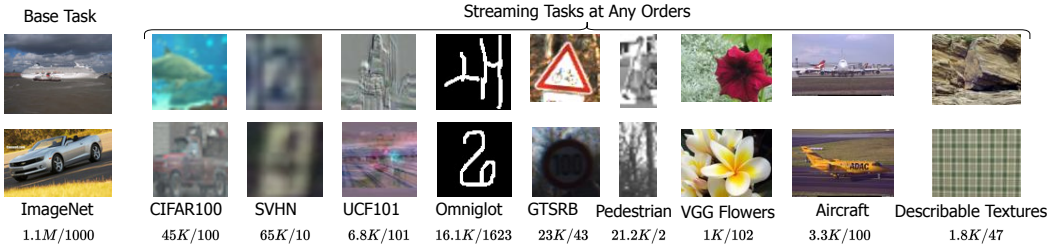
Supplementary: CHEEM: Continual Learning by Reuse, New, Adapt and Skip – A Hierarchical Exploration-Exploitation Approach

Chinmay Savadikar
North Carolina State University
csavadi@ncsu.edu

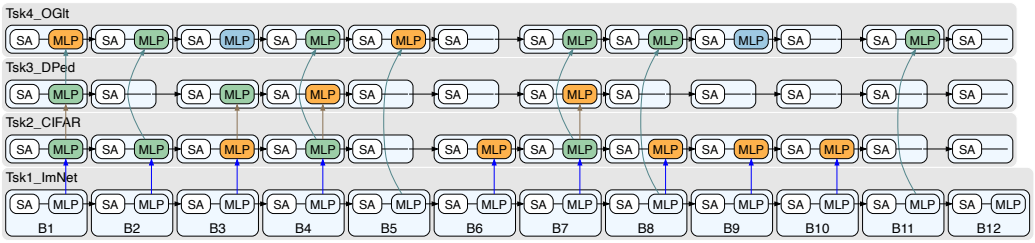
Michelle Dai
Johns Hopkins University
mdai12@jh.edu

Tianfu Wu
North Carolina State University
tianfu_wu@ncsu.edu

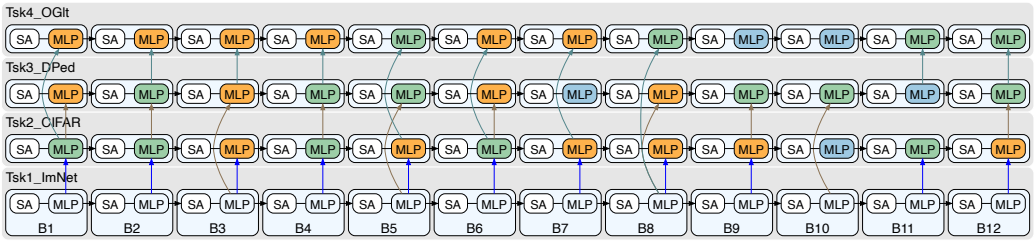
A. Examples of CHEEM learned continually on the VDD benchmark



(a) **The VDD benchmark** [21] consisting of tasks of different nature with `#training images/#classes` significantly varying across different tasks.



(b) From **ViT-Base trained on Tsk1_ImNet** (with blocks B1 to B12), our CHEEM learns sensible task-tailored models that reflect task complexity. For example, when learning Daimer Pedestrian Classification (Tsk3_DPed), CHEEM learns to `Skip` 8 MLP blocks and Reuse most of the architecture. When learning Omniglot (Tsk3_Oglt), which has a larger shift from ImageNet, CHEEM learns to Adapt the ImageNet parameters in Blocks 1 and 5, adds New operations in Blocks 3 and 9, and `Skips` blocks 6, 10 and 12.



(c) From **DEiT-Tiny trained on Tsk1_ImNet** (with blocks B1 to B12), our CHEEM learns to use multiple Adapt and New operations, without `Skip` operations selected, sensibly different from those with more `Skip` and less New operations learned based on the stronger ViT-Base model.

Figure 6. Examples of CHEEM learning task-tailored models.

B. Effects of streaming task orders

We verify the effect of different task orders on the performance of CHEEM. Table 11 shows that CHEEM is robust to task orders on the MTIL benchmark.

Table 11. Results of learning CHEEM on the MTIL benchmark with three different streaming task orders.

SUN	**Airc**	**DTD**	**F101**	**Cars**	**C101**	**CIFAR**	**ESAT**	**Flwr**	**MNIST**	**Pets**	**Avg. Acc.**	**Avg. Frgt.**
68.59	67.87	69.15	89.02	83.60	84.41	90.47	98.56	97.82	99.65	93.05	85.65	1.38
C101	**CIFAR**	**ESAT**	**Flwr**	**MNIST**	**Pets**	**DTD**	**Cars**	**F101**	**Airc**	**SUN**	**Avg. Acc.**	**Avg. Frgt.**
78.23	90.36	98.42	97.76	99.66	91.77	69.15	84.48	89.30	66.13	66.86	84.74	2.47
MNIST	**SUN**	**Flwr**	**DTD**	**C101**	**Cars**	**Pets**	**F101**	**CIFAR**	**Airc**	**ESAT**	**Avg. Acc.**	**Avg. Frgt.**
99.63	68.58	98.11	67.87	84.52	84.39	92.53	88.50	90.88	69.10	97.78	85.63	1.28

