CONTINUAL LEARNING VIA LEARNING CONTINUAL MEMORY IN VISION TRANSFORMER

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper explores continual learning (CL) using Vision Transformer (ViT) in streaming tasks under the challenging exemplar-free class-incremental (ExfCCL) setting. We formulate ExfCCL as a learning problem consisting of two key sub-systems: (i) task ID inference for test data, which selects appropriate task-specific head classifiers to accounting for varying class distributions across tasks and streams, and (ii) a dynamic learning-to-grow feature backbone that balances stability and plasticity, mitigating catastrophic forgetting through task synergies. Following the common protocol that the first task can train a ViT sufficiently well as the base model, we address these sub-systems from a continual memory learning perspective. To support task ID inference, we utilize an external memory mechanism that maintains task centroids computed by the base ViT throughout CL. For the feature backbone, we identify optimal placements for internal (parameter) memory to enable a dynamic, task-synergy guided growing feature backbone. We propose a Hierarchical Exploration-Exploitation (HEE) sampling-based neural architecture search (NAS) method that effectively learns task synergies by continually and structurally updating internal memory with four basic operations: `reuse`, `adapt`, `new`, and `skip`. Our approach, dubbed **Continual Hierarchical-Exploration-Exploitation Memory (CHEEM)**, is evaluated on the challenging Visual Domain Decathlon (VDD) and ImageNet-R benchmarks, demonstrating its effectiveness.

1 INTRODUCTION

Developing continual learning machines is a key objective in Artificial Intelligence (AI), aiming to replicate human-like adaptability and the ability to learn-to-learn, enabling proficiency in streaming tasks. Despite their advances, state-of-the-art Deep Neural Networks (DNNs) still lack true biological intelligence in the realm of continual learning and are particularly hindered by the critical issue of *catastrophic forgetting* when exposed to streaming tasks in dynamic environments (McCloskey & Cohen, 1989; Thrun & Mitchell, 1995).

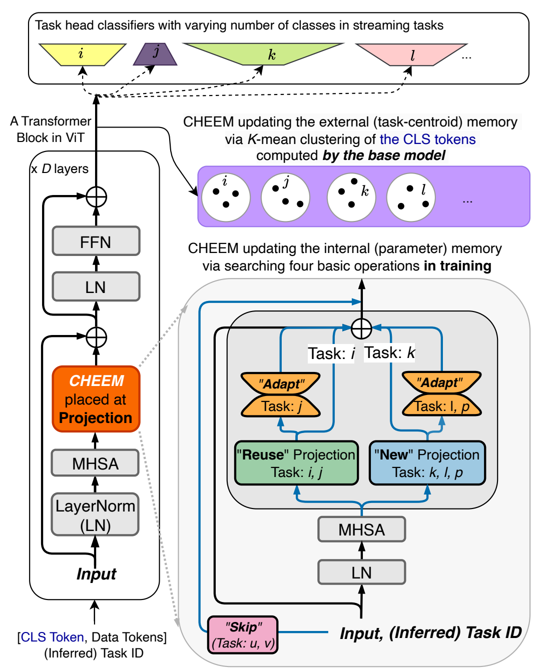


Figure 1: Illustration of the proposed CHEEM.

To address catastrophic forgetting, two primary categories of continual learning methods have emerged: exemplar-based methods (Aljundi et al., 2019b; Hayes et al., 2019; Wu et al., 2019) and exemplar-free methods (Kirkpatrick et al., 2017; Li et al., 2019; Wang et al., 2022d;c;a). While both have shown promising progress, exemplar-free methods are especially appealing due to their ability to learn new tasks without retaining any data from previous tasks. Furthermore, continual learning has evolved from the traditional task-incremental protocol, where task IDs of test data are available during inference, to the more challenging class-incremental protocol, in

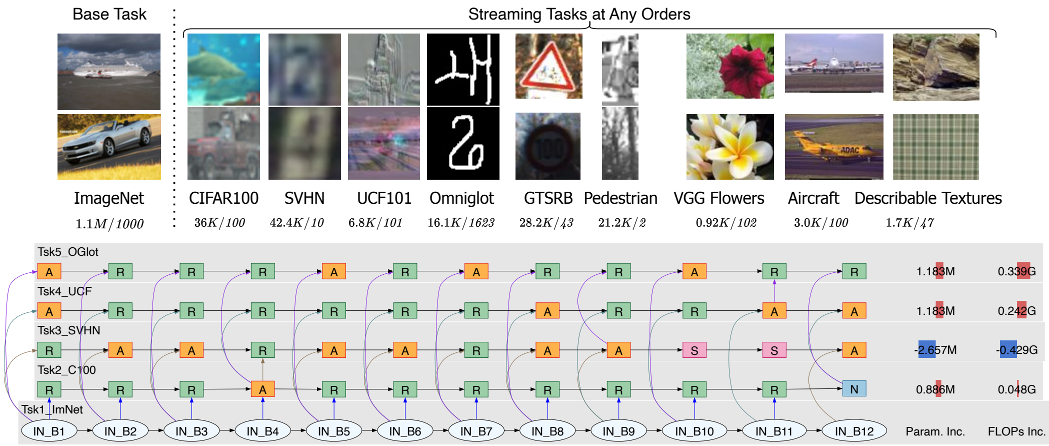


Figure 2: **Top:** The challenging VDD benchmark (Rebuffi et al., 2017a) consisting of 10 tasks of different nature with `#training images/#classes` significantly varying across different tasks. **Bottom:** Starting from the base task (Tsk1_ImNet) trained ViT (Dosovitskiy et al., 2021) (e.g., the 12-layer ViT-B model consisting of IN_B1 to IN_B12), together with the best performance on VDD in experiments, our CHEEM can learn sensible memory structures: *e.g.*, Tsk3_SVHN learns to `skip` both B10 and B11 with both mode complexity and FLOPs reduced. Tsk4_UCF and Tsk5_OGlot learn to `adapt` the first layers to account for the domain shifts. We show the first 5 tasks for clarity (see more in the supplementary). The last two columns show the number of new task-specific parameters and added FLOPs, in comparison with the base model, subject to R euse, A dapt, N ew and S kip.

which task IDs are unknown during inference. This paper focuses on exemplar-free class-incremental continual learning (ExfCCL).

Recently, the emergence of powerful pretrained Transformer models (Vaswani et al., 2017; Dosovitskiy et al., 2021; Radford et al., 2021) has driven significant interest in ExfCCL using pretrained and frozen Vision Transformers (ViTs) (Dosovitskiy et al., 2021), primarily explored through the lens of prompt-tuning or prefix-tuning (Wang et al., 2022d;c;a; Smith et al., 2023a). However, this approach presents two main drawbacks: (i) A single frozen pretrained Transformer backbone cannot accommodate all streaming tasks of diverse nature, such as those found in the VDD benchmark (Rebuffi et al., 2017a), shown in Fig. 2. While this method maximizes the stability of the feature backbone, it relies on input or prefix prompts to address plasticity by explicitly leveraging the self-attention mechanisms. (ii) Due to the quadratic complexity of ViTs concerning the number of input tokens, incorporating task-specific input prompts or layer-wise prefix prompts into a frozen pretrained Transformer backbone significantly increases computational demands. Continual learning where computational requirements always grow for new tasks by design regardless of task complexity fails to reflect true intelligence.

The challenge of structurally and dynamically updating a pretrained Transformer backbone for ExfCCL remains an open problem, as it requires achieving a balance between the backbone's stability and plasticity while enabling dynamic computation and mitigating catastrophic forgetting.

Moreover, ExfCCL involves tackling the challenge of designing task head classifiers, which can either explicitly infer task IDs to select task-specific head classifiers (Wang et al., 2022a) or share a growing, common head classifier (Wang et al., 2022d;c; Smith et al., 2023a). The former approach is straightforward and can help in learning task-specific components and enhancing the plasticity of the backbone, but may suffer from low average precision of the task ID inference for certain tasks. In contrast, the latter approach resorts to more sophisticated prefix prompt tuning to induce plasticity in the feature backbone and can create a discrepancy between training, where head segments from previous tasks are masked out in the softmax function of loss computation, and testing, where the entire head is used to infer class labels. This discrepancy can lead to instability when dealing with streaming tasks that involve a varying number of classes among tasks, such as those encountered in the VDD benchmark.

Our aim in this paper is to study ExfCCL using ViTs. As illustrated in Fig. 1, we formulate it as a problem of learning continual memory in ViT, which has two components: (i) The internal memory enabling a dynamic learning-to-grow feature backbone that balances stability and plasticity, mitigating catastrophic forgetting through **task synergies, in which a new task learns automatically to reuse/adapt modules from previous similar tasks, or to introduce new modules when needed, or to skip some modules when it appears to be an easier task** (see the bottom of 2). The internal parameter memory learning presents alternative perspectives to the input and prefix prompting based methods (Wang et al., 2022d;c; Smith et al., 2023a; Wang et al., 2022a). (ii) The external memory enabling task ID inference for test data, for which we adopt a method proposed in (Wang et al., 2022a) for its simplicity.

We follow the common protocol in the prior art that the first task (e.g., ImageNet-1k (Russakovsky et al., 2015)) can train a ViT sufficiently well as the base model. To enable a dynamic, learning-to-grow feature backbone starting from the base model, we identify and provide simple yet effective solutions for two key challenges: **(i) Which modules in a ViT should be reused, adapted, newly created, or skipped for a new task in ExfCCL?** It is computationally impractical, and counterproductive to the stability-plasticity trade-off, to make all ViT components dynamic (as opposed to a fully frozen model). We designate the output projection layer following multi-head self-attention (MHSA) as *the task-synergy internal (parameter) memory* that will be structurally and dynamically updated. **(ii) How can we represent and continually learn this task-synergy internal memory to enable dynamic memory structures across streaming tasks?** We propose organizing the memory using a mixture of experts (MoEs) similar in spirit to (Riquelme et al., 2021), as depicted in Fig. 1 and illustrated through the "task-factorized" visualization in Fig. 2. To learn to select the optimal synergy operation from the four choices (`reuse, adapt, new`, and `skip`), we introduce an effective hierarchical exploration-exploitation (HEE) sampling-based neural architecture search (NAS) method. This approach is inspired by the task-incremental learn-to-grow method (Li et al., 2019). Our proposed method is termed **CHEEM** (*Continual Hierarchical-Exploration-Exploitation Memory*).

Our Contributions. This paper makes three main contributions to the field of exemplar-free class-incremental continual learning (ExfCCL) using ViT. (i) It presents a hierarchical task-synergy exploration-exploitation sampling based NAS method for learning task-aware dynamic models continually with respect to four operations: `Skip`, `Reuse`, `Adapt`, and `New` to mitigate catastrophic forgetting. (ii) It identifies a "sweet spot" in ViT as the task-synergy internal (parameter) memory, i.e., the output projection layers after MHSA in ViT. It also presents a new usage for the class-token `CLS` in ViT as the internal memory updating guidance, in addition to leveraging it in maintaining the external (task-centroid) memory for task ID inference on the fly. (iii) It is the first work, to the best of our knowledge, to evaluate continual learning with ViTs on the large-scale, diverse and imbalanced VDD benchmark (Rebuffi et al., 2017a), with better performance than the prior art.

2 Approach

2.1 Problem Formulation of CHEEM in ExfCCL

We start with a vanilla D-layer ViT model (e.g., the 12-layer ViT-Base) (Dosovitskiy et al., 2021). The left of Fig. 1 shows a ViT block. Denote by $x_{L,d}$ an input sequence consisting of L tokens encoded in a d-dimensional space. In ViTs, the first token is the so-called class-token, `CLS`. The remaining $L-1$ tokens are formed by patchifying an input image and then embedding patches, together with additive positional encoding. A ViT block is defined by,

$$z_{L,d} = x_{L,d} + \text{Proj}(\text{MHSA}(\text{LN}_1(x_{L,d}))), \tag{1}$$

$$y_{L,d} = z_{L,d} + \text{FFN}(\text{LN}_2(z_{L,d})))), \tag{2}$$

where LN$(\cdot)$ represents the layer normalization (Ba et al., 2016), and Proj$(\cdot)$ is a linear transformation fusing the multi-head outputs from MHSA module. The FFN is often implemented by a multi-layer perceptron (MLP) with a feature expansion layer MLP^{u} and a feature reduction layer MLP^{d} with a nonlinear activation function (such as the GELU (Hendrycks & Gimpel, 2016)) in the between.

Denote by $\mathcal{T} = \{T_1, \cdots, T_t, \cdots, T_N\}$ a stream of tasks in continual learning, where each task T_t consists of a training set D_t^{train} and a testing set D_t^{test}. We make no restrictive assumptions regarding the nature, order, or number of classes in streaming tasks, either per task or in total. For example, there are 9 diverse, streaming tasks in the VDD benchmark (2).

Denote by (f_1, C_1) the ViT (Dosovitskiy et al., 2021) base model (e.g., the 12-layer ViT-Base) trained on the first task T_1 (e.g., ImageNet-1k (Russakovsky et al., 2015)), where f_1 is the backbone and C_1 the head classifier. Denote by $(\mathcal{F}_t, \mathcal{C}_t)$ the sequentially and continually learned model after task T_t

(for $t \geq 1$). $\mathcal{F}_t$ is the super ViT backbone structurally and dynamically updated from the base model $f_1 = \mathcal{F}_1$, and $\mathcal{C}_t = \{C_1, \cdots, C_t\}$ consisting of task-specific head classifiers each of which is trained from scratch. Let $\Theta_t = \mathcal{F}_t \setminus \mathcal{F}_{t-1}$ be task-specific backbone parameters, which we term **the internal parameter memory** in ExfCCL to exploit task synergies. We note that ExfCCL requires no use of exemplar data of previous tasks in any forms in learning (Θ_t, C_t) for task T_t.

In inference, since the task IDs of test data are unknown. For a test sample x, we will need to infer its task ID on the fly. Following the prior art (Wang et al., 2022d;c; Smith et al., 2023a; Wang et al., 2022a), we use the base model $f_1(\cdot)$ as the query function $q()$, and use $q(x)$ to retrieve the task ID from **the external memory** for which we adopt the method proposed in (Wang et al., 2022a) for its simplicity. With task ID available (provided in training or inferred in testing), we can allocate the task-specific backbone $f_t \subset \mathcal{F}_N$ for task T_t. The task-specific model is then specified by (f_t, C_t).

2.2 Constructing the External Memory

We choose to explicitly infer task IDs for test data (see Appendix A for the analyses). We adopt the method proposed in S-Prompts (Wang et al., 2022a). For a task t, we leverage K-mean clustering of the `CLS` tokens of its training images computed by the query function $q(\cdot)$ (i.e., the base model f_0). Denote by $Z_t = \{z_t^1, \cdots, z_t^K\}$ the clustered task centroids for task T_t, where for simplicity we use the same $K = 5$ across tasks. We have the external memory, $\Psi = \cup_{t=1}^N Z_t$ after training. In inference, for a testing sample x, we first compute its `CLS` token using the same query function $q(x)$, denoted by z. The task ID is then inferred via K-NN retrieval, $KNN(z, \Psi)$, either by retrieving the class ID of the Top-1 NN centroid in the external memory Ψ, or by majority voting of the class ID from the K-NN centroids.

