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

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]
Pramuditha Perera[1] Giovanni Paolini[1] Stefano Soatto[1]
AWS AI Labs[1] UCLA[2]
benbowman314@math.ucla.edu zancato@amazon.it
{aachille,mttrager,pramudi,paoling,soattos}@amazon.com

Abstract

We introduce À-la-carte Prompt Tuning (APT), a transformer-based scheme to tune prompts on distinct data so that they can be arbitrarily composed at inference time. The individual prompts can be trained in isolation, possibly on different devices, at different times, and on different distributions or domains. Furthermore each prompt only contains information about the subset of data it was exposed to during training. During inference, models can be assembled based on arbitrary selections of data sources, which we call à-la-carte learning. À-la-carte learning enables constructing bespoke models specific to each user's individual access rights and preferences. We can add or remove information from the model by simply adding or removing the corresponding prompts without retraining from scratch. We demonstrate that à-la-carte built models achieve accuracy within 5% *of models trained on the union of the respective sources, with comparable cost in terms of training and inference time. For the continual learning benchmarks Split CIFAR-100 and CORe50, we achieve state-of-the-art performance.*

1. Introduction

As large neural network models make their way into commercial applications, the basic paradigm of training them on a monolithic dataset leads to a number of challenges. First, as new data become available, updating the whole model can be prohibitively expensive. Even when training time is not an issue, some users may still require access and maintenance of previous versions of the model to avoid disruptions of their downstream workflows. Second, owners of the training data may modify their sharing preferences at any time, leading to datasets that shrink over time (machine unlearning) or to different subsets of the training data being usable by different users (compartmentalization). Finally, the users themselves may want to use custom subsets of the data to better tailor their model to their use cases (model customization).

These challenges are well known and addressed separately in different fields such as continual learning, forgetting, and model adaption. However, in order for a commercial system to be viable at scale, these issues have to be tackled concurrently. Ideally, one would have a large model that each user can run, trained using only data the specific user wants and has rights to, that can evolve without the need for fine-tuning as new data becomes available, or as individual data owners exercise their right to have their data erased ("the right to be forgotten").

We refer to the problem of building such a model as *à-la-carte learning* since, depending on the data availability and the user, the service may need to select and use different data chunks from a menu of available training data. More specifically, let $\mathcal{D} = \{D_1, \dots, D_n\}$ be a variable collection of data sources (a *data pool*). In à-la-carte learning a user at inference time can specify a subset $S \subset \mathcal{D}$ of training data together with an input sample x to receive a personalized *à-la-carte* output $f(x, S)$ from the model f. Critically, the output $f(x, S)$ must not depend on any data source $D_i \notin S$.

À-la-carte learning can be naïvely tackled in two ways. The service could pre-train one model for each possible subset of the data pool, and serve each user the most powerful model they have rights to. While optimal from the user view-point, this requires a prohibitive exponential complexity $O(2^{|\mathcal{D}|})$ in both training time and storage. On the other extreme, the service could train a separate model on each data source individually and, at inference time, ensemble all models obtained from the sources in S. This requires only linear $O(|\mathcal{D}|)$ training time complexity to pre-train each model, but still has a significant storage cost. Furthermore due to the ensembling inference time is signifi-

*Work done during an internship at AWS AI Labs.

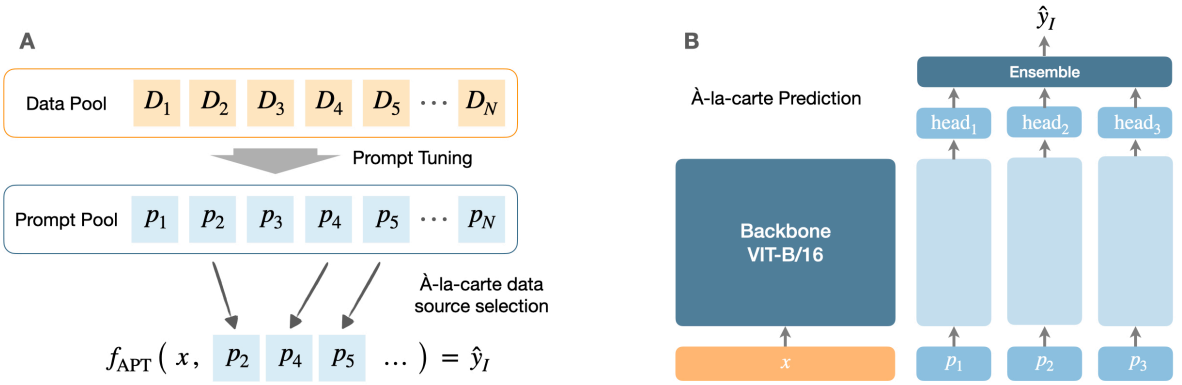


Figure 1. **À-la-carte Learning and APT.** Given a pool of multiple data sources, the goal of À-la-carte Learning is to allow the user to select – at inference time – an arbitrary subset $S \subset \mathcal{D}$ of sources to use. The performance of the à-la-carte model should be comparable to the performance 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 prevents the prompts from interfering with each other and ensembles the individual outputs to construct the final prediction.

cantly increased while also potentially suffering from lower performance than the ideal "paragon" model trained on the union of sources in S. The goal of à-la-carte learning is to achieve performance as close as possible to the paragon without significantly increasing inference or training time.

