Task Difficulty Aware Parameter Allocation & Regularization for Lifelong Learning

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

Abstract

Parameter regularization or allocation methods are effective in overcoming catastrophic forgetting in lifelong learning. However, they solve all tasks in a sequence uniformly and ignore the differences in the learning difficulty of different tasks. So parameter regularization methods face significant forgetting when learning a new task very different from learned tasks, and parameter allocation methods face unnecessary parameter overhead when learning simple tasks. In this paper, we propose the **P**arameter **A**llocation & **R**egularization (PAR)*, which adaptively select an appropriate strategy for each task from parameter allocation and regularization based on its learning difficulty. A task is easy for a model that has learned tasks related to it and vice versa. We propose a divergence estimation method based on the Nearest-Prototype distance to measure the task relatedness using only features of the new task. Moreover, we propose a time-efficient relatedness-aware sampling-based architecture search strategy to reduce the parameter overhead for allocation. Experimental results on multiple benchmarks demonstrate that, compared with SOTAs, our method is scalable and significantly reduces the model's redundancy while improving the model's performance. Further qualitative analysis indicates that PAR obtains reasonable task-relatedness.*

1. Introduction

Recently, the lifelong learning [9] ability of neural networks, i.e., learning continuously from a continuous sequence of tasks, has been extensively studied. It is natural for human beings to constantly learn and accumulate knowledge from tasks and then use it to facilitate future learning. However, classical models [13, 18, 41] suffer *catastrophic forgetting* [12], i.e., the model's performance on learned tasks deteriorates rapidly after learning a new one.

To overcome the catastrophic forgetting, many parameter regularization or allocation methods have been proposed. *Parameter regularization methods* [10, 16, 19, 21, 23, 25, 27] alleviate forgetting by adding a regularization term to the loss function and perform well when the new task does not differ much from learned tasks. *Parameter allocation methods* based on static models [7, 15, 31, 36] and dynamic models [2, 20, 24, 26, 29, 30, 34, 38, 39, 42, 45, 47] allocate different parameters to different tasks and can adapt to new tasks quite different from learned tasks. However, the above methods solve all tasks in a sequence uniformly, and ignore the differences of learning difficulty of different tasks. This leads to the significant forgetting in parameter regularization methods when learning a new task which is quite different from learned tasks, and also leads to unnecessary parameter cost in parameter allocation methods when learning some simple tasks.

In this paper, we propose a difficulty-aware method **P**arameter **A**llocation & **R**egularization (PAR). As shown

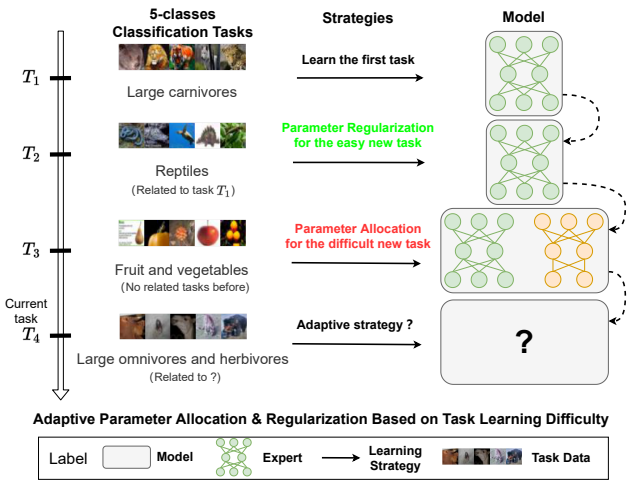


Figure 1. PAR adaptively selects a strategy from regularization and allocation to handle each task according to its learning difficulty. The difficulty depends not only on the task itself, but also on whether the model has previously learned tasks related to it.

*Corresponding author: Yin Zhang.

in Fig. 1, we assume that the learning difficulty of a task in continual learning depends not only on the task itself, but also on the accumulated knowledge in the model. A new task is easy to adapt for a model if it has learned related tasks before and vice versa. Based on the assumption, the PAR adaptively adopts parameter allocation for difficult tasks and parameter regularization for easy tasks. Specifically, the PAR divides tasks into *task groups* and assigns each group a dedicated *expert* model. Given a new task, the PAR measures the relatedness between it and existing task groups at first. If the new task is related to one of the existing groups, it is easy for the corresponding expert. The PAR adds the task to the related group and learns it by the expert via the parameter regularization. Otherwise, the new task is difficult for all existing experts, and the PAR assigns it to a new task group and allocates a new expert to learn it.

There are two challenges in this work: the measurement of relatedness and the parameter explosion associated with parameter allocation. For the first one, we try to measure the relatedness by the KL divergence between feature distributions of tasks. However, the KL divergence is intractable and needs to be estimated since the feature distributions of tasks are usually unknown. In addition, the constraint of lifelong learning that only data of the current task are available exacerbates the difficulty of estimation. To solve above problems, inspired by the divergence estimation based on k-NN distance [44], we propose the divergence estimation method based on prototype distance, which only depends on the data of the current task. For the second one, we try to reduce parameter overhead per expert by searching compact architecture for it. However, the low time and memory efficiency is an obstacle to applying architecture search for a sequence of tasks in lifelong learning. To improve the efficiency of architecture search, we propose a relatedness-aware sampling-based hierarchical search. The main contributions of this work are as follows:

- We propose a lifelong learning framework named Parameter Allocation & Regularization (PAR), which selects an appropriate strategy from parameter allocation and regularization for each task based on the learning difficulty. The difficulty depends on whether the model has learned related tasks before.
- We propose a divergence estimation method based on prototype distance to measure the distance between the new task and previous learned tasks with only data of the new task. Meanwhile, we propose a relatedness-aware sampling-based architecture search to reduce the parameter overhead of parameter allocation.
- Experimental results on CTrL, Mixed CIFAR100 and F-CelebA, CIFAR10-5, CIFAR100-10, CIFAR100-20 and MiniImageNet-20 demonstrate that PAR is scalable and significantly reduces the model redundancy while improving the model performance. Exhaustive ablation studies show the effectiveness of components in PAR and the visualizations show the reasonability of task distance in PAR.

