Supplementary Material for Dense Network Expansion for Class Incremental Learning

Zhiyuan Hu[1] Yunsheng Li[2] Jiancheng Lyu[3] Dashan Gao[3] Nuno Vasconcelos[1]
[1]UC San Diego [2]Microsoft Cloud + AI [3]Qualcomm AI Research
{z8hu, nvasconcelos}@ucsd.edu, yunshengli@microsoft.com, {jianlyu, dgao}@qti.qualcomm.com

In this supplementary material, we discuss more details about the architecture of the proposed Dense Network Expansion (DNE) method, the experiment setup and add several experiments to further validate the effectiveness of DNE.

A. Model Design

A.1. MHSA block

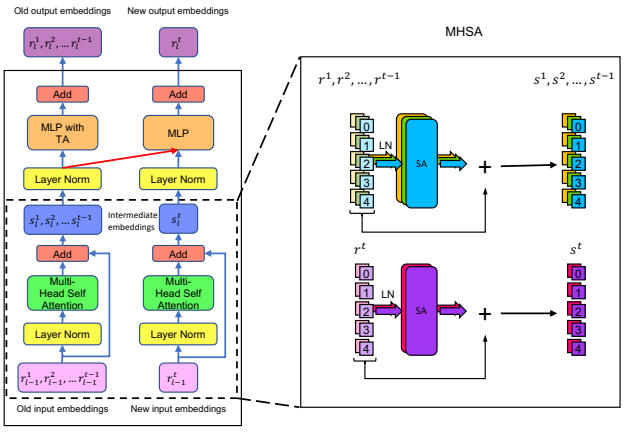

Figure A. In MHSA, heads belong to old tasks are processed with fixed old self-attention blocks. Only the newly added self-attention blocks are trained.

In DNE, cross-task attention is only used in MLP layers of a transformer block. In MHSA a "sparse" connection is used, as is illustrated in Figure A. Specifically, we split the input r into different heads based on Equation 7 and similar to Equation 15 of main paper, which are then processed as:

$$u = [u^{1,1}, u^{1,2}, \dots, u^{1,H_1}, u^{2,1}, \dots, u^{t,H_t}] \in \mathbb{R}^{P\times D\times H} \tag{1}$$

$$u^{i,j} = \text{SA}^{i,j}(\text{LN}(r^{i,j})) \in \mathbb{R}^{P\times D} \tag{2}$$

where SA is the self-attention block as in [1], $u^{i,j}$ is the intermediate output for head j in task i. Note that, only the self-attention blocks of the current task (i.e. $\text{SA}^{i,j}$ with $i = t$) will be trained. After this, all the heads from the same task are fused for the output s:

$$s = s^1 \oplus s^2 \oplus \cdots \oplus s^t \in \mathbb{R}^{P\times(D\times H)} \tag{3}$$

$$s^i = r^i + \text{FC}^i(u^{i,1} \oplus u^{i,2} \oplus \cdots \oplus u^{i,H_i}) \in \mathbb{R}^{P\times(D\times H_i)} \tag{4}$$

where FC is a linear layer and s^i the output embedding for i-th task.

A.2. Task tokens

Transformer-based CIL methods [3, 7–9], typically include an extra block f_{L+1}, where global learned task tokens $e = [e^1, e^2, \dots, e^t]$ are used to produce task-specified features and are then taken as the inputs of the classification layer, according to

$$e' = f_{L+1}(r_L, e; \theta_{L+1}) \tag{5}$$

$$g(x;\theta) = h(e';\phi), \tag{6}$$

to provide the classifier with increased task sensitivity. For network expansion based methods, $e^1, e^2, \dots, e^{t-1}$ will be fixed and only e^t is trained at current task t.

A.3. Training objectives

Given an input image x with label y. Let $g(x;\theta)$ be the current model and $g'(x;\theta')$ the fixed model of the last task, $Y = \sum_{i=1}^{t} |\mathcal{Y}_i|$, and $Y' = \sum_{i=1}^{t-1} |\mathcal{Y}_i|$ the total number of classes in current task and last task.

The classification loss is given by:

$$\mathcal{L}_{ce} = \text{CE}(g(x;\theta), y) \tag{7}$$

where CE is the cross-entropy function.

