Input Anchoring Makes Your Model a Deep Analytic Continual Learner

Anonymous authors
Paper under double-blind review

ABSTRACT

In class-incremental learning, models attempt to learn from a stream of tasks while having limited storage capacity. The strongest guarantee a continual learner can make is 'absolute memorization' which, beyond simply avoiding catastrophic forgetting, promises to produce a model identical to its full replay counterpart without needing the original dataset. Recent work in Analytic Class Incremental Learning (ACIL) has demonstrated that linear layers with input dimension d can be made into absolute memorizers with storage complexity $O(d^2)$. A variety of approaches leverage this by applying a wide linear layer atop large, frozen pre-trained models. While ACIL methods achieve absolute memorization in linear regression models, no work has to our knowledge yet achieved absolute memorization in deep neural networks. In this paper, we define Input Anchoring and prove that degree-L deep neural networks with this property achieve absolute memorization with storage complexity $O((d+1)^{2L})$. We identify previously proposed architectures which already have the input-anchoring property, and introduce our own simple and performant input-anchored architecture: the Anchored Multi-Layer Perceptron (AMLP). We show that the benefits of absolute memorization can be achieved with polynomial storage and time complexity via our Loss-Anchoring Points (LAP) approximation method, and leverage this finding to achieve state-of-the-art performance over the latest ACIL methods at matched storage and model width on pre-trained model class-incremental benchmarks.

1 INTRODUCTION

In continual learning (CL), models are fed a stream of data divided into tasks which shift over a sequence, and they are evaluated on their performance across all tasks. In the class incremental learning (CIL) scenario, which is the most challenging variant, the tasks feature disjoint sets of classes and inputs do not feature task identities, so models must learn not only to perform each task but also to recognize to which task a given input belongs (van de Ven et al., 2022; Masana et al., 2023). The key constraint which makes the scenario challenging is that storage capacity (measured in data points, parameters, and any other form of state maintained across task boundaries) is limited; a model cannot simply cache all past data, and naive parametric optimization overfits to the latest tasks, a phenomenon known as catastrophic forgetting (McCloskey & Cohen, 1989; Kirkpatrick et al., 2017).

The strongest claim a continual learner can make is 'absolute memorization', where it guarantees that its resulting model is identical to its full replay counterpart (a model which was trained with access to all information, new and old, at the same time) while using only a small fraction of the required storage. This goes a step beyond simply avoiding catastrophic forgetting: it enables a system to not only retain prior performance but improve it with new data over time.

Recent work on Analytic Class Incremental Learning (ACIL) (Zhuang et al., 2022; 2024; Wang et al., 2026; Zhao et al., 2026) and its methodological twin, RanPAC, (McDonnell et al., 2023; Peng et al., 2025), has demonstrated that linear regression models can achieve absolute memorization with storage constant with respect to the number of data points by storing a set of $O(d^2)$ sufficient statistics for input dimension d. This works because linear regression models, when trained with mean squared error, have loss landscapes which are entirely determined by cross and autocorrelation matrices (Saxe et al., 2014) (for data points (x, y) sampled from some distribution $\mathcal{D}$, these are: $\mathbb{E}_{\mathcal{D}}[xx^T]$, $\mathbb{E}_{\mathcal{D}}[yx^T]$). Hence, a model trained on these statistics will have identical closed form solutions when

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

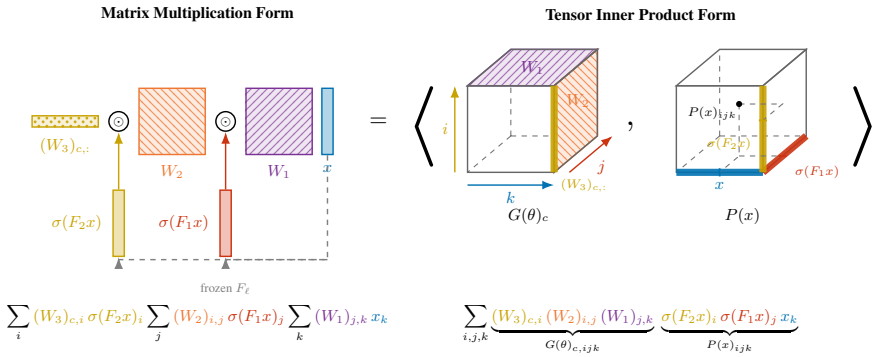


Figure 1: Forward pass of an Anchored MLP at $L = 3$ in matrix-multiplication form (left) and tensor-inner-product form (right, Proposition 1). On the right the learnable parameters are separated from the data: $G(\theta)$ depends only on the weight matrices W, and $P(x)$ only on the datapoint x and the frozen random matrices F (each cube entry is the product of the corresponding elements on its sides and edges, taken as 1 when blank). Theorem 1 shows that this separation yields absolute memorization.

fit analytically, and identical parameter trajectories when trained with full-batch gradient descent. These papers achieve state-of-the-art results by freezing large, pre-trained models (Dosovitskiy et al., 2021; Radford et al., 2021) and training only a final linear regression head, trading representational flexibility for the guarantee of absolute memorization with constant data storage.

This invites a natural question: can the same property be extended to deep neural networks, removing ACIL's requirement that every weight outside the final linear head be frozen? In this work, we show that it can. By adjusting the activation structure of a deep learning model to make it multi-affine in several arguments, a property we call *input anchoring*, we construct deep models whose forward passes are linear over a space of exponentially many features, and which are therefore absolute memorizers in exactly the sense that linear regression models are. Specifically, we call a model f_θ input anchored if for fixed anchor maps ϕ_ℓ with no learnable parameters $f_\theta(x) = F_\theta(\phi_1(x), \phi_2(x), \ldots, \phi_L(x))$ where F_θ is affine in each argument separately (Definition 1; Figure 1).

Input anchoring comes with an immediate cost: caching all sufficient statistics, while constant in the number of datapoints, is exponential in depth. With all widths equal to d, the statistics of an L-layer model number $O((d+1)^{2L})$, severe enough that at the scales of most datasets and models in prior literature (Zhuang et al., 2022; McDonnell et al., 2023; Zhao et al., 2026) it is far more efficient to store the entire dataset than the sufficient statistics. To remedy this, we show that a low rank CP approximation (Kolda & Bader, 2009) of the sufficient statistic tensors can match or exceed the performance of the full dataset at scale (§4.3),, and that one can both fit and train on this approximation without ever materializing exponentially large tensors (§3.3).

1.1 Contributions

1. **The Input Anchoring Property:** We define *Degree-L input anchoring* (Definition 1) and show that for any input-anchored network the population squared-error risk (the whole landscape, not merely its minimiser) is determined by two moment tensors of the data (§3.1). At degree one, this is exactly the property leveraged by ACIL methods; higher degrees extend it to deeper models. Π-net CCP (Chrysos et al., 2020) and GaLU (Fiat et al., 2019) already satisfy this definition.
2. **The Anchored MLP Architecture:** We introduce the AMLP, a multi-layer perceptron template which generalizes GaLU models, and show that it is input-anchored (§3.2). It beats the random feature head of ACIL models and meets or exceeds a ReLU head with the same width and number of learnable parameters on CIFAR-100 (Srivastava et al., 2014): input anchoring does not collapse the head to a linear regression over its anchor features.
3. **The Loss Anchoring Points (LAP) Approximation Method:** We introduce LAP, which fits a rank-m CP decomposition to an input-anchored model's sufficient statistics without ever materialising the full exponentially large tensors (§3.3), and show that it achieves

performance comparable to prior state-of-the-art at matched storage and exceeding its full replay counterpart on large datasets (§4.3).

2 Related Work

2.1 Class Incremental Learning (CIL)

In Class Incremental Learning, data arrives to the learner sequentially in a set of T tasks, and models are evaluated on both their ability to recognize and perform all previously seen tasks (van de Ven et al., 2022). Building on the taxonomies of Zhou et al. (2024c), Zhou et al. (2024b), and Zhao et al. (2026), the field can be generally divided into the following approaches: *Regularization-based* methods attempt to minimize catastrophic forgetting by constraining gradient updates or model weights to reduce deviation on important parameters (Kirkpatrick et al., 2017; Li & Hoiem, 2016). *Replay-based* methods attempt to cache, compress, or generate data points such that new models can be trained on them (Rebuffi et al., 2017; Buzzega et al., 2020). *Architectural-based* approaches attempt to expand their models parameters or prompts over time to allow adaptation to new information, frequently while freezing important aspects of the underlying model to prevent forgetting (Mallya & Lazebnik, 2018; Wang et al., 2022b;a; Smith et al., 2023; Zhou et al., 2024a; Wu et al., 2025). While often empirically effective, these approaches are frequently heuristic based and lack strong guarantees on their ability to memorize data. In contrast, *Analytic-based* methods, use sufficient statistics to analytically compute full-replay solutions over frozen pre-trained models (Zhuang et al., 2022; McDonnell et al., 2023).

2.2 Analytic CIL

Initially proposed by Zhuang et al. (2022), Analytic Class Incremental Learning achieves absolute memorization on linear regression tasks over d input features by storing the $O(d^2)$ autocorrelation and cross-correlation statistics of the introduction, which are sums over examples and therefore update additively when a new task arrives. ACIL methods leverage this fact by freezing a pre-trained backbone, expanding its features through a random projection (Rahimi & Recht, 2007), and thereafter maintaining a regularized feature autocorrelation matrix recursively across tasks (Zhuang et al., 2022). This recipe has experienced a long lineage of iteration. RanPAC independently arrives at the same recursive least-squares solution and applies it atop adapted ViTs (McDonnell et al., 2023; Dosovitskiy et al., 2021; Zhou et al., 2024a); DS-AL adds a secondary analytic head built to reduce residual error (Zhuang et al., 2024); LoRanPAC continually truncates a singular value decomposition of the lifted features to improve stability (Peng et al., 2025); ORACIL routes inputs among multiple analytic heads via a conflict graph (Wang et al., 2026); and VILA, our primary baseline, fuses an adapted ViT branch trained on the first task with frozen CLIP features (Radford et al., 2021) via ℓ_2 normalized concatenation before the analytic head (Zhao et al., 2026).

