Appendix

Section A describes the steps of saliency map generation using Grad-CAM. Section B provides the dataset statistics used in different experiments. Pseudo-code of the episodic memory update in EPR is given in Section C. List of hyperparameters used for the baseline algorithms and our method is given in Section D. Additional results are provided in Section E.

A. Saliency Method : Grad-CAM

Gradient-weighted Class Activation Mapping (**Grad-CAM**) [41] is a saliency method that uses gradients to determine the impact of specific feature map activations on a given prediction. Since later layers in the convolutional neural network capture high-level semantics [26], taking gradients of a model output with respect to the feature map activations from one such layers identifies which high-level semantics are important for the model prediction. In our analysis, we select this layer and refer to as *target layer* [15]. List of *target layer* for different experiments is given in Table A.1.

Table A.1. Target layer names in PyTorch package for saliencies generated by different network architectures in Grad-CAM for different datasets.

Dataset	Network	Target Layer
Split CIFAR	ResNet18 (reduced)	`layer4.1.shortcut`
Split miniImageNet	ResNet18 (reduced)	`layer4.1.shortcut`
Split CUB	ResNet18	`net.layer4.1.conv2`

Let's consider the target layer has M feature maps where each feature map, $A^m \in \mathbb{R}^{u\times v}$ is of width u and height v. Also consider, for a given image ($I \in \mathbb{R}^{W\times H\times C}$) belonging to class c, the pre-softmax score of the image classifier is y_c. To obtain the class-discriminative saliency map, Grad-CAM first takes derivative of y_c with respect to each feature map A^m. These gradients are then global-average-pooled over u and v to obtain importance weight, α^c_m for each feature map:

$$\alpha^c_m = \frac{1}{uv}\sum_{i=1}^{u}\sum_{j=1}^{v}\frac{\partial y_c}{\partial A^m_{ij}}, \tag{A.1}$$

where A^m_{ij} denotes location (i,j) in the feature map A^m. Next, these weights are used for computing linear combination of the feature map activations, which is then followed by ReLU to obtain the localization map :

$$L^c_{Grad-CAM} = \text{ReLU}\,(\sum_{m=1}^{M}\alpha^c_m A^m) \tag{A.2}$$

This map is of the same size ($u \times v$) of A^m. Finally, saliency map, $I_{sm} \in \mathbb{R}^{W\times H}$ is generated by upsampling $L^c_{Grad-CAM}$ to the input image resolution using bilinear interpolation.

$$I_{sm} = \texttt{Upsample}\,(L^c_{Grad-CAM}) \tag{A.3}$$

B. Dataset Statistics

Table B.1. Statistics of the CIFAR-100, miniImageNet and CUB datasets used in task-incremental learning experiments.

	Split CIFAR	Split miniImageNet	Split CUB
num. of tasks	20	20	20
input size ($W \times H \times C$)	$32 \times 32 \times 3$	$84 \times 84 \times 3$	$224 \times 224 \times 3$
num. of classes/task	5	5	10
num. of training samples/tasks	2,500	2,500	300
num. of test samples/tasks	500	500	290