In parallel with the classifier $h(x;\theta)$, a $|\mathcal{Y}_t + 1|$ way auxiliary classifier $\hat{h}(x;\theta)$ is added over the final feature (in DNE the e') to generate the auxiliary logits:

$$\hat{g}(x;\theta) = \hat{h}(e';\hat{\phi}) \in \mathbb{R}^{|\mathcal{Y}_t|+1} \tag{8}$$

where $\hat{\phi}$ is the parameters of the auxiliary classifier.

To supervise this logit, an auxiliary label for input x is given by:

$$\hat{y} = \begin{cases} 1 & \text{if} \quad y \leq Y' \\ y+1 & \text{else} \end{cases} \tag{9}$$

The auxiliary labels consider all previous classes as an "outlier" class so that the newly added blocks can learn the knowledge that is complementary to previous tasks. This is implemented by the task expert loss:

$$\mathcal{L}_{te} = \text{CE}(\hat{g}(x;\theta), \hat{y}) \tag{10}$$

The distillation loss is defined as:

$$\mathcal{L}_{dis} = \text{KL}(\text{SoftMax}(g(x;\theta)[:Y']), \text{SoftMax}(g'(x;\theta))) \tag{11}$$

where KL is the KL divergence, SoftMax is the SoftMax function and $g(x;\theta)[:Y']$ is the first Y' outputs of $g(x;\theta)$.

The final loss is the weighted average of the previous three losses:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{ce} + \lambda_2 \mathcal{L}_{te} + \lambda_3 \mathcal{L}_{dis} \tag{12}$$

where λ_1, λ_2 and λ_3 are the weights.

B. Implementation Details

B.1. Class balanced tuning

Similar to [4], we first train the entire network $g(x;\theta)$ with the dataset D_t from current task and the memory buffer$\mathcal{M}$. However, since the size of D_t is typically much larger than $\mathcal{M}$, $g(x;\theta)$ trained on $D_t \cup \mathcal{M}$ is heavily biased towards the classes in current task t. To reduce this bias, we subsample the data in D_t to build a subset D'_t, such that all the categories have the same number of input data, i.e. $|\{y|(x,y) \in D'_t, y = i\}| = |\{y|(x,y) \in \mathcal{M}, y = j\}|, \forall i,j$. After that, the backbone is fixed, only the last layer $f_{L+1}(r_L, e; \theta_{L+1})$ and the classifier $h(e';\phi)$ is tuned with $D'_t \cup \mathcal{M}$.

B.2. Dimension of variants of comparison methods

In Figure 6(a)(b) of the main paper, we compare the accuracy-scale trade-off between DNE and other comparison methods, specifically, we change the feature dimension or backbone network of comparison methods to change their model sizes. The detailed setups are summarized in Table A, note that, the methods use the identical setups on ImageNet100 and CIFAR100 so we only list the setup of backbone, N_s, feature dimension (denoted as Dim in the Table) and the FLOPs F in this setup.

Method	Backbone	Dim	F
iCaRL [5]	ResNet18	768	2.50G
	ResNet18	864	3.16G
	ResNet18	1024	4.44G
PODNet [2]	ResNet18	768	2.50G
	ResNet18	864	3.16G
	ResNet18	1024	4.44G
Dytox [3]	Transformer	512	2.46G
	Transformer	576	3.12G
	Transformer	672	4.24G
DER [10]	ResNet18	320	2.62G
	ResNet18	352	3.16G
	ResNet18	400	4.08G
FOSTER [6]	ResNet18	768	2.50G
	ResNet18	864	3.16G
	ResNet18	1024	4.44G
	ResNet34	512	2.32G
	ResNet50	512	2.62G
	ResNet101	512	5.04G
	ResNet152	512	7.48G

Table A. Detailed setups of the comparison methods

B.3. Hyperparameters

We train DNE for 500 epochs and do class balanced tuning for 20 epochs. The learning rate is 2.5×10^{-4} and weight decay is 1×10^{-6}, we use SGD optimizer to train our model. The batch is 256 and the model is trained on 4 GPUs in parallel. The memory buffer is set as 2000 and we use the herd selection algorithm [5] to update the memory buffer. In the 6-layer transformer, we set the patch size as 4 on CIFAR100 and on ImageNet100 the patch size is 16. The dimension of each head is 32. We empirically set $\lambda_1 = \lambda_3 = 1$ and $\lambda_2 = 0.1$.

