

# SUPPLEMENTARY FILE FOR “DeL: BIOLOGICALLY PLAUSIBLE DENDRITIC LEARNING ENABLES CLASS-INCREMENTAL LEARNING ”

**Anonymous authors**

Paper under double-blind review

## A APPENDIX A. METHOD

### A.1 LAYER CONNECTIONS OF DeL

We incorporate synaptic plasticity into the architectural design of DeL. Inspired by neuroscience, the connections between synapses and dendrites form distinct patterns. Combinations of these patterns appear to exhibit various forms of synaptic plasticity. The input data stream is normalized to lie within the interval  $(-1, 1)$ . We employ learnable sigmoid functions to evaluate synaptic strength. Accordingly, four types of connections are defined, as illustrated in Figure 1. **Positive Connection** indicates that the strength of the synaptic signal transmitted to the dendrite increases with the input value, thereby activating the dendritic layer. Conversely, **Negative Connection** indicates that the synaptic signal strength increases as the input value decreases. **Constant 0** and **Constant 1** represent continuous inhibition and continuous activation of the dendrite, respectively. By leveraging these diverse connection patterns, DeL can extract discriminative features.

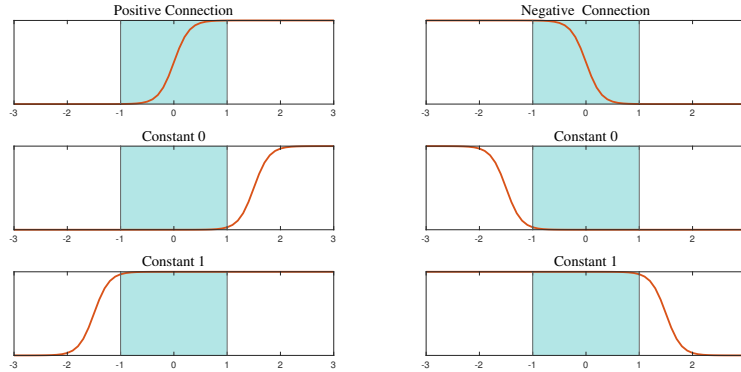


Figure 1: The connections between synaptic layer and dendritic layer.

### A.2 WEIGHT ALIGNMENT OF DeL

In class-incremental learning (CIL), parameter drift can exacerbate catastrophic forgetting. As the model learns new classes, feature conflicts arise, causing it to adjust its parameters to accommodate the new tasks. This adaptation often leads to the forgetting of previously learned tasks. Additionally, in the classifier, the increase in the number of classes causes a shift in the original parameters’ confidence and influence across different categories. Therefore, it is necessary to calibrate the parameters associated with old tasks to mitigate such drift. Inspired by the study in Shi et al. (2021), we apply a weight alignment strategy to address this limitation. Let  $\mathbf{w} \in \mathbb{R}^{D \times M \times N}$  denote the weights of DeL, where  $D$  is the input dimension,  $M$  is the number of dendritic branches, and  $N$  is the number

Table 1: The details of each dataset.

Dataset	# Class	# of Total	# of Train	# of Test	$B$	Inc	$N_t$	$N_{ts}$	Buffer	Resolution
CIFAR100	100	60000	50000	10000	0	10	100	100	20	$224 \times 224$
CUB200	200	11788	9430	2358	0	20	32	3	10	$224 \times 224$
VTAB	50	10415	1796	8619	0	10	2	1	5	$224 \times 224$
ImageNet-A	200	7500	5981	1519	0	20	38	7	1	$224 \times 224$
ImageNet-R	200	30000	24000	6000	0	20	7	40	20	$224 \times 224$
Omnibench	300	95682	89697	5985	0	30	250	18	20	$224 \times 224$

of classes. For each class,  $\mathbf{w}$  is formulated as,

$$\mathbf{w} = \begin{pmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,D} \\ w_{2,1} & w_{2,2} & \dots & w_{2,D} \\ \vdots & \vdots & \ddots & \vdots \\ w_{M,1} & w_{M,1} & \dots & w_{M,D} \end{pmatrix} \times N \quad (1)$$

where  $w_{i,j}$  means the weight of the  $i$ th synapse into the  $j$ th dendrite.