C. Memory Update Algorithm

Algorithm 2 Procedure for saliency guided episodic memory update in EPR

procedure UPDATEMEMORY($\mathcal{M}_E, \mathcal{M}_T, f_\theta$, EPF, W_p)
  `XAI`: Procedure for saliency map generation; S_{sm}: stride; t^k: task-ID
  Initialize: $\mathbf{I}_p \leftarrow [\,]$; $\mathbf{c} \leftarrow [\,]$; $\mathbf{x}_{cord} \leftarrow [\,]$; $\mathbf{y}_{cord} \leftarrow [\,]$; $\mathbf{P}_{pred} \leftarrow [\,]$ ▷ Initialize for memory selection
  for $(I, k, c) \sim \mathcal{M}_T$ **do** ▷ Sample one example at a time without replacement from $\mathcal{M}_T$
    $I_{sm} \leftarrow \texttt{XAI}(f_\theta, I, c)$ ▷ generate saliency map using Equation 1
    $x_{cord}, y_{cord} \leftarrow \texttt{average-pool}(I_{sm}, W_p, S_{sm})$ ▷ get corner coordinates of the most salient region in input,I
    $I_p \leftarrow I(x_{cord} : x_{cord} + W_p, y_{cord} : y_{cord} + W_p)$ ▷ get patch from Equation 4
    $I_p^{'} \leftarrow \texttt{Zero-pad}(I_p, x_{cord}, y_{cord})$
    $pred \leftarrow f_\theta(I_p^{'})$ ▷ check model prediction after zero-padding
    $\mathbf{I}_p \leftarrow [\mathbf{I}_p, I_p]$ ▷ add patch
    $\mathbf{P}_{pred} \leftarrow [\mathbf{P}_{pred}, pred]$ ▷ add prediction
    $\mathbf{c} \leftarrow [\mathbf{c}, c]$ ▷ add class label
    $\mathbf{x}_{cord} \leftarrow [\mathbf{x}_{cord}, x_{cord}]$ ▷ add x_{cord}
    $\mathbf{y}_{cord} \leftarrow [\mathbf{y}_{cord}, y_{cord}]$ ▷ add y_{cord}
  end for
  $(\mathbf{I}_p, \mathbf{c}, \mathbf{x}_{cord}, \mathbf{y}_{cord}) \leftarrow \texttt{select-patches}(\mathbf{I}_p, \mathbf{c}, \mathbf{x}_{cord}, \mathbf{y}_{cord}, \mathbf{P}_{pred}, \text{EPF})$ ▷ see section 6: memory patch selection
  $t^k \leftarrow k$
  $\mathcal{B}_{\mathcal{M}_E} \leftarrow (\mathbf{I}_p, t^k, \mathbf{c})$
  $\mathcal{M}_E \leftarrow \mathcal{M}_E \cup \{(\mathcal{B}_{\mathcal{M}_E}, \mathbf{x}_{cord}, \mathbf{y}_{cord})\}$ ▷ update episodic memory
  return $\mathcal{M}_E$
end procedure

D. List of Hyperparameters

List of hyperparameters used for both baseline methods and our approach is provided in Table D.1. EPF values used in different experiments in our method are given in Table D.2.

Table D.1. Hyperparameters grid considered for the baselines and our approach. The best values are given in parentheses. Here, 'lr' represents learning rate. In the table, we represent Split CIFAR as 'cifar', Split miniImageNet as 'minImg' and Split CUB as 'cub'. EPF is experience packing factor and $\mathcal{M}_T$ is the temporary ring buffer in EPR.

Methods	**Hyperparameters**
Finetune	lr : 0.003, 0.01, 0.03 (cifar, minImg, cub), 0.1, 0.3, 1.0
EWC	lr : 0.003, 0.01, 0.03 (cifar, minImg, cub), 0.1, 0.3, 1.0 regularization, λ : 0.1, 1, 10 (cifar, minImg, cub), 100, 1000
RRR	lr : 0.003, 0.01 (cub), 0.03, 0.1, 0.3, 1.0 regularization : 10, 100 (cub), 1000
A-GEM	lr : 0.003, 0.01, 0.03 (cifar, minImg, cub), 0.1, 0.3, 1.0
MER	lr : 0.003, 0.01, 0.03 (cifar, minImg), 0.1 (cub), 0.3, 1.0 with in batch meta-learning rate, γ : 0.01, 0.03, 0.1 (cifar, minImg, cub), 0.3, 1.0 current batch learning rate multiplier, s : 1, 2, 5 (cifar, minImg, cub), 10
MEGA-I	lr : 0.003, 0.01, 0.03 (cifar, minImg, cub), 0.1, 0.3, 1.0 sensitivity parameter, ϵ : $1e^{-5}$, $1e^{-4}$, 0.001, 0.01 (cifar, minImg, cub), 0.1
DER++	lr : 0.003, 0.01, 0.03 (minImg, cub), 0.1 (cifar), 0.3, 1.0 regularization α : 0.1 (minImg), 0.2 (cifar), 0.5 (cub), 1.0 regularization, β : 0.5 (cifar, minImg, cub), 1.0
ASER	lr : 0.003, 0.01, 0.03 (cub), 0.1 (cifar, minImg), 0.3, 1.0 K : 3 (cifar, minImg, cub); N_c : 100 (cifar, miniImg), 150 (cub), 250
ER-Reservoir	lr : 0.003, 0.01, 0.03 (cub), 0.1 (cifar, minImg), 0.3, 1.0
ER-RING	lr : 0.003, 0.01, 0.03 (cifar, minImg, cub), 0.1, 0.3, 1.0
HAL	lr : 0.003, 0.01, 0.03 (cifar, minImg), 0.1, 0.3, 1.0 regularization, λ : 0.01, 0.03, 0.1, 0.3 (minImg), 1 (cifar), 3, 10 mean embedding strength, γ : 0.01, 0.03, 0.1 (cifar, minImg), 0.3, 1, 3, 10 decay rate, β : 0.5 (cifar, minImg) gradient steps on anchors, k : 100 (cifar, minImg)
Multitask	lr : 0.003, 0.01, 0.03 (cifar, minImg, cub), 0.1, 0.3, 1.0
EPR (ours)	lr (task-incremental) : 0.01, 0.03 (cub), 0.05 (minImg), 0.1 (cifar), 0.3, 1.0 lr (class-incremental) : 0.01, 0.05 (cifar, minImg), 0.1 examples per class temporarily stored in $\mathcal{M}_T$: $\gamma \times$ EPF; γ : 2(cub), 5 (cifar,minImg) stride, S_{sm} : 1 (cifar, minImg), 2, 3 (cub)

