Remembering Transformer for Continual Learning

Yuwei Sun[1,2], Ippei Fujisawa[1], Arthur Juliani[3], Jun Sakuma[2,4,*], Ryota Kanai[1,*]
[1]Araya, [2]RIKEN AIP, [3]Microsoft Research, [4]Tokyo Institute of Technology

Abstract

Neural networks encounter the challenge of Catastrophic Forgetting (CF) in continual learning, where new task knowledge interferes with previously learned knowledge. We propose Remembering Transformer, inspired by the brain's Complementary Learning Systems (CLS), to tackle this issue. Remembering Transformer employs a mixture-of-adapters and a generative model-based routing mechanism to alleviate CF by dynamically routing task data to relevant adapters. Our approach demonstrated a new SOTA performance in various vision continual learning tasks and great parameter efficiency.

1. Introduction

The Catastrophic Forgetting (CF) problem arises in the sequential learning of neural networks, wherein the new task knowledge tends to interfere with old knowledge, with previously learned tasks forgotten. The ability to learn a new task without interfering with previously learned ones is of great importance to achieve continual learning. An intuitive approach to CF is model fine-tuning [18]. A model retains performance on previous tasks by replaying old task samples during training on a new task. Retraining the model on the old task data alleviates the forgetting in continual learning. Nevertheless, as the number of encountered tasks increases, it necessitates the storage of a large number of old task samples for memory replay. In addition, memory replay cannot eliminate knowledge interference among tasks, leading to suboptimal performance.

Biological neural networks exhibit apparent advantages via the Complementary Learning Systems (CLS) [10, 13]. In CLS, the Hippocampus rapidly encodes task data and then consolidates the task knowledge into the Cortex by forming new neural connections. The hippocampus evolved a novelty detection mechanism to facilitate the consolidation by switching among neural modules in the Cortex for various tasks [4, 7]. Inspired by the CLS in the brain, we propose a novel Remembering Transformer architecture based on mixture-of-adapters and generative routing to address the drawbacks of conventional continual learning. In particular, we propose a generative model-gated mixture-of-adapter architecture for continual learning with a pretrained Vision Transformer. A collection of generative models routes task data to the most relevant adapter by remembering different tasks' data distributions and adaptively activate adapters to finetune the pretrained Transformer model.

The main contributions of this work include: (1) We propose the Remembering Transformer inspired by the biological Complementary Learning Systems to tackle the Catastrophic Forgetting. (2) This study explores two of the most difficult real-world scenarios in continual learning, i.e., class-incremental learning without task identity information during inference, and learning with a model parameter size constraint. (3) The empirical results, including an ablation study, demonstrate the superiority of Remembering Transformer compared to a broad spectrum of existing methods, in terms of parameter efficiency and model performance in various vision continual learning tasks.

2. Methodology

In this section, we present the assumptions and technical underpinnings of Remembering Transformer (Figure 1).

