A METHOD

A.1 MEMORY FOOTPRINT FOR TEST-TIME ADAPTATION

When a learning algorithm looks for deployment on edge devices, it is unavoidable to take the two folds into account: (1) whether the maximal memory consumption of the algorithm can be fitted into a mobile device; (2) whether the dynamic and instantly-free space of device memory (RAM) is sufficient for the adaptation. The current mobile devices generally have memory in GB levels. For example, current general mobile devices such as Samsung S20 have 2GB or more memory, Raspberry Pi has 1GB or more memory. However, unlike the training or adaptation on a high-capacity GPU server where all the resources (memory and CPU time) can be allocated for training, the memory on mobile devices may be temporarily and partially allocated for the operating system and other background applications. The situation could be even more severe for mobile devices. Thus, squeezing the memory footprint dynamically is crucial for edge devices.

To avoid losing focus of the paper, we only consider the cache for gradient computation on batch-normalization layers, which has been a substantially-larger memory footprint compared to those of the widely-studied model and gradient memory in the literature (Yuan et al., 2021; Akin et al., 2019; Wang et al., 2022b). Admittedly, our work does not cover all memory consumption in the life-span of model adaptation. In Table 1, we only compute the memory costs (cache) of the back-propagation but not the forward operations, because the memory costs of the latter will be released immediately. The actual memory occupation in hardware, like NVIDIA GPU, will be enlarged due to the reservation for faster inference. The holistic solution that jointly optimizes the inference time and memory requires extra efforts on low-level computation and could be hardware-dependent. Thus, we leave the solution as a open problem for future study.

B EXPERIMENTAL DETAILS AND SUPPLEMENTARY

B.1 IMPLEMENTATION DETAILS

Hyper-parameters. All test-time adaptation objectives are optimized by stochastic gradient descent (SGD) with a momentum of 0.9. Tent and EATA utilize a batch size of 64 with a learning rate of 0.005 (0.00025) for CIFAR-10 (CIFAR100 and ImageNet). In our implementation, we use 0.0025 (0.0001) as learning rates to stabilize the training with smaller batch sizes. EATA uses 2,000 samples to estimate a Fisher matrix for anti-forgetting regularization. For MECTA, we set the threshold β_{th} for stopping layer training as 0.005 for CIFAR10/100 and 0.00125 for ImageNet-C. The cache pruning rate is set to be 0.7 for all datasets.

We implement our algorithm using PyTorch 1.12.1, cudatoolkit 11.6 on NVIDIA Tesla T4 GPUs. The codes of baselines are provided by the open sourced codes of EATA[1]. For gradient checkpointing, we use the official implementation from PyTorch, `torch.utils.checkpoint`. For each stage in the ResNet with m blocks, we will split the blocks sequentially into $\lfloor m/2 \rfloor$ segments. Therefore, ResNet50 will be split into 7 segments, approximatedly equal to $\sqrt{50}$.

Measuring cache sizes. Without GC, we measure the cache size by summing up the tensor size of all features z^l in a network. With GC, we estimate the cache size as two parts: one is the segment cache sizes and the other is the maximal cache for backwarding inside a segment.

Measuring full footprint. In Fig. 1, we track the tensors that are cached in the GPU memory using the public tool[2]. The memory tracker will find all the PyThorch tensor variables in the garbage collector of Python. We also leverage tool, `torch.cuda.memory_cached`, provided by PyTorch to estimate the maximal GPU memory costs including non-tensor variables.

[1] https://github.com/mr-eggplant/EATA

[2] https://github.com/Oldpan/Pytorch-Memory-Utils

Implementation of stochastic cache. During forward, we will only store the remained values of z^l and the indexes of the remained channels (denoted as R). Later, we compute the gradient by

$$\sum_{n=1}^{B} \frac{\partial \ell_n}{\partial \gamma_i^l} = \begin{cases} \sum_{n=1}^{B} \sum_{j=1}^{W} \sum_{k=1}^{H} \frac{\partial \ell_n}{\partial a_{i,j,k}^l} z_{n,i,j,k}^l, & i \in R, \\ 0, & i \notin R. \end{cases}$$

As the zero values do not need cache, the implementation can effectively reduce memory consumption with a small extra space for storing the index set R.