C. Full Learned CHEEM on MTIL

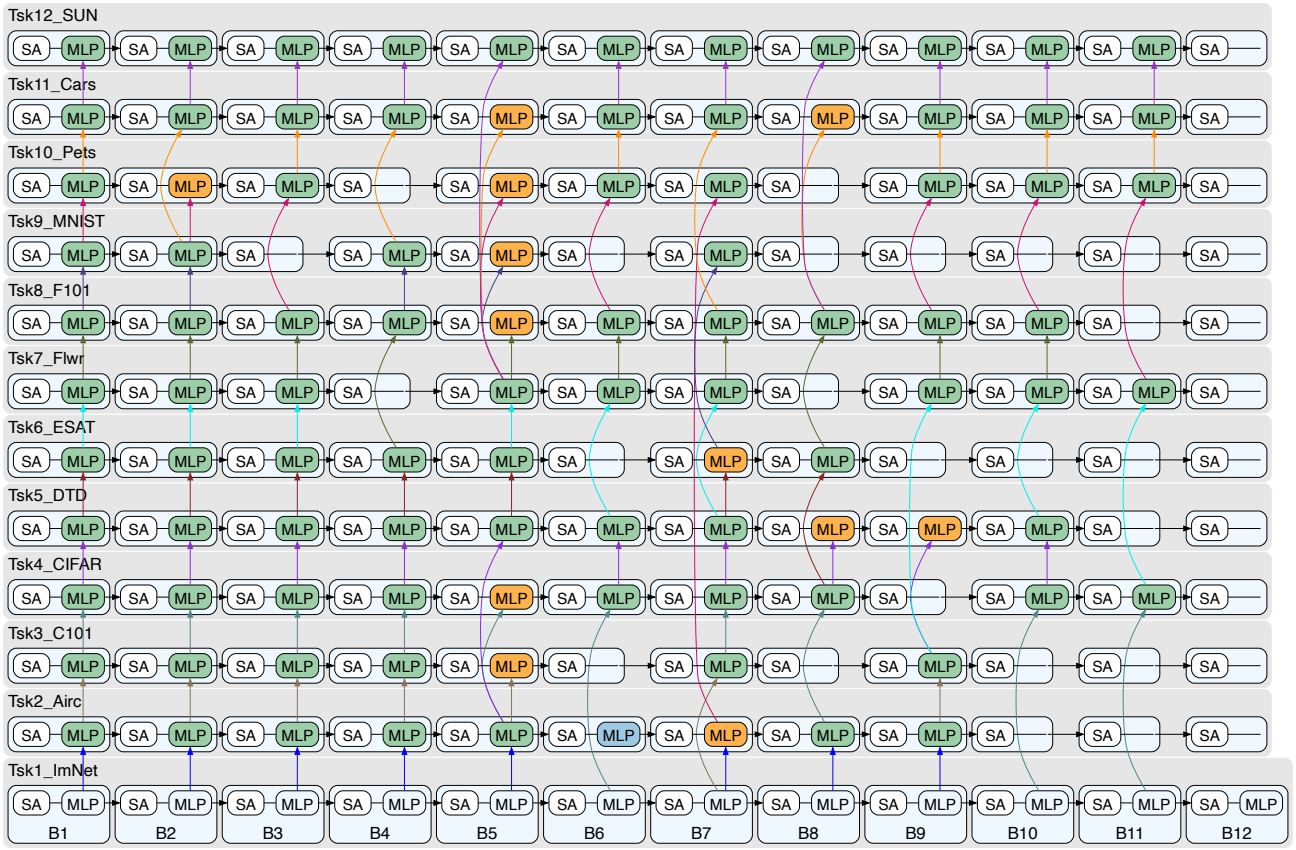


Figure 7. Figure 3b from the main text reproduced on the full benchmark. From **ViT-Base trained on Tsk1_ImNet** (with blocks B1 to B12), our CHEEM learns sensible task-tailored models that reflect task complexity. For example, when learning Caltech 101 (Tsk3_C101), CHEEM learns to `Skip` 5 MLP blocks and Reuse most of the architecture. In contrast, when learning FGVC Aircraft (Tsk1_Airc), which is a more complex task with larger distribution shift from ImageNet due to its fine-grained nature, CHEEM learns to Adapt the ImageNet parameters in Block 7, adds a New operation in Block 6, and `Skips` the last 3 MLP blocks. When learning MNIST, CHEEM `skips` 8 MLP blocks, accounting for the easy nature of the task.

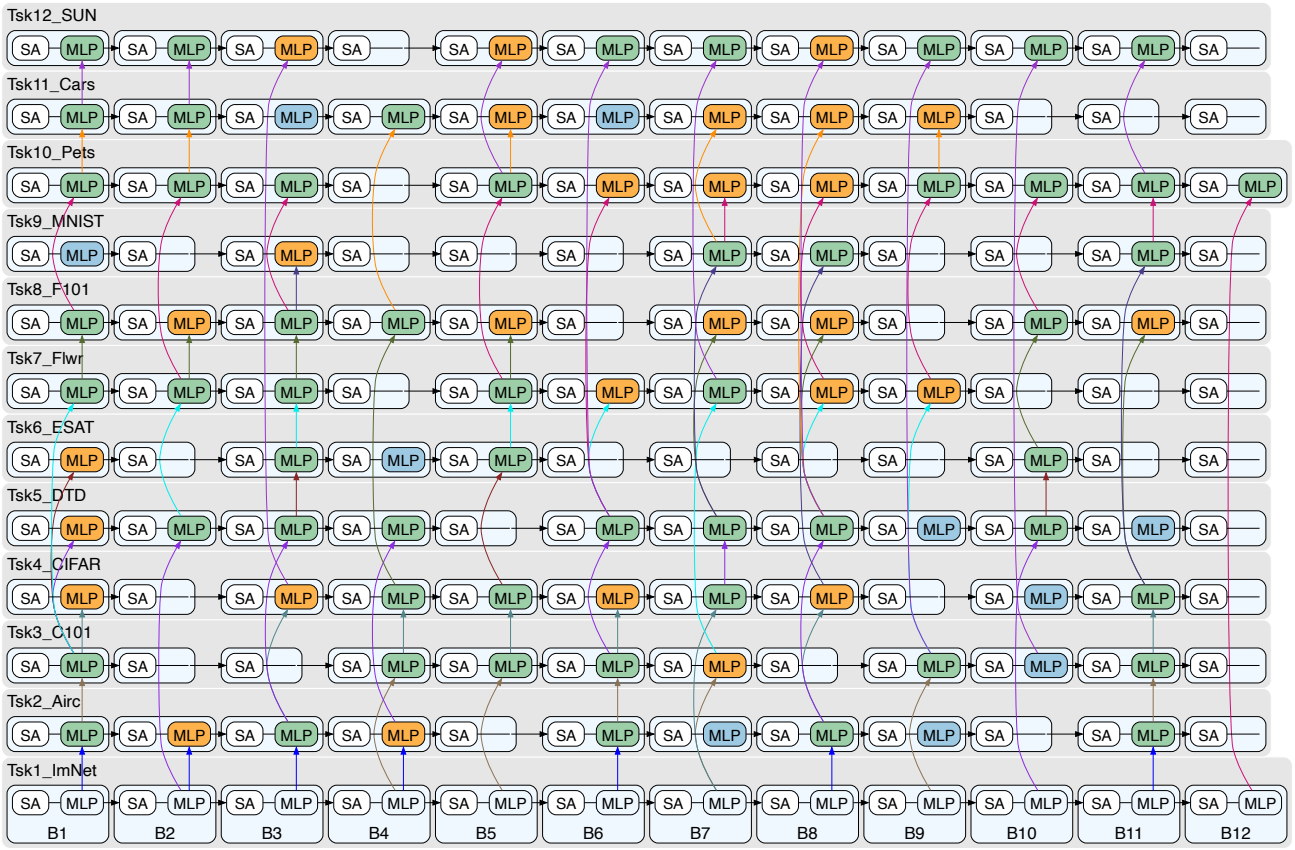


Figure 8. **ViT-Base trained on Tsk1_ImNet** (with blocks B1 to B12), with **Pure Exploration in CHEEM**. While pure exploration accounts for task complexity through the `skip` operation, it also adds more many more Adapt and New operations as compared to the proposed Hierarchical Exploration-Exploitation scheme (Figure 7). This shows that the HEE sampling scheme can effectively leverage task synergies and reuse previous parameter memories.