C. Experiments

Spatial-Task Attention. We provide a detailed analysis on Spatial-Task Attention(STA) in this experiments. As is illustrated in Figure 2 and 3 of main paper. Attentions in STA are divided into 4 different groups: *Same Patch Same Head* (SPSH), *Different Patch Same Head* (DPSH), *Same Patch Different Head* (SPDH) and *Different Patch Different Head* (DPDH). Standard attention module learns the *Independent Attention* (IA) which includes SPSH and DPSH. This leverages the spatial attention across patches. To implement the joint Spatial-Task Attention, SPDH or DPDH need to be included. This leads to three variants of STA: (IA)+SPDH, (IA)+DPDH and (IA)+SPDH+DPDH. To evalute the effects of these 4 groups of attention in IA and STA, we consider several evaluation metrics. Specifically, the *Portion* of each group of attention, which is the sum of each

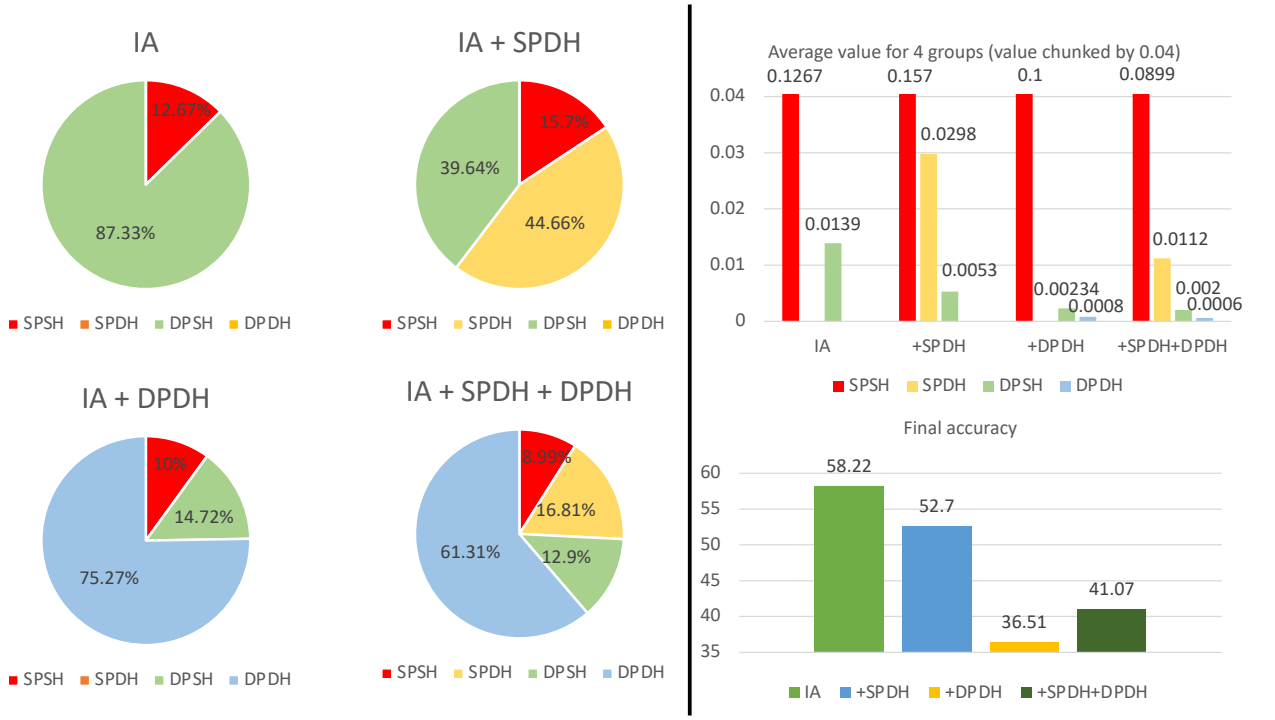

Figure B. Left: Portion of 4 groups attention within the entire attention matrix for IA model and three variants of STA model. Top-right: Average value of 4 groups attention for IA and three variants of STA. Bottom-right: Final accuracy of IA and three variants of STA.

type of attention divided by the sum of the entire attention matrix. The *Average Value* of each group of attention and the *Final Accuracy* of IA and three variants of STA. The experiments are conducted on CIFAR100 with $N_s = 10$.

