Supplementary Materials of
A Unified Continual Learning Framework with General Parameter-Efficient Tuning

Qiankun Gao[1] Chen Zhao[†2] Yifan Sun[3] Teng Xi[3] Gang Zhang[3] Bernard Ghanem[2] Jian Zhang[†1]
[1]Peking University Shenzhen Graduate School [2]King Abdullah University of Science and Technology (KAUST) [3]Baidu Inc.
gqk@stu.pku.edu.cn chen.zhao@kaust.edu.sa zhangjian.sz@pku.edu.cn

In the supplementary materials, we further validate the proposed LAE framework by providing the following:

- Section A: Additional Experimental Details.
- Section B: Additional Experimental Results.
- Section C: Investigation on Prompt Learning and Selection from Pool in Prompt-Pool-based Approaches.

A. Additional Experimental Details

Data Augmentation. We adopt a very simple data augmentation strategy for training, following L2P [14] and DualPrompt [13]. *1)* Images are randomly resized to 224×224 using the bilinear interpolation algorithm. *2)* Images are normalized by min-max (for ViT [1]) or standard deviation (for Swin Transformer [7] and ConvNeXt [8]) normalization. *3)* Images are randomly flipped from horizontal. During inference, images are resized to 256×256 and cropped to 224×224 from central. All other approaches take the same data augmentation strategy as ours for fair comparisons. The PyTorch-like code is present in Algorithm 1.

Hyper-Parameter. Our LAE introduced two additional hyper-parameters, *i.e.*, the weight decay α of the Exponential Moving Average (EMA) algorithm and freezing epochs of the online Parameter-Efficient Tuning (PET) module. We did not intentionally search for these parameters and set α to a value very close to 1, such as the default value 0.9999 we used. The number of freezing epochs can be determined by the change in loss after freezing the online PET module and is typically set to the value where the loss no longer decreases. We set this value to 3 for CIFAR100 and scaled it proportionally for ImageNet-R, on all of which our LAE achieved superior performance than other competitors.

Training, Inference and Evaluation. The training and inference of our LAE framework are very easy to implement, the PyTorch-like pseudocode is provided in Algorithms 2. It is important to note that our evaluation metric A_{10} (Equation 15 in the paper) is slightly different from the following metric used by original L2P and DualPrompt:

$$A_{10} = \frac{1}{10}\sum_{j=1}^{10}\frac{1}{|\mathcal{D}_j^{test}|}\sum_{(\mathbf{x},y)\in\mathcal{D}_j^{test}}\mathbb{1}\left(\hat{y}=y\right), \qquad \text{(I)}$$

where $\mathcal{D}_j^{test}$ is the test set of the j^{th} task. We train and evaluate on three different class orders, while L2P, DualPrompt, and ESN [12] only evaluate on one class order in their original papers. Additionally, ESN uses a different pre-trained checkpoint from L2P and DualPrompt, but we correct this issue when using its code. The above differences lead to slightly different experimental results reported in their original papers from the data reported by us.

B. Additional Experimental Results

20-Task Benchmark Results. To further validate the efficacy of our LAE in longer-term Continual Learning scenarios, we split the CIFAR100 [4] and ImageNet-R [10] datasets into 20 tasks, each containing 5 (for CIFAR100) or 10 (for ImageNet-R) classes. We then conducted experiments and reported the mean and standard deviation of three runs in different class orders in Tables I and II. Similar to the 10-task experiments in the paper, we plot the task-by-task evaluation results in Figure Ia and Ib for the 20-task experiments on CIFAR100 and ImageNet-R. From these tables and figures, we can observe a wider performance gap between our LAE and other competitors compared to the 10-task experiments, suggesting that our LAE is more effective at mitigating forgetting and achieving a better stability-plasticity balance in longer-term Continual Learning.