We note that the such constructed external memory does not break the exemplar-free protocol since we do not use exemplars in any forms from previous tasks. We also note that we focus on the learning of the internal parameter memory. The decoupled design between the external memory and the internal memory will enable us to integrate and/or develop more advanced task ID inference approaches in future work, while potentially promoting our proposed CHEEM in the field of ExfCCL.

2.3 Identifying the Task-Synergy Internal Memory

The proposed identification process is straightforward. Without introducing any modules handling forgetting, we compare both the task-to-task forward transferrability and the sequential forgetting for different components in a ViT block. **Our intuition is that a desirable component for placing the task-synergy parameter memory must enable strong transferrability with manageable forgetting, while being lightweight to account for the trade-off between stability and plasticity.**

To that end, we use the VDD benchmark (Rebuffi et al., 2017a) (see Fig. 2). We first train a ViT-Base (Dosovitskiy et al., 2021) on the first task, ImageNet (Russakovsky et al., 2015), as the base model $f_1(\cdot)$. To measure the task-to-task transferability, we *individually fine-tune* f_1 in a task-to-task transfer learning manner for the remaining 9 streaming tasks. Let $f_{t|1}$ be the backbone fine-tuned for task T_t (for $t \geq 1$), and C_t the head classifier trained from scratch. The average Top-1 accuracy is:

$$A\mathbf{A} = \frac{1}{N} \sum_{t=1}^{N} \text{Acc}(T_t; f_{t|1}, C_t) \tag{3}$$

where Acc() uses the Top-1 classification accuracy.

Table 1: Results of identifying the optimal placement of our CHEEM in ViT by testing 11 components (Eqns. 1 and 2).

Index	Finetuned Component	$A\mathbf{A}$ (Eqn. 3)	$A\mathbb{F}$ (Eqn. 4)
1	$LN_1 + LN_2$	81.76	21.24
2	FFN	84.20	44.76
3	MLP^d	83.66	37.99
4	LN_2	80.04	16.35
5	MHSA + LN_1	85.26	54.38
6	LN_1	81.18	19.04
7	Query	81.57	19.69
8	Key	81.56	19.19
9	Query+Key	81.49	31.10
10	Value	84.99	37.58
11	Projection (**CHEEM**)	85.11	30.50
Classifier w/ Frozen Backbone		70.78	-

To measure the sequential forgetting, we *continually fine-tune* the backbone started from f_1 on the 9 tasks in a randomly sampled and fixed streaming order (as shown in Fig. 2). Let $f_{1:t}$ be the backbone trained sequentially and continually after task T_t and C_t is its head classifier. The average forgetting (Chaudhry et al., 2018) on the first $N-1$ streaming tasks is,

$$A\mathbb{F} = \frac{1}{N-1} \sum_{t=1}^{N-1} \left(\max_{j \in [t,N]} a_{j,t} - a_{N,t} \right), \tag{4}$$

where $a_{j,t} = \text{Acc}(T_t; f_{1:j}, C_t)$.

As shown in Table 1, we compare 11 components or composite components in ViT. Consider the strong forward transfer ability, manageable forgetting, maintaining simplicity and for less invasive implementation in practice, **we select the Projection layer after the MHSA as the task-synergy internal (parameter) memory** (Fig. 1) to realize our proposed CHEEM for ExfCCL. We note that the last row of Table 1 shows the result of a conventional transfer learning setting in which only the head classifier is trained with the backbone frozen, which clearly shows that one frozen backbone can not fit all streaming tasks, as aforementioned and entailing dynamic updating of the backbone in continual learning.

2.4 Learning the Task-Synergy Internal Memory

The proposed internal memory of our CHEEM is represented by a MoEs. Starting with the base ViT model f_1, the internal memory at the l-th layer in ViT consists of a single expert defined by a tuple,

$$\mathtt{E}_l^{(1,)} = (\theta_l^{(1,)}, \mu_l^1), \tag{5}$$

where the subscript represents the layer index and the list-based superscript shows which task(s) use this expert. $\theta_l^{(1,)}$ are the parameters of the projection layer and $\mu_l^1 \in R^d$ is the associated mean class-token (`CLS`) pooled from the training dataset after the model is trained, which is task specific (as indicated by the superscript). For example, if an expert is reused by another task (say, 3) in continual learning, we will have $\mathtt{E}_l^{(1,3,)} = (\theta_l^{(1,3,)}, \mu_l^1, \mu_l^3)$.

As shown in Fig. 3, for a new task t, **learning to update CHEEM consists of:** i) the Supernet construction, ii) the Supernet training, and iii) the target network selection and finetuning.

2.4.1 Supernet Construction via `Reuse`, `Adapt`, `New` and `Skip`

For clarity, we consider a single layer l for a new task, and the current memory consists of two experts, $\{\mathtt{E}_l^{(1,)}, \mathtt{E}_l^{(2,)}\}$ (Fig. 3, left). We utilize four operations in the Supernet construction:

- R euse the projection layer from a previous (similar) task for a new task.
- A dapt by adding a new lightweight layer on top of the projection layer of a previous task, implemented by a MLP with one squeezing hidden layer.
- N ew by adding a new projection layer, which enables the model to handle dissimilar situations.
- S kip the entire MHSA block, which encourages the adaptivity accounting for the diverse nature of tasks.

The Supernet is constructed by `reusing` and `adapting` each existing expert at layer l, and adding a `new` and a `skip` expert (the bottom of Fig. 3). The newly added `adapt` MLPs and `new` projection layers will be trained from scratch using the data of a new task only. The right-top of Fig. 3 shows the `Adapt` operation on top of $\mathtt{E}_l^{(2,)}$ is learned and added, $\mathtt{E}_l^{(3,)} = (\theta_l^{(3,)}, \mu_l^3)$ where $\theta_l^{(3,)}$ represents parameters of the `adapt` MLPs learned for the task 3, and μ_l^3 is the mean `CLS` token pooled for the task 3. The expert $\mathtt{E}_l^{(2,)}$ is updated to $\mathtt{E}_l^{(2,3,)} = (\theta_l^{(2,3,)}, \mu_l^2)$ indicating its weights are shared with task 3. We provide implementation details for `Adapt` in the Appendix D.5.

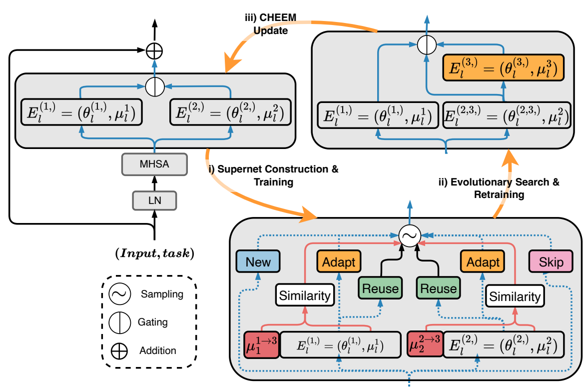


Figure 3: Illustration of CHEEM learning via NAS.

2.4.2 Supernet Training via the Proposed HEE Sampling-based NAS

To train the Supernet constructed for a new task t, we build on the SPOS method (Guo et al., 2020) due to its efficiency. The basic idea of SPOS is to train a single-path sub-network from the Supernet by sampling an expert at every layer in each mini-batch of training. One key aspect is the sampling strategy. The vanilla SPOS method uses uniform sampling (i.e., the *pure exploration* (PE) strategy, Fig. 4 left). **We propose an exploitation strategy** (Fig. 4 right), which utilizes a hierarchical sampling

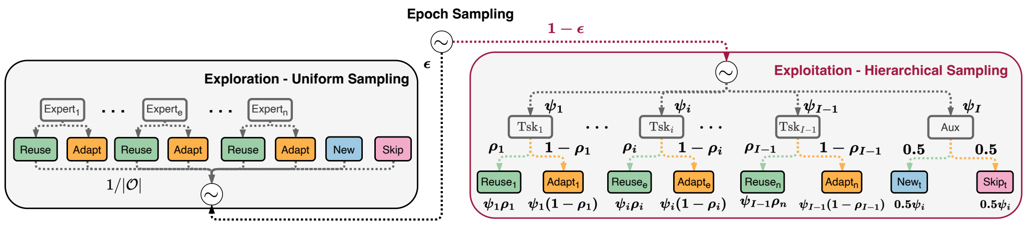


Figure 4: Illustration of the proposed hierarchical task-synergy exploration-exploitation (HEE) sampling based NAS. It integrates the vanilla exploration strategy (left) and the proposed exploitation strategy (right) with an epoch-wise scheduling.

method that forms the categorical distribution over the operations in the search space *explicitly based on task synergies computed based on the pooled task-specific* `CLS` *tokens*.

As illustrated in the right of Fig. 4, at each layer l in the Supernet, for a new task t, our proposed 3-level HEE sampling is realized by:

- **The epoch-level sampling:** At the beginning of an epoch in the Supernet training, we choose the pure exploration strategy with a probability of ϵ_1 (e.g., 0.3), and the exploitation strategy with a probability of $1 - \epsilon_1$.
- **The task-level sampling:** There are $t - 1$ previous tasks, some of which may use `Skip` at the l-th layer and will not be used in sampling. We introduce an auxiliary task to handle sampling `New` and `Skip` for the current task t. *The task-level sampling is based on a categorical distribution* $(\psi_1, \cdots, \psi_i, \cdots, \psi_{I-1}, \psi_I)$, which is computed by the Softmax function over the similarity scores defined below, where $I \leq t$.
- **The operation-level sampling:** With a previous task i sampled with the probability ψ_i, we sample the `Reuse` and `Adapt` operation using a Bernoulli distribution $(\rho_i, 1 - \rho_i)$, where ρ_i is the success rate computed by the Sigmoid function of the task similarity score (defined below).

The Task Similarity Score. Consider one expert $\mathtt{E}_l^{(i,j,)}$ at the l-th layer which is shared by two previous tasks i and j with their mean `CLS` tokens μ_l^i and μ_l^j respectively, we compute the mean `CLS` tokens using the models for task i and j on the training dataset D_t^{train} of the current task t, $\mu_l^{i \to t}$ and $\mu_l^{j \to t}$. The task similarity between task i and t is computed by,

$$S_l^{i,t} = \mathtt{NormCosine}(\mu_l^i, \mu_l^{i \to t}), \tag{6}$$

where $\mathtt{NormCosine}(\cdot, \cdot)$ is the Normalized Cosine Similarity, which is calculated by scaling the Cosine Similarity score between -1 and 1 using the minimum and the maximum Cosine Similarity scores from all the experts in all the MHSA blocks of the ViT. This normalization is necessary to increase the difference in magnitudes of the similarities between tasks, which results in better Expert sampling distributions during the sampling process in our experiments.

The Auxiliary Task. For a new task t, to handle the `New` and `Skip` experts at each layer l for which we do not have direct similarity scores. Instead, we introduce an auxiliary task, `Aux` (see the right of Fig. 4) which gives equally-likely chance to select `New` or `Skip` expert. For the `Aux` task itself, the similarity score between it and task t is defined by,

$$S_l^{aux,t} = -\max_{i=1}^{t-1} S_l^{i,t}, \tag{7}$$

which intuitively means we probabilistically resort to the `New` operation or the `Skip` operation when other experts turn out not "helpful" for the task t.

2.4.3 Target Network Selection and Finetuning

After the Supernet is trained, we adopt the same evolutionary search used in the SPOS method (Guo et al., 2020) based on the proposed HEE sampling strategy with a different epoch sampling probability ϵ_2 ($\epsilon_2 > \epsilon_1$, e.g., $\epsilon_2 = 0.5$, to encourage more exploration during the evolutionary search). The evolutionary search is performed on the validation set to select the path which gives the best validation accuracy. After the target network for a new task is selected, we retrain the newly added layers by the `New` and `Adapt` operations from scratch (random initialization), rather than keeping the weights from the Supernet training. This is based on the observations in network pruning that it is the neural architecture topology that matters and that the warm-up weights may not need to be preserved to ensure good performance on the target dataset (Liu et al., 2019b).

Table 2: Results on the VDD benchmark (Rebuffi et al., 2017a) using ViT-B/8 (Dosovitskiy et al., 2021) over 3 different seeds. The last two columns show the performance under the task-incremental continual learning (TCL) setting.

Method	C100 36k/*100*	SVHN 42.4k/*10*	UCF 6.8k/*101*	OGlt 16.1k/*1623*	GTSR 28.2k/*43*	DPed 21.2k/*2*	Flwr 0.92k/*102*	Airc. 3.0k/*100*	DTD 1.7k/*47*	$A\mathbb{A}$ (Eqn. 8)	$A\mathbb{F}$ (Eqn. 4)	TCL $A\mathbb{A}$	TCL $A\mathbb{F}$
L2P++	79.16	0.62	5.40	14.05	62.82	4.09	2.97	2.73	2.98	19.42 ± 0.09	5.49 ± 0.37	68.68 ± 0.67	10.25 ± 0.77
DualPrompt	82.92	15.27	12.19	33.71	80.12	7.65	3.17	3.47	3.23	26.86 ± 0.40	3.54 ± 0.27	76.57 ± 0.23	2.57 ± 0.26
CODA-Prompt	88.06	18.09	12.16	53.95	95.65	19.42	6.83	6.34	5.62	34.01 ± 0.99	8.0 ± 0.72	75.83 ± 0.60	6.25 ± 0.60
S-Prompts	87.38	88.53	64.05	72.17	98.53	**99.65**	**96.63**	45.49	**58.07**	78.95 ± 0.07	0.32 ± 0.32	79.71 ± 0.08	0.0 ± 0.0
EWC	79.15	85.62	47.03	64.57	90.65	54.56	55.26	34.87	54.49	62.91 ± 0.81	19.82 ± 0.69	75.00 ± 0.67	6.33 ± 0.30
L2 Regularization	82.59	65.57	51.57	26.83	95.62	98.50	83.04	33.80	51.67	65.47 ± 0.33	7.23 ± 0.51	69.63 ± 0.17	3.00 ± 0.47
Experience Replay	56.68	6.23	33.45	74.33	7.00	0.02	64.25	27.03	37.55	34.06 ± 0.69	44.89 ± 0.72	54.55 ± 2.67	22.67 ± 3.01
Our CHEEM	**88.56**	**95.63**	**75.05**	**83.81**	**99.15**	99.64	90.92	**55.53**	56.42	**82.74** ± 0.54	0.33 ± 0.00	**84.65** ± 0.33	0.0 ± 0.0

3 EXPERIMENTS

Data: We use the VDD benchmark (Rebuffi et al., 2017a) and the Split ImageNet-R(etention) benchmark introduced in Wang et al. (2022c), derived from (Hendrycks et al., 2021). The VDD benchmark (Fig. 2) consists of 10 tasks with significantly varying distributions of classes per task. The vanilla ImageNet-R dataset consists of 200 classes and 30k images in total. The Split ImageNet-R benchmark is constructed by randomly splitting the ImageNet-R dataset into 10 tasks each of which has 20 classes. To test the effects of imbalance class distributions in streaming tasks, we construct another Split ImageNet-R benchmark consisting of 6 tasks with imbalance number of classes (5, 10, 15, 20, 50, 100) randomly sampled from the 200 total classes without replacement for each task. More details are provided in Appendix. E.