B.2 More experimental results

MECTA in different network architectures. It is known that larger and deeper models will merit the robustness of neural networks (Hendrycks et al., 2020a). In Fig. 4, we compare the EATA and EATA+MECTA on varying model depths. We adopt the protocol in Table 2 but reduce the batch size to 32 to accommodate the huge memory cost of deeper networks. The experiments are conducted with 4 perturbations. By increasing the depth of ResNet, the cache size increases steeply. Instead, MECTA reduces the cache consumption to a low level and achieves even better accuracy. Beyond ResNet, we evaluate more network architectures in Table 5. For example, MobileNet (Howard et al., 2017) is designed for edge devices with limited computation resources. Since our method only modify the batch-normalization layers, it can be easily plugged into these networks. Except wide ResNet (WRN), our method outperforms EATA with much lower cache sizes and higher accuracy.

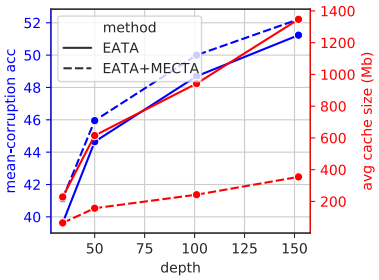

Figure 4: Evaluation on different ResNet models with varying depth.

Table 5: Evaluation on different model architectures retrived from PyTorch pre-trained models.

Architecture	Alg.	Acc. (%) Avg	Cache (Mb) Avg	Cache (Mb) Max
MobileNetV2 (Howard et al., 2017)	EATA	26.8	854.8	854.8
	EATA+MECTA	**28.0**	**233.5**	**250.5**
MobileNetV3 (Howard et al., 2017)	EATA	30.8	563.2	563.2
	EATA+MECTA	**32.2**	**158.1**	**163.7**
VGG19+BN (Simonyan & Zisserman, 2015)	EATA	34.9	1901.1	1901.1
	EATA+MECTA	**35.7**	**544.8**	**564.9**
WRN101×2 (Zagoruyko & Komodakis, 2017)	EATA	**54.7**	2716.7	2716.7
	EATA+MECTA	53.2	**803.2**	**810.4**
ResNet101	EATA	41.2	1885.0	1885.0
	EATA+MECTA	**51.3**	**449.1**	**569.8**
ResNet152	EATA	44.0	2694.3	2694.3
	EATA+MECTA	**52.9**	**635.6**	**813.6**
ResNeXt101 32×8d (Xie et al., 2017)	EATA	53.9	3802.1	3802.1
	EATA+MECTA	**54.5**	**1094.7**	**1136.1**
DenseNet121 (Huang et al., 2018)	EATA	**45.9**	2005.4	2005.4
	EATA+MECTA	42.2	**594.3**	**597.6**
EfficientNetV2-S (Tan & Le, 2021)	EATA	45.9	1556.3	1556.3
	EATA+MECTA	**47.1**	**454.4**	**463.3**

Does layer-sparse training help? In Fig. 5, we evaluate EATA with fewer trainable layers, which can reduce the cache size, following the protocol in Table 2. During adaptation, we keep k deepest layers to be trained and freeze other layers, which is denoted as Lk in the figure. We also include the EATA+GC where we use gradient checkpointing for the trainable layers. We observe that reducing the trainable layers can significantly decrease the cache size which is even lower than MECTA. However, the corresponding accuracy is significantly decreased by 5% compared to EATA+MECTA meanwhile. In comparison, though MECTA also uses the layer-sparse training, our method presents the best accuracy-memory trade-off in the experiment. The key difference is that our method sparsifies the training only on demand, specifically when a layer is well adapted without need for further training.

Does MECTA works on even smaller batches? In Table 6, we extend Table 1 with more batch sizes. One interesting observation is that our method outperforms or is comparable to other baselines given even smaller caches. For example, given a batch size of 16, EATA+MECTA can outperforms BN at the best batch size, when MECTA reduces the cache size to 71 Mb on average compared to the 134 Mb by BN.

