

APPENDIX

A RELATED WORKS

Continual learning. Continual learning methodologies (Parisi et al., 2019; De Lange et al., 2021; Masana et al., 2022) can be broadly classified into three categories: *regularization-based*, *replay-based*, and *parameter-isolation* methods. *Regularization-based* approaches typically introduce a regularization term in the loss function to constrain changes to parameters relevant to prior tasks. These can further be categorized as data-focused (Li & Hoiem, 2017; Kim et al., 2023), leveraging knowledge distillation from previously trained models, or prior-focused (Kirkpatrick et al., 2017; Zenke et al., 2017; Aljundi et al., 2018), estimating parameter importance as a prior for the new model. Recent research proposed enforcing weight updates within the null space of feature covariance (Wang et al., 2021; Tang et al., 2021). *Replay-based* methods rely on memory and rehearsal mechanisms to recall episodic memories of past tasks during training, thereby keeping the loss low in those tasks. Two main strategies are: exemplar replay - which stores selected training samples (Riemer et al., 2018; Buzzega et al., 2020; Chaudhry et al., 2018; Prabhu et al., 2020; Chaudhry et al., 2019; Liang & Li, 2024) and generative replay - with models that synthesize previous data with generative models (Shin et al., 2017; Wu et al., 2018). *Parameter isolation* methods aim to learn task-specific sub-networks within a shared network. Various techniques, such as Piggyback (Mallya et al., 2018), PackNet (Mallya & Lazebnik, 2018), SupSup (Wortsman et al., 2020), HAT (Serra et al., 2018), and Progressive Neural Network (Rusu et al., 2016), allocate and combine parameters for individual tasks. While effective in task-aware settings, these methods are most suited for scenarios with a known task sequence or oracle.

Representations in neural networks. Understanding and comparing the representations at different layers in deep neural networks is an active area of research, and tools such as CKA (Kornblith et al., 2019) emerged to measure the similarity of representations across layers. Several works have investigated how network representations behave during continual learning: Ramasesh et al. (2020) noticed that the most forgetting occurs in the deeper network layers, Zhao et al. (2023) likewise demonstrated that only a subset of modules is sensitive to the changes during continual learning, and recent work of Masarczyk et al. (2023) showed that neural networks split into parts that build different representations. Similar insights motivated continual learning methods that enforce stability through replay (Liu et al., 2020a; Pawlak et al., 2022) or regularization (Douillard et al., 2020) at the level of intermediate network layers. In representation learning, several works demonstrated that probing classifiers learned on top of intermediate network representations tend to perform relatively well (Davari et al., 2022), although usually worse than the final classifier. Early-exit techniques (Panda et al., 2016; Teerapittayanon et al., 2016; Kaya et al., 2019) use intermediate representations to reduce the inference cost through dynamic inference that enables skipping later model layers. Several works on early-exits also propose more advanced strategies (Liao et al., 2021; Sun et al., 2021; Han et al., 2022; Wójcik et al., 2023) that improve the effectiveness. Use of multiple classifiers in continual learning has been explored in an ensembling-like manner (Liu et al., 2020b), and (Yan et al., 2024) utilized intermediate classifiers for online continual learning and different motivations. Our method is dedicated to offline continual learning.

B ANALYSIS RESULTS FOR 5-TASK SPLITS

In this section, we include results with 5-tasks split corresponding to the experiments from Sections 2.1 to 2.3. Figures 7 to 9 demonstrate that our previous insights also hold for different task split.

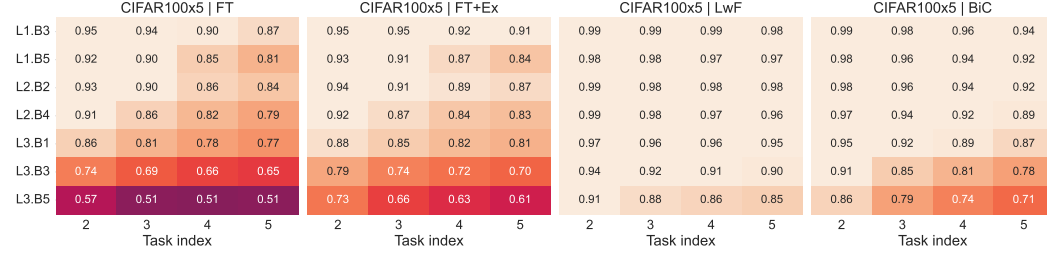


Figure 7: CKA of the first task representations across different ResNet32 layers (L1.B3-L3.B5) through continual learning on CIFAR100 split into 5 tasks.

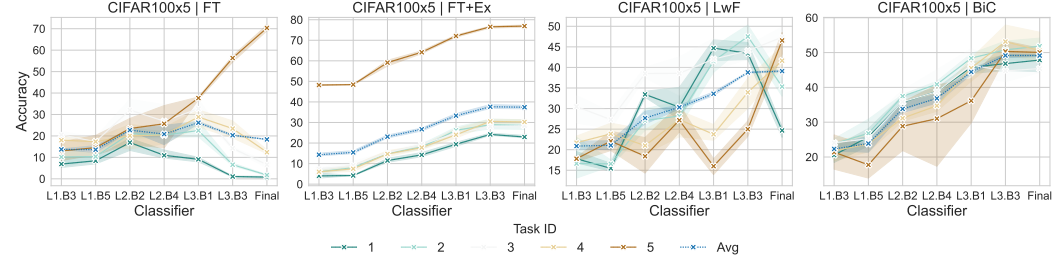


Figure 8: Per-task final accuracy of the auxiliary classifiers trained with linear probing on top of several network layers and final network classifier on CIFAR100 split into 5 tasks.

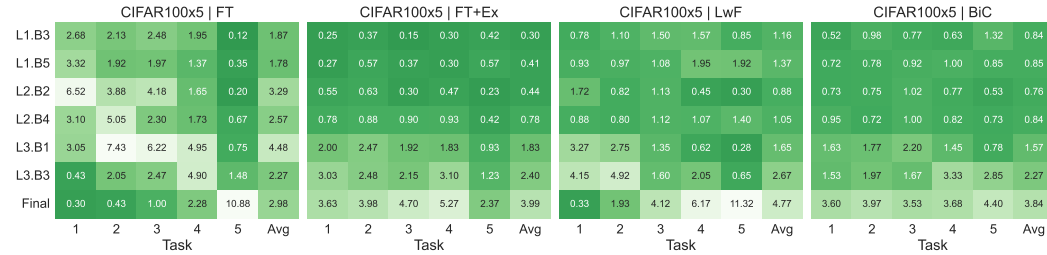


Figure 9: Unique accuracy (a subset of samples that a single given classifier classifies correctly) of auxiliary classifiers and final network classifier for different task data on CIFAR100 split into 5 tasks.

C ANALYSIS FOR METHODS WITH ENABLED GRADIENT PROPAGATION

We also replicate the analysis from Section 2 for methods with gradient propagation enabled through the network, using both 5 and 10 task split. In Figures 10 to 12 we observe similar patterns as in case of networks without gradient propagation, which validates our idea to train the network together with the classifiers.

C.1 CKA

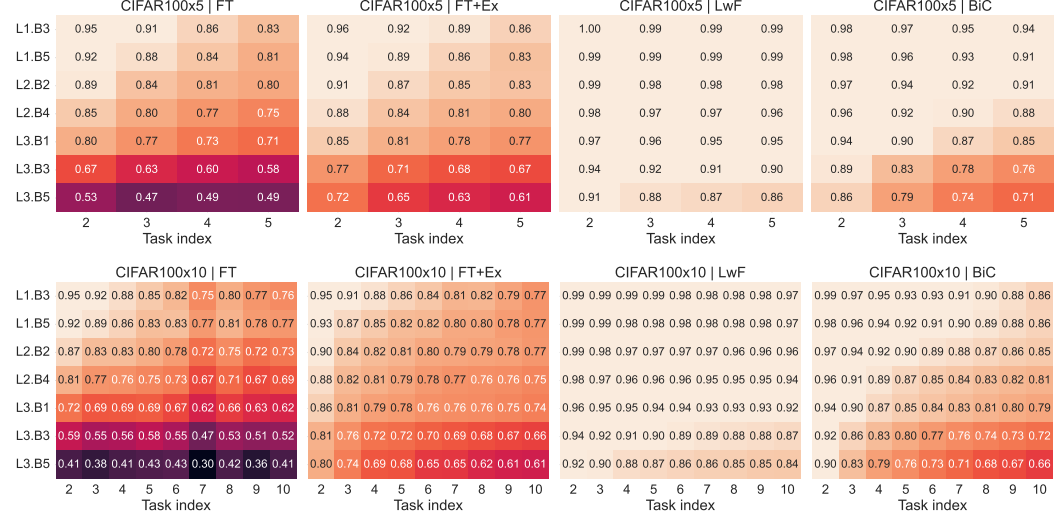


Figure 10: CKA of the first task representations across different ResNet32 layers (L1.B3-L3.B5) through continual learning on CIFAR100, **with enabled gradient propagation**.