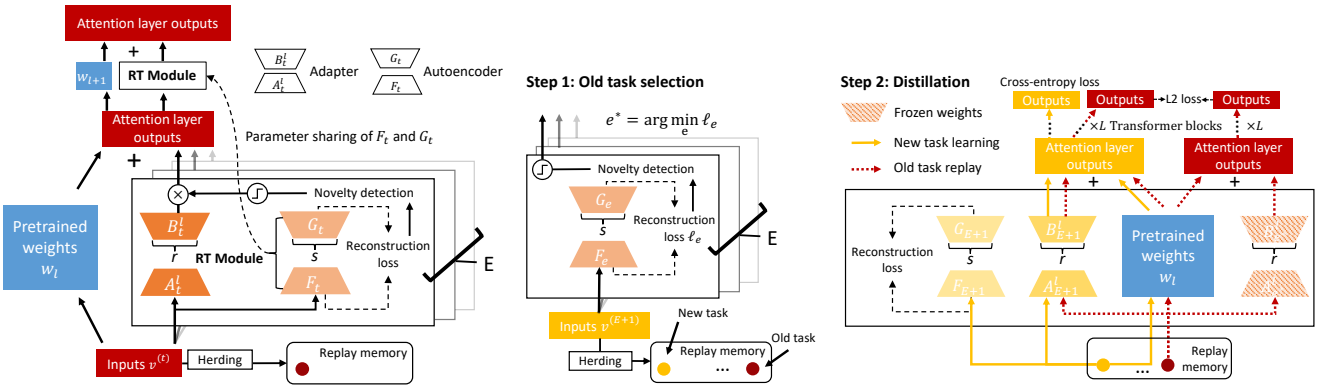
2.1. Continual learning task setting

We aim to tackle continual learning on a sequence of tasks $\{T^{(1)}, T^{(2)}, \ldots, T^{(N)}\}$, where $T^{(t)} = \{x_i^t, y_i^t\}_{i=1}^{K_t} \subset T$ is a labeled dataset including K_t pairs of instances $x^{(t)}$ and their corresponding labels $y^{(t)}$. We investigate the most challenging setting of class-incremental learning, where each task $T^{(t)}$ is defined by a non-overlapping subset of a dataset $T^{(t)} \subset \{x_i, y_i\}_{i=1}^{K}$ where $K = \sum_{t=1}^{N} K_t$. $Y^t \subset Y$ is changing over time, i.e., $Y^i \cap Y^j = \emptyset \ \forall i \neq j$. The posterior $P(X|Y)$ is consistent across tasks $T^{(t)} \subset T$. During inference, we evaluate on $P(X) \sim \frac{1}{K} \sum_{t=1}^{N} P(X|Y)P_t(Y)$ *without the task identity information* t.

2.2. Mixture-of-Adapters in Vision Transformers

Our method is inspired by the Mixture-of-Experts (MoE) [1, 3, 16], with diverse neural modules constituting a comprehensive neural network. Different parts of the neural net-

*Equal advising; Corresponding author: yuwei_sun@araya.org



(a) Remembering Transformer (RT) leverages a mixture of low-rank RT modules that are sparsely activated with generative model-based novelty detection.

(b) Adapter fusion based on knowledge distillation with a limited capacity E. We update the adapter $\{B^l_{E+1}, A^l_{E+1}\}_{l=1}^L$ based on the new data and the soft probability output of the old task replay memory.

Figure 1. The scheme of the proposed Remembering Transformer.

work learn and adapt independently without disrupting the knowledge that other parts have acquired, mitigating interference between tasks. The intuition is to learn a diverse collection of adapter neural networks using a pretrained Vision Transformer (ViT) and adaptively select the most relevant adapters for different tasks.

ViT partitions an image into a sequence of non-overlapping patches. Let $x \in \mathbb{R}^{H\times W\times C}$ be an image input, where (H, W) is the resolution of the image and C is the number of channels. x is separated into a sequence of 2D patches $x_p \in \mathbb{R}^{N\times(P^2\cdot C)}$, where (P, P) is the resolution of each image patch and $\frac{HW}{P^2}$ is the number of patches. These patches are mapped to tokens $v_p \in \mathbb{R}^{\frac{HW}{P^2}\times D}$ with a learnable linear projection. A learnable 1D position embedding $v_0 \in \mathbb{R}^D$ is prepended to the tokens to retain positional information, resulting in $v \in \mathbb{R}^{(\frac{HW}{P^2}+1)\times D}$.

