-split-CIFAR-100 Superclass dataset. The compared expansion-based methods include Single Task Learning (STL), Progressive Neural Network (PNN) [18], Dynamically Expandable Networks (DEN) [29], Reinforced Continual Learning (RCL) [26], APD [28]. Notably, STL is the upper bound of Task-IL methods since it trains a separate network for each task. Besides, we also compare BMKP with a memory-based method GPM [19]. In this experiment, all methods and BMKP use the LeNet-5 as the network backbone, and we report the classification accuracy over all learned tasks and the net-

Table 8. Diagram of memory usage (MB) of each components in BMKP during incremental learning on CIFAR100.

| Methods \ Task No. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Adam-NSCL [23] | 38.42 | 38.42 | 38.42 | 38.42 | 38.42 | 38.42 | 38.42 | 38.42 | 38.42 | 38.42 |
| GPM [19] | 21.09 | 27.61 | 29.94 | 31.18 | 31.57 | 31.74 | 31.83 | 31.83 | 32.08 | 32.10 |
| Pattern Basis | 0.31 | 0.35 | 0.36 | 0.36 | 0.36 | 0.36 | 0.37 | 0.37 | 0.37 | 0.37 |
| Knowledge Representations | 2.26 | 4.84 | 7.46 | 10.12 | 12.80 | 15.49 | 18.19 | 20.90 | 23.62 | 26.34 |
| BMKP | 2.57 | 5.19 | 7.82 | 10.48 | 13.16 | 15.85 | 18.56 | 21.27 | 23.99 | 26.71 |

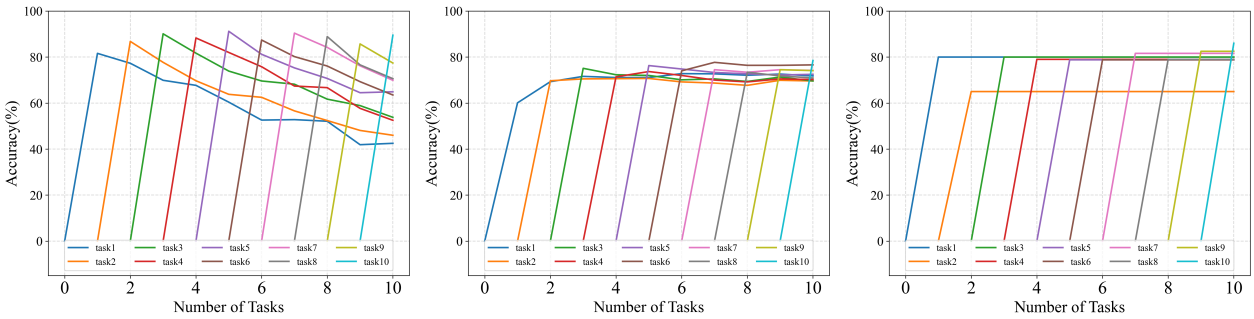


Figure 4. Diagram of ACC (%) of each task during incremental learning on CIFAR-100.

work capacity compared with the standard LeNet-5. The results are reported in Table 6, both BMKP and GPM are memory-based methods and do not expand network capacity. Besides, BMKP achieves the best performance with an average accuracy of 57.82% with fixed network capacity.

### 5.5. Comparison with More Tasks

This section evaluates the performance of BMKP with a longer sequence of tasks. Following Adam-NSCL, we conduct experiments on 20-split CIFAR-100 and 25-split Tiny-ImageNet, where 20-split CIFAR-100 is constructed by splitting CIFAR-100 into 20 tasks, and 25-split-TinyImageNet is constructed by splitting Tiny-ImageNet into 25 tasks. The comparison methods in this experiment include Elastic Weight Consolidation (EWC) [11], Memory Aware Synapses (MAS) [1], MUC-MAS [13], Synaptic Intelligence (SI) [31], Learning without Forgetting (LwF) [12], InstAParam [6], GD-WILD [16], A-GEM [5], MEGA [9], OWM [30], and Adam-NSCL [23]. Table 7 reports the comparison results. On 20-split CIFAR-100, our method achieves the best accuracy of 85.81%, which is 8.26% higher than the second-best performance. As for the 25-split Tiny-ImageNet, BMKP overwhelms all other comparison methods and achieves 80.01% average accuracy. This experiment shows that BMKP has the incremental ability for the longer sequence of tasks.

### 5.6. The Growth of Model size

The size growth of each part in BMKP is shown in Table 8. As we can see, the size of the pattern basis (i.e., $B$) is quite stable, but the size of the knowledge representations (i.e., $A$) increase linearly with the number of learned tasks.

### 5.7. Plasticity and Stability Analysis

Figure 4 illustrates the classification accuracy of each task during incremental learning on CIFAR-100. As can be seen, for DER, each task performance peaks after finishing training and decays with subsequent task learning. Because the training data for episodic-memory based methods include sufficient new-task samples and partial old-task samples, the sufficient new-task samples guarantee the high plasticity for DER, while the partial old-task samples may not support the original data distribution and can not entirely prevent forgetting. In contrast, the performance of GPM for each task is stable due to its gradient projection mechanism. However, the performance of each task after training is smaller than BMKP and GPM, revealing that the gradient direction restriction leads to low plasticity of GPM. As for our BMKP, the working memory guarantees high plasticity, and the long-term memory ensures the non-degrading performance of each task.