Comparison with CODA-Prompt. The contemporary CODA-Prompt [11] approach demonstrates remarkable performance. Nevertheless, upon reviewing the authors'

† Corresponding authors.
Code is available at https://github.com/gqk/LAE.

Algorithm 1 Data Augmentation Code (PyTorch-like)

```
def build_train_transform(model):
   transforms = [T.RandomResizedCrop(224), T.RandomHorizontalFlip(), T.ToTensor()]
   if not isinstance(model, VisionTransformer):
      transforms.append(T.Normalize(mean=IMAGENET_MEAN, std=IMAGENET_STD))
   return T.Compose(transforms)

def build_inference_transform(model):
   transforms = [T.Resize(256), T.CenterCrop(224), T.ToTensor()]
   if not isinstance(model, VisionTransformer):
      transforms.append(T.Normalize(mean=IMAGENET_MEAN, std=IMAGENET_STD))
   return T.Compose(transforms)
```

Algorithm 2 Training and Inference Code (PyTorch-like)

```
# model: the pre-trained model; pet_on: online PET module; pet_off: offline PET module;

def train(model, pet_on, pet_off, dataloader, optimizer, task_id, alpha):
   model = attach(model, pet_on)
   for e in range(MAX_EPOCHS):
      if e == 0 and not_the_first_task(task_id):
         freeze(pet_on)
      elif e == NUM_FREEZING_EPOCHS:
         unfreeze(pet_on)

      for input, target in dataloader:
         pred = mask(model(input), task_id) # Eq. (8) in the paper
         loss = cross_entropy(pred, target) # Eq. (8) in the paper
         optimizer.zero_grad()
         loss.backward()
         optimizer.step()
         ema_update(pet_off, pet_on, alpha) # Eq. (13) in the paper

def inference(model, pet_on, pet_off, input):
   pred_on, pred_off = attach(model, pet_on)(input), attach(model, pet_off)(input)
   pred = max(softmax(pred_on, dim=-1), softmax(pref_off, dim=-1)) # Eq. (14) in the paper
   return argmax(pred)
```

Table I: 20-Task Benchmark Results on CIFAR100. The PET modules are inserted into the first 5 transformer blocks of the standard ViT-B/16 pre-trained on the ImageNet21k dataset. The "5, 10, 20" indicate the size of PET modules.

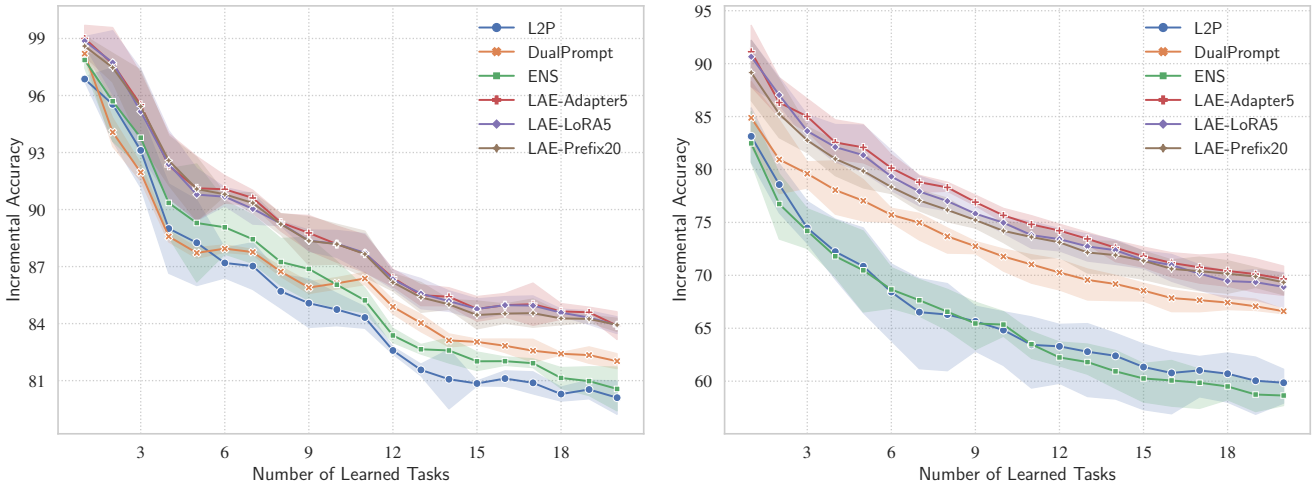
Approach	PET Module	A_{20} (↑)	$\bar{A}_{20}$ (↑)
L2P [14]	Prompt	80.10±0.72	85.29±0.50
DualPrompt [13]	Prefix20	82.02±0.32	89.50±0.11
ESN [12]	Prompt	80.56±0.94	90.47±1.19
LAE (Ours)	Adapter5	83.89±0.60	**92.35**±0.55
	Adapter10	83.81±0.35	92.32±0.57
	LoRA5	83.92±0.36	92.15±0.47
	LoRA10	83.35±0.20	91.71±0.88
	Prefix10	83.82±0.18	92.07±0.72
	Prefix20	**83.93**±0.28	92.21±0.53