All of these methods share the same recipe and hence the same weaknesses. Exactness holds only above a frozen feature extractor, so the backbone cannot change after the first task without breaking equivalence with the full-data solve. The head is linear over frozen random ReLU features, so performance is upper-bounded by kernel regression with the random-feature kernel of a 2-layer ReLU network (Rahimi & Recht, 2007; Jacot et al., 2018), and the resulting under-fitting has been documented (Zhuang et al., 2024). Finally, the $O(d^2)$ cache can exceed the size of the data it summarizes (§4.2).

2.3 Architectural Lineage of Input Anchoring

Two separate lines of work precede the *input anchoring* property we propose in this paper. We demonstrate that both of these model classes are input-anchored and, as a result, that both can serve as absolute memorizers in continual learning, but neither model class to our knowledge has ever previously been applied in the continual learning setting.

Gated Linear Units (Fiat et al., 2019) observe that in a ReLU neuron, $\max\{x^\top w, 0\} = \mathbf{1}\left[x^\top w \geq 0\right] \cdot (x^\top w)$, the same parameter both gates the neuron and sets its value, and decouple the two roles as $g_{w,u}(x) = \mathbf{1}\left[x^\top u \geq 0\right] \cdot (x^\top w)$ with the gate u frozen at initialization. At $u = w$ and infinite width, GaLU coincides with the corresponding ReLU network in the lazy regime (Jacot et al., 2018; Chizat

et al., 2019), and empirically it approaches ReLU performance whenever the ReLU network's gates do not move substantially during training.

Π-nets (Chrysos et al., 2020) make the output an explicit high-degree polynomial of the input by replacing conventional activation functions with repeated Hadamard products of linear maps, achieving parity with ReLU networks (Chrysos et al., 2023; Cheng et al., 2024) and admitting a treatment via analytic geometry (Kileel et al., 2019; Usevich et al., 2025). The variant we demonstrate has the input-anchored property is the coupled CP decomposition (CCP): for input x, weight matrices U_i and W, and Hadamard product $\odot$, $f(x) = Wz_n$ where $z_n = (U_{[n]}^\top x) \odot z_{n-1} + z_{n-1}$ and $z_1 = U_1 x$.

3 METHOD

Overview We first lay out the formal definition of input anchoring (Definition 1) and show that such a model is a linear regression over a set of exponentially many features, so that any point on its MSE loss landscape can be computed from two moment tensors which accumulate additively across tasks with no cached data (Theorem 1): this is absolute memorization. Because these moment tensors are exponentially large, specifically $O((d+1)^{2L})$ for model width d and degree L (Proposition 2), we then introduce the Loss Anchoring Points method (Algorithm 1) and show that training on its output as though it were a dataset with MSE loss is a tractable approximation of training on the moment statistics (Theorem 2). Throughout, we reference Figure 1 to provide intuition.

Notation Data are pairs (x, y) with $x \in \mathbb{R}^d$ and $y \in \mathbb{R}^C$. Throughout, $\odot$ is the elementwise (Hadamard) product, $\otimes$ the tensor (outer) product, $x^{\otimes L}$ the L-fold outer power, and $\langle \cdot, \cdot \rangle$ the Frobenius inner product. We use hats (e.g. $\hat{\mathcal{L}}$) to denote quantities computed over an LAP approximation as opposed to the true dataset, and we assume that both each class has an equal number of points and each task has an equal number of points for simplicity (this assumption is easily removed by giving datasets probabilities).

3.1 INPUT ANCHORING

We begin with the definition of input anchoring. Intuitively, this definition is designed to capture all architectures which obey a tensor decomposition trick similar to the one from Figure 1. Specifically, we mean tricks which allow us to convert a chain of matrix multiplications and element-wise products into an inner product of a data tensor ($P(x)$) and a learnable parameter tensor ($G(\theta)$). This enables absolute memorization by enabling us to drive the expectation past the parameter terms such that the loss is a function of two moment tensors (Theorem 1). In its most general form, we recognize this trick can be applied to any model which is multi-affine in L-many chunks of its input:

Definition 1 (Degree-L input anchored)**.** For layer widths $k_1, \ldots, k_L \in \mathbb{N}_{\geq 2}$, let $\mathcal{H} := \mathbb{R}^{k_1} \otimes \cdots \otimes \mathbb{R}^{k_L}$. An architecture $f_{\theta,\Phi} : \mathbb{R}^d \to \mathbb{R}^C$ with anchor functions $\Phi = (\phi_\ell : \mathbb{R}^d \to \mathbb{R}^{k_\ell})_{\ell \in [L]}$ is *degree-L input anchored* if:

(IA1) (*Multi-affinity*) There is a function $F_\theta : \mathbb{R}^{k_1} \times \cdots \times \mathbb{R}^{k_L} \to \mathbb{R}^C$ which is affine in each of its arguments (which we denote $v_1 \in \mathbb{R}^{k_1}, \ldots, v_L \in \mathbb{R}^{k_L}$) separately and smooth in θ, such that for every Φ and every $x \in \mathbb{R}^d$,

$$f_{\theta,\Phi}(x) \;=\; F_\theta\big(\phi_1(x), \ldots, \phi_L(x)\big)$$

(IA2) (*Sensitivity*) Writing $\frac{\partial^L F_\theta}{\partial v_1 \cdots \partial v_L}$ as the L-fold mixed derivative (a constant tensor, by (IA1)), the top-degree coefficients span the full tensor space:

$$\forall c \in [C] \;:\; \operatorname{span}\left\{ \left(\frac{\partial^L F_\theta}{\partial v_1 \cdots \partial v_L} \right)_c \;:\; \theta \in \Theta \right\} \;=\; \mathcal{H}$$

The sensitivity condition is designed to prevent lower degree input-anchored models from trivially meeting the definition of higher degree input-anchored models by 0-ing any coefficients on higher order terms (e.g. $\frac{\partial F_\theta}{\partial v_L} = 0$). We call the data-dependent functions 'anchor' functions because rather than operating over the previous output of the model as activation functions typically do, they instead operate over the original, unchanging input.

We can make this definition more concrete by recognizing that the Multi-affinity and Sensitivity conditions have natural coefficient requirements: F_θ must necessarily be in the form of a set of coefficients which can be multiplied over the outer product of the anchor functions.

Proposition 1 (Coefficient form of Definition 1)**.** *An architecture meets* (IA1) *if and only if there exists a unique $g(\theta, A) \in \mathbb{R}^C \otimes \big(\bigotimes_{\ell\in A} \mathbb{R}^{k_\ell}\big)$, smooth in θ, for each $A \subseteq [L]$ such that for every Φ, $x \in \mathbb{R}^d$, and $c \in [C]$,*

$$f_{\theta,\Phi}(x)_c \;=\; \sum_{A\subseteq[L]} \Big\langle g(\theta, A)_c,\; \bigotimes_{\ell\in A} \phi_\ell(x) \Big\rangle, \qquad \bigotimes_{\ell\in\emptyset} \phi_\ell(x) := 1.$$

Moreover, as $g(\theta, [L]) = \frac{\partial^L F_\theta}{\partial v_1 \cdots \partial v_L}$, it meets (IA2) *if and only if* $\operatorname{span}\{\, g(\theta, [L])_c \;:\; \theta \in \Theta\} \;=\; \mathcal{H}$. *(Proof in Appendix A.)*

In Figure 1 these tensors are visualized in the Tensor Inner Product Form. Following directly from this version of the definition, we can see that the parameter tensor and the data tensor are completely separated, making their interaction linear. This yields the following critical property:

Theorem 1 (MSE loss at any location is computable from sufficient statistics)**.** Let $f_{\theta,\Phi}$ be input-anchored with coefficient tensors defined by $g(\cdot, A)$ for $A \subseteq [L]$, and define the following:

$$G(\theta)_c := \bigoplus_{A\subseteq[L]} \operatorname{vec}(g(\theta, A)_c), \qquad P(x, \Phi) = \bigoplus_{A\subseteq[L]} \operatorname{vec}\left(\bigotimes_{\ell\in A} \phi_\ell(x)\right) \tag{1}$$

$$\mathcal{M} \;=\; \mathbb{E}\big[P(x, \Phi)^{\otimes 2}\big], \qquad \mathcal{V}_c \;=\; \mathbb{E}\big[y_c\, P(x, \Phi)\big]. \tag{2}$$

Then

$$\mathcal{L}_{\text{MSE}}(\theta) \;=\; \sum_{c=1}^{C} \Big[\langle G(\theta)_c \otimes G(\theta)_c,\, \mathcal{M}\rangle \;-\; 2\,\langle G(\theta)_c,\, \mathcal{V}_c\rangle\Big] \;+\; \mathbb{E}\,\|y\|^2 \tag{3}$$

Proof. Expand $\|f - y\|^2 \;=\; \sum_c f_c^2 \;-\; 2\sum_c y_c f_c \;+\; \|y\|^2$. By Proposition 1, $f_c(z)^2 \;=\; \big\langle G(\theta)_c \otimes G(\theta)_c,\, P(x, \Phi)^{\otimes 2}\big\rangle$ and $y_c f_c(z) = \langle G(\theta)_c,\, y_c P(x, \Phi)\rangle$; take expectations and use linearity. Gradients follow because $\theta \mapsto G(\theta)_c$ is smooth. □

Therefore $\mathcal{L}_{\text{MSE}}$ and $\nabla_\theta \mathcal{L}_{\text{MSE}}$ at *every* θ are computable from $(\mathcal{M}, \mathcal{V}, \mathbb{E}||y||^2)$ without storing any data points: full-batch gradient descent on the statistics produces the same iterates as full-batch gradient descent on the data, for any initialization and hyperparameters. At $L = 1$ with ϕ_1 the identity, $\mathcal{M} = \mathbb{E}[xx^\top]$, $\mathcal{V} = \mathbb{E}[yx^\top]$, and the ridge solution is $\mathcal{V}(\mathcal{M} + \gamma I)^{-1}$: precisely the setting of Zhuang et al. (2022), whose degree-one landscape is the one analyzed by Saxe et al. (2014). Because the statistics are expectations, they also accumulate across a task stream with no data retained: if task t has statistics $\mathcal{M}^{(t)} := \mathbb{E}_{\mathcal{D}_t}[P(x, \Phi)^{\otimes 2}]$ and $\mathcal{V}^{(t)}$, then $\mathcal{M} = \frac{1}{T}\sum_t \mathcal{M}^{(t)}$ and $\mathcal{V} = \frac{1}{T}\sum_t \mathcal{V}^{(t)}$ (Corollary 1, Appendix A).