C.2 INDIVIDUAL CLASSIFIER ACCURACY

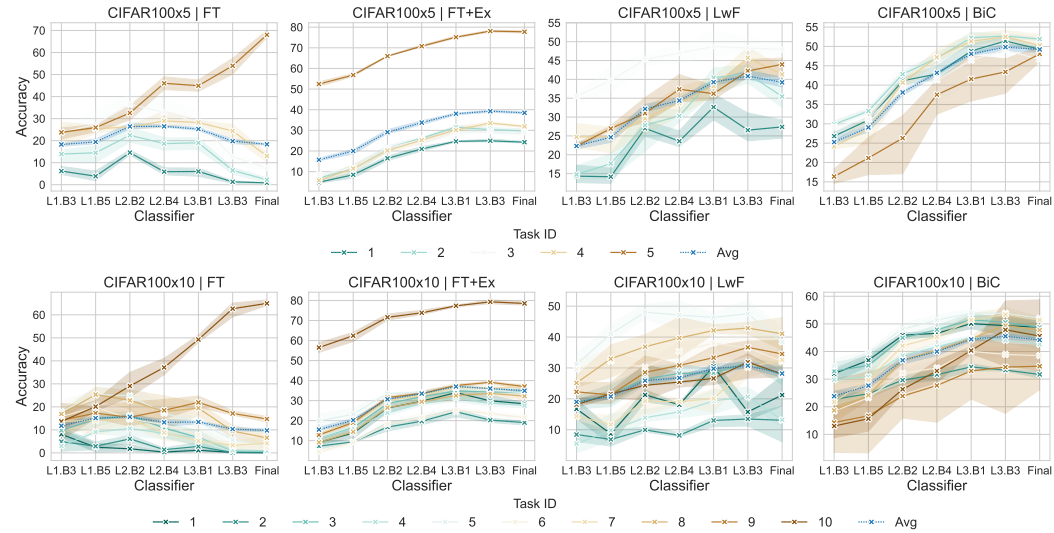


Figure 11: Per-task final accuracy of the auxiliary classifiers trained with linear probing on top of several network layers and final network classifier on CIFAR100, **with enabled gradient propagation**.

C.3 UNIQUE ACCURACY

CIFAR100x5 FT							CIFAR100x5 FT+Ex							CIFAR100x5 LwF							CIFAR100x5 BiC						
L1.B3	2.48	3.25	2.07	2.47	0.50	2.15	0.18	0.18	0.27	0.08	0.42	0.23	0.98	0.72	1.47	1.20	0.72	1.02	0.65	0.68	1.33	0.35	0.47	0.70			
L1.B5	0.78	2.25	3.15	2.58	0.45	1.84	0.42	0.60	0.18	0.58	0.47	0.45	0.53	0.85	1.30	0.75	0.70	0.83	0.52	0.60	0.73	0.60	0.83	0.66			
L2.B2	7.77	5.68	4.40	2.08	0.60	4.11	0.85	0.83	1.08	0.58	0.42	0.75	1.77	1.33	0.75	0.63	0.55	1.01	0.88	0.83	0.98	1.05	0.58	0.87			
L2.B4	1.90	3.20	3.33	3.07	1.00	2.50	1.40	1.48	1.73	1.22	0.45	1.26	0.48	1.18	0.82	0.97	1.62	1.01	0.77	1.02	0.82	1.45	1.52	1.11			
L3.B1	2.53	5.18	3.57	3.00	0.68	2.99	2.37	2.63	2.07	1.85	0.90	1.96	2.58	3.10	1.28	1.15	0.97	1.82	1.10	1.53	1.37	1.43	1.75	1.44			
L3.B3	0.47	1.22	1.45	3.25	0.58	1.39	1.80	1.60	1.90	2.12	0.95	1.67	0.92	2.63	1.13	2.57	1.62	1.77	1.55	1.42	1.83	1.93	1.30	1.61			
Final	0.38	0.30	0.90	1.00	6.98	1.91	2.48	2.68	2.85	2.83	1.40	2.45	2.10	1.97	2.15	2.52	3.82	2.51	1.95	2.75	2.42	2.05	4.78	2.79			
	1	2	3	4	5	Avg	1	2	3	4	5	Avg	1	2	3	4	5	Avg	1	2	3	4	5	Avg			
Task							Task							Task							Task						

CIFAR100x10 FT												CIFAR100x10 FT+Ex												CIFAR100x10 LwF												CIFAR100x10 BiC											
L1.B3	7.50	3.93	3.27	1.83	3.07	4.83	3.67	1.93	2.83	0.27	3.31	0.37	0.60	0.30	0.27	0.77	0.23	0.60	0.20	0.40	0.70	0.44	3.20	1.97	2.20	0.57	0.93	1.93	1.60	0.87	1.73	0.80	1.58	0.70	0.17	0.87	1.30	0.70	0.50	0.27	0.73	0.60	0.74				
L1.B5	2.07	1.03	5.50	5.23	6.23	6.10	4.23	4.33	3.97	0.27	3.90	0.63	0.50	0.63	1.17	0.60	0.90	0.30	0.60	0.57	0.43	0.63	0.37	0.83	1.33	1.47	1.57	1.90	0.67	1.70	0.80	1.03	1.17	0.70	0.70	0.73	0.87	0.93	1.00	0.67	0.67	0.83	0.50	0.76			
L2.B2	1.37	4.20	5.63	5.10	5.13	2.90	5.23	3.80	2.20	0.50	3.61	1.67	1.17	1.40	1.60	1.73	1.57	1.70	1.63	1.70	0.57	1.47	1.77	0.73	1.10	1.23	1.20	0.60	0.87	1.13	0.70	1.13	1.05	1.40	1.03	1.03	1.23	0.97	1.17	1.03	0.93	1.03	0.73	1.06			
L2.B4	0.17	0.43	0.10	2.97	2.80	2.87	2.23	2.43	2.20	0.43	1.96	1.57	1.73	1.70	1.73	1.23	1.67	1.40	1.40	1.07	0.57	1.41	0.50	0.50	1.20	1.23	1.23	0.97	0.83	0.83	0.80	0.40	0.85	0.97	1.23	1.07	1.07	1.17	1.27	0.97	1.57	1.07	1.43	1.18			
L3.B1	0.70	1.60	2.13	1.83	1.77	1.87	1.07	3.40	3.27	0.87	1.85	2.83	3.40	2.53	2.57	1.70	3.03	1.97	1.93	1.63	1.07	2.27	4.77	1.53	1.33	2.60	1.30	1.70	0.53	1.37	0.80	0.83	1.68	1.37	1.87	1.47	1.70	1.43	1.87	1.90	1.67	1.93	1.87	1.71			
L3.B3	0.13	0.10	0.43	0.20	1.50	0.63	0.70	0.93	1.53	2.10	0.83	1.30	1.23	1.23	1.13	1.10	2.10	1.03	1.57	1.67	0.73	1.31	0.47	1.57	2.13	2.60	1.53	1.97	2.00	1.40	2.00	2.23	1.78	1.10	1.13	1.20	1.10	1.60	1.60	1.47	2.30	1.90	2.30	1.59			
Final	0.07	0.10	0.33	0.20	0.33	0.90	1.27	0.70	1.27	3.87	0.90	1.40	1.43	2.37	1.70	1.83	2.67	2.83	2.50	2.43	0.97	2.01	2.17	1.97	2.50	1.17	0.93	2.00	4.23	2.23	2.90	2.13	2.22	1.50	1.87	1.83	2.10	1.90	2.53	2.20	1.33	4.20	3.03	2.43			
	1	2	3	4	5	6	7	8	9	10	Avg	1	2	3	4	5	6	7	8	9	10	Avg	1	2	3	4	5	6	7	8	9	10	Avg	1	2	3	4	5	6	7	8	9	10	Avg			
Task												Task												Task												Task											

Figure 12: Unique accuracy (a subset of samples that a single given classifier classifies correctly) of auxiliary classifiers and final network classifier for different task data on CIFAR100, **with enabled gradient propagation**.