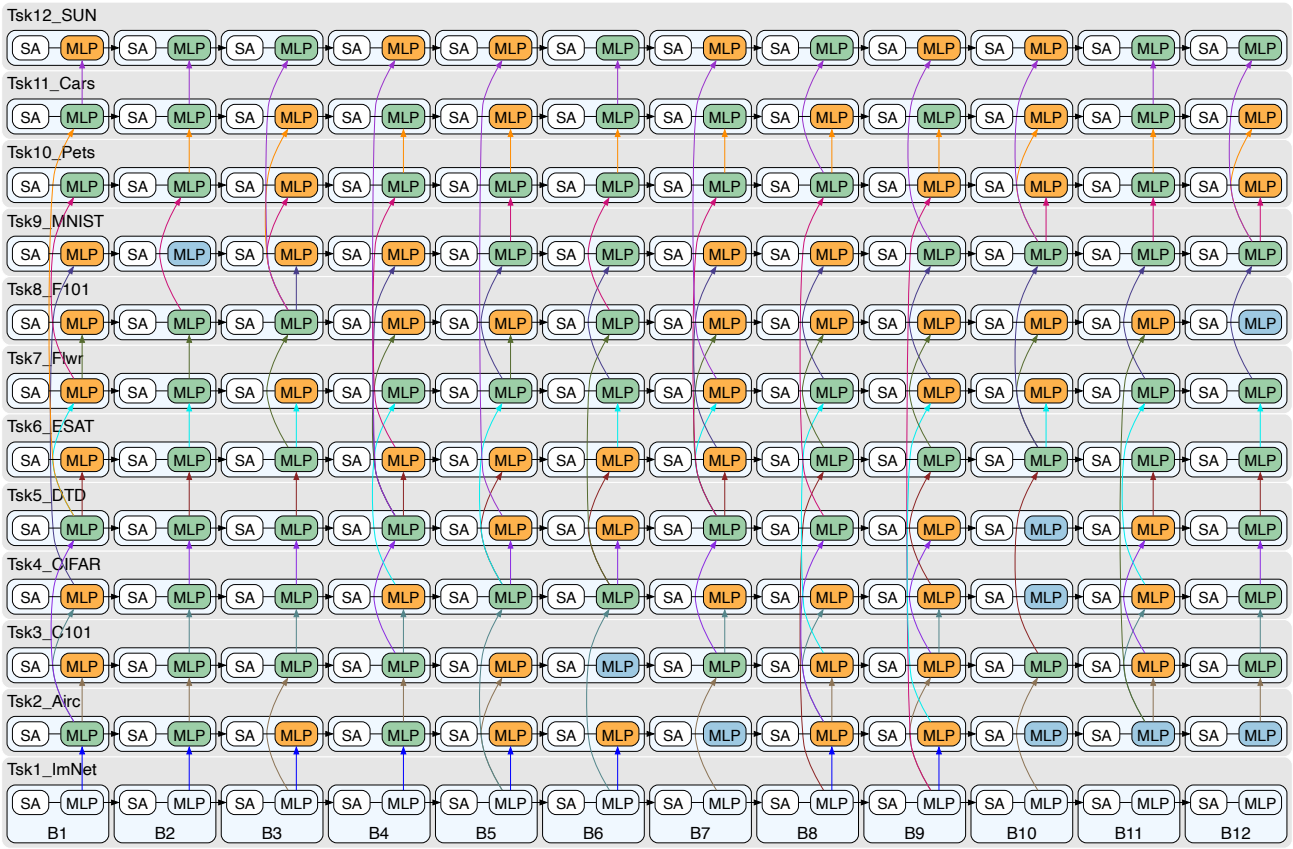


Figure 9. Figure 3c reproduced on the full benchmark. From **DEiT-Tiny trained on Tsk1_ImNet** (with blocks B1 to B12), our CHEEM learns to use multiple Adapt and New operations, without `Skip` operations selected, sensibly different from those with more `Skip` and less New operations learned based on the stronger ViT-Base model.

D. Full Results

Table 12. **MTIL**: Full results on the MTIL benchmark, extending Tables 2 and 4 in the main text.