Proposition 2 (Storage Complexity)**.** *Let $k_1 = k_2 = \cdots = k_L := d$ and $C = O(d^L)$. Then $\mathcal{M}$ has $(1 + d)^{2L}$ entries and the sufficient statistics occupy $O((1 + d)^{2L})$ floats, independent of the number of datapoints N. (Proof in Appendix A.)*

We show how to leverage Theorem 1 without storing or computing these tensors through a low rank CP approximation (Kolda & Bader, 2009) in §3.3.

3.2 Input-Anchored Architectures

To provide a concrete example of an input-anchored architecture, we now introduce our Anchored MLP, a template which generalizes GaLU's (Fiat et al., 2019) multi-affine structure to an arbitrary number of layers and arbitrary set of activation functions. We then show that two previously proposed architectures, GaLU itself and Π-net's CCP, already satisfy Definition 1.

Definition 2 (Anchored MLP)**.** Fix $L \geq 2$, widths $k_1, \ldots k_L$ such that $k_1 = d$, the dimension of the input data, an element-wise activation function $\sigma : \mathbb{R} \to \mathbb{R}$ and random frozen weight matrices $F_1, \ldots, F_{L-1}$ where $F_\ell \in \mathbb{R}^{k_{\ell+1}\times k_1}$. Define $\phi_1(x)$ to be the identity and for $\ell \geq 2$ let $\phi_\ell(x) = \sigma(F_{\ell-1}x)$. For trainable weight matrices $W_1, \ldots W_L$ where $W_\ell \in \mathbb{R}^{k_{\ell+1}\times k_\ell}$ we define:

$$h_1 = W_1\phi_1(x), \qquad h_\ell = W_\ell(\phi_\ell(x) \odot h_{\ell-1}) \qquad f_{\theta,\Phi}(x) = h_L \tag{4}$$

Algorithm 1 Loss Anchoring Points: fitting m atoms to the sufficient statistics

Require: a degree-L input anchored architecture $f_{\theta,\Phi}$ with coefficient function g, a dataset $\mathcal{D} = \{(x_j, y_j)\}_{j=1}^{N}$, a memory size m, and a step budget t_{budget}

Ensure: atoms $\hat{\mathcal{D}} = \{(\hat{x}_i, \hat{y}_i)\}_{i=1}^{m}$

let $\mathcal{A} := \{A \subseteq [L] : \exists \theta,\ g(\theta, A) \neq 0\}$ and $\kappa(a,b) := \sum_{A\in\mathcal{A}} \prod_{\ell\in A} \langle \phi_\ell(a),\, \phi_\ell(b)\rangle$

cache $S_{\mathcal{M}} \leftarrow \frac{1}{N^2}\sum_{j,j'} \kappa(x_j, x_{j'})^2$ and $S_{\mathcal{V}} \leftarrow \frac{1}{N^2}\sum_{j,j'} \langle y_j, y_{j'}\rangle\, \kappa(x_j, x_{j'})$

initialise $\hat{\mathcal{D}}$ with m rows of $\mathcal{D}$ drawn uniformly at random without replacement

for $t = 1, \ldots, t_{\text{budget}}$ **do**

  $J_{\mathcal{M}} \leftarrow \frac{1}{m^2}\sum_{i,i'\in[m]} \kappa(\hat{x}_i, \hat{x}_{i'})^2 - \frac{2}{mN}\sum_{i\in[m],\, j\in[N]} \kappa(\hat{x}_i, x_j)^2 + S_{\mathcal{M}}$ ▷ See Lemma 1

  $J_{\mathcal{V}} \leftarrow \frac{1}{m^2}\sum_{i,i'\in[m]} \langle \hat{y}_i, \hat{y}_{i'}\rangle\, \kappa(\hat{x}_i, \hat{x}_{i'}) - \frac{2}{mN}\sum_{i\in[m],\, j\in[N]} \langle \hat{y}_i, y_j\rangle\, \kappa(\hat{x}_i, x_j) + S_{\mathcal{V}}$

  $\hat{\mathcal{D}} \leftarrow \mathrm{Adam}\big(\hat{\mathcal{D}},\ \nabla_{(\hat{x},\hat{y})}(J_{\mathcal{M}} + J_{\mathcal{V}})\big)$ (Kingma & Ba, 2015)

end for

return $\hat{\mathcal{D}}$, $J := J_{\mathcal{M}} + J_{\mathcal{V}}$

We now show that the Anchored MLP is, in fact, input-anchored.

Proposition 3. *The AMLP architecture with L weight matrices is degree L input anchored.*

A proof of this proposition can be found in Appendix A.

Remark 1. Fiat et al. (2019)'s model is an AMLP with $L = 2$ and $\sigma(x) := \mathbb{1}(x > 0)$. While they study it for its benefits in proving theoretical bounds, we demonstrate its advantages for continual learning.

Chrysos et al. (2020)'s CCP architecture is input-anchored because the polynomial activations it builds are naturally multi-affine.

Proposition 4. *Chrysos et al. (2020)'s CCP architecture with $L-1$ layers is degree L input anchored.*

A proof of this can be found in Appendix A. We introduce AMLP rather than adopting CCP because its generality admits arbitrary activation functions and layer widths, and because CCP's skip connections make its moment tensors more costly to approximate (in Algorithm 1 their $|\mathcal{A}|$ is 2^{L-1} instead of 1).

3.3 Loss Anchoring Points: Avoiding exponential space and time with a CP approximation

In this section, we present via Algorithm 1 a method which compresses the high dimensional tensor into a rank m CP decomposition (Kolda & Bader, 2009) without needing exponential space. We show that its objective is exactly the distance between the approximate and true moment tensors (Lemma 1), that approximations stack across task boundaries (Lemma 2), and hence that when the approximation is exact, LAP recovers absolute memorization (Theorem 2).

Lemma 1 (Algorithm 1 builds an approximation of $\mathcal{M}$ and $\mathcal{V}$)**.** *Let $P(x,\Phi)_{\mathcal{A}}$, $\mathcal{M}_{\mathcal{A}}$, and $\mathcal{V}_{\mathcal{A}}$ be the quantities of equation 1–equation 2 with the direct sum restricted to $A \in \mathcal{A}$, and let $\hat{\mathcal{M}}_{\mathcal{A}}$, $\hat{\mathcal{V}}_{\mathcal{A}}$ be the same quantities over $\hat{\mathcal{D}}$. Then the objective J of Algorithm 1 satisfies $J = \left\|\mathcal{M}_{\mathcal{A}} - \hat{\mathcal{M}}_{\mathcal{A}}\right\|_F^2 + \left\|\mathcal{V}_{\mathcal{A}} - \hat{\mathcal{V}}_{\mathcal{A}}\right\|_F^2$.*

A proof of this lemma can be found in Appendix A. Since $\hat{\mathcal{D}}$ is itself a dataset, Theorem 1 applied to it says that MSE training on the atoms is exactly training on $(\hat{\mathcal{M}}, \hat{\mathcal{V}})$; what remains is whether per-task approximations compose.

Lemma 2 (LAP approximations stack over task boundaries)**.** *Let $\mathcal{D} = \bigcup_{t=1}^{T} \mathcal{D}_t$ be a mixture of T datasets each with LAP approximations $\hat{\mathcal{D}}_t$ which achieve for some error bound ε losses $J_t \leq \varepsilon$. Then their union, $\hat{\mathcal{D}} := \bigcup_{t=1}^{T} \hat{\mathcal{D}}_t$, satisfies $J(\mathcal{D}, \hat{\mathcal{D}}) \leq \varepsilon$.*

The proof (Appendix A) follows from Corollary 1 and the convexity of $\|\cdot\|_F^2$.

Theorem 2 (When realizable, LAP's approximation recovers absolute memorization in CL)**.** Let there exist for a given input anchored architecture $f_{\theta,\Phi}$ a set of datasets $\{\hat{\mathcal{D}}_t\}_{t=1}^T$ each of size at most m that each reproduce exactly the sufficient statistics $\mathcal{M}^{(t)}$ and $\mathcal{V}^{(t)}$ of their corresponding true datasets $\{\mathcal{D}_t\}_{t=1}^T$. Then, stacking the global minimizers of Algorithm 1's objective for each task and training $f_{\theta,\Phi}$ with full-batch gradient descent on MSE Loss will produce identical model weights to the matched algorithm with the same initialization and hyperparameters which trains with access to the full dataset.

A proof of this theorem can be found in Appendix A.