Baselines of ExfCCL. We compare with S-Prompts (Wang et al., 2022a), Learning to Prompt (L2P) Wang et al. (2022d), DualPrompt Wang et al. (2022c) and CODA-Prompt Smith et al. (2023a). On the VDD benchmark, we also test two regularization based methods, the L2 Parameter Regularization (Smith et al., 2023b) and the EWC (Kirkpatrick et al., 2017), and a strong baseline of Experience Replay (with iCARL (Rebuffi et al., 2017b) for buffer updates) using the total buffer size 20k and 10 exemplars per class. For EWC, L2 and iCarl, we freeze the backbone model and only keep the output projection layers (where CHEEM is placed) in the MHSA blocks trainable for better understanding the advantages of structurally updating them in our CHEEM. More details are in Appendix F and G.

Metric: In addition to average forgetting (Eqn. 4), we compare average accuracy in ExfCCL setting,

$$A\mathbb{A} = \frac{1}{N}\sum_{t=1}^{N} \text{Acc}(T_t; f_{1:N}, C_t). \tag{8}$$

3.1 RESULTS ON THE VDD BENCHMARK

Table 2 shows the results. We use the ImageNet-1k (the 1st task in the VDD) trained ViT-B/8 as the base model. For fair comparisons with prompting based baselines, we evaluate average accuracy and forgetting on the remaining 9 tasks in the VDD. **Our proposed CHEEM obtains significantly better performance than all the baselines.** We analyze the results as follows.

i) The Advantage of Task-Aware Dynamic Backbone Over Frozen Backbone: CHEEM vs S-Prompts. CHEEM adopts the same task ID inference method (i.e., the external memory) proposed by S-Prompts (Wang et al., 2022a). The improvement by CHEEM with above 3% absolute average accuracy increase clearly shows the advantage of our proposed internal parameter memory for maintaining task-aware dynamic backbones, against the method of prepending retrieved task-specific prompts in S-Prompts. Table 2 also shows a potential for harnessing prompt-based methods in CHEEM. S-Prompts performs better on tasks that are similar to the first ImageNet task and have less training data such as Flwr (918 training images) and DTD (1692 training images).

ii) Explicit Task ID Inference: With vs Without. The three baselines, L2P (Wang et al., 2022d), DualPrompt (Wang et al., 2022c) and CODA-Prompt (Smith et al., 2023a), use a shared head classifier in inference, which, as aforementioned in the introduction, suffers from the discrepancy between training and testing. Due to the significant imbalance class distributions in the VDD, their average accuracy undergo catastrophic drops (19.42% by L2P, 26.86% by DualPrompt and 34.01% by CODA-Prompt, vs 78.95% by S-Prompts and 82.74% by our CHEEM).

To understand the effects of head classifiers better, we also compare the performance when task IDs are provided in inference (i.e., Task-incremental CL, TCL), so the three methods can use task-specific head segments of the shared classifier, and both S-Prompts and our CHEEM do not need the external task-centroid memory. The last two columns of Table 2 show the results. *We have three observations:*

- The performance of L2P, DualPrompt and CODA-Prompt are drastically boosted (e.g., from 34.01% to 75.83% by CODA-Prompt), showing the importance of explicitly inference task IDs for imbalance class distributions of streaming tasks in continual learning. Our CHEEM under CCL is still significantly better than the three methods under TCL (by more than 6%), **which reinforces the importance of the task-aware dynamic backbone learned via our CHEEM**.
- The performance difference between CCL and TCL by our CHEEM is small, around 2%, showing that the task ID inference method (Wang et al., 2022a) is effective in handling the imbalance class distributions in the VDD benchmark.
- In terms of average forgetting under CCL and TCL, we should compare the ratio between average forgetting and average accuracy, rather than the average forgetting rates in isolation, for which the first three methods undergo severer forgetting under CCL as intuitively expected. Our CHEEM does not have forgetting under TCL.

iii) Stability vs Plasticity of the Backbone. To further show the advantage of our CHEEM learning to update the feature backbone from the base model, we compare with EWC (Kirkpatrick et al., 2017), L2 Regularization (Smith et al., 2023b) and Experience Replay (Rebuffi et al., 2017b). The three methods regularize the weights, including the shared head classifier, in learning new tasks using different formulations. Comparing with L2P, DualPromt and CODA-Prompt, under CCL, the regularization based methods work better, showing the negative effects of the discrepancy of handling the shared head in training and testing by the three prompting based methods, as well as a potential of integrating them. Comparing with S-Prompts, the plasticity of updating the entire backbone with regularization is less effective than the plasticity introduced by prompts while keeping the backbone frozen in S-Prompts. Comparing with our CHEEM, the three regularization based methods suffer from significant forgetting, highlighting the balanced stability and plasticity via our CHEEM.

Table 3: Results on two constructed Split ImageNet-R benchmarks, averaged across 3 different task orders. The base model is ImageNet-1k trained ViT-B/16 sourced from (Wightman, 2019).

Method	Imbalanced (6 tasks with (5, 10, 15, 20, 50, 100) classes per task)				Balanced (10 tasks with 20 classes per task)			
	$A\mathbb{A}$ (Eqn. 8)	$A\mathbb{F}$ (Eqn. 4)	TCL $A\mathbb{A}$	TCL $A\mathbb{F}$	$A\mathbb{A}$ (Eqn. 8)	$A\mathbb{F}$ (Eqn. 4)	TCL $A\mathbb{A}$	TCL $A\mathbb{F}$
L2P++	64.44 ± 1.38	8.62 ± 4.02	88.21 ± 0.28	0.50 ± 0.36	73.01 ± 0.57	5.80 ± 0.29	89.08 ± 0.38	0.48 ± 0.02
DualPrompt	67.20 ± 1.38	6.95 ± 3.96	89.46 ± 0.57	0.45 ± 0.24	73.01 ± 0.55	3.35 ± 0.36	90.07 ± 0.19	0.29 ± 0.19
CODAPrompt	**67.66** ± 3.09	7.06 ± 2.97	90.69 ± 0.44	0.46 ± 0.23	**76.48** ± 0.25	5.04 ± 0.12	91.86 ± 0.34	0.45 ± 0.10
Our CHEEM	66.97 ± 0.43	8.44 ± 3.55	**91.48** ± 0.63	0.0 ± 0.0	64.72 ± 0.28	8.73 ± 0.41	**92.47** ± 0.64	0.0 ± 0.0

3.2 Results On the Split ImageNet-R Benchmark

The ImageNet-R dataset is created to challenge models trained using ImageNet. So, by design, the external memory of our CHEEM that is based on ImageNet-1k trained base model will not work well. Table 3 shows the results which we analyze as follows:

i) Randomly Assigned and *Imbalanced* Classes in Streaming Task Challenge All of ExfCCL Methods We Test. Under CCL, the three prompting based methods suffer from catastrophic drop of performance from balanced to imbalanced settings (e.g., from 76.48 ± 0.25% to 67.66 ± 3.09% by CODA-Prompt, see Fig. 5 and analyses in Appendix A). Our CHEEM is on-par with DualPrompt and CODA-Prompt. Our CHEEM is also stable from balanced to imbalanced scenarios. Under TCL, all methods obtain significant performance boost, and our CHEEM is better than all others. Similarly, we envision that leveraging the dynamically learned backbone by our CHEEM in constructing the external memory will be a promising direction to be studied in future work.

ii) Randomly Assigned Yet *Balanced* Classes in Streaming Tasks Challenge the Task-Centroid based Task ID Inference. On the one hand, for streaming tasks with balanced but randomly sampled classes per task (without replacement), our CHEEM under CCL has the worst performance (64.72% vs 76.48% by CODA-Prompt), which was caused by the low average precision of task ID inference (see Fig. 6 and analyses in Appendix B). Our CHEEM under TCL obtains the best performance (92.47%), which shows that the learned task-aware backbone is expressive. Overall, task centroids computed using the CLS tokens in the base model are not able to distinguish tasks from each other with high accuracy. One potential solution is to leverage the dynamically learned backbone by our CHEEM in constructing the external memory, considering its superior performance under TCL, at the expense of more costly task ID inference. On the other hand, *although it is interesting to test continual learning approaches using the Split ImageNet-R benchmark, the random composition of* *classes in a task is not natural in comparison with scenarios in natural human learning. Unlike the Split ImageNet-R, the VDD benchmark may suit continual learning better from perspective the perspective of real-world scenarios, for which our CHEEM works the best.*

4 VISUALIZATION AND ABLATION STUDIES

Due to space limit, we present visualizations of the learned CHEEM (examples like Fig. 2) in Appendix C. We also conduct ablation studies including: **(i)** The proposed HEE sampling significantly outperforms pure exploration (PE) sampling in Appendix D.1. **(ii)** The proposed HEE-empowered SPOS NAS significantly outperforms the DARTS (Liu et al., 2019a) based NAS used in the learn-to-grow method (Li et al., 2019) in Appendix D.2. **(iii)** The proposed internal memory learning method outperforms three state-of-the-art methods: SupSup (Wortsman et al., 2020), EFT (Verma et al., 2021) and LL (Ge et al., 2023) under the task-to-task transfer learning setting in Appendix D.3. And, more are in Appendix D.3, D.4 and D.6.

5 RELATED WORK

Experience Replay Based approaches aim to retain some exemplars, in the form of either raw data or latent features, from the previous tasks and replay them to the model along with the data from a new task (Aljundi et al., 2019b; Rebuffi et al., 2017b; Aljundi et al., 2019a; Balaji et al., 2020; Bang et al., 2021; Chaudhry et al., 2021; Pham et al., 2021), or *generative replay methods* (Shin et al., 2017; Cong et al., 2020) which replay exemplars sampled from the learned generators along with the data from the current task. For exemplar-free continual learning, *Regularization-based Approaches* explicitly prevent model parameters from deviating too far from their stable values learned on previous tasks when learning a new task (Aljundi et al., 2018; 2019c; Douillard et al., 2020; Nguyen et al., 2018; Kirkpatrick et al., 2017; Li & Hoiem, 2018; Zenke et al., 2017; Schwarz et al., 2018).

Dynamic Models aim to use different parameters for each task to eliminate the use of stored exemplars. Dynamically Expandable Network (Yoon et al., 2018) adds neurons to a network based on learned sparsity constraints and heuristic loss thresholds. PathNet (Fernando et al., 2017) and RPSNet (Rajasegaran et al., 2019) learn task-specific submodules or paths. Progressive Neural Networks (Rusu et al., 2016) learn a new network per task and adds lateral connections to the previous tasks' networks. The L2G (Li et al., 2019) uses Differentiable Architecture Search (DARTS) (Liu et al., 2019a) to determine if a layer can be reused, adapted, or renewed for a task, which is tested for ConvNets and the learning-to-grow operations are applied uniformly at each layer in a ConvNet. Our method is motivated by the L2G method, but with significant differences. NAS has also been explored by (Gao et al., 2023a) for continual learning on small scale tasks. Approaches that learn task-specific components on top of a backbone include (Ge et al., 2023; Verma et al., 2021; Wortsman et al., 2020; Abati et al., 2020).

Recently, there has been increasing interest in continual learning using Vision Transformers (Wang et al., 2022d;c; Xue et al., 2022; Ermis et al., 2022; Douillard et al., 2022; Pelosin et al., 2022; Yu et al., 2021; Li et al., 2022; Iscen et al., 2022; Wang et al., 2022a;b; Mohamed et al., 2023; Gao et al., 2023b; Zhou et al., 2023). *Prompt Based approaches* learn external parameters appended to the data tokens that encode task-specific information useful for classification (Wang et al., 2022d;a; Douillard et al., 2022; Smith et al., 2023a; Jung et al., 2023; Tang et al., 2023; Zhou et al., 2024). Our proposed method is complementary to prompting-based based approaches.

6 CONCLUSION

This paper presents a method of transforming Vision Transformers (ViTs) for exemplar-free class-incremental continual learning (ExfCCL), dubbed **CHEEM** (Continual Hierarchical-Exploration-Exploitation Memory). Our CHEEM consists of the external (task-centroid) memory and the internal (parameter) memory. The former is for task ID inference for test data based on clustered task centroids in training. The latter is realized by a proposed Hierarchical-Exploration-Exploitation (HEE) sampling based neural architecture search algorithm. The external and internal memory are maintained in a decoupled way. Our CHEEM is tested on two challenging benchmarks, the VDD benchmark and the continual ImageNet-R benchmark. It obtains state-of-the-art performance on the VDD outperforming the prior art by a large margin, with sensible CHEEM strutures continually learned. It obtains on-par performance on the imbalanced ImageNet-R benchmark, while performs worse on the balanced one, for which we analyze the reason thoroughly due to the external memory.

REFERENCES

Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020, pp. 3930–3939. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.00399. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Abati_Conditional_Channel_Gated_Networks_for_Task-Aware_Continual_Learning_CVPR_2020_paper.html.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss (eds.), Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part III, volume 11207 of Lecture Notes in Computer Science, pp. 144–161. Springer, 2018. doi: 10.1007/978-3-030-01219-9_9. URL https://doi.org/10.1007/978-3-030-01219-9_9.

Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pp. 11849–11860, 2019a. URL https://proceedings.neurips.cc/paper/2019/hash/15825aee15eb335cc13f9b559f166ee8-Abstract.html.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pp. 11816–11825, 2019b. URL https://proceedings.neurips.cc/paper/2019/hash/e562cd9c0768d5464b64cf61da7fc6bb-Abstract.html.

Rahaf Aljundi, Marcus Rohrbach, and Tinne Tuytelaars. Selfless sequential learning. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net, 2019c. URL https://openreview.net/forum?id=Bkxbrn0cYX.

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. CoRR, abs/1607.06450, 2016. URL http://arxiv.org/abs/1607.06450.

Yogesh Balaji, Mehrdad Farajtabar, Dong Yin, Alex Mott, and Ang Li. The effectiveness of memory replay in large scale continual learning. CoRR, abs/2010.02418, 2020. URL https://arxiv.org/abs/2010.02418.

Jihwan Bang, Heesu Kim, Youngjoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021, pp. 8218–8227. Computer Vision Foundation / IEEE, 2021. doi: 10.1109/CVPR46437.2021.00812. URL https://openaccess.thecvf.com/content/CVPR2021/html/Bang_Rainbow_Memory_Continual_Learning_With_a_Memory_of_Diverse_Samples_CVPR_2021_paper.html.

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. CoRR, abs/1308.3432, 2013. URL http://arxiv.org/abs/1308.3432.

Hakan Bilen, Basura Fernando, Efstratios Gavves, Andrea Vedaldi, and Stephen Gould. Dynamic image networks for action recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pp. 3034–3042. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.331. URL https://doi.org/10.1109/CVPR.2016.331.

Arslan Chaudhry, Puneet Kumar Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss (eds.), Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XI, volume 11215 of Lecture Notes in Computer Science, pp. 556–572. Springer, 2018. doi: 10.1007/978-3-030-01252-6_33. URL https://doi.org/10.1007/978-3-030-01252-6_33.