Figure B summarizes these results. In the independent attention, DPSH dominates the attention matrix, so that the spatial attention across different patches is properly learned. However, as SPDH is introduced (IA+SPDH model), the portion of DPSH decreases from 87.33% to 39.64%. This is reasonable, as SPDH leverages attention of *exactly the same patch* in different heads, which have very similar representations. As a result, the average value of SPDH (0.0298) is significantly larger than DPSH (0.0053). The attention module is attracted by the cross-head, or cross-task attention so the spatial connections are not well learned. The accuracy also decreases from 58.22% to 52.70%. When the DPDH is added (IA+DPDH model and IA+SPDH+DPDH model), the attention matrix is completed dominated by DPDH (75.27% and 61.31%). Representations of different patches and different heads are weakly related, this is ture, as the average value of DPDH is merely 0.0008 or 0.0006, significantly smaller than SPDH and DPSH. However, DPDH has much more entries within the attention matrix. Suppose P patches of H heads are considered in STA. SPSH will include HP entry, SPDH includes $HP(H-1)$ entries, DPSH includes $HP(P-1)$ entries and the rest $HP(HP-H-P+1)$ entries all belong to DPDH! In a common setup where $H = 16$ and $P = 64$, 92% the entries belong to DPDH. Both SPDH and DPSH are diluted by small-valued but innumerous-numbered DPDH entries. As a result, the final accuracy dropped significantly. Compare to the independent attention model which reaches 58.22% accuracy, IA+DPDH reaches only 386.51% and IA+SPDH+DPDH reaches 41.07%. Based on these observations, we disentangle the spatial and task attention in our proposed DNE model.

CTA vs depth. We ablated the impact of adding CTA to each layer of the DNE model. Starting from the NE model (no CTA connections), we gradually add CTA to each layer, from shallow to deep. Figure C left shows that CTA is beneficial at all layers, with larger gains for shallower layers. While adding CTA to layer 1-3 increases LA by about 2%, for deep layers the gain is about 1%. This is not surprising, since the low level features of the shallow layers are more reusable by all tasks. Higher layers have more semantic features, specialized to the task classes. These results show that CTA enables old experts to share knowledge with new experts, and this is useful at all semantic levels.

Memory buffer size. We measure the performances of DNE under different memory settings. The memory buffer size is reduced from 2000 to 200. It can be shown from the right of Figure C that DNE consistently outperforms other baseline methods in various memory settings.

TAB matrices. The TAB of Figure 5 in main paper has three learned matrices, W_q, W_k, and W_v. DNE shares W_q, W_k, across tasks and has flexible W_v per task. This re-

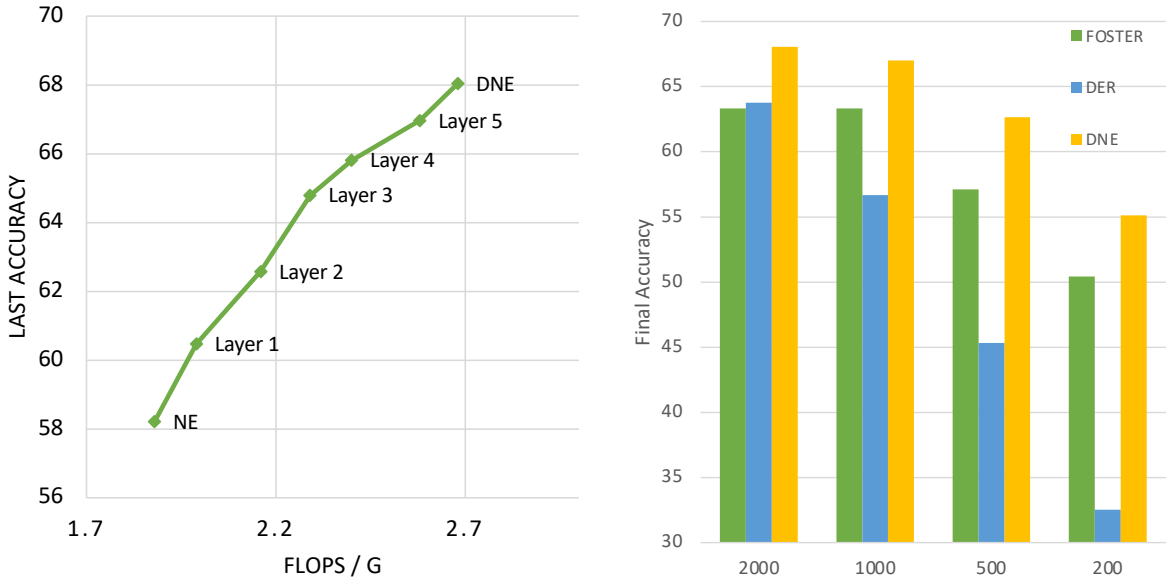

Figure C. Ablation Study, experiments conducted on CIFAR100 with $N_s = 10$. Left: Gradually add CTA across layers. Right: Effect of different memory buffer size.