To address these key issues, we propose **À-la-carte Prompt Tuning (APT)**. APT leverages vision transformers and prompt tuning to solve the à-la-carte learning problem. First, APT converts each dataset D_i into a learned prompt p_i, thus transforming the data pool into a *prompt pool*. Then at inference time, given a subset of sources S to use, APT retrieves all corresponding prompts and concatenates them together with the input. Surprisingly, we show that in most cases APT has performance comparable to the paragon of joint learning with all data in S. Moreover, since each prompt is trained on an individual dataset, information is naturally compartmentalized. Thanks to the small size of prompts and an efficient forwarding method, APT is significantly cheaper (in both storage and inference time) than ensembling models.

Importantly however, we note that simply concatenating different prompts that were trained separately leads to destructive interference in the attention block which corrupts the representations (see Table 2). To address this problem, we introduce a modified attention mechanism that eliminates such interference, while also significantly reducing the inference time when multiple prompts are concatenated. A priori, this change comes with a small reduction in expressive power and in the ability to capture synergistic information between data sources. However, one of our main contributions is to show that the resulting drop in accuracy is generally modest, while providing far more valuable benefits to scalability, maintainability, and privacy.

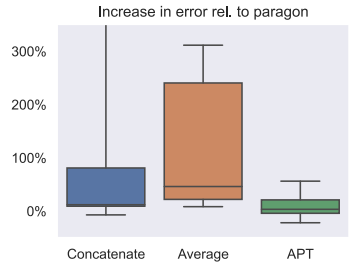
We empirically demonstrate the advantage of APT-based à-la-carte learning for forgetting and continual learning (both domain-incremental and class-incremental). We observe that in most cases the performance of APT is within 5% of the performance of the paragon at a fraction of the cost. We also show that APT outperforms all comparable baselines with the advantage of computational scalability from the structured attention mechanism.

Summary of our contributions.

1. We introduce the À-la-carte Learning problem to address continual learning, machine unlearning, and model customization concurrently.
2. We propose APT, an efficient method to address À-la-carte Learning based on visual prompt tuning and a modified attention mechanism.
3. We demonstrate that for most tasks APT achieves accuracy within 5% of paragon performance even when each individual prompt has access to an order of magnitude less data
4. We show that APT with a simple prompt weighting mechanism achieves state-of-the-art performance on continual learning benchmarks Split CIFAR-100 and CORe50.

2. Related Work

Prompt Tuning. Prompting originated from natural language processing by prepending "hard" language prompts to inputs to inform a pre-trained language model about the task to be solved [2,23]. It was then discovered that one can



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 excluding the paragon in each row is marked in bold.

optimize "soft" prompts in the embedding space in a differentiable fashion, with competitive performance to fine-tuning [18, 21, 24]. This technique also proved useful when applied to vision transformers [15]. The idea of extending pre-trained transformers using prompt tokens with attention masking was introduced in [35]. We use the same attention masking scheme in our à-la-carte learning implementation. The ensembling of soft and hard prompts was considered in [18] and [34] respectively.

Continual Learning. Prompt tuning applied to the continual learning problem has been considered in [5, 37, 38]. [5] augment a fixed backbone with small task tokens that can be trained during episodes and added to the model incrementally. In [38] they query collections of prompts from a prompt pool on an instance-wise basis to be concatenated at inference time. The query mechanism is supervised and consequently the compositionality of prompts is emergent from the supervision. By contrast, we select prompts from a pool on a per-user basis and achieve composability of prompts through structured attention. In [37] they address the domain incremental learning problem by training prompts independently on each domain and constructing a set of reference prototypes for the domain via K-means. At inference time, given an input x they select the prompt according to the closest reference prototype to the embedding of the point x. In our APT Weight (APT-W) scheme (described in Sec. 6) rather than select a single prompt we weight the prompts according to the instance embedding's distance to the closest prototype.

Forgetting. Forgetting in deep networks [10, 11] is challenging. [9] utilizes a ResNet-50 where they train a linearization of the network starting from a pre-trained checkpoint. Due to the linear parameterization, forgetting is much more tractable and they can get a bound on the mutual information after a certain number of forgetting steps. [14] offers forgetting for linear/logistic models, and [29] offer forgetting techniques in the convex setting. [1] investigated training distinct networks on separate shards of data. We run this same procedure to benchmark our APT approach. The novelty with the prompt tuning approach is that the memory overhead is minimal, and inference can be done at the cost of a single forward pass.