Method	Airc	C101	CIFAR	DTD	ESAT	Flwr	F101	MNIST	Pets	Cars	SUN	Avg. Acc	Avg. Frgt.
ViT-Base													
Full Finetuning	69.87	98.32	90.66	77.46	98.78	97.87	88.46	99.70	92.85	85.42	69.89	88.12 ± 0.04	-
LoRA Finetuning	63.86	97.77	91.35	77.59	98.84	98.83	88.41	99.69	93.14	80.26	71.96	87.43 ± 0.01	-
CHEEM (MLP^{Down}, HEE)	69.77	84.86	90.27	68.48	98.31	97.54	89.48	99.60	92.88	84.94	68.58	85.88 ± 0.29	1.73 ± 0.05
CHEEM (MLP^{Down}, PE)	69.97	84.96	90.21	66.74	97.97	97.32	86.97	99.50	92.32	82.11	64.06	84.74 ± 0.26	1.72 ± 0.05
CHEEM (Attn Proj, HEE)	69.92	83.00	90.44	66.54	98.31	97.53	88.94	99.60	92.90	85.50	68.62	85.57 ± 0.27	1.67 ± 0.03
EWC	39.10	40.90	43.93	12.98	61.43	22.24	51.81	96.20	60.65	12.64	48.46	44.58 ± 6.35	23.80 ± 6.53
CODA-Prompt	0.91	19.14	75.60	7.39	38.26	24.40	84.62	97.32	36.02	12.61	46.17	40.22 ± 1.22	25.25 ± 1.78
DualPrompt	3.08	14.40	83.96	3.48	46.45	6.00	85.46	68.43	24.13	5.88	30.78	33.82 ± 0.35	22.11 ± 0.42
L2P	1.22	17.35	78.81	3.46	30.39	4.67	78.47	16.83	23.45	4.62	33.40	26.61 ± 0.16	30.96 ± 0.27
S-Prompts	53.78	82.54	88.26	65.44	96.71	98.51	84.64	99.23	92.88	70.09	65.79	81.62 ± 0.35	1.64 ± 0.05
DIKI	52.29	91.68	89.10	63.95	96.31	30.22	86.55	98.37	92.24	70.22	69.74	76.42 ± 0.04	1.96 ± 0.02
LoRA (MLP^{Down})	63.78	85.68	90.52	67.98	98.41	98.51	87.26	99.69	92.51	79.79	67.59	84.70 ± 0.01	1.64 ± 0.11
DEiT Tiny													
Method	**Airc**	**C101**	**CIFAR**	**DTD**	**ESAT**	**Flwr**	**F101**	**MNIST**	**Pets**	**Cars**	**SUN**	**Avg. Acc**	**Avg. Frgt.**
Full Finetuning	43.17	94.64	83.55	64.88	98.67	68.90	79.88	99.65	86.57	54.83	52.99	75.25 ± 0.12	-
LoRA Finetuning	39.92	93.71	81.04	63.37	98.59	74.38	76.25	99.58	87.38	53.80	52.97	74.64 ± 0.08	-
CHEEM (MLP^{Down}, HEE)	52.51	80.59	79.67	57.43	97.86	73.94	77.89	99.60	87.37	61.73	51.02	74.51 ± 0.28	1.86 ± 0.04
CHEEM (MLP^{Down}, PE)	53.03	80.50	80.16	57.66	97.86	80.37	78.11	99.62	85.95	62.43	49.81	75.05 ± 0.12	1.85 ± 0.06
CHEEM (Attn Proj, HEE)	50.65	80.35	78.44	56.77	97.75	77.23	77.71	99.55	86.84	61.72	51.27	74.39 ± 0.13	1.95 ± 0.03
EWC	37.38	13.94	48.87	0.00	83.14	0.00	50.65	93.72	30.44	2.89	27.57	35.33 ± 0.32	7.34 ± 0.55
CODA-Prompt	0.00	1.77	2.75	0.04	0.32	0.00	22.46	3.94	5.80	0.27	24.45	5.62 ± 0.25	42.58 ± 0.81
DualPrompt	0.66	42.28	59.13	3.03	42.04	0.86	42.10	55.06	47.42	5.75	41.47	30.89 ± 0.29	17.53 ± 0.27
L2P	0.11	39.46	47.87	4.11	29.80	1.02	37.07	0.83	50.15	1.29	43.97	23.24 ± 0.14	25.81 ± 0.37
S-Prompts	36.00	79.08	71.58	50.50	93.87	72.27	67.97	98.66	87.44	40.01	43.22	67.33 ± 0.38	1.80 ± 0.02
DIKI	33.95	76.57	71.13	54.84	92.66	71.79	70.46	97.40	87.61	40.13	47.41	67.63 ± 0.06	1.76 ± 0.01
LoRA (MLP^{Down})	39.48	78.89	78.11	54.38	97.80	73.66	74.80	99.58	85.88	53.53	45.61	71.06 ± 0.02	1.87 ± 0.00

Table 13. **VDD**: Full results on VDD benchmark, extending Table 3 and 5 in the main text.

Method	CIFAR	DPed	OGlt	SVHN	UCF	GTSR	Flwr	Airc	DTD	Avg. Acc	Avg. Frgt.
ViT-Base											
Full Finetuning	90.65	99.97	86.06	97.75	79.54	99.35	98.03	70.29	76.99	88.74 ± 0.11	-
LoRA Finetuning	91.44	99.50	79.43	97.42	73.36	98.95	98.96	64.03	77.64	86.75 ± 0.11	-
CHEEM (MLP^{Down}, HEE)	90.06	99.59	83.32	95.87	73.96	97.09	97.48	67.13	75.85	86.71 ± 0.23	0.35 ± 0.02
CHEEM (Attn Proj, HEE)	89.90	99.58	83.08	96.26	74.49	97.27	97.56	70.55	76.42	87.23 ± 0.22	0.34 ± 0.01
EWC	83.69	97.69	6.91	77.43	25.92	78.20	0.06	5.98	19.91	43.98 ± 1.34	5.09 ± 1.14
CODA-Prompt	37.69	1.29	6.87	54.52	2.32	49.22	39.18	7.48	25.16	24.86 ± 2.19	26.11 ± 0.75
DualPrompt	82.34	4.04	14.37	15.02	13.41	64.42	27.20	15.29	16.37	28.05 ± 0.85	3.18 ± 0.51
L2P	86.64	4.98	14.75	6.63	14.19	27.89	25.59	16.71	18.12	23.94 ± 0.72	8.98 ± 0.64
S-Prompts	88.34	99.47	57.38	94.23	55.07	87.90	98.48	53.52	72.59	78.55 ± 0.09	0.36 ± 0.04
DIKI	86.54	98.20	57.70	63.44	52.10	72.66	36.45	53.53	72.82	65.94 ± 0.05	0.11 ± 0.01
LoRA (MLP^{Down})	90.18	99.21	79.43	96.35	73.10	97.39	98.54	64.01	76.19	86.04 ± 0.11	0.34 ± 0.03
DEiT Tiny											
Method	**CIFAR**	**DPed**	**OGlt**	**SVHN**	**UCF**	**GTSR**	**Flwr**	**Airc**	**DTD**	**Avg. Acc**	**Avg. Frgt.**
Full Finetuning	83.50	99.97	69.71	97.24	57.97	98.95	69.04	44.46	65.02	76.21 ± 0.07	-
LoRA Finetuning	81.29	99.96	76.93	96.37	54.83	98.16	74.37	40.66	63.67	76.25 ± 0.30	-
CHEEM (MLP^{Down}, HEE)	75.75	97.73	81.64	95.30	57.26	93.11	74.76	45.91	64.13	76.18 ± 0.10	1.03 ± 0.01
CHEEM (Attn Proj, HEE)	74.70	97.85	80.43	95.22	57.46	93.68	75.75	46.55	62.11	75.97 ± 0.36	1.09 ± 0.01
EWC	79.39	93.96	0.03	60.13	4.97	64.41	0.00	0.58	0.00	33.72 ± 0.15	1.52 ± 0.08
CODA-Prompt	2.07	0.00	0.02	1.55	0.02	0.56	0.36	0.30	5.16	1.12 ± 0.08	37.56 ± 0.40
DualPrompt	47.87	4.48	28.60	11.53	2.54	75.67	0.40	0.57	2.61	19.36 ± 0.55	10.54 ± 0.49
L2P	56.24	1.38	0.80	0.26	2.15	37.43	1.24	0.17	3.90	11.51 ± 0.76	20.90 ± 1.72
S-Prompts	68.58	97.24	46.05	85.87	43.44	80.13	74.78	36.72	58.90	65.75 ± 0.27	0.90 ± 0.02
DIKI	65.54	97.44	44.89	45.55	40.78	64.49	72.37	34.41	59.38	58.32 ± 0.05	0.62 ± 0.00
LoRA (MLP^{Down})	74.26	97.69	76.87	94.96	52.68	93.09	73.75	40.52	62.22	74.01 ± 0.34	1.07 ± 0.02