Model	W_q	W_k	W_v	$LA \uparrow$	$AA \uparrow$
MLP	-	-	s	63.80	70.67
	-	-	f	67.08	73.22
DNE	s	s	s	64.44	71.17
	f	f	s	64.92	71.38
	f	f	f	67.23	73.13
	s	s	f	**68.04**	**73.68**

Table B. Performance on CIFAR100, $N_s = 10$, $k = 1$, for different configurations of TAB matrices. 's': shared across tasks, 'f': flexible.

Cross-task attention in: l_q, l_k, l_v of MHSA	Cross-task attention in: MLP	$LA \uparrow$	$AA \uparrow$
✗	✗	58.22	67.63
✓	✗	66.29	72.42
✗	✓	**68.04**	**73.68**
✓	✓	67.93	73.46

Table C. Cross-task attention in MHSA and MLP

duces to an MLP across tasks when W_q, W_k, are eliminated and $A_p^{ij} = 1, \forall i, j$ in Equation (23) of main paper. We ablated the impact of sharing the different matrices. Table B shows that the DNE configuration has the best trade-off between accuracy and model size.

Cross-task attention in MHSA. MHSA and MLP are the main components of the transformer block. MHSA mainly learns spatial attentions between different patches while MLP fuse the features of different channels. In DNE, cross-task attention(CTA) is only equipped in MLP. However, there are still linear layers in the MHSA block, specifically the three linear layers l_q, l_k, l_v to generate query, key and value vectors in the self-attention blocks. By applying CTA to these linear layers, the MHSA could be able to first learn a feature that encodes information across tasks and then learn the spatial relationships. In Table C, we evaluate the effects of CTA in the linear layers of MHSA and MLP. The experiments are conducted on CIFAR100 with $N_s = 10$.

Cross-task attention in: FC_1 of MLP	Cross-task attention in: FC_2 of MLP	$LA \uparrow$	$AA \uparrow$
✗	✗	58.22	67.63
✓	✗	66.82	73.07
✗	✓	66.74	72.77
✓	✓	**68.04**	**73.68**

Table D. Cross-task attention in different linear layers MLP, FC_1 is the first linear layer of MLP and FC_2 is the second linear layer of MLP

Using CTA in MHSA and MLP can both significantly improve the performances. CTA in MHSA(MLP) increases the last accuracy by 8.07%(9.82%) and the average incremental accuracy by 4.79%(6.05%), compared to the model without CTA. But using CTA in both MHSA and MLP does not further boost the performances. Linear layers in MHSA and the MLP are doing similar things while MHSA is more focusing on spatial connections. Thus it is more natural to implement the CTA in MLP block only.

Cross-task attention in different layers of MLP. MLP has two linear layers in which CTA can be added. In Table D we evaluate the effects of CTA in these two linear layers. Experiments are conducted on CIFAR100 with $N_s = 10$. The results show that using CTA in either the first or the second linear layer can improve the last accuracy by

about 8.5% and the average incremental accuracy by about 5.5%. But by using CTA in both linear layers, the performances can be further improved. In DNE we use CTA in both linear layers to reach better performances.

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1

[2] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *European Conference on Computer Vision*, pages 86–102. Springer, 2020. 2

[3] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9285–9295, 2022. 1, 2

[4] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. *arXiv preprint arXiv:1910.09217*, 2019. 2

[5] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017. 2

[6] Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Foster: Feature boosting and compression for class-incremental learning. *arXiv preprint arXiv:2204.04662*, 2022. 2

[7] Zhen Wang, Liu Liu, Yiqun Duan, Yajing Kong, and Dacheng Tao. Continual learning with lifelong vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 171–181, 2022. 1

[8] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. *arXiv preprint arXiv:2204.04799*, 2022. 1

[9] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149, 2022. 1

[10] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3014–3023, 2021. 2