2. Related Work

2.1. Lifelong Learning

Many methods have been proposed to overcome catastrophic forgetting. *Replay methods* try to replay samples of previous tasks when learning a new task from an *episodic memory* [32,40] or a *generative memory* [3,28,37]. *Parameter regularization methods*, including the *prior-focused regularization* [16, 19, 23, 27] and the *data-focused regularization* [10, 21, 25], try to alleviate forgetting by introducing a regularization term in the loss function of the new task. *Parameter allocation methods* based on the *static model* [7,15,31,36] and the *dynamic model* [2,20,24,26,29,30,34,38,39,42,45,47] overcome catastrophic forgetting by allocating different parameters to different tasks.

Methods with relatedness. Several methods also consider the utility of task relatedness [2, 15]. Expert Gate [2] assigns dedicated experts and auto-encoders for tasks and calculates the task relatedness by reconstruction error of auto-encoders. The task relatedness is used to transfer knowledge from the most related previous task. CAT [15] defines the task similarity by the positive knowledge transfer. It focuses on selectively transferring the knowledge from similar previous tasks and dealing with forgetting between dissimilar tasks by hard attention. In this paper, we propose a divergence estimation method based on prototype distance to calculate the task relatedness used to measure the task learning difficulty.

Methods with NAS. Static model based methods [7,31] try to search a sub-model for each task with neural architecture search (NAS). Dynamic model based methods [20, 42, 46] adopt NAS to select an appropriate model expansion strategy for each task and face the high GPU memory, parameter, and time overhead. Instead, in this paper, we propose a relatedness-aware hierarchical cell-based architecture search to search compact architecture for each expert with a lower GPU memory and time overhead. [8,14] adopt NAS on multi-task learning.

2.2. Cell-based NAS

Neural architecture search (NAS) [11,33] aims to search for efficient neural network architectures from a pre-defined search space in a data-driven manner. To reduce the size of the search space, *cell-based* NAS methods [22, 48, 49] try to search for a cell architecture from a pre-defined cell search space, where a cell is a tiny convolutional network mapping an $H \times W \times F$ tensor to another $H' \times W' \times F'$. The final model consists of a pre-defined number of stacked

cells. The cell in NAS is similar to the residual block in the residual network (ResNet), but its architecture is more complex and is a directed acyclic graph (DAG). The operations in the search space are usually parameter-efficient. NAS methods usually produce more compact architectures than hand-crafted designs.

3. Method

We focus on the task-incremental scenario of lifelong learning. Specifically, the model learns a sequence of tasks $\mathcal{T} = \{T_1, \ldots, T_t, \ldots, T_N\}$ one by one and the task id of sample is available during both training and inference. Each task T_t has a training dataset, $D^t_{train} = \{(x^t_i, y^t_i); i = 1, \ldots, n^t_{train}\}$, where y^t_i is the true label and n^t_{train} is the number of training examples. Similarly, we denote the validation and test dataset of task T_t as D^t_{valid} and D^t_{test}.

As shown in Fig. 2, the PAR divides previously learned tasks into task groups $\mathcal{G}$ and assigns each group $\mathcal{G}_i$ a dedicated expert model E_i. For a new task, PAR calculates the distances, which reflect the relatedness, between it and existing task groups at first. Then, it adopt an appropriate learning strategy from parameter allocation and regularization for the new task based on the distances.

3.1. Task Distance via Nearest-Prototype Distance

The PAR calculates the distance between the new task and each existing task at first, then it averages the distances of tasks in the same group as the distance between the new task and the group.

We calculate the distance between two tasks by the KL divergence between their distributions. However, the KL divergence is intractable because the true distributions of tasks are usually agnostic. Instead, we estimate the KL divergence by the features of data. To enhance estimation stability, we introduce an extra pre-trained extractor[1] to generate a robust feature X^t_i for each image x^t_i in task T_t. *The pre-trained extractor is only used for divergence estimation and does not affect the learning of tasks.* The number of extra parameters introduced by the model is a constant and does not affect the scalability of our method.

k-Nearest Neighbor Distance Given the features of distributions, a classical divergence estimation method for multidimensional densities is the k-Nearest Neighbor (k-NN) distance [44]. Suppose the feature distributions of the new task T_i and an existing task T_j are q_i and q_j respectively, and the features $X^i = \{X^i_1, \ldots, X^i_n\}$ and $X^j = \{X^j_1, \ldots, X^j_m\}$ are drawn independently from distributions q_i and q_j, where n and m are the numbers of samples. The k-NN distance of feature X^i_l in X^i and X^j are denoted by $\nu_k(l)$ and $\rho_k(l)$ respectively. Specifically, the $\rho_k(l)$ is the Euclidean distance from X^i_l to its k-NN in $\{X^i_z\}_{z \neq l}$. Similarly, the $\nu_k(l)$ is the distance from X^i_l to its k-NN in X^j. Then, the KL divergence between distributions q_i and q_j are estimated as follows:

$$KL(q_i||q_j) \approx \widehat{KL}(q_i||q_j) = \frac{d}{n}\sum_{l=1}^{n} \log \frac{\nu_k(l)}{\rho_k(l)} + \log \frac{m}{n-1}, \tag{1}$$

where d is the feature dimension.

The k-NN distance is asymmetric and this is consistent with the asymmetry of the KL divergence. The motivation behind Eq. (1) is that the k-NN distance is the radius of a d-dimensional open Euclidean ball used for k-NN density estimation. Readers can refer to [44] for more details such as convergence analysis.