When the model learns new task with  $\Delta$  classes, the weights are updated to  $\mathbf{w} \in \mathbb{R}^{D \times M \times (N+\Delta)}$  to accommodate the new tasks. We denote the old and new weights as  $\mathbf{w}^o = \mathbf{w}[:, :, : N]$  and  $\mathbf{w}^n = \mathbf{w}[:, :, N : N + \Delta]$ , respectively. Notably, weight alignment is performed at the synaptic layer, as its parameters directly influence synaptic plasticity and synaptic strength. To compute the alignment coefficient  $\gamma$ , we calculate the  $\ell_2$  norms of the old and new weights as follows:

$$\begin{aligned} \mathbf{w}_o &= \sqrt{\sum_{i=1}^N |\mathbf{w}_{i,:}^o|^2}, \\ \mathbf{w}_n &= \sqrt{\sum_{i=1}^N |\mathbf{w}_{i,:}^n|^2}, \\ \gamma &= \frac{\text{mean}(\mathbf{w}_o)}{\text{mean}(\mathbf{w}_n)} \end{aligned} \quad (2)$$

where  $\mathbf{w}_{i,:}^o$  and  $\mathbf{w}_{i,:}^n$  mean the weights of the  $i$ th synaptic layer for old and new tasks. The old weights are aligned by  $\gamma$ , i.e.,  $\mathbf{w}^o = \gamma \mathbf{w}^o$ .

## B APPENDIX B. EXPERIMENT

### B.1 DETAILS OF DATASETS

In this study, we use six datasets to evaluate the model’s performance. The details of these datasets are summarized in Table 1, including the total number of samples, and the sizes of the training and test sets for each dataset.  $B$  denotes the number of classes that have been learned, with all datasets starting from class 0. Inc refers to the number of newly introduced classes at each incremental step.  $N_t$  and  $N_{ts}$  represent the minimum number of samples per class in the training and test sets, respectively. Due to the presence of datasets with extremely limited samples (e.g., only one sample per class), we set different rehearsal buffer sizes (Buffer) for rehearsal-based methods to account for such imbalance. The resolution of all images are resized to  $224 \times 224$ .

### B.2 REPRODUCIBILITY DETAILS

To enhance the reproducibility of our experiments, we summarize the training parameters used across all methods. These include batch size (BS), number of epochs for the base session ( $E_b$ ), learning rate ( $lr$ ), learning rate decay ( $d_{lr,b}$ ), and weight decay ( $d_{w,b}$ ). Separate parameter settings are provided for initial learning and incremental learning. The parameters for incremental learning are denoted as  $E_I$  (epochs),  $lr_I$  (learning rate),  $d_{w,I}$  (weight decay), and  $d_{lr,I}$  (learning rate decay), respectively. For complete details, please refer to Table 2. In particular, for VTAB dataset  $lr$  and  $lr_I$  are set to 0.01. The details version of used package are also listed in Table 3.

### B.3 THE DETAILS OF BASELINE METHODS

For all baseline methods, ResNet18 is used as the backbone. Both the main classifier and auxiliary classifier adopt the cross-entropy loss function Mao et al. (2023). Furthermore, due to differences

Table 2: The details of hyper-parameters.

Method	BS	Initial Learning				Incremental Learning			
		$E_b$	$lr$	$D_{w,b}$	$D_{L,b}$	$E_I$	$lr_I$	$D_{w,I}$	$D_{L,I}$
Finetune	128	200	0.1	0.1	5e-4	80	0.1	2e-4	0.1
Replay	128	200	0.1	0.1	5e-4	170	0.1	2e-4	0.1
LwF	128	200	0.1	0.1	0.005	250	0.1	2e-4	0.1
iCaRL	128	200	0.1	0.1	5e-4	170	0.1	2e-4	0.1
DER	128	200	0.1	0.1	5e-4	170	0.1	2e-4	0.1
TagFex	64	200	0.1	0.1	5e-4	170	0.1	2e-4	0.1

Table 3: The details of software version.

