Task Difficulty Aware Parameter Allocation & Regularization for Lifelong Learning - Supplementary

Wenjin Wang, Yunqing Hu, Qianglong Chen, Yin Zhang
Zhejiang University, Hangzhou, China
{wangwenjin,yunqinghu,chenqianglong,zhangyin98}@zju.edu.cn

A. Method

A.1. Algorithm

Algorithm 1: PAR

```
Input: Task sequence $T_1, \dots, T_N$; Parameters $\alpha, \beta$
1  Search cell and create a expert to learn task $T_1$.
2  for $t$ in $[2, N]$ do
3      Find the group $g^*$ with the smallest distance $s_{t,g^*}$.
4      if $s_{t,g^*} < \alpha$ then
5          Learn $T_t$ by expert $g^*$ with regularization.
6      else if $s_{t,g^*} < \beta$ then
7          Reuse cell and create a expert to learn $T_t$.
8      else
9          Search cell and create a expert to learn $T_t$.
10 end
```

We summarize the process of the PAR in the Algorithm 1.

A.2. Fine-grained Search

In our method, the number of experts and parameters are proportional to the number of task groups, mitigating the growth of parameter overhead. To further reduce the overhead of each expert, we adopt NAS to search for compact architectures for experts. Each *expert* in our method is stacked with multiple cells and the search for architecture is equivalent to the search for the appropriate cell. Since the time overhead of NAS becomes unbearable as the number of tasks increases, to improve the efficiency of architectural search in lifelong learning, we propose a relevance-aware sampling-based architecture search strategy. Specifically, as shown in Figure 2 of the origin paper, we construct a hierarchical search space. The coarse-grained search space contains cells used by existing experts and an unknown cell which will be searched from the fine-grained search space. Following the common practice [4,8], the fine-grained search space is a directed acyclic graph (DAG) with 7 nodes (two input nodes i_1, i_2, an ordered sequence of intermediate nodes n_1, n_2, n_3, n_4, and an output node). The input nodes are defined as the outputs in the previous two layers and the output is concatenated from intermediate nodes. Each intermediate node is connected to all of its predecessors by directed candidate edges, which are associated with 6 candidate operations that are efficient in terms of the number of parameters.

To search a cell for the new task, at first, we introduce a hyper-parameter β. When $s_{t,g^*} \leq \beta$, we directly reuse the cell of *expert* E_{g^*} for the task T_t. A task distance greater than α and less than β indicates that the new task is not enough to share the *expert* with group $\mathcal{G}_{g^*}$, but can use the same architecture. When $s_{t,g^*} > \beta$, we assign the unknown cell to the new *expert* and adopt an sampling-based NAS method MDL [8] to determine it. Specifically, for each candidate edge, we denote the probability, sampling epochs, and most recent performance of its candidate operations as P, $\mathcal{H}^e$, and $\mathcal{H}^a$ respectively, each of which is a real-valued column vector of length 8. We update them through multiple epochs of sampling to obtain a compact cell. In each epoch, to update $\mathcal{H}^e$ and $\mathcal{H}^a$, we sample an operation for each edge to form a cell and evaluate it by training a model stacked by it one epoch. Then, to update the probability P, we define the differential of sampling epochs as a 6×6 matrix $\Delta\mathcal{H}^e$ where $\Delta\mathcal{H}^e_{i,j} = \mathcal{H}^e_i - \mathcal{H}^e_j$. Similarly, we define the differential of performance as $\Delta\mathcal{H}^a$. The probability P is updated as follows:

$$P_i \leftarrow P_i + \gamma\Big(\sum_j \mathbb{I}(\Delta\mathcal{H}^e_{i,j} < 0, \Delta\mathcal{H}^a_{i,j} > 0) - \sum_j \mathbb{I}(\Delta\mathcal{H}^e_{i,j} > 0, \Delta\mathcal{H}^a_{i,j} < 0)\Big) \tag{1}$$

where γ is a hyper-parameter and $\mathbb{I}$ is the indicator function. The probabilities of operations with fewer sampling epochs and higher performance are enhanced and vice versa. For the final cell, operation with the highest probability in each edge is selected, then edges with top-2 probabilities for each intermediate node are used. The node value is equal to element-wise addition of results of these edges.