Table II: 20-Task Benchmark Results on ImageNet-R. The PET modules are inserted into the first 5 transformer blocks of the standard ViT-B/16 pre-trained on the ImageNet21k dataset. The "5, 10, 20" indicate the size of PET modules.

Approach	PET Module	A_{20} (↑)	$\bar{A}_{20}$ (↑)
L2P [14]	Prompt	59.85±1.38	66.33±2.46
DualPrompt [13]	Prefix20	66.61±0.24	76.94±1.39
ESN [12]	Prompt	58.65±0.83	70.94±1.88
LAE (Ours)	Adapter5	69.66±1.16	81.69±1.00
	Adapter10	69.19±1.25	**81.78**±0.77
	LoRA5	68.91±1.40	80.99±1.17
	LoRA10	69.07±1.49	81.12±1.09
	Prefix10	**69.67**±0.86	79.97±0.97
	Prefix20	69.34±0.84	79.90±1.08

released code, we identified three potential sources of unfair comparison: 1) A distinct ImageNet-R train-test split in contrast to DualPrompt. 2) The model is pretrained on ImageNet-21k and subsequently fine-tuned on ImageNet-1K. 3) Varied training strategies, such as the number of epochs and learning rates. Our initial experiments reveal that when utilizing DualPrompt's train-test split, CODA-Prompt consistently underperforms our LAE. To ensure a fair evaluation, we adopt CODA-Prompt's settings for our experiments and extend our assessment to the DomainNet [9] dataset. All results are showcased in Table III, where we present average forgetting rates instead of average incremental accuracy.

Sensitive Analysis on EMA's Weight Decay. Weight decay α plays an important role in the knowledge accumulation of the offline PET module. A small value can lead to the integration of too much unstable new knowledge during the learning process, while a large value can result in the offline PET module being unable to effectively absorb new knowledge. In all of our experiments in the paper, we set the weight decay of EMA to 0.9999, which is the default value



(a) CIFAR100 (b) ImageNet-R

Figure I: Task-by-Task Incremental Accuracy on two 20-task benchmarks. The lines illustrate the task-by-task evaluation results of L2P [14], DualPrompt [13], ENS [12], and our LAE framework with different PET modules.

Table III: Comparison with CODA-Prompt on 5-task DomainNet Benchmark, 5- and 10-task ImageNet-R benchmarks. "A_N" and "F_N" are last incremental accuracy and last average forgetting for N-task benchmarks respectively.

Approach			Joint-FT	CODA-Prompt	LAE (Prefix10)
DomainNet	5-task	A_5 (↑)	74.91	67.11	68.37
		F_5 (↓)	-	13.79	8.33
ImageNet-R	5-task	A_5 (↑)	81.08	75.32	76.69
		F_5 (↓)	-	6.09	6.17
	10-task	A_{10} (↑)	81.08	74.31	74.43
		F_{10} (↓)	-	5.63	5.22

Table IV: The sensitiveness w.r.t. EMA's weight decay α.