D MULTI-CLASSIFIER PERFORMANCE UPPER BOUND ANALYSIS

To quantify the potential of the auxiliary classifiers, we consider an oracle multi-classifier network as an upper bound for our method. When evaluating such an oracle, we obtain predictions from all of its classifiers and consider a prediction correct when at least one classifier (auxiliary classifier or the original network classifier) is correct. Therefore, the oracle has an ideal algorithm for combining classifier predictions and always returns the 'best case' prediction from all the classifiers. We measure the difference between the accuracy of such an oracle network and the accuracy of a standard single-classifier network and present the results for first task data, last task data, and average over all the tasks in Table 8 and Table 9 for both linear probing and ACs. As in our previous analysis in Section 2, exemplar-free methods show more variance in the performance across tasks. However, the average difference across all tasks is also significant for exemplar-based methods, with the oracle for the best-performing method - BiC - achieving approximately a 30-40% relative increase in overall accuracy. Those results indicate that, while our simple setup achieves consistent improvement, it is still far from the best-case utilization of multi-classifier networks and still leaves room for future work.

Table 8: Upper bound on accuracy improvement on 5 tasks of CIFAR100 when using oracle multi-classifier network, trained with linear probing and auxiliary classifiers.

	Task 1	Task 2	Task 3	Task 4	Task 5	Avg
CIFAR100x5, Linear Probing						
FT	31.82±2.23	46.90±1.60	54.22±0.59	43.30±1.24	6.57±0.96	36.56±0.52
FT+Ex	12.28±0.10	14.58±0.33	12.72±1.80	13.87±0.93	11.37±0.71	12.96±0.62
LwF	35.07±2.03	28.88±2.51	20.92±2.01	15.25±1.13	9.55±1.47	21.93±0.20
BiC	15.15±0.91	17.00±2.33	18.73±1.50	17.93±2.26	16.03±2.39	16.97±0.50
CIFAR100x5, Auxiliary Classifiers						
FT	24.65±2.82	45.07±5.77	56.87±3.68	48.82±4.36	9.05±0.65	36.89±1.33
FT+Ex	14.42±0.60	17.42±1.12	15.70±1.58	15.30±1.59	10.88±0.40	14.74±0.66
LwF	18.52±1.89	22.57±2.84	21.57±0.93	18.57±0.95	15.07±2.08	19.26±0.61
BiC	15.88±0.96	18.42±0.46	18.83±2.29	18.72±0.06	15.43±3.06	17.46±0.61

Table 9: Upper bound on accuracy improvement on 10 tasks of CIFAR100 when using oracle multi-classifier network, trained with linear probing and auxiliary classifiers.

	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9	Task 10	Avg
CIFAR100x10, Linear Probing											
FT	30.00±7.52	16.10±3.90	40.47±6.03	23.37±4.57	48.70±0.72	41.43±5.52	32.80±1.78	44.40±4.83	28.87±0.97	9.03±1.72	31.52±0.64
FT+Ex	17.17±1.24	16.63±0.42	18.40±2.72	18.10±1.44	17.67±0.78	16.30±1.31	18.90±0.79	15.77±1.30	17.00±0.44	11.90±1.32	16.78±0.38
LwF	48.93±6.24	30.30±3.00	33.63±3.35	28.93±5.05	25.60±2.13	21.50±5.50	13.30±2.17	14.47±1.50	11.33±1.24	8.40±1.54	23.64±1.29
BiC	19.03±1.10	20.73±2.04	20.50±2.74	20.83±2.37	17.83±1.12	19.17±0.67	16.83±3.16	18.90±0.53	20.77±4.80	17.73±4.31	19.23±0.70
CIFAR100x10, Auxiliary Classifiers											
FT	12.93±0.38	14.30±4.69	35.07±0.21	25.70±4.90	46.27±4.57	38.70±2.77	32.50±2.17	43.77±0.93	34.73±3.46	9.20±0.62	29.32±0.98
FT+Ex	20.03±1.56	17.77±1.47	18.87±0.60	20.33±1.70	18.13±2.25	20.50±0.78	18.97±1.95	17.47±1.36	19.13±2.48	12.30±0.80	18.35±0.01
LwF	22.80±3.46	12.60±4.18	22.93±1.65	21.50±2.91	28.30±1.80	25.33±4.65	12.70±1.49	21.53±3.31	18.00±5.12	15.27±2.77	20.10±0.74
BiC	18.87±2.76	20.60±0.46	21.53±2.74	22.70±2.52	18.60±1.51	19.87±2.01	17.17±1.88	19.77±0.57	18.77±8.36	17.73±7.03	19.56±0.78

E DYNAMIC INFERENCE RULE ABLATION

In Table 10 we demonstrate the accuracy of two variants of dynamic inference for different confidence thresholds λ for CIFAR100. We compare the standard, early-exit paradigm, where the network returns a final classifier prediction in case no classifier meets the exit rule, and the paradigm used in our experiments where the network defaults to the most confident prediction. Using the most confident prediction outperforms the standard early-exit rule, which is consistent with our analysis that showed that the last classifier is not always the best in continual learning and that the early classifiers exhibit lower forgetting for earlier task data.

Table 10: Comparison between dynamic inference performance with different confidence thresholds λ when using maximum confidence prediction (MC) and final classifier prediction (Last) as the default output for multi-classifier networks trained with linear probing (LP) or jointly with the network with gradient propagation (AC). Using max confidence prediction yields better accuracy.

	λ	FT (LP)	FT+Ex (LP)	LwF (LP)	BiC (LP)	FT (AC)	FT+Ex (AC)	LwF (AC)	BiC (AC)
CIFAR100x5									
Last MC	0.5	24.14±1.35 24.73±1.48	28.43±0.87 28.33±0.96	37.34±0.09 36.95±0.20	48.35±0.23 48.48±0.40	27.64±0.97 28.10±1.03	28.87±0.32 28.85±0.31	38.08±0.87 38.36±0.59	48.15±0.40 48.37±0.31
Last MC	0.75	23.67±1.32 26.66±1.70	35.60±1.26 35.09±1.38	40.21±0.08 39.70±0.10	49.25±0.33 49.74±0.48	26.00±0.67 28.91±1.07	36.27±0.28 36.18±0.17	39.41±1.01 40.33±0.76	49.45±0.73 50.19±0.63
Last MC	0.9	20.98±0.99 26.70±1.55	37.27±1.10 36.57±1.33	39.97±0.27 40.07±0.10	49.18±0.36 49.89±0.52	22.38±0.33 28.44±1.04	38.22±0.19 38.27±0.13	39.28±1.28 40.50±0.91	49.35±0.65 50.39±0.65
Last MC	0.95	19.91±0.76 26.74±1.40	37.48±0.98 36.77±1.29	39.62±0.23 40.07±0.08	49.15±0.31 49.89±0.52	20.69±0.24 28.25±1.05	38.50±0.17 38.62±0.21	39.25±1.36 40.53±0.94	49.24±0.63 50.40±0.66
Last MC	0.98	19.19±0.27 26.83±1.23	37.46±0.92 36.81±1.27	39.40±0.21 40.10±0.07	49.14±0.32 49.89±0.52	19.49±0.18 28.19±1.08	38.55±0.38 38.72±0.24	39.22±1.34 40.55±0.95	49.24±0.65 50.40±0.68
Last MC	0.99	18.94±0.15 26.82±1.22	37.44±0.88 36.83±1.27	39.32±0.22 40.10±0.07	49.14±0.32 49.89±0.52	19.08±0.20 28.20±1.09	38.55±0.44 38.75±0.27	39.22±1.35 40.55±0.95	49.23±0.64 50.40±0.68
Last MC	1.0	18.39±0.08 26.82±1.19	37.43±0.85 36.83±1.27	39.11±0.26 40.10±0.07	49.14±0.32 49.89±0.52	18.35±0.30 28.18±1.07	38.51±0.43 38.75±0.26	39.22±1.34 40.55±0.95	49.22±0.65 50.40±0.68
CIFAR100x10									
Last MC	0.5	15.22±1.64 16.05±1.95	27.03±0.90 27.04±0.87	28.69±0.79 28.06±0.93	42.37±1.60 42.62±1.75	15.85±1.10 16.79±1.34	27.68±0.42 27.72±0.37	28.96±1.29 29.09±1.10	42.40±1.17 42.67±1.44
Last MC	0.75	14.12±0.98 17.47±1.51	33.69±1.20 33.84±1.03	31.10±0.87 30.17±0.77	43.95±1.69 44.59±1.75	13.31±1.17 16.77±1.22	34.40±0.47 34.64±0.41	28.65±1.72 29.72±1.19	44.93±0.93 45.83±1.51
Last MC	0.9	12.19±0.51 17.71±1.33	34.61±1.05 35.41±0.87	31.03±0.92 30.54±0.76	43.87±1.63 44.82±1.71	11.25±1.04 16.82±1.14	35.83±0.51 36.59±0.52	28.30±1.83 29.79±1.21	44.55±0.62 46.14±1.46
Last MC	0.95	11.25±0.55 17.74±1.31	34.49±1.10 35.58±0.92	30.63±1.10 30.59±0.67	43.79±1.72 44.84±1.72	10.62±0.83 16.89±1.11	35.63±0.43 36.92±0.39	28.27±1.79 29.79±1.21	44.33±0.53 46.17±1.45
Last MC	0.98	10.51±0.35 17.77±1.30	34.27±1.13 35.61±0.90	30.17±1.19 30.60±0.68	43.77±1.72 44.84±1.73	10.09±0.73 16.88±1.08	35.29±0.39 36.97±0.40	28.22±1.81 29.79±1.21	44.23±0.54 46.19±1.47
Last MC	0.99	10.19±0.37 17.77±1.30	34.26±1.10 35.61±0.89	30.00±1.20 30.60±0.68	43.76±1.72 44.84±1.73	9.92±0.69 16.88±1.08	35.10±0.45 36.98±0.40	28.20±1.83 29.79±1.21	44.21±0.53 46.19±1.47
Last MC	1.0	9.79±0.33 17.77±1.30	34.22±1.08 35.62±0.89	29.65±1.19 30.60±0.69	43.75±1.72 44.84±1.73	9.76±0.63 16.88±1.08	34.93±0.40 36.97±0.39	28.19±1.84 29.79±1.21	44.19±0.51 46.19±1.47