Arslan Chaudhry, Albert Gordo, Puneet K. Dokania, Philip H. S. Torr, and David Lopez-Paz. Using hindsight to anchor past knowledge in continual learning. In Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021, pp. 6993–7001. AAAI Press, 2021. doi: 10.1609/AAAI.V35I8.16861. URL https://doi.org/10.1609/aaai.v35i8.16861.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014, pp. 3606–3613. IEEE Computer Society, 2014. doi: 10.1109/CVPR.2014.461. URL https://doi.org/10.1109/CVPR.2014.461.

Yulai Cong, Miaoyun Zhao, Jianqiao Li, Sijia Wang, and Lawrence Carin. GAN memory with no forgetting. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/bf201d5407a6509fa536afc4b380577e-Abstract.html.

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 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (eds.), Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XX, volume 12365 of Lecture Notes in Computer Science, pp. 86–102. Springer, 2020. doi: 10.1007/978-3-030-58565-5_6. URL https://doi.org/10.1007/978-3-030-58565-5_6.

Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, pp. 9275–9285. IEEE, 2022. doi: 10.1109/CVPR52688.2022.00907. URL https://doi.org/10.1109/CVPR52688.2022.00907.

Beyza Ermis, Giovanni Zappella, Martin Wistuba, Aditya Rawal, and Cédric Archambeau. Continual learning with transformers for image classification. In IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2022, New Orleans, LA, USA, June 19-20, 2022, pp. 3773–3780. IEEE, 2022. doi: 10.1109/CVPRW56347.2022.00422. URL https://doi.org/10.1109/CVPRW56347.2022.00422.

Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. CoRR, abs/1701.08734, 2017. URL http://arxiv.org/abs/1701.08734.

Qiang Gao, Zhipeng Luo, Diego Klabjan, and Fengli Zhang. Efficient architecture search for continual learning. IEEE Trans. Neural Networks Learn. Syst., 34(11):8555–8565, 2023a. doi: 10.1109/TNNLS.2022.3151511. URL https://doi.org/10.1109/TNNLS.2022.3151511.

Qiankun Gao, Chen Zhao, Yifan Sun, Teng Xi, Gang Zhang, Bernard Ghanem, and Jian Zhang. A unified continual learning framework with general parameter-efficient tuning. In IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023, pp. 11449–11459. IEEE, 2023b. doi: 10.1109/ICCV51070.2023.01055. URL https://doi.org/10.1109/ICCV51070.2023.01055.

Yunhao Ge, Yuecheng Li, Di Wu, Ao Xu, Adam M. Jones, Amanda Sofie Rios, Iordanis Fostiropoulos, shixian wen, Po-Hsuan Huang, Zachary William Murdock, Gozde Sahin, Shuo Ni, Kiran Lekkala, Sumedh Anand Sontakke, and Laurent Itti. Lightweight learner for shared knowledge lifelong learning. Transactions on Machine Learning Research, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=Jjl2c8kWUc.

Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (eds.), Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XVI, volume 12361 of Lecture Notes in Computer Science, pp. 544–560. Springer, 2020. doi: 10.1007/978-3-030-58517-4_32. URL https://doi.org/10.1007/978-3-030-58517-4_32.

Tyler L. Hayes, Nathan D. Cahill, and Christopher Kanan. Memory efficient experience replay for streaming learning. In International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019, pp. 9769–9776. IEEE, 2019. doi: 10.1109/ICRA.2019.8793982. URL https://doi.org/10.1109/ICRA.2019.8793982.

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 2016, Las Vegas, NV, USA, June 27-30, 2016, pp. 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90.

Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. CoRR, abs/1606.08415, 2016. URL http://arxiv.org/abs/1606.08415.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. In 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021, pp. 8320–8329. IEEE, 2021. doi: 10.1109/ICCV48922.2021.00823. URL https://doi.org/10.1109/ICCV48922.2021.00823.

Ahmet Iscen, Thomas Bird, Mathilde Caron, Alireza Fathi, and Cordelia Schmid. A memory transformer network for incremental learning. In 33rd British Machine Vision Conference 2022, BMVC 2022, London, UK, November 21-24, 2022, pp. 388. BMVA Press, 2022. URL https://bmvc2022.mpi-inf.mpg.de/388/.

Dahuin Jung, Dongyoon Han, Jihwan Bang, and Hwanjun Song. Generating instance-level prompts for rehearsal-free continual learning. In IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023, pp. 11813–11823. IEEE, 2023. doi: 10.1109/ICCV51070.2023.01088. URL https://doi.org/10.1109/ICCV51070.2023.01088.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. URL http://arxiv.org/abs/1412.6980.

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. doi: 10.1073/pnas.1611835114. URL https://www.pnas.org/doi/abs/10.1073/pnas.1611835114.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. Science, 350(6266):1332–1338, 2015. doi: 10.1126/science.aab3050. URL https://www.science.org/doi/abs/10.1126/science.aab3050.

Duo Li, Guimei Cao, Yunlu Xu, Zhanzhan Cheng, and Yi Niu. Technical report for ICCV 2021 challenge sslad-track3b: Transformers are better continual learners. CoRR, abs/2201.04924, 2022. URL https://arxiv.org/abs/2201.04924.

Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, volume 97 of Proceedings of Machine Learning Research, pp. 3925–3934. PMLR, 2019. URL http://proceedings.mlr.press/v97/li19m.html.

Zhizhong Li and Derek Hoiem. Learning without forgetting. IEEE Trans. Pattern Anal. Mach. Intell., 40(12):2935–2947, 2018. doi: 10.1109/TPAMI.2017.2773081. URL https://doi.org/10.1109/TPAMI.2017.2773081.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net, 2019a. URL https://openreview.net/forum?id=S1eYHoC5FX.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net, 2019b. URL https://openreview.net/forum?id=rJlnB3C5Ym.

Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew B. Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. CoRR, abs/1306.5151, 2013. URL http://arxiv.org/abs/1306.5151.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of Psychology of Learning and Motivation, pp. 109–165. Academic Press, 1989. doi: https://doi.org/10.1016/S0079-7421(08)60536-8. URL https://www.sciencedirect.com/science/article/pii/S0079742108605368.

Abdelrahman Mohamed, Rushali Grandhe, K. J. Joseph, Salman H. Khan, and Fahad Shahbaz Khan. D^{3}former: Debiased dual distilled transformer for incremental learning. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023 - Workshops, Vancouver, BC, Canada, June 17-24, 2023, pp. 2421–2430. IEEE, 2023. doi: 10.1109/CVPRW59228.2023.00240. URL https://doi.org/10.1109/CVPRW59228.2023.00240.

Stefan Munder and Dariu M. Gavrila. An experimental study on pedestrian classification. IEEE Trans. Pattern Anal. Mach. Intell., 28(11):1863–1868, 2006. doi: 10.1109/TPAMI.2006.217. URL https://doi.org/10.1109/TPAMI.2006.217.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011, 2011. URL http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational continual learning. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, 2018. URL https://openreview.net/forum?id=BkQqq0gRb.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In Sixth Indian Conference on Computer Vision, Graphics & Image Processing, ICVGIP 2008, Bhubaneswar, India, 16-19 December 2008, pp. 722–729. IEEE Computer Society, 2008. doi: 10.1109/ICVGIP.2008.47. URL https://doi.org/10.1109/ICVGIP.2008.47.

Francesco Pelosin, Saurav Jha, Andrea Torsello, Bogdan Raducanu, and Joost van de Weijer. Towards exemplar-free continual learning in vision transformers: an account of attention, functional and weight regularization. In IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2022, New Orleans, LA, USA, June 19-20, 2022, pp. 3819–3828. IEEE, 2022. doi: 10.1109/CVPRW56347.2022.00427. URL https://doi.org/10.1109/CVPRW56347.2022.00427.

Quang Pham, Chenghao Liu, and Steven C. H. Hoi. Dualnet: Continual learning, fast and slow. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, pp. 16131–16144, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/86a1fa88adb5c33bd7a68ac2f9f3f96b-Abstract.html.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang (eds.), Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event, volume 139 of Proceedings of Machine Learning Research, pp. 8748–8763. PMLR, 2021. URL http://proceedings.mlr.press/v139/radford21a.html.

Jathushan Rajasegaran, Munawar Hayat, Salman H. Khan, Fahad Shahbaz Khan, and Ling Shao. Random path selection for continual learning. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pp. 12648–12658, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/83da7c539e1ab4e759623c38d8737e9e-Abstract.html.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pp. 506–516, 2017a. URL https://proceedings.neurips.cc/paper/2017/hash/e7b24b112a44fdd9ee93bdf998c6ca0e-Abstract.html.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pp. 5533–5542. IEEE Computer Society, 2017b. doi: 10.1109/CVPR.2017.587. URL https://doi.org/10.1109/CVPR.2017.587.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, pp. 8583–8595, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/48237d9f2dea8c74c2a72126cf63d933-Abstract.html.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. Int. J. Comput. Vis., 115(3):211–252, 2015. doi: 10.1007/S11263-015-0816-Y. URL https://doi.org/10.1007/s11263-015-0816-y.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. CoRR, abs/1606.04671, 2016. URL http://arxiv.org/abs/1606.04671.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In Jennifer G. Dy and Andreas Krause (eds.), Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018, volume 80 of Proceedings of Machine Learning Research, pp. 4535–4544. PMLR, 2018. URL http://proceedings.mlr.press/v80/schwarz18a.html.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pp. 2990–2999, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/0efbe98067c6c73dba1250d2beaa81f9-Abstract.html.

James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogério Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023, pp. 11909–11919. IEEE, 2023a. doi: 10.1109/CVPR52729.2023.01146. URL https://doi.org/10.1109/CVPR52729.2023.01146.

James Seale Smith, Junjiao Tian, Shaunak Halbe, Yen-Chang Hsu, and Zsolt Kira. A closer look at rehearsal-free continual learning. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023 - Workshops, Vancouver, BC, Canada, June 17-24, 2023, pp. 2410–2420. IEEE, 2023b. doi: 10.1109/CVPRW59228.2023.00239. URL https://doi.org/10.1109/CVPRW59228.2023.00239.

Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. CoRR, abs/1212.0402, 2012. URL http://arxiv.org/abs/1212.0402.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. Neural Networks, 32:323–332, 2012. doi: 10.1016/J.NEUNET.2012.02.016. URL https://doi.org/10.1016/j.neunet.2012.02.016.

Yu-Ming Tang, Yi-Xing Peng, and Wei-Shi Zheng. When prompt-based incremental learning does not meet strong pretraining. In IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023, pp. 1706–1716. IEEE, 2023. doi: 10.1109/ICCV51070.2023.00164. URL https://doi.org/10.1109/ICCV51070.2023.00164.

Sebastian Thrun and Tom M. Mitchell. Lifelong robot learning. Robotics Auton. Syst., 15(1-2): 25–46, 1995. doi: 10.1016/0921-8890(95)00004-Y. URL https://doi.org/10.1016/0921-8890(95)00004-Y.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. Journal of Machine Learning Research, 9(86):2579–2605, 2008. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Vinay Kumar Verma, Kevin J. Liang, Nikhil Mehta, Piyush Rai, and Lawrence Carin. Efficient feature transformations for discriminative and generative continual learning. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021, pp. 13865–13875. Computer Vision Foundation / IEEE, 2021. doi: 10.1109/CVPR46437.2021.01365. URL https://openaccess.thecvf.com/content/CVPR2021/html/Verma_Efficient_Feature_Transformations_for_Discriminative_and_Generative_Continual_Learning_CVPR_2021_paper.html.

Yabin Wang, Zhiwu Huang, and Xiaopeng Hong. S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022, 2022a. URL http://papers.nips.cc/paper_files/paper/2022/hash/25886d7a7cf4e33fd44072a0cd81bf30-Abstract-Conference.html.

Zhen Wang, Liu Liu, Yiqun Duan, Yajing Kong, and Dacheng Tao. Continual learning with lifelong vision transformer. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, pp. 171–181. IEEE, 2022b. doi: 10.1109/CVPR52688.2022.00027. URL https://doi.org/10.1109/CVPR52688.2022.00027.

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 Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (eds.), Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXVI, volume 13686 of Lecture Notes in Computer Science, pp. 631–648. Springer, 2022c. doi: 10.1007/978-3-031-19809-0_36. URL https://doi.org/10.1007/978-3-031-19809-0_36.

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 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, pp. 139–149. IEEE, 2022d. doi: 10.1109/CVPR52688.2022.00024. URL https://doi.org/10.1109/CVPR52688.2022.00024.

Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019.

Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in superposition. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/ad1f8bb9b51f023cdc80cf94bb615aa9-Abstract.html.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, pp. 374–382. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00046. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Wu_Large_Scale_Incremental_Learning_CVPR_2019_paper.html.

Mengqi Xue, Haofei Zhang, Jie Song, and Mingli Song. Meta-attention for vit-backed continual learning. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, pp. 150–159. IEEE, 2022. doi: 10.1109/CVPR52688.2022.00025. URL https://doi.org/10.1109/CVPR52688.2022.00025.

Peng Ye, Baopu Li, Yikang Li, Tao Chen, Jiayuan Fan, and Wanli Ouyang. β-darts: Beta-decay regularization for differentiable architecture search. In IEEE/CVF Conference on Computer

Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, pp. 10864–10873. IEEE, 2022. doi: 10.1109/CVPR52688.2022.01060. URL https://doi.org/10.1109/CVPR52688.2022.01060.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, 2018. URL https://openreview.net/forum?id=Sk7KsfW0-.

Pei Yu, Yinpeng Chen, Ying Jin, and Zicheng Liu. Improving vision transformers for incremental learning. CoRR, abs/2112.06103, 2021. URL https://arxiv.org/abs/2112.06103.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In Doina Precup and Yee Whye Teh (eds.), Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, volume 70 of Proceedings of Machine Learning Research, pp. 3987–3995. PMLR, 2017. URL http://proceedings.mlr.press/v70/zenke17a.html.

Da-Wei Zhou, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need. CoRR, abs/2303.07338, 2023. doi: 10.48550/ARXIV.2303.07338. URL https://doi.org/10.48550/arXiv.2303.07338.

Da-Wei Zhou, Hai-Long Sun, Han-Jia Ye, and De-Chuan Zhan. Expandable subspace ensemble for pre-trained model-based class-incremental learning. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024, pp. 23554–23564. IEEE, 2024. doi: 10.1109/CVPR52733.2024.02223. URL https://doi.org/10.1109/CVPR52733.2024.02223.