Package	Version
Python	3.12.0
PyTorch	2.5.1+cu124
SciPy	1.15.3
Scikit-learn	1.7.1
OpenCV-Python	4.11.0.86
einops	0.8.1
grad-cam	1.5.5
Pillow	11.3.0
NumPy	2.3.2

in learning strategies, we apply the loss functions as originally proposed in each respective CIL method. For rehearsal-based methods, the loss function is formulated as,  $\mathcal{L} = \text{CE}(\mathbf{x}, y)$ , where CE denotes the standard cross-entropy between the input  $x$  and its corresponding label  $y$ . In contrast, expansion-based methods dynamically adjust their network structures. Specifically, their output is defined as,  $f(\mathbf{x}) = \mathbf{w}[\bar{\phi}_o(\mathbf{x}), \phi_n(\mathbf{x})]$ , where  $\bar{\phi}_o(\mathbf{x})$  is the frozen feature representation from the old backbone, and  $\phi_n(\mathbf{x})$  is the feature representation extracted by the newly added backbone for incremental classes. Accordingly, the loss function is computed as:

$$\mathcal{L} = \sum_{k=1}^{|\mathcal{Y}_b|} -\mathcal{I}(y=k) \log \mathcal{S}_k(\mathbf{w}^\top [\bar{\phi}_o(\mathbf{x}), \phi_n(\mathbf{x})]) \quad (3)$$

where  $\mathcal{I}(\cdot)$  is the indicator function,  $\mathcal{S}_k(\cdot)$  denotes the softmax probability of class  $k$ , and  $|\mathcal{Y}_b|$  is the number of classes at the current incremental step. Knowledge distillation generally utilizes Kullback–Leibler (KL) divergence Hershey & Olsen (2007) to measure the difference between two probability distributions. The KL divergence loss is defined as:

$$\mathcal{L}_{KL}(P, Q) = \sum_j q_j \log \frac{q_j}{p_j} \quad (4)$$

where  $P = \{p_1, p_2, \dots, p_j\}$  and  $Q = \{q_1, q_2, \dots, q_j\}$  represent the output and target distributions, respectively, both obtained using a softened softmax function:

$$p_j = \frac{e^{y_j/t}}{\sum_{i=1}^n e^{y_i/t}} \quad (5)$$

where  $t$  is the temperature parameter used to smooth the logits. A higher temperature produces a softer probability distribution, facilitating better knowledge transfer from teacher to student.

#### B.4 ABLATION STUDY

In the main body of the paper, we present key results to analyze the performance differences across various layers. In this supplementary file, we provide more detailed results in Table 4. Our findings reveal that activation and normalization applied to the dendritic layer significantly degrade the performance of DeL. This is primarily because the dendritic layer integrates multiple synaptic signals, resulting in relatively large signal magnitudes. Certain activation functions, such as sigmoid

Table 4: The ablation study of Synaptic Plasticity on CIFAR 100 with **DER**. The best performance is shown in bold. DA and DN represent the activation and normalization of the dendritic layer. Soma denotes the output activation for downstream task.

	SA	SN	DA	DN	Soma	CIFAR100		VTAB	
						$\bar{A}$	$A_L$	$\bar{A}$	$A_L$
	✗	✗	✗	✗	✗	71.93	61.18	58.03	49.30
	✓	✗	✗	✗	✗	<b>74.68</b>	<b>64.31</b>	<b>64.51</b>	<b>56.22</b>
	✓	✓	✗	✗	✗	73.69	63.58	63.85	56.19
	✓	✓	✓	✗	✗	2.93	1.00	5.12	1.65
	✓	✓	✗	✓	✗	2.93	1.00	2.70	0.75
	✓	✓	✓	✓	✗	2.93	1.00	2.70	0.75
	✓	✓	✓	✓	✓	2.93	1.00	2.70	0.75

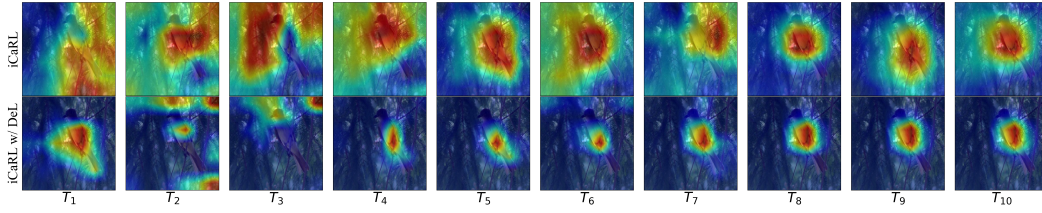


Figure 2: Grad-CAM visualization of complex environment.