α	0.999	0.9999	0.99999
A_{10} (↑)	71.40±1.02	72.66±0.63	72.58±0.40
$\bar{A}_{10}$ (↑)	78.04±1.03	78.91±0.89	78.67±0.94

in the timm library. Our experimental results in Table IV demonstrate that this value yields the best performance.

Memory and computation complexity. Our LAE requires two forward passes (one with $\boldsymbol{\theta}pet^{off}$ and the other with $\boldsymbol{\theta}pet^{on}$) per inference sample, yielding computational costs on par with L2P, DualPrompt, and the contemporary approach CODA-Prompt. Additionally, due to the constant number of parameters maintained across all tasks, LAE introduces fewer new parameters, as illustrated in Table V.

C. Prompt Learning and Selection from Pool

L2P [14] and DualPrompt [13] are two representative approaches that leverage prompt tuning [5] to address the problem of Continual Learning. L2P first proposes to use a pool to store prompts shared across tasks, where a set of prompts that match the sample are selected from the pool to predict the sample's label. In contrast, DualPrompt directly learns a set of task-specific E-Prompts for each task and stores them in the pool. During inference, the best-matched prompts (*i.e.*, the prompts learned for the task that the sample belongs to) are selected for the given sample.

https://github.com/huggingface/pytorch-image-models

Table V: The statistics of introduced parameters by approaches on 10-task benchmarks. "A10", "L10" and "P20" indicate Adapter10, LoRA10 and Prefix20 respectively.

Approach	DualPrompt	LAE (A10)	LAE (L10)	LAE (P20)
#Param. (M)	1.03	0.15	0.29	0.29

The performance of these approaches is influenced by two key factors. *1)* The ability to learn optimal prompts for each task is crucial for achieving better plasticity, *i.e.*, the ability to learn new knowledge. Better performance can be achieved only by sufficiently learning new knowledge while retaining as much previous knowledge as possible. *2)* the ability to accurately select the best-matched prompts for the inference sample is more critical. Because even if optimal prompts are learned for each task, inference using the wrong prompts can still result in poor prediction results. Following, we take DualPrompt as an example to investigate these two abilities of prompt-pool-based approaches.

To begin with, we assume that DualPrompt can learn the optimal E-prompts for each task. We then evaluate whether it can accurately select the right E-prompts during inference. As shown in Figure II, we observe that the E-prompts selected by DualPrompt are completely accurate after learning the first task. However, as the number of learned tasks increases, the accuracy of the prompt selection gradually decreases. By the time the 10th task is learned, the selec-

Table VI: Evaluation results on all tasks using 10 sets of task-specific E-Prompts. "#E-Prompts" denotes the index of the E-Prompts, *e.g.*, "1" indicates evaluation using the first task's E-Prompts.

#E-Prompts	1	2	3	4	5	6	7	8	9	10
A_{10} (↑)	63.78	66.93	67.57	67.88	68.48	68.38	68.80	68.88	68.87	68.47
$\bar{A}_{10}$ (↑)	69.11	72.54	72.60	72.63	72.69	72.68	72.72	72.73	72.73	72.69

Table VII: Evaluation results on each task using 10 sets of task-specific E-Prompts. "#E-Prompts" denotes the index of the E-Prompts, *e.g.*, "1" indicates evaluation using the first task's E-Prompts.