E. Effect of Exploration Probability (ϵ_1, ϵ_2) and Tolerance Threshold (τ)

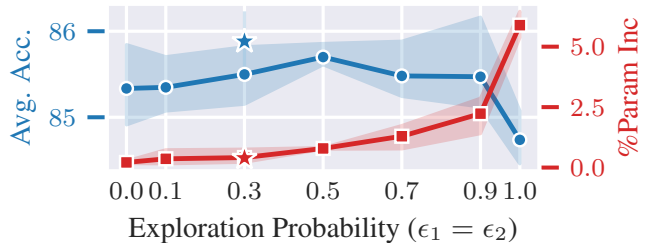


Figure 10a: Effect of the exploration probability on the MTIL benchmark, with exploration probabilities ϵ_1 (supernet training) and ϵ_2 (evolutionary search) set equal. As ϵ increases, average accuracy first rises, then falls, while the average number of additional parameters per task increases monotonically. This is due to more new operations being learned; $\epsilon = 0.3$ strikes a good balance. **Setting $\epsilon < 0.5$ controls the addition of new operations while maintaining performance**. $\epsilon_1 = 0.3$ and $\epsilon_2 = 0.5$ used in our experiments (denoted by ★) improve accuracy further without increasing parameters. In sum, ϵ governs the number of `reuse` (exploitation), `adapt`, and `new` (exploration) operations.

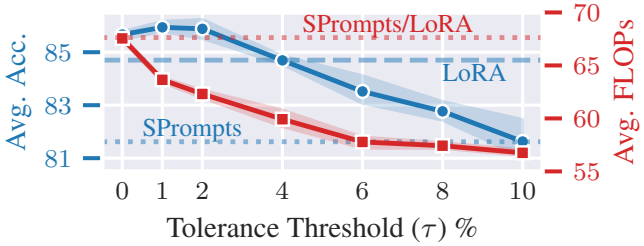


Figure 10b: A higher Tolerance Threshold reduces **average FLOPs per task** but also lowers **average accuracy**, as it permits more `skip` operations to persist in the population during evolutionary search, even if their accuracy is lower (within the tolerance margin). A 2% threshold, used in our experiments, offers a good trade-off. At $\tau = 6\%$, CHEEM still surpasses SPrompts in average accuracy (dotted blue line) while using significantly fewer FLOPs, beyond which the FLOPs plateau. SPrompt FLOPs (dotted red line) closely match those of LoRA, so the same line is used. At $\tau = 4\%$, CHEEM matches LoRA's average accuracy (dashed blue line) with substantially fewer FLOPs. Thus, with $\tau \leq 4\%$, CHEEM matches or exceeds LoRA in accuracy while reducing FLOPs.

F. Generalization to non-ImageNet backbones

Table 14 shows that CHEEM is effective on models pretrained on datasets and objectives beyond ImageNet.

Table 14. Results on the **MTIL** benchmark using **CLIP** ViT-B/16 [20].

Method	Avg. Acc. ↑	Frgt ↓	FLOPs ↓
CODA-P	27.6 ± 1.4	42.1 ± 1.4	70.3
L2P	37.2 ± 0.1	23.9 ± 0.6	70.3
DualPrompt	44.6 ± 0.4	15.0 ± 0.4	70.3
S-Prompts	83.1 ± 0.1	1.2 ± 0.0	67.6
DIKI	77.9 ± 0.1	**1.0** ± 0.0	**42.5**
LoRA-CL	**86.8** ± 0.3	1.2 ± 0.0	68.2
Tuna	66.7 ± 0.3	22.7 ± 0.4	405.0
T2T-Finetune	86.3 ± 0.1	-	-
Our CHEEM	**86.4** ± 0.2	1.2 ± 0.0	64.3

Table 15. Results on the **VDD** benchmark using **CLIP** ViT-B/16.

Method	Avg. Acc. ↑	Frgt ↓	FLOPs ↓
CODA-P	16.3 ± 0.7	34.4 ± 0.4	70.3
L2P	23.0 ± 1.1	21.6 ± 1.2	70.3
DualPrompt	29.8 ± 1.4	16.8 ± 1.8	70.3
S-Prompts	78.4 ± 0.1	0.4 ± 0.0	67.6
DIKI	69.4 ± 0.1	2.4 ± 0.0	**42.5**
LoRA-CL	85.8 ± 0.0	0.4 ± 0.0	68.2
Tuna	71.8 ± 0.2	10.4 ± 0.8	337.0
T2T-Finetune	86.6 ± 2.3	-	-
Our CHEEM	84.5 ± 1.3	0.4 ± 0.0	63.0

G. Smaller model for Task ID Recognition

Tables 16 and 17 present the results of using a smaller frozen backbone for task ID recognition alongside a larger backbone for final class prediction. For a fair comparison, we also include MoEAdapter4CL [27] as a baseline. MoEAdapters4CL uses a pretrained AlexNet backbone for task identification by training an autoencoder per task in the feature space, and infers the task ID at test time using the reconstruction error.

The results show that CHEEM can effectively decouple task identification and classification by using a lighter model (AlexNet or DEiT-Tiny) for task ID prediction and a stronger model (ViT-Base) for final classification. While using AlexNet for task identification leads to a noticeable drop in accuracy, replacing it with DEiT-Tiny results in only a negligible performance degradation. In fact, combining DEiT-Tiny for task prediction with ViT-Base for classification achieves performance comparable to using ViT-Base for both tasks, while significantly reducing computational cost (FLOPs).

Furthermore, CHEEM consistently outperforms MoEAdapters4CL when using either AlexNet or DEiT-Tiny for task identification. This demonstrates that CHEEM successfully leverages lightweight models for efficient task recognition without sacrificing overall performance, effectively combining efficiency and accuracy.

Table 16. Comparison of Average Accuracy and Forgetting **on the MTIL benchmark** with three seeds.