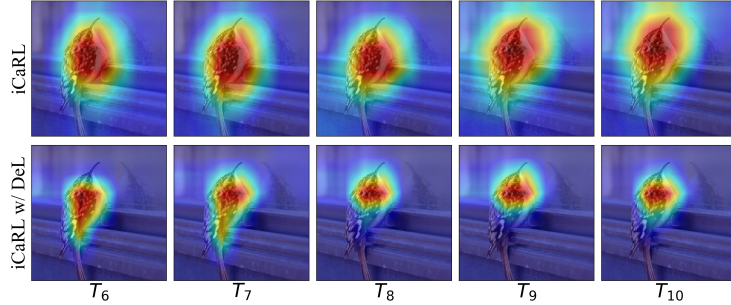


Figure 3: Grad-CAM visualization of distinctive features.

and tanh, tend to homogenize the outputs of dendrites, thereby severely limiting synaptic plasticity. Likewise, normalization diminishes the diversity of signal responses. Therefore, we suggest that the activation strategy for the dendritic layer should be carefully designed to preserve the model’s representational capacity.

## B.5 VISUALIZATIONS

In addition, we present Grad-CAM visualizations to demonstrate the advantages of DeL in CIL. We observed that DeL can effectively handle complex scenarios by accurately capturing features in challenging environments, which further supports its scalability in CIL (Figure 2). Moreover, as shown in Figure 3, DeL’s focus area is relatively broad during the initial learning phases, but it becomes increasingly concentrated as the incremental learning process progresses. This behavior suggests that DeL is capable of compressing key features from previously learned tasks, thereby preserving more memory capacity for learning new tasks under a fixed memory constraint. We provide an illustration of t-SNE Maaten & Hinton (2008) to visualize feature representations. From this figure, it is evident that DeL enables the model to extract more distinctive features. Specifically, iCaRL augmented with DeL enhances the separability among different classes, making the reversed features more class-specific and thereby helping to mitigate forgetting.

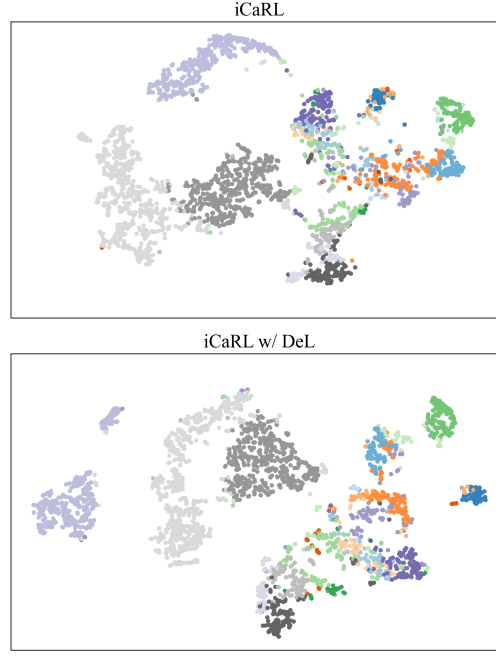


Figure 4: The visualization of t-SNS on VTAB.

Table 5: The trainable parameters of methods with and without DeL on different datasets.

Methods	CIFAR100	CUB200	VTAB	ImageNet-A	ImageNet-R	OmniBench
Finetune	10.70 M	10.75 M	10.68 M	10.75 M	10.75 M	10.79 M
w/ DeL	10.74 M	10.84 M	10.70 M	10.84 M	10.84 M	10.92 M
Replay	10.70 M	10.75 M	10.68 M	10.75 M	10.75 M	10.79 M
w/ DeL	10.74 M	10.84 M	10.70 M	10.84 M	10.84 M	10.92 M
LwF	10.70 M	10.75 M	10.68 M	10.75 M	10.75 M	10.79 M
w/ DeL	10.74 M	10.84 M	10.70 M	10.84 M	10.84 M	10.92 M
iCaRL	10.70 M	10.75 M	10.68 M	10.75 M	10.75 M	10.79 M
w/ DeL	10.74 M	10.84 M	10.70 M	10.84 M	10.84 M	10.92 M
DER	11.15 M	11.65 M	10.79 M	11.65 M	11.65 M	12.14 M
w/ DeL	11.65 M	12.64 M	10.92 M	12.64 M	12.64 M	13.62 M
TagFex	26.31 M	26.82 M	25.96 M	26.82 M	26.82 M	27.32 M
w/ DeL	26.81 M	27.82 M	26.09 M	27.82 M	27.82 M	28.80 M