3. Preliminaries

Vision Transformer. We use vision transformers [4] as our backbone architecture, due to both good accuracy on downstream tasks and ease of prompting. An image $x \in \mathbb{R}^{H\times W\times C}$ is split into N patches $x^{(1)}, \ldots, x^{(N)}$, which are represented as d-dimensional tokens $z^{(i)} = Ex^{(i)} + e^{(i)}_{\text{pos}} \in \mathbb{R}^d$ through a learned linear embedding E and a set of positional encodings $\{e^{(i)}_{\text{pos}}\}_{i=1}^N$. We add a special learnable *class* token $z^{(0)}$ that is shared by all inputs. The input to the first layer of the transformer is then given by $\mathbf{z}_0 := [z^{(0)}, z^{(1)}, \ldots, z^{(N)}]$ which is the concatenation of the class token and the tokens corresponding to the image patches. Let F^ℓ_θ denote the ℓ^{th} attention layer of the transformer, where θ denotes the parameters of the model. The output tokens of the ℓ^{th} layer are given by

$$\mathbf{z}_\ell := F^\ell_\theta(\mathbf{z}_{\ell-1}).$$

Let $z^{(0)}_L$ be the output of the class token at the last transformer layer. We use a linear head to output a probability distribution $\hat{y}$ of the input's label:

$$\hat{y} := \text{softmax}(\text{head}_\theta(z^{(0)}_L))$$

where $\text{head}_\theta(x) = Wx + b$ is a learned fully connected layer.

Visual Prompting. Like convolutional networks, pre-trained vision transformers can be adapted to new downstream tasks by fine-tuning their weights θ. However,

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

$$[\mathbf{z}_L, p_L] = F_\theta^L \circ \ldots \circ F_\theta^1([\mathbf{z}_0, p_0]).$$

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 θ 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 $[k] := \{1, \ldots, k\}$. We consider a classification task where $\mathcal{X}$ is the input domain and $\mathcal{Y}$ is the label space.

4. À-la-carte Prompt Tuning

Suppose we have a pre-trained backbone f_θ and a pool of additional data sources $\mathcal{D} := \{D_1, \ldots, D_n\}$. We focus in particular on the case where all sources in $\mathcal{D}$ pertain to the same task and share the input and label space $D_i \subset \mathcal{X} \times \mathcal{Y}$.[1] Ideally we would like to fine-tune the backbone using all data in $\mathcal{D}$ by minimizing the loss:

$$L_{\mathcal{D}}(\theta) = \sum_{(x,y) \in \bigcup \mathcal{D}} \ell(f_\theta(x), ya).$$

However, it is often the case (see Section 5) that the collection of data sources $\mathcal{D}$ changes over time as data is added or removed. It may also be the case that different users of the model may want to use different subsets of the data to better cover their use cases (model customization) or may only have access rights to certain subsets of the data (compartmentalization).

À-la-carte Learning. To remedy this, *at inference time*, given any subset $I \subset [n]$ we would like to be able to use a model that uses data exclusively from $\mathcal{D}_I := \bigcup_{i \in I} D_i$. A trivial option is to fine-tune in advance the parameters θ_I on each possible subset I minimizing the loss

$$L_{\mathcal{D}_I}(\theta_I) := \sum_{(x,y) \in \mathcal{D}_I} \ell(f(x; \theta_I), y)$$

and, given I at inference, select the corresponding θ_I and use it to form the model $f(x; \theta_I)$. However, since there are 2^n possible subsets $I \subset [n]$ it is prohibitively expensive to fine-tune a separate model for each I, both from a compute-time and storage cost perspective. It would also require training 2^n new models each time a source of data is added, which becomes infeasible quickly.

Naïve À-la-carte Prompt Tuning. To reduce the computational cost while satisfying all requirements of À-la-carte Learning, we suggest an alternative strategy based on composition of prompts trained on individual data sources. For each $i \in [n]$ we train a prompt p_i and classifier head head_i on the data D_i using the loss function

$$L_{D_i}(p^{(i)}, \mathrm{head}_i) := \sum_{(x,y) \in D_i} \ell(f(x; p^{(i)}), y)$$

where the dependence of $f(x; p^{(i)})$ on head_i above has been suppressed for ease of notation. Given a set of indices $I = \{i_1, \ldots, i_{|I|}\}$ we denote with $\mathbf{p}^{(I)} = [p^{(i_1)}, \ldots, p^{(i_{|I|})}]$ the concatenation of all prompt tokens corresponding to each data source in $\mathcal{D}_I$. The final output of the transformer is given by