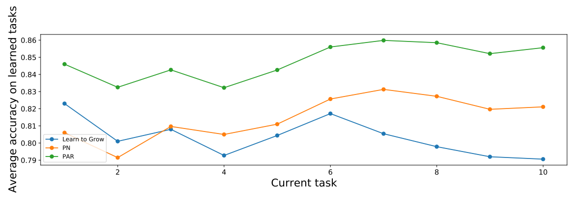


Figure 1. Average accuracy on learned tasks in CIFAR100-10.

B. Experiment

B.1. Benchmarks

To evaluate the performance of our method, we conduct experiments on multiple benchmarks for task incremental learning.

At first, we evaluate our method on two benchmarks containing a sequence of mixed similar and dissimilar tasks, including CTrL [5] and Mixed CIFAR100 and F-CelebA [2]. CTrL [5] includes 6 streams of visual image classification tasks. If t is a task in the stream, CTrL denotes a task as t^- and t^+ whose data is sampled from the same distribution as t, but with a much smaller or larger labeled dataset, respectively. Moreover, t' and t'' are tasks that are similar to task t, while there are no relation between t_i and t_j for all $i \neq j$. Then, the 6 streams in CTrl are as follows: $S^- = (t_1^+, t_2, t_3, t_4, t_5, t_1^-)$ is used to evaluate the ability of direct transfer; $S^+ = (t_1^-, t_2, t_3, t_4, t_5, t_1^+)$ is used to evaluate the ability of knowledge update; $S^{\text{in}} = (t_1, t_2, t_3, t_4, t_5, t_1')$ and $S^{\text{out}} = (t_1, t_2, t_3, t_4, t_5, t_1'')$ are used to evaluate the transfer to similar input and output distributions respectively; $S^{\text{pl}} = (t_1, t_2, t_3, t_4, t_5)$ is used to evaluate the plasticity; S^{long} consists of 100 tasks and is used to evaluate the scalability. Following the MNTDP [6], we list the details of streams in Table other than stream $\mathcal{S}^{\text{long}}$. The details of stream $\mathcal{S}^{\text{long}}$ are provided in the file "ctrl_long_info.txt" in supplementary. Similarly, Mixed CIFAR100 and F-CelebA [2] including mixed similar tasks from F-CelebA and dissimilar tasks from CIFAR100 [3]. F-CelebA consists of 10 tasks selected from LEAF [1] that containing images of a celebrity labeled by whether he/she is smiling or not. CIFAR100 is split to 10 tasks and each task has 10 classes.

Further, we conduct experiments on classical task incremental learning benchmarks including CIFAR10-5, CIFAR100-10, CIFAR100-20 and MiniImageNet-20. CIFAR10-5 is constructed by dividing CIFAR10 [3] into 5 tasks and each task has 2 classes. Similarly, CIFAR100-10 and CIFAR100-20 are constructed by dividing CIFAR100 [3] into 10 tasks with 10 classes and 20 tasks with 5 classes respectively. MiniImageNet-20 is constructed by dividing MiniImageNet [7] into 20 tasks and each task has 5 classes.

B.2. Comparison with Baselines

The performance of PAR on classical benchmarks CIFAR100-20 and CIFAR10-5 are listed in Tab. 2 and Tab. 3. Experimental results show that, compared with baselines, PAR can achieve better performance.

To track the performance of the model over time/task, the Fig. 1 presents the average accuracy curve on learned tasks in the CIFAR100-10

We also compare the time efficiency of our method with other methods on CIFAR100-10. Experimental results on Tab. 4 show that, thanks to the hierarchical architectural search, our method is faster than other approaches based on architecture search, such as LTG. Our method is faster than PN, which is a parameter allocation method without NAS based on dynamic model. The reason is that PN uses the entire model to learn each new task and the GPU memory, parameters, and time overhead of PN explode as the task number increases.