F AC-ENHANCED METHODS ON LONGER TASK SEQUENCES

In Table 11 and Figures 13 and 14 we present results for experiments with 20 and 50 task split, following the setup from Section 4. For the 50-task split, we use a growing memory of 20 exemplars instead of a constant memory of 2000 due to the task size. Non-replay-based methods perform poorly on longer sequences of tasks, especially on 50 tasks, but ACs robustly enhance the method performance in all tested scenarios.

Table 11: Additional results for AC-enhanced methods with longer sequences of tasks on CIFAR100

Method	FT	FT+Ex	GDumb	ANCL	BiC	DER++	ER	EWC	LwF	LODE	SSIL	Avg
CIFAR100x20												
Base	4.72 \pm 0.75	32.35 \pm 0.26	23.68 \pm 1.08	19.34 \pm 0.32	38.81 \pm 1.02	34.50 \pm 0.33	30.94 \pm 0.53	5.51 \pm 0.36	18.97 \pm 1.20	37.90 \pm 0.37	36.86 \pm 1.21	25.78 \pm 0.32
+AC	7.33\pm0.45	37.16\pm0.63	30.11\pm0.36	20.87\pm0.78	42.03\pm0.18	36.45\pm0.81	36.10\pm0.07	9.70\pm0.18	19.63\pm0.75	41.85\pm0.49	39.78\pm0.24	29.18\pm0.10
Δ	+2.61 \pm 0.61	+4.81 \pm 0.86	+6.43 \pm 0.98	+1.53 \pm 0.91	+3.22 \pm 1.01	+1.95 \pm 1.05	+5.16 \pm 0.60	+4.19 \pm 0.40	+0.66 \pm 1.69	+3.95 \pm 0.65	+2.92 \pm 1.03	+3.40 \pm 0.22
CIFAR100x50												
Base	0.97 \pm 0.51	21.60 \pm 0.88	13.48 \pm 0.89	5.94 \pm 0.37	24.91 \pm 0.67	16.79 \pm 1.38	18.35 \pm 0.38	1.63 \pm 0.55	5.09 \pm 0.51	23.56 \pm 1.82	22.66 \pm 0.39	14.09 \pm 0.28
+AC	1.64\pm0.19	26.39\pm0.16	17.63\pm0.90	6.41\pm0.59	30.10\pm0.36	22.59\pm0.39	22.96\pm0.29	2.12\pm0.02	5.40\pm0.60	28.53\pm1.29	25.77\pm0.74	17.23\pm0.18
Δ	+0.67 \pm 0.67	+4.79 \pm 0.88	+4.15 \pm 0.60	+0.47 \pm 0.22	+5.19 \pm 0.70	+5.80 \pm 1.01	+4.61 \pm 0.52	+0.49 \pm 0.57	+0.31 \pm 0.88	+4.97 \pm 2.06	+3.11 \pm 1.12	+3.14 \pm 0.44

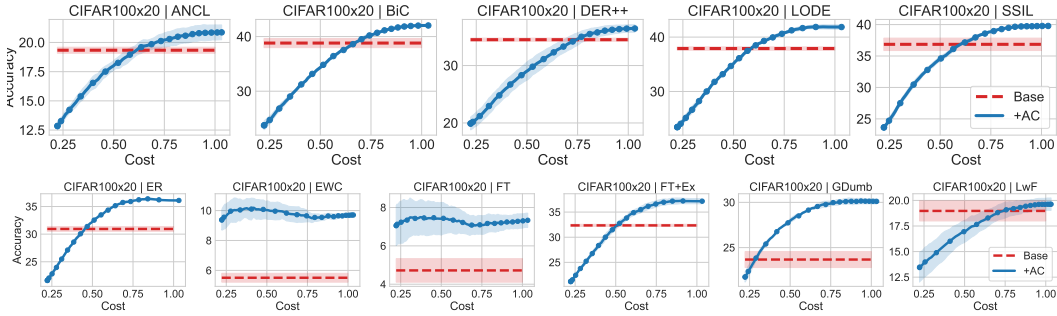


Figure 13: Dynamic inference plots for 20 task split of CIFAR100.

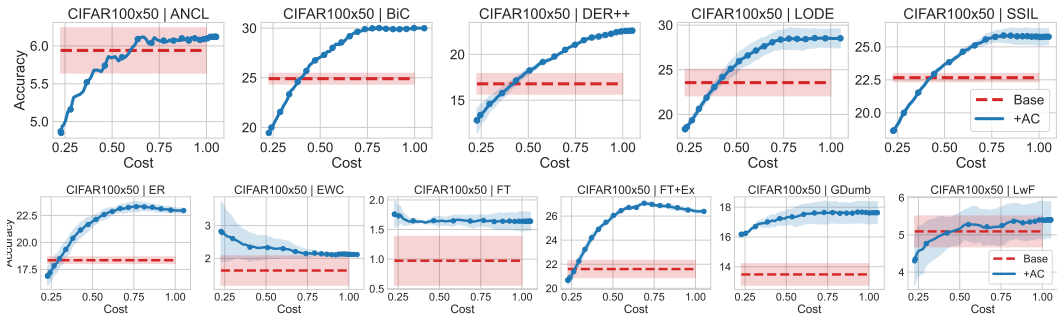


Figure 14: Dynamic inference plots for 50 task split of CIFAR100.

G ADDITIONAL DYNAMIC INFERENCE PLOTS

In this section, we present dynamic inference plots obtained for experiments performed in Section 4 or complementary to those experiments.

G.1 STANDARD BENCHMARKS

In Figure 5, we present dynamic inference results for 10 task splits of CIFAR100 and ImageNet100 using ANCL, BiC, ER, LODE, and SSIL. In this section, we provide corresponding results for the 10-task split and the rest of the analyzed methods: EWC, FT, FT+Ex, GDumb, and LwF.

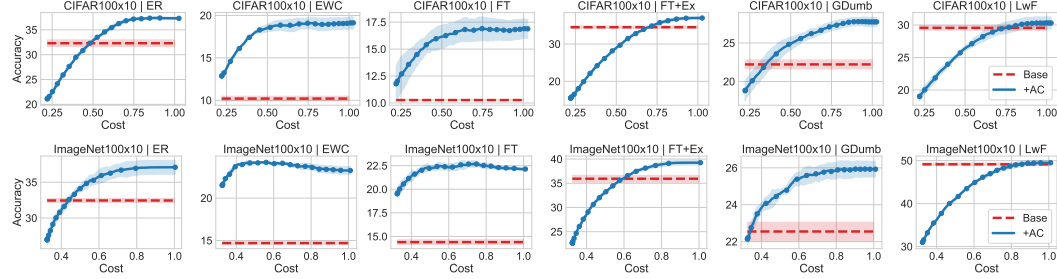


Figure 15: Dynamic inference plots as in Figure 5 for additional continual learning methods extended with auxiliary classifiers on CIFAR100 and ImageNet100 split into 10 tasks.