Nearest-Prototype Distance However, the calculation of the k-NN distance involves features of two tasks T_i and T_j and violates the constraint of lifelong learning that only data of the current task T_i are available. To overcome this problem, we propose the Nearest-Prototype distance to replace the k-NN distance. Specifically, suppose the set of classes in task T^i is C^i, for each class $c \in C^i$, we maintain a prototype feature U^i_c, which is the mean of features of samples belonging to the class c. Then, the Nearest-Prototype distance of X^i_l to X^i is defined as follows:

$$\rho(l) = ||X^i_l - U^i_{c(l)}||, \tag{2}$$

where $||\cdot||$ is the Euclidean distance and $c(l)$ is the class of X^i_l. Similarly, we denote the set of classes in task T_j by C^j and the prototype features of X^j for each class $c \in C^j$ by U^j_c. Then, the Nearest-Prototype distance of X^i_l to X^j is denoted as

$$\nu(l) = \min_c ||X^i_l - U^j_c||_2, c \in C^j. \tag{3}$$

Then, by replacing the k-NN distances $\nu_k(l)$ and $\rho_k(l)$ in Eq. (1) by the the Nearest-Prototype distances $\nu(l)$ and $\rho(l)$, the KL divergence estimated by the Nearest-Prototype distance is as follows:

$$KL(q_i||q_j) \approx \widetilde{KL}(q_i||q_j) = \sum_{c \in C^i} \frac{1}{n_c} \sum_{l=1}^{n_c} \log \frac{\nu(l)}{\rho(l)}, \tag{4}$$

where constant terms are omitted because we are only concerned with the relative magnitude of KL divergence. The Eqs. (2) to (4) only involve features of the new task T_i and prototypes of the existing task T_j, which satisfies the constraint of lifelong learning. The storage cost of prototypes is small and negligible.

The motivation behind Eq. (4) is intuitive. The Nearest-Prototype distance $\rho(l)$ reflects the relationship between the

[1] We adopt a ResNet18 pre-trained on the ImageNet. It does not introduce unfairness because the generated features are not involved in model training and prediction.

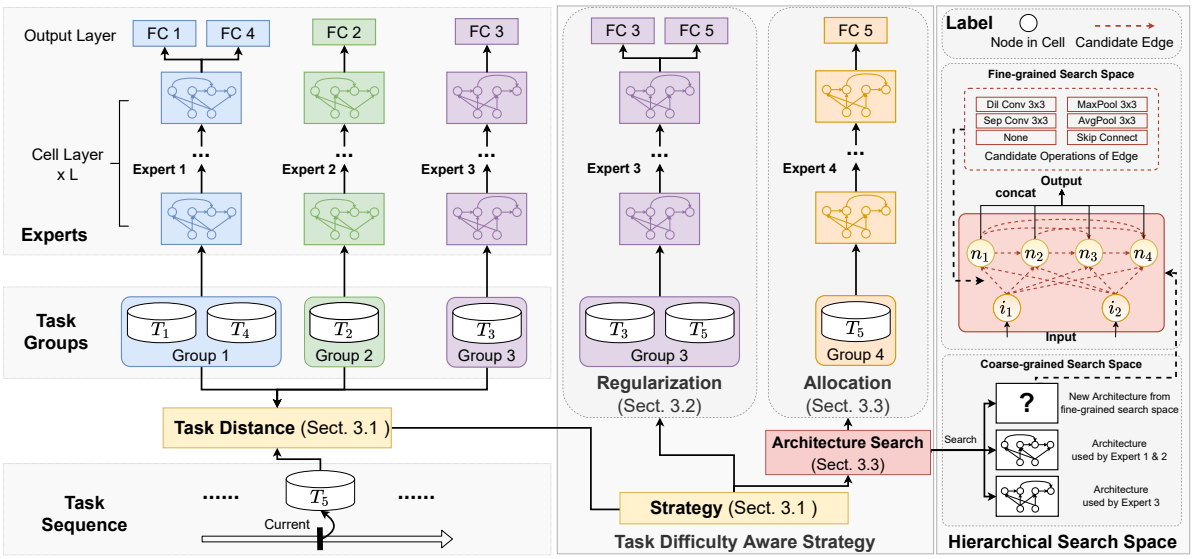


Figure 2. **The architecture of PAR.** PAR divides previously learned tasks into task groups and each group has a dedicated expert model. Given a new task T_5, PAR calculates the distance between it and existing task groups to measure its learning difficulty. Then, PAR learns the new task by parameter regularization if it is easy to learn. Otherwise, PAR learns the new task by parameter allocation and search compact architecture for each expert by a relatedness-aware hierarchical architecture search.

sample X_l^i in task T_i and its class prototype in feature space. The Nearest-Prototype distance $\nu(l)$ reflects the relationship between the sample X_l^i and class prototypes of existing task T_j in feature space. If the value of $\rho(l)$ and $\nu(l)$ are equal for each X_l^i, the class prototypes of two tasks are close. In this case, the distribution of the two tasks are similar, and the estimated KL divergence by Eq. (4) is 0.

Adaptive Learning Strategy Given the distance between the new task T_i and each existing tasks by KL divergence, we denote the distance $s'_{i,g}$ between the task T_i and a task group $\mathcal{G}_g$ by the average of distances of tasks in the group as follows:

$$s'_{i,g} = \frac{1}{|\mathcal{G}_g|} \sum_{j'}^{|\mathcal{G}_g|} KL(q_i || q_{j'}), \tag{5}$$

where q_i and $q_{j'}$ represent feature distributions of task T_i and the j'-th task in group $\mathcal{G}_g$. The $s'_{i,g} \in [0, \infty)$ since the range of the KL divergence is $[0, \infty)$. However, we try to use the distance to reflect the relatedness, which is usually measured by a value between 0 and 1. So we map the $s'_{i,g}$ into $[0, 1]$ by a monotone increasing function as follows:

$$s_{i,g} = \min(s'_{i,g}, 1 - e^{-2 \times s'_{i,g}}), \tag{6}$$

where e is the Euler's number.

Suppose the smallest distance between the new task T_t and existing groups is s_{t,g^*} and the corresponding task group is $\mathcal{G}_{g*}$, the PAR selects the learning strategy by comparing the s_{t,g^*} with a hyper-parameter α. If $s_{t,g^*} \leq \alpha$, the new task is added to this related group $\mathcal{G}_{g^*}$ and learned by parameter regularization. Otherwise, the new task is assigned to a new group and learned by parameter allocation because no existing task group is related to it.