B.3. Ablation Study

We analyze the impact of hyper-parameter β and the results are listed in Tab. 5. We set the value of α to 0.5 and experimental results show that the performance of our method is stable when the value of β is different. As expected, the results show that β has a major impact on the time overhead of our approach because it affects the frequency of architecture search in the coarse-grained and fine-grained spaces.

C. More visualizations of task distance

To further analyze the validity of our approach, we present heat-maps of task distance in CIFAR100-10, CIFAR100-coarse and stream $\mathcal{S}^{\text{pl}}$. As shown in Fig. 2a, many tasks in CIFAR100-10 are relevant (< 0.5) to each other and can share the same expert. On the contrary, as shown in Fig. 2b, in the stream $\mathcal{S}^{\text{pl}}$, which contains several tasks that vary widely and is designed for evaluating the plasticity, the task distance obtained by our method is relatively large. The task distance in CIFAR100-coarse is shown in Fig. 3.

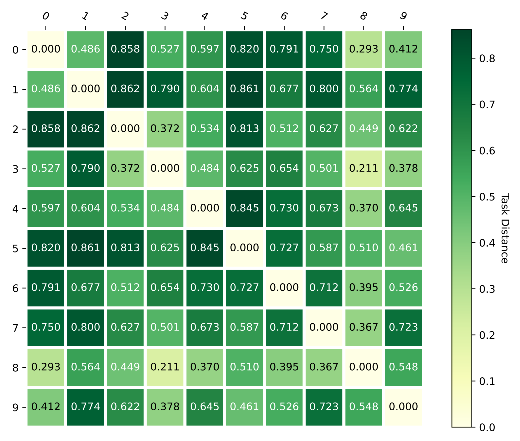
References

[1] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. LEAF: A Benchmark for Federated Settings. *arXiv:1812.01097 [cs, stat]*, Dec. 2019. 2

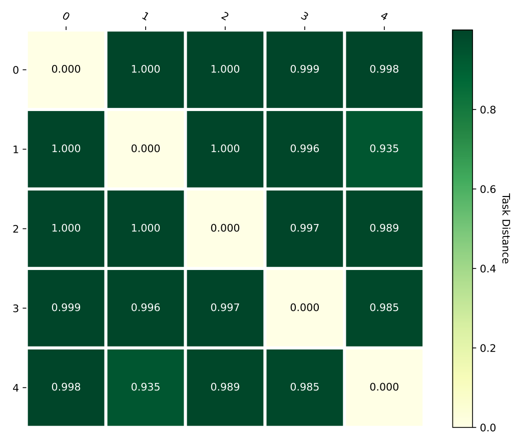
[2] Zixuan Ke, Bing Liu, and Xingchang Huang. Continual Learning of a Mixed Sequence of Similar and Dissimilar Tasks. In *NeurIPS 2020*, volume 33, pages 18493–18504. Curran Associates, Inc., 2020. 2

Stream		T_1	T_2	T_3	T_4	T_5	T_6
	Datasets	Cifar-10	MNIST	DTD	F-EMNIST	SVHN	Cifar-10
$\mathcal{S}^-$	# Train Samples	4000	400	400	400	400	400
	# Val Samples	2000	200	200	200	200	200
	Datasets	Cifar-10	MNIST	DTD	F-MNIST	SVHN	Cifar-10
$\mathcal{S}^+$	# Train Samples	400	400	400	400	400	4000
	# Val Samples	200	200	200	200	200	2000
	Datasets	R-MNIST	Cifar-10	DTD	F-MNIST	SVHN	R-MNIST
$\mathcal{S}^{\text{in}}$	# Train Samples	4000	400	400	400	400	50
	# Val Samples	2000	200	200	200	200	30
	Datasets	Cifar-10	MNIST	DTD	F-MNIST	SVHN	Cifar-10
$\mathcal{S}^{\text{out}}$	# Train Samples	4000	400	400	400	400	400
	# Val Samples	2000	200	200	200	200	200
	Datasets	MNIST	DTD	F-MNIST	SVHN	Cifar-10	
$\mathcal{S}^{\text{pl}}$	# Train Samples	400	400	400	400	4000	
	# Val Samples	200	200	200	200	2000	

Table 1. Details of streams in CTrL.