Algorithm 1 never materialises an object of size $(1+d)^L$ or larger: it runs in time polynomial in N, d, and m and uses $O(NLd)$ working memory beyond the $m(d+C)$ floats of the atoms (Proposition 5, Appendix A). LAP therefore replaces an exponential cache with m atoms at a polynomial fitting cost; we now test what accuracy this buys.

4 Experiments

4.1 Experimental Setup

Protocol and backbone. We follow an identical protocol to Zhao et al. (2026), and use their released code and checkpoints for everything up to the classifier head. The backbone we use underneath our AMLP is VILA's asymmetric dual branch architecture: a ViT-B/16 (Dosovitskiy et al., 2021) equipped with adapters (Zhou et al., 2024a), trained on the first task $\mathcal{D}_0$ and thereafter frozen, alongside a frozen CLIP-ViT-B/16 (Radford et al., 2021) universal branch. The two branches' features (dimension 768 and 512 respectively) are ℓ_2-normalized and concatenated into a fused representation which is the input x to every head we evaluate. VILA then lifts these 1280-dimensional features into a 16384-dimensional space via a frozen Gaussian random projection followed by a ReLU activation function and fits a ridge-regression classifier by recursive least-squares. We replace this lifted linear head with an AMLP head (Definition 2) of matched width and depth and train it with gradient descent on mean-squared error against centered one-hot targets. All other components of VILA, including their Candidate Semantic Enhancement system, are left unchanged. Training and fitting procedures are given in Algorithm 2 (Appendix B).

Datasets and metrics. Experiments 1 and 3 use VILA's eight benchmarks and class splits: CIFAR-100 (Krizhevsky et al., 2009), ImageNet-R (Hendrycks et al., 2021), CUB-200 (Wah et al., 2011), UCF-101 (Soomro et al., 2012), FGVC-Aircraft (Maji et al., 2013), Stanford Cars (Krause et al., 2013), Food-101 (Bossard et al., 2014), and SUN-397 (Xiao et al., 2010) with task counts T listed in Table 1. Experiment 2 uses a class-balanced 1.1M-image subset of Places365 (Zhou et al., 2018) (§4.3). We report the same metrics as VILA: *last-task accuracy* A_T, the accuracy on all classes after task T, and *average incremental accuracy* $\bar{A} := \frac{1}{T}\sum_{t=1}^{T} A_t$. All numbers we compute are a mean over 3 seeds.

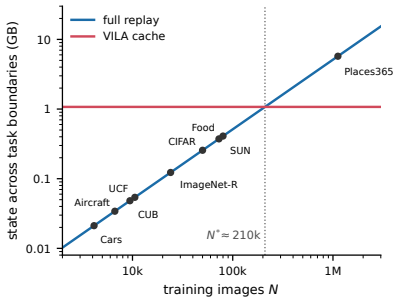
Hyperparameter selection. Rigorous per-benchmark tuning is at odds with the CIL setting, since selecting hyperparameters against final accuracy requires evaluating on all tasks. We therefore tune hyperparameters once on CIFAR-100 with $T = 20$ and transfer them to every experiment (sweeps in Appendix B). For the AMLP this yields dropout 0.3 (Srivastava et al., 2014) and learning rate $3e-4$; for VILA GD, an ablation where we train VILA's frozen feature head with gradient descent, dropout 0.0 and learning rate $3e-4$. All gradient descent methods use a budget of 4000 steps with a cosine schedule (Loshchilov & Hutter, 2017). The AMLP's activation is $\sigma(x) = s(32x)$ with s the sigmoid: the differentiable gate closest to GaLU's step, which Experiment 3 shows is within noise of the alternatives, for discussion see Appendix B. VILA's ridge parameter λ is selected by its own leave-one-out procedure on the base task.

4.2 Experiment 1: Replacing the Analytic Head at Matched Width

Experiment 1 asks whether replacing VILA's lifted linear head with a deep input-anchored head improves accuracy at matched width, holding everything else fixed. Rows above the inner rule in Table 1 are the published numbers of Zhao et al. (2026) for the analytic-based methods (McDonnell

Table 1: Last-task accuracy A_T (%) per benchmark, with average A_T and average incremental accuracy $\bar{A}$ (%) over all twelve settings, extending Zhao et al. (2026)'s Table 2. Rows above the inner rule are their published numbers; rows below are our runs on their released code. T denotes the number of tasks. Best and second-best are bold and blue, computed over every method in the full table (Table 2, Appendix D), which also reports $\bar{A}$ per benchmark and eight further published baselines.

Dataset	CIFAR		ImageNet-R		CUB	UCF	Aircraft		Cars	Food	SUN		Avg.	
$T =$	10	20	20	40	20	20	10	20	20	20	30	50	A_T	$\bar{A}$
RanPAC	86.40	86.36	71.83	71.70	88.93	97.73	59.23	58.33	66.81	84.56	79.65	79.75	77.61	82.95
LoRanPAC	91.32	90.34	74.95	73.42	**90.25**	98.22	59.38	59.32	70.17	86.84	80.77	80.76	79.64	85.23
VILA Analytic (Published)	91.94	91.18	85.28	84.20	89.36	98.71	68.02	65.56	90.28	88.89	83.73	83.55	85.06	89.98
VILA Analytic (Reproduced)	92.24	91.40	85.31	84.32	89.35	98.67	66.93	66.14	90.25	88.96	83.39	83.34	85.02	90.00
VILA Analytic (Double Features)	92.46	91.67	86.00	84.73	89.20	98.84	67.05	66.47	**90.82**	89.41	83.97	84.15	85.40	90.17
VILA GD	92.12	91.31	84.81	83.27	88.82	98.57	64.00	62.82	90.41	88.91	83.51	83.53	84.34	89.65
AMLP + VILA (Ours)	**92.69**	**91.87**	**86.51**	**85.81**	89.61	**98.87**	**68.51**	**67.91**	90.52	**89.71**	**84.09**	**84.20**	**85.86**	**90.44**



Method	**Storage** (GB)	**Final** Acc. (%)	**Avg. Inc.** Acc. (%)
VILA Analytic	1.07	54.93	67.98
AMLP, full data	5.74	55.73	**68.85**
AMLP, coreset	1.07	54.95	67.18
AMLP, LAP (ours)	1.07	**56.10**	68.80

Figure 2: **Left:** state carried across task boundaries versus training-set size at VILA's dimensions (fp32). Replay and the analytic cache cross at N^*=210k images, above all eight standard benchmarks and 5× below our Places365 subset; exact AMLP statistics are off-chart at 1.8 PB. **Right:** Places365, with every compressed method pinned to the cache size.

et al., 2023; Peng et al., 2025; Zhao et al., 2026); their published prompt (Wang et al., 2022b;a; Smith et al., 2023), adapter (Zhou et al., 2024a; Wang et al., 2025), LoRA (Wu et al., 2025; He et al., 2025), and VLM (Zhou et al., 2025) baselines, together with $\bar{A}$ per benchmark, are reproduced in Table 2 (Appendix D). We do not re-run them. The rows below the rule are our runs on their released code and differ only in the head: *VILA Analytic (Reproduced)* is their method as released. *VILA Analytic (Double Features)* doubles the random-projection width to $32,768$, so that the analytic head has at least as many features as our AMLP's frozen projection and learned projections combined (F and W matrices); it tests whether the gap can be closed by capacity alone. *VILA GD* keeps VILA's lifted linear head but trains it by gradient descent under the same procedure as our AMLP; it tests whether the gap is an optimizer effect. *AMLP + VILA* is our method. Because every benchmark here is small enough that a full replay buffer (all data across all tasks) is cheaper than VILA's cache, AMLP is trained on full replay: by Theorem 2, this is the optimal LAP approximation when the atom budget m exceeds N.

AMLP + VILA has the best average on both metrics, improving last-task accuracy over the reproduced analytic head by 0.84 points and average incremental accuracy by 0.44. The two controls locate the gain. Doubling the analytic head's features recovers roughly half of it ($+0.38$ / $+0.17$), so width alone does not close the gap. Training the linear head by gradient descent instead of analytically *loses* accuracy (-0.68 / -0.35), so the gain is not an optimizer effect either; if anything, gradient descent is a handicap that the deeper head overcomes. AMLP is best or second-best on every column except CUB and Cars, where LoRanPAC and the Double-Features control respectively lead on A_T, both by under half a point.

Honest Storage Accounting. Counting as storage all fp32 state carried across a task boundary: a full replay buffer holds $N \times d$ floats and grows with the stream; an analytic cache for a linear regression head holds d^2 floats regardless of N, which at VILA's lifted dimension of 16384 is 1.07 GB; exact input-anchored statistics are likewise constant in N but occupy roughly 1.8 PB at VILA's dimensions (Proposition 2); and LAP holds $m(d + C)$ floats with m free, so its storage can

be sized to any budget. Figure 2 (left) plots the first two: they cross at $N^* \approx 210$k images, and every benchmark in the standard PTM-CIL suite (Zhou et al., 2024b) lies below N^*, where the replay buffer is 2.6–$35\times$ smaller than the cache. The constant-storage advantage that motivates analytic methods has therefore not been exercised by the benchmarks on which they were established. Our Places365 subset lies $5\times$ past the crossover, where non-trivial compression is required to achieve absolute memorization at VILA's storage scale.

4.3 Experiment 2: Matched Storage Comparison at Scale

Experiment 2 asks what accuracy each method buys at the analytic cache's storage budget, on a stream large enough that a full-replay solution is no longer more efficient to store. We apply the *VILA analytic* and *AMLP + VILA (Ours)* models of Experiment 1, with the same hyperparameters, to a class-balanced subset of Places365 (Zhou et al., 2018): 3063 images from each of 365 classes, 1.1M images in total. Task 0 holds five classes and is used to train VILA's adapter branch; the remaining 360 classes arrive one per task. At this scale a full replay buffer of VILA's fused features requires 5.74 GB.