3.2. Parameter Regularization

A new task T_t is easy for the expert model E_{g^*} if it is related to the group G_{g^*}. PAR reuses the expert E_{g^*} to learn this task by parameter regularization. Inspired by LwF [21], we adopt a parameter regularization method based on knowledge distillation. Specifically, the loss function consists of a training loss $\mathcal{L}_{\text{new}}$ and a distillation loss $\mathcal{L}_{\text{old}}$. The training loss encourages expert E_{g^*} to adapt to the new task T_t and is the cross-entropy for classification as follows:

$$\mathcal{L}_{\text{new}} = - \sum_{(x,y) \in D^t_{train}} \log(p_{E_{g^*}}(y|x)). \tag{7}$$

The distillation loss encourages the expert E_{g^*} to maintain performance on previous tasks in the group G_{g^*}. To calculate it, we record the logits $\mathbf{y}^j$ of the output head of each previous task T_j for each sample x of task T_t. The distillation loss is as follows:

$$\mathcal{L}_{\text{old}} = - \sum_{(x,y) \in D^t_{train}} \sum_{T_j \in \mathcal{G}_{g^*}} \mathbf{y}^j \cdot \log(\mathbf{p}_{\mathbf{E}_{g^*}}(\mathbf{x})^j), \tag{8}$$

where $\mathbf{y}^j$ and $\mathbf{p}_{\mathbf{E}_{g^*}}(\mathbf{x})^j$ are vectors with lengths equal to the number of categories of the previous task T_j. The total loss is

$$\mathcal{L}_{\text{PR}} = \mathcal{L}_{\text{new}} + \lambda \mathcal{L}_{\text{old}}, \tag{9}$$

	Method	$\mathcal{S}^-$		$\mathcal{S}^+$		$\mathcal{S}^{\text{in}}$		$\mathcal{S}^{\text{out}}$		$\mathcal{S}^{\text{pl}}$		$\mathcal{S}^{\text{long}}$	
		AP	AF	AP	AF	AP	AF	AP	AF	AP	AF	AP	AF
	Finetune	0.180	-0.330	0.240	-0.230	0.180	-0.310	0.150	-0.370	0.210	-0.300	0.200	-0.400
(1)	EWC	0.540	-0.010	0.370	-0.040	0.430	-0.120	0.510	-0.030	0.300	-0.170	0.270	-0.300
(2)	ER	0.410	-0.130	0.430	-0.070	0.380	-0.170	0.540	-0.070	0.450	-0.080	-	-
(3)	HAT*	0.570	-0.010	0.570	0.000	0.580	-0.010	0.600	0.000	0.580	0.000	0.240	-0.100
(4)	Independent	0.560	0.000	0.570	0.000	0.570	0.000	0.610	0.000	0.590	0.000	0.570	0.000
	PNN	0.620	0.000	0.520	0.000	0.570	0.000	0.620	0.000	0.540	0.000	-	-
	SG-F	0.636	0.000	0.615	0.000	0.655	0.000	0.641	0.000	0.620	0.000	-	-
	MNTDP-S	0.630	0.000	0.560	0.000	0.570	0.000	0.640	0.000	0.550	0.000	0.680	0.000
	MNTDP-D	0.670	0.000	0.610	0.000	0.600	0.000	0.680	0.000	0.620	0.000	0.750	0.000
	LMC	0.666	0.000	0.601	-0.014	0.695	0.000	0.667	-0.010	0.616	-0.035	0.639	-
	PAR(our)	**0.706**	0.000	**0.670**	-0.017	**0.699**	0.000	**0.694**	0.000	**0.700**	0.000	**0.773**	-0.016

Table 1. The average performance (AP) and average forgetting (AF) on the six streams ($\mathcal{S}^-$, $\mathcal{S}^+$, $\mathcal{S}^{\text{in}}$, $\mathcal{S}^{\text{out}}$, $\mathcal{S}^{\text{pl}}$, $\mathcal{S}^{\text{long}}$) in the CTRL benchmark. We compare the PAR with four types of baselines: (1) parameter regularization, (2) replay, (3) parameter allocation with static model, and (4) parameter allocation with dynamic model. The * corresponds to model using the AlexNet backbone.

where λ is a hyper-parameter to balance training and distillation loss. *Note that our method is without memory and does not require storing samples for previous tasks.*

However, the expert E_{g^*} may over-fit the new task whose sample size is far less than tasks in the group $\mathcal{G}_{g^*}$, even though the new task is related to the group. This leads to interference with previously accumulated knowledge in the expert E_{g^*}. To avoid the above problem, PAR records the maximum sample size of tasks in each task group. Suppose the maximum sample size in the group $\mathcal{G}_{g^*}$ is Q, PAR freezes the parameters of expert E_{g^*} except for the task-specific classification head during the learning if the sample size of the new task is less than 10 percent of Q. By transferring the existing knowledge in expert E_{g^*}, only optimizing the classification header is sufficient to adapt to the new task because the new task is related to the group $\mathcal{G}_{g^*}$.

3.3. Parameter Allocation

If no existing groups are related to the new task T_t, PAR assigns it to a new task group $\mathcal{G}_{M+1}$ with a new expert E_{M+1}, where M is the number of existing groups. We adopt the cross-entropy loss for the task T_t as follows:

$$\mathcal{L}_{\text{PA}} = -\sum_{(x,y)\in D^t_{train}} \log(p_{E_{M+1}}(y|x)). \tag{10}$$

The number of experts and parameters in PAR is proportional to the number of task groups, mitigating the growth of parameters. To further reduce the parameter overhead of each expert, we adopt cell-based NAS (refer to Sec. 2.2 for details) to search for a compact architecture for it.

Each expert in PAR is stacked with multiple cells and the search for expert architecture is equivalent to the search for the cell architecture. Since the time overhead of NAS becomes unbearable as the number of tasks increases, we propose a relatedness-aware sampling-based architecture search strategy to improve efficiency.