In continual learning, for each task $T^{(t)} \subseteq T$, we employ the Low-Rank Adaptation (LoRA) method [6] for efficient fine-tuning with low-rank decomposition matrices. Notably, we employ the trainable decomposition matrices to the attention weights of ViT's different layers. For each attention layer l, linear transformations are applied to the query, key, value, and output weights $(W_l^Q, W_l^K, W_l^V, W_l^O)$. For $W_l \in \mathbb{R}^{D\times D} \in (W_l^Q, W_l^K, W_l^V, W_l^O)$, the parameter update ΔW_l is then computed by $W_l + \Delta W_l = W_l + B^l A^l$, where $A^l \in \mathbb{R}^{D\times r}, B^l \in \mathbb{R}^{r\times D}$, and r is a low rank $r \ll D$. For an input token $v^{(t)}$, the output of the lth attention layer is $\hat{W}_l v^{(t)} + \Delta W_l v^{(t)} = \hat{W}_l v^{(t)} + B^l A^l v^{(t)}$ where $\hat{\cdot}$ indicates untrainable parameters. For each task $T^{(t)}$, $\theta^t_{\text{adapter}} = \{A^l_t, B^l_t\}_{l=1}^L$ is added to ViT $\{\theta_{\text{ViT}}, \theta^t_{\text{adapter}}\}$.

2.3. Generative model-based novelty detection for expert routing

An adapter is added when a new task is detected. We discuss the proposed novelty detection mechanism for effective expert routing in the Mixture-of-Adapters of ViT. In particular, a routing neural network outputs gating weights $W_g \in \mathbb{R}^E$ that indicate which adapter is employed for a specific task, where E is the number of existing adapters. Conventional routing functions in MoE struggle with the more complex scenario of continual learning, where sequential inputs are involved and gating weights need continual updates. To tackle this challenge, we propose a generative model-based novelty detection method tailored for the Mixture-of-Adapters architecture in the pretrained ViT.

A generative model encodes tokens from a specific task and assesses the familiarity of a new task in relation to the encoded knowledge of the old task. Notably, we employ a low-rank autoencoder (AE) θ_{AE} that consists of an encoder and decoder, each of which leverages one linear transformation layer $F \in \mathbb{R}^{D\times s}$ and $G \in \mathbb{R}^{s\times D}$ where s is a low rank $s \ll D$. We train an AE to encode tokens of the embedding layer output $V^{(t)}$, which are flattened and passed through a Sigmoid activation $\sigma(\cdot)$, $\hat{V^{(t)}} = GF\sigma(V^{(t)})$. The Sigmoid activation constraints the reconstructed tokens between the range of 0 and 1 to facilitate learning. Then, the AE is updated based on the mean squared error loss in Eq. (1). Additionally, for parameter efficiency, the routing model θ_{AE} is globally shared across different layers of ViT.

$$\ell_t(\theta_{\text{AE}}) = (\sigma((V^{(t)}) - \hat{V^{(t)}})^2, \qquad \hat{\theta}_{\text{AE}} = \arg\min_{\theta_{\text{AE}}} \ell_t. \tag{1}$$

During inference, the collection of AEs $\hat{\theta}^e_{\text{AE}} \in \{\hat{G}_e, \hat{F}_e\}_{e=1}^E$ trained on various tasks provide estimates of a new task input v's novelty in relation to the previously learned tasks with the reconstruction loss $\ell_e(v)$. A sample is better reconstructed by an AE trained on a similar task. Consequently, the task e^* corresponding to the AE with the minimum reconstruction loss is the most relevant and the adapter $\theta^{e^*}_{\text{adapter}}$ is leveraged. We devise the novelty detection mechanism for the gating weights W_g as follows:

$$\ell_e = (\sigma(v) - \hat{G}_e \hat{F}_e \sigma(v))^2,$$
$$e^* = \arg\min_e \ell_e, \tag{2}$$
$$W_g(e) = \begin{cases} 1 & \text{if } e = e^* \\ 0 & \text{otherwise.} \end{cases}$$

The output v_l of the lth attention layer is then computed by $\hat{W}_l v_{l-1} + \sum_{e=1}^{E} W_g(e) B_e^l A_e^l v_{l-1}$, where v_{l-1} is the output of the $l-1$th attention layer.

