Saliency-Guided Hidden Associative Replay for Continual Learning

Guangji Bai
Department of Computer Science
Emory University
guangji.bai@emory.edu

Qilong Zhao
Department of Computer Science
Emory University
qzhao31@emory.edu

Xiaoyang Jiang
Department of Computer Science
Emory University
jxxxxxygoat@gmail.com

Liang Zhao[*]
Department of Computer Science
Emory University
liang.zhao@emory.edu

Abstract

Continual Learning (CL) is a burgeoning domain in next-generation AI, focusing on training neural networks over a sequence of tasks akin to human learning. Amongst various strategies, replay-based methods have emerged as preeminent, echoing biological memory mechanisms. However, these methods are memory-intensive, often preserving entire data samples—an approach inconsistent with humans' selective memory retention of salient experiences. While some recent works have explored the storage of only significant portions of data in episodic memory, the inherent nature of partial data necessitates innovative retrieval mechanisms. Addressing these nuances, this paper presents the **S**aliency-Guided **H**idden **A**ssociative **R**eplay for **C**ontinual Learning (**SHARC**). This novel framework synergizes associative memory with replay-based strategies. SHARC primarily archives salient data segments via sparse memory encoding. Importantly, by harnessing associative memory paradigms, it introduces a content-focused memory retrieval mechanism, promising swift and near-perfect recall, bringing CL a step closer to authentic human memory processes. Extensive experimental results demonstrate the effectiveness of our proposed method for various continual learning tasks. [2]

1 Introduction

Continual learning (CL) represents a vital advancement for next-generation AI [17]. While traditional supervised learning is well-established, CL remains in its nascent stages. The main challenge is to prevent *Catastrophic Forgetting* [16] as agents acquire new tasks, ensuring they retain earlier knowledge. In order to address this problem, researchers have put forward several strategies. *Replay-based methods* [20, 6, 2], which utilizes a small memory to store previous data and reuse them when learning new tasks, have emerged as a particularly effective solution. However, a potential bottleneck of this approach is its memory-intensive nature, as entire data samples are conserved. The vast storage requirements of replay-based methods and their divergence from natural memory processes necessitate exploration into more efficient and human-like strategies for continual learning.

While there are pioneering works [23, 3] in replay-based CL that explored the idea of storing only the salient or partial aspects of data into episodic memory, challenges arise due to the inherent nature of partial data. Since these fragments are not directly usable as model input, an effective retrieval technique becomes indispensable. Inspired by that the human brain, especially the hippocampus,

[*]Corresponding author.
[2]Code available at `https://github.com/BaiTheBest/SHARC`.

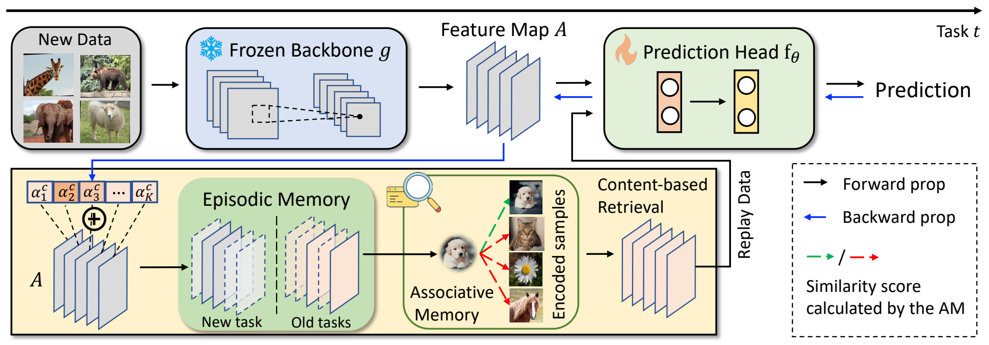


Figure 1: Overview of our proposed SHARC framework (best viewed in color).

employs associative recall for *content-based* memory retrieval [10, 19], achieving a remarkable recall accuracy close to perfection. As such, for systems aiming to emulate human-like continual learning, there is an evident inspiration to design techniques that mirror the associative and content-based retrieval processes inherent in human cognition.

To address the aforementioned challenges, this paper introduces the **S**aliency-Guided **H**idden **A**ssociative **R**eplay for **C**ontinual Learning (**SHARC**), marking the inception of a Continual Learning framework that seamlessly integrates associative memory into replay-based techniques. SHARC distinguishes itself from existing replay-based CL methodologies in two pivotal aspects: First, rather than archiving complete samples within episodic memory, SHARC conserves only the most salient segments through sparse memory encoding. More crucially, drawing inspiration from the principles of associative memory, we have crafted a content-centric memory retrieval module that boasts swift and impeccable recall capabilities. Our **contribution** includes 1). We develop a novel neural-inspired replay-based continual learning framework to handle catastrophic forgetting. 2). We propose to leverage associative memory for efficient memory storage and recovery. 3). We demonstrate our model's efficacy and superiority with extensive experiments.

2 Proposed Method

2.1 Saliency-Guided Memory Encoding with Structured Sparsity

Different from many earlier works that store raw images for replay, we consider first encoding raw data into high-level representations and storing them. Formally, our model $\mathbf{y} = f_\theta(g(\mathbf{x}))$ is composed of a pre-trained backbone g and a trainable prediction head f_θ. The output of $g(\mathbf{x})$ is a tensor $A \in \mathbb{R}^{H\times W\times K}$, where H, W are the dimension of the feature map and K is the number of channels. To achieve sparse representation, we consider saliency-based methods [25] that measure the importance of the neurons by their first-order gradients. Specifically, denote $\boldsymbol{\alpha}^c = [\alpha_1^c, \alpha_2^c, \cdots, \alpha_K^c]$, where α_i^c is the saliency score over channel i, the masked feature map A' is

$$A' = \mathrm{TR}_{H,W,K}\left(\mathbb{1}\{|\boldsymbol{\alpha}^c| > Q_\mu\} \otimes \mathbf{J}_{H,W}\right) \odot A, \tag{1}$$