As shown in Fig. 2, we construct a hierarchical search space. The coarse-grained search space contains cells used by existing experts and an unknown cell whose architecture will come from the fine-grained search space. Following common practice [22, 48], the fine-grained search space is a directed acyclic graph (DAG) with seven 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 represent the outputs of the previous two layers. The output is concatenated from all intermediate nodes. Each intermediate node has a directed candidate edge from each of its predecessors. Each candidate edge is associated with six parameter-efficient candidate operations.

To search a cell for the new task, we introduce a hyper-parameter β. If $s_{t,g^*} \leq \beta$, PAR reuses 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 related enough to share the same expert with tasks in the group $\mathcal{G}_{g^*}$ but can use the same architecture. If $s_{t,g^*} > \beta$, PAR assigns the unknown cell to the new expert and adopts a sampling-based NAS method named MDL [48] to determine its architecture. Due to limited space, we leave the details of the MDL in the supplementary.

4. Experiments

4.1. Experimental Settings

Benchmarks We adopt two benchmarks the CTrL [42] and the Mixed CIFAR100 and F-CelebA [15], which con-

Method	Finetune	HAT	CAT	PAR
AP	0.6155	0.6178	0.6831	**0.7122**

Table 2. We compare the average performance (AP) of PAR with two parameter allocation methods based on static model on the mixed CIFAR100 and F-CelebA benchmark.

#	Method	CIFAR100-10		MiniImageNet-20	
		AP(%)	AF(%)	AP(%)	AF(%)
	Finetune	0.280	-0.560	0.378	-0.314
1	EWC	0.608	-0.061	0.574	-0.144
	MAS	0.579	-0.050	0.368	-0.238
	LwF	0.655	-0.088	0.589	-0.169
2	GPM*	0.725	-	0.604	-
	A-GEM	0.677	-0.120	0.524	-0.152
3	Independent	0.820	0.000	0.787	0.000
	PN	0.821	0.000	0.781	0.000
	Learn to Grow	0.791	0.000	-	-
	MNTDP-S	0.750	-0.000	-	-
	MNTDP-D	0.830	-0.000	-	-
	PAR	**0.853**	-0.020	**0.816**	-0.022

Table 3. PAR outperforms (1) parameter regularization methods, (2) replay methods, and (3) parameter allocation methods with dynamic model on classical benchmarks. The * corresponds to model using the AlexNet backbone.

tain mixed similar and dissimilar tasks. CTrL [42] includes 6 streams of image classification tasks: S^- is used to evaluate the ability of direct transfer, S^+ is used to evaluate the ability of knowledge update, S^{in} and S^{out} are used to evaluate the transfer to similar input and output distributions respectively, S^{pl} is used to evaluate the plasticity, S^{long} consists of 100 tasks and is used to evaluate the scalability. Similarly, Mixed CIFAR100 and F-CelebA [15] including mixed 10 similar tasks from F-CelebA and 10 dissimilar tasks from CIFAR100 [17].

Further, we adopt classical task incremental learning benchmarks: CIFAR10-5, CIFAR100-10, CIFAR100-20 and MiniImageNet-20. CIFAR10-5 is constructed by dividing CIFAR10 [17] into 5 tasks and each task has 2 classes. Similarly, CIFAR100-10 and CIFAR100-20 are constructed by dividing CIFAR100 [17] into 10 10-classification tasks and 20 5-classification tasks respectively. MiniImageNet-20 is constructed by dividing MiniImageNet [43] into 20 tasks and each task has 5 classes. We leave the results on the CIFAR10-5 and CIFAR100-20 in the supplementary.

Baselines Firstly, we compare our method with two simple baselines Finetune and Independent. While Finetune learns tasks one-by-one without any constraints, Independent builds a model for each task independently. Then, we present parameter regularization methods, including EWC [16], LwF [21], and MAS [1], and replay methods, including ER [6], GPM [35], and A-GEM [5]. Meanwhile, we compare parameter allocation methods with the static model, including HAT [36] and CAT [15], and parameter allocation methods with the dynamic model, including PN [34], Learn to Grow [20], SG-F [24], MNTDP [42], LMC [29] with PAR.

Method	CIFAR100-10 (10 tasks)		S^{long} (100 tasks)	
	AP	$\mathcal{M}$(M)	AP	$\mathcal{M}$(M)
Finetune	0.180	0.607	0.200	0.607
Independent	0.820	6.070	0.570	60.705
PN	0.821	12.422	-	-
Learn to Grow	0.791	1.926	-	-
MNTDP-S	0.750	6.100	0.680	39.658
MNTDP-D	0.830	6.075	0.750	25.508
PAR	**0.853**	1.400	**0.773**	13.475

Table 4. Compared with existing parameter allocation methods based on dynamic model, PAR obtains better average performance (AP) with fewer model parameters ($\mathcal{M}$, M=1e6) .

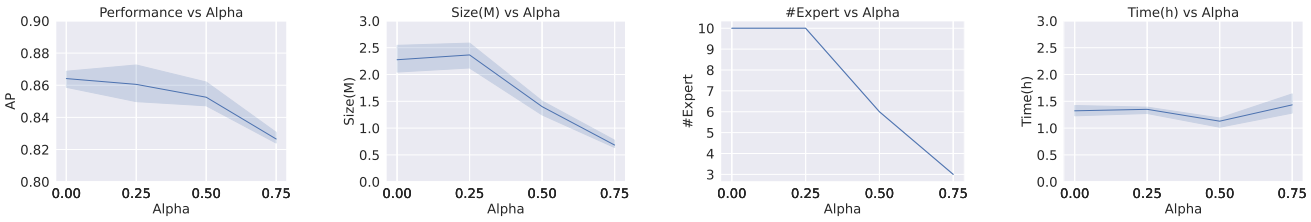
#	Allocation Strategy			Reg.	AP	$\mathcal{M}$(M)	T(h)
	Fixed	FS	CS				
1	✓				0.855	2.236	1.0
2		✓			0.861	2.277	1.9
3		✓	✓		0.870	1.900	1.2
4				✓	0.775	0.244	1.0
5	✓			✓	0.834	1.129	1.0
6		✓		✓	0.856	1.210	1.4
PAR		✓	✓	✓	0.853	1.400	1.1

Table 5. Ablation studies of components in PAR on the CIFAR100-10. The Fixed represents using a fixed cell from DARTs. The FS and CS represent fine-grained and coarse-grained search space respectively. The Reg. represents the parameter regularization. The PAR makes the trade off among average performance (AP), parameter overhead ($\mathcal{M}$), and time overhead (T).