2.4. Adapter fusion based on knowledge distillation

Remembering Transformer reduces the model parameter size by leveraging the low-rank adapter and low-rank generative routing. With the increasing number of tasks, it is encountered with increasing model parameters. To further improve its parameter efficiency, *we investigate the case where a maximum number of adapters exists.* Our aim is to aggregate similar adapters through knowledge distillation [5] that transfers the knowledge of a selected old adapter $\theta^{e^*}_{\text{adapter}}$ to the new adapter $\theta^{t}_{\text{adapter}}$ when the model capacity E for new adapters is reached, i.e., $t = E + 1$.

Moreover, to reduce the computational cost of knowledge distillation, we leverage the Herding method [15, 19] to obtain a small set of representative tokens for each task. In particular, given tokens from a specific class y, $V_y^{(t)} = \{v_{y,1}^{(t)}, v_{y,2}^{(t)}, \ldots, v_{y,K_t^y}^{(t)}\}$ where K_t^y is the total number of samples from class y, we compute the class center μ_y based on the latent features learned by the autoencoder $\hat{\theta}^t_{\text{AE}} := \{\hat{G}_t, \hat{F}_t\}$. $\mu_y \leftarrow \frac{1}{K_t^y} \sum_{i=1}^{K_t^y} \hat{F}_t \sigma(v_{y,i}^{(t)})$ is employed as the most representative token of class y. Then, the distance of each token to the class center $||\mu_y - \hat{F}_t \sigma(v_{y,i}^{(t)})||$ is ranked in ascending order. For each class $y \in Y^t$, we select the top-$\frac{M}{U}$ tokens (U is the number of classes and M is thse replay memory size) with the least distance to the class center and add them to the replay memory Ξ_t of task t.

The old adapter e^* for knowledge distillation is selected based on the reconstruction loss in Eq. (2). We compute the soft probability output $f(\Xi_{e^*}; \{\hat{\theta}_{\text{ViT}}, \theta^{e^*}_{\text{adapter}}\})$ of the old adapter when taking its replay memory samples Ξ_{e^*} as input. Then, the new adapter $\theta^{E+1}_{\text{adapter}}$ is updated based on both the new task $\{V^{E+1}, Y^{E+1}\}$ and the replay memory of the old task $\{\Xi_{e^*}, f(\Xi_{e^*}; \{\hat{\theta}_{\text{ViT}}, \theta^{e^*}_{\text{adapter}}\})\}$. We employ the cross-entropy loss $\ell_{\text{CE}}(\cdot)$ and the L2 loss $\ell_{\text{L2}}(\cdot)$ for training on the new and old tasks, respectively. We formulate the optimization process for the adapter fusion as follows:

$$\ell_{\text{CE}} = -\sum_{i=1}^{K_{E+1}} f(x_i^{E+1}; \{\hat{\theta}_{\text{ViT}}, \theta^{E+1}_{\text{adapter}}\}) \log(y_i^{E+1}),$$
$$\ell_{\text{L2}} = \sum_{i=1}^{M} (f(v_i^{e^*}; \{\hat{\theta}_{\text{ViT}}, \theta^{e^*}_{\text{adapter}}\}) - f(v_i^{e^*}; \{\hat{\theta}_{\text{ViT}}, \theta^{E+1}_{\text{adapter}}\}))^2, \tag{3}$$
$$\hat{\theta}^{E+1}_{\text{adapter}} = \arg\min_{\theta^{E+1}_{\text{adapter}}} \alpha \cdot \ell_{\text{CE}} + (1-\alpha) \cdot \ell_{\text{L2}},$$

where α is a coefficient to balance the two loss items.