(a) CIFAR100-10



(b) $\mathcal{S}^{\text{pl}}$

Figure 2. The heat-maps of task distances.

Method	CIFAR100-20	
	AP	AF
Finetune	0.554	-0.259
EWC	0.556	-0.235
iCaRL	0.581	-0.242
A-GEM	0.544	-0.220
GCL	0.745	-0.065
ACL	0.781	0.000
FAS	0.808	0.000
PAR(ours)	0.861	-0.043

Table 2. Performance on CIFAR100-20.

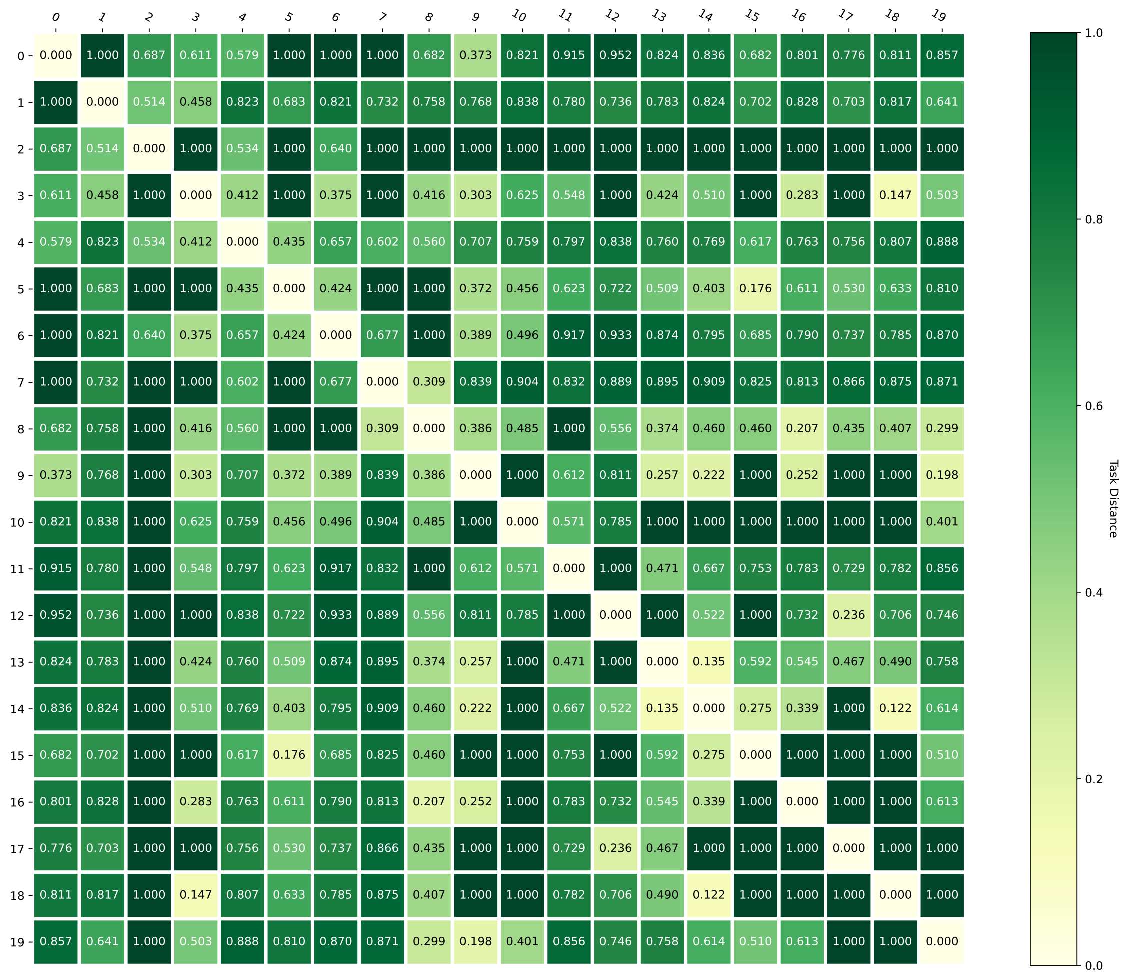
[3] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. 2