Metrics Denote the performance of the model on T_j after learning task T_i as $r_{i,j}$ where $j \leq i$. Suppose the current task is T_t, the *average performance (AP)* and *average forgetting (AF)* are as follows:

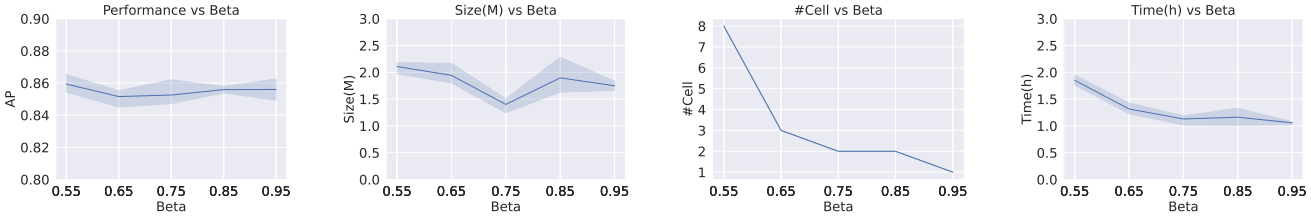
$$AP = \frac{1}{t}\sum_{j=1}^{t} r_{t,j}, AF = \frac{1}{t}\sum_{j=1}^{t} r_{t,j} - r_{j,j}. \quad (11)$$

To evaluate the parameter overhead, we denote the total number of the model as $\mathcal{M}$.



(a) Average performance (AP) (b) Model size (c) Experts (d) Time

Figure 3. Ablation studies of α (Alpha) when $\beta = 0.75$. The larger α encourages the model to reuse experts. so the model size and number of experts decreases with the increase of α. The α is independent of the architecture search, so the learning time is stable relative to it.



(a) Average performance (AP) (b) Model size (c) Cells (d) Time

Figure 4. Ablation studies of β (Beta) when $\alpha = 0.5$. The β has little effect on the average performance and size of model, and conversely has a great effect on the learning time because the determine the frequency of architecture search. The larger β encourages the model to reuse architectures of existing experts.

Implementation details We implement PAR by the PyTorch and open the source code[2]. We set the number of cells of each expert in PAR as 4, and set the α to 0.5 and the β to 0.75 for all tasks. We adopt the SGD optimizer whose initial learning rate is 0.01 and anneals following a cosine schedule. We also set the momentum of SGD to 0.9 and search weight decay from $[0.0003, 0.001, 0.003, 0.01]$ according to the validation performance. The results are averaged across three runs with different random seeds.

Without special instructions, following MNTDP [42], we adopt a light version of ResNet18 [13] with multiple output heads as the backbone for baselines. For the performance on CTrL, we report the results from MNTDP [42] and LMC [29]. For the performance on mixed CIFAR100 and F-CelebA, we report the results from CAT [15]. For the performance on classical benchmarks, we: report the results of RPSnet [31], InstAParam [7], and BNS [30] from origin papers; adopt the implementation of [4] for memory-based methods; and adopt our implementation for the others.

4.2. Comparison with baselines

At first, we evaluate PAR on benchmarks with mixed similar and dissimilar tasks. As shown in Tab. 1, PAR outperforms baselines on all six streams in CTrL. One reason is that PAR allows knowledge transfer among related tasks while preventing interference among tasks not related. For

[2]https://github.com/WenjinW/PAR

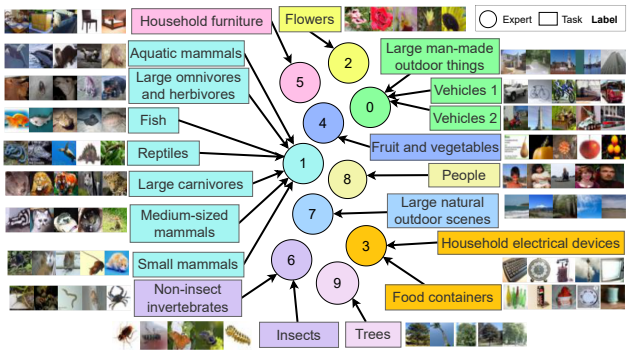
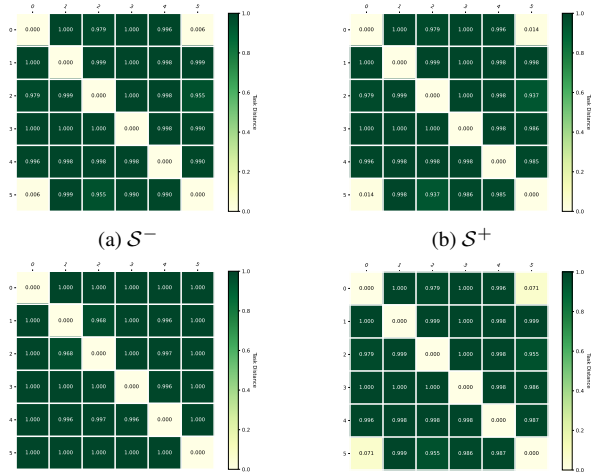


Figure 5. Visualization of *task groups* and *experts* on CIFAR100-Coarse. The PAR groups semantically related tasks into the same group. We distinguish the different groups by different colors.

example, performance on stream $\mathcal{S}^+$ shows that PAR can update the knowledge of the expert for the task t_1^- by the task t_1^+ with the same distribution and more samples. Performance on streams $\mathcal{S}^-$ and $\mathcal{S}^{\text{out}}$ shows that our method can transfer knowledge from experts of tasks related to the new task. Another reason is that a task-tailored architecture helps each expert perform better. Moreover, performance on stream $\mathcal{S}^{\text{long}}$ shows that PAR is scalable for the long task sequence. Similarly, results in Tab. 2 show that PAR outperforms parameter allocation methods with static models.