$$[\mathbf{z}_L, \mathbf{p}_L^{(I)}] := F_\theta^L \circ \ldots \circ F_\theta^1([\mathbf{z}_0, \mathbf{p}^{(I)}])$$

where θ are the frozen parameters of the backbone transformer. Each output token $p_L^{(i)}$ corresponding to a prompt $p^{(i)}$ can be used to generate a prediction

$$\hat{y}^{(i)} := \mathrm{softmax}(\mathrm{head}_i(p_L^{(i)})).$$

The final prediction is made by ensembling the predictions made by each individual prompt $p^{(i)}$ (see also Figure 1):

$$\hat{y}_I := \frac{1}{|I|} \sum_{i \in I} \hat{y}^{(i)}.$$

Since each prompt only contains information about its own source, the model output $\hat{y}_I$ depends only on the sources in $\mathcal{D}_I$. Moreover, after the initial cost $O(|\mathcal{D}|)$ to train each prompt p_i, any subset I of sources can be combined at inference time with constant cost $O(1)$. Hence, this procedure satisfies the requirements for à-la-carte learning.

However, in Figure 2 we see that the performance of this naïve implementation of à-la-carte prompt tuning by concatenating prompts severely underperforms the paragon of using a single prompt trained from scratch on all the datasets in $\mathcal{D}_I$. The same is true for other composition mechanisms, such as averaging prompts. We hypothesise that this is due to the prompts, which were trained individually, corrupting the representations at inference time when concatenated due to destructive interference in the attention mechanism of the transformer.

Structured Attention. To remedy this we follow the technique in [35]. First, we mask the attention so that the $\mathbf{z}_\ell$

[1] We do not however need to assume that all sources contain samples from all classes. The backbone f_θ may be pre-trained on the same task as $\mathcal{D}$ (in which case $\mathcal{D}$ provides additional data to tune the model) or may be pre-trained on an unrelated proxy task (e.g., ImageNet or web-scale data).

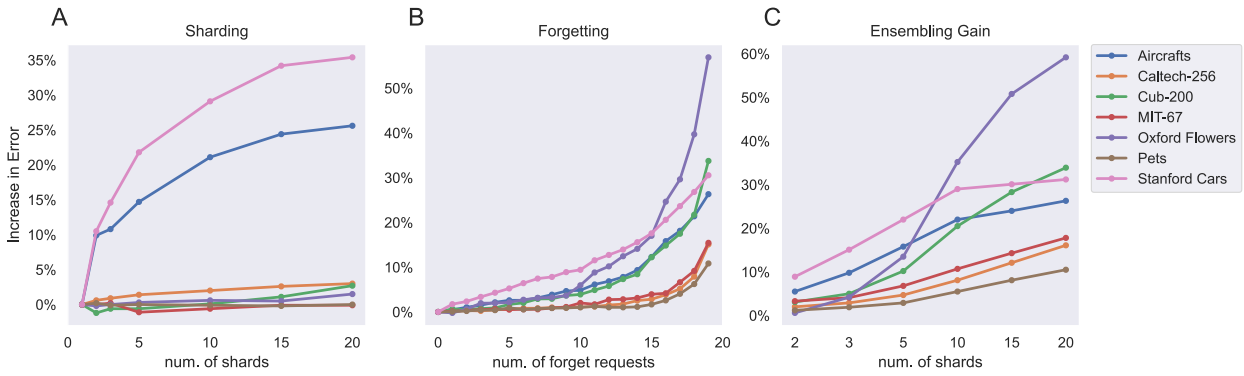


Figure 3. **(A) Error increase of APT compared to paragon.** We split a training set into a varying number of equal sized shards chosen uniformly at random. We then use APT to combine prompts learned individually on each shard, and measure the increase in error compared to the paragon of training on 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.** We simulate a sequence of data removal requests starting from a pool of 20 sources and removing one source at the time. We report the increase in error compared to using the full data. We see that APT degrades gracefully as desired, while also ensuring perfect data removal. **(C) Gain of using ensembles instead of individual prompts.** We split a train set in a varying number of shards, and show the difference between the accuracy of APT prompt composition and the average accuracy of the individual prompts. For large number of shards, individual prompts don't have enough information to classify accurately but APT can combine them to create a much stronger classifier (with up to 60% better accuracy).