Method	Task ID	Avg. Acc	Avg. Frgt.	FLOPs
ViT-Base				
MoEAdapter4CL	AlexNet	71.5 ± 3.3	7.4 ± 1.2	35.4
CHEEM	ViT-Base	**85.9** ± 0.3	1.7 ± 0.1	62.3
$\text{CHEEM}_{\text{lite}}$	AlexNet	79.9 ± 0.2	4.6 ± 0.2	34.1
$\text{CHEEM}_{\text{lite}}$	DEiT-Tiny	84.2 ± 0.5	2.1 ± 0.1	35.0
DEiT-Tiny				
MoEAdapter4CL	AlexNet	53.7 ± 1.6	9.1 ± 1.3	3.7
CHEEM	DEiT-Tiny	**74.5** ± 0.3	1.9 ± 0.0	4.5
$\text{CHEEM}_{\text{lite}}$	AlexNet	70.8 ± 0.6	4.1 ± 0.2	3.5

Table 17. Comparison of Average Accuracy and Forgetting **on the VDD benchmark** with three seeds.

Method	Task ID	Avg. Acc	Avg. Frgt.	FLOPs
ViT-Base				
MoEAdapter4CL	AlexNet	70.2 ± 4.6	0.5 ± 0.1	35.4
CHEEM	ViT-Base	**86.7** ± 0.2	0.4 ± 0.0	61.6
$\text{CHEEM}_{\text{lite}}$	AlexNet	83.8 ± 0.1	2.0 ± 0.1	28.8
$\text{CHEEM}_{\text{lite}}$	DEiT-Tiny	85.6 ± 0.3	1.1 ± 0.0	29.3
DEiT-Tiny				
MoEAdapter4CL	AlexNet	65.6 ± 0.5	1.6 ± 0.1	3.7
CHEEM	DEiT-Tiny	**76.18** ± 0.1	1.0 ± 0.0	4.5
$\text{CHEEM}_{\text{lite}}$	AlexNet	76.9 ± 0.2	1.0 ± 0.1	3.1

H. Experiment Details

Pretrained Models: We initialize the pretrained ViT-B/16 and DEiT-Tiny/16 models from the checkpoint available in `timm`. Both models use a patch size of 16 and a resolution of 224×224. The ViT-B/16 checkpoint has been pretrained on ImageNet 21k and finetuned on ImageNet1k. The DEiT-Tiny/16 checkpoint has been trained on ImageNet1k. All our experiments use the same checkpoints. We refer readers to [6] for the architecture details of ViT-B/16 and [25] for the architecture details of DEiT-Tiny/16.

Our experiments are conducted using PyTorch and leverage `timm` for architecture implementation. In all our experiments, we use the Adam optimizer [10] with no weight decay. For experiments with CHEEM, we use a learning rate of 0.001, 50 epochs for the supernet training and 20 epochs for finetuning. During supernet training, we use an exploration probability of $\epsilon = 0.3$, and use $\epsilon = 0.5$ during the target network selection to encourage more exploration. We do not perform any data augmentations, and simply resize the images to 224×224. We adapt the implementation from `https://github.com/GT-RIPL/CODA-Prompt` to perform experiments on CODA-Prompt, DualPrompts and L2P, and use our own implementations for the other baseline methods. We use a single Nvidia A100 GPU for all our experiments.

H.1. Details of the MTIL benchmark

The MTIL benchmark [28] consists of 11 tasks: FGVC-Aircraft [15], Caltech101 [14], CIFAR100 [11], Describable Textures [5], EuroSAT [8], VGG-Flowers [18], Food101 [3], MNIST [13], Oxford Pets [19], Stanford Cars [7], SUN397 [26]. We use the official training and testing splits provided in the constituent datasets. We use the official validation splits for the evolutionary search, and create our own splits when official split is not provided by randomly sampling 10% of the training dataset.

Table 18. Number of samples in the training, validation, and test sets used in the experiments on the MTIL benchmark, along with the number of categories.

Task	#Train	#Validation	#Test	#Classes
FGVC Aircraft	3334	3333	3333	100
Caltech101	5465	608	2604	101
CIFAR100	45000	5000	19850	100
Describable Textures	1880	1880	1880	47
EuroSAT	17010	1890	8100	10
VGG-Flowers	1020	1020	6149	102
Food-101	68175	7575	25250	101
MNIST	54000	6000	10000	10
Oxford Pets	3312	368	3669	37
Stanford Cars	7329	815	8041	196
SUN397	17865	1985	19850	397

H.2. Details of the VDD benchmark

The VDD benchmark [21] consists of 10 tasks: ImageNet-1k [22], CIFAR100 [11], SVHN [17], UCF101 Dynamic Images (UCF) [2, 23], Omniglot [12], German Traffic Signs (GTSR) [24], Daimler Pedestrian Classification (DPed) [16], VGG Flowers [18], FGVC-Aircraft [15], and Describable Textures (DTD) [5]. All the images in the VDD benchmark have been scaled such that the shorter side is 72 pixels. However, for a more realistic evaluation, we reconstruct the VDD benchmark with the original images and splits. Except for UCF101, Omniglot, and Daimler Pedestrian Classification, we use the official train, validation and test splits (when a validation split is not avaiable, we construct a validation split by randomly sampling 10% of the training data.). Due to a lack of high resolution images for UFC101, Omniglot, and Daimler Pedestrian Classification, we use the splits and the images provided by the VDD benchmark and resize the images to 224×224.

Table 19. Number of samples in the training, validation, and test sets used in the the experiments on the VDD benchmark, along with the number of categories.

Task	#Train	#Validation	#Test	#Classes
ImageNet12	1108951	123216	49000	1000
CIFAR100	45000	5000	19850	100
SVHN	65931	7326	26032	10
UCF	6827	758	1952	101
Omniglot	16068	1785	6492	1623
GTSR	23976	2664	12630	43
DPed	21168	2352	5880	2
VGG-Flowers	1020	1020	6149	102
FGVC Aircraft	3334	3333	3333	100
Describable Textures	1880	1880	1880	47

I. Theoretical Analysis of Local vs. Global Argmax of Head Classifiers in Continual Learning

As seen in Section 4.4, CODA-Prompt, DualPrompt and L2P perform significantly worse than LoRA-C and CHEEM. This large drop is attributed to the discrepancy between local and global softmax. We verify this in Table 20, which shows that when provided with a task ID to retrieve the appropriate local part of the head during inference, the performance of CODA-Prompt, DualPrompt and L2P is significantly better, almost approaching S-Prompts and DIKI. We provide a theoretical analysis in the following sections.

I.1. The problem

In continual learning, we have N tasks, each with a different number of classes. Let task t have C_t classes, so by time T we have observed tasks $1, \ldots, T$ with a total of $\sum_{t=1}^{T} C_t$

Table 20. Acc_{Global} refers to the average accuracy (Eqn. 4) calculated using the global head, and Acc_{Local} refers to the same but by masking the logits not belonging to the task. Acc_{Train} refers to the accuracy calculated after the training on a task is complete, averaged over all the tasks.