APPENDIX

OUTLINE

In this Appendix, we elaborate on the following aspects that are not presented in the submission due to the space limit:

- **Section A – Analyses of head classifier design choices** in continual learning.
- **Section B – Analysis of the discriminative capacity of centroids for task ID inference**
- **Section C – More examples of learned CHEEM** by our proposed hierarchical exploration-exploitation (HEE) sampling scheme and by the vanilla pure exploration (PE) sampling scheme.
- **Section D - Ablation Studies**:
  - **Section D.1 – Comparing Pure Exploration vs. our propoed Hierarchical Exploration-Exploitaiton sampling scheme for SPOS NAS**
  - **Section D.2 – Comparing proposed HEE empowered SPOS NAS with DARTS**: We compare with DARTS, which has been used by Learn-to-Grow and show that our proposed method outperforms L2G with DARTS.
  - **Section D.3 – Can CHEEM be used for task-to-task transfer?**
  - **Section D.4 – Can we use other components in ViTs as CHEEM?**: We compare with the Query/Key/Value layer and the FFN layer, and verify the effectiveness of the proposed identification in the main paper.
  - **Section D.5 – How is the `Adapt` operation implemented?**: We elaborate two implementation methods of the `Adapt` operation and compare their performance.
  - **Section D.6 - Effects of task orders.**
- **Section E – Details of the two benchmarks** tested in the experiments: the Visual Domain Decathlon (VDD) (Rebuffi et al., 2017a) benchmark and the ImageNet-R (Hendrycks et al., 2021) benchmark.
- **Section F – The base model and its training details**: the Vision Transformer (ViT) model specification (ViT-B/8) and how it is initially training on the ImageNet in our experiments.
- **Section G – Experimental settings and training hyperparameters in our implementation** on the VDD benchmark and the 5-dataset benchmark.
- **Section H** - Details of Modifying SupSup, EFT and LL on the VDD Benchmark to work with Vision Transformers:

A TASK-SPECIFIC HEAD SELECTION VIA TASK ID INFERENCE VS SHARING A GROWING, COMMON TASK HEAD

As aforementioned in the introduction of main text, explicitly inferring task IDs from the external memory for test data is to select the task head classifiers as done in (Wang et al., 2022a), while also enabling selecting task-specific feature backbones (exploited in our CHEEM). An alternative approach is to share a growing, common task head as done in (Wang et al., 2022d;c; Smith et al., 2023a). The former can suffer from low average precision, while for the latter we observe serious instability caused by imbalanced classes in stream tasks due to the discrepancy in training and testing (see Table 2 and Table 3).

To show the instability, let $\mathcal{C} = (z_1, z_2, \cdots, z_K)$ be the logits output of the common head. During training, the task ID for each $z_k \in \mathcal{C}$ is known, and the logits of previous tasks are masked out (using $-\infty$) before computing the softmax (i.e., task-specific heads are used in training). During testing, the label for a testing sample is predicated by $\arg\max_{k\in[1,K]} z_k$, across the entire head. This discrepancy between training and testing imposes a very strong assumption of the stability of logits in training and testing, which entails that the task-specific head segment for a testing example will underlyingly resemble the 1-hot vector, such that it can hold across the entire head. Although (Wang et al., 2022d;c; Smith et al., 2023a) show strong performance using the same number of classes per task (20) in ImageNet-R (Hendrycks et al., 2021), we observe that slightly varying the number of classes per task, while keeping everything else the same, can lead to around 10% drop of performance in ImageNet-R. For more challenging task-class distributions as in the VDD (Fig. 2), (Wang et al., 2022d;c; Smith et al., 2023a) have a catastrophic drop of performance.

To analyze this further, we probe the model after the training on the final task is complete. For each task, we first make predictions using the task specific part of the head (i.e., the task-incremental setting) and calculate the entropy of predicted probabilities of all the samples with the task-specific head. Then, we make predictions with the full head (i.e., the class-incremental setting), including all the classes in the previous tasks, and track samples for which the logit with the maximum values changes to one of the classes from the previous tasks, which we refer to as **switched samples**.

As shown in Fig. 5, we observe that the entropy for the switched samples is consistently higher than those of the un-switched samples. Furthermore, the variance of the the entropy consistently goes down as the tasks progress, which indicates that for later tasks, only the predictions that are highly confident remain unswitched. However, the entropy for the switched samples shows a large variance. This indicates that sharp predictions are not a sufficient condition to avoid switching.

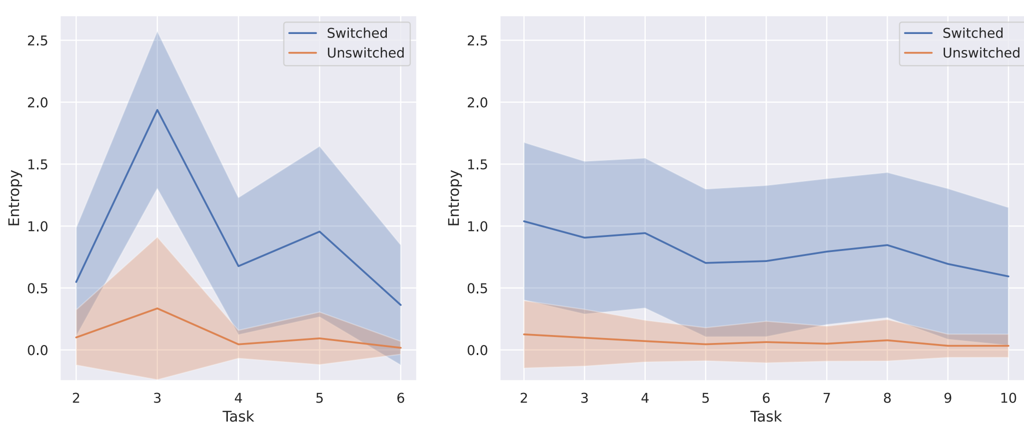


Figure 5: Mean entropy for the switched and unswitched samples from each task on the Split ImageNet-R benchmarks using CODA-Prompt. *Left:* Imbalanced ImageNet-R with varying number of classes (5,10,15,20, 50, 100) per task and 6 tasks in total. *Right:* Balanced ImageNet-R with same number (20) of classes per task and 10 tasks in total.

B The Pros and Cons of Base Model Computed Task Centroids as the External Memory for Task ID Inference

To understand effects of the external task-centroid memory we adopted from S-Prompts (Wang et al., 2022a), we analyze the separability of the K-Mean clusters of different tasks on the VDD (Rebuffi et al., 2017a) (see results in Table 2) and Balanced ImageNet-R benchmarks (see results in Table 3) by visualizing them using t-SNE (van der Maaten & Hinton, 2008) in Fig. 6.

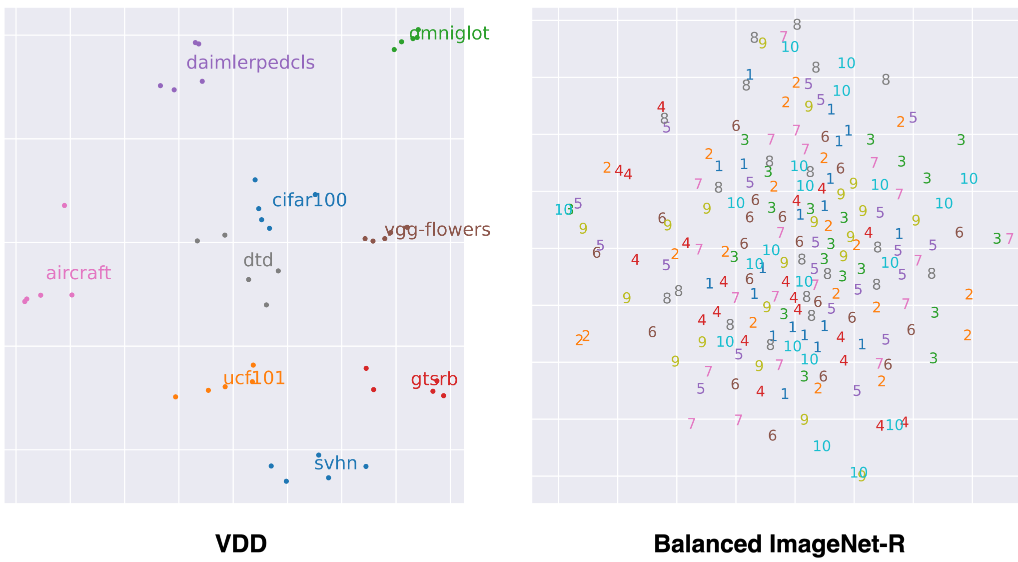


Figure 6: The t-SNE visualization of task centroids using the frozen base model `CLS` tokens of training examples of each task. We run K-Mean on the VDD benchmark with $K = 5$ per task. We use the mean `CLS` token per class (20 centroids per task) on the balanced Imagenet-R benchmark, which we found works better.

For our proposed CHEEM, we can observe that the external task-centroid memory is effective on the VDD, providing sufficiently high recall rates for the internal memory of CHEEM and resulting in overall high average accuracy and negligible average forgetting. On the contrary, the external task-centroid memory of the balanced Imagenet-R benchmark explains why our CHEEM performs much worse under CCL. We note that the external memory is constructed based on the frozen base model. As discussed in the main text, a potential improvement is to leverage dynamically updated backbones via the internal memory of our CHEEM in computing the task centroids at the expense of increased task ID inference cost. That is also to shift from our current decoupled external and internal memory design to integrative / coupled external and internal memory.

Together with the experimental results that the prompting based methods (L2P, DualPrompt and CODA-Prompt) can handle balanced ImageNet-R better in terms of preventing switched samples as analyzed in Appendix A, and that regularized based method (EWC and L2 Regularization in Table 2) can handle VDD better than prompting based methods, we envision there are opportunities of exploring the integration between them and the internal memory of our CHEEM for more robust ExfCCL across scenarios including both VDD and balanced/imbalanced ImageNet-R.

C Examples of CHEEM Learned Sequentially and Continually

Fig. 7 shows the CHEEM learned sequentially and continually via the proposed HEE-based NAS on the VDD benchmark (Rebuffi et al., 2017a) with three different random seeds.

As comparisons, Fig. 8 shows the structure updates learned using the vanilla PE-based NAS. It can be seen that pure exploration does not reuse components from similar tasks. The pure exploration based method adds many unnecessary `Adapt` and `New` operations even though the tasks are similar (e.g., ImNet → C100), verifying the effectiveness of the proposed sampling method. While the pure exploration scheme adds many `Skip` operations, thereby reducing the overall FLOPs, the average accuracy is low by a large margin, about 6%. This shows that the pure exploration scheme cannot learn to choose operations in a task synergy aware way.

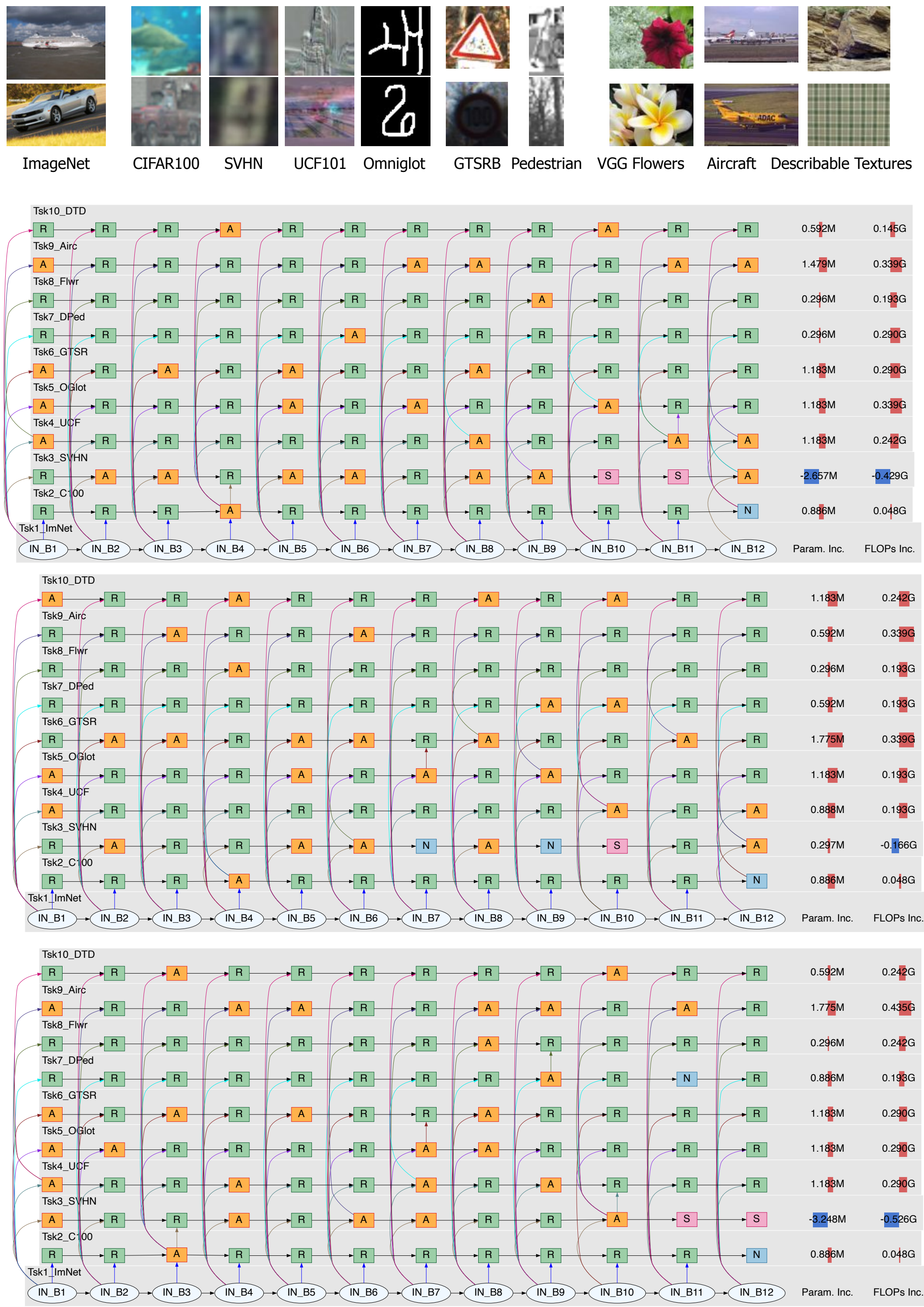


Figure 7: Examples of the task-synergy memory (CHEEM) learned on the VDD benchmark (Rebuffi et al., 2017a) with the task sequence shown in the top **using our proposed HEE-based NAS** and three different random seeds. The overall performance is reported in Table 2 in the main paper. S, R, A and N represent `Skip`, `Reuse`, `Adapt` and `New` respectively. The last two columns show the number of new task-specific parameters and added FLOPs respectively, in comparison with the first task, ImNet model. Overall, the learned task synergies make intuitive sense and remain relatively stable across different random seeds.