tokens do not attend to the prompts, and the prompts do not attend each other (see Figure 4). This ensures the result of forwarding each prompt $p^{(i)}$ through the network is unaffected by the presence of the other prompts. However, this reduces the power of the prompts to modify the forward pass of the network. To compensate, at each layer ℓ of the transformer and for each prompt $p^{(i)}$ we add a set of d_{mem} learnable memory tokens $\mathbf{m}_\ell^{(i)} \in \mathbb{R}^{d_{\text{mem}} \times d}$. These memory tokens can be attended by the prompts but cannot attend to anything. While a similar result could be obtained by using longer prompts instead of using memory tokens, [35] notes that this solution gives comparable accuracy with a significantly reduced inference time. Due to the structured attention, a single forward pass of the backbone transformer can be performed on $\mathbf{z}_0$ independent of the prompts. Subsequently at each layer l each prompt $p_\ell^{(i)}$ can perform cross attention to query itself and $[\mathbf{z}_\ell, \mathbf{m}_\ell^{(i)}]$. While self-attention has quadratic complexity in the sequence length, this implementation has $O(N^2 + (N + d_{mem})|I|)$ complexity as opposed to $O((N + |I|)^2)$ complexity for self-attention without memory. Consistent with [35] in our implementation we set $d_{mem} = 5$. Consequently $N^2 \gg (N + d_{mem})$, and thus adding a prompt, and thus increasing $|I|$, only marginally increases inference time relative to the fixed cost of a forward pass for the backbone transformer $O(N^2)$. By contrast classic model ensembling would have inference cost $O(|I|N^2)$ as one must do a forward pass through each model. Furthermore each prompt corresponds to $12 \times d_{mem} + 1$ tokens, which amounts to a number of parameters less than .06% of the backbone model. Thus the memory overhead of storing the prompts is also marginal.

	$\mathbf{z}_\ell$	$p_\ell^{(1)}$	$p_\ell^{(2)}$	$p_\ell^{(3)}$	$\mathbf{m}_\ell^{(1)}$	$\mathbf{m}_\ell^{(2)}$	$\mathbf{m}_\ell^{(3)}$
$\mathbf{z}_\ell$	✓	✗	✗	✗	✗	✗	✗
$p_\ell^{(1)}$	✓	✓	✗	✗	✓	✗	✗
$p_\ell^{(2)}$	✓	✗	✓	✗	✗	✓	✗
$p_\ell^{(3)}$	✓	✗	✗	✓	✗	✗	✓

Figure 4. **Attention Masking Table.** The rows correspond to queries and the columns correspond to keys. The cells marked with ✓ denote where attention is performed and the cells marked ✗ denote where attention is masked.

À-la-carte Prompt Tuning. Our final proposal for efficient À-la-carte Learning, which we call À-la-carte Prompt Tuning (APT), combines the composition of individual prompts with the structured attention mechanism. In Figure 2 we see that APT outperforms the naïve baselines in almost all cases, and importantly it not prone to the same catastrophic failures (e.g. on Aircrafts and Stanford Cars). Moreover its performance is close or better[2] than the paragon performance (training a prompt directly on the union of all datasets) on all datasets except Aircrafts and Stanford Cars. In the following, we explore particularly interesting applications of À-la-carte learning, and we empirically test the performance of APT in different settings.

[2]The results better than the paragon can be attributed to the regularization effect of ensembling prompts trained on different subsets of the data.

Dataset	No Sharding	2 Shards	3 Shards	5 Shards	10 Shards	15 Shards	20 Shards
Head-only (in-domain)	90.8%	90.8%	90.8%	90.4%	90.1%	89.5%	88.5%
APT (in-domain)	**91.4%**	**91.5%**	**91.3%**	**91.4%**	**91.0%**	**90.6%**	**90.0%**
Head-only (out-of-domain)	59.7%	56.5%	53.5%	50.5%	45.0%	41.6%	40.5%
APT (out-of-domain)	**76.1%**	**65.9%**	**63.4%**	**57.9%**	**51.0%**	**46.8%**	**45.6%**

Table 1. **Head-only ensembling vs. APT.** We compare the performance of APT to ensembling classifier heads (without prompts) trained on distinct shards chosen uniformly at random. We group the datasets MIT-67, Cub-200, Caltech-256, Pets, and Flowers as "in-domain" due to their alignment with ImageNet21k and group the datasets Aircrafts and Stanford Cars as "out-of-domain" due to their difference with the pretraining. We report the average accuracy for the datasets within each group. We see that APT consistently outperforms Head-only ensembling, and the difference is most pronounced for out-of-domain datasets.

5. Applications of À-la-carte Learning

Decentralized Learning. We may have datasets $D_1, \ldots, D_n$ stored across n different servers or devices. Each server can train a prompt p_i on D_i in isolation. At inference time, we can assemble the prompts $p_1, \ldots, p_n$ on a central server and perform inference using $\mathbf{p}_{[n]}$. Each server can train their prompt p_i without exposing or uploading their raw data to the central server. This is useful whenever it is not possible to efficiently aggregate the data across the different devices, or if the individual devices are not willing to expose their raw data. We note that in Federated learning one typically looks at a different setting where a single central model is trained but the gradients are computed locally and then shared. Since a single model is trained via gradients aggregated across all the sources, this does not solve the à-la-carte learning problem and does not allow forgetting a source of data or firewalling a particular user from a source of data. Nevertheless, the two approaches are not mutually exclusive and we believe integrating them is an interesting avenue of research.