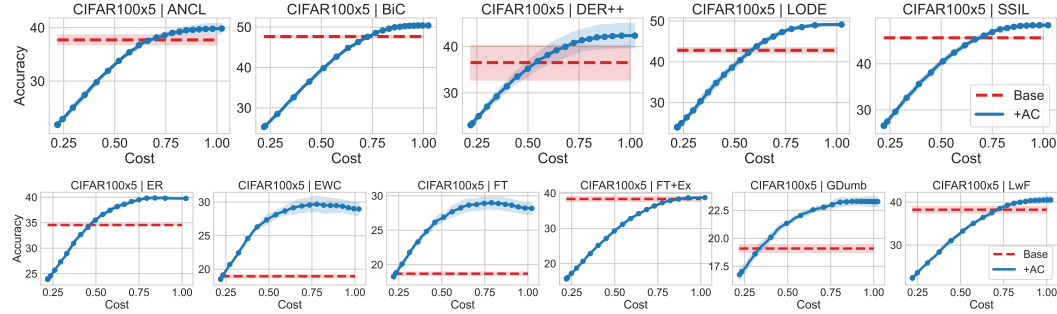


Figure 16: Dynamic inference plots for continual learning methods extended with auxiliary classifiers on CIFAR100 split into 5 tasks.

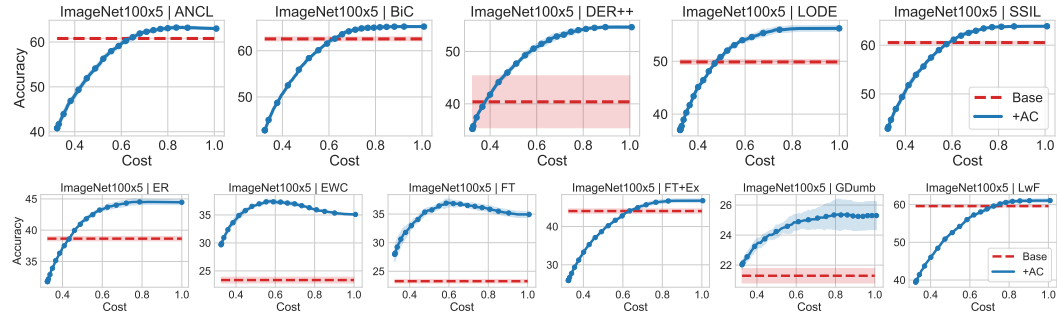


Figure 17: Dynamic inference plots for continual learning methods extended with auxiliary classifiers on ImageNet100 split into 5 tasks.

G.2 DYNAMIC INFERENCE PLOTS FOR WARM START SETTING

In this section, we provide dynamic accuracy plots for the warm start setting described in Section 4.

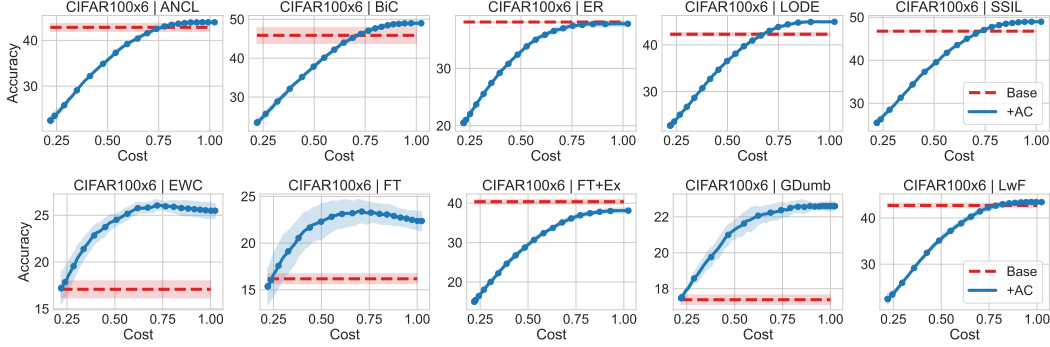


Figure 18: Dynamic inference plots for continual learning methods extended with auxiliary classifiers compared with the baselines corresponding to the results from the Table 2 for CIFAR100 starting from 50 classes and then training on the sequence of 5 tasks with 10 classes each.

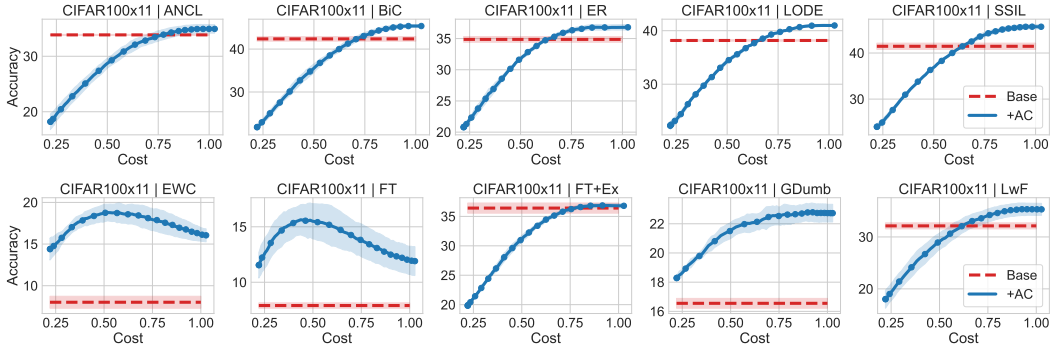


Figure 19: Dynamic inference plots for continual learning methods extended with auxiliary classifiers compared with the baselines corresponding to the results from the Table 2 for CIFAR100 starting from 50 classes and then training on the sequence of 10 tasks with 5 classes each.

G.3 DYNAMIC ACCURACY WITH DIFFERENT NUMBER OF ACS

In this section, we provide dynamic accuracy plots for experiments performed with different numbers of ACS in Section 4.

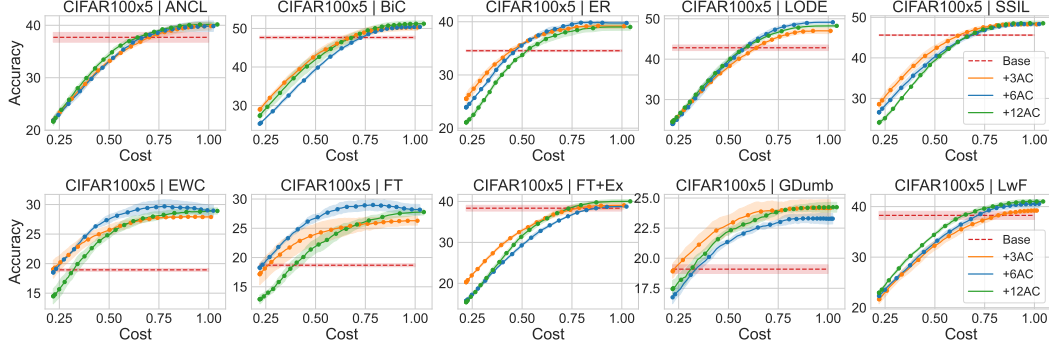


Figure 20: Dynamic inference plots for continual learning methods extended using a different number of auxiliary classifiers on CIFAR100 split in 5 tasks.

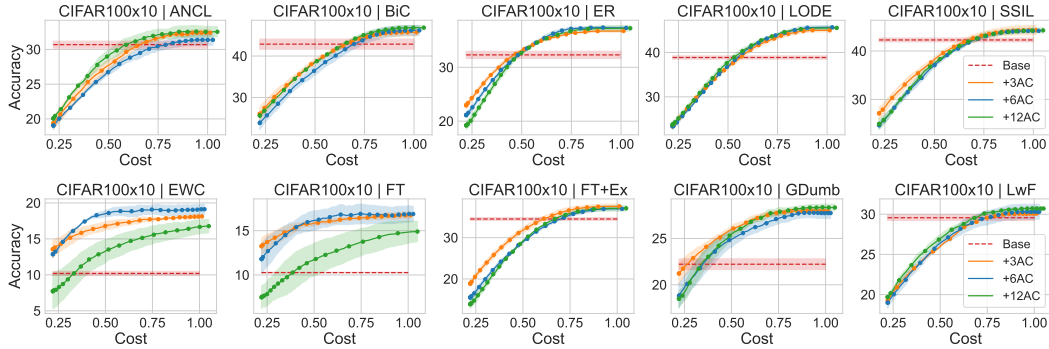


Figure 21: Dynamic inference plots for continual learning methods extended using a different number of auxiliary classifiers on CIFAR100 split in 10 tasks.

G.4 DYNAMIC ACCURACY FOR DIFFERENT CLASSIFIER ARCHITECTURES

In this section, we provide dynamic accuracy plots for experiments performed with different classifier architectures in Section 4.