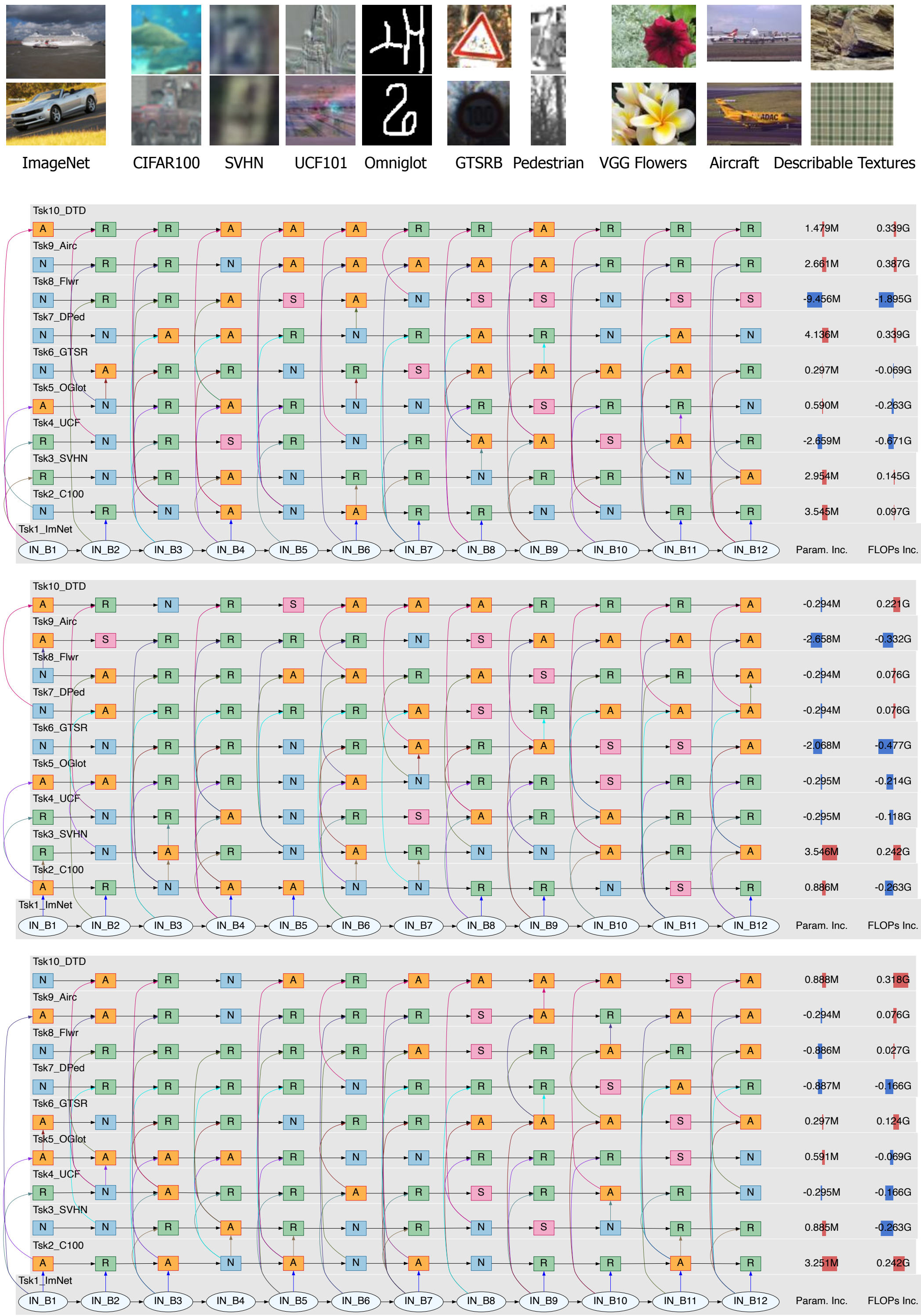


Figure 8: Examples of the task-synergy memory (CHEEM) learned on the VDD benchmark (Rebuffi et al., 2017a) with the task sequence shown in the top **using the vanilla PE-based NAS** and three different random seeds. S, R, A and N represent `Skip`, `Reuse`, `Adapt` and `New` respectively. The overall performance is reported in Table 4 in this appendix and our proposed HEE-based NAS significantly improves the performance by an absolute 6% average accuracy. In terms of the learned CHEEM, the PE-based NAS leads to much more `New` operations, which shows it is less effective in terms of leveraging task synergies.

D Ablation Studies

D.1 Hierarchical Exploration and Exploitation (HEE) vs. Pure Exploration (PE)

We verify the effectiveness of our HEE sampling (Table 4), which is significantly better by a large margin, 6% absolute average accuracy increase.

Table 4: The effectiveness of our proposed hierarchical exploration and exploitation sampling empowered SPOS NAS (Fig. 4). The structure updates learned by the PE strategy are visualized in Fig. 8 in the Appendix.

Method	C100	SVHN	UCF	OGlt	GTSR	DPed	Flwr	Airc.	DTD	Avg. Accuracy	Avg. Forgetting
PE	80.34	94.75	72.63	81.43	99.13	99.59	72.45	38.27	37.22	75.09 ± 0.81	0.32 ± 0.00
HEE	88.56	95.63	75.05	83.81	99.15	99.64	90.92	55.53	56.42	82.74 ± 0.54	0.33 ± 0.0

D.2 Comparison with DARTS

Table 5 shows the effectiveness of the proposed HEE empowered SPOS NAS against DARTS, as used in the original learn-to-grow method. Our proposed HEE empowered SPOS NAS outperforms DARTS, and the more advanced β-DARTS (Ye et al., 2022) by a large margin. We compare with Learn-to-Grow under a task-incremental continual learning (TCL) setting as used in the original formulation of L2G by (Li et al., 2019).

Table 5: The effectiveness of our proposed HEE sampling empowered SPOS NAS (Fig. 4). The two L2G variants use the same ViT-B base model as our HEE.

Method	C100	SVHN	UCF	OGlt	GTSR	DPed	Flwr	Airc.	DTD	Avg. Accuracy
L2G (DARTS)	88.47	85.20	79.22	80.19	99.28	98.06	76.14	39.29	46.01	77.45 ± 2.41
L2G (β-DARTS)	88.95	94.73	75.31	79.76	99.84	99.76	78.86	34.50	47.09	78.14 ± 0.54
HEE	90.93	95.96	80.74	83.25	99.94	99.96	94.12	58.90	60.05	84.65 ± 0.33

D.3 Results by Task-to-Task Transfer based continual Learning

To evaluate the transfer learning capabilities of CHEEM, we compare with three state-of-the-art methods: Supermasks in Superposition (SupSup) (Wortsman et al., 2020), Efficient Feature Transformation (EFT) (Verma et al., 2021) and Lightweight Learner (LL) (Ge et al., 2023) that modify the backbone model separately for each taskm, which we refer to as a task-to-task transfer setting. We modify all the methods to work with ViTs. We provide the details of the modifications in Section H. All three methods are fine-tuned on a new task for 50 epochs. For CHEEM, we use 50 epochs for the supernet training and 30 epochs for fine-tuning.

Table 6 demonstrates that CHEEM achieves the highest average accuracy on the VDD benchmark. Although its performance on individual tasks may be lower, CHEEM exhibits robustness to domain variations, as reflected in its overall average accuracy. In contrast, other methods show varying performance depending on the domain. SupSup performs well on tasks that are significantly different from the base ImageNet task, such as UCF 101, Omniglot, and SVHN, but performs poorly on tasks closely related to ImageNet. Conversely, EFT and LL excel on tasks similar to ImageNet but struggle with out-of-distribution tasks. Despite its lower individual task performance, CHEEM's consistent performance across diverse downstream tasks highlights its adaptability and makes it more general.

Table 6: Results on the VDD benchmark (Rebuffi et al., 2017a) using ViT-B/8 (Dosovitskiy et al., 2021) under the task-to-task transfer based continual learning protocol. The learned CHEEM are visualized in Fig. 10 in the Appendix.

Method	ImNet	C100	SVHN	UCF	OGlt	GTSR	DPed	Flwr	Airc.	DTD	Avg. Accuracy
SupSup	82.65	89.96	**96.05**	**81.68**	**84.60**	**99.97**	99.97	78.76	44.18	51.60	81.14 ± 0.04
EFT	82.65	91.86	93.51	73.89	75.62	99.58	**99.98**	96.34	48.17	**64.40**	82.60 ± 0.07
LL	82.65	**91.92**	93.90	75.63	77.07	99.71	99.96	**96.47**	49.33	64.34	83.10 ± 0.02
Our CHEEM	82.65	90.93	95.96	80.74	83.25	99.94	99.96	94.12	**58.90**	60.05	$\mathbf{84.65} \pm 0.33$

D.4 CHEEM PLACED AT OTHER VIT COMPONENTS

Table 7 shows the performance comparisons with other four different components in the ViT (the Query/Key/Value linear projection layer and the FFN block) used in realizing the proposed CHEEM. The Query/Key/Value component as the CHEEM does not perform as well as the Projection component. The FFN block as the CHEEM performs only slightly better than the Projection layer, but at the expense of a much larger parameter cost. This reinforces our identification above.

Table 7: Results of ablation study on other components of the ViT used for realizing the CHEEM. The results have been averaged over 3 different seeds.

Component	ImNet	C100	SVHN	UCF	OGlt	GTSR	DPed	Flwr	Airc.	DTD	Avg. Accuracy	Avg. Param. Inc./task (M)
Projection	82.65	90.54	96.12	75.53	83.81	99.93	99.88	91.21	55.59	59.18	83.44 ± 0.50	1.06 ± 0.04
Query	82.65	89.66	93.74	71.53	82.02	99.87	99.89	90.03	49.57	59.40	81.84 ± 0.32	2.38 ± 0.12
Key	82.65	89.29	94.77	72.25	81.86	99.86	99.90	88.86	51.72	60.46	82.16 ± 0.17	2.41 ± 0.03
Value	82.65	84.94	95.90	75.85	84.68	99.89	99.89	86.54	48.83	55.37	81.46 ± 0.25	1.70 ± 0.11
FFN	82.65	91.05	96.08	76.96	85.22	99.94	99.94	93.79	56.74	59.61	84.20 ± 0.28	2.31 ± 0.28

D.5 IMPLEMENTATION DETAILS OF THE `Adapt` OPERATION

How to `Adapt` in a sustainable way? The proposed `Adapt` operation will effectively increase the depth of the network in a plain way. In the worst case, if too many tasks use `Adapt` on top of each other, we will end up stacking too many MLP layers together. This may lead to unstable training due to gradient vanishing. Shortcut connections (He et al., 2016) have been shown to alleviate the gradient vanishing and exploding problems, making it possible to train deeper networks. We introduce the shortcut connection in adding a MLP `Adapt` operation. We test two different implementations: with shortchut in all the three components (supernet training, target network selection and target network finetuing) versus with shortcut only in target network finetuning (i.e., without shortcut in the NAS including both supernet training and target network selection).

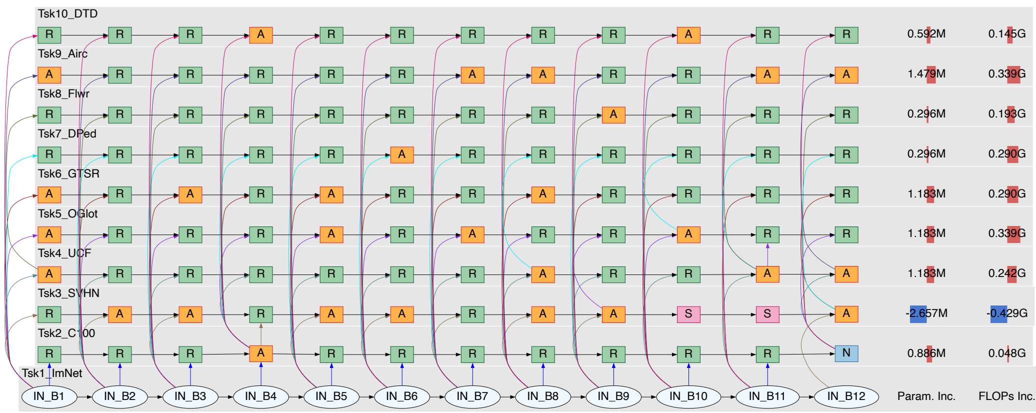
Table 8: Results of the ablation study on the implementation of the `Adapt` operation: with (w/) vs without (w/o) shortcut connection for the MLP `Adapt` layer in NAS. The first two rows are for the sequential and continual paradigm and the last two rows for the task-to-task (T2T) transfer based paradigm.

Shortcut in NAS	ImNet	C100	SVHN	UCF	OGlt	GTSR	DPed	Flwr	Airc.	DTD	Avg. Accuracy	Avg. Param. ↑/task (M)	Avg. FLOPs ↑/task (G)
w/o	82.65	90.54	96.12	75.53	83.81	99.93	99.88	91.21	55.59	59.18	83.44 ± 0.50	1.06 ± 0.04	0.17 ± 0.01
w/	82.65	91.18	96.18	82.34	86.03	99.91	99.95	91.60	58.90	58.56	84.73 ± 0.19	2.01 ± 0.18	0.49 ± 0.13
w/o (T2T)	82.65	90.93	95.96	80.74	83.25	99.94	99.96	94.12	58.90	60.05	84.65 ± 0.33	2.61 ± 0.15	-0.19 ± 0.09
w/ (T2T)	82.65	91.24	99.25	84.14	85.99	99.97	99.95	94.64	60.34	61.63	85.68 ± 0.16	3.23 ± 0.12	0.01 ± 0.02

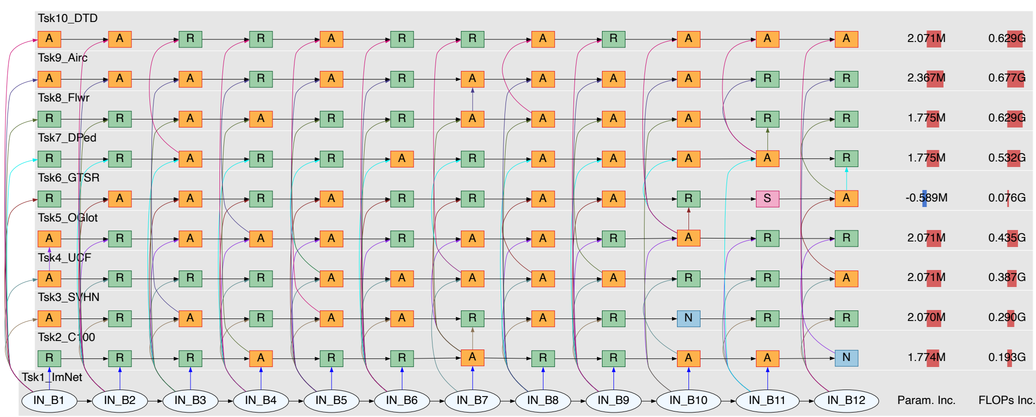
Table 8 shows the performance comparisons on the VDD Benchmark under the continual learning paradigm. In terms of sequentially introduced complexities, a more compact model is learned without the shortcut in the `Adapt` during NAS (supernet training and target network selection) as evidenced by the number of additional parameters and the increase in FLOPs. Using the shortcut in both supernet training and target network selection results in twice the parameter increase, and almost $4\times$ increase in FLOPs. Fig. 9 and Fig. 10 show comparisons of the learned CHEEM by the two implementation methods under the two paradigms respectively.

Remarks. We have two remarks as follows.

- We use the more parameter-efficient implementation (i.e., w/o shortcut for the `Adapt` in NAS) in the main paper for both the continual learning and the task-to-task transfer learning paradigms, even though the counterparts have better performance.
- We note that although the T2T paradigm results in larger parameter increase per task, its computational costs are relatively lower due to either more `Skip` operations learned and/or the fact that there is no `Adapt`-on-`Adapt` operations since it is task-to-task transfer based learning.