Model Versioning. Different users may have different rights in terms of which datasets they are permitted to access. For each user A we can associate a set of indices $I \subset [n]$ based on which datasets they have rights to. Then the version of the model we offer to user A would be given by $f(x; \theta_I)$. Aside from dataset rights, individuals may wish to add or drop data from the influence of a model simply for performance reasons. A dataset D_i may be useful for user A but introduce performance degradations for user B. À-la-carte learning allows us to include or not include the prompt θ_i for different users. Furthermore since the prompts do not need to be trained at the same time, we can add prompts at later points in time to update the model according to new data.

Forgetting. Forgetting a source D_i is easy, as we simply need to delete its associated prompt p_i. However, a service may periodically get requests to forget specific samples (x, y). Retraining a model from scratch each time a forget request is received can be prohibitively expensive. Furthermore even if the economic cost of training is no issue, satisfying the forget request immediately requires suspending the service until the retraining has completed which can induce service delays. Following [1], we can partition our dataset $\mathcal{D}$ into disjoint "shards" of equal size chosen uniformly at random so that $\mathcal{D} = \bigcup_{i \in [n]} D_i$. Then anytime we receive a request to forget a specific data point $(x, y) \in \mathcal{D}$ we only need to retrain the prompt p_i corresponding to the shard $\mathcal{D}_i$ that (x, y) belongs to. Furthermore the forget request can be satisfied immediately without any downtime to the service as the service can drop the prompt p_i from the model while it is being retrained and form predictions using the remaining prompts in the meantime.

Continual Learning. We can let D_i each correspond to a different training episode. Then in a continual learning setting at each episode i we train a prompt p_i on D_i and let our model after the specific training episode be $f(x; \mathbf{p}_I)$ where $I = \{1, 2, \ldots, i\}$.

6. Experiments

In all experiments we use a VIT-B/16 [4] pre-trained on ImageNet-21k. Unless explicitly stated otherwise, we use the pre-trained model `vit_base_patch16_384` from the timm[3] library in PyTorch [32].

Datasets. We evaluate APT on the datasets MIT-67 [33], Cub-200-2011 [36], FGVC-Aircrafts [28], Oxford Flowers [30], Caltech-256 [13], Oxford Pets [31], and Stanford Cars [16]. Based on the distance from the ImageNet21k pre-training, similarly to [19] we classify the datasets MIT-67, Cub-200-2011, Oxford Flowers, Caltech-256, and Oxford Pets as "in-domain" datasets and classify the datasets FGVC-Aircrafts and Stanford Cars as "out-of-domain" datasets. To test APT on class incremental learning problem we use Split CIFAR-100 [17] (10 training episodes, 10 classes per episode) and for domain incremental learning we use CORe50 (8 training domains, 3 test domains) [25, 26].

[3] https://github.com/rwightman/pytorch-image-models

Dataset	No Sharding	2 Shards	3 Shards	5 Shards	10 Shards	15 Shards	20 Shards	50 Shards
MIT-67	86.2%	86.2%	86.0%	87.3%	86.8%	86.3%	86.3%	84.2%
Cub-200	86.6%	87.8%	87.2%	87.2%	86.5%	85.5%	83.9%	79.9%
Caltech-256	91.7%	91.1%	90.8%	90.3%	89.7%	89.1%	88.7%	87.1%
Pets	93.3%	93.1%	93.4%	93.3%	93.4%	93.5%	93.3%	92.3%
Aircrafts	71.0%	61.1%	60.2%	56.3%	49.9%	46.6%	45.4%	36.9%
Flowers	99.1%	99.3%	99.1%	98.8%	98.5%	98.6%	97.6%	97.7%
Stanford Cars	81.2%	70.7%	66.6%	59.4%	52.1%	47.0%	45.8%	39.1%
Average	87.0%	84.2%	83.3%	81.8%	79.6%	78.1%	77.3%	73.9%

Table 2. **Accuracy of shard ensembles.** Accuracy of ensembling prompts trained on disjoint shards chosen uniformly at random. We see that for many datasets the performance of the ensemble is close to the paragon of prompt tuning on the entire dataset, despite each predictor of the dataset only seeing a fraction of the entire dataset.

Dataset	Finetuning	Head-only	Bias+Head	Deep PT	Deep Shared PT	Shallow PT	FT vs. PT gap
MIT-67	87.1%	85.6%	87.2%	86.2%	86.5%	86.0%	-0.9%
Cub-200	88.4%	87.0%	89.4%	86.6%	86.4%	85.6%	-1.8%
Caltech-256	93.5%	90.4%	93.0%	91.7%	91.3%	90.4%	-1.8%
Pets	94.5%	92.2%	94.9%	93.3%	92.9%	91.6%	-1.2%
Aircrafts	75.6%	54.8%	75.6%	71.0%	68.2%	62.1%	-4.6%
Flowers	97.4%	98.8%	99.4%	99.1%	98.9%	98.5%	1.7%
Stanford Cars	84.3%	64.5%	86.6%	81.2%	78.6%	69.6%	-3.1%
Avg	88.7%	81.9%	89.4%	87.0%	86.1%	83.4%	-1.7%

Table 3. **Finetuning vs. Prompt Tuning.** We compare different finetuning methods to prompt tuning. In the "Head-only" column only the linear classifier head is trained. In "Bias+Head" the bias's as well as the classifier head are trained. "Deep PT" is prompt tuning with memory tokens at each layer. "Deep Shared PT" is prompt tuning where the memory tokens are shared across the layers. In "Shallow PT" a single prompt is tuned without memory tokens. "FT vs. PT Gap" reports the accuracy of Deep PT minus the accuracy of Finetuning.

Comparison of model-tuning methods. Since our method is based on prompt-tuning, in Table 3 we measure how it compares to standard fine-tuning. Consistent with [15], we see that on most datasets prompt tuning is competitive (within 2% accuracy) with finetuning and outperforms head-only tuning, especially on out-of-domain datasets. We also observe that per-layer memory tokens (Deep PT) have the best trade-off between accuracy and computational cost, motivating our design choice to use it.

Decrease in performance due to sharding. Given a sharded dataset, we aim to establish whether composing per-shard prompts using APT achieves a comparable performance to training a prompt on all available data (paragon). Following [1], we split the training set into disjoint shards of equal size. The splitting is done by selecting samples uniformly at random, hence the number of examples per class can slightly vary across shards and smaller shards may not have examples from all classes. We train prompts on each of the shards in isolation and then compose a model using APT. The test accuracies as we increase the number of splits are reported in Table 2. Figure 3 (A) shows the increase in test error of the APT method relative to the paragon. As expected, the accuracy of APT generally decreases as the number of splits increases. However, for many datasets the drop off in accuracy is surprisingly small: on the in-domain datasets for 10-20 shards the accuracy of APT is within 2-5% of the accuracy of the paragon of training on the entire dataset. The main exceptions are out-of-domain datasets, where we observe a steeper accuracy drop when splitting the dataset. We hypothesize that for out-of-domain dataset, synergistic information between datapoints of different shards is more important for the training process.

Importance of composing prompts. In Figure 3 (C) we plot the gap between the average individual prompt accuracy and the accuracy of APT. We see that as the number of shards increases, the difference grows. This implies that while the performance of the ensemble may drop off slowly, that the performance of the individual predictors is deteriorating. This demonstrates that on large and fragmented data pools, individual prompts do not have enough information to classify correctly, and aggregating their information through the APT composition mechansim is essential.

Ablations. We perform a series of ablations to piece out the essential components of APT. To understand the effect of the attention masking, in Figure 2 we compare APT to the naïve method of concatenating all prompts without struc-

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

Table 4. **Performance on Split CIFAR-100 and CORe50.** Reporting average accuracy on the test set. Numbers for the non-APT methods are reported in [7] or [8]. For fair comparison against [7, 8] we have changed the resolution of our VIT to 224 from 384. Since APT does not train with a memory buffer we compare against the memoryless versions of L2P and S-Prompts.

tured attention. We see that naive concatenation performs almost uniformly worse than APT on average and has significantly higher variance, failing with very low accuracies in some cases. To isolate the effect of prompt tuning on the success of APT, in Table 1 we compare our APT method to training a simple head-only classifier on each shard. We see that APT uniformly outperforms its head-only counterpart, and that the difference is especially pronounced for out-of-domain datasets.

Forgetting sources. In Figure 3 (B) we plot the increase in error of the APT method after a certain number of shards (and their corresponding prompts) are deleted. This simulates a setting where a service provider receives a sequence of forget requests and consequently must remove prompts from the model. We see that starting with 20 shards, even after removing 10 shards for most the datasets the decline in accuracy is approximately 5% or less despite utilizing half the data. Since training time is directly proportional to the number of samples in the training set, this implies that we can reduce the cost of retraining after a forget request by an order of magnitude with a negligible drop in accuracy. Furthermore as shown in Figure 3 (B) we can handle a large number of forget requests sequentially without retraining before accuracy meaningfully declines. Moreover, since adding and removing sources are symmetric operations for APT, the same plot can be interpreted in reverse as showing the performance of APT in incrementally learning from an increasing set of data sources.

Class Incremental Learning. Oftentimes one wishes to add new classes to the model incrementally. In this section we explore class-incremental-learning (CIL) where at different training episodes we have access to different classes. To evaluate APT in this setting, we use the Split CIFAR-100 benchmark where the dataset is split into 10 disjoint sets of classes, with each subset containing 10 classes each. We train prompts on each subset in isolation. At inference time, we simply concatenate the class predictions from the individual prompts in line with our APT scheme. In Tab. 4 we report the results of APT in this setting. Out-of-the-box APT outperforms all the baselines and has a comparable performance to L2P [8]. We note that an advantage of L2P is the ability to dynamically select the prompts based on the test sample. Since prompts in APT are compositional by construction, we can easily implement a similar mechanism. Similarly to [37] we perform K-means on each episode in the embedding space to extract reference prototypes for that episode ($K = 20$), then at inference time we weight each episode prompt based on the distance of the instance's embedding from that episode's prototypes. See details in Sec. B in the supplementary material for the exact weighting scheme. We denote this method in the tables as APT-Weight (APT-W), and note that using this hard-coded weighting strategy – in contrast with L2P's learned prompt selection mechanism – APT-W outperforms L2P. We note that this weighting scheme still satisfies the à-la-carte learning requirement since the reference prototypes for each source are constructed independently of the other sources.

Domain Incremental Learning. Oftentimes one encounters data from different domains at different points in time. In the continual learning literature this setting is referred to as domain-incremental-learning (DIL). In this section we evaluate APT on domain incremental learning. In Tab. 4 we report the results of running APT on the CORe50 domain incremental learning benchmark. The CORe50 dataset contains data from 8 training domains and 3 test domains. By training prompts independently on each of the training domains, out-of-the-box APT outperforms all other methods on CORe50. Weighting the prompts in the APT-W scheme seems to give only a marginal increase (0.25%) in performance.

7. Conclusion

We introduced the general problem of *À-la-carte Learning* and an efficient solution to the problem using À-la-carte Prompt Tuning (APT). We demonstrate that models constructed à la carte are competitive with models trained on the union of the respective sources, with added benefits for privacy and customization. Furthermore APT achieves state-of-the-art performance for both class incremental learning and domain incremental learning. While APT offers one solution to the À-la-carte Learning problem, we emphasize that this problem is more general and deserves further study in order to develop competitive machine learning methods that respect users' data usage and privacy rights.

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, 12

[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

[12] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 11

[13] Griffin, Holub, and Perona. Caltech 256, Apr 2022. 6, 12

[14] Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models. In *AISTATS*, pages 2008–2016, 2021. 3

[15] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIII*, page 709–727, Berlin, Heidelberg, 2022. Springer-Verlag. 3, 4, 7

[16] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013. 6, 12

[17] Alex Krizhevsky. Learning multiple layers of features from tiny images. pages 32–33, 2009. 6, 12

[18] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics. 3

[19] Hao Li, Pratik Chaudhari, Hao Yang, Michael Lam, Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Rethinking the hyperparameters for fine-tuning. In *International Conference on Learning Representations*, 2020. 6

[20] Junnan Li, Ramprasaath R. Selvaraju, Akhilesh Deepak Gotmare, Shafiq Joty, Caiming Xiong, and Steven Hoi. Align before fuse: Vision and language representation learning with momentum distillation. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. 12

[21] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4582–4597. Association for Computational Linguistics, 2021. 3

[22] Zhizhong Li and Derek Hoiem. Learning without forgetting. In *ECCV*, 2016. 8

[23] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, aug 2022. Just Accepted. 2

[24] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland, May 2022. Association for Computational Linguistics. 3

[25] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot*

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. Fine-grained visual classification of aircraft. Technical report, 2013. 6, 12

[29] Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. In Vitaly Feldman, Katrina Ligett, and Sivan Sabato, editors, *Proceedings of the 32nd International Conference on Algorithmic Learning Theory*, volume 132 of *Proceedings of Machine Learning Research*, pages 931–962. PMLR, 16–19 Mar 2021. 3

[30] Maria-Elena Nilsback and Andrew Zisserman. A visual vocabulary for flower classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1447–1454, 2006. 6, 12

[31] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012. 6, 12

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Curran Associates Inc., Red Hook, NY, USA, 2019. 6

[33] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 413–420, 2009. 6, 12

[34] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 18–24 Jul 2021. 3

[35] Mark Sandler, Andrey Zhmoginov, Max Vladymyrov, and Andrew Jackson. Fine-tuning image transformers using learnable memory. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12155–12164, June 2022. 3, 4, 5, 11

[36] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 6, 12

[37] Yabin Wang, Zhiwu Huang, and Xiaopeng Hong. S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. 3, 8

[38] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149, 2022. 3, 11

[39] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. 11