Task-Agnostic Guided Feature Expansion for Class-Incremental Learning

Supplementary Material

7. Additional Related Works

In Section 2, we discuss related works closely related to this work. Here we provide some additional related works in CIL. **Rehearsal memory** is used to store exemplars of previous tasks and replay at follow-up tasks. It makes the learned feature less forgetful by adjusting the input distribution towards the learned tasks. Many works focus on how to select exemplars [27, 34, 41, 50]. Exemplars can also be obtained by generative models [40].

Regularization methods [22, 39, 61] come from various ideas. [22] proposes to restrict the updates of important parameters. [61] proposes a dual augmentation framework to make the eigenvalues of the representation's covariance matrix larger. [39] proposes to make the representation scatter uniformly, making the representation contains more information about the input sample.

Model distillation uses the model trained on previous tasks as a teacher and distillation losses to keep the previously learned knowledge in the feature. LwF [25] proposes to use the response of the old model to guide the training of the new model's old tasks. PODNet [9] uses the pooled intermediate feature maps of the ResNet to be the distillation target in training.

Pre-Trained Model-based methods leverage pretrained models and adapt the model for class-incremental learning [47–49]. [49] uses visual prompt tuning [21] to learn a prompt for each task. [48] proposes the dual prompt scheme in ViT. Due to the head start of the pre-trained models in learning representations, these methods outperform the methods which train the model from scratch, even without the rehearsal memory samples.

Other perspectives to boost CIL are also considered. In the parameter space, [31] studies the linear mode connectivity in CIL and proposes to enhance the linear mode connectivity between learned models. [26] also considers the linear mode connectivity between learned models and proposes to combine two models learned in different ways to get better linear mode connectivity.

[7] uses Grad-CAM to generate attention maps for distillation. [13] generates adversarial samples iteratively to perform drift estimation for old class prototypes. [36] leverages the part information of the images, forming prototypical part layers on the model, improving the interpretability of the model. [56] proposes a locality-preserving attention module to remedy the locality degradation during the training of CIL. [57] considers enlarging the all-layer margin on the rehearsal samples, applying feature augmentations by input gradients.

8. More Implementation Details

In Section 5.1, we describe the experiment settings for the experiment section. Here we provide more implementation details. We provide more configurations about TagFex in Table 7. We use the same configurations for CIFAR and ImageNet unless additional specification. We use AutoAugment [5] for creating augmented samples in the training of the task-agnostic model. We perform our experiments on a server with 4 RTX3090 GPUs. For the pruning threshold, we adaptively choose the threshold according to the expected compression rate 0.4.

9. Performance Results with Standard Deviations.

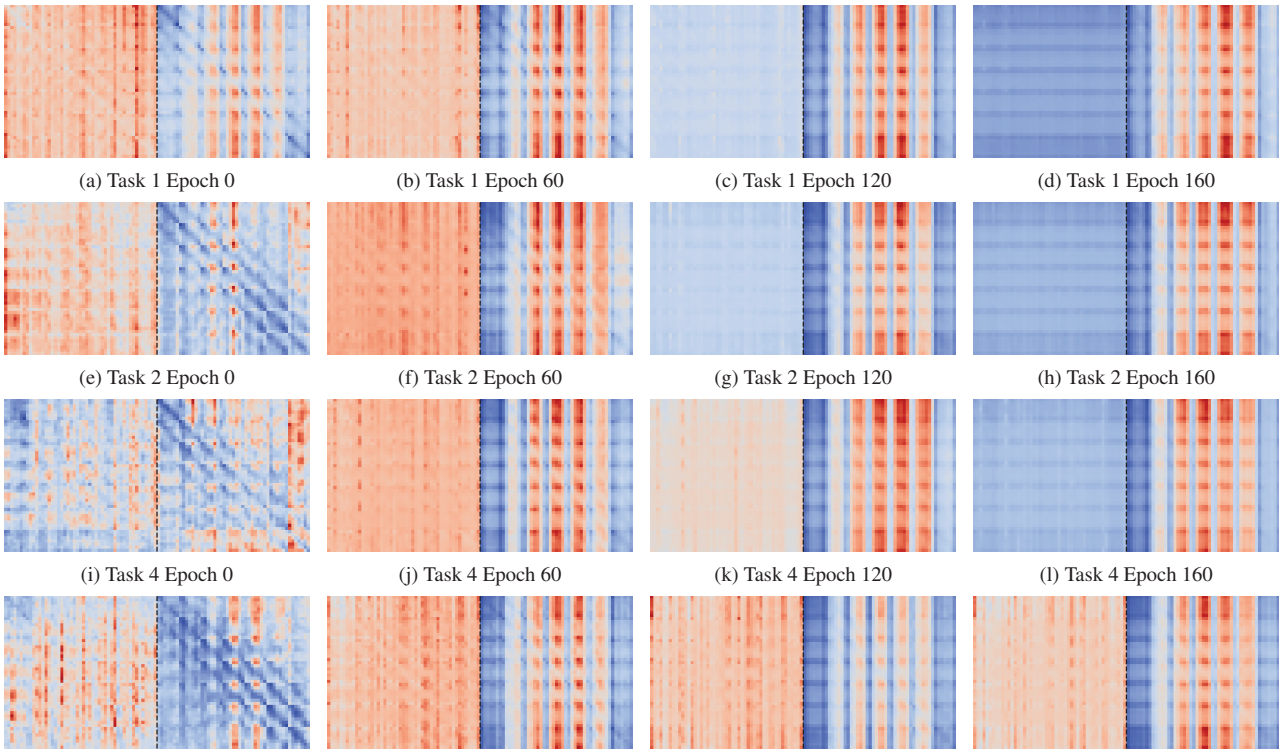
In Section 5.2, we compare the performance of TagFex with some baselines. Here, we provide performance results on CIFAR100 and ImageNet100 with standard deviations of 5 runs in Table 8.