[4] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: DIFFERENTIABLE ARCHITECTURE SEARCH. In *ICLR 2019*, page 13, 2018. 1

[5] Tom Veniat, Ludovic Denoyer, and MarcAurelio Ranzato. Efficient Continual Learning with Modular Networks and Task-Driven Priors. In *ICLR 2021*, Sept. 2020. 2

[6] Vinay Kumar Verma, Kevin J. Liang, Nikhil Mehta, Piyush Rai, and Lawrence Carin. Efficient Feature Transformations for Discriminative and Generative Continual Learning. In *CVPR 2021*, pages 13865–13875, 2021. 2

[7] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching Networks for One Shot Learning. In *Advances in Neu-*



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0.000	1.000	0.687	0.611	0.579	1.000	1.000	1.000	0.682	0.373	0.821	0.915	0.952	0.824	0.836	0.682	0.801	0.776	0.811	0.857
1	1.000	0.000	0.514	0.458	0.823	0.683	0.821	0.732	0.758	0.768	0.838	0.780	0.736	0.783	0.824	0.702	0.828	0.703	0.817	0.641
2	0.687	0.514	0.000	1.000	0.534	1.000	0.640	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
3	0.611	0.458	1.000	0.000	0.412	1.000	0.375	1.000	0.416	0.303	0.625	0.548	1.000	0.424	0.510	1.000	0.283	1.000	0.147	0.503
4	0.579	0.823	0.534	0.412	0.000	0.435	0.657	0.602	0.560	0.707	0.759	0.797	0.838	0.760	0.769	0.617	0.763	0.756	0.807	0.888
5	1.000	0.683	1.000	1.000	0.435	0.000	0.424	1.000	1.000	0.372	0.456	0.623	0.722	0.509	0.403	0.176	0.611	0.530	0.633	0.810
6	1.000	0.821	0.640	0.375	0.657	0.424	0.000	0.677	1.000	0.389	0.496	0.917	0.933	0.874	0.795	0.685	0.790	0.737	0.785	0.870
7	1.000	0.732	1.000	1.000	0.602	1.000	0.677	0.000	0.309	0.839	0.904	0.832	0.889	0.895	0.909	0.825	0.813	0.866	0.875	0.871
8	0.682	0.758	1.000	0.416	0.560	1.000	1.000	0.309	0.000	0.386	0.485	1.000	0.556	0.374	0.460	0.460	0.207	0.435	0.407	0.299
9	0.373	0.768	1.000	0.303	0.707	0.372	0.389	0.839	0.386	0.000	1.000	0.612	0.811	0.257	0.222	1.000	0.252	1.000	1.000	0.198
10	0.821	0.838	1.000	0.625	0.759	0.456	0.496	0.904	0.485	1.000	0.000	0.571	0.785	1.000	1.000	1.000	1.000	1.000	1.000	0.401
11	0.915	0.780	1.000	0.548	0.797	0.623	0.917	0.832	1.000	0.612	0.571	0.000	1.000	0.471	0.667	0.753	0.783	0.729	0.782	0.856
12	0.952	0.736	1.000	1.000	0.838	0.722	0.933	0.889	0.556	0.811	0.785	1.000	0.000	1.000	0.522	1.000	0.732	0.236	0.706	0.746
13	0.824	0.783	1.000	0.424	0.760	0.509	0.874	0.895	0.374	0.257	1.000	0.471	1.000	0.000	0.135	0.592	0.545	0.467	0.490	0.758
14	0.836	0.824	1.000	0.510	0.769	0.403	0.795	0.909	0.460	0.222	1.000	0.667	0.522	0.135	0.000	0.275	0.339	1.000	0.122	0.614
15	0.682	0.702	1.000	1.000	0.617	0.176	0.685	0.825	0.460	1.000	1.000	0.753	1.000	0.592	0.275	0.000	1.000	1.000	1.000	0.510
16	0.801	0.828	1.000	0.283	0.763	0.611	0.790	0.813	0.207	0.252	1.000	0.783	0.732	0.545	0.339	1.000	0.000	1.000	1.000	0.613
17	0.776	0.703	1.000	1.000	0.756	0.530	0.737	0.866	0.435	1.000	1.000	0.729	0.236	0.467	1.000	1.000	1.000	0.000	1.000	1.000
18	0.811	0.817	1.000	0.147	0.807	0.633	0.785	0.875	0.407	1.000	1.000	0.782	0.706	0.490	0.122	1.000	1.000	1.000	0.000	1.000
19	0.857	0.641	1.000	0.503	0.888	0.810	0.870	0.871	0.299	0.198	0.401	0.856	0.746	0.758	0.614	0.510	0.613	1.000	1.000	0.000