Further, we evaluate the PAR on classical benchmarks.



(a) $\mathcal{S}^{-}$ (b) $\mathcal{S}^{+}$

(c) $\mathcal{S}^{\text{in}}$ (d) $\mathcal{S}^{\text{out}}$

Figure 6. The heat-maps of task distances of four streams in CTrL benchmark. The PAR can recognize the related tasks in $\mathcal{S}^{-}$, $\mathcal{S}^{+}$, and $\mathcal{S}^{\text{out}}$ exactly.

Results in Tab. 3 show that PAR outperforms parameter regularization methods, memory-based methods, and parameter allocation methods with dynamic model on the CIFAR100-10 and MiniImageNet-20.

At last, we analyze the parameter overhead[3] of PAR on CIFAR100-10 and $\mathcal{S}^{\text{long}}$. Compared with baselines, the PAR obtains better average performance with fewer model parameters. The reason is that the the allocated experts in PAR have compact architecture and are parameter efficient. Performance on the stream $\mathcal{S}^{\text{long}}$ shows that PAR is scalable.

4.3. Ablation Studies

We analyze the effects of components in PAR and the results are listed in Tab. 5. First, we evaluate the impact of hierarchical architecture search on parameter allocation. Compared with using a fixed cell from DARTs [22] (#1), searching cells from fine-grained space can improve model performance (#2). Combined with coarse-grained space, searching from the hierarchical space can further improve performance while reducing the time and parameter overhead (#3). Then, we find that parameter regularization alone is time efficient, but its performance is low (#4). Finally, by selecting an appropriate strategy from parameter allocation and regularization based on the learning difficulty of each new task, PAR make a trade off among average performance, parameter overhead, and time overhead.

We analyze the impact of important hyper-parameters α and β in PAR on the CIFAR100-10. The α determines the reuse experts during the learning and the larger α encourages the model to reuse experts. Results in Fig. 3 show that the model size and number of experts decreases with the increase of α. The β determines the frequency of architecture search and the larger β encourages the model to reuse architectures of existing experts Results in Fig. 4 show that the learning time decreases with the increase of β.

[3]We ignore the parameter overhead of the extractor (11.18M) in divergence estimation, which is fixed and will not become a bottleneck with the increase of tasks.

4.4. Visualization

To analyze the task distance in PAR, we construct a new benchmark CIFAR100-coarse by dividing CIFAR100 into 20 tasks based on the coarse-grained labels. The illustration in Fig. 5 shows that the PAR can obtain reasonable task groups. For example, the PAR adaptively puts tasks about animals into the same groups, such as the Aquatic mammals, Large omnivores and herbivores, Fish, and so on. The Food containers and Household electrical devices are divided into the same group since they both contain cylindrical and circular objects. It also finds the relation between Non-insect invertebrates and Insects.

Moreover, we visualize the heat maps of task distances obtained by PAR on four streams of CTrL in Fig. 6. The distance between the first and the last task on streams $\mathcal{S}^{-}$, S^{+} and $\mathcal{S}^{\text{out}}$ are small, which is consistent with the fact that they all come from the same data distribution. Instead, the distribution of the last task on stream S^{in} is perturbed from the distribution of the first task, so their distance is large. The above results show that PAR can obtain reasonable task distance and relatedness.

5. Conclusion

In this paper, we propose a new method named **P**arameter **A**llocation & **R**egularization (PAR). It allows the model to adaptively select an appropriate strategy from parameter allocation and regularization for each task based on the task learning difficulty. Experimental results on multiple benchmarks demonstrate that PAR is scalable and significantly reduces the model redundancy while improving performance. Moreover, qualitative analysis shows that PAR can obtain reasonable task relatedness. Our method is flexible, and in the future, we will introduce more relatedness estimation, regularization, and allocation strategies into PAR.

Acknowledgments

This work was supported by the NSFC projects (No. 62072399, No. U19B2042), Chinese Knowledge Center for Engineering Sciences and Technology, MoE Engineering Research Center of Digital Library, China Research Centre on Data and Knowledge for Engineering Sciences and Technology, and the Fundamental Research Funds for the Central Universities.

References

[1] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory Aware Synapses: Learning what (not) to forget. In *ECCV*, pages 139–154, 2018. 6

[2] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert Gate: Lifelong Learning With a Network of Experts. In *CVPR*, pages 3366–3375, 2017. 1, 2

[3] Ali Ayub and Alan Wagner. EEC: Learning to Encode and Regenerate Images for Continual Learning. In *ICLR 2021*, 2020. 2

[4] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark Experience for General Continual Learning: A Strong, Simple Baseline. In *NeurIPS 2020*, page 11, 2020. 7

[5] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient Lifelong Learning with A-GEM. In *ICLR 2019*, Sept. 2018. 6

[6] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. On Tiny Episodic Memories in Continual Learning. *arXiv:1902.10486 [cs, stat]*, June 2019. 6

[7] Hung-Jen Chen, An-Chieh Cheng, Da-Cheng Juan, Wei Wei, and Min Sun. Mitigating Forgetting in Online Continual Learning via Instance-Aware Parameterization. In *NeurIPS 2020*, page 12, 2020. 1, 2, 7

[8] Zhi-Qi Cheng, Xiao Wu, Siyu Huang, Jun-Xiu Li, Alexander G Hauptmann, and Qiang Peng. Learning to transfer: Generalizable attribute learning with multitask neural model search. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 90–98, 2018. 2

[9] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *TPAMI*, pages 1–1, 2021. 1