After the knowledge distillation, the old adapter e^* is removed and the old task is rerouted to the newly learned adapter $E+1$, i.e., $\text{Gate}(e^*) \rightarrow E+1$ where $\text{Gate}(\cdot)$ is a function to project the result of the generative routing to the new gate. $\text{Gate}(\cdot)$ is initialized with an identity projection $\text{Gate}_{\text{init}}(e) = e$. As a result, we formulate the routing weights as follows: $W_g(e) = \begin{cases} 1 & \text{if } e = \text{Gate}(e^*) \\ 0 & \text{otherwise.} \end{cases}$

3. Experiments

In this section, we present the settings and extensive empirical results for Remembering Transformer, comparing with a wide range of conventional continual learning methods.

3.1. Settings

Datasets We evaluated the model performance based on continual learning datasets adapted from CIFAR10 and CIFAR100 [9]. We investigated class-incremental learning tasks where each dataset is split into several subsets of equal numbers of classes and each subset is a task. In particular, we divided the CIFAR-10 dataset into five splits. The first task consists of classes 0 - 1 and the second task consists of classes 2 - 3 and so on. We divided the CIFAR100 dataset into 10 splits and 20 splits, respectively.

Metrics All tasks were trained for $\frac{\text{total epochs}}{\text{\# of tasks}}$ epochs before training on a different task. AEs were trained once after switching to a different task based on the novelty detection mechanism. We measure average accuracy over all tasks at the end of training, i.e., $\text{Acc} = \frac{1}{N} \sum_{i=1}^{N} \text{Acc}_i$ where Acc_i is the accuracy on the ith task after model learning the final task $T^{(N)}$. We evaluated the memory footprint by measuring the trainable model parameter size. We report the mean and standard deviation of three individual experiments with random seeds, with each seed resampling the splits to avoid any favorable class split.

Model architecture and hyperparameters We employ the base size of the Vision Transformer (ViT) [3], which consists of 12 layers, 12 attention heads for each layer, a hidden dimension of 768, and an MLP dimension of 3072. We added the mixture-of-adapters architecture for each attention layer of ViT. We employed a low rank of one for both adapters and autoencoders' latent space. The hyperparameters were chosen based on a grid search. Batch sizes of 128 and 1 were employed for training and test, respectively. We utilized the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay of 0.01. We employed learning rates of 0.001 and 0.005 for updating the adapters and generative models, respectively. We employed a coefficient of 0.5 in the loss function of the knowledge distilla-

Method	CIFAR10/5	CIFAR100/10	CIFAR100/20	Average
SupSup [20]	26.2 ± 0.46	33.1 ± 0.47	12.3 ± 0.30	23.87
HyperNet [17]	47.4 ± 5.78	29.7 ± 2.19	19.4 ± 1.44	32.17
MUC [11]	53.6 ± 0.95	30.0 ± 1.37	14.4 ± 0.93	32.67
PASS [22]	47.3 ± 0.97	36.8 ± 1.64	25.3 ± 0.81	36.47
LwF.R [12]	54.7 ± 1.18	45.3 ± 0.75	44.3 ± 0.46	48.10
Mnemonicsy [12]	64.1 ± 1.47	51.0 ± 0.34	47.6 ± 0.74	54.23
DER++ [2]	66.0 ± 1.27	55.3 ± 0.10	46.6 ± 1.44	55.97
IL2A[21]	92.0 ± 0.23	60.3 ± 0.14	57.9 ± 0.31	70.07
CLOM [8]	88.0 ± 0.48	65.2 ± 0.71	58.0 ± 0.45	70.40
SSRE [23]	90.5 ± 0.61	65.0 ± 0.27	61.7 ± 0.15	72.40
FeTrIL [14]	90.9 ± 0.38	65.2 ± 0.16	61.5 ± 0.73	72.53
Ours	**99.3** ± 0.24	**86.1** ± 3.09	**79.9** ± 9.03	**88.43**

Table 1. Average accuracy in the class-incremental learning tasks.

