InTAct: Interval-based Task Activation Consolidation for Continual Learning

Patryk Krukowski[*]
Doctoral School of Exact and Natural Sciences
Jagiellonian University
Akces NCBR

Jan Miksa
Doctoral School of Exact and Natural Sciences
Jagiellonian University
IDEAS Research Institute

Piotr Helm
Doctoral School of Exact and Natural Sciences
Jagiellonian University

Jacek Tabor
Jagiellonian University
Centre for Credible AI

Paweł Wawrzyński
IDEAS Research Institute

Przemysław Spurek
Jagiellonian University
IDEAS Research Institute

Abstract

Continual learning seeks to acquire new knowledge while preserving previously learned representations. Despite the success of prompt-based methods, they remain fragile in domain-incremental learning scenarios, where shifts in the input distribution cause representation drift in shared layers and lead to forgetting. We introduce InTAct[2], a method that preserves functional behavior in shared representations without freezing parameters or retaining past data. InTAct constrains updates within activation regions associated with prior domains, while allowing flexible adaptation elsewhere, thereby stabilizing neuron functionality rather than directly restricting parameter values. The approach is architecture-agnostic, integrates seamlessly with existing prompt-based frameworks, and matches or outperforms state-of-the-art methods on domain-incremental benchmarks.

1 Introduction

Continual learning aims to enable neural networks to acquire new knowledge while retaining previously learned information. This requires balancing stability, the retention of past knowledge, with plasticity, the ability to adapt to new tasks. Unlike humans, deep neural networks often struggle to maintain this balance [1]. As they adapt to new data distributions, their internal representations may drift, altering features important for earlier tasks and ultimately leading to catastrophic forgetting [2].

A large body of prior work addresses this issue by constraining parameter updates through regularization, architectural modifications, or optimization-based strategies [3, 4, 5]. While effective to some extent, these methods predominantly operate in the parameter space and rarely provide explicit control over how the network's function evolves across tasks. Moreover, despite the large number of continual learning strategies, most existing approaches lack rigorous mathematical guarantees against forgetting. Methods that do provide formal assurances typically rely on *interval arithmetic* (IA) in the weight space, as exemplified by frameworks such as InterContiNet [6] and HINT [7].

[*]patryk.krukowski@doctoral.uj.edu.pl
[2]https://github.com/pkrukowski1/Prompt_InTAct

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

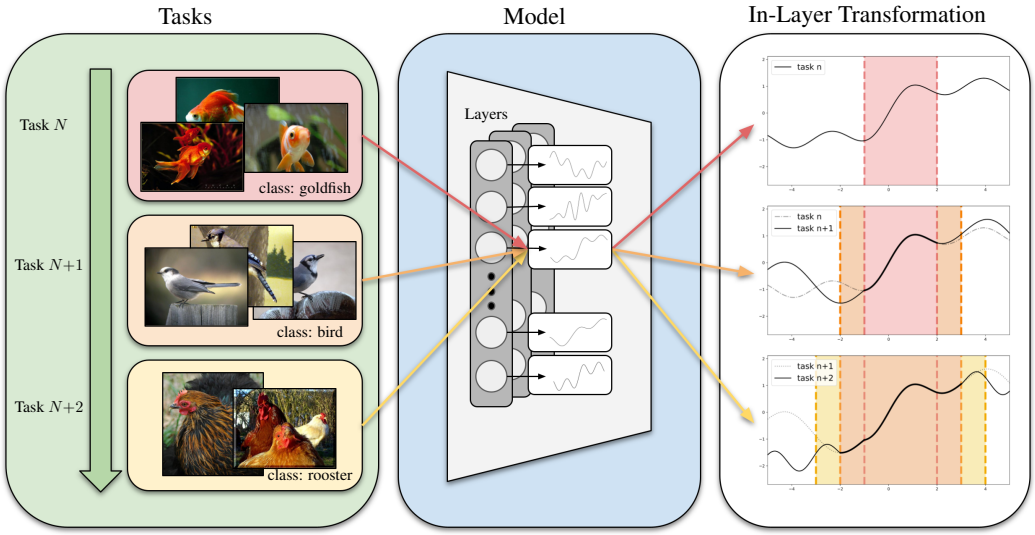


Figure 1: **Method Overview.** During training on task $(N+1)$, InTAct preserves the functional region established by task N (pink) while allowing the model to learn new knowledge in a distinct region (orange). After training, these regions merge into an expanded protected region (yellow), defining where the layer's transformation should remain stable in future updates. The dashed curves illustrate transformations from earlier tasks, highlighting that their behaviors remain preserved within their respective regions, while adaptation is freely allowed outside them.

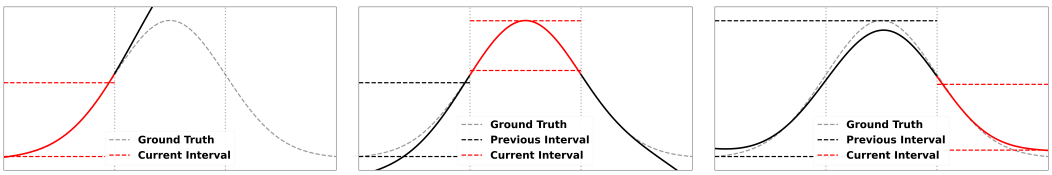
However, constraining high-dimensional parameter updates is computationally expensive and does not necessarily ensure the model's functional stability.

To address these limitations, we introduce InTAct (Interval-based Task Activation Consolidation), a novel approach that enforces functional stability by directly regulating neural activations rather than model parameters. InTAct identifies neuron-wise activation intervals that characterize stable operating regions established during previous tasks. These intervals are aggregated at the layer level to form multidimensional hypercubes, which delineate regions of the feature space corresponding to previously acquired knowledge (see Fig. 1). During subsequent training, InTAct constrains model updates within these regions while permitting unrestricted adaptation outside them. By enforcing prediction invariance whenever internal signals remain within the preserved activation intervals, InTAct provides a tractable and efficient layer-wise mathematical guarantee of functional stability, which avoids the excessive conservativeness of end-to-end interval propagation.

When learning new tasks, the resulting hypercubes act as elastic boundaries: they encourage the model to remain within trusted regions for familiar concepts, while still allowing exploration beyond these regions to accommodate novel information. This mechanism naturally balances stability and plasticity. Crucially, instead of restricting the model's parameters, InTAct directly regulates activations, which constitute the model's observable functional behavior. This results in a more direct and effective control over how the network evolves over time.

Although the proposed framework is architecture-agnostic and applicable to a wide range of continual learning scenarios, its integration with prompt-based continual learning methods [8, 9] proves particularly effective. In this setting, InTAct achieves state-of-the-art (SOTA) performance on challenging domain-incremental benchmarks. Our evaluations confirm that InTAct effectively stabilizes internal representations and consistently surpasses baselines without the need for data replay. The main contributions of this work are:

1. A novel continual learning framework, InTAct, which enforces functional stability by constraining neuron activations through interval-based representations rather than parameter-level restrictions.