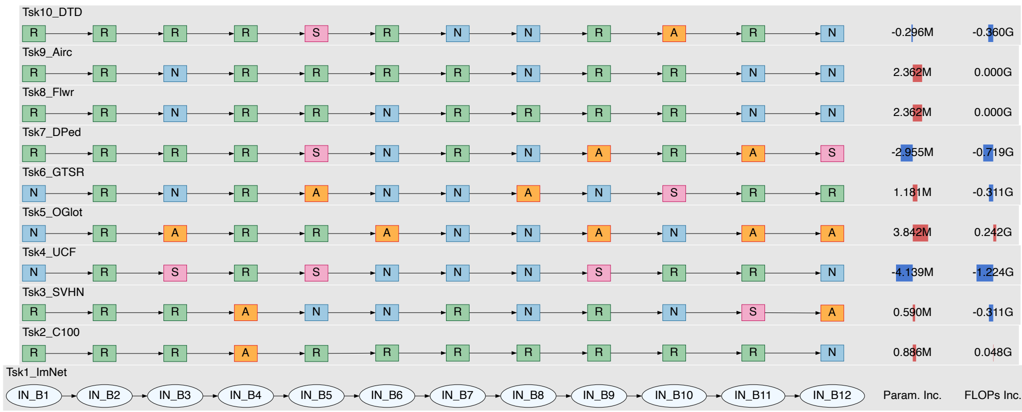


(a) An example of CHEEM learned **without** the shortcut for the MLP `Adapt` layer in NAS (the same one as the 2nd row in Fig. 7).

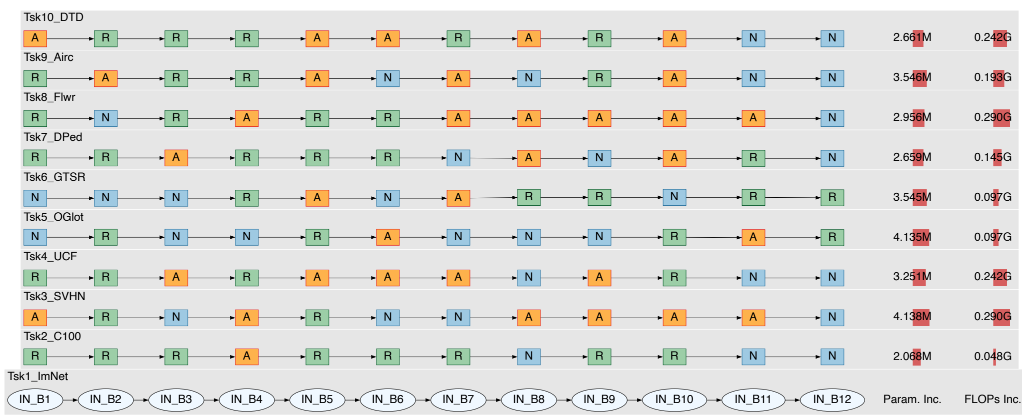


(b) An example of CHEEM learned **with** the shortcut for the MLP `Adapt` layer in NAS. More `Adapt` on top of `Adapt` operations are learned.

Figure 9: Comparisons between CHEEM learned by two different implementations of the MLP `Adapt` operation under the sequential and continual learning paradigm. S, R, A and N represent `Skip`, `Reuse`, `Adapt` and `New` respectively.



(a) An example of CHEEM learned **without** the shortcut for the MLP `Adapt` layer in NAS.



(b) An example of CHEEM learned **with** the shortcut for the MLP `Adapt` layer in NAS. More `Adapt` operations are learned.

Figure 10: Comparisons between CHEEM learned by two different implementations of the `Adapt` operation under the task-to-task (T2T) transfer based lifelong learning setting. Here all the 9 tasks are transferred from the base Tsk1_ImNet model, so we omit the arrows linking the blocks for clarity. S, R, A and N represent `Skip`, `Reuse`, `Adapt` and `New` respectively.

D.6 EFFECTS OF TASK ORDERS

We investigate the effects of task orders of the 9 tasks in the VDD benchmark. We test four more task sequences in addition to the one presented in the main paper. Overall, The CHEEM learned by our proposed HEE-based NAS achieve similar performance across different task orders, and consistently significantly outperform those learned by the vanilla PE-based NAS. We note that since the task ID inference is based on a frozen backbone, it is independent of the task order. Hence, for simplicity, we evaluate the effect of task order in a task-incremental setting.

Table 9 reports the performance. Fig. 11, Fig. 12, Fig. 13 and Fig. 14 show the learned CHEEM using our proposed HEE-based NAS.

Table 9: Results of ablation study on CHEEM learning with four different task orders using both our proposed HEE-based NAS and the vanilla PE-based NAS. The results have been averaged over 3 different seeds.

NAS	ImNet	OGlt	UCF	Airc.	Flwr	SVHN	DTD	GTSR	DPed	C100	Avg. Accuracy	Avg. Param. Inc./task (M)
HEE	82.65	84.32	75.27	54.32	90.29	95.83	57.89	99.92	99.72	89.96	83.02 ± 0.31	1.25 ± 0.15
PE	82.65	77.41	70.12	39.40	64.35	94.12	37.02	99.83	99.41	70.78	73.51 ± 0.80	2.86 ± 0.14
NAS	ImNet	DPed	SVHN	DTD	Airc.	OGlt	C100	GTSR	Flwr	UCF	Avg. Accuracy	Avg. Param. Inc./task (M)
HEE	82.66	99.94	95.83	58.56	42.43	83.55	89.98	99.95	91.99	75.67	82.06 ± 1.28	1.42 ± 0.05
PE	82.66	99.65	95.04	45.66	35.87	77.62	71.51	99.85	66.11	63.99	73.79 ± 0.50	2.85 ± 0.12
NAS	ImNet	UCF	C100	OGlt	GTSR	DTD	Flwr	SVHN	DPed	Airc.	Avg. Accuracy	Avg. Param. Inc./task (M)
HEE	82.66	79.73	90.75	84.93	99.90	58.14	91.27	96.05	99.89	54.06	83.74 ± 0.51	1.37 ± 0.05
PE	82.66	74.49	74.17	78.76	99.91	41.01	70.49	94.15	99.25	37.77	75.27 ± 2.41	2.75 ± 0.16
NAS	ImNet	Flwr	UCF	OGlt	GTSR	DPed	C100	Airc.	DTD	SVHN	Avg. Accuracy	Avg. Param. Inc./task (M)
HEE	82.66	87.52	77.17	84.20	99.92	99.80	90.30	54.49	56.83	96.03	82.89 ± 0.58	1.41 ± 0.09
PE	82.66	72.75	76.31	78.47	99.89	99.42	70.09	34.95	39.89	93.72	74.81 ± 1.40	2.70 ± 0.11

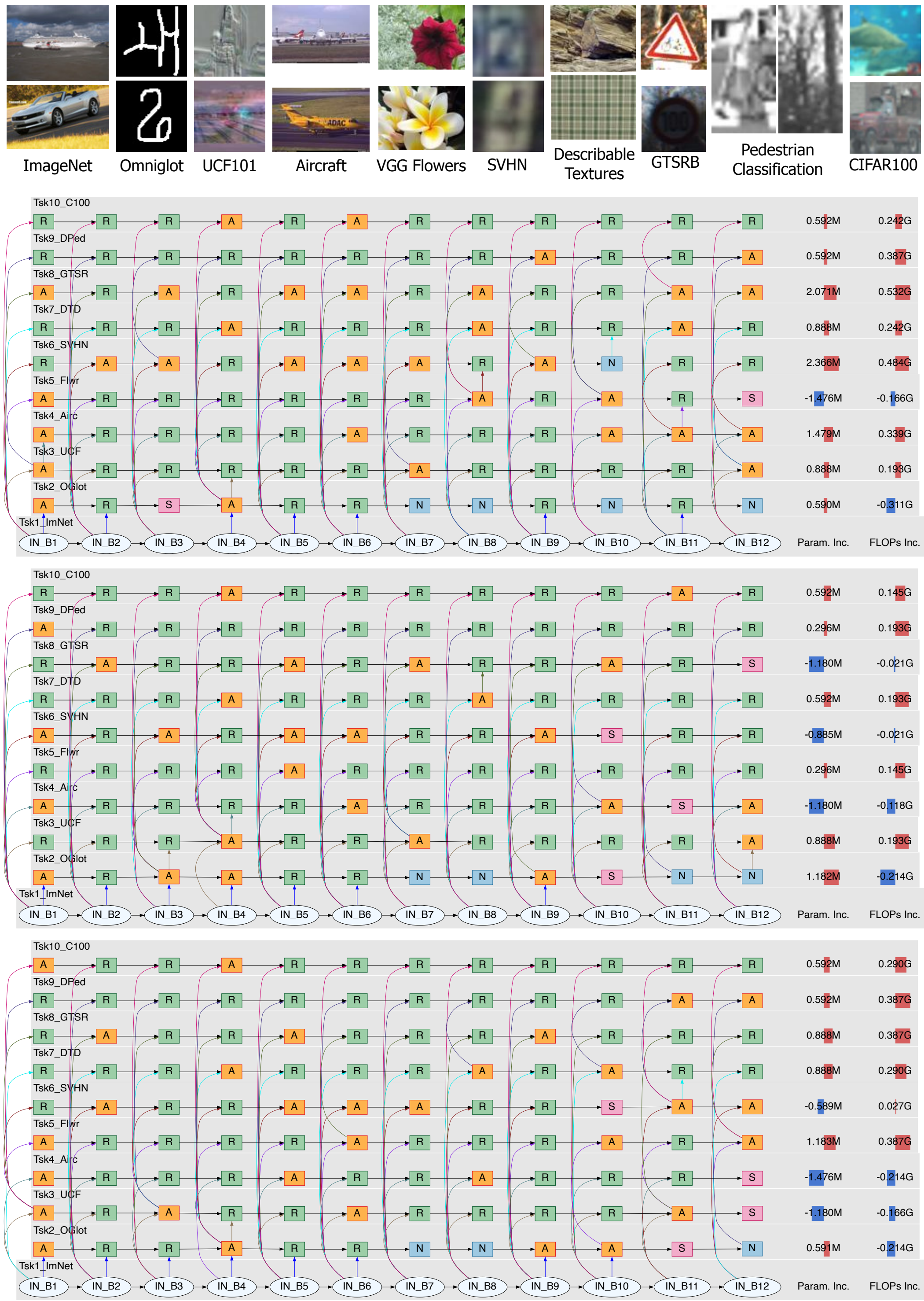


Figure 11: Examples of the task-synergy memory (CHEEM) learned on the VDD benchmark (Rebuffi et al., 2017a) with the task sequence shown in the top **using our proposed HEE-based NAS** and three different random seeds. The overall performance is reported in Table 9. S, R, A and N represent `Skip`, `Reuse`, `Adapt` and `New` respectively. The last two columns show the number of new task-specific parameters and added FLOPs respectively, in comparison with the first task, ImNet model. Overall, the learned task synergies make intutive sense and remain relatively stable across different random seeds.

Figure 12: Examples of the task-synergy memory (CHEEM) learned on the VDD benchmark (Rebuffi et al., 2017a) with the task sequence shown in the top **using our proposed HEE-based NAS** and three different random seeds. The overall performance is reported in Table 9. S, R, A and N represent `Skip`, `Reuse`, `Adapt` and `New` respectively. The last two columns show the number of new task-specific parameters and added FLOPs respectively, in comparison with the first task, ImNet model. Overall, the learned task synergies make intutive sense and remain relatively stable across different random seeds.

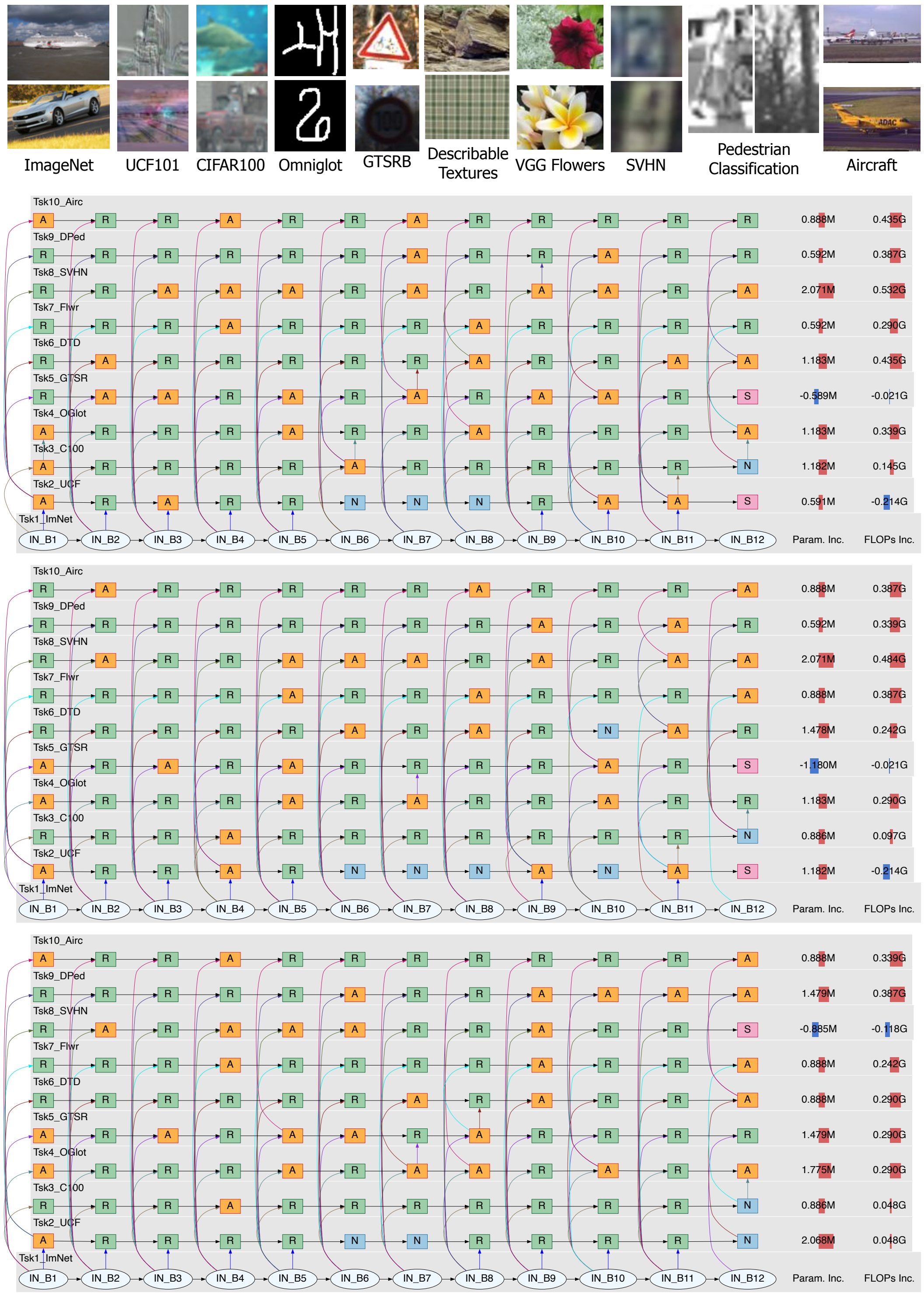


Figure 13: Examples of the task-synergy memory (CHEEM) learned on the VDD benchmark (Rebuffi et al., 2017a) with the task sequence shown in the top **using our proposed HEE-based NAS** and three different random seeds. The overall performance is reported in Table 9. S, R, A and N represent `Skip`, `Reuse`, `Adapt` and `New` respectively. The last two columns show the number of new task-specific parameters and added FLOPs respectively, in comparison with the first task, ImNet model. Overall, the learned task synergies make intutive sense and remain relatively stable across different random seeds.