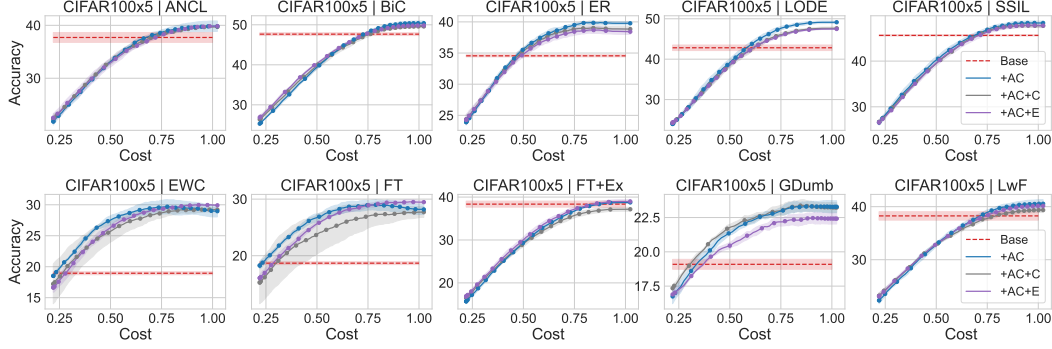


Figure 22: Dynamic inference plots for continual learning methods extended with different auxiliary classifier architecture on CIFAR100 split in 5 tasks.

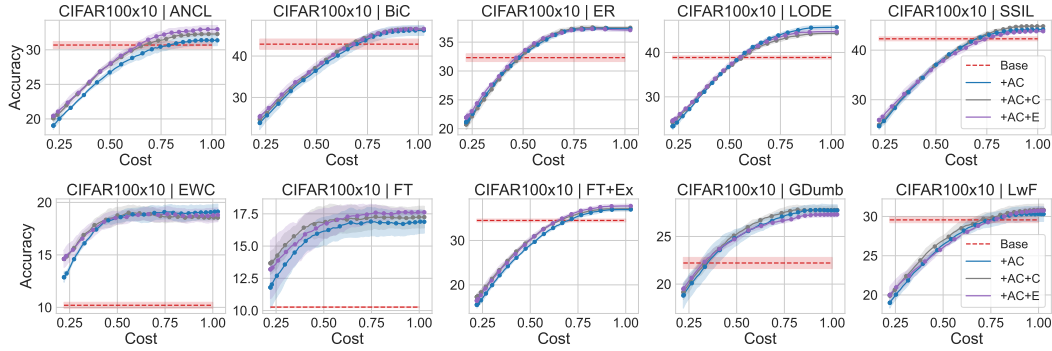


Figure 23: Dynamic inference plots for continual learning methods extended with different auxiliary classifier architecture on CIFAR100 split in 10 tasks.

G.5 VGG19 NETWORKS

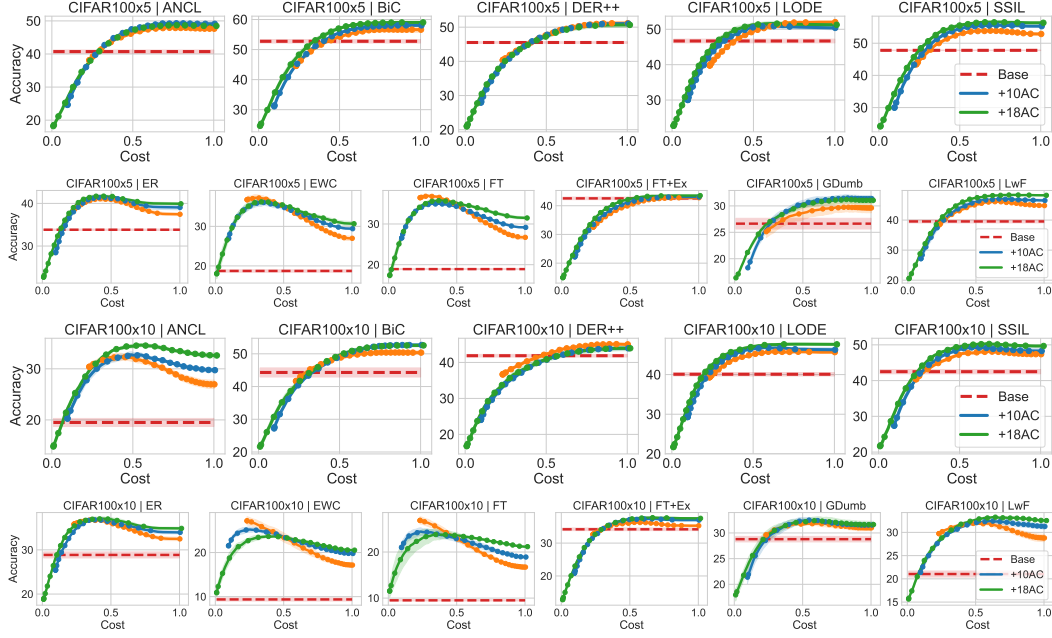


Figure 24: Dynamic inference experiments on CIFAR100 with VGG19 network corresponding to the ones from Section 4. As VGG networks are deeper, we attach either 6, 10, or 18 ACs. AC-enhanced methods perform even better than in the case of ResNet, matching the accuracy of the base method at a small fraction of its compute and outperforming the baseline significantly at the full computational budget.

G.6 VISION TRANSFORMERS

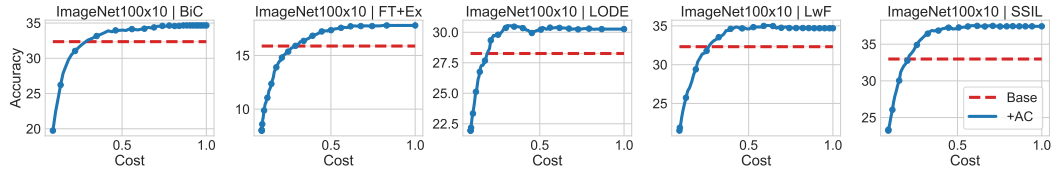


Figure 25: Dynamic inference plots for several continual learning methods extended with auxiliary classifiers compared with the baselines using Vision Transformer trained from scratch on ImageNet100 split into 5 tasks.

H EXPERIMENTAL COMPARISON BETWEEN OUR METHOD AND LINEAR PROBING

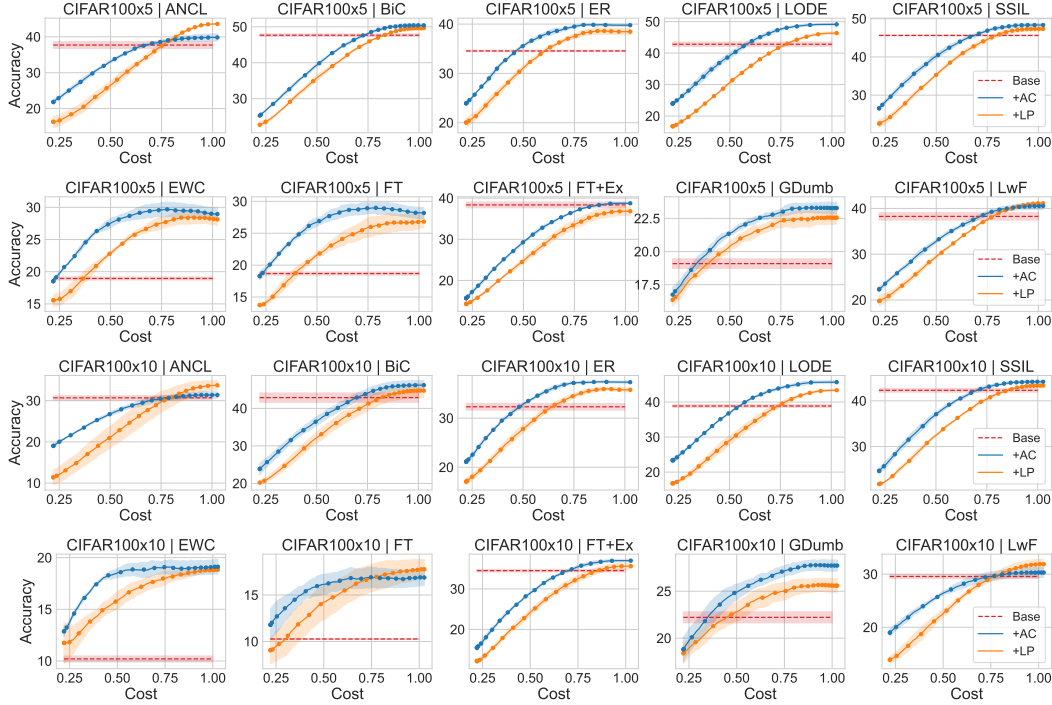


Figure 26: Dynamic inference plots for several continual learning methods extended with auxiliary classifiers when using auxiliary classifiers with enabled gradient propagation (AC) or without (LP). Enabled propagation generally improves the results at low to mid-computation budgets, with ANCL and LwF being the only outliers at high computational budgets. This variation could be explained by the variance in intermediate classifier predictions shown in Section 2.2.