(a) Task 1: Initial learning and interval computation.

(b) Task 2: Adaptation with stability constraints.

(c) Task 3: Global accumulation without forgetting.

Figure 2: **Illustration of localized functional preservation in InTAct.** A model is trained sequentially on three tasks to approximate different regions of a Gaussian-shaped function. The dashed lines denote the protected intervals (1D hypercubes) estimated after each task. During subsequent training stages, these intervals constrain functional drift within previously occupied regions while allowing the function to adapt freely in unexplored regions of the input space. As a result, InTAct preserves prior knowledge locally without restricting plasticity globally.

2. We introduce an IA-based formulation together with a formal layer-wise stability theorem showing that InTAct preserves functional behavior locally within protected regions of the feature space while maintaining plasticity outside these regions.
3. InTAct seamlessly integrates with prompt-based architectures, achieving SOTA performance on domain-incremental benchmarks without replay or parameter regularization.[3]

2 Related Works

For an extended overview, see Appendix A.

Continual Learning. Traditional continual learning methods follow three main paradigms. *Replay-based* approaches [10, 11, 12] retain exemplars or generate synthetic data [13] to approximate past distributions, at the cost of additional memory. *Regularization-based* methods [1, 14, 15] preserve previous knowledge by constraining parameter updates through penalties or projections, but often struggle to maintain stability over long task sequences. Finally, *parameter-isolation* approaches [16, 17, 18] allocate task-specific parameters or subspaces, often relying on task IDs at inference time.

Prompt-Based Continual Learning. Recent architectures leverage frozen pre-trained backbones, directing adaptation via learnable prompts. L2P [8] and DualPrompt [9] utilize a pool of keys and prompts to query task-specific knowledge. CODA-Prompt [19] extends this paradigm via attention-based decomposition, while C-Prompt [20] shares compositional prompt components across domains. Despite their success, these methods remain vulnerable to *representational drift* in shared components, particularly prompt keys and the classification head. This drift is especially detrimental in domain-incremental learning (DIL), where task boundaries are inherently ambiguous. KA-Prompt [21] observes that prompts learned for different domains are misaligned component-wise, so fusing them across domains mixes conflicting knowledge. To address this, it initializes new prompts from relevant historical ones identified via greedy search and encourages consistency with them during training.

Interval Arithmetic in Continual Learning. While IA [22, 23] offers a principled way to obtain rigorous numerical bounds, its use in continual learning remains largely unexplored. Existing methods primarily apply IA in the *parameter space*. InterContiNet [6] identifies safe weight intervals for multi-task compatibility, while HINT [7] uses hypernetworks to map interval-based embeddings to network parameters. However, propagating intervals through deep networks suffers from the *wrapping effect* [23], which causes bounds to expand with network depth and become excessively loose. Rather than performing interval propagation, InTAct uses IA locally to construct static *hypercubes* around representations of previously learned tasks. These bounds are then used to regularize subsequent representations, promoting functional stability without the computational cost and bound explosion of full interval propagation.

[3] Integration details are provided in Appendix K.

3 Method

InTAct mitigates forgetting by protecting geometric regions of the feature space instead of regularizing model weights. Specifically, it constructs deterministic bounding boxes around historical representations, guaranteeing invariant classifier outputs within protected regions while preserving representation consistency across tasks.

Conceptual Overview. Fig. 2 illustrates the intuition behind InTAct. After learning Task 1, the method constructs a protected region around its representations. During training on Task 2, InTAct enforces functional invariance within this region while allowing unconstrained adaptation outside it. As a result, the model can acquire new knowledge in previously unused regions of the feature space without disrupting prior tasks, naturally balancing stability and plasticity.

Our framework is designed for prompt-based continual learning, where learnable prompts adapt a frozen backbone to produce task-specific representations. Although the backbone itself remains fixed, both the prompt pool and the shared classifier evolve continuously across tasks, making the model susceptible to representation drift and catastrophic forgetting. InTAct mitigates this effect by directly constraining the geometry of the intermediate feature space. Since final predictions depend entirely on these representations, the proposed framework jointly preserves classifier behavior within historical feature regions while anchoring prompts to ensure that previously learned samples remain mapped to these protected regions. This design preserves prior knowledge while maintaining the flexibility required to learn new tasks, as confirmed by our experimental results. Importantly, the proposed activation-space constraints are architecture-agnostic and can be easily integrated into existing continual learning frameworks (see Appendix K).

3.1 Activation Interval Representation

To preserve historical knowledge without storing replay samples, we construct a compact geometric representation of the feature distribution. This representation is applied both to the `[CLS]` token embeddings and to intermediate classifier activations.

Let $\mathcal{Z}_t \subset \mathbb{R}^d$ denote the set of feature representations extracted from task t. Rather than modeling the full distribution explicitly, we approximate its support using a quantile-based hypercube that captures the dominant activation region while reducing the influence of extreme values. Specifically, we retain the central part of the distribution and exclude the tails beyond the $(1-\alpha)$- and α-quantiles, where $\alpha \in (0.5, 1]$. Unless stated otherwise, we set $\alpha = 0.95$.

For each feature dimension j, we define the task-specific activation interval $[\underline{z}_{t,j}, \overline{z}_{t,j}]$. The corresponding task-specific hypercube is obtained as the Cartesian product of these intervals:

$$\mathcal{H}_t = \left[\underline{\mathbf{z}}_t, \overline{\mathbf{z}}_t\right] = \prod_{j=1}^{d} \left[\underline{z}_{t,j}, \overline{z}_{t,j}\right],$$

where $\underline{z}_{t,j}$ and $\overline{z}_{t,j}$ are the $(1-\alpha)$- and α-quantiles, respectively, of the activations $\{z_j : \mathbf{z} \in \mathcal{Z}_t\}$ along dimension j. Throughout the paper, we use the term *hypercube* to refer to this axis-aligned box, which may have different side lengths across dimensions. Accordingly, $\mathbf{z} \in [\underline{\mathbf{z}}, \overline{\mathbf{z}}]$ denotes that $\underline{z}_j \leq z_j \leq \overline{z}_j$ for every j.