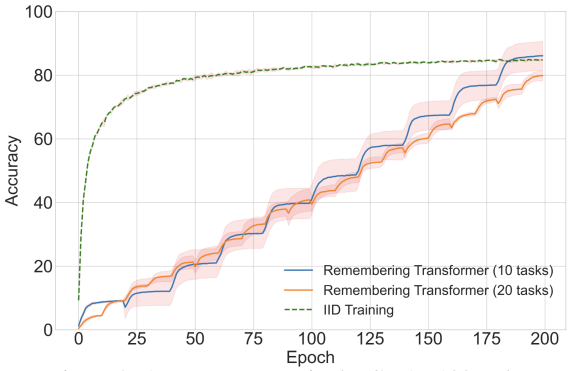


Figure 2. Accuracy curves in the CIFAR100 tasks.

tion. We fine-tuned the model for 50 epochs in CIFAR10 dataset and 200 epochs in CIFAR100 dataset. The autoencoders were trained for 10 epochs for all the datasets. All experiments were based on PyTorch and four A100 GPUs. The code would be made publicly available.

3.2. Class-incremental learning tasks

Average task accuracy We conducted a comprehensive evaluation based on the average task accuracy. We compared it to a wide range of conventional continual learning methods. The results (Table 1) revealed that Remembering Transformer significantly enhanced model performance in challenging class-incremental learning tasks without task identity information, establishing a new SOTA performance. Remembering Transformer surpassed the second-best method, FeTrIL, in vision tasks by 15.90%.

We demonstrate the accuracy curves for the class-incremental learning tasks of CIFAR100/10 and CIFAR100/20 in Figure 2. For comparison, we evaluated an oracle model using IID training, where tasks are defined as identical instances of the same dataset with all classes. The Remembering Transformer achieves competitive performance compared to the oracle model, effectively learning different tasks based on the generative routing approach. Additionally, the Remembering Transformer even outperforms the oracle method in the CIFAR100/10 task.

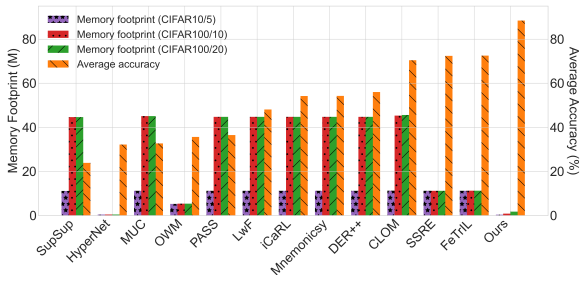


Figure 3. Memory footprint and average accuracy comparison.

Capacity (M)	Accuracy (%)
0.37 (E=5)	**99.3**
0.22 (E=3)	93.2
0.15 (E=2)	87.1

Table 2. Capacity vs accuracy.

Replay (#)	Accuracy (%)
512	**93.2**
256	90.5
128	86.6

Table 3. #Replay vs accuracy.

Memory footprint We evaluated the memory footprint by measuring the trainable model parameter size. The results in Figure 3 demonstrate that the Remembering Transformer is a parameter-efficient method for continual learning compared to a wide range of existing methods. Although OWM and HyperNet had smaller model sizes, they could not achieve competitive performance. In contrast, the Remembering Transformer achieved the new SOTA performance while retaining a relatively small memory footprint.

3.3. Varying the model capacity

The adapter fusion method leverages knowledge distillation to transfer old task knowledge to the newly learned adapter. To demonstrate whether the model can learn a sequence of new tasks with a limited capacity of the adapters, we limited the model capacity to $E = 2, 3, 5$. We then evaluated on the CIFAR10/5 task with a replay memory size of $M = 512$. Moreover, we varied the replay memory size $M = 128, 256, 512$, while employing a capacity $E = 3$. Table 3 demonstrates that Remembering Transformer can learn a number of tasks more than available adapters, retaining competitive performance. For instance, when $E = 3$ and $M = 512$, Remembering Transformer achieved a performance of 93.2% with a memory footprint of 0.22M. In contrast, FeTrIL obtained a performance of 90.9% with a much larger memory footprint of 11.18M (Table 1).

4. Conclusions

Remembering Transformer leverages the mixture-of-adapter architecture for parameter-efficient continual learning. The novelty detection method identifies task similarities and automatically routes samples to existing adapters without task identity information. Our results demonstrated the superiority of the proposed method over existing continual learning methods, even with limited model capacity.

References

[1] James Urquhart Allingham, Florian Wenzel, Zelda E. Mariet, and et al. Sparse moes meet efficient ensembles. *arXiv:2110.03360*, 2021. 1

[2] Pietro Buzzega, Matteo Boschini, Angelo Porrello, and et al. Dark experience for general continual learning: a strong, simple baseline. In *NeurIPS*, 2020. 4

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, and et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 1, 3

[4] Ruy Gómez-Ocádiz, Massimiliano Trippa, Chun-Lei Zhang, Lorenzo Posani, Simona Cocco, Rémi Monasson, and Christoph Schmidt-Hieber. A synaptic signal for novelty processing in the hippocampus. *Nature Communications*, 13 (1):4122, 2022. 1

[5] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *arXiv:1503.02531*, 2015. 3

[6] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 2

[7] Alex Kafkas and Daniela Montaldi. How do memory systems detect and respond to novelty? *Neuroscience letters*, 680:60–68, 2018. 1

[8] Gyuhak Kim, Sepideh Esmaeilpour, Changnan Xiao, and Bing Liu. Continual learning based on OOD detection and task masking. In *CVPR Workshops*, pages 3855–3865. IEEE, 2022. 4

[9] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. 3

[10] Dharshan Kumaran, Demis Hassabis, and James L McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences*, 20(7):512–534, 2016. 1

[11] Yu Liu, Sarah Parisot, Gregory G. Slabaugh, and et al. More classifiers, less forgetting: A generic multi-classifier paradigm for incremental learning. In *ECCV (26)*, pages 699–716. Springer, 2020. 4

[12] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *CVPR*, pages 12242–12251. Computer Vision Foundation / IEEE, 2020. 4

[13] James L McGaugh. Memory–a century of consolidation. *Science*, 287(5451):248–251, 2000. 1

[14] Grégoire Petit, Adrian Popescu, Hugo Schindler, and et al. Fetril: Feature translation for exemplar-free class-incremental learning. In *WACV*, pages 3900–3909. IEEE, 2023. 4

[15] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, pages 5533–5542. IEEE Computer Society, 2017. 3

[16] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, and et al. Scaling vision with sparse mixture of experts. In *NeurIPS*, 2021. 1

[17] Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F. Grewe. Continual learning with hypernetworks. In *ICLR*, 2020. 4

[18] Tianchun Wang, Wei Cheng, Dongsheng Luo, and et al. Personalized federated learning via heterogeneous modular networks. In *ICDM*, 2022. 1

[19] Max Welling. Herding dynamical weights to learn. In *ICML*, pages 1121–1128. ACM, 2009. 3

[20] Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, and et al. Supermasks in superposition. In *NeurIPS*, 2020. 4

[21] Fei Zhu, Zhen Cheng, Xu-Yao Zhang, and Chenglin Liu. Class-incremental learning via dual augmentation. In *NeurIPS*, pages 14306–14318, 2021. 4

[22] Fei Zhu, Xu-Yao Zhang, Chuang Wang, and et al. Prototype augmentation and self-supervision for incremental learning. In *CVPR*, pages 5871–5880. Computer Vision Foundation / IEEE, 2021. 4

[23] Kai Zhu, Wei Zhai, Yang Cao, and et al. Self-sustaining representation expansion for non-exemplar class-incremental learning. In *CVPR*, pages 9286–9295. IEEE, 2022. 4