Table D.2. Experience Packing Factor (EPF) for different n_{sc} used in our (a) task-incremental learning and (b) class-incremental learning experiments. Input image width, W for CIFAR, miniImageNet and CUB dataset are $32, 84$ and 224 respectively. For given n_{sc}, EPF and W, corresponding memory patch sizes (W_p) are also given in the table.

(a)

	Split CIFAR		**Split miniImageNet**		**Split CUB**	
n_{sc}	EPF	W_p	EPF	W_p	EPF	W_p
2	3	26	5	53	7	119
1	2	22	3	48	4	112
0.75	1	27	2	51	3	112
0.5	1	22	2	42	2	112

(b)

		CIFAR-100 (20 Tasks)		**miniImageNet** (10 Tasks)	
$\lvert M_E \rvert$	n_{sc}	EPF	W_p	EPF	W_p
2k	20	25	28	25	75
1k	10	13	28	13	73

E. Additional Results

Table E.1. Performance comparison of different experience replay methods for different memory sizes in task-incremental learning setup. Number of memory slots per class, n_{sc}=$\{0.5, 0.75\}$ refers to memory size, $|\mathcal{M}_E|$=$\{42, 64\}$ for CIFAR and miniImageNet, and $|\mathcal{M}_E|$=$\{85, 128\}$ for CUB. Average and standard deviations are computed over 5 runs for different random seeds.

		Split CIFAR		**Split miniImageNet**		**Split CUB**	
n_{sc}	**Methods**	ACC (%)	BWT	ACC (%)	BWT	ACC (%)	BWT
-	Finetune	42.9 ± 2.07	- 0.25 ± 0.03	34.7 ± 2.69	- 0.26 ± 0.03	55.7 ± 2.22	- 0.13 ± 0.03
0.75	MEGA-I	48.9 ± 1.68	- 0.21 ± 0.01	43.8 ± 1.58	- 0.14 ± 0.01	61.5 ± 2.08	- 0.08 ± 0.01
	DER++	50.0 ± 1.81	- 0.19 ± 0.02	47.2 ± 1.54	- 0.12 ± 0.01	64.8 ± 1.61	- 0.06 ± 0.01
	ER-RING	50.4 ± 0.85	- 0.21 ± 0.02	44.9 ± 1.49	- 0.14 ± 0.02	64.0 ± 1.29	- 0.05 ± 0.01
	EPR (Ours)	**56.8 ± 1.59**	**- 0.12 ± 0.02**	**51.1 ± 1.47**	**- 0.06 ± 0.01**	**70.7 ± 0.72**	**- 0.03 ± 0.01**
0.5	MEGA-I	43.7 ± 1.26	- 0.26 ± 0.02	39.6 ± 2.35	- 0.18 ± 0.02	57.7 ± 0.62	- 0.11 ± 0.01
	DER++	47.5 ± 1.58	- 0.21 ± 0.01	45.6 ± 0.56	- 0.13 ± 0.01	62.5 ± 1.45	- 0.08 ± 0.01
	ER-RING	44.6 ± 0.84	- 0.27 ± 0.01	39.1 ± 1.38	- 0.20 ± 0.02	59.2 ± 0.97	- 0.10 ± 0.01
	EPR (Ours)	**55.6 ± 0.54**	**- 0.13 ± 0.02**	**49.2 ± 1.20**	**- 0.07 ± 0.01**	**70.3 ± 0.91**	**- 0.03 ± 0.01**