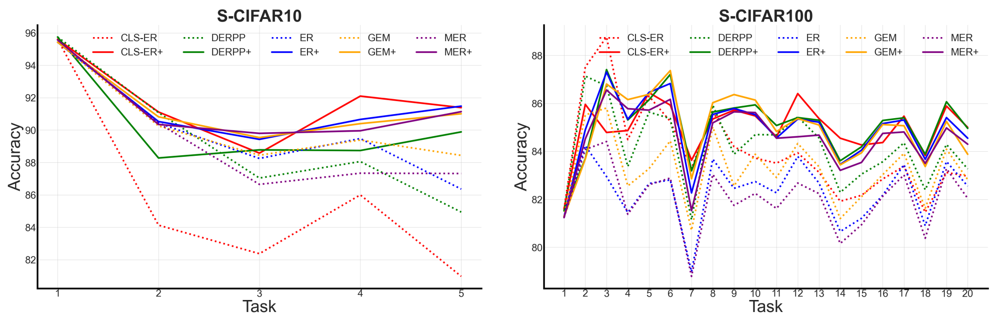
where $\mathbb{1}\{\cdot\}$ is the indicator function, Q_μ is the threshold for masking out the bottom μ quantile of channels. $\mathbf{J}$ denotes the all-one matrix and $\mathrm{TR}(\cdot)$ denotes tensor reshaping operation.

Our design offers a notable advantage of *structured* sparsity. This characteristic is hardware-friendly and immediately translates into reduced memory costs, requiring no additional system-level optimizations. By simply discarding channels with saliency scores below a specified threshold, the remaining feature maps maintain a consistent tensor shape.

2.2 Associative Memory Retrieval for Replay

How to leverage associative memory in the continual learning setting is under-explored. Content-based retrieval and noise tolerance of the associative memory allow us to increase the sparsity of the masked feature maps and achieve maximal memory saving. The fast recall reduces the computational overhead for memory retrieval hence our method can be applied to various replay-based baselines.

Formally, an AM $\mathcal{A}(\mathbf{x}, \boldsymbol{\omega})$ can be implemented as a recurrent or feed-forward neural network [10, 19, 26], where $\mathbf{x}$ and $\boldsymbol{\omega}$ denote the input and model parameters of the associative memory, respectively. The *read* and *write* operations of an associative memory are implemented based on an *energy function*.



(a) Task-IL S-CIFAR10 (b) Task-IL S-CIFAR100

Figure 2: Learning curves of multiple models with/without SHARC on S-CIFAR10 and S-CIFAR100 in Task-IL scenario. Models with/without SHARC are shown in solid/dotted lines.

For example, [26] considers the sum of prediction errors across all network layers:

$$E\big(\mathbf{x}_{0:L}, \boldsymbol{\omega}_{0:L}\big) = \|\mathbf{x}_L - \boldsymbol{\omega}_L\|_2^2 + \lambda \sum\nolimits_{\ell=0}^{L-1} \|\mathbf{x}_\ell - \mathcal{A}_\ell(\mathbf{x}_{\ell+1}, \boldsymbol{\omega}_\ell)\|_2^2. \tag{2}$$

2.3 Training Pipeline

Our proposed method involves training the classifier f_θ and the associative memory $\mathcal{A}$.

Training classifier. In each incremental, we update the model parameters of the continual learning classifier f_θ by using the new coming data and the replay samples. The key here is that we use associative memory to retrieve the "complete" feature map and then feed it to the classifier for memory replay. Formally, the training objective of the classifier can be formulated as follows

$$\begin{aligned} &\theta^* = \operatorname{argmin}_\theta \sum\nolimits_{(\mathbf{x},t,\mathbf{y})} \ell\big(f_{\boldsymbol{\theta}}(g(\mathbf{x}), t), \mathbf{y}\big) + \sum\nolimits_{k<t} \ell\big(f_{\boldsymbol{\theta}}, \tilde{\mathcal{M}}_k\big) \\ &\text{where} \quad \tilde{\mathcal{M}}_k = \operatorname{argmin}_{\mathbf{x}}\, E\big(\mathbf{x}_{0:L}, \boldsymbol{\omega}_{0:L}\big) \text{ with } \mathbf{x}_0 \text{ initialized as } A'_k \in \mathcal{M}_k, \end{aligned} \tag{3}$$

where the first row is the continual learning objective. $\tilde{\mathcal{M}}_k$ denotes the retrieved episodic memory, i.e., the feature maps recalled by associative memory.

Training associative memory. Given feature maps coming from new tasks in each incremental, we need to write those feature maps into the associative memory such that we can ask it to retrieve the complete feature map given a partial cue later. Formally, given feature maps A_t from new task t, writing them into associative memory corresponds to solving the following optimization problem

$$\boldsymbol{\omega} = \operatorname{argmin}_{\boldsymbol{\omega}}\, E\big(\mathbf{x}_{0:L}, \boldsymbol{\omega}_{0:L}\big) \text{ with } \mathbf{x}_0 \text{ fixed as } A_t, \tag{4}$$

where we minimize Eq. 2 w.r.t. parameter $\boldsymbol{\omega}$ while keeping the input $\mathbf{x} = A$.

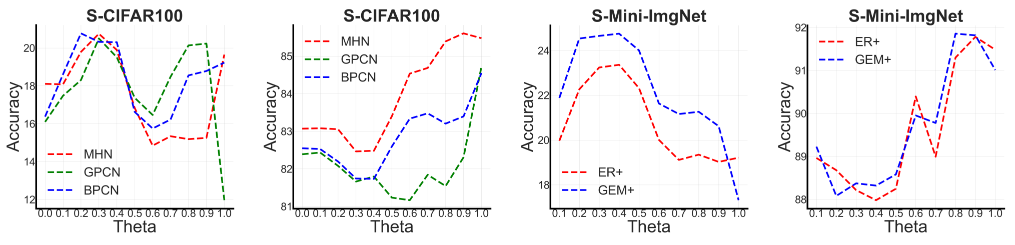
3 Experiment

In this section, we evaluate our proposed method SHARC for CL tasks. Detailed experiment settings can be found in the appendix due to the limited space.