10. Comparison for the Number of Parameters

In Section 5.2 and Table 1, we provide the performance results of pruned TagFex-P on various scenarios, and describe the efficacy on reducing the number of parameters. Here, we provide detailed comparison for the average number of parameters of the inference model. It is shown in Table 5. We compare the average incremental number of parameters for inference, which the values are averaged across each stages. As we can see, TagFex holds the same number of parameters as DER. With pruning techniques, TagFex-P requires less number of parameters to achieve comparable performance.

Table 5. Comparison for number of parameters. Values are shown in millions.

Methods	CIFAR100		ImageNet100	
	10-10	50-10	10-10	50-10
iCaRL [34]	0.46	0.46	11.2	11.2
BiC [50]	0.46	0.46	11.2	11.2
DyTox [10]	10.7	-	11.0	-
DER [51]	61.6	39.2	61.6	39.2
TagFex	61.6	39.2	61.6	39.2
TagFex-P	11.6	9.8	14.4	11.3



(a) Task 1 Epoch 0 (b) Task 1 Epoch 60 (c) Task 1 Epoch 120 (d) Task 1 Epoch 160

(e) Task 2 Epoch 0 (f) Task 2 Epoch 60 (g) Task 2 Epoch 120 (h) Task 2 Epoch 160

(i) Task 4 Epoch 0 (j) Task 4 Epoch 60 (k) Task 4 Epoch 120 (l) Task 4 Epoch 160

(m) Task 6 Epoch 0 (n) Task 6 Epoch 60 (o) Task 6 Epoch 120 (p) Task 6 Epoch 160

Figure 8. More visualizations on merge attention maps.

11. CKA Similarities of Learned Features in Expansion-based Methods

In Section 1, we compare the average CKA similarities of the learned features in expansion-based methods. Here we show the detailed numbers for the CKA similarities between each expanded feature. The results are shown in Figure 9, from which we can conclude that the features extracted by the models learned by DER are relatively similar, with maximum CKA 0.48 and minimum 0.27, especially when comparing to TagFex, with maximum 0.34 and minimum 0.14.