	ViT-B			DEiT-Tiny		
Method	$\textbf{Acc}_{Global}$	$\textbf{Acc}_{Local}$	$\textbf{Acc}_{Train}$	$\textbf{Acc}_{Global}$	$\textbf{Acc}_{Local}$	$\textbf{Acc}_{Train}$
CODA-Prompt	40.22 ± 1.22	79.70 ± 0.61	86.18 ± 0.02	5.62 ± 0.25	34.72 ± 1.62	67.53 ± 0.37
DualPrompt	33.82 ± 0.35	83.61 ± 0.13	84.63 ± 0.09	30.89 ± 0.29	68.17 ± 0.24	71.25 ± 0.10
L2P	26.61 ± 0.16	80.03 ± 0.58	84.95 ± 0.11	23.24 ± 0.14	60.79 ± 0.67	71.47 ± 0.08
S-Prompts	81.62 ± 0.35	84.48 ± 0.18	84.48 ± 0.18	67.33 ± 0.38	70.71 ± 0.40	70.71 ± 0.40
DIKI	76.42 ± 0.04	84.50 ± 0.04	84.50 ± 0.04	67.63 ± 0.06	70.86 ± 0.07	70.86 ± 0.07
CHEEM	85.88 ± 0.29	88.68 ± 0.16	88.68 ± 0.16	74.51 ± 0.28	78.11 ± 0.31	78.11 ± 0.31

classes. We train a shared feature extractor $\phi(\mathbf{x}) \in \mathbb{R}^d$ and a growing head classifier composed of task-specific segments $W^t \in \mathbb{R}^{d \times C_t}$.

During training of task t, only the segment W^t is updated and used in a softmax over the C_t classes for the current task. However, at inference, for a new test sample $\mathbf{x}$ belonging (in truth) to task t^*, the entire head is used: we compute logits for *all* classes seen so far, and choose the global $\arg\max$. We denote:

- *Local argmax*:

$$\hat{y}_{\text{local}}(\mathbf{x}) \;=\; \arg \max_{c \in \{1,\dots,C_{t^*}\}} \; z_{t^*,c}(\mathbf{x}),$$

  where $z_{t^*,c}(\mathbf{x}) = \langle W^t_{(\cdot,c)}, \phi(\mathbf{x}) \rangle$ are the logits restricted to task t^*.
- *Global argmax*:

$$\hat{y}_{\text{global}}(\mathbf{x}) \;=\; \arg \max_{(t,c) \in \{1,\dots,T\} \times \{1,\dots,C_t\}} \; z_{t,c}(\mathbf{x}).$$

We are interested in the probability that these two predictions coincide:

$$\Pr\big(\hat{y}_{\text{local}}(\mathbf{x}) \;=\; \hat{y}_{\text{global}}(\mathbf{x})\big).$$

Below is a stylized theoretical analysis of why and how often these two can match, highlighting the factors that influence this probability.

I.2. Distribution of Logits and Task Separation

Let $z_{t,c}(\mathbf{x})$ be the logit for class c in task t for sample $\mathbf{x}$. We may approximate $z_{t,c}(\mathbf{x})$ by a random variable with mean $\mu_{t,c}$ and variance $\sigma^2_{t,c}$, e.g.,

$$z_{t,c}(\mathbf{x}) \;\approx\; \mu_{t,c} + \epsilon_{t,c}, \quad \epsilon_{t,c} \sim \mathcal{N}(0, \sigma^2_{t,c}).$$

In reality, these means and variances depend on how well the feature $\phi(\mathbf{x})$ and the weights W^t are aligned, but we treat them as parameters to illustrate.

Define:

$$\max_{c \in C_{t^*}} z_{t^*,c}(\mathbf{x}) \;\text{ (the local max for the correct task)}, \tag{1}$$

$$\max_{(t \neq t^*)} \max_{c \in C_t} z_{t,c}(\mathbf{x}) \;\text{ (the max out-of-task logit)}. \tag{2}$$

For $\hat{y}_{\text{local}} = \hat{y}_{\text{global}}$, we need

$$\max_{c \in C_{t^*}} z_{t^*,c}(\mathbf{x}) \;\;\geq\;\; \max_{(t \neq t^*)} \max_{c} z_{t,c}(\mathbf{x}).$$

Hence the distribution of all out-of-task logits relative to the best in-task logit is crucial.

I.3. Probability of Matching Local and Global Argmax

I.3.1. A Basic Two-Class Example

Consider just one class c^* in the true task vs. one class k in an *other* task. Suppose

$$z_{t^*,c^*} \sim \mathcal{N}(\mu^*, \sigma^2), \quad z_{t',k} \sim \mathcal{N}(\mu', \sigma^2).$$

The probability that $z_{t^*,c^*} \geq z_{t',k}$ is

$$\Pr(z_{t^*,c^*} \geq z_{t',k}) = \Pr(z_{t^*,c^*} - z_{t',k} \geq 0) = \Phi\Big(\frac{\mu^* - \mu'}{\sqrt{2}\,\sigma}\Big),$$

where Φ is the standard normal CDF.

I.3.2. Many Classes from Different Tasks

Now suppose there are C_{t^*} classes in the correct task, and $M = \sum_{t \neq t^*} C_t$ classes outside. Let the local maximum

$$Z^* \;=\; \max_{c \in \{1,\dots,C_{t^*}\}} \; z_{t^*,c},$$

and let $Z_1, \dots, Z_M$ represent the logits of the M out-of-task classes. Then

$$\Pr(\hat{y}_{\text{local}} = \hat{y}_{\text{global}}) \;=\; \Pr\Big(Z^* \;\geq\; \max\{Z_1, \dots, Z_M\}\Big).$$

If Z^* is (roughly) $\mathcal{N}(\mu_{\text{local}}, \sigma^2_{\text{local}})$ and each Z_j is $\mathcal{N}(\mu_o, \sigma^2_o)$ (independent simplification), then

$$\Pr\big(Z^* \geq Z_j \text{ for all } j\big) \;=\; \int \Big[\Pr(Z_j \leq z)\Big]^M \, F_{Z^*}(z)\, dz.$$

When $\mu_{\text{local}} > \mu_o$, this probability is high for moderate M, but as M grows, the chance that *some* out-of-task class logit exceeds Z^* increases, unless the gap $\mu_{\text{local}} - \mu_o$ is large.