Performance Comparison. Table 1 compares six replay-based methods before and after combining them with SHARC in the Task-IL scenario. Overall, the methods used in conjunction with SHARC offer significant improvements in most cases. Such contrast exists in all settings (different datasets, models, and buffer sizes). In particular, CLS-ER [2] equipped with SHARC achieves a 12.9% improvement in ACC on S-CIFAR10 with buffer size 200. From a methodological perspective, rehearsal-based methods (e.g., ER [6]) offer greater improvements than constraint-based methods (e.g., GEM [15]). Rehearsal-based methods can benefit more from masking because masking reduces the memory space for samples, allowing more previous samples to be reviewed. Furthermore, in most cases on S-CIFAR100 and S-MiniImgNet, the BWT increases or even becomes positive when using SHARC, indicating that SHARC is highly resistant to forgetting. As the buffer size decreases, the complexity of the task increases. Achieving good performance with smaller buffer sizes is the spirit of continual learning. Based on this consideration, we further investigate the learning curve for a minimum buffer size of 200. As shown in Figure 2, methods equipped with SHARC clearly prevail in the figure, indicating that they have been steadily improved during the learning process.

Table 1: **Performance comparison on image classification datasets (Task-IL).** $^+$ denotes the corresponding method combined with our SHARC framework.

Buffer	Model	S-CIFAR-10		S-CIFAR-100		S-Mini-ImgNet	
		ACC (↑)	BWT (↑)	ACC (↑)	BWT (↑)	ACC (↑)	BWT (↑)
-	JOINT	93.49 ± 0.61	43.14 ± 2.07	87.57 ± 0.89	67.99 ± 1.53	74.95 ± 0.7	70.02 ± 0.81
-	SGD	92.31 ± 0.54	-0.38 ± 0.82	85.83 ± 0.35	3.08 ± 2.14	76.2 ± 0.41	3.98 ± 0.75
200	GEM	88.44 ± 1.11	-4.6 ± 2.24	82.82 ± 0.62	0.2 ± 1.69	72.23 ± 1.26	-1 ± 1.82
	GEM$^+$	91.01 ± 1.04	-1.5 ± 1.31	83.88 ± 0.52	-0.04 ± 1.2	76.13 ± 0.98	3.49 ± 1.47
	A-GEM	90.52 ± 3.29	-1.07 ± 1.57	85.33 ± 0.58	2 ± 1	75.18 ± 1.11	2.33 ± 1.5
	A-GEM$^+$	91.72 ± 1.02	-0.35 ± 2.08	85.55 ± 0.88	1.25 ± 0.63	76.54 ± 0.97	4.14 ± 1.71
	ER	86.36 ± 1.33	-5.04 ± 1.72	82.55 ± 0.47	-0.71 ± 1.59	71.66 ± 1.44	-1.53 ± 2.04
	ER$^+$	91.48 ± 1.18	-0.93 ± 1.54	84.55 ± 0.62	0.71 ± 0.61	73.68 ± 0.59	0.71 ± 0.97
	MER	87.32 ± 1.39	-2.3 ± 3.83	82.04 ± 0.63	-0.83 ± 1.49	71.2 ± 1.43	-1.99 ± 1.53
	MER$^+$	91.14 ± 1.62	-0.9 ± 2.46	84.3 ± 0.92	0.41 ± 1.26	73.54 ± 0.58	0.7 ± 1.06
	DER++	84.94 ± 1.95	-6.45 ± 1.91	83.27 ± 0.76	0.32 ± 1.47	72.92 ± 1.09	-0.13 ± 1.44
	DER++$^+$	89.89 ± 1.34	-2.72 ± 2.04	84.96 ± 0.97	0.62 ± 1.44	74.59 ± 0.87	2 ± 1.27
	CLS-ER	80.97 ± 2.11	-12.6 ± 4.39	82.97 ± 0.32	-1.95 ± 1.5	73.67 ± 1.05	-1.75 ± 0.4
	CLS-ER$^+$	91.39 ± 0.7	-0.94 ± 1.18	85 ± 0.41	1.27 ± 0.68	77 ± 0.45	2.7 ± 0.97
500	GEM	88.02 ± 2.61	-3.84 ± 1.19	82.81 ± 0.66	0.06 ± 1.66	73.6 ± 1.13	0.23 ± 1.27
	GEM$^+$	91.53 ± 1.17	-0.05 ± 1.58	84.37 ± 1.03	1.63 ± 0.79	75.56 ± 0.93	3.2 ± 1.61
	A-GEM	90.81 ± 2.97	0.31 ± 2.81	85.44 ± 0.28	2.32 ± 0.84	75.59 ± 1.15	2.78 ± 1.61
	A-GEM$^+$	92.32 ± 0.67	0.22 ± 0.7	85.89 ± 0.82	3.25 ± 1.01	75.65 ± 0.86	3.21 ± 1.56
	ER	88.05 ± 1.51	-1.88 ± 3.62	82.7 ± 0.63	-0.09 ± 0.5	71.83 ± 1.17	-1.36 ± 1.5
	ER$^+$	91.64 ± 0.66	-0.82 ± 0.67	84.77 ± 1.57	1.85 ± 1.72	72.94 ± 0.63	0.41 ± 1.08
	MER	88.33 ± 1.87	-3.35 ± 1.6	82.11 ± 0.5	-0.36 ± 0.91	70.69 ± 1.07	-2.28 ± 1.41
	MER$^+$	91.69 ± 0.73	-0.39 ± 1.56	84.06 ± 1.33	1.43 ± 1.46	72.63 ± 0.37	-0.28 ± 1.08
	DER++	86.73 ± 2.77	-4.79 ± 1.26	83.04 ± 0.58	0.76 ± 1.64	72.05 ± 0.87	-0.95 ± 1.52
	DER++$^+$	90.46 ± 1.3	-2.71 ± 1.01	85.13 ± 1.57	1.81 ± 1.01	73.85 ± 0.79	1.5 ± 1.5
	CLS-ER	82.54 ± 3.06	-8.64 ± 5.52	81.34 ± 0.9	-2.27 ± 1.8	72.11 ± 0.38	-3.41 ± 0.88
	CLS-ER$^+$	90.94 ± 1.49	-2.22 ± 1.51	85.36 ± 0.83	1.82 ± 1.66	76.27 ± 0.52	2.05 ± 0.82



(a) Class-IL AM (b) Task-IL AM (c) Class-IL SA (d) Task-IL SA

Figure 3: (a) & (b): comparison of different associative memories. (c) & (d): sensitivity analyses of the masking threshold.