[10] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning Without Memorizing. In *CVPR*, pages 5138–5146, 2019. 1, 2

[11] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural Architecture Search: A Survey. *JMLR*, page 21, 2019. 2

[12] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):8, 1999. 1

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Las Vegas, NV, USA, June 2016. IEEE. 1, 7

[14] Siyu Huang, Xi Li, Zhi-Qi Cheng, Zhongfei Zhang, and Alexander Hauptmann. Gnas: A greedy neural architecture search method for multi-attribute learning. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 2049–2057, 2018. 2

[15] 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. 1, 2, 5, 6, 7

[16] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. 1, 2, 6

[17] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. 6

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. 1

[19] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming Catastrophic Forgetting by Incremental Moment Matching. In *NeurIPS 2017*, volume 30, pages 4652–4662, 2017. 1, 2

[20] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to Grow: A Continual Structure Learning Framework for Overcoming Catastrophic Forgetting. In *ICML*, page 10, 2019. 1, 2, 6

[21] Zhizhong Li and Derek Hoiem. Learning without Forgetting. *IEEE transactions on pattern analysis and machine intelligence*, page 14, 2017. 1, 2, 4, 6

[22] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: DIFFERENTIABLE ARCHITECTURE SEARCH. In *ICLR 2019*, page 13, 2018. 2, 5, 8

[23] Huihui Liu, Yiding Yang, and Xinchao Wang. Overcoming Catastrophic Forgetting in Graph Neural Networks. In *AAAI 2021*, 2020. 1, 2

[24] Jorge A. Mendez and Eric Eaton. Lifelong Learning of Compositional Structures. In *ICLR 2021*, Sept. 2020. 1, 2, 6

[25] Qier Meng and Satoh Shin'ichi. ADINet: Attribute Driven Incremental Network for Retinal Image Classification. In *CVPR 2020*, pages 4033–4042, 2020. 1, 2

[26] Zichen Miao, Ze Wang, Wei Chen, and Qiang Qiu. Continual Learning with Filter Atom Swapping. In *International Conference on Learning Representations*, Sept. 2021. 1, 2

[27] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. VARIATIONAL CONTINUAL LEARNING. In *ICLR*, page 18, 2018. 1, 2

[28] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi. Learning to Remember: A Synaptic Plasticity Driven Framework for Continual Learning. In *CVPR*, pages 11321–11329, 2019. 2

[29] Oleksiy Ostapenko, Pau Rodriguez, Massimo Caccia, and Laurent Charlin. Continual Learning via Local Module Composition. In *NeurIPS 2021*, May 2021. 1, 2, 6, 7

[30] Qi Qin, Wenpeng Hu, Han Peng, Dongyan Zhao, and Bing Liu. BNS: Building Network Structures Dynamically for Continual Learning. In *NeurIPS 2021*, May 2021. 1, 2, 7

[31] Jathushan Rajasegaran, Munawar Hayat, Salman H Khan, Fahad Shahbaz Khan, and Ling Shao. Random Path Selection for Continual Learning. In H. Wallach, H. Larochelle, A.

Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, editors, *NeurIPS 2019*, pages 12669–12679. Curran Associates, Inc., 2019. 1, 2, 7

[32] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental Classifier and Representation Learning. In *CVPR*, pages 2001–2010, 2017. 2

[33] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A Comprehensive Survey of Neural Architecture Search: Challenges and Solutions. *ACM Computing Surveys*, 2020. 2

[34] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive Neural Networks. *arXiv:1606.04671 [cs]*, June 2016. 1, 2, 6

[35] Gobinda Saha and Kaushik Roy. Gradient Projection Memory for Continual Learning. In *ICLR 2021*, Sept. 2020. 6

[36] Joan Serrà, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming Catastrophic Forgetting with Hard Attention to the Task. In *ICML*, page 10, 2018. 1, 2, 6

[37] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual Learning with Deep Generative Replay. In *NeurIPS 2017*, pages 2990–2999. Curran Associates, Inc., 2017. 2

[38] Pravendra Singh, Pratik Mazumder, Piyush Rai, and Vinay P. Namboodiri. Rectification-Based Knowledge Retention for Continual Learning. In *CVPR 2021*, pages 15282–15291, 2021. 1, 2

[39] Pravendra Singh, Vinay Kumar Verma, Pratik Mazumder, Lawrence Carin, and Piyush Rai. Calibrating CNNs for Lifelong Learning. In *NeurIPS 2020*, volume 33, 2020. 1, 2

[40] Binh Tang and David S. Matteson. Graph-Based Continual Learning. In *ICLR 2021*, Sept. 2020. 2

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *NeurIPS 2017*, pages 5998–6008. Curran Associates, Inc., 2017. 1

[42] Tom Veniat, Ludovic Denoyer, and MarcAurelio Ranzato. Efficient Continual Learning with Modular Networks and Task-Driven Priors. In *ICLR 2021*, Sept. 2020. 1, 2, 5, 6, 7

[43] 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. 6

[44] Qing Wang, Sanjeev R. Kulkarni, and Sergio Verdu. Divergence Estimation for Multidimensional Densities Via k-Nearest-Neighbor Distances. *IEEE Transactions on Information Theory*, 55(5):2392–2405, 2009. 2, 3

[45] Yeming Wen, Dustin Tran, and Jimmy Ba. BatchEnsemble: An Alternative Approach to Efficient Ensemble and Lifelong Learning. In *ICLR*, Sept. 2019. 1, 2

[46] Ju Xu and Zhanxing Zhu. Reinforced Continual Learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *NeurIPS 2018*, pages 899–908. Curran Associates, Inc., 2018. 2

[47] Shipeng Yan, Jiangwei Xie, and Xuming He. DER: Dynamically Expandable Representation for Class Incremental Learning. In *CVPR 2021*, pages 3014–3023, 2021. 1, 2

[48] 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. 2, 5

[49] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning Transferable Architectures for Scalable Image Recognition. In *CVPR 2018*, pages 8697–8710, Salt Lake City, UT, 2018. IEEE. 2