12. More Visualizations on Merge Attention Maps

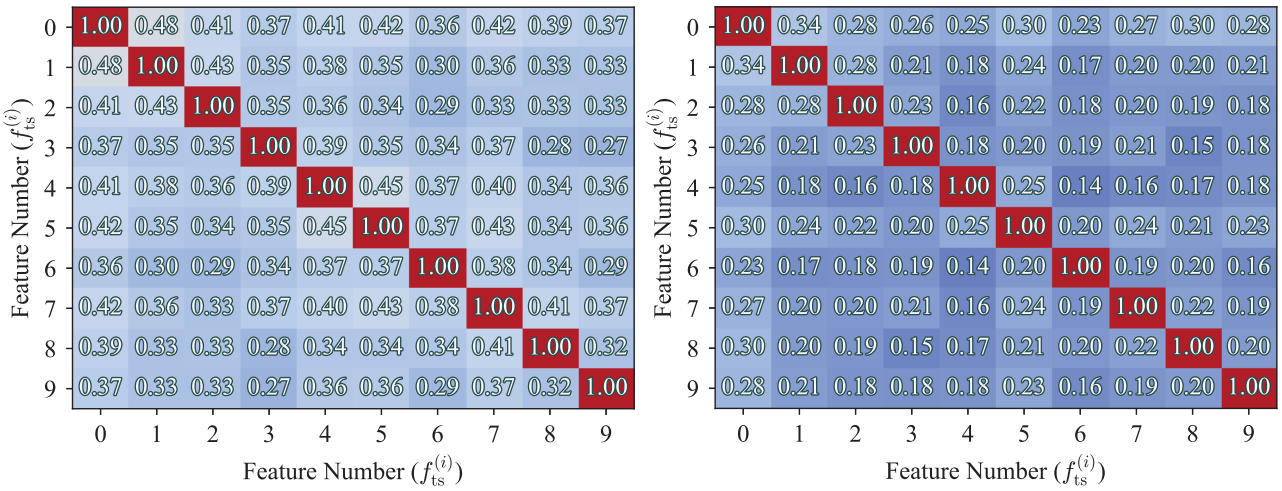
In Section 5.4, we discuss the evolution of the merge attention maps for TagFex. Here we provide more visualizations on merge attention maps in Figure 8. As we can see from the figures, in each incremental task, the high attention values are first on the task-agnostic side, and then transfer to the task-specific side, which represents task-agnostic features are useful for the task at early epochs, and such information is absorbed by the task-specific model. In addition, for later tasks (task 6 in the figure, sub-figure (m)(n)(o)(p)), the attention on the task-agnostic side at the end of the task is not as small as previous tasks (sub-figure (d)(h)(l)). It indicates that the task-agnostic model contains more and more features during the continual self-supervised learning.

13. Performance Experiments on Fine-grained Dataset

To verify the performance gain of TagFex on fine-grained datasets, we perform experiments on CUB200 100-20 with memory size 2000. The results are shown in Table 6, which shows strong performance gain over DER.

Table 6. Performance results on CUB200. TagFex outperforms DER on such fine-grained dataset.

Methods	CUB200 100-20	
	Avg	Last
DER	53.07	52.88
TagFex	56.56	54.37



(a) CKA feature similarities between each model learned by DER.

(b) CKA feature similarities between each model learned by TagFex.

Figure 9. CKA feature similarities. Features learned by TagFex are more diverse and less correlated.

Table 7. More implementation details for TagFex.

General Configurations		Task-Agnostic Model	
Base Epochs	200	InfoNCE Temperature τ	0.2
Incremental Epochs	170	Projection Hidden Dim.	2048
Batch Size	128	Projection Embedding Dim.	1024
Learning Rate	0.1	Predictor	Linear
Merge Attention # of heads	8	Transferring Temperature	2

Table 8. Performance results on CIFAR100 and ImageNet100 with standard deviations.

Datasets	Methods	10-10		50-10	
		Last	Avg	Last	Avg
CIFAR100	DER	64.35$\pm$0.11	75.36$\pm$0.06	65.27$\pm$0.10	72.60$\pm$0.05
	TagFex	68.23$\pm$0.13	78.45$\pm$0.06	70.33$\pm$0.13	75.87$\pm$0.06
	TagFex-P	67.34$\pm$0.18	78.02$\pm$0.10	69.26$\pm$0.15	74.24$\pm$0.08
ImageNet100	DER	66.71$\pm$0.16	77.18$\pm$0.08	71.08$\pm$0.11	77.71$\pm$0.06
	TagFex	70.84$\pm$0.19	79.27$\pm$0.11	75.54$\pm$0.11	80.64$\pm$0.07
	TagFex-P	69.21$\pm$0.17	78.56$\pm$0.10	74.13$\pm$0.12	79.85$\pm$0.07