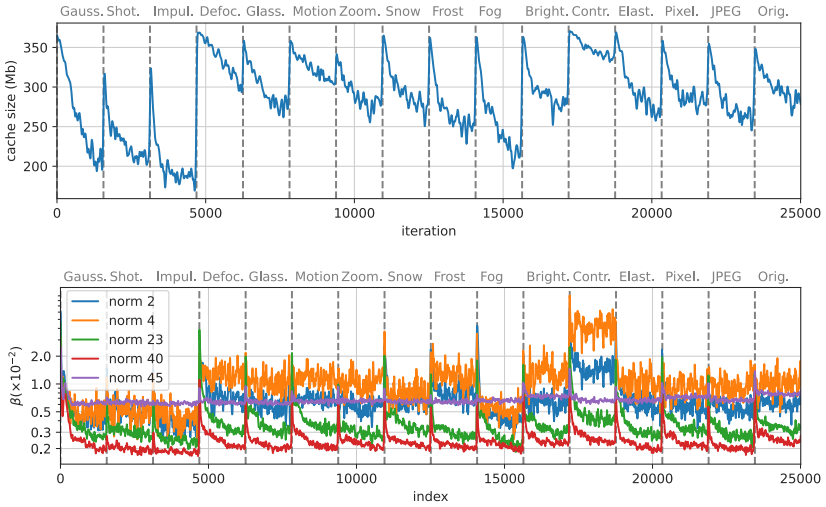

Figure 6: Dynamic cache size and β using MECTA on ImageNet-C.

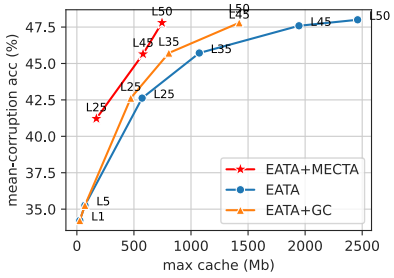

Figure 5: Compare MECTA to layer-sparse training using ResNet50 and batch size of 64. L1 means that only the deepest 1 layer is trained. Likewise, L45 denotes the deepest 45 layers.

Does MECTA-B works with BN adaptation and Tent? We show that MECTA-B can generally works well with BN adaptation and Tent, in Table 7. Consistent on all three backbone methods, the MECTA-B can outperforms EMA and base methods without extra hyper-parameters. EMA and MECTA-B can salvage BN and Tent from pooer performance using small batch sizes.

How does MECTA mitigate forgetting? Comparing the Tent+MECTA-B results with Tent+MECTA with 16-sized batches in Table 6, we notice that MECTA is more effective than MECTA-B on improving Tent. Thus, we conclude that the pruning and adaptive training is essential for the anti-forgetting.

More shift-accuracy evaluation. We consider more cases of (K, k) pairs in Table 8. In all three trials, MECTA-B reduce the shift-accuracy drops w.r.t. the baseline. Given more new-domain samples, e.g., $k = 5$, the shift-accuracy using EMA and MECTA-B becomes higher than the baseline, implying the quick convergence of the adaptation.

Cache size and β by iteration. We extend Fig. 3 to a full life-cycle version in Fig. 6. For evaluating β, we run the experiment on ImageNet-C using ResNet50 and we use MECTA with EATA for adaptation. In more corruptions, we find the periodic fluctuation of β by the distributional shifts, which results in the dynamic cache sizes.

Table 6: Continual evaluation on three datasets with the highest severity level 5 regarding accuracy (%). For a fair comparison, batch sizes (BS) are chosen such that the corresponding cache sizes are lower than those of BN with batch size of 128. Blue cells highlight the accuracy that is the highest among all methods, and the bold texts indicate the best accuracy given the same base algorithm.