Figure 3. Heat-map of task distance in CIFAR100-coarse.

ral Information Processing Systems, volume 29. Curran Associates, Inc., 2016. 2

[8] Xiawu Zheng, Rongrong Ji, Lang Tang, Baochang Zhang, Jianzhuang Liu, and Qi Tian. Multinomial Distribution Learning for Effective Neural Architecture Search. In *ICCV*, pages 1304–1313, 2019. 1

References

[1] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. LEAF: A Benchmark for Federated Settings. *arXiv:1812.01097 [cs, stat]*, Dec. 2019. 2

[2] Zixuan Ke, Bing Liu, and Xingchang Huang. Continual Learning of a Mixed Sequence of Similar and Dissimilar Tasks. In *NeurIPS 2020*, volume 33, pages 18493–18504. Curran Associates, Inc., 2020. 2

[3] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. 2

[4] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: DIFFERENTIABLE ARCHITECTURE SEARCH. In *ICLR 2019*, page 13, 2018. 1

[5] Tom Veniat, Ludovic Denoyer, and MarcAurelio Ranzato. Efficient Continual Learning with Modular Networks and Task-Driven Priors. In *ICLR 2021*, Sept. 2020. 2

[6] Vinay Kumar Verma, Kevin J. Liang, Nikhil Mehta, Piyush Rai, and Lawrence Carin. Efficient Feature Transformations for Discriminative and Generative Continual Learning. In *CVPR 2021*, pages 13865–13875, 2021. 2

[7] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching Networks for One Shot Learning. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. 2

[8] Xiawu Zheng, Rongrong Ji, Lang Tang, Baochang Zhang, Jianzhuang Liu, and Qi Tian. Multinomial Distribution Learning for Effective Neural Architecture Search. In *ICCV*, pages 1304–1313, 2019. 1

Method	CIFAR10-5	
	AP(%)	AF(%)
Finetune	0.530	-0.410
EWC	0.649	-0.202
IMM	0.761	-0.104
MAS	0.587	-0.089
LwF	0.867	-0.071
GPM†	0.886	-0.027
RPSnet	0.670	-
InstAParam	0.838	-
Independent	0.930	0.000
PN	0.925	0.000
Learn to Grow	0.912	0.000
BNS†	0.914	0.007
PAR	**0.946**	-0.006

Table 3. Performance on CIFAR10-5. †corresponds to performance based on AlexNet.

Method	Average Performance(AP)	Time(h)
Finetune	0.180	0.5
PN	0.821	1.6
Learn to Grow	0.791	3.0
PAR	0.849	0.9

Table 4. Comparison of time efficiency on CIFAR100-10.

β	Average Performance	$\mathcal{M}$(M)	Time(h)
0.5	0.841	12.306	1.4
0.7	0.844	12.147	1.4
1.0	0.849	12.165	0.9
1.2	0.849	12.213	1.0
1.5	0.846	12.165	0.9

Table 5. Ablation study of the hyper-parameter β. The α is set to 0.5.