In Section 2.4, we advocate for training the network and ACs jointly with enabled gradient propagation, as it leads to better performance of individual classifiers. In this section, we investigate the final performance of linear probing classifiers in comparison with jointly trained ACs on CIFAR100. For ACs use the same setup as in Section 4, and in the case of linear probing the only difference is that the classifiers are trained without gradient propagation. We show the final performance of both settings in Table 12, and also demonstrate their cost-accuracy characteristics in Figure 26. Aside from distillation-based exemplar-free methods, ACs outperform probing accuracy, and in all cases, networks learned with probing still achieve an improvement upon the baseline. However, dynamic accuracy curves highlight that enabled gradient propagation allows AC methods to achieve greater accuracy at lower computational costs due to the ability to learn better early classifiers.

I AC ARCHITECTURE AND TRAINING DETAILS

In our main experiments, we follow the insights from Kaya et al. (2019) and Wójcik et al. (2023) in our design of multi-classifier networks. We place the ACs after layers that perform roughly 15%, 30%, 45%, 60%, 75%, 90% of the computations of the full network. For convolutional networks, ACs are composed of pooling layers to reduce the input size for the fully connected networks that produce the predictions. For experiments on ViT, we apply a fully connected classifier on top of the LayerNorm layer on the first token. All the classifiers in our model are composed of heads for each task, and we add a new head upon encountering a new task.

Our main objective used for training the network on any given task is a weighted sum of losses for each classifier. For continual learning methods, we use the additional losses alongside the cross-entropy and weigh the total loss. We train the model for each task jointly with all the ACs, updating

Table 12: Comparison between final results when using intermediate classifiers trained together with the network (AC) or trained with linear probing (LP). Training classifiers together generally yields better performance, with the only noticeable exception being exemplar-free distillation-based methods (ANCL and LwF), which could be caused by significant variance in per-task accuracy of intermediate classifiers as shown in Section 2.2.

Method	FT	FT+Ex	GDumb	ANCL	BiC	ER	EWC	LwF	LODE	SSIL	Avg
CIFAR100x5											
Base	18.68 \pm 0.31	38.35 \pm 0.86	19.09 \pm 0.44	37.71 \pm 1.14	47.66 \pm 0.43	34.55 \pm 0.21	18.95 \pm 0.29	38.26 \pm 0.98	42.82 \pm 0.84	45.62 \pm 0.16	34.17 \pm 0.27
+LP	26.82 \pm 1.19	36.83 \pm 1.27	22.56 \pm 0.63	43.60\pm0.19	49.62 \pm 0.07	38.47 \pm 0.78	28.13 \pm 1.11	41.13\pm0.33	46.35 \pm 0.45	47.33 \pm 0.60	38.09 \pm 0.45
+AC	28.18\pm1.07	38.75\pm0.26	23.29\pm0.54	39.83 \pm 1.22	50.40\pm0.68	39.77\pm0.32	28.96\pm1.13	40.55 \pm 0.95	49.13\pm0.35	48.35\pm0.50	38.72\pm0.61
CIFAR100x5											
Base	10.27 \pm 0.05	34.51 \pm 0.40	22.22 \pm 0.72	30.69 \pm 0.62	42.87 \pm 1.51	32.31 \pm 0.82	10.20 \pm 0.35	29.56 \pm 0.44	38.87 \pm 0.45	42.29 \pm 0.49	29.38 \pm 0.26
+LP	17.77\pm1.30	35.62 \pm 0.89	25.60 \pm 0.91	33.72\pm1.38	44.74 \pm 2.31	35.78 \pm 0.46	18.84 \pm 0.19	31.88\pm1.11	43.37 \pm 0.31	43.33 \pm 0.08	33.06 \pm 0.17
+AC	16.88 \pm 1.08	36.97\pm0.39	27.74\pm0.73	31.37 \pm 0.94	46.19\pm1.47	37.32\pm0.28	19.12\pm0.88	30.31 \pm 1.14	45.67\pm0.52	44.17\pm0.28	33.57\pm0.22

all the parameters of the network. We follow the weight scheduler from Kaya et al. (2019) and progressively increase loss weights for different ACs over the training phase to the values matching their computational cost (e.g. the weight for the first classifier for ResNet32 would increase up to 0.15, for the second classifier to 0.30, and so on). For ResNet18 we use 6 ACs and set weights to [0.3, 0.4, 0.55, 0.65, 0.8, 0.9], as the network contains only 8 blocks whose computational cost distributes approximately like that. For experiments with 12 ACs, we attach classifiers to all blocks L1.B3-L3.B4 and interpolate the weights from the standard setting. For 3 ACs, we use blocks L1.B3, L2.B2, and L3.B1 with weights [0.15, 0.45, 0.75]. When training ViT or VGG networks, for each model block we use multiplies of a given base weight (e.g. 0.08 for ViT-base, 0.05 or 0.09 for 18 and 10 AC setup for VGG19). For example, we set the AC weights for 11 ACs in ViT as [0.08, 0.16, ..., 0.80, 0.88]. Different loss weights for each AC serve to stabilize the training and mitigate overfitting in the earlier layers, which may have lower learning capacity.

We train the ResNet32 models on CIFAR100 for 200 epochs on each task, using SGD optimizer with a batch size of 128 with a learning rate initialized to 0.1 and decayed by a rate of 0.1 at the 60th, 120th, and 160th epochs. For training ResNet18 on ImageNet100, we change the scheduler to cosine with a linear warmup and train for 100 epochs with 5 epochs of warmup, as we find it to converge to similar results in a shorter time. For ViT, we use AdamW and train each task for 100 epochs with a learning rate of 0.01 and batch size of 64. We also use a cosine scheduler with a linear warmup for 5 epochs. We use a fixed memory of 2000 exemplars selected with herding (Rebuffi et al., 2017). For ER each batch is balanced between old and new data, and for SSIL we use a 4:1 ratio of new to old data. Otherwise, for other exemplar-based methods, we follow the standard FACIL procedure for exemplars and just add them to the training data without any balancing.

J CLASSIFIER SELECTION AND ACCURACY DURING INFERENCE

In Figures 27 and 28 we show the final distribution of the selected classifiers per task and on average across the test dataset for the CIFAR100 experiments presented in Table 1. We also present the corresponding accuracy of each classifier, assuming the classifier was selected for static inference as described in Section 3. We observe that early classifiers are not selected that often. However, when selected, intermediate classifiers (L2.B2-L3.B3) usually exhibit accuracy on par or better than the final classifier. This hints at the cause of the improvement observed from adding the ACs.