Table 9. Incremental learning results of BMKP with different $\lambda$ for the first or the last 5 layers on CIFAR-100.

| Layers | $\lambda$ | 1 | 2 | 5 | 10 |
|---|---|---|---|---|---|
| **First 5** | ACC (%) | 79.62 | 80.84 | 80.05 | 79.79 |
| | Memory (MB) | 26.75 | 26.29 | 26.12 | 26.48 |
| **Last 5** | ACC (%) | 79.62 | 79.35 | 80.18 | 79.55 |
| | Memory (MB) | 26.75 | 25.89 | 24.78 | 24.70 |

### 5.8. Effectiveness of $L_{reg}$ for Different Layer

In this section, we tried to tune the weight of $L_{reg}$ for different layers. As shown in Table 9, higher weights (2 for the first five layers) for shallower layers bring BMKP better performance (in both accuracy and memory usage).

## References


[1] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *European Conference on Computer Vision*, pages 139–154, 2018. 6

[2] Elahe Arani, Fahad Sarfraz, and Bahram Zonooz. Learning fast, learning slow: a general continual learning method based on complementary learning system. In *International Conference on Learning Representations*, 2022. 2

[3] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and SIMONE CALDERARA. Dark experience for general continual learning: A strong, simple baseline. In *Advances in Neural Information Processing Systems*, volume 33, pages 15920–15930, 2020. 2, 4

[4] Arslan Chaudhry, Albert Gordo, Puneet Kumar Dokania, Philip Torr, and David Lopez-Paz. Using hindsight to anchor past knowledge in continual learning. In *AAAI Conference on Artificial Intelligence*, 2021. 2

[5] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *International Conference on Learning Representations*, 2019. 6

[6] Hung-Jen Chen, An-Chieh Cheng, Da-Cheng Juan, Wei Wei, and Min Sun. Mitigating forgetting in online continual learning via instance-aware parameterization. In *Advances in Neural Information Processing Systems*, volume 33, pages 17466–17477, 2020. 6

[7] Rosemary A Cowell, Morgan D Barense, and Patrick S Sadil. A roadmap for understanding memory: Decomposing cognitive processes into operations and representations. *Eneuro*, 6(4), 2019. 1

[8] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2019. 2

[9] Yunhui Guo, Mingrui Liu, Tianbao Yang, and Tajana Rosing. Improved schemes for episodic memory-based lifelong learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 1023–1035, 2020. 6

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 5

[11] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. 6

[12] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2017. 2, 6

[13] Yu Liu, Sarah Parisot, Gregory Slabaugh, Xu Jia, Ales Leonardis, and Tinne Tuytelaars. More classifiers, less forgetting: A generic multi-classifier paradigm for incremental learning. In *European Conference on Computer Vision*, pages 699–716, 2020. 6

[14] David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, volume 30, pages 6467–6476, 2017. 2

[15] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018. 5

[16] Inyoung Paik, Sangjun Oh, Taeyeong Kwak, and Injung Kim. Overcoming catastrophic forgetting by neuron-level plasticity control. In *AAAI Conference on Artificial Intelligence*, volume 34, pages 5339–5346, 2020. 6

[17] Maureen Ritchey and Rose A Cooper. Deconstructing the posterior medial episodic network. *Trends in cognitive sciences*, 24(6):451–465, 2020. 1

[18] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. 5

[19] Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations*, 2021. 1, 2, 4, 5, 6

[20] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pages 4528–4537, 2018. 2

[21] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, volume 30, 2017. 2

[22] Kenneth W Spence and Janet Taylor Spence. *Psychology of learning and motivation*. Academic Press, 1967. 1

[23] Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 184–193, 2021. 2, 5, 6

[24] Zifeng Wang, Tong Jian, Kaushik Chowdhury, Yanzhi Wang, Jennifer Dy, and Stratis Ioannidis. Learn-prune-share for lifelong learning. In *IEEE International Conference on Data Mining*, pages 641–650, 2020. 5

[25] Zifeng Wang, Zheng Zhan, Yifan Gong, Geng Yuan, Wei Niu, Tong Jian, Bin Ren, Stratis Ioannidis, Yanzhi Wang, and Jennifer Dy. Sparcl: sparse continual learning on the rdge. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. 5

[26] Ju Xu and Zhanxing Zhu. Reinforced continual learning. In *Advances in Neural Information Processing Systems*, volume 31, pages 907–916, 2018. 5

[27] Hongxu Yin, Pavlo Molchanov, Jose M. Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K. Jha, and Jan Kautz. Dreaming to distill: data-free knowledge transfer via deepinversion. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2020. 2

[28] Jaehong Yoon, Saehoon Kim, Eunho Yang, and Sung Ju Hwang. Scalable and order-robust continual learning with additive parameter decomposition. In *International Conference on Learning Representations*, 2020. 5

[29] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018. 5

[30] Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1(8):364–372, 2019. 6

[31] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995, 2017. 2, 6

[32] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5871–5880, June 2021. 2