To preserve constraints from previously learned tasks without retaining past data, we maintain a cumulative hypercube $\mathcal{H}^{(\leq t)}$ that aggregates all historical intervals. After completing task t, the global boundary is updated as

$$\mathcal{H}^{(\leq t)} = \left[\min\left(\underline{\mathbf{z}}^{(\leq t-1)}, \underline{\mathbf{z}}_t\right), \max\left(\overline{\mathbf{z}}^{(\leq t-1)}, \overline{\mathbf{z}}_t\right)\right],$$

where $[\underline{\mathbf{z}}^{(\leq t-1)}, \overline{\mathbf{z}}^{(\leq t-1)}] = \mathcal{H}^{(\leq t-1)}$, and $\min(\cdot,\cdot)$ and $\max(\cdot,\cdot)$ denote element-wise minimum and maximum operators, respectively. This update computes the interval hull $\mathcal{H}^{(\leq t)} = \mathcal{H}^{(\leq t-1)} \sqcup \mathcal{H}_t$, i.e., the smallest hypercube enclosing the task-specific hypercubes of all previously observed tasks. At the same time, the resulting geometric representation remains memory-efficient, as storing the hypercube requires only lower and upper bounds for each feature dimension. Consequently, the memory footprint grows linearly with the feature dimensionality while remaining independent of the number of stored samples or tasks.

3.2 Functional Preservation via Regularization

We now describe how InTAct preserves classifier behavior within protected regions of the feature space. For clarity, we first consider a single linear layer. In deep architectures, the same mechanism is applied at multiple stages of the network by constructing protected hypercubes directly in the intermediate activation spaces. Consequently, we address stability in the full model by controlling the functional drift induced by each of these linear transformations separately. For notational simplicity, we omit explicit layer indices in the following derivation.

Consider a layer $f(\mathbf{z}) = \mathbf{W}\mathbf{z} + \mathbf{b}$, where $\mathbf{z}$ denotes an input representation, such as a `[CLS]` token or an intermediate hidden activation. During continual learning, the layer parameters evolve from $(\mathbf{W}, \mathbf{b})$ to $(\mathbf{W}', \mathbf{b}')$ as new tasks are acquired. The resulting functional change induced at representation $\mathbf{z}$ can therefore be written as

$$\Delta f(\mathbf{z}) = \Delta\mathbf{W}\mathbf{z} + \Delta\mathbf{b},$$

where $\Delta\mathbf{W} = \mathbf{W}' - \mathbf{W}$, $\Delta\mathbf{b} = \mathbf{b}' - \mathbf{b}$, and $\Delta f_i(\mathbf{z})$ denotes the i-th component of $\Delta f(\mathbf{z})$.

Our objective is to bound this drift within regions associated with previously learned representations, thereby preserving historical decision behavior while allowing adaptation outside the protected regions. Extensions to other layer types, along with a discussion of why we do not apply the regularizer to nonlinear functions, are provided in Appendix H.

Tractability via Interval Arithmetic. Directly enforcing stability for every point inside the protected region $\mathcal{H}^{(\leq t-1)}$ is computationally intractable. Instead, we employ *interval arithmetic (IA)* to derive rigorous bounds on the maximum functional drift over the entire region simultaneously.

Let $\mathcal{H}^{(\leq t-1)} = [\underline{\mathbf{z}}, \overline{\mathbf{z}}]$ denote the cumulative activation hypercube up to task $t-1$, where we omit the superscripts on its bounds for brevity. For each output unit i, we decompose the corresponding weight update $\Delta\mathbf{w}_i$ (the i-th row of $\Delta\mathbf{W}$) into its positive and negative components:

$$(\Delta\mathbf{w}_i)^+ = \max(\mathbf{0}, \Delta\mathbf{w}_i), \qquad (\Delta\mathbf{w}_i)^- = \max(\mathbf{0}, -\Delta\mathbf{w}_i),$$

where the maximum is taken element-wise. Using interval propagation (see Appendix G), the exact lower and upper bounds of the induced output drift over $\mathcal{H}^{(\leq t-1)}$ are given by

$$\underline{\delta}_i = \min_{\mathbf{z}\in\mathcal{H}^{(\leq t-1)}} \Delta f_i(\mathbf{z}) = (\Delta\mathbf{w}_i)^+\underline{\mathbf{z}} - (\Delta\mathbf{w}_i)^-\overline{\mathbf{z}} + \Delta b_i,$$
$$\overline{\delta}_i = \max_{\mathbf{z}\in\mathcal{H}^{(\leq t-1)}} \Delta f_i(\mathbf{z}) = (\Delta\mathbf{w}_i)^+\overline{\mathbf{z}} - (\Delta\mathbf{w}_i)^-\underline{\mathbf{z}} + \Delta b_i.$$

We define the internal representation drift loss ($\mathcal{L}_{\text{IntDrift}}$) as

$$\mathcal{L}_{\text{IntDrift}} = \frac{\lambda_{\text{IntDrift}}}{N} \sum_{i=1}^{N} \Big((\underline{\delta}_i)^2 + (\overline{\delta}_i)^2 \Big), \tag{1}$$

where N is the number of output units in the considered layer and $\lambda_{\text{IntDrift}}$ controls the regularization strength. Minimizing this loss penalizes deviations of the output bounds across all units, thereby limiting worst-case functional drift within the protected activation region. For brevity, we omit the explicit arguments of the loss functions throughout the paper.

Mathematical Stability Guarantee. Classical regularization-based continual learning methods constrain parameter updates in order to preserve previous knowledge. In contrast, InTAct directly constrains the functional behavior of the model inside regions occupied by historical representations. This is achieved by penalizing deviations in the model's output bounds through $\mathcal{L}_{\text{IntDrift}}$.

Theorem 1. *Let $\mathcal{H} = [\underline{\mathbf{z}}, \overline{\mathbf{z}}]$ be the protected hypercube with strictly positive dimensional diameters $\text{diam}_j(\mathcal{H}) = \overline{z}_j - \underline{z}_j$. If the internal drift loss satisfies $\mathcal{L}_{\text{IntDrift}} \leq \varepsilon$ for some $\varepsilon > 0$, then the functional drift for any representation $\mathbf{z}$ and output unit i is bounded by*

$$|\Delta f_i(\mathbf{z})| \leq \mathcal{O}(\sqrt{\varepsilon}) + \sum_{j=1}^{d} \left(\frac{d_j(\mathbf{z})}{\text{diam}_j(\mathcal{H})} \right) \mathcal{O}(\sqrt{\varepsilon}),$$

where

$$d_j(\mathbf{z}) = \min_{\mathbf{z}' \in \mathcal{H}} |z_j - z'_j|$$

denotes the distance from $\mathbf{z}$ *to the historical hypercube along the* j*-th dimension. The implicit multiplicative constants in* $\mathcal{O}(\cdot)$ *depend on* N *and* $\lambda_{\text{IntDrift}}$ *(see Appendix I).*

Theorem 1 establishes a localized, layer-wise upper bound on functional drift in the feature space. For representations inside the protected region ($d_j(\mathbf{z}) = 0$ for all j), the bound reduces to $\mathcal{O}(\sqrt{\varepsilon})$. Outside this region, the bound increases with the normalized distance from the historical hypercube, allowing greater functional flexibility farther from previously observed regions. Since the bound holds for any representation $\mathbf{z}$, it also covers historical activations excluded by the quantile-based construction. Empirically, most activation coordinates of previously learned tasks remain within the cumulative hypercube throughout training, with coverage ranging from 96.1% to 97.0% after the final task on ImageNet-R (Tab. 11 in Appendix L).