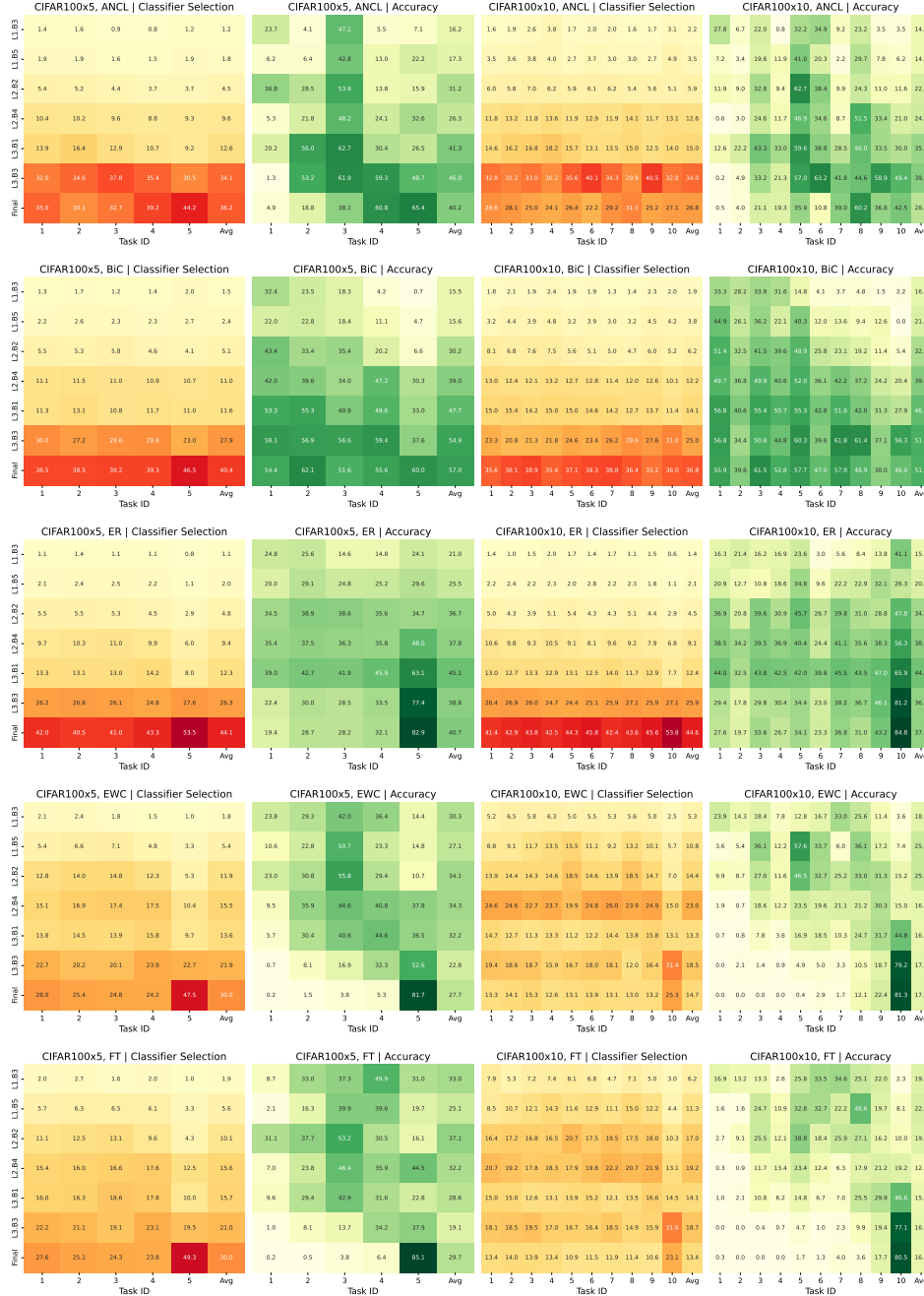


Figure 27: Distribution and accuracy of classifiers for ANCL, BiC, ER, EWC and FT.

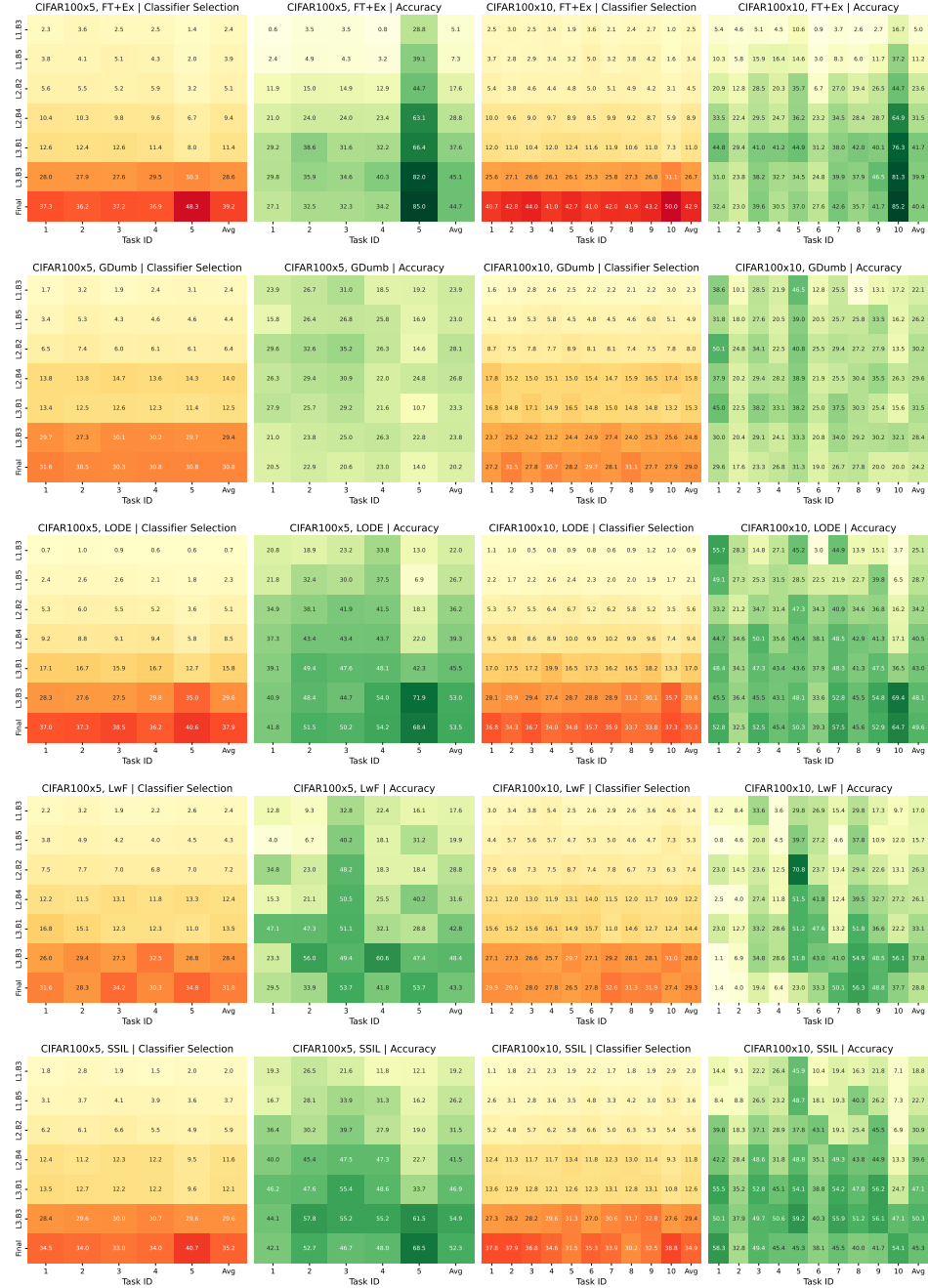


Figure 28: Distribution and accuracy of classifiers for FT+Ex, GDumb, LODE, LwF and SSIL.

K SINGLE AC ABLATION

In this section, we perform leave-one-out ablation of the setting explored in Section 4 for CIFAR100 on methods explored in Section 2. Namely, we train the model with only one auxiliary classifier out of the original six, with the classifier weight equal to 1. during training and present dynamic inference results in Figure 29. For non-naïve methods, we observe that 5th or 6th AC achieve the best performance. Interestingly for finetuning, later AC yield lower performance, which is consistent with our observations on more native stability in early layers. All tested AC setups achieve comparable performance at the full computational budget, but compared to our based setup of using all 6 ACs they tend to underperform at lower compute budget. Slightly better performance of single AC setup for FT+Ex and LwF hints that AC placement in our work could be further optimized. However, overall similar performance across all the tested scenarios prove the robustness of our idea.

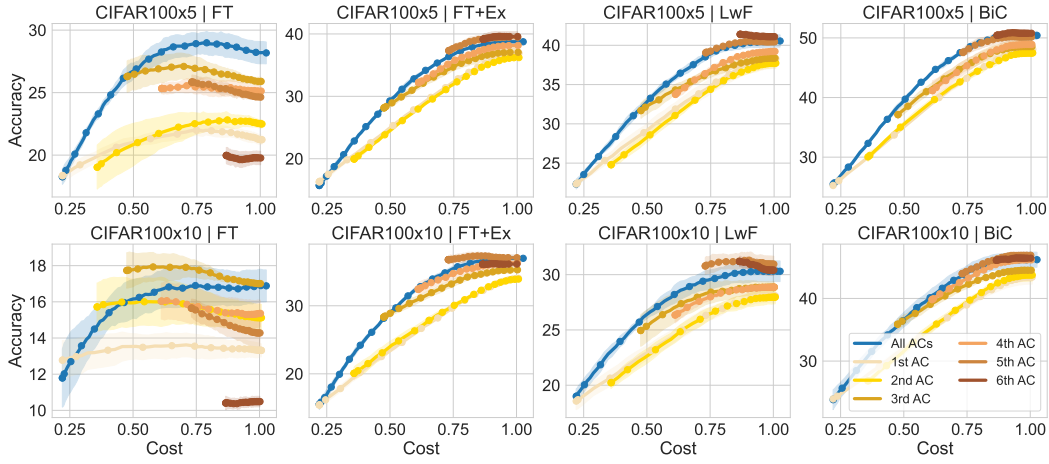


Figure 29: Leave one out AC ablation for FT, FT+Ex, LwF and BiC.