#E-Prompts	1	2	3	4	5	6	7	8	9	10
Task 1	69.64	73.27	72.77	72.94	73.76	73.10	72.61	72.94	72.44	72.28
Task 2	67.23	71.73	71.03	70.89	71.87	71.03	71.31	71.31	71.59	70.61
Task 3	63.93	67.16	70.90	71.89	70.40	69.90	70.65	70.40	69.40	69.90
Task 4	54.61	60.17	64.52	68.35	67.48	67.30	64.52	64.35	63.65	62.96
Task 5	61.01	64.52	65.64	65.36	68.16	68.16	67.04	65.50	65.22	65.78
Task 6	59.33	63.73	63.73	65.14	66.37	67.43	67.43	66.55	66.20	66.20
Task 7	57.02	58.93	58.41	57.71	57.89	57.71	61.18	62.22	62.39	60.49
Task 8	61.47	61.92	62.14	61.03	61.47	62.81	65.26	68.37	67.26	65.03
Task 9	74.85	76.50	74.70	75.15	75.30	74.70	77.25	77.10	78.89	77.40
Task 10	65.53	67.72	69.36	68.40	68.95	68.95	68.81	69.08	69.63	71.41

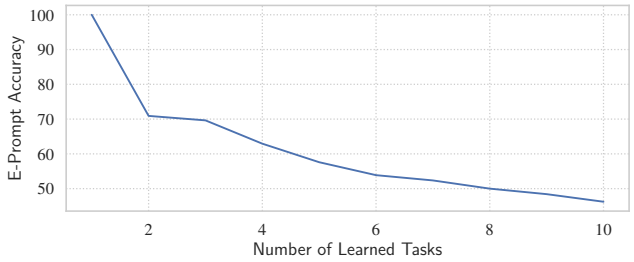


Figure II: The E-Prompts selection accuracy of DualPrompt on the test set of ImageNet-R.

tion accuracy drops to below 50%. Therefore, we conclude that if DualPrompt cannot address this issue, it is difficult to apply it to longer-term Continual Learning scenarios.

In addition, according to Figure 3 in our paper, we observe that DualPrompt performs worse than our LAE when learning the first task, regardless of whether our LAE uses Adapter [2] with fewer parameters, the LoRA [3] with the equivalent number of parameters, or Prefix [6] (*i.e.*, DualPrompt's E-Prompt) with slightly more parameters than DualPrompt. This indicates that Prompt/Prefix may not be as effective as Adapter and LoRA in learning new knowledge on these two datasets, as well as DualPrompt could not learn the optimal E-Prompts for the first task because they did not calibrate the Prefix like our LAE. This suggests that it is necessary to explore different Parameter-Efficient Tuning (PET) methods and calibrate PET modules.

Moreover, our naive baseline only uses one set of Prefixes, while DualPrompt learns a set of Prefixes for each task, totaling 10 sets, yet they achieve similar performance. We evaluate all 10 tasks using the 10 sets of E-Prompts learned by DualPrompt separately, and the results in Table VI show that the differences in the last and average incremental accuracy using the 2nd-10th sets of E-Prompts are very small. Table VII presents the prediction results of each task using each set of E-Prompts, for most tasks, the prediction results using the 2nd-10th sets of E-Prompts are very close. These analyses reveal that from the learning of the second task, task-specific E-Prompts tend to become homogeneous. According to our analysis in the paper, an important reason for this is that the adaptation speed of the Prefix is much slower than classifiers and other PET modules (*i.e.*, Adapter [2] and LoRA [3]).

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. 1

[2] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019. 4

[3] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *Pro-*

ceedings of the International Conference on Learning Representations (ICLR), 2022. 4

[4] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Technical Report*, 2009. 1

[5] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021. 3

[6] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021)*, 2021. 4

[7] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 1

[8] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1

[9] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1406–1415, 2019. 2

[10] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 2015. 1

[11] James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1

[12] Yabin Wang, Zhiheng Ma, Zhiwu Huang, Yaowei Wang, Zhou Su, and Xiaopeng Hong. Isolation and impartial aggregation: A paradigm of incremental learning without interference. 2023. 1, 2, 3

[13] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer G. Dy, and Tomas Pfister. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 1, 2, 3

[14] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer G. Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2, 3