## B.6 ANALYSIS OF EFFECTIVENESS

We use a hit matrix  $\mathbf{H} \in \mathbb{R}^{N \times D}$  to record how frequently each node is activated for images belonging to the same category, where  $N$  is the number of classes and  $D$  is the number of nodes. For a sample from class  $i$ , its activation vector is defined as  $\mathbf{a}_i = [a_{i,1}, \dots, a_{i,D}]$ , where  $a_{i,j}$  denotes the activation of the  $j$ th node. We binarize the activations using a threshold of 0.5—specifically, if  $a_{i,j} > 0.5$ , we set  $a_{i,j} = 1$ ; otherwise,  $a_{i,j} = 0$ . The threshold of 0.5 is chosen because it is the mean value of the sigmoid activation function. Thereby, the hit matrix is then defined as:

$$\mathbf{H} = \begin{pmatrix} c_{1,1} & \dots & c_{1,D} \\ c_{2,1} & \dots & c_{2,D} \\ \vdots & \ddots & \vdots \\ c_{N,1} & \dots & c_{N,D} \end{pmatrix} \quad (6)$$

where  $c_{i,j} = \sum \mathbf{a}_i$  represents the total activation count of the  $j$ th node for all samples in class  $i$ . To compute the entropy, we first normalize the hit matrix by adding an additional row to represent

Table 6: The training time of methods with and without DeL on different datasets.

Methods	CIFAR100	CUB200	VTAB	ImageNet-A	ImageNet-R	OmniBench
Finetune	0:32	0:41	0:17	0:33	1:23	3:08
w/ Del	0:33	0:42	0:17	0:33	1:24	3:17
Replay	0:44	1:10	0:22	2:36	0:44	6:54
w/ Del	0:36	1:03	0:18	2:14	0:37	6:54
LwF	1:07	1:08	0:21	2:29	0:51	7:48
w/ Del	1:07	1:08	0:21	2:27	0:50	7:53
iCaRL	1:43	1:53	0:23	0:55	3:56	10:07
w/ Del	1:57	1:55	0:24	0:56	3:59	10:15
DER	3:10	3:19	1:27	7:07	0:31	17:01
w/ Del	3:20	3:25	1:29	7:20	0:39	17:47
TagFex	6:06	5:40	0:52	12:19	2:16	7:01
w/ Del	6:12	5:43	0:51	12:30	2:17	8:02

the number of times each node remained inactive. This ensures that the sum across all classes (including inactivity) equals the total number of samples in the test set. Dividing each entry in the updated matrix by the total number of samples gives a probability matrix, from which we compute the entropy for each node in the  $k$ th layer using:

$$E = - \sum_{i=1}^D p(x_i) \log_2 p(x_i) \quad (7)$$

where  $p(x_i)$  is the normalized activation probability for node  $i$ .

## B.7 ANALYSIS OF COMPLEXITY

We evaluate the computational complexity of DeL. As shown in Table 5, we compared the number of parameters between baseline models with and without DeL. Despite incorporating a more biologically realistic dendritic structure, DeL introduces only a marginal increase in parameter count compared to traditional MLP-based models. In our experiments, the average increase is 2%, with a minimum of 0.4% and a maximum of 10%. Additionally, the training times are reported in Table 6. It can be observed that the training time did not increase significantly, with the added overhead limited to a few minutes.

## REFERENCES

- John R Hershey and Peder A Olsen. Approximating the kullback leibler divergence between gaussian mixture models. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 4, pp. IV–317. IEEE, 2007.
- Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.
- Anqi Mao, Mehryar Mohri, and Yutao Zhong. Cross-entropy loss functions: Theoretical analysis and applications. In *International Conference on Machine learning*, pp. 23803–23828. PMLR, 2023.
- Xiangwei Shi, Yunqiang Li, Xin Liu, and Jan van Gemert. Weightalign: Normalizing activations by weight alignment. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 9788–9795. IEEE, 2021.