I.4. Factors Influencing the Match Probability

1. **Feature Separation Across Tasks.** If $\phi(\mathbf{x})$ strongly separates tasks, then for $\mathbf{x}$ from task t^*, out-of-task logits $z_{t,c}$ for $t \neq t^*$ are consistently lower. This increases the probability of $\hat{y}_{\text{local}} = \hat{y}_{\text{global}}$.
2. **Logit Magnitude & Variance.** Even if the *means* of the correct task's logits exceed those of other tasks, high variance or overlap can cause out-of-task classes to occasionally exceed the correct task's maximum.
3. **Regularization and Task Order.** Continual-learning methods that regularize old task weights or use replay data reduce the chance of weight drift, making it less likely that earlier or other tasks overshadow the correct one.
4. **Task Size Differences.** Larger tasks (more classes) or tasks that were trained earlier might have stronger classifier weights. Conversely, smaller tasks might have very tight, well-separated features. Both can affect how likely a mismatch is.

I.5. A Rough Illustrative Bound

As a simplistic illustration, suppose:

- For task t^*, the local maximum logit Z^* has mean μ^* and variance σ^{*2}.
- All out-of-task classes have means $\mu_o < \mu^*$ and variance σ_o^2.
- There are M out-of-task classes in total.

Then

$$\Pr(\hat{y}_{\text{local}} = \hat{y}_{\text{global}}) \;\approx\; \int \Big[\Pr(Z_o \le z)\Big]^M F_{Z^*}(z)\, dz,$$

where Z_o is the logit distribution for a single out-of-task class and F_{Z^*} is the PDF of Z^*. If μ^* is sufficiently larger than μ_o (and variances are not too large), Z^* will, with high probability, exceed *all* M out-of-task logits. But as M grows large, this event can become less likely unless the margin $\mu^* - \mu_o$ is also large.

I.6. Remarks

Overall, the probability that the local argmax (over the correct task only) coincides with the global argmax (over all tasks/classes) depends on:

- How well the feature extractor ϕ separates tasks, so that out-of-task logits stay low for samples of task t^*.
- The relative scale and calibration of classifier weights W^t across tasks.
- The total number of classes from other tasks that could "compete" and produce a large logit by chance.

Table 21. Ablation studies of identifying where to place our proposed CHEEM in ViT by testing 11 components or composite components (Eqns. 3 and 4).

Index	Finetuned Component	Avg. Acc.	Avg. Forgetting
1	$\text{LN}_1 + \text{LN}_2$	81.76	21.24
2	FFN	84.20	44.76
3	MLP^{Down}	83.66	37.99
4	LN_2	80.04	16.35
5	$\text{MHSA} + \text{LN}_1$	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	85.11	30.50

If tasks are well-separated (and the classifier is carefully regularized or calibrated), this probability can be very high. Conversely, if many classes from older or different tasks produce comparably large logits, the global $\arg\max$ may differ from the local $\arg\max$ more frequently as the number of tasks and classes increases.

J. Identifying the Task-Synergy Internal Memory in ViTs

The left of Fig. 2 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,

$$z_{L,d} = x_{L,d} + \text{Proj}\Big(\text{MHSA}\big(\text{LN}_1(x_{L,d})\big)\Big), \tag{3}$$

$$y_{L,d} = z_{L,d} + \text{FFN}\Big(\text{LN}_2(z_{L,d})\Big), \tag{4}$$

where $\text{LN}(\cdot)$ represents the layer normalization [1], and $\text{Proj}(\cdot)$ is a linear transformation fusing the multi-head outputs from MHSA module. The MHSA realizes the dot-product self-attention between Query and Key, followed by aggregating with Value, where Query/Key/Value are linear transformatons of the input token sequence. The FFN is often implemented by a multi-layer perceptron (MLP) with a feature expansion layer MLP^{Up} and a feature reduction layer MLP^{Down} with a nonlinear activation function (such as the GELU [9]) in the between, i.e., $\text{FFN}(\cdot) = \text{MLP}^{\text{Down}}\Big(\text{GELU}\big(\text{MLP}^{\text{Up}}(\cdot)\big)\Big)$.

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 [21] (see Fig. 6). We first train a ViT-Base [6] on the first task, ImageNet [22], 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 defined by Equation 4 where $\text{Acc}()$ uses the Top-1 classification accuracy.

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. 3a in the main text). Let $F_{1:t}$ be the backbone trained sequentially and continually after task T_t and H_t is its head classifier. The average forgetting [4] on the first $N-1$ streaming tasks is defined by Equation 5, where $a_{j,t} = \text{Acc}(T_t; F_{1:j}, H_t)$.

As shown in Table 21, 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 either the Projection layer after the MHSA or the MLP^{Down} as the task-synergy internal (parameter) memory** to realize our proposed CHEEM for ExfCCL (Fig. 2). We test both in experiments and provide ablation studies in Section 4.7.

References

[1] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016. 10

[2] 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*, pages 3034–3042. IEEE Computer Society, 2016. 8

[3] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 - mining discriminative components with random forests. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI*, pages 446–461. Springer, 2014. 8

[4] Arslan Chaudhry, Puneet Kumar Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XI*, pages 556–572. Springer, 2018. 11

[5] 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*, pages 3606–3613. IEEE Computer Society, 2014. 8

[6] 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. 8, 11

[7] Timnit Gebru, Jonathan Krause, Yilun Wang, Duyun Chen, Jia Deng, and Li Fei-Fei. Fine-grained car detection for visual census estimation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 4502–4508. AAAI Press, 2017. 8

[8] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Introducing eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 204–207. IEEE, 2018. 8

[9] Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016. 10

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 8

[11] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 8

[12] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. 8

[13] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998. 8

[14] Fei-Fei Li, Marco Andreeto, Marc'Aurelio Ranzato, and Pietro Perona. Caltech 101, 2022. 8

[15] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew B. Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *CoRR*, abs/1306.5151, 2013. 8

[16] Stefan Munder and Dariu M. Gavrila. An experimental study on pedestrian classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(11):1863–1868, 2006. 8

[17] 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. 8

[18] 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*, pages 722–729. IEEE Computer Society, 2008. 8

[19] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pages 3498–3505. IEEE Computer Society, 2012. 8

[20] 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 *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, pages 8748–8763. PMLR, 2021. 6

[21] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 506–516, 2017. 1, 8, 11

[22] 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. 8, 11

[23] 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. 8

[24] 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. 8

[25] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, pages 10347–10357. PMLR, 2021. 8

[26] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, pages 3485–3492. IEEE Computer Society, 2010. 8

[27] Jiazuo Yu, Yunzhi Zhuge, Lu Zhang, Ping Hu, Dong Wang, Huchuan Lu, and You He. Boosting continual learning of vision-language models via mixture-of-experts adapters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23219–23230, 2024. 7

[28] Zangwei Zheng, Mingyuan Ma, Kai Wang, Ziheng Qin, Xiangyu Yue, and Yang You. Preventing zero-shot transfer degradation in continual learning of vision-language models. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 19068–19079. IEEE, 2023. 8