In our implementation, interval regularization is applied to the affine layers of the classifier. Extending interval propagation through the entire ViT backbone would suffer from the wrapping effect, resulting in excessively conservative bounds [24]. Consequently, the theorem guarantees localized functional stability in the classifier rather than providing an end-to-end guarantee for the entire model. Preserving the backbone representations is addressed through complementary regularization mechanisms introduced next. Together, these mechanisms promote stability while retaining the flexibility needed to learn new tasks. The full proof is provided in Appendix I.

Feature Space Regularization. For the guarantee in Theorem 1 to remain effective in prompt-based architectures, historical representations must remain within their protected regions throughout continual learning. However, continuously updating prompts can cause representations of previously learned samples to drift, potentially moving them outside the protected hypercube and weakening the resulting stability guarantee.

To mitigate this effect, we introduce a region-selective feature consistency objective that anchors historical representations within their protected regions. Specifically, we define the coordinate-wise binary gate $m_{i,j} = \mathbb{I}[\underline{z}_j \le z_{i,j}^{(t)} \le \overline{z}_j]$, where $\mathbb{I}[\cdot]$ denotes the indicator function and $[\underline{\mathbf{z}}, \overline{\mathbf{z}}] = \mathcal{H}^{(\le t-1)}$. This gate enforces consistency only on the coordinates of the current representation that lie within the protected hypercube (a per-sample gate would rarely be active; see Appendix L). Additional analysis of how the clipping threshold α affects the stability–plasticity trade-off is provided in Appendix L (Tab. 8). We then define the feature regularization term as

$$\mathcal{L}_{\text{Feat}} = \frac{\lambda_{\text{Feat}}}{\sum_{i,j} m_{i,j}} \sum_{i=1}^{N_B} \sum_{j=1}^{d} m_{i,j} \big(z_{i,j}^{(t)} - z_{i,j}^{(t-1)}\big)^2, \tag{2}$$

where N_B denotes the batch size, $\mathbf{z}_i^{(t-1)}$ and $\mathbf{z}_i^{(t)}$ are the representations of the i-th sample in the current training batch, produced by the frozen model after task $t-1$ and the current model, respectively. No data from previous tasks is replayed, and λ_{Feat} controls the regularization strength. Consequently, the objective preserves the coordinates within the protected region while leaving those outside it free to adapt to new tasks.

The feature consistency objective complements Theorem 1. For an input with representations $\mathbf{z}^{(t-1)}$ and $\mathbf{z}^{(t)}$, the output change of a protected layer is $f'(\mathbf{z}^{(t)}) - f(\mathbf{z}^{(t-1)}) = \Delta f(\mathbf{z}^{(t)}) + \mathbf{W}(\mathbf{z}^{(t)} - \mathbf{z}^{(t-1)})$. The theorem bounds the first term, and coordinates inside $\mathcal{H}^{(\le t-1)}$ do not contribute to its distance-dependent part, while $\mathcal{L}_{\text{Feat}}$ limits the contribution of exactly these coordinates to the second term. Thus, minimizing $\mathcal{L}_{\text{Feat}}$ encourages the current model to preserve the frozen model's representations inside the protected region, without explicitly storing or replaying historical samples.

The effectiveness of the proposed constraints additionally depends on maintaining compact activation regions. If task representations become excessively dispersed, the cumulative hypercube expands unnecessarily, reducing the free space available for future adaptation. To encourage compactness, we minimize the intra-task variance around the batch centroid $\boldsymbol{\mu}^{(t)} = \frac{1}{N_B} \sum_{i=1}^{N_B} \mathbf{z}_i$, using

$$\mathcal{L}_{\text{Var}} = \frac{\lambda_{\text{Var}}}{N_B\, d} \sum_{i=1}^{N_B} \left\| \mathbf{z}_i - \boldsymbol{\mu}^{(t)} \right\|_2^2, \tag{3}$$

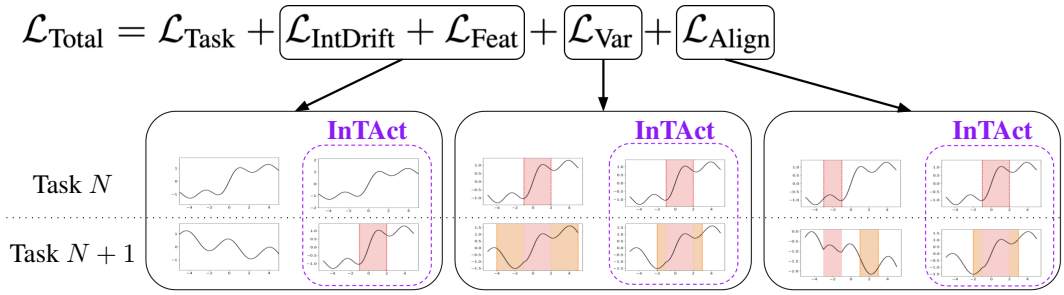


Figure 3: **Overview of the proposed regularization framework.** $\mathcal{L}_{\text{IntDrift}}$ and $\mathcal{L}_{\text{Feat}}$ preserve functional behavior within protected hypercubes corresponding to previously learned tasks (pink). $\mathcal{L}_{\text{Var}}$ encourages compact activation regions for newly acquired tasks (orange), while $\mathcal{L}_{\text{Align}}$ limits excessive expansion of the cumulative feature space by aligning task centers across tasks. Together, these constraints produce compact and stable feature regions, illustrated by the purple contours.

where λ_{Var} controls the strength of the compactness regularization. In Appendix L, we additionally analyze the resulting hypercube volumes and how much of the cumulative hypercube is occupied by the task-specific hypercubes.