		Noise			Blur				Weather				Digital					Acc.	Cache	
Alg.	BS	Gauss.	Shot.	Impul.	Defoc.	Glass.	Motion	Zoom.	Snow	Frost	Fog	Bright.	Contr.	Elast.	Pixel.	JPEG	Orig.	Avg	Avg	Max
CIFAR10-C																				
BN	32	80.2	82.0	78.7	91.2	79.2	89.9	91.0	86.9	86.6	84.5	91.9	88.8	84.2	86.8	81.8	93.2	86.1	34	34
BN	64	81.3	82.7	79.6	91.9	79.9	90.8	91.6	87.5	87.0	85.5	92.3	89.4	85.3	87.5	82.6	93.7	86.8	67	67
BN	128	**81.5**	**83.2**	**79.9**	**92.4**	**80.7**	**91.1**	**92.1**	**87.9**	**87.7**	**85.7**	**92.7**	**89.8**	**85.6**	**87.6**	**83.0**	**94.2**	**87.2**	134	134
Tent	8	68.8	46.3	16.4	12.8	11.8	8.9	9.9	10.7	10.3	10.2	10.3	10.2	10.1	10.0	10.2	10.2	16.7	100	100
	16	81.8	82.2	71.8	68.9	52.8	43.7	39.0	30.7	22.8	17.7	10.1	6.7	7.4	8.2	8.1	9.0	35.1	199	199
	32	86.0	87.5	84.1	88.8	81.2	85.5	86.6	85.5	86.6	83.5	88.5	87.0	83.0	85.7	80.7	87.2	85.5	398	398
+MECTA	16	86.7	88.3	85.1	90.0	79.9	85.7	88.2	85.4	86.4	83.5	88.7	88.0	83.0	86.2	79.6	86.2	85.7	41	60
+MECTA	32	**87.0**	**88.6**	**85.7**	**91.8**	**84.5**	**90.9**	**92.5**	**90.1**	**91.3**	**88.8**	**93.2**	**92.0**	**88.0**	**90.5**	**86.0**	**93.8**	**89.7**	62	119
EATA	8	74.5	70.9	66.6	69.7	55.5	54.6	47.2	38.4	33.1	35.1	35.2	18.8	12.2	14.5	13.3	8.5	40.5	100	100
	16	83.4	84.3	81.3	86.4	77.0	83.8	86.5	84.5	84.7	83.9	88.1	87.9	80.9	85.5	79.3	87.9	84.1	199	199
	32	85.7	87.6	85.7	89.8	82.5	88.3	90.0	88.2	88.8	88.7	91.4	91.0	85.5	89.5	85.3	90.6	88.0	398	398
+MECTA	16	86.5	88.2	84.6	89.8	83.0	88.5	90.4	88.4	88.9	88.8	91.7	91.1	85.8	89.5	85.3	92.3	88.3	45	60
+MECTA	32	**87.1**	**89.3**	**86.8**	**92.0**	**85.6**	**90.7**	**92.3**	**90.8**	**91.1**	**89.7**	**93.3**	**92.5**	**88.0**	**91.0**	**86.9**	**93.5**	**90.0**	71	119
CIFAR100-C																				
BN	32	56.2	57.1	54.7	70.5	56.3	68.6	70.2	63.0	63.5	56.6	71.9	68.0	62.2	64.8	56.7	73.8	63.4	34	34
BN	64	56.8	58.7	55.8	71.5	57.7	69.5	71.5	64.1	64.3	57.4	73.1	69.0	63.6	66.1	58.4	75.3	64.6	67	67
BN	128	**57.6**	**59.0**	**56.6**	**72.5**	**58.2**	**69.9**	**71.8**	**64.7**	**64.8**	**57.9**	**73.5**	**69.8**	**64.3**	**66.7**	**58.6**	**75.8**	**65.1**	134	134
Tent	8	52.9	53.9	46.7	50.5	31.2	29.7	23.7	14.3	10.0	6.7	5.9	3.4	3.9	3.6	3.5	3.5	21.5	100	100
	16	56.8	61.4	59.6	68.5	56.1	65.3	66.9	59.7	60.0	54.9	65.0	57.0	54.2	56.2	46.8	58.6	59.2	199	199
	32	58.5	**63.0**	**61.6**	71.8	60.3	69.7	71.8	65.1	66.3	60.9	72.4	68.7	64.7	67.8	59.7	74.3	66.0	398	398
+MECTA	16	58.4	**63.0**	**61.1**	73.3	60.7	71.3	73.4	66.2	66.4	61.0	73.5	67.7	65.5	68.2	59.3	75.5	66.5	50	60
+MECTA	32	**58.6**	61.5	60.1	**73.9**	**61.0**	**72.4**	**74.9**	**66.8**	**68.1**	**61.9**	**75.3**	**71.3**	**66.7**	**70.5**	**61.7**	**78.1**	**67.7**	77	120
EATA	8	52.1	54.2	53.2	65.3	51.5	63.8	64.9	59.1	58.5	53.9	66.8	63.2	56.3	61.1	52.9	70.1	59.2	100	100
	16	57.3	60.5	58.5	69.9	57.1	68.9	69.8	63.7	64.4	59.4	71.6	67.9	62.8	67.1	58.2	74.4	64.5	199	199
	32	58.4	**62.4**	**60.9**	72.1	59.5	70.3	72.4	66.3	66.5	**62.2**	74.4	70.8	65.3	69.4	61.0	76.9	66.8	398	398
+MECTA	16	58.4	61.5	59.3	**73.2**	59.8	**71.6**	73.1	**66.9**	66.9	60.8	74.4	70.9	65.0	68.8	60.6	77.2	66.8	52	60
+MECTA	32	**58.7**	**62.0**	**60.0**	73.0	**60.5**	71.3	**73.9**	66.7	**67.3**	**62.0**	**75.3**	**71.7**	**66.3**	**69.8**	**61.5**	**78.0**	**67.4**	81	120
ImageNet-C																				
BN	64	38.4	41.6	38.8	29.1	32.1	40.1	46.4	44.1	46.8	54.8	67.7	35.1	53.7	64.7	54.6	73.5	47.6	206	206
BN	128	**39.2**	**42.6**	**39.6**	**29.9**	**32.9**	**40.8**	**47.4**	**45.0**	**47.7**	**55.8**	**68.5**	**36.0**	**54.8**	**65.4**	**55.7**	**74.2**	**48.5**	411	411
Tent	8	33.8	16.5	0.8	0.4	0.3	0.4	0.4	0.4	0.4	0.3	0.4	0.2	0.3	0.4	0.3	0.4	3.5	307	307
	16	43.3	46.1	42.8	25.8	14.8	5.0	1.3	0.7	0.7	0.7	0.8	0.6	0.7	0.7	0.7	0.7	11.6	615	615
	32	46.1	50.6	49.2	37.2	38.2	41.2	42.5	36.6	36.2	38.8	46.8	27.5	30.5	34.3	26.9	34.8	38.6	1230	1230
+MECTA	16	**48.0**	**52.9**	**51.8**	**38.9**	42.0	46.5	49.5	44.0	43.9	47.7	56.6	38.3	44.1	49.5	43.2	52.7	46.8	158	187
+MECTA	32	47.0	52.3	51.7	38.5	**43.1**	**47.5**	**51.5**	**47.4**	**49.3**	**54.2**	**64.4**	**46.2**	**55.6**	**62.4**	**57.0**	**68.8**	**52.3**	269	373
EATA	8	34.1	37.0	35.0	27.5	28.1	35.5	38.6	39.6	39.7	47.8	56.6	35.5	44.1	53.3	46.7	63.2	41.4	307	307
	16	44.4	47.1	45.4	39.0	39.4	47.4	49.7	49.7	48.4	57.6	64.3	47.8	54.5	61.7	56.3	69.5	51.4	615	615
	32	49.0	52.3	51.1	44.4	45.2	52.3	55.1	54.0	52.7	61.3	67.6	52.7	58.8	65.3	60.4	72.3	55.9	1230	1230
+MECTA	16	48.6	51.7	49.7	42.9	44.4	51.6	55.4	54.5	53.4	62.5	70.0	51.8	60.1	67.6	61.7	**74.7**	56.3	162	187
+MECTA	32	**50.3**	**54.1**	**52.6**	**44.6**	**47.0**	**53.6**	**57.0**	**55.5**	**54.8**	**62.9**	**70.3**	**54.3**	**61.1**	**68.4**	**63.0**	74.6	**57.8**	288	373

Table 7: Ablation study of MECTA-B on ImageNet-C with the highest severity level 5 regarding accuracy (%) and a batch size of 16. Blue cells highlight the accuracy that is the highest among all methods, and the bold texts indicate the best accuracy among ablations of EMA and MECTA-B.

	EMA	MECTA-B	Noise			Blur				Weather				Digital					
Alg.	$\beta = 0.1$	auto β	Gauss.	Shot.	Impul.	Defoc.	Glass.	Motion	Zoom.	Snow	Frost	Fog	Bright.	Contr.	Elast.	Pixel.	JPEG	Orig.	Avg
BN	✗	✗	33.7	36.5	34.0	24.6	27.2	35.0	40.5	39.1	42.3	49.4	62.7	30.7	47.6	59.1	48.9	69.1	42.5
	✓	✗	36.4	40.4	36.4	28.6	31.8	40.1	47.0	43.2	46.4	53.9	67.8	32.3	53.8	64.9	55.2	74.2	47.0
	✓	✓	**37.8**	**41.9**	**37.7**	**29.3**	**32.7**	**40.6**	**47.9**	**44.0**	**47.2**	**55.0**	**68.6**	**32.9**	**55.0**	**65.8**	**55.9**	**74.8**	**47.9**
Tent	✗	✗	43.3	46.1	42.8	25.8	14.8	5.0	1.3	0.7	0.7	0.7	0.8	0.6	0.7	0.7	0.7	0.7	11.6
	✓	✗	48.5	52.6	51.0	37.8	36.4	38.6	38.3	29.7	25.8	23.0	24.9	6.1	6.6	5.3	1.6	2.3	26.8
	✓	✓	**50.0**	**53.9**	**52.2**	**38.7**	**38.3**	**40.9**	**41.3**	**33.8**	**31.4**	**30.6**	**34.8**	**12.6**	**14.5**	**13.0**	**7.0**	**9.2**	**31.4**
EATA	✗	✗	44.4	47.1	45.4	39.0	39.4	47.4	49.7	49.7	48.4	57.6	64.3	47.8	54.5	61.7	56.3	69.5	51.4
	✓	✗	50.4	53.1	51.9	45.2	45.9	53.9	56.3	55.6	54.2	63.1	69.6	53.2	60.5	67.2	62.0	74.0	57.3
	✓	✓	**51.7**	**54.8**	**53.6**	**46.0**	**47.5**	**54.9**	**57.7**	**56.9**	**55.3**	**64.2**	**70.5**	**54.2**	**61.6**	**68.1**	**63.3**	**74.9**	**58.4**

Table 8: Evaluation of k-new K-old shift accuracy by EATA. Average accuracy (AA $\%$) and worst accuracy (WA $\%$) are reported for each target perturbation. Values in the brackets denote the difference between the current method and the base method using batch statistics.

K	k	EMA	MECTA-B	Impul.		Motion		Fog		Elast.	
		$\beta = 0.1$	Auto β	AA	WA	AA	WA	AA	WA	AA	WA
49	1	✗	✗	35.5	34.6	37.0	36.8	50.1	49.8	48.3	47.8
		✓	✗	35.4 (**-0.1**)	30.6 (-4.0)	26.6 (-11.0)	20.1 (-16.7)	41.0 (-9.1)	25.9 (-23.9)	42.8 (**-5.5**)	39.3 (-8.5)
		✓	✓	34.4 (-1.1)	32.0 (**-2.6**)	28.6 (**-8.4**)	25.7 (**-11.1**)	43.6 (**-6.5**)	39.1 (**-10.7**)	42.6 (-5.7)	41.2 (**-6.6**)
9	1	✗	✗	34.1	33.4	35.7	35.6	49.4	48.9	47.4	47.1
		✓	✗	32.8 (-1.3)	27.0 (-6.4)	25.0 (-10.7)	17.3 (-18.3)	38.4 (-11.0)	23.9 (-25.0)	41.1 (-6.3)	35.7 (-11.4)
		✓	✓	34.1 (**0.0**)	32.3 (**-1.1**)	27.4 (**-8.3**)	23.1 (**-12.5**)	42.4 (**-7.0**)	36.5 (**-12.4**)	41.9 (**-5.5**)	40.0 (**-7.1**)
45	5	✗	✗	35.4	34.6	36.9	36.5	50.3	49.3	48.5	47.7
		✓	✗	38.4 (+3.0)	37.5 (+2.9)	35.4 (-1.5)	30.6 (-5.9)	53.3 (+3.0)	50.2 (0.9)	52.0 (+3.5)	50.8 (+3.1)
		✓	✓	37.8 (+2.4)	37.2 (+2.6)	35.4 (-1.5)	33.0 (**-3.5**)	51.5 (+1.2)	49.6 (+0.3)	50.8 (+2.3)	49.6 (+1.9)