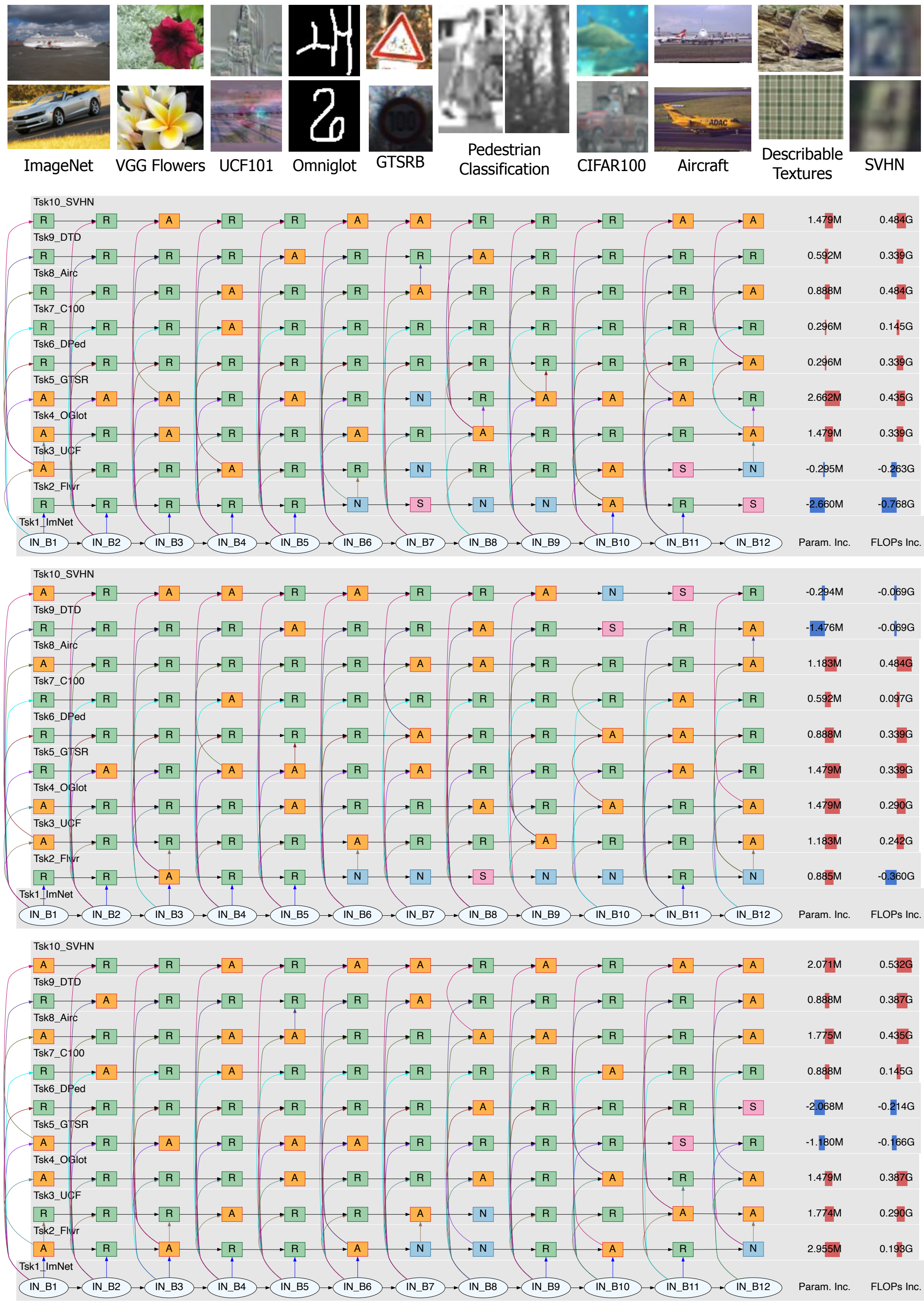


Figure 14: Examples of the task-synergy memory (CHEEM) learned on the VDD benchmark (Rebuffi et al., 2017a) with the task sequence shown in the top **using our proposed HEE-based NAS** and three different random seeds. The overall performance is reported in Table 9. S, R, A and N represent `Skip`, `Reuse`, `Adapt` and `New` respectively. The last two columns show the number of new task-specific parameters and added FLOPs respectively, in comparison with the first task, ImNet model. Overall, the learned task synergies make intutive sense and remain relatively stable across different random seeds.

E Dataset Details

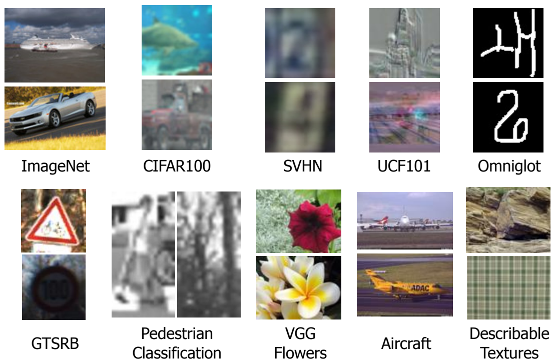


Figure 15: Example images from the VDD benchmark (Rebuffi et al., 2017a). Each task has a significantly different domain than others, making VDD a challenging benchmark for lifelong learning.

Table 10: The number of samples in training, validation and testing sets per task used in our experiments on the VDD benchmark (Rebuffi et al., 2017a).

Task	Train	Validation	Test	#Categories
ImageNet12	1108951	123216	49000	1000
CIFAR100	36000	4000	10000	10
SVHN	42496	4721	26040	10
UCF	6827	758	1952	101
Omniglot	16068	1785	6492	1623
GTSR	28231	3136	7842	43
DPed	21168	2352	5880	2
VGG-Flowers	918	102	1020	102
Aircraft	3001	333	3333	100
DTD	1692	188	1880	47

E.1 The VDD Benchmark

It consists of 10 tasks: ImageNet-1k (Russakovsky et al., 2015), CIFAR100 (Krizhevsky et al., 2009), SVHN (Netzer et al., 2011), UCF101 Dynamic Images (UCF) (Soomro et al., 2012; Bilen et al., 2016), Omniglot (Lake et al., 2015), German Traffic Signs (GTSR) (Stallkamp et al., 2012), Daimler Pedestrian Classification (DPed) (Munder & Gavrila, 2006), VGG Flowers (Nilsback & Zisserman, 2008), FGVC-Aircraft (Maji et al., 2013), and Describable Textures (DTD) (Cimpoi et al., 2014). All the images in the VDD benchmark have been scaled such that the shorter side is 72 pixels. Table 10 shows the number of samples in each task. Fig. 15 shows examples of images from each task of the VDD benchmark. In our experiments, we use 10% of `the official training data` from each of the tasks for validation (e.g., used in the target network selection in Section 3.2.3 in main text), and report the accuracy on `the official validation set` due to the unavailability of the ground-truth labels for the `official test data`. In Table 10, the `train, validation` and `test` splits are thus referred to 90% of the official training data, 10% of the official training data, and the entire official validation data respectively. When finetuning the learned architecture (i.e., the searched target network) for each task, we use `the entire official training data` to train and report results on `the official validation set`.

E.2 ImageNet-R Benchmark

The ImageNet-R(etention) (Hendrycks et al., 2021) dataset contains art, cartoons, deviantart, graffiti, embroidery, graphics, origami, paintings, patterns, plastic objects, plush objects, sculptures, sketches, tattoos, toys, and video game renditions of ImageNet classes. It has renditions of 200 ImageNet classes resulting in 30,000 images. The Split ImageNet-R benchmark proposed by (Wang et al., 2022c) uses the ImageNet-R dataset to build a continual learning benchmark specifically for studying methods that use model pretrained on ImageNet (Russakovsky et al., 2015). ImageNet-R poses challenges for such methods because of the diversity within the same class. The Split-ImageNet benchmark proves to be challenging for experience-replay based approaches because of this intra-class variance, as well as replay-free methods that use a frozen backbone from ImageNet as the accuracy of the standard models on ImageNet-R is low. We use the same training and validation splits as those used by (Smith et al., 2023a). For the balanced evaluation, we divide the data set into 10 tasks with 20 classes each, and report results across 3 runs with random class orders. For the imbalanced evaluation, we construct 6 tasks with 5, 10, 15, 20, 50 and 100 classes. We report results across 3 runs with tasks, with varying task orders.

F THE BASE VISION TRANSFORMER: VIT-B/8

We use the base Vision Transformer (ViT) model, with a patch size of 8×8 (ViT-B/8) model from (Dosovitskiy et al., 2021). The base ViT model contains 12 Transformer blocks. A Transformer block is defined by stacking a Multi-Head Self-Attention (MHSA) block and a Multi-Layer Perceptron (MLP) block with resudual connections for each block. ViT-B/8 uses 12 attention heads in each of the MHSA blocks, and a feature dimension of 768. The MLP block expands the dimension size to 3072 in the first layer and projects it back to 768 in the second layer. For all the experiments, we use an image size of 72×72 following the VDD setting. We base the implementation of the ViT on the `timm` package (Wightman, 2019).

Training the Base Model To train the ViT-B/8 model, we use the ImageNet data provided by the VDD benchmark (the `train` split in Table 10). To save the training time, we initialize the weights from the ViT-B/8 trained on the full resolution ImageNet dataset (224×224) and available in the `timm` package, and finetune it for 30 epochs on the downsized version of ImageNet (72×72) in the VDD benchmark. We use a batch size of 2048 split across 4 Nvidia Quadro RTX 8000 GPUs. We follow the standard training/finetuning recipes for ViT models. The file `cheem/artifacts/imagenet_pretraining/args.yaml` in our code folder provides all the training hyperparameters used for training the the ViT-B/8 model on ImageNet. During testing, we take a single center crop of 72×72 from an image scaled with the shortest side to scaled to 72 pixels.

G SETTINGS AND HYPERPARAMETERS IN LEARNING CHEEM

Starting with the ImageNet trained ViT-B/8, the proposed CHEEM learning consists of three components: *supernet training, evolutionary search for target network selection, and target network finetuning*.

Table 11: Data augmentations for the 9 tasks in the VDD benchmark.

Task	Scale and Crop	Hor. Flip	Ver. Flip
CIFAR100	Yes	p=0.5	No
Aircraft	Yes	p=0.5	No
DPed	Yes	p=0.5	No
DTD	Yes	p=0.5	p=0.5
GTSR	Yes	p=0.5	No
OGlt	Yes	No	No
SVHN	Yes	No	No
UCF101	Yes	p=0.5	No
Flwr.	Yes	p=0.5	No

Table 12: Data augmentations used for each task in the 5-Datasets benchmark.

Task	Scale and Crop	Hor. Flip
MNIST	Yes	No
not-MNIST	Yes	No
SVHN	Yes	No
CIFAR100	Yes	p=0.5
Fashion MNIST	Yes	No

Data Augmentations A full list of data augmentations used for the VDD benchmark is provided in Table 11, and the data augmentations used for the tasks in the 5-datasets benchmark is provided in Table 12. The augmentations are chosen so as not to affect the nature of the data. Scale and Crop transformation scales the image randomly between 90% to 100% of the original resolution and takes a random crop with an aspect ratio sampled from a uniform distribution over the original aspect ratio ± 0.05. In evaluating the supernet and the finetuned model on the validation set and test set respectively, images are simply resized to 72×72 with bicubic interpolation.

Supernet Training *The VDD Benchmark*: For each task, we train the supernet for 100 epochs, unless otherwise stated. We use a label smoothing of 0.1. We use a learning rate of 0.001 and the Adam optimier (Kingma & Ba, 2015) with a Cosine Decay Rule. We use a batch size of 512, and ensure the minimum number of batches in an epoch is 15 (via repeatedly sampling when the number of total samples of a task is not sufficient). As stated in the paper, for the Exploration-Exploitation sampling scheme, we use an exploration probability $\epsilon = 0.3$.

The 5-datasets Benchmark: We use the same hyperparameters as those used in the VDD Benchmark, but train the supernet for 50 epochs to account for its relatively lower complexity.

L2G with DARTS and β-DARTS: We train the supernet of the Learn-to-Grow (L2G) (Li et al., 2019) for 50 epochs on the VDD benchmark and 25 epochs on the 5-datasets benchmark, since DARTS simultaneously trains all sub-networks (i.e. the entire supernet) at each epoch. We use a weight of 1 for the beta loss in all the experiments with β-DARTS.

Evolutionary Search The evolutionary search is run for 20 epochs. We use a population size of 50. We use 25 candidates both in the mutation stage and the crossover stage. The top 50 candidates are retained. The crossover is performed among the top 10 candidates, and the top 10 candidates are mutated with a probability of 0.1. For the Exploration-Exploitation sampling scheme, we use an exploration probability $\epsilon = 0.5$ when generating the initial population.

Finetuning The target network for a task selected by the evolutionary search is finetuned for 30 epochs with a learning rate of 0.001, Adam optimizer, and a Cosine Learning Rate scheduler. Drop Path of 0.25 and label smoothing of 0.1 are used for regularization. We use a batch size of 512, and a minimum of 30 batches are drawn.

We use a single Nvidia A100 GPU for all the experiments.

H Modifying SupSup, EFT and LL to Work With ViTs

In the main paper, we compare with Supermasks in Superposition (SupSup) (Wortsman et al., 2020), Efficient Feature Transformation (EFT) (Verma et al., 2021), and Lightweight Learner (LL) (Ge et al., 2023) in Table 5 under the task-to-task transfer learning paradigm. The three methods are originally developed for Convolutional Neural Networks. We modify them to be compatible with ViTs for a fair comparison with our CHEEM.

We use the same ViT-B/8 base model (Sec. F) for SupSup, EFT and LL. For the SupSup method (Wortsman et al., 2020), we learn masks for the weights of the final linear projection layer of the Multi-Head Self-Attention block using the straight through estimator (Bengio et al., 2013). We apply the EFT (Verma et al., 2021) on all the linear layers in the ViT-B/8 (i.e., all the Query/Key/Value projection layers, the final projection layer, and the FFN layers) by scaling their activation maps via the Hadamard product with learnable scaling vectors, following the original proposed formulation for fully-connected layers in the EFT (Verma et al., 2021). For the LL method (Ge et al., 2023) which learns a task-specific bias vector that is added to all the feature maps of convolutional layers, we learn a similar bias vector and add it to the output of all the linear layers of the ViT.