Finally, even when individual task representations are compact, a single cumulative hypercube may grow excessively if consecutive tasks occupy distant regions of the feature space. To mitigate this expansion, we introduce an inter-task alignment objective that encourages the center $\mathbf{c}_t$ of the current batch to move toward the center $\mathbf{c}_{t-1}$ of the historical hypercube:

$$\mathcal{L}_{\text{Align}} = \lambda_{\text{Align}} \frac{\left\| \mathbf{m} \odot \left(\mathbf{c}_t - \mathbf{c}_{t-1} \right) \right\|_2^2}{r_{t-1} + \eta}, \tag{4}$$

where $\mathbf{c}_t = \frac{1}{2}(\min_i \mathbf{z}_i + \max_i \mathbf{z}_i)$ denotes the element-wise mid-range of the current batch, and $m_j = 1$ if the interval spanned by the current batch along dimension j does not overlap the corresponding historical interval $[\underline{z}_j, \overline{z}_j]$, and 0 otherwise. The symbol $\odot$ denotes element-wise multiplication. For the cumulative hypercube $\mathcal{H}^{(\leq t-1)} = [\underline{\mathbf{z}}, \overline{\mathbf{z}}]$, its center and average radius are defined as $\mathbf{c}_{t-1} = \frac{1}{2}(\underline{\mathbf{z}} + \overline{\mathbf{z}})$ and $r_{t-1} = \frac{1}{2d}\sum_{j=1}^{d}(\overline{z}_j - \underline{z}_j)$, respectively. The constant $\eta > 0$ ensures numerical stability, while λ_{Align} controls the strength of the regularization. Together, $\mathcal{L}_{\text{Var}}$ and $\mathcal{L}_{\text{Align}}$ encourage compact task-specific representations and alignment with the historical feature region, limiting unnecessary hypercube expansion while preserving the flexibility to adapt to new tasks.

Composite Training Objective. The final optimization objective combines the task-specific cross-entropy (CE) loss with the proposed stability and geometric regularization terms:

$$\mathcal{L}_{\text{Total}} = \mathcal{L}_{\text{CE}} + \underbrace{\mathcal{L}_{\text{IntDrift}} + \mathcal{L}_{\text{Feat}}}_{\text{Functional Stability}} + \underbrace{\mathcal{L}_{\text{Var}} + \mathcal{L}_{\text{Align}}}_{\text{Geometric Structuring}}.$$

The interaction between these components is illustrated in Fig. 3. The complete training procedure is provided in Appendix J.

4 Experiments

We evaluate InTAct in the DIL setting, focusing on three key areas: comparative performance against SOTA methods, plasticity under distribution shifts, and the individual contribution of each loss component to the stability-plasticity trade-off.

4.1 Experimental Setup

Datasets and Baselines. We evaluate InTAct on four standard continual learning benchmarks: DomainNet, ImageNet-R, ImageNet-C, and ImageNet-Mix. The proposed method is compared against classical regularization-based baselines (EWC, LwF) as well as recent SOTA approaches,

Table 1: **Benchmarking against the SOTA.** We report AA across four diverse DIL benchmarks as the mean over ten runs for InTAct and over three runs for the baselines. InTAct achieves the highest average performance, with the largest gains on ImageNet-R and ImageNet-Mix. On DomainNet and ImageNet-C, InTAct remains highly competitive, matching or slightly exceeding the best baseline results. Presented results are achieved by integrating InTAct into KA-Prompt. Baseline results are sourced from [21].

Method	Ref.	DomainNet	ImageNet-R	ImageNet-C	ImageNet-Mix	**Average**
EWC	*PNAS 2017*	47.10±0.58	51.83±0.33	65.20±1.23	57.82±0.90	55.49±0.42
LwF	*T-PAMI 2017*	54.85±0.06	57.11±0.95	69.01±1.21	65.27±0.93	61.56±0.45
L2P	*CVPR 2022*	53.74±0.04	56.55±0.33	77.86±0.44	64.20±0.34	63.09±0.16
S-Prompts	*NeurIPS 2022*	44.08±0.05	27.23±0.16	60.25±0.28	24.25±0.19	38.95±0.09
DualPrompt	*ECCV 2022*	55.18±0.02	59.47±1.00	78.52±0.30	63.57±0.17	64.19±0.26
ESN	*AAAI 2023*	45.74±0.25	16.39±0.23	68.34±0.39	14.99±0.53	36.37±0.19
CODA-P	*CVPR 2023*	55.81±0.03	55.21±0.33	78.25±0.16	64.92±0.04	63.55±0.09
InfLoRA	*CVPR 2024*	48.76±0.30	41.20±1.65	53.12±0.16	35.38±0.27	44.62±0.43
Cprompt	*CVPR 2024*	52.88±0.12	59.48±1.09	70.08±0.07	63.64±0.74	61.52±0.33
C-Prompt	*IJCV 2024*	58.66±0.05	62.43±0.49	79.84±0.38	65.35±0.52	66.57±0.20
KA-Prompt	*ICML 2025*	62.91±0.14	66.51±0.36	85.43±0.56	70.35±0.32	71.30±0.19
InTAct	*This Paper*	**62.96**±0.09	**67.90**±0.44	**85.75**±0.31	**70.94**±0.41	**71.89**±0.17

including L2P, DualPrompt, CODA-Prompt, KA-Prompt, and InfLoRA, with particular emphasis on prompt-based continual learning methods. We report the Average Accuracy (AA) metric; its formal definition is provided in Appendix F.

Implementation Details. For fair comparison, we follow KA-Prompt in using the same task splits, continual learning protocols, and ViT-B/16 backbone pre-trained on ImageNet-21k without ImageNet-1k fine-tuning [25]. Results for InTAct are averaged over ten runs with different random seeds. For experiments combining CODA-Prompt with InTAct, including ablations and sensitivity analyses, we use a ViT-B/16 backbone additionally fine-tuned on ImageNet-1k [26]. Further implementation details and hyperparameter configurations are provided in Appendix C.

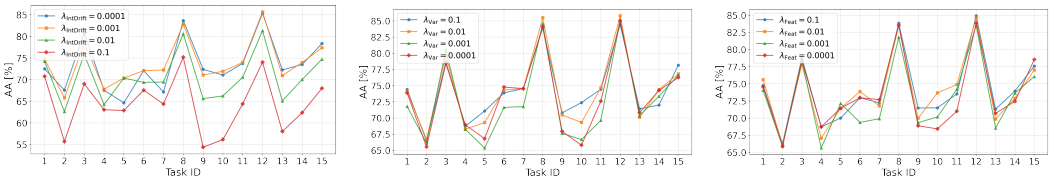
4.2 Results

Main Results. Table 1 reports the AA across four DIL benchmarks. Overall, InTAct achieves the highest average performance among the compared methods. In particular, it establishes new state-of-the-art results on ImageNet-R and ImageNet-Mix, outperforming the strongest baseline, KA-Prompt, by **1.39 p.p.** and **0.59 p.p.**, respectively. InTAct also achieves the best result on ImageNet-C.

On DomainNet, InTAct performs on par with KA-Prompt. Importantly, InTAct is integrated into KA-Prompt without modifying its prompt selection or training procedures, demonstrating that the proposed regularization can improve or match the original architecture across all evaluated benchmarks. This indicates that the observed gains arise from the proposed feature-space regularization rather than architecture-specific modifications.