Associative Memory Comparison. As shown in Figures 3(a) and 3(b), we found that Modern Hopfield Network (MHN) [19] favors more towards the task-incremental setting while BayesPCN [26] favors more towards class-incremental setting. This is potentially due to that, in BayesPCN a forgetting mechanism is implemented, which can help mitigate the memory overload of the associative memory when too many samples to memorize.

Masking Threshold Sensitivity Analysis. We conduct sensitivity analysis on the threshold Q_μ in Eq 1. Masking threshold Q_μ is defined as a certain percentile value and feature maps with importance below the threshold will be masked. As shown in Figure 3(d), the optimal Theta in Task-IL is between 0.8 to 0.9. As shown in Figure 3(c), the optimal Theta in Class-IL is between 0.3 to 0.5. This indicates that the optimal thresholds show different trends in Task-IL and Class-IL.

4 Conclusion

We propose SHARC, a novel framework that bridges the gap between current AI models and humans in continual learning. Combining associative memory and interpretive techniques, SHARC enables efficient, near-perfect recall of seen samples in a human-like manner. As a generic framework, SHARC can be seamlessly adapted to any replay-based approach, thus improving their performance in different continual learning scenarios. We demonstrate the effectiveness of our framework with abundant experimental results. Our proposed SHARC framework consistently improves several SOTA replay-based methods on multiple benchmark datasets.

References

[1] D. J. Amit and D. J. Amit. *Modeling brain function: The world of attractor neural networks*. Cambridge university press, 1989.

[2] E. Arani, F. Sarfraz, and B. Zonooz. Learning fast, learning slow: A general continual learning method based on complementary learning system. *arXiv preprint arXiv:2201.12604*, 2022.

[3] G. Bai, C. Ling, Y. Gao, and L. Zhao. Saliency-augmented memory completion for continual learning. In *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*, pages 244–252. SIAM, 2023.

[4] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

[5] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

[6] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. Torr, and M. Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[8] T. L. Hayes, K. Kafle, R. Shrestha, M. Acharya, and C. Kanan. Remind your neural network to prevent catastrophic forgetting. In *European Conference on Computer Vision*, pages 466–483. Springer, 2020.

[9] D. Hebb. The organization of behavior. *New York*, 1949.

[10] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[11] Y. Huang and R. P. Rao. Predictive coding. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2(5):580–593, 2011.

[12] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[13] D. Krotov and J. Hopfield. Large associative memory problem in neurobiology and machine learning. *arXiv preprint arXiv:2008.06996*, 2020.

[14] Z. Li and D. Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[15] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30:6467–6476, 2017.

[16] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[17] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

[18] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin. Variational autoencoder for deep learning of images, labels and captions. *Advances in neural information processing systems*, 29:2352–2360, 2016.

[19] H. Ramsauer, B. Schäfl, J. Lehner, P. Seidl, M. Widrich, T. Adler, L. Gruber, M. Holzleitner, M. Pavlović, G. K. Sandve, et al. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*, 2020.

[20] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[21] M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.

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

[23] G. Saha and K. Roy. Saliency guided experience packing for replay in continual learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5273–5283, 2023.

[24] T. Salvatori, Y. Song, Y. Hong, L. Sha, S. Frieder, Z. Xu, R. Bogacz, and T. Lukasiewicz. Associative memories via predictive coding. *Advances in Neural Information Processing Systems*, 34:3874–3886, 2021.

[25] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[26] J. Yoo and F. Wood. Bayespcn: A continually learnable predictive coding associative memory. *Advances in Neural Information Processing Systems*, 35:29903–29914, 2022.

[27] W. Zhou, S. Chang, N. Sosa, H. Hamann, and D. Cox. Lifelong object detection. *arXiv preprint arXiv:2009.01129*, 2020.

A Appendix

A.1 Experimental Details

A.1.1 Dataset Details

We expand upon the datasets used for our experiments in this section. We highlighted the sentence that describes the domain drift within each dataset.

- **CIFAR-10:** The CIFAR-10 dataset is a comprehensive collection of $60,000$ 32×32 color images divided into 10 distinct classes, with $6,000$ images per class. This dataset is further split into $50,000$ training images and $10,000$ test images, allowing for effective model evaluation.
- **CIFAR-100:** The dataset is similar to CIFAR-10 and is composed of 100 classes, each containing 600 images. Specifically, within the 600 images, there are 500 images used for training and 100 images designated for testing purposes. It is important to note that the 100 classes are actually comprised of 20 classes, where each class is further divided into 5 sub-classes. Therefore, the total count of 100 classes is obtained by multiplying 5 and 20 $(5 \times 20 = 100)$.
- **Mini-ImageNet:** The Mini-ImageNet dataset contains 100 classes with a total of $60,000$ color images. Each class has 600 samples, and the size of each image is 84×84 pixels. Typically, the class distribution between the training and testing sets of this dataset is $80 : 20$. Compared to the CIFAR-10 dataset, the Mini-ImageNet dataset is more complex but is better suited for prototype design and experimental research. [6]

A.1.2 Details of Comparison Methods

In this paper, we compare our proposed SHARC with several SOTA replay-based methods as well as regularization-based methods. Specifically,

- **ER,** a rehearsal-based method that utilizes the average of parameter update gradients from the current task's samples alongside samples from episodic memory to update the learning agent. This method, known as ER (Episodic Regularization), offers a computationally efficient alternative to GEM (Gradient Episodic Memory) and has demonstrated successful performance when dealing with a limited memory buffer.
- **MER,** a rehearsal-based model that harnesses the power of an episodic memory. MER employs a unique loss function that approximates the dot products of the gradients of current and previous tasks, thereby mitigating the issue of forgetting. To ensure a fair and comprehensive comparison with other methods, we adjust the experimental setting by setting the number of inner gradient steps to 1 for each outer meta-update, while maintaining a mini-batch size of 10. This adjustment allows us to establish a more consistent framework for evaluating the performance of MER alongside other approaches, specifically in terms of the number of stochastic gradient descent (SGD) updates. By presenting these findings, we aim to shed light on the effectiveness and practicality of MER as a rehearsal-based model in the context of meta-learning [21].
- **GEM,** who utilizes an episodic memory buffer to store past experiences and gradients. By incorporating both the current task's gradient and the gradients of previous tasks from the episodic memory, GEM ensures that valuable information from prior tasks is retained while accommodating new learning. To prevent catastrophic forgetting, the algorithm employs a constrained optimization approach, projecting the current gradient onto a subspace that preserves knowledge from previous tasks [15].
- **A-GEM,** takes a step further than GEM by incorporating an adaptive mechanism that updates the model's parameters based on both the current task's gradient and the gradients of previous tasks stored in the episodic memory. This allows AGEM to effectively preserve knowledge from prior tasks while adapting to new tasks.
- **CLS-ER,** an innovative algorithm that utilizes a dual-memory learning mechanism to enhance performance in continual learning tasks. In this approach, the episodic memory serves as a repository for storing samples encountered during the learning process. On the other hand, semantic memories play a crucial role in constructing short-term and long-term memories of the learned representations from the working model [2].
- **DER++,** a combination of rehearsal, knowledge distillation, and regularization techniques. This approach leverages the network's logits sampled at different stages of the optimization trajectory. This approach promotes consistency with the network's past experiences [4].

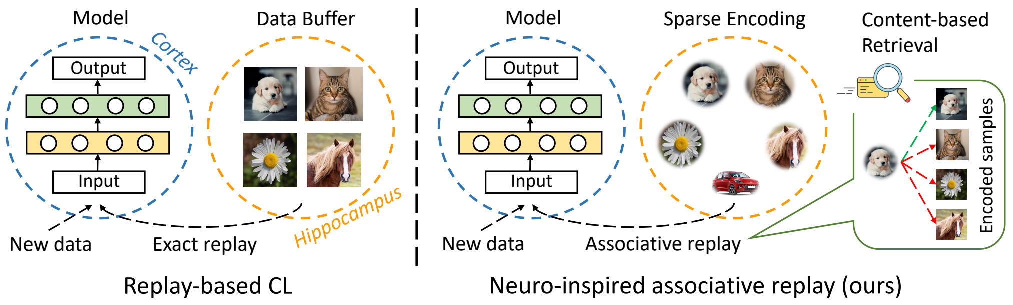


Figure 4: Motivation and overview of our proposed framework. **Left:** Typical replay-based approaches maintain and replay a small episodic memory of previous samples, which is inspired by the cortex and hippocampus in the human brain. **Right:** Our memory buffer is equipped with a forgetting mechanism to drop uninformative episodes, and a content-addressable associative memory is used to achieve fast and high-accuracy data retrieval.

A.1.3 Hyper-Parameter Setting

All experiments are conducted on a 64-bit machine with an NVIDIA T4 Tensor Core GPU which has 320 Turing Tensor cores, 2560 CUDA cores, 16GB memory, and Intel® Xeon® Platinum 8259CL CPU @ 2.50GHz. The learning rate for all datasets is uniformly set to be 0.1. Down below we report the hyper-parameter unique to some models applied in our experiment.

- **ER:**
  'lr': 0.1
- **MER:**
  'lr': 0.1, 'gamma': 0.5, 'batch num': 1
- **GEM:**
  'lr': 0.1, 'gamma': 0.5
- **AGEM:**
  'lr': 0.1
- **DER++:**
  'lr': 0.1, 'alpha': 0.1, 'beta': 0.5
- **CLSER:**
  'lr': 0.1, 'reg weight': 0.15, 'stable model update freq': 0.1, 'stable model alpha': 0.999, 'plastic model update freq': 0.3, 'plastic model alpha': 0.999

A.2 Preliminaries

In this section, we provide more discussion about some preliminaries in this paper.

A.2.1 Associative Memory

When applied to Computer Science problems, associative memories come in two high-level forms: *auto-associative* and *hetero-associative* memories. While both are able to recall patterns given a set of inputs, auto-associative memories are primarily focused on recalling a pattern X when provided a partial or noisy variant of X. By contrast, hetero-associative memories are able to recall not only patterns of different sizes from their inputs but can be leveraged to map concepts between categories (hence "hetero-associative"). One common example from the literature is a hetero-associative memory that might recall the embedded animal concept of "monkey" given the embedded food concept of "banana". Since all forms of AM are focused on the actual content being stored and retrieved, they are also commonly referred to as *content-addressable memories* (CAM) in the literature.

Classical Hopfield Network. One of the earliest and probably the most well-known auto-associative memory are Hopfield Networks [10]. The original Hopfield Networks are discrete where they operate by storing binary-pattern inputs into the weights of a fully-connected neural network using a local update rule. For an input $\mathbf{x} \in \{-1, 1\}^d$ containing d binary values, a Hopfield Network contains d^2 real-valued connections, i.e., $\mathbf{W} \in \mathbb{R}^{d\times d}$. The Hopfield learning algorithm specifies a write over n

binary memories $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$, represented as column vectors, by accumulating their outer product

$$\mathbf{W} = \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^{\intercal}, \quad \mathbf{W}[p,q] = \sum_{i=1}^{n} \mathbf{x}_i[p] \mathbf{x}_i^{\intercal}[q]. \tag{5}$$

This specific write update is termed the *Hebbian update rule* as it follows the "fire together, wire together" principle proposed by psychologist Donald Hebb as a model of synaptic learning [9]. The Hopfield Network *read* operation involves minimizing an *energy function*

$$E(\mathbf{W}, \boldsymbol{\xi}) = -\frac{1}{2} \boldsymbol{\xi}^{\intercal} \mathbf{W} \boldsymbol{\xi}, \tag{6}$$

where $\boldsymbol{\xi} \in \mathbb{R}^d$ is the *state* of the network, initialized as the initial query, and then optimized to a stable state known as the *attractor*.

Modern Hopfield Network. Hopfield Networks serve as an interesting weight-based method of memory storage. However, although they use optimization as a method of memory retrieval, their learning rule is not differentiable due to the use of discrete states. Modern Hopfield Network (MHN [19]) introduces a new energy function instead of that in Eq. 6. Specifically, MHN generalizes the energy function to continuous-valued patterns and adds a quadratic term, i.e.,

$$E(\mathbf{X}, \boldsymbol{\xi}, \beta) = -\mathrm{LSE}(\beta, \mathbf{X}^{\intercal} \boldsymbol{\xi}) + \frac{1}{2} \boldsymbol{\xi}^{\intercal} \boldsymbol{\xi} + \beta^{-1} \log(N) + \frac{1}{2} M^2, \tag{7}$$

where $\mathbf{X}$ is the matrix form of N *continuous* stored patterns $\mathbf{x}_i$, $i = 1, 2, \cdots, N$, M is the largest norm of all stored patterns, LSE stands for the LogSumExp function with coefficient β.

Predictive Coding Network. The Predictive Coding Network is a computational model that aims to explain how the brain processes sensory information and makes predictions about future sensory inputs. It is based on the concept of predictive coding, which suggests that the brain constantly generates predictions about upcoming sensory inputs and updates these predictions based on the actual sensory feedback it receives. In machine learning, the predictive coding network is implemented as an energy-based associative memory model that has set the state-of-the-art on a number of image associative recall tasks.

A.3 Related Work

Continual Learning (CL). Catastrophic forgetting is a long-standing problem [22] in continual learning which has been recently tackled in a variety of visual tasks such as image classification [12, 20], object detection [27], etc.

Existing techniques in CL can be divided into three main categories [17]: 1) *regularization-based approaches*, 2) *dynamic architectures* and 3) *replay-based approaches*. Regularization-based approaches alleviate catastrophic forgetting by either adding a regularization term to the objective function [12] or knowledge distillation over previous tasks [14]. Dynamic architecture approaches adaptively accommodate the network architecture (e.g., adding more neurons or layers) in response to new information during training. Dynamic architectures can be explicit if new network branches are grown, or implicit, if some network parameters are only available for certain tasks. Replay-based approaches alleviate the forgetting of deep neural networks by replaying stored samples from the previous history when learning new ones and have been shown to be the most effective method for mitigating catastrophic forgetting.

Replay-based CL. Replay-based methods mainly include three directions: rehearsal methods, constrained optimization, and pseudo rehearsal. *Rehearsal methods* directly retrieve previous samples from a limited size memory together with new samples for training [6, 8, 2]. While simple in nature, this approach is prone to overfitting the old samples from the memory. As an alternative, *constrained optimization* methods formulate backward/forward transfer as constraints in the objective function. GEM [15] constrains new task updates to not interfere with previous tasks by projecting the estimated gradient on the feasible region outlined by previous task gradients through first-order Taylor series approximation. A-GEM [5] further extended GEM and made the constraint computationally more efficient. Finally, *pseudo-rehearsal* methods typically utilize generative models such as GAN [7] or VAE [18] to generate previous samples from random inputs and have shown the ability to generate high-quality images recently [22]. Readers may refer to [17] for a more comprehensive survey on continual learning.

Associative Memory (AM). In general, the attractor-based mechanism [1] is typically used for the implementation of AMs, which are models that store and recall patterns. Pattern recall (associative recall) is a process whereby an associative memory, upon receiving a potentially corrupted memory query, retrieves the associated value from memory. One of the earliest and probably the most well-known associative memory are Hopfield Networks [10]. Hopfield networks are a class of recurrent artificial neural networks that have gained prominence for their ability to model associative memory and pattern recognition. The modern Hopfield network refers to an updated version of the original Hopfield network [19, 13]. The modern Hopfield network incorporates enhancements and modifications to improve its performance and overcome some limitations of the original model. More recently, predictive coding networks [11] have provided a new perspective for the design of AM, and such works [24, 26] have shown strong performance on recall tasks.

A.4 Problem Formulation

We consider supervised continual learning in this paper. Following the learning protocol in [5], we consider a training set $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_T\}$ consisting of T tasks, where $\mathcal{D}_t = \{(\mathbf{x}_i^{(t)}, \mathbf{y}_i^{(t)})\}_{i=1}^{n_t}$ contains n_t input-target pairs $(\mathbf{x}_i^{(t)}, \mathbf{y}_i^{(t)}) \in \mathcal{X} \times \mathcal{Y}$. While each learning task arrives sequentially, we make the assumption of *locally i.i.d*, i.e., $\forall\ t, (\mathbf{x}_i^{(t)}, \mathbf{y}_i^{(t)}) \overset{iid}{\sim} P_t$, where P_t denotes the data distribution for task t and $i.i.d$ for *independent and identically distributed*. Given such a stream of tasks, the goal is to train a learning agent $f_{\boldsymbol{\theta}} : \mathcal{X} \rightarrow \mathcal{Y}$, parameterized by $\boldsymbol{\theta}$, which can be queried *at any time* to predict the target $\mathbf{y}$ given associated unseen input $\mathbf{x}$ and task id t. Moreover, we require that such a learning agent can only store a small amount of seen samples in an episodic memory $\mathcal{M}$ with a fixed budget. Given predictor $f_{\boldsymbol{\theta}}$, the loss on the episodic memory of task k is defined as

$$\ell(f_{\boldsymbol{\theta}}, \mathcal{M}_k) := |\mathcal{M}_k|^{-1} \sum\nolimits_{(\mathbf{x}_i, k, \mathbf{y}_i)} \phi(f_{\boldsymbol{\theta}}(\mathbf{x_i}, k), \mathbf{y}_i),\ \forall\ k < t, \tag{8}$$

where ϕ can be *e.g.* cross-entropy or MSE. In general, a large body of *replay-based* continual learning methods seeks to optimize for the following loss function at t-th task

$$\min\nolimits_{\boldsymbol{\theta}}\ \mathcal{L}_{CL}(\boldsymbol{\theta}),\ \text{ where } \mathcal{L}_{CL}(\boldsymbol{\theta}) = \sum\nolimits_{(\mathbf{x}, t, \mathbf{y})} \ell(f_{\boldsymbol{\theta}}(\mathbf{x}, t), \mathbf{y}) + \sum\nolimits_{k<t} \ell(f_{\boldsymbol{\theta}}, \mathcal{M}_k), \tag{9}$$

which is an aggregation of the losses on the current task and replay data. After the training of task t, a subset of training samples will be stored in the episodic memory, i.e., $\mathcal{M} = \mathcal{M} \cup \{(\mathbf{x}_i^{(t)}, \mathbf{y}_i^{(t)})\}_{i=1}^{m_t}$, where m_t is the memory buffer size for the current task.

A.5 Pseudo-Code of SHARC

Here, we provide the pseudo-code of our proposed method SHARC. Specifically, the training procedure is summarized in Algorithm 1.

A.6 Additional Experimental Results

Table 2: **Performance comparison on image classification datasets (Task-IL).** The mean and standard deviation are calculated based on five runs with varying seeds. $^+$ denotes the corresponding method combined with our SHARC framework.

Buffer	Model	S-CIFAR-10		S-CIFAR-100		S-Mini-ImgNet	
		ACC (↑)	BWT (↑)	ACC (↑)	BWT (↑)	ACC (↑)	BWT (↑)
-	JOINT	72.85 ± 2.18	61.26 ± 8.55	45.87 ± 1.22	45.55 ± 1.45	47.08 ± 0.77	46.25 ± 0.98
-	SGD	20.47 ± 0.78	-90.16 ± 0.92	8.55 ± 1.39	-78.24 ± 0.93	12.21 ± 0.75	-67.11 ±0.77
1000	GEM	89.07 ± 1.09	-3.28 ± 1.61	82.38 ± 0.93	2.59 ± 1.08	73.78 ± 1	0.38 ± 1.53
	GEM$^+$	92.14 ± 0.39	-0.66 ± 1.93	85.56 ± 0.76	1.97 ± 1.8	75.11 ± 0.59	3.69 ± 1.79
	A-GEM	91.01 ± 2.86	-0.46 ± 1.09	85.7 ± 0.45	2.53 ± 1.43	75.86 ± 0.92	3.02 ± 1.38
	A-GEM$^+$	92.13 ± 0.49	-0.44 ± 2.05	86.67 ± 0.87	3.46 ± 1.05	75.42 ± 0.88	4.02 ± 1.74
	ER	89.56 ± 1.09	-1.33 ± 3.47	83.58 ± 1.12	1.39 ± 1.17	71.45 ± 1.01	-1.71 ± 1.37
	ER$^+$	91.81 ± 0.25	-0.83 ± 1.82	85.71 ± 0.63	2.34 ± 1.2	74.18 ± 0.82	2.53 ± 1.54
	MER	89.05 ± 2.37	-2.82 ± 3.1	82.5 ± 1.05	0.1 ± 2.23	70.17 ± 1.22	-3.16 ± 1.66
	MER$^+$	91.7 ± 0.55	-0.78 ± 1.01	85.08 ± 1.19	1.53 ± 1.06	72.37 ± 0.87	0.29 ± 1.7
	DER++	87.56 ± 1.87	-2.14 ± 4.63	83.57 ± 0.64	1.62 ± 1.31	71.83 ± 1.12	-0.98 ± 1.46
	DER++$^+$	90.4 ± 0.63	-3.1 ± 0.77	85.87 ± 0.92	2.54 ± 1.22	74.09 ± 0.85	2.82 ± 2
	CLS-ER	83.02 ± 2.81	-8.88 ± 4.33	82.67 ± 0.73	-0.28 ± 1.5	71.97 ± 0.43	-3.69 ± 1.02
	CLS-ER$^+$	89.81 ± 1.77	0.85 ± 3.75	85.17 ± 0.61	1.24 ± 2.02	76.35 ± 0.41	1.97 ± 0.52

Algorithm 1 SHARC Training

```
Require: Continual learning classifier $f_\theta$, associative memory $\mathcal{A}(\cdot, \boldsymbol{\omega})$, training continuum $\mathcal{D}^{train}$,
    dropping threshold $\mu$, optimizer OPT, forgetting frequency $R$, total number of tasks $T$.
 1: $\mathcal{M}_t \leftarrow \{\}, \forall\, t = 1, 2, \cdots, T$                          ▷ Initialize episodic memory
 2: for $t = 1$ to $T$ do
 3:     $\tilde{\mathcal{M}}_k \leftarrow \text{OPT}_{\mathbf{x}}(\mathbf{x}, \mathcal{M}_{k<t}, \boldsymbol{\omega}),\ \forall\, k < t$ as Eq. 9      ▷ Associative memory *read*
 4:     for $\mathcal{B}_t \sim \mathcal{D}_t^{train}$ do
 5:         $\theta \leftarrow \text{OPT}_\theta(\theta, \mathcal{B}_t, \tilde{\mathcal{M}}_{k<t})$ as Eq. 9            ▷ Train the classifier
 6:         for $(x, y) \in \mathcal{B}_t$ do
 7:             $A = g(x)$
 8:             $A' = \text{TR}_{H,W,K}\big(\mathbb{1}\{|\boldsymbol{\alpha}^c| > Q_\mu\} \otimes \mathbf{J}_{H,W}\big) \odot A$      ▷ Channel-wise sparsity
 9:             $\mathcal{M}_t \leftarrow \mathcal{M}_t \cup (A', y)$                       ▷ Update episodic memory
10:         end for
11:     end for
12:     $\boldsymbol{\omega} \leftarrow \text{OPT}_\omega(\omega, A_t)$ as Eq. 4                      ▷ Associative memory *write*
13:     if $t \,\%\, R == 0$ then
14:         Bayesian Training by $\boldsymbol{\omega} \leftarrow \text{OPT}_\omega(\omega)$         ▷ Associative memory *forgetting*
15:     end if
16: end for
```

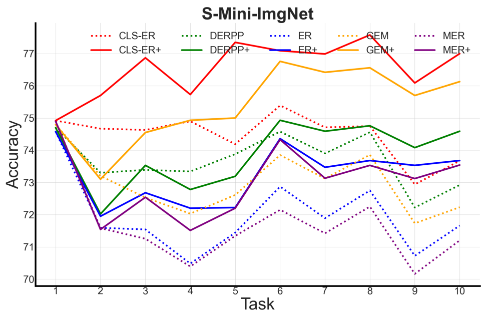
In general, when combined with SHARC, the methods used show notable enhancements in most scenarios. Specifically, in the experiments conducted with a buffer size of 1000 and CIFAR-10, the maximum improvement in accuracy reaches approximately 7%.

Table 3: **Performance comparison on image classification datasets (Class-IL).** The mean and standard deviation are calculated based on five runs with varying seeds. $^+$ denotes the corresponding method combined with our SHARC framework.

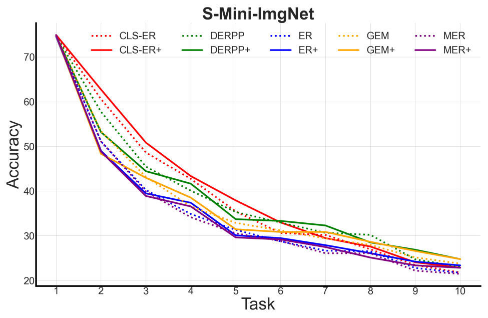
Buffer	**Model**	**S-CIFAR-10**		**S-CIFAR-100**		**S-Mini-ImgNet**	
		ACC (↑)	BWT (↑)	ACC (↑)	BWT (↑)	ACC (↑)	BWT (↑)
-	JOINT	72.85 ± 2.18	61.26 ± 8.55	45.87 ± 1.22	45.55 ± 1.45	47.08 ± 0.77	46.25 ± 0.98
-	SGD	20.47 ± 0.78	-90.16 ± 0.92	8.55 ± 1.39	-78.24 ± 0.93	12.21 ± 0.75	-67.11 ±0.77
1000	GEM	32.13 ± 6.79	-56.21 ± 13.61	23.39 ± 5.47	-48.35 ± 5.51	31.92 ± 5.29	-42.6 ± 6.71
	GEM$^+$	40.68 ± 3.18	-49.08 ± 11.47	27.49 ± 3.49	-44.72 ± 3.92	33.54 ± 2.24	-40.21 ± 4.06
	A-GEM	26.2 ± 8.53	-80.68 ± 7.31	13.38 ± 3.81	-73.22 ± 4.95	16.52 ± 1.54	-62.86 ± 2.1
	A-GEM$^+$	28.93 ± 3.67	-78.13 ± 5.81	16.04 ± 3.47	-70.52 ± 3.49	17.58 ± 1.71	-60.32 ± 1.7
	ER	33.82 ± 8.42	-53.69 ± 15.06	23.13 ± 5.06	-53.35 ± 5.2	30.77 ± 1.85	-43.9 ± 2.47
	ER$^+$	37.74 ± 2.45	-43.36 ± 14.88	26.75 ± 1.1	-49.72 ± 0.86	30.26 ± 1.95	-43.64 ± 1.07
	MER	32.87 ± 8.8	-50.42 ± 18.37	23.59 ± 3.26	-51.08 ± 4.59	29.73 ± 2.36	-44.68 ± 2.92
	MER$^+$	36.94 ± 5.38	-44.32 ± 14.72	25.22 ± 0.8	-49.68 ± 1.81	29.71 ± 1.37	-43.58 ± 1.47
	DER++	31.37 ± 7.22	-52.59 ± 16.29	19.43 ± 7.88	-59.2 ± 7.87	29.33 ± 1.83	-45.58 ± 2.29
	DER++$^+$	35.51 ± 7.44	-36.06 ± 11.08	26.35 ± 1.84	-52.05 ± 2.85	29.42 ± 2.02	-44.78 ± 2.61
	CLS-ER	24.13 ± 8.02	-33.43 ± 23.68	25.71 ± 2.91	-51.68 ± 2.84	30.3 ± 1.51	-47.22 ± 1.7
	CLS-ER$^+$	26.34 ± 5.38	-43.81 ± 7.09	23.99 ± 4.36	-53.42 ± 5.22	30.21 ± 1.25	-46.88 ± 1.73

Our method demonstrates strong performance on the image classification dataset task, with improvements observed across various metrics compared to the original method. Notably, even with a large buffer, we achieve an average improvement of approximately 3% in accuracy, providing compelling evidence of the effectiveness of our approach.

It is evident that our approach significantly enhances the model across multiple dimensions. Notably, in terms of task accuracy, all algorithms show an improvement of approximately 2%, with this value consistently increasing as the number of tasks grows. Particularly noteworthy is the observation that while the accuracy of the original algorithms tends to decrease with larger tasks, our method continues to increase in accuracy, demonstrating its effectiveness in handling a large number of multi-tasks. These findings strongly indicate the superior performance of our method in scenarios involving numerous tasks.



(a) Task-IL S-Mini-ImgeNet



(b) Class-IL S-Mini-ImgeNet

Figure 5: Learning curves of mumltiple models with / without SHARC on S-Mini-ImgeNet. Models with / without SHARC are shown in solid / dotted lines. Buffer size for all models is 200.