Three variants share the AMLP head and differ only in what is stored. *Full data* retains the fused features and labels of every training image. *coreset* retains a uniformly random subset of those rows with integer labels. *LAP* retains the atoms $(\hat{x}_i, \hat{y}_i)$ produced by Algorithm 1. As classes generally arrive one-by-one, we run Algorithm 1 once per class and drop any class labels outside of the approximation class to save on storage (see Algorithm 2 for details). The atoms are fit once per task on that task's data and concatenated, which Lemma 2 licenses: the error of the union is bounded by the per-task errors. The coreset and LAP budgets are both set to the cache size of 1.07 GB, allowing 574 points per class for each. Fitting details are in Appendix B.

Results are located in Figure 2 (right). At 1.07GB of storage, LAP is best on both metrics, beating *VILA Analytic* by $+1.17$ and the uniform *coreset* by $+1.15$ in last-task accuracy, with corresponding gains on average incremental accuracy. The coreset comparison isolates the value of the fit itself: with 22% fewer rows, atoms chosen to match the moment tensors outperform rows chosen at random. Against the full-data reference, which stores $5.4\times$ more, LAP nearly recovers its average incremental accuracy (-0.05) and exceeds its last-task accuracy by 0.37.

That last result is unexpected: a model trained on an approximation of the data should not end the stream ahead of one trained on the true data. The simplest explanation, that $4{,}000$ steps are too few to fit 1.1M rows and LAP merely converges faster, is ruled out by Figure 3 (Appendix C): at $48{,}000$ steps the *full data* solution still does not match *LAP* at $4{,}000$. Since the two variants differ only in their training set, we postulate that the low-rank approximation itself acts as a regularizer; Appendix C discusses why this reading is consistent with the split between the two metrics. Whatever the mechanism, the practical reading is unambiguous: *LAP* exceeds its full replay counterpart in last-task accuracy while discarding over 80% of its required storage.

5 Conclusion

We introduce *input anchoring*, a structural property for neural networks under which the mean-squared-error loss of a depth-L network is determined everywhere by two moment tensors of the data, extending the degree-one absolute memorization of analytic class-incremental learning to deep models for the first time, at a storage cost of $O((d+1)^{2L})$. Loss Anchoring Points fits a rank-m CP approximation to these tensors via Gram matrices so that MSE training on its atoms approximates absolute memorization. Empirically, an AMLP head with LAP surpasses the random-feature head of prior ACIL work at matched storage and width, and beats its own full replay counterpart in last-task accuracy with 20% of the storage.

Limitations and Future Work. Theorem 1 is exact only for the full statistics, while LAP is an approximation, and our results are proven for MSE loss only. Our experiments place anchored heads on frozen pre-trained backbones to compare like-for-like with prior analytic work; the theory does not require this, and end-to-end input-anchored models (transformers, CNNs) are a natural next step, as are richer decompositions of the moment tensors (e.g. tensor-train (Kolda & Bader, 2009)).

AI USE STATEMENT

We use artificial intelligence tools to help find related works, convert mathematical notation and diagrams into latex, critique paper drafts, and monitor experiments. We do not rely on AI for understanding of related works, discovery of contributions, or experimental design.

REFERENCES

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 - Mining Discriminative Components with Random Forests. In *European Conference on Computer Vision*, pp. 446–461, 2014. doi: 10.1007/978-3-319-10599-4_29. URL https://mlanthology.org/eccv/2014/bossard2014eccv-food/.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark Experience for General Continual Learning: A Strong, Simple Baseline. In *Neural Information Processing Systems*, 2020. URL https://mlanthology.org/neurips/2020/buzzega2020neurips-dark/.

Yixin Cheng, Grigorios Chrysos, Markos Georgopoulos, and Volkan Cevher. Multilinear Operator Networks. In *International Conference on Learning Representations*, 2024. URL https://mlanthology.org/iclr/2024/cheng2024iclr-multilinear/.

Lénaïc Chizat, Edouard Oyallon, and Francis Bach. On Lazy Training in Differentiable Programming. In *Neural Information Processing Systems*, pp. 2937–2947, 2019. URL https://mlanthology.org/neurips/2019/chizat2019neurips-lazy/.

Grigorios G. Chrysos, Stylianos Moschoglou, Giorgos Bouritsas, Yannis Panagakis, Jiankang Deng, and Stefanos Zafeiriou. P-nets: Deep polynomial neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Grigorios G. Chrysos, Bohan Wang, Jiankang Deng, and Volkan Cevher. Regularization of Polynomial Networks for Image Recognition. In *Conference on Computer Vision and Pattern Recognition*, pp. 16123–16132, 2023. doi: 10.1109/CVPR52729.2023.01547. URL https://mlanthology.org/cvpr/2023/chrysos2023cvpr-regularization/.

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. URL https://mlanthology.org/iclr/2021/dosovitskiy2021iclr-image/.

Jonathan Fiat, Eran Malach, and Shai Shalev-Shwartz. Decoupling gating from linearity, 2019. URL https://arxiv.org/abs/1906.05032.

Jiangpeng He, Zhihao Duan, and Fengqing Zhu. Cl-lora: Continual low-rank adaptation for rehearsal-free class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 30534–30544, June 2025.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The Many Faces of Robustness: A Critical Analysis of Out-of-Distribution Generalization. In *International Conference on Computer Vision*, pp. 8340–8349, 2021. doi: 10.1109/ICCV48922.2021.00823. URL https://mlanthology.org/iccv/2021/hendrycks2021iccv-many/.

Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf.

Joe Kileel, Matthew Trager, and Joan Bruna. On the Expressive Power of Deep Polynomial Neural Networks. In *Neural Information Processing Systems*, pp. 10310–10319, 2019. URL https://mlanthology.org/neurips/2019/kileel2019neurips-expressive/.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015. URL https://mlanthology.org/iclr/2015/kingma2015iclr-adam/.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114. URL https://www.pnas.org/doi/abs/10.1073/pnas.1611835114.

Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3): 455–500, 2009.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, 2013.

Alex Krizhevsky et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Zhizhong Li and Derek Hoiem. Learning Without Forgetting. In *European Conference on Computer Vision*, pp. 614–629, 2016. doi: 10.1007/978-3-319-46493-0_37. URL https://mlanthology.org/eccv/2016/li2016eccv-learning/.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts. In *International Conference on Learning Representations*, 2017. URL https://mlanthology.org/iclr/2017/loshchilov2017iclr-sgdr/.

Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft, 2013. URL https://arxiv.org/abs/1306.5151.

Arun Mallya and Svetlana Lazebnik. PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. doi: 10.1109/CVPR.2018.00810. URL https://mlanthology.org/cvpr/2018/mallya2018cvpr-packnet/.

Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D. Bagdanov, and Joost van de Weijer. Class-incremental learning: Survey and performance evaluation on image classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(5):5513–5533, May 2023. ISSN 0162-8828. doi: 10.1109/TPAMI.2022.3213473. URL https://doi.org/10.1109/TPAMI.2022.3213473.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pp. 109–165. Academic Press, 1989. doi: https://doi.org/10.1016/S0079-7421(08)60536-8. URL https://www.sciencedirect.com/science/article/pii/S0079742108605368.

Mark D. McDonnell, Dong Gong, Amin Parvaneh, Ehsan Abbasnejad, and Anton van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 12022–12053. Curran Associates, Inc., 2023. doi: 10.52202/075280-0526. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/2793dc35e14003dd367684d93d236847-Paper-Conference.pdf.

Liangzu Peng, Juan Elenter, Joshua Agterberg, Alejandro Ribeiro, and Rene Vidal. LoRanPAC: Low-Rank Random Features and Pre-Trained Models for Bridging Theory and Practice in Continual Learning. In *International Conference on Learning Representations*, 2025. URL https://mlanthology.org/iclr/2025/peng2025iclr-loranpac/.

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 *International Conference on Machine Learning*, volume 139, pp. 8748–8763, 2021. URL https://mlanthology.org/icml/2021/radford2021icml-learning/.

Ali Rahimi and Benjamin Recht. Random Features for Large-Scale Kernel Machines. In *Neural Information Processing Systems*, pp. 1177–1184, 2007. URL https://mlanthology.org/neurips/2007/rahimi2007neurips-random/.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental Classifier and Representation Learning. In *Conference on Computer Vision and Pattern Recognition*, 2017. doi: 10.1109/CVPR.2017.587. URL https://mlanthology.org/cvpr/2017/rebuffi2017cvpr-icarl/.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact Solutions to the Nonlinear Dynamics of Learning in Deep Linear Neural Networks. In *International Conference on Learning Representations*, 2014. URL https://mlanthology.org/iclr/2014/saxe2014iclr-exact/.

James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11909–11919, 2023. doi: 10.1109/CVPR52729.2023.01146.

Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild, 2012. URL https://arxiv.org/abs/1212.0402.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL https://mlanthology.org/jmlr/2014/srivastava2014jmlr-dropout/.

Konstantin Usevich, Ricardo Augusto Borsoi, Clara Dérand, and Marianne Clausel. Identifiability of Deep Polynomial Neural Networks. In *Advances in Neural Information Processing Systems*, 2025. URL https://mlanthology.org/neurips/2025/usevich2025neurips-identifiability/.

Gido M. van de Ven, Tinne Tuytelaars, and Andreas S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022. doi: 10.1038/s42256-022-00568-3. URL https://doi.org/10.1038/s42256-022-00568-3.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. Technical report, California Institute of Technology, 2011.

Guanjie Wang, Hongyu Sun, Wanjia Li, and Yanhua Dong. Oracil: Conflict-graph-based order-robust analytic class-incremental learning. *Electronics*, 15(13), 2026. ISSN 2079-9292. doi: 10.3390/electronics15132941. URL https://www.mdpi.com/2079-9292/15/13/2941.

Yan Wang, Da-Wei Zhou, and Han-Jia Ye. Integrating task-specific and universal adapters for pre-trained model-based class-incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 806–816, October 2025.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Dualprompt: Complementary prompting for rehearsal-free continual learning. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (eds.), *Computer Vision – ECCV 2022*, pp. 631–648, Cham, 2022a. Springer Nature Switzerland. ISBN 978-3-031-19809-0.

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 (CVPR)*, pp. 139–149, June 2022b.

Yichen Wu, Hongming Piao, Long-Kai Huang, Renzhen Wang, Wanhua Li, Hanspeter Pfister, Deyu Meng, Kede Ma, and Ying Wei. SD-loRA: Scalable decoupled low-rank adaptation for class incremental learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=5U1rlpX68A.

Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. SUN Database: Large-Scale Scene Recognition from Abbey to Zoo. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3485–3492, 2010. doi: 10.1109/CVPR.2010.5539970. URL https://mlanthology.org/cvpr/2010/xiao2010cvpr-sun/.

Binyu Zhao, Wei Zhang, Xingrui Yu, Zhaonian Zou, and Ivor Tsang. Advancing analytic class-incremental learning through vision-language calibration. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=1sH1AEeylU.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, June 2018. ISSN 0162-8828. doi: 10.1109/tpami.2017.2723009. URL https://repositorio.unal.edu.co/handle/unal/82524.

Da-Wei Zhou, Zi-Wen Cai, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need. *International Journal of Computer Vision*, 133(3), 2024a. ISSN 0920-5691. doi: 10.1007/s11263-024-02218-0. URL https://doi.org/10.1007/s11263-024-02218-0.

Da-Wei Zhou, Hai-Long Sun, Jingyi Ning, Han-Jia Ye, and De-Chuan Zhan. Continual learning with pre-trained models: A survey. In Kate Larson (ed.), *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pp. 8363–8371. International Joint Conferences on Artificial Intelligence Organization, 8 2024b. doi: 10.24963/ijcai.2024/924. URL https://doi.org/10.24963/ijcai.2024/924. Survey Track.

Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Class-incremental learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):9851–9873, 2024c. doi: 10.1109/TPAMI.2024.3429383.

Da-Wei Zhou, Kai-Wen Li, Jingyi Ning, Han-Jia Ye, Lijun Zhang, and De-Chuan Zhan. External knowledge injection for clip-based class-incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3314–3325, October 2025.

Huiping Zhuang, Zhenyu Weng, Hongxin Wei, Renchunzi Xie, Kar-Ann Toh, and Zhiping Lin. Acil: Analytic class-incremental learning with absolute memorization and privacy protection. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 11602–11614. Curran Associates, Inc., 2022. doi: 10.52202/068431-0843. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/4b74a42fc81fc7ee252f6bcb6e26c8be-Paper-Conference.pdf.

Huiping Zhuang, Run He, Kai Tong, Ziqian Zeng, Cen Chen, and Zhiping Lin. DS-AL: A dual-stream analytic learning for exemplar-free class-incremental learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(15):17237–17244, March 2024. doi: 10.1609/aaai.v38i15.29670. URL https://ojs.aaai.org/index.php/AAAI/article/view/29670.

A PROOFS

Proof of Proposition 1.

Proof. ($\Leftarrow$) Each term $\left\langle g(\theta, A)_c, \bigotimes_{\ell \in A} v_\ell \right\rangle$ is linear in every v_ℓ with $\ell \in A$ and constant in every v_ℓ with $\ell \notin A$, so the sum over $A \subseteq [L]$ is affine in each argument separately, and it is smooth in θ because each $g(\theta, A)$ is.

($\Rightarrow$) We argue by induction on L. For $L = 1$, a function affine in v_1 is $a + \langle b, v_1 \rangle$ for unique $a \in \mathbb{R}$ and $b \in \mathbb{R}^{k_1}$. For $L > 1$, affinity in v_L gives $F_\theta(v_1, \ldots, v_L)_c = a_c(v_1, \ldots, v_{L-1}) + \langle b_c(v_1, \ldots, v_{L-1}), v_L \rangle$, where a_c and each coordinate of b_c are affine in each of $v_1, \ldots, v_{L-1}$ separately (fix v_L at 0 and at basis vectors to see this). Applying the inductive hypothesis to a_c and to each coordinate of b_c yields the claimed expansion, with the terms of a_c contributing the subsets $A \not\ni L$ and the terms of b_c contributing the subsets $A \ni L$.

Uniqueness follows because the monomials $\{v \mapsto \bigotimes_{\ell \in A} v_\ell\}_{A \subseteq [L]}$ are linearly independent as functions on $\mathbb{R}^{k_1} \times \cdots \times \mathbb{R}^{k_L}$: they are distinct multivariate monomials, of degree at most one in each coordinate. Since Φ ranges over all anchor functions, the expansion of F_θ in the v_ℓ transfers verbatim to $f_{\theta,\Phi}(x) = F_\theta(\phi_1(x), \ldots, \phi_L(x))$.

Smoothness of each $g(\theta, A)$ in θ follows because every coordinate of $g(\theta, A)$ is a finite signed sum of values of F_θ at fixed points (obtained by evaluating F_θ at $v_\ell \in \{0, e_{i_\ell}\}$ and applying inclusion–exclusion over A), and F_θ is smooth in θ.

Finally, differentiating the expansion once in each of $v_1, \ldots, v_L$ kills every term with $A \neq [L]$ and leaves $g(\theta, [L])$, so $\frac{\partial^L F_\theta}{\partial v_1 \cdots \partial v_L} = g(\theta, [L])$ and the two forms of (IA2) coincide. □

Corollary 1 (Additivity)**.** If $\mathcal{D}_t$ is the dataset of the tth task in a CIL sequence with statistics $\mathcal{M}^{(t)} := \mathbb{E}_{\mathcal{D}_t}[P(x, \Phi)^{\otimes 2}]$ and $\mathcal{V}^{(t)}$, then $\mathcal{M} = \frac{1}{T}\sum_{t=1}^{T} \mathcal{M}^{(t)}$ and $\mathcal{V} = \frac{1}{T}\sum_{t=1}^{T} \mathcal{V}^{(t)}$.

Proof. Linearity of expectation applied to equation 2, using the equal-size assumption on tasks from §3. □

Proof of Proposition 2.

Proof. By definition, $P(x, \Phi)$ has dimension $\sum_{A \subseteq [L]} \prod_{\ell \in A} k_\ell$. This is $\sum_{A \subseteq [L]} d^{|A|} = \sum_{\ell=0}^{L} \binom{L}{\ell} d^\ell$. By the binomial theorem, we have this is $(1 + d)^L$. The rest follows directly. □

Proof of Proposition 3: The AMLP is input-anchored.

Proof. We show this via Proposition 1 constructively. Recognize that, for $g(\theta, A) := 0$ for $A \subset [L]$ and $g(\theta, [L])_i := (W_1)_{i_1, i_0}(W_2)_{i_2, i_1} \ldots (W_L)_{i_L, i_{L-1}}$, that the AMLP satisfies (IA1) (it satisfies the Frobenius inner product equality, and it is smooth in θ). Then recognize that we can construct in g via θ all elements of the standard basis of $\mathcal{H}$, defined by the index of the one entry via some zero indexed $e \in [C] \times [k_1] \times [k_2] \times \cdots \times [k_L]$, via

$$(W_j)_{i_{j-1}, i_j} = 1 \text{ if } i_{j-1} = e_{j-1} \text{ and } i_j = e_j \text{; else } 0$$

Because we can construct any element of the standard basis, the span of $g(\cdot, [L])_c$ is $\mathcal{H}$, and AMLP satisfies (IA2). □

Proof of Proposition 4: CCP is input-anchored.

Proof. Write the CCP with linear maps $U_1, \ldots, U_L \in \mathbb{R}^{d \times k}$ and output map $W \in \mathbb{R}^{C \times k}$, so that $z_1 = U_1^\top x$, the $L - 1$ Hadamard layers give $z_n = (U_n^\top x) \odot z_{n-1} + z_{n-1} = (\mathbf{1} + U_n^\top x) \odot z_{n-1}$ for $2 \leq n \leq L$, and $f(x) = W z_L$. Unrolling the recursion and expanding the product,

$$f(x)_c \;=\; \sum_{i \in [k]} W_{c,i}\,(U_1^\top x)_i \prod_{n=2}^{L} \big(1 + (U_n^\top x)_i\big) \;=\; \sum_{S \subseteq \{2, \ldots, L\}} \;\sum_{i \in [k]} W_{c,i}\,(U_1^\top x)_i \prod_{n \in S} (U_n^\top x)_i.$$

Take L anchor functions, all equal to the identity, $\phi_\ell(x) = x$ with $k_\ell = d$. For $A = \{1\} \cup S$ the inner sum is $\big\langle g(\theta, A)_c, \bigotimes_{\ell \in A} x \big\rangle$ with $g(\theta, A)_c := \sum_{i \in [k]} W_{c,i}\,(U_1)_{:,i} \otimes \bigotimes_{n \in S} (U_n)_{:,i}$, and $g(\theta, A) := 0$ for every $A \not\ni 1$. Each $g(\theta, A)$ is polynomial, hence smooth, in $\theta = (W, U_1, \ldots, U_L)$, so the CCP satisfies (IA1) by Proposition 1 with $\mathcal{A} = \{A \subseteq [L] : 1 \in A\}$, $|\mathcal{A}| = 2^{L-1}$.

For (IA2), the top-degree coefficient is $g(\theta, [L])_c = \sum_{i \in [k]} W_{c,i}\,(U_1)_{:,i} \otimes \cdots \otimes (U_L)_{:,i}$. Setting $W_{c,1} = 1, W_{c,i} = 0$ for $i \neq 1$, and choosing the first columns of $U_1, \ldots, U_L$ freely realises every rank-one tensor $a_1 \otimes \cdots \otimes a_L \in \mathcal{H}$, and the rank-one tensors span $\mathcal{H}$. Hence $\operatorname{span}\{g(\theta, [L])_c : \theta \in \Theta\} = \mathcal{H}$ for every c. □

Proof of Lemma 1: Algorithm 1 computes the distance to the true tensors.

Proof. We have that

$$||\hat{\mathcal{M}}_{\mathcal{A}} - \mathcal{M}_{\mathcal{A}}||^2 = ||\hat{\mathcal{M}}_{\mathcal{A}}||^2 - 2\left\langle \hat{\mathcal{M}}_{\mathcal{A}}, \mathcal{M}_{\mathcal{A}} \right\rangle + ||\mathcal{M}_{\mathcal{A}}||^2 \tag{5}$$

Let $K_A := \prod_{\ell \in A}[k_\ell]$ be the set of indices for a given tensor $g(\cdot, A)$.

Then,

$$\begin{aligned}
||\hat{\mathcal{M}}_{\mathcal{A}}||^2 &= \frac{1}{m^2} \sum_{A,A' \in \mathcal{A}} \sum_{v \in K_A} \sum_{v' \in K_{A'}} \left(\sum_{i \in [m]} \prod_{\ell \in A} \phi_\ell(\hat{x}_i)_{v_\ell} \prod_{\ell' \in A'} \phi_{\ell'}(\hat{x}_i)_{v'_{\ell'}} \right)^2 \\
&= \frac{1}{m^2} \sum_{A,A' \in \mathcal{A}} \sum_{v \in K_A} \sum_{v' \in K_{A'}} \sum_{i,i' \in [m]} \prod_{\ell \in A} \phi_\ell(\hat{x}_i)_{v_\ell} \prod_{\ell' \in A'} \phi_{\ell'}(\hat{x}_i)_{v'_{\ell'}} \prod_{\ell \in A} \phi_\ell(\hat{x}_{i'})_{v_\ell} \prod_{\ell' \in A'} \phi_{\ell'}(\hat{x}_{i'})_{v'_{\ell'}} \\
&= \frac{1}{m^2} \sum_{i,i' \in [m]} \left(\sum_{A \in \mathcal{A}} \sum_{v \in K_A} \prod_{\ell \in A} \phi_\ell(\hat{x}_i)_{v_\ell} \phi_\ell(\hat{x}_{i'})_{v_\ell} \right) \left(\sum_{A' \in \mathcal{A}} \sum_{v' \in K_{A'}} \prod_{\ell' \in A'} \phi_{\ell'}(\hat{x}_i)_{v'_{\ell'}} \phi_{\ell'}(\hat{x}_{i'})_{v'_{\ell'}} \right) \\
&= \frac{1}{m^2} \sum_{i \in [m]} \sum_{i' \in [m]} \kappa(\hat{x}_i, \hat{x}_{i'})^2
\end{aligned}$$

Likewise,

$$||\mathcal{M}_{\mathcal{A}}||^2 = S_{\mathcal{M}}, \qquad \left\langle \hat{\mathcal{M}}_{\mathcal{A}}, \mathcal{M}_{\mathcal{A}} \right\rangle = \frac{1}{mN} \sum_{i \in [m]} \sum_{j \in [N]} \kappa(\hat{x}_i, x_j)^2$$

Therefore, $||\hat{\mathcal{M}}_{\mathcal{A}} - \mathcal{M}_{\mathcal{A}}||^2 = J_{\mathcal{M}}$.

The $||\hat{\mathcal{V}}_{\mathcal{A}} - \mathcal{V}_{\mathcal{A}}||^2$ term follows by the same expansion with $\kappa(\cdot,\cdot)^2$ replaced by $\langle y, y' \rangle \, \kappa(\cdot,\cdot)$ throughout, yielding $J_{\mathcal{V}}$. □

Proof of Lemma 2: LAP approximations stack over task boundaries.

Proof. By Corollary 1, $\mathcal{M}_{\mathcal{A}} = \frac{1}{T}\sum_{t=1}^{T} \mathcal{M}_{\mathcal{A}}^{(t)}$, and similarly $\mathcal{M}_{\mathcal{A}} - \hat{\mathcal{M}}_{\mathcal{A}} = \frac{1}{T}\sum_{t=1}^{T} \mathcal{M}_{\mathcal{A}}^{(t)} - \hat{\mathcal{M}}_{\mathcal{A}}^{(t)}$. By the convexity of $\|\cdot\|_F^2$, $J_{\mathcal{M}}(\mathcal{D}, \hat{\mathcal{D}}) := \left\| \mathcal{M}_{\mathcal{A}} - \hat{\mathcal{M}}_{\mathcal{A}} \right\|_F^2 = \left\| \frac{1}{T}\sum_{t=1}^{T} \mathcal{M}_{\mathcal{A}}^{(t)} - \hat{\mathcal{M}}_{\mathcal{A}}^{(t)} \right\|_F^2 \leq \frac{1}{T}\sum_{t=1}^{T} \left\| \mathcal{M}_{\mathcal{A}}^{(t)} - \hat{\mathcal{M}}_{\mathcal{A}}^{(t)} \right\|_F^2$. The same argument applies to each $\hat{\mathcal{V}}_c$, and summing the two bounds yields the claim. □

Proof of Theorem 2: LAP recovers absolute memorization when realizable.

Proof. If a dataset $\hat{\mathcal{D}}$ with at most m data-points exists which produces the same moment tensors as the true dataset, $\mathcal{D}$, then there exists a dataset which achieves $J_t = 0$. By Lemma 1 then, there must exist a choice of outputs from Algorithm 1 which achieve $J_t = 0$, which is the minimum possible value for J_t as it is a sum of Frobenius norms. By Lemma 2, stacking these datasets achieves error across all datasets of 0, and, therefore, by Theorem 1 applied to $\hat{\mathcal{D}}$, training over the union with MSE loss on $f_{\theta,\Phi}$ is identical to training over the full dataset. □

Proposition 5 (Complexity of Algorithm 1)**.** *Let $k_1 = \cdots = k_L := d$, $C = O(d)$, and $m \leq N$, $d \leq N$. Then Algorithm 1 runs in*

$$O\big(\, |\mathcal{A}|\, L\, d\, N^2 \;+\; t_{\textit{budget}}\, |\mathcal{A}|\, L\, d\, m\, N \,\big)$$

and uses $O(NLd)$ working memory beyond the $m(d+C)$ floats of the atoms. In particular, no object of size $(1+d)^L$ or larger is ever materialised.

Algorithm 2 Class-incremental training with an anchored head and a Loss Anchoring Point memory

Require: a task stream $\mathcal{T}_1, \ldots, \mathcal{T}_T$ with disjoint class sets $\mathcal{C}_1, \ldots, \mathcal{C}_T$ and data sets $\mathcal{D}_1, \ldots, \mathcal{D}_T$, number of classes per group G, number of points per group m, number of steps per task $t_{\text{train steps}}$, Adam hyperparameters, and Algorithm 1 hyperparameters
Ensure: head weights $\theta_T = (W_1, W_{\text{out}})$ trained over all T tasks

Follow VILA's Setup including task-1 tuning of the adapter, but use a matched width and depth AMLP head instead of a random projection and linear head
initialise the memory $\hat{\mathcal{D}} \leftarrow \emptyset$
for task 1: $\theta \leftarrow \mathrm{Adam}(\theta, \nabla_\theta \mathcal{L}_{\text{MSE}}(\mathcal{D}_1))$ for $t_{\text{train steps}}$ ▷ task 1 trains on its real rows
for $t = 2, \ldots, T$ **do**
    divide the incoming data by class into $|C_t|/G$ groups
    for each group **do** ▷ one independent fit per class group for efficiency
        $\hat{\mathcal{D}}_G,\ J_G \leftarrow$ Algorithm 1$\big(f_\theta,\ \Phi,\ \mathcal{D}_G,\ m, |G|,\ t_{\text{budget}}\big)$
        keep only labels corresponding to current group's classes
        $\hat{\mathcal{D}} \leftarrow \hat{\mathcal{D}} \cup \hat{\mathcal{D}}_G$
    end for
    discard $\mathcal{D}_t$
    $\theta \leftarrow \mathrm{Adam}(\theta, \nabla_\theta \mathcal{L}_{\text{MSE}}(\hat{\mathcal{D}}))$ for $t_{\text{train steps}}$
end for
return θ_T

Proof. Cache the anchor features $\phi_\ell(x_j)$ for every $\ell \in [L]$ and $j \in [N]$, which occupies $O(NLd)$ floats and is the only object whose size depends on N. Given cached features, one evaluation of $\kappa(a, b)$ costs $O(|\mathcal{A}|Ld)$: for each $A \in \mathcal{A}$, a product of $|A| \leq L$ inner products of length d. The constants $S_{\mathcal{M}}$ and $S_{\mathcal{V}}$ require κ on all N^2 pairs (the $\langle y_j, y_{j'} \rangle$ factors cost $O(C) = O(d)$ each), giving the first term. Each step of the loop evaluates κ on $m^2 + mN \leq 2mN$ pairs, together with the corresponding gradients with respect to $\hat{x}_i$ and $\hat{y}_i$, which cost a constant factor more by reverse-mode differentiation; over t_{budget} steps this gives the second term. No quantity indexed by a subset A is ever formed as a tensor: κ contracts each factor $\langle \phi_\ell(a),\ \phi_\ell(b) \rangle$ to a scalar before multiplying. □

B Experimental Details

Hyperparameter sweeps. For experiments that run gradient descent, dropout is selected by a linear sweep over $[0, 0.9]$ with a step size of 0.1 at learning rate $3e-4$, and then learning rate is swept over $\{3e-6, 1e-5, 3e-5, 1e-4, 3e-4, 1e-3, 3e-3, 1e-2\}$ at the best performing dropout, all on CIFAR-100 with $T = 20$. The selected values are reported in §4.1 and used unchanged for every experiment.

Choice of gate. We call a step gate the most comparable AMLP activation to ReLU, rather than ReLU itself, following the logic established by Fiat et al. (2019): $\mathrm{ReLU}(Wx)_i = \mathrm{step}(W_i \cdot x) \cdot (W_i \cdot x)$; whereas, an AMLP with step activation computes $\mathrm{step}(F_i \cdot x) \cdot W_i \cdot x$. The sigmoid with a high multiplier, $s(32x)$, approaches this step function while remaining continuously differentiable, which Algorithm 1 requires for its gradients with respect to $\hat{x}$.

Training and fitting procedure.

C Ablations

Experiment 3 asks two questions about the head in isolation, on CIFAR-100 ($T = 20$) with full replay: whether input anchoring costs accuracy relative to an unconstrained ReLU head of the same width, and whether the choice of anchored gate matters. Figure 4 sweeps head dropout for five heads. *step*, *identity*, and *sigmoid* are AMLPs with σ set to $\mathbb{1}(x > 0)$, the identity, and $s(32x)$ respectively; with the step gate the AMLP is exactly a GaLU network (Fiat et al., 2019). *ReLU head* is a standard two-layer ReLU MLP of the same width and trainable parameter count. *VILA GD* is the same as in Experiment 1.

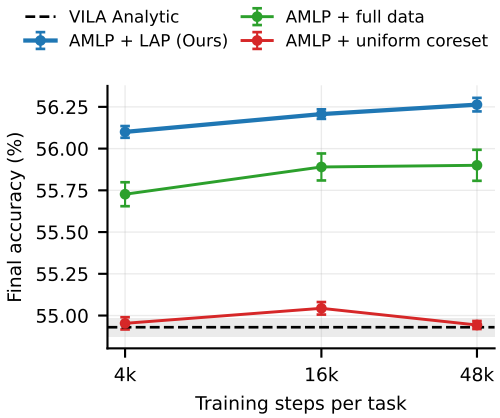


Figure 3: Final accuracy against head-training steps per task, mean ± s.e. over 3 seeds. LAP's advantage over the full data solution is not an artifact of small step budget.

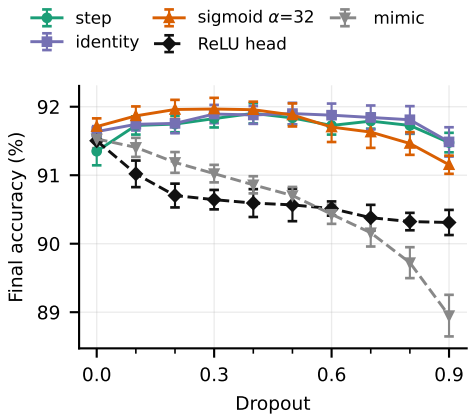


Figure 4: Final test accuracy on CIFAR-100 (20 tasks, full data) against head dropout, mean ± s.e. over 3 seeds. The three input-anchored gates hold or improve under dropout; both ReLU heads decay from their optimum at zero.

On the first question, none of the anchored gates is worse than the ReLU head at any dropout, and at their respective optima all three exceed it. Input anchoring restricts the head to a multi-affine form, but on this benchmark the restriction is free; in particular the head does not collapse to a linear regression over the dL anchor features. On the second, the three anchored gates are within noise of one another across the sweep, which is why we fix the sigmoid gate for its differentiability rather than tuning it. The sweep also shows that regularization interacts with the two families differently: the anchored heads improve under moderate dropout and peak near 0.3, whereas both ReLU heads are best at zero and decay monotonically.

LAP as a regularizer: the metric split. In Experiment 2, *LAP* exceeds *full data* in last-task accuracy but trails it slightly in average incremental accuracy. This split is consistent with the regularization reading offered in §4.3: average incremental accuracy weights early tasks, when the head has capacity to spare and regularization has little to offer, whereas last-task accuracy is measured with all 365 classes competing, which is where regularization has the most to offer. Figure 3 rules out the alternative explanation that the full-data run is simply under-trained.

D Full Results for Experiment 1

Table 2 reproduces the full table of Zhao et al. (2026), extended with our runs, including the eight published non-analytic baselines omitted from Table 1 and average incremental accuracy $\bar{A}$ per benchmark.

Table 2: Full version of Table 1: extension of Zhao et al. (2026)'s Table 2 of last-task accuracy A_T (%) and average incremental accuracy $\bar{A}$ (%) among PTM-based methods across diverse benchmarks. Rows above each inner rule are from their published table; rows below it are our runs on their released code. T denotes the total number of tasks. The best and second-best results are highlighted in bold and blue, respectively.

Metric	Dataset	CIFAR		ImageNet-R		CUB	UCF	Aircraft		Cars	Food	SUN		Avg.
	$T =$	10	20	20	40	20	20	10	20	20	20	30	50	
A_T	L2P	84.29	77.70	70.13	65.07	64.21	72.41	28.74	18.90	20.52	70.03	57.41	54.98	57.03
	DualPrompt	82.30	77.23	65.98	60.13	65.95	76.17	25.50	10.14	18.66	69.22	62.23	62.03	56.29
	CODA-Prompt	86.85	79.59	70.33	61.02	68.45	76.09	27.24	15.54	15.31	72.32	64.88	58.49	58.01
	APER-Adapter	85.80	82.77	70.52	66.70	84.68	86.66	30.96	30.93	41.69	77.98	72.51	72.46	66.97
	TUNA	91.09	90.42	77.33	75.55	88.46	94.51	54.49	47.52	54.05	79.40	75.09	75.30	75.27
	SD-LoRA	87.50	82.28	71.20	63.27	70.46	73.29	40.29	23.76	35.49	42.91	54.06	39.21	56.98
	CL-LoRA	88.17	84.91	78.40	73.50	73.92	79.23	45.57	26.76	40.46	76.42	71.34	69.27	67.33
	ENGINE	79.29	79.10	80.47	80.03	78.54	87.91	58.69	55.51	89.49	83.88	78.16	77.99	77.42
	RanPAC	86.40	86.36	71.83	71.70	88.93	97.73	59.23	58.33	66.81	84.56	79.65	79.75	77.61
	LoRanPAC	91.32	90.34	74.95	73.42	**90.25**	98.22	59.38	59.32	70.17	86.84	80.77	80.76	79.64
	VILA Analytic (Published)	91.94	91.18	85.28	84.20	89.36	98.71	68.02	65.56	90.28	88.89	83.73	83.55	85.06
	VILA Analytic (Reproduced)	92.24	91.40	85.31	84.32	89.35	98.67	66.93	66.14	90.25	88.96	83.39	83.34	85.02
	VILA Analytic (Double Features)	92.46	91.67	86.00	84.73	89.20	98.84	67.05	66.47	**90.82**	89.41	83.97	84.15	85.40
	VILA GD	92.12	91.31	84.81	83.27	88.82	98.57	64.00	62.82	90.41	88.91	83.51	83.53	84.34
	AMLP + VILA (Ours)	**92.69**	**91.87**	**86.51**	**85.81**	89.61	**98.87**	**68.51**	**67.91**	90.52	**89.71**	**84.09**	**84.20**	**85.86**
$\bar{A}$	L2P	89.23	84.20	75.86	72.04	75.93	83.58	41.79	32.32	29.92	77.62	68.54	66.45	66.46
	DualPrompt	87.36	84.85	72.53	68.23	76.44	86.05	40.12	23.57	31.05	79.58	73.64	73.17	66.38
	CODA-Prompt	91.30	86.17	75.93	66.48	76.57	84.87	30.50	19.44	33.08	81.39	76.95	72.07	66.23
	APER-Adapter	90.91	88.49	77.20	74.02	91.29	91.18	39.92	41.36	55.62	85.14	80.33	80.43	74.66
	TUNA	94.60	94.38	82.59	80.77	93.18	97.49	63.18	58.75	69.05	88.11	83.92	84.32	82.53
	SD-LoRA	92.31	88.89	73.82	65.05	76.87	84.29	57.29	39.22	48.28	21.40	35.45	25.52	59.03
	CL-LoRA	92.41	90.87	84.83	80.63	82.82	89.25	59.71	43.77	56.43	84.47	79.66	77.74	76.88
	ENGINE	86.87	87.12	86.21	85.98	86.32	92.97	69.69	68.79	94.03	89.86	85.04	85.09	84.83
	RanPAC	90.93	91.27	77.71	77.32	92.97	98.83	68.10	68.29	67.77	90.05	86.04	86.12	82.95
	LoRanPAC	94.60	93.90	79.90	79.02	**93.48**	99.19	67.99	69.27	80.04	91.70	86.74	86.89	85.23
	VILA Analytic (Published)	95.09	94.72	89.81	89.16	93.16	99.35	76.26	75.74	**94.19**	**93.32**	89.43	89.53	89.98
	VILA Analytic (Reproduced)	95.12	94.69	89.55	89.06	92.72	99.28	76.91	76.77	93.62	92.77	89.72	89.76	90.00
	VILA Analytic (Double Features)	95.15	94.74	90.00	89.32	92.52	**99.41**	76.78	77.13	93.93	92.95	89.94	90.17	90.17
	VILA GD	95.00	94.66	89.26	88.02	92.28	99.26	75.55	75.49	93.93	92.69	89.71	89.92	89.65
	AMLP + VILA (Ours)	**95.32**	**94.86**	**90.34**	**89.91**	92.74	99.40	**77.62**	**77.81**	93.77	93.21	**90.07**	**90.25**	**90.44**