These results suggest that parameter-space regularization may be insufficient to fully preserve feature representations under substantial domain shifts, as constraining weight updates does not necessarily prevent changes in the feature space. By directly regularizing functional changes within protected regions, InTAct offers an alternative way to promote stability while retaining flexibility for future adaptation. Although the gains on corruption-heavy benchmarks, particularly ImageNet-C and ImageNet-Mix, are more modest, they are consistent with the potential benefits of stabilizing feature representations under distribution shifts.

Integration with Other Prompt-Based Methods. The gains over KA-Prompt are relatively modest, which may be partly explained by its exponential moving average of classifier weights, which already helps mitigate classifier drift. In contrast, methods without this mechanism benefit more substantially: integrating InTAct into L2P, DualPrompt, and CODA-Prompt yields improvements of 4.1–13.0 p.p. (see Appendix K).



(a) Sensitivity to $\lambda_{\text{IntDrift}}$ changes (b) Sensitivity to λ_{Var} changes (c) Sensitivity to λ_{Feat} changes

Figure 5: Hyperparameter sensitivity of CODA-Prompt + InTAct on ImageNet-R: final accuracy on each task, averaged over three independent runs. (a) Lower drift penalties preserve classifier plasticity, allowing adaptation to new tasks. (b) Stronger variance regularization promotes more compact representations and improves stability. (c) A sufficiently strong feature consistency penalty is crucial for preventing drift in the backbone's representations.

AA Evolution Analysis on ImageNet-R. Fig. 4 shows the evolution of AA on ImageNet-R over 15 sequential tasks. Since InTAct is built on KA-Prompt, it starts at a comparable performance level and surpasses all competing methods from $t = 7$ onward. While classical baselines such as EWC and LwF exhibit a sharp performance drop around the 9th task, InTAct maintains a substantially more stable trajectory, with only minor degradation as new tasks are introduced. Furthermore, compared with recent prompt-based approaches, including KA-Prompt and C-Prompt, InTAct exhibits a more favorable balance between stability and plasticity. By the final task, the strongest baselines remain below 67% AA.

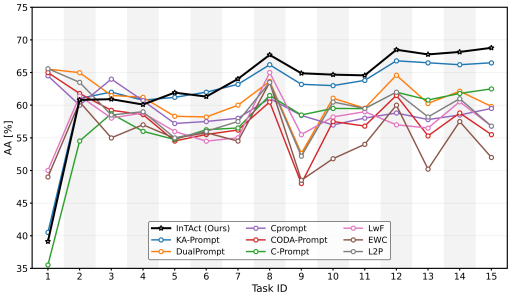


Figure 4: **AA Evolution on ImageNet-R.** Above plot reports the AA across 15 incremental tasks. InTAct (black line with stars) outperforms SOTA prompt-based baselines (e.g., KA-Prompt, DualPrompt) and regularization methods (e.g., EWC, LwF) from $t = 7$ onward. Notably, our method maintains high accuracy even during challenging task transitions (e.g., $t = 9$), where most other methods exhibit significant forgetting. Baseline results are sourced from [21].

4.3 Ablation Study

To assess the contribution of each component in InTAct, we conduct a systematic ablation study on the ImageNet-R benchmark (Table 2) using CODA-Prompt + InTAct; the absolute AA therefore differs from Tab. 1, which uses KA-Prompt and a different backbone. Specifically, we remove one regularization term at a time to isolate its role in the stability–plasticity trade-off.

The results demonstrate that all components contribute to the final performance, with the interval drift constraint playing the most critical role. Removing the proposed interval-based regularization term ($\mathcal{L}_{\text{IntDrift}}$) causes the largest degradation, reducing AA by **3.48** p.p. This confirms that directly constraining functional drift within protected feature regions is the primary mechanism responsible for preserving previously learned knowledge across domain shifts.

The feature consistency term ($\mathcal{L}_{\text{Feat}}$) is also important, with its removal decreasing performance by **1.79** p.p. This indicates that stabilizing intermediate representations remains necessary even when the classifier behavior is explicitly constrained.

The remaining geometric regularization terms further improve the structure of the feature space. Removing the compactness objective ($\mathcal{L}_{\text{Var}}$) lowers AA by **0.95** p.p., suggesting that unconstrained feature dispersion leads to larger protected regions and increased inter-task interference. Similarly, excluding the alignment term ($\mathcal{L}_{\text{Align}}$) results in a **0.47** p.p. drop, indicating that aligning task centers helps limit unnecessary expansion of the cumulative hypercube and preserves future plasticity. An extension of Tab. 2 to all pairwise removals of the four terms is provided in Tab. 5 in Appendix L; the strongest interaction occurs between $\mathcal{L}_{\text{IntDrift}}$ and $\mathcal{L}_{\text{Var}}$, whose joint removal causes a drop nearly twice the sum of their individual drops. In contrast, the gradients of $\mathcal{L}_{\text{IntDrift}}$ and $\mathcal{L}_{\text{Align}}$ remain nearly orthogonal during training (average cosine similarity of at most 0.012; Appendix L), indicating that the alignment term does not work against the interval drift constraint.

4.4 Sensitivity Analysis on ImageNet-R

We analyze the sensitivity of InTAct to its main regularization hyperparameters on the ImageNet-R benchmark, using CODA-Prompt + InTAct.

Internal Drift Regularization ($\lambda_{\text{IntDrift}}$). Fig. 5a shows that smaller values of $\lambda_{\text{IntDrift}}$ yield higher accuracy, while stronger regularization degrades performance. This highlights the need to balance stability and plasticity: although functional drift must be controlled, the classifier needs sufficient flexibility to adapt its decision boundaries to the substantial domain shifts in ImageNet-R.

Table 2: **Ablation Study.** Impact of removing individual loss components of CODA-Prompt + InTAct on ImageNet-R. We report the performance drop (Δ) relative to the baseline InTAct. The results are averaged over 3 random seeds.

$\mathcal{L}_{\text{IntDrift}}$	$\mathcal{L}_{\text{Var}}$	$\mathcal{L}_{\text{Align}}$	$\mathcal{L}_{\text{Feat}}$	**AA (%)**	Δ
✓	✓	✓	✓	**74.23**	—
×	✓	✓	✓	70.75	-3.48
✓	×	✓	✓	73.28	-0.95
✓	✓	×	✓	73.76	-0.47
✓	✓	✓	×	72.44	-1.79

Variance Regularization (λ_{Var}). As illustrated in Fig. 5b, increasing λ_{Var} generally improves performance by encouraging more compact feature representations. By reducing the overall volume of the cumulative hypercube, this regularization limits unnecessary expansion of the protected region, thereby reducing inter-task interference and preserving feature-space capacity for future adaptation.

Feature Stability (λ_{Feat}). Fig. 5c shows that stronger feature consistency regularization generally improves accuracy, although not monotonically. By penalizing deviations from the frozen model's representations, $\mathcal{L}_{\text{Feat}}$ mitigates representational drift caused by prompt updates, helping preserve the mapping of historical inputs to their protected feature regions.

4.5 Extended Analysis

Appendix L extends the evaluation of InTAct with CIL experiments, sensitivity to the hypercube coverage parameter (α), and analyses of historical activation coverage, hypercube compactness, and occupancy over long task sequences.

5 Conclusion

We introduced InTAct, a continual learning framework that mitigates catastrophic forgetting by directly constraining functional behavior in feature space rather than regularizing individual model parameters. Leveraging IA, the proposed method preserves stability within regions associated with previously learned tasks while maintaining flexibility in unconstrained regions of the representation space. This explicit separation between stability and plasticity enables effective continual adaptation without relying on replay buffers or architectural expansion. Extensive experiments demonstrate that InTAct consistently improves existing prompt-based continual learning methods and achieves SOTA performance in the DIL setting.

Limitations. Our theoretical guarantee provides a local, layer-wise upper bound on functional drift, conditional on the internal drift loss being sufficiently small and historical representations remaining within the protected hypercube. Consequently, the strength of the guarantee depends on the effectiveness of the optimization and the geometry of the protected region. Larger hypercubes cover a broader portion of the feature space but may yield looser bounds, whereas smaller hypercubes provide tighter guarantees over more limited regions. Furthermore, our approach relies on hypercubes to capture regions occupied by historical representations, which may not fully reflect the geometry of complex feature distributions. Exploring alternative representations of historical regions that provide a more flexible characterization of the feature space is an important direction for future work.

Acknowledgments. The work of P. Spurek and P. Krukowski was supported by the National Centre of Science (Poland) Grant No. 2023/50/E/ST6/00068. We gratefully acknowledge Polish high-performance computing infrastructure PLGrid (HPC Centers: ACK Cyfronet AGH, WCSS, CI TASK) for providing computer facilities and support within computational grant no. PLG/2026/019272.

References

[1] 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, March 2017.

[2] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press, 1989.

[3] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:475–480, 2019.

[4] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

[5] Marc Masana, Xialei Liu, Bartlomiej Twardowski, Mikel Menta, Andrew D. Bagdanov, and Joost van de Weijer. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5513–5533, 2022.

[6] Maciej Wołczyk, Karol Piczak, Bartosz Wójcik, Lukasz Pustelnik, Paweł Morawiecki, Jacek Tabor, Tomasz Trzcinski, and Przemysław Spurek. Continual learning with guarantees via weight interval constraints. In *International Conference on Machine Learning*, pages 23897–23911. PMLR, 2022.

[7] Patryk Krukowski, Anna Bielawska, Kamil Ksiazek, Paweł Wawrzyński, Paweł Batorski, and Przemysław Spurek. Hint: Hypernetwork approach to training weight interval regions in continual learning. *Information Sciences*, 717:122261, 2025.

[8] 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)*, pages 139–149, 2022.

[9] 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 *European Conference on Computer Vision (ECCV)*, pages 631–648, 2022.

[10] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *International Conference on Learning Representations*, 2018.

[11] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *Advances in neural information processing systems*, volume 33, pages 15920–15930, 2020.

[12] Edoardo Urettini and Antonio Carta. Online curvature-aware replay: Leveraging 2nd order information for online continual learning. In *International Conference on Machine Learning*, 2025.

[13] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in neural information processing systems*, volume 30, 2017.

[14] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[15] Zhen Wang, Liu Liu, Yiqun Duan, Yajing Kong, and Dacheng Tao. Continual learning with lifelong vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 171–181, 2022.

[16] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[17] Joan Serrà, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pages 4548–4557, 2018.

[18] Yan-Shuo Liang and Wu-Jun Li. InfLoRA: Interference-free low-rank adaptation for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 23638–23647, 2024.

[19] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11909–11919, 2023.

[20] Zichen Liu, Yuxin Peng, and Jiahuan Zhou. Compositional prompting for anti-forgetting in domain incremental learning. *International Journal of Computer Vision*, 132:5783 – 5800, 2024.

[21] Kunlun Xu, Xu Zou, Gang Hua, and Jiahuan Zhou. Componential prompt-knowledge alignment for domain incremental learning. *ICML*, 2025.

[22] Germund Dahlquist and Åke Björck. *Numerical methods in scientific computing, volume I*. SIAM, 2008.

[23] Ramon E. Moore, R. Baker Kearfott, and Michael J. Cloud. *Introduction to interval analysis*. SIAM, 2009.

[24] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models, 2019.

[25] 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, 2021.

[26] Andreas Peter Steiner, Alexander Kolesnikov, Xiao-Hua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *Trans. Mach. Learn. Res.*, 2022, 2021.

[27] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations*, 2018.

[28] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning, 2019. arXiv preprint arXiv:1902.10486.

[29] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning, 2022.

[30] Yan-Shuo Liang and Wu-Jun Li. Loss decoupling for task-agnostic continual learning. In *Advances in Neural Information Processing Systems*, volume 36, 2024.

[31] Chenshen Wu, Luis Herranz, Xialei Liu, Yaxing Wang, Joost van de Weijer, and Bogdan Raducanu. Memory replay gans: Learning to generate new categories without forgetting. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[32] Sanghwan Kim, Lorenzo Noci, Antonio Orvieto, and Thomas Hofmann. Achieving a better stability-plasticity trade-off via auxiliary networks in continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11930–11939, 2023.

[33] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence, 2017.

[34] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget, 2018.

[35] Zhikang Chen, Abudukelimu Wuerkaixi, Sen Cui, Haoxuan Li, Ding Li, Jingfeng Zhang, Bo Han, Gang Niu, Houfang Liu, Yi Yang, Sifan Yang, Changshui Zhang, and Tianling Ren. Learning without isolation: Pathway protection for continual learning. In *International Conference on Machine Learning*, 2025.

[36] Yajing Kong, Liu Liu, Zhen Wang, and Dacheng Tao. Balancing stability and plasticity through advanced null space in continual learning. In *European Conference on Computer Vision*, pages 219–236, 2022.

[37] Shixiang Tang, Dapeng Chen, Jinguo Zhu, Shijie Yu, and Wanli Ouyang. In *Layerwise optimization by gradient decomposition for continual learning*, pages 9634–9643, 2021.

[38] Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 184–193, 2021.

[39] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks, 2016. arXiv preprint arXiv:1606.04671.

[40] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks, 2022.

[41] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82, 2018.

[42] Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in superposition. In *Advances in Neural Information Processing Systems*, volume 33, pages 15173–15184, 2020.

[43] Yabin Wang, Zhiheng Ma, Zhiwu Huang, Yaowei Wang, Zhou Su, and Xiaopeng Hong. Isolation and impartial aggregation: A paradigm of incremental learning without interference. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 10209–10217, 2023.

[44] Zhanxin Gao, Jun Cen, and Xiaobin Chang. Consistent prompting for rehearsal-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28463–28473, 2024.

[45] Paul Janson, Wenxuan Zhang, Rahaf Aljundi, and Mohamed Elhoseiny. A simple baseline that questions the use of pretrained-models in continual learning. In *NeurIPS 2022 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2022.

[46] Dipam Goswami, Yuyang Liu, Bartłomiej Twardowski, and Joost van de Weijer. Fecam: Exploiting the heterogeneity of class distributions in exemplar-free continual learning, 2024.

[47] Lu Yu, Bartlomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6982–6991, 2020.

[48] Marco Toldo and Mete Ozay. Bring evanescent representations to life in lifelong class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16732–16741, 2022.

[49] Tamasha Malepathirana, Damith Senanayake, and Saman Halgamuge. Napa-vq: Neighborhood-aware prototype augmentation with vector quantization for continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11674–11684, 2023.

[50] Wuxuan Shi and Mang Ye. Prototype reminiscence and augmented asymmetric knowledge aggregation for non-exemplar class-incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1772–1781, 2023.

[51] Grégoire Petit, Adrian Popescu, Hugo Schindler, David Picard, and Bertrand Delezoide. Fetril: Feature translation for exemplar-free class-incremental learning. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 3911–3920, 2023.

[52] Dipam Goswami, Albin Soutif-Cormerais, Yuyang Liu, Sandesh Kamath, Bart Twardowski, and Joost van de Weijer. Resurrecting old classes with new data for exemplar-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28525–28534, 2024.

[53] Da-Wei Zhou, Hai-Long Sun, Han-Jia Ye, and De-Chuan Zhan. Expandable subspace ensemble for pre-trained model-based class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23554–23564, 2024.

[54] Xuan Liu and Xiaobin Chang. Lora subtraction for drift-resistant space in exemplar-free continual learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 15308–15318, 2025.

[55] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation, 2021.

[56] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1406–1415, 2019.

[57] 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. *ICCV*, 2021.

[58] Dan Hendrycks and Thomas G. Dietterich. Benchmarking neural network robustness to common corruptions and surface variations, 2019.

[59] Luc Jaulin, Michel Kieffer, Olivier Didrit, and Eric Walter. *Applied Interval Analysis: With Examples in Parameter and State Estimation, Robust and Control*. Springer, London, 2001.

A Extended Overview of Related Works

This section reviews classical and recent approaches to continual learning. We first introduce the three main paradigms: replay-based, regularization-based, and parameter-isolation methods. We then discuss parameter-efficient approaches, focusing on prompt-based continual learning, followed by methods addressing representational drift in feature space. Finally, we review applications of interval arithmetic (IA) to continual learning.

Continual Learning. Continual learning methods [3, 4, 5] can generally be divided into three categories. *Replay-based* methods use memory or rehearsal mechanisms to retain information from previous tasks. These include exemplar replay, which stores selected training samples [10, 27, 28, 11, 29, 30, 12], and generative replay, which synthesizes previously observed data using generative models [13, 31]. *Regularization-based* methods constrain parameter updates to preserve previously acquired knowledge. Such constraints can be derived from current-task data [14, 32] or information

accumulated from previous tasks [1, 33, 34, 35]. Other approaches restrict weight updates to the null space of previous-task feature covariance [36, 37, 38, 15], while [30] uses gradient information from previous tasks to construct a subspace for the LoRA dimensionality-reduction matrix. Finally, *parameter-isolation* methods allocate task-specific parameters or subnetworks to prevent interference. Examples include Progressive Neural Networks [39, 40], Piggyback [41], PackNet [16], HAT [17], SupSup [42], ESN [43], and InfLoRA [18]. Although effective in task-aware settings, many such methods require task identification at inference time.

Prompt-Based Continual Learning. Prompt-based continual learning adapts a frozen pre-trained backbone using a small set of learnable parameters called *prompts*. Typically, these are vectors prepended to input embeddings or inserted into intermediate layers, steering the model toward new tasks without modifying the backbone weights. A key development is the use of a *prompt pool*. L2P [8] maintains a shared set of prompt vectors and selects a sparse subset by matching input features to learned prompt keys. This enables instance-wise prompt selection at inference time without requiring task identity. Subsequent methods introduce additional structure and constraints on prompt learning. DualPrompt [9] separates prompts into general components shared across tasks and task-specific components that capture task-dependent knowledge. CODA-Prompt [19] employs decomposed attention to dynamically combine prompt components, aiming to reduce interference between tasks. Compositional Prompting (C-Prompt) [20] promotes the sharing of compositional prompt components across domains, while Consistent Prompting (CPrompt) [44] encourages consistency between task-specific prompts to mitigate representational drift. More recently, KA-Prompt [21] addresses component-wise misalignment between prompts learned for different domains by initializing new prompts from relevant historical ones and encouraging consistency with them during training. Despite their effectiveness in parameter-efficient, rehearsal-free adaptation, prompt-based methods can still suffer from representational drift and classifier instability, particularly in domain-incremental learning (DIL). Although the backbone remains frozen, prompt updates alter the representations passed to the classifier, while substantial distribution shifts may require adapting its decision boundaries. Thus, preserving pre-trained weights alone does not ensure stable predictions across domains.

Feature Stability in Continual Learning. A related line of research addresses the drift of previously learned representations as new tasks are introduced. Many approaches rely on class prototypes [45, 46] and aim to preserve their positions in feature space. SDC [47] and [48] estimate and compensate for feature drift using current-task data after each training phase. NAPA-VQ [49] and Prototype Reminiscence [50] leverage topological information to reshape historical prototypes, while FeTrIL [51] translates features to align old and new class distributions. ADC [52] generates adversarial pseudo-exemplars of new classes to adjust previous prototypes, and EASE [53] uses semantic-guided prototype complementation to update old-class representations. Another approach [54] constructs a drift-resistant feature space in which model weights can adapt to new tasks while limiting interference with previously learned representations. More broadly, representation-level regularization seeks to constrain changes in intermediate activations rather than directly restricting parameter updates. Such approaches are closely related to activation-based methods that preserve task-specific regions of feature space, although efficiently characterizing and maintaining these regions over long task sequences remains challenging.

Interval-Based Methods in Continual Learning. IA [22, 23] has also been explored in continual learning. InterContiNet [6] constrains network weights to task-specific intervals, whose intersection defines configurations that maintain performance across tasks. HINT [7] instead constructs intervals in the embedding space and uses a hypernetwork to map them to network weights. Together, these methods explore how interval representations can preserve regions of parameter or feature space associated with previously learned tasks.

B Preