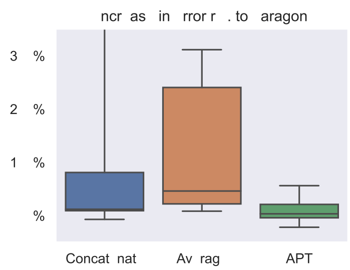
À-la-carte Prompt Tuning (APT): Combining Distinct Data Via Composable Prompting

Benjamin Bowman[1,2]* Alessandro Achille[1] Luca Zancato[1] Matthew Trager[1]

Figure 1. **À-la-carte Learning and APT.** Given a pool of multiple data sources, the goal of À-la-carte Learning is to allow the user to select – at inference time – an arbitrary subset S D of sources to use. The performance of the à-la-carte model should be comparable to the performance of a model trained on S. **(A)** APT enables efficient À-la-carte Learning by converting each source into a prompt, and composing together the relevant prompts at inference time. **(B)** To perform inference, APT uses a modified attention mechanism that



Dataset	Concatenate	Average	APT	Paragon
MIT-67	84.6%	85.1%	**86.2%**	86.2%
Cub-200	85.2%	84.6%	**87.8%**	86.6%
Caltech-256	**91.1%**	87.9%	**91.1%**	91.7%
Pets	**93.8%**	91.4%	93.1%	93.3%
Aircrafts	56.5%	16.7%	**61.1%**	71.0%
Flowers	84.5%	96.3%	**99.3%**	99.1%
Stanford Cars	60.3%	26.1%	**70.7%**	81.2%

Figure 2. **Naive prompt composition vs. APT.** We compare different methods of combining prompts. We split the training dataset into two equal sized shards then train prompts on each of the two shards in isolation. We then compare the test accuracies after combining the prompts using different methods. For the column “Concat” we concatenate the prompts without structured attention and average ensemble their predictions. For the column “Avg” we simply average the prompts and classifier head as parameters and then take the single prediction. The column “APT” denotes our method. Numbers more than 10% below APT in each row are marked red; numbers more than 2% below APT are marked orange. The best method excludingarag8 w 0iag8 w 058(each)0259(ro)25(w) 0isare marked97.955 Tdoptim3(sge.57259(sofncat“58201(pro

prompting can also be used as an alternative adaptation mechanism for vision transformers [15, 35]. Let D be a supervised dataset for a downstream task. A new learnable *prompt token* p_0 is attached to the transformer's input, so that the final output is given by

$$[z_L; p_L] = F^L \circ \cdots \circ F^1([z_0; p_0]).$$

To predict the downstream task label, the head of the pre-trained model is discarded to be replaced by a new head which is trained on the final prompt token

$$\hat{y} = \mathrm{softmax}(\mathrm{head}(p_L)).$$

Both p_0 and head are trained on D, while the parameters of the pre-trained backbone are frozen.

Notation. We denote with $\ell(\hat{y}, y)$ the cross entropy loss, and for a natural number $k \in \mathbb{N}$ we let $[k] :=$ fTJ/F11 9.926 Tf 8.07 0 1Td [(0)]TJ/F11 9.926f 10.282
Wadapt t1ngwhere-166(:)]TJ/F59 9.96262 Tf 45.979 0 Td [(N)]TJ/F95 9.9621.71 f 0.874 0V

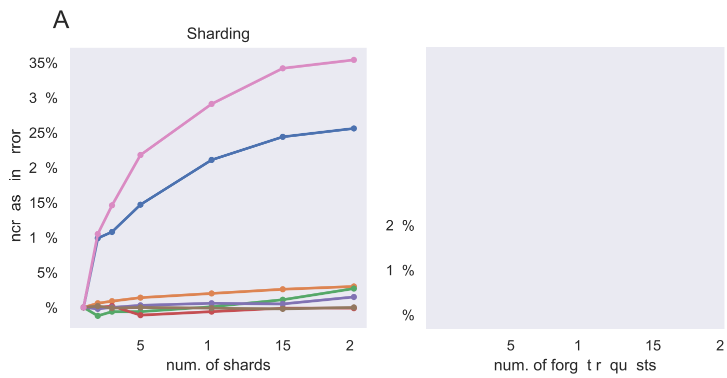


Figure 3. **(A) Error increase of APT compared to paragon.** We split a training set into a varying number of equal sized shards chosen uniformly at random. We then use APT to combine prompts learned individually on each shard, and measure the increase in error compared to the paragon of training on all data together. For most datasets, the performance of the APT is within a few percent of the paragon, even when the dataset is split in up to 20 parts. *Aircrafts* and *Stanford Cars* are the main exceptions, possibly due to the large domain shift between the backbone pretraining and those tasks. **(B) Satisfying forgetting requests.**

Dataset	

Method	CIFAR-100	CORe50
APT	83.63	90.89
APT-W	**85.21**	**91.14**
L2P [8]	83.83	78.33
S-iPrompts [7]	N/A	83.13
S-liPrompts [7]	N/A	89.06
LwF [22]	60.69	75.45
EWC [6]	47.01	74.82

References

[1] Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 141–159, 2021. 3, 6, 7, 11

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 2

[3] Ekin Dogus Cubuk, Barret Zoph, Jon Shlens, and Quoc Le. Randaugment: Practical automated data augmentation with a reduced search space. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18613–18624. Curran Associates, Inc., 2020. 11

[4] 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 *International Conference on Learning Representations*, 2021. 3, 6

[5] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9285–9295, June 2022. 3

[6] Kirkpatrick et al. Overcoming catastrophic forgetting in neural networks. *PNAS*, 114, 12 2016. 8

[7] Yabin Wang et al. S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning. In *NeurIPS*, 2022. 8

[8] Zifeng Wang et al. Learning to prompt for continual learning. In *CVPR*, 2022. 8

[9] Aditya Golatkar, Alessandro Achille, Avinash Ravichandran, Marzia Polito, and Stefano Soatto. Mixed-privacy forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 792–801, June 2021. 3

[10] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 3

[11] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. In *European Conference on Computer Vision*, pages 383–398. Springer, 2020. 3

[19R.

Learning, volume 78 of *Proceedings of Machine Learning Research*, pages 17–26. PMLR, 13–15 Nov 2017. 6, 12

[26] Vincenzo Lomonaco, Davide Maltoni, and Lorenzo Pellegrini. Fine-grained continual learning. *ArXiv*, abs/1907.03799, 2019. 6, 12

[27] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 11

[28] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi.

Supplementary Material

A. Details of APT Weight (APT-W)

In this section we describe the details of the APT Weight (APT-W) scheme. Let $D = \{D_1, \ldots, D_n\}$ be a collection of sources. Consistent with APT for each source D_i we train a prompt $p^{(i)}$ and a classifier head head_i using only the data in D_i. Then, differing with classical APT, for each source D_i we perform K-means ($K = 20$) in the embedding space to construct a set of prototypes ${}^{(i)}_{1}, \ldots, {}^{(i)}_{K}$. More concretely for each $(x, y) \in D_i$ we forward the input x through the transformer to get the final embedding sequence $[\mathbf{z}_L(x)]$

Table 5. **Dataset sample/class counts.** We list the number of training images, test images, and classes for each of the datasets. We also provide a link to download the data.

Dataset	Training Images	Testing Images	# Classes	URL
MIT-67 [33]	5360	1340	67	https://web.mit.edu/torralba/www/indoor.html
CUB-200 [36]	5994	5794	200	https://www.vision.caltech.edu/datasets/cub_200_2011/
Caltech-256 [13]	15418	15189	257	https://authors.library.caltech.edu/7694/
Oxford Pets [31]	3680	3669	37	https://www.robots.ox.ac.uk/~vgg/data/pets/
FGVC-Aircrafts [28]	6667	3333	100	https://www.robots.ox.ac.uk/~vgg/data/fgvc-aircraft/
Oxford Flowers [30]	2040	6149	102	https://www.robots.ox.ac.uk/~vgg/data/flowers/102/
Stanford Cars [16]	8144	8041	196	https://ai.stanford.edu/~jkrause/cars/car_dataset.html
CIFAR-100 [17]	50,000	10,000	100	https://www.cs.toronto.edu/~kriz/cifar.html
CORe50 [25, 26]	119,894	44,972	50	https://vlomonaco.github.io/core50/

Dataset	2 Shards	3 Shards	5 Shards	10 Shards	15 Shards	20 Shards	50 Shards
MIT-67	3.1%	0.9%	0.6%	0.7%	0.6%	0.9%	0.5%
Cub-200	3.3%	1.1%	1.5%	0.8%	1.0%	0.8%	1.3%
Caltech-256	3.0%	1.4%	0.8%	0.2%	0.6%	0.4%	0.2%
Pets	1.4%	0.2%	0.3%	0.0%	-0.1%	0.2%	0.3%
Aircrafts	6.2%	5.2%	5.0%	3.8%	3.2%	3.7%	3.0%
Flowers	0.7%	4.2%	0.2%	0.6%	0.7%	1.3%	2.8%
Stanford Cars	9.0%	7.6%	5.6%	5.5%	4.9%	5.1%	4.7%
Average	3.81%	2.94%	2.0%	1.66%	1.56%	1.77%	1.83%

Table 6. **Average vs. majority vote.** We report the accuracy of average ensembling minus the accuracy of majority vote. We observe that average ensembling uniformly outperforms majority vote.

backbone transformer has a different pretraining, instead of using ImageNet21k we experiment with loading the VIT-B/16 from the visual encoder of the multimodal model ALBEF [20]. In Table 7 we report the accuracies of APT applied to this checkpoint. We see that the performance of APT for the ALBEF visual encoder decays more quickly as the number of shards increases relative to the ImageNet21k numbers reported in Table 2. For example for the visual encoder of ALBEF, for 10 shards only the datasets MIT-67, Caltech-256, and Pets are within 5% performance of the paragon, whereas by contrast for the ImageNet21k checkpoint all datasets except for Aircrafts and Stanford Cars are within 5% performance of paragon even when the number of shards is twice as large, namely 20. Thus we conclude that the pretraining of the backbone transformer is highly pertinent for the performance of APT. This is sensible as due to the structured attention the APT prompts do not modify the internal representations of the backbone, and thus are unable to provide compensation whenever the backbone representations are deficient.

Finetuning. While inference and storage for the APT method is less costly than ensembling finetuned models, it is worthwhile to ask how the two compare in terms of classification accuracy. In Table 8 we report the accuracies for the sharding experiment using finetuning instead of APT. Specifically we finetune separate models on each shard which are then ensembled at inference time. By comparing the results in Table 2 to the results in Table 8, we see that APT uniformly outperforms finetuning in terms of classification accuracy, and the gap becomes most pronounced as the number of shards increases. Specifically, for 20 and 50 shards APT has average accuracy of 77.3% and 73.9% respectively compared to 41.5% and 25.6% for finetuning. We believe this is due to finetuning being more susceptible to overfitting when there are fewer data in contrast to APT which uses a fixed backbone and thus has a stronger inductive bias.

Dataset	No Sharding	2 Shards	3 Shards	5 Shards	10 Shards	15 Shards	20 Shards	50 Shards
MIT-67	89.1%	87.5%	88.4%	88.9%	89.0%	88.4%	88.4%	87.3%
Cub-200	78.8%	71.6%	66.9%	57.1%	54.9%	48.5%	46.2%	39.6%
Caltech-256	91.4%	88.8%	89.2%	88.7%	88.9%	88.6%	87.8%	86.2%
Pets	91.1%	89.9%	88.2%	87.0%	86.3%	83.1%	81.0%	66.1%
Aircrafts	72.6%	60.5%	54.4%	51.1%	40.2%	38.7%	35.9%	30.7%
Flowers	93.4%	85.1%	83.1%	81.7%	80.7%	78.1%	76.2%	67.8%
Stanford Cars	83.3%	78.2%	76.5%	70.7%	63.7%	59.5%	55.9%	43.6%
Average	85.67%	80.23%	78.1%	75.03%	71.96%	69.27%	67.34%	60.19%

Table 7. **Sharding from ALBEF pretraining.** We report the accuracies for the sharding experiment using the ALBEF checkpoint.

Dataset	No Sharding	2 Shards	3 Shards	5 Shards	10 Shards	15 Shards	20 Shards	50 Shards
MIT-67	87.1%	86.1%	83.8%	81.9%	74.4%	69.9%	68.8%	44.9%
Cub-200	88.4%	81.8%	76.4%	70.9%	54.4%	42.5%	32.5%	5.9%
Caltech-256	93.5%	90.3%	87.8%	85.8%	81.0%	78.2%	74.1%	52.0%
Pets	94.5%	93.6%	92.6%	91.2%	89.7%	84.2%	81.6%	55.8%
Aircrafts	75.6%	51.1%	44.5%	36.2%	24.1%	23.0%	19.2%	12.3%
Flowers	97.4%	75.3%	56.1%	39.8%	15.6%	11.1%	2.2%	2.0%
Stanford Cars	84.3%	53.3%	39.4%	28.2%	19.2%	16.2%	11.8%	6.4%
Average	88.69%	75.93%	68.66%	62.00%	51.20%	46.44%	41.46%	25.61%

Table 8. **Sharding using finetuning.** We report the accuracy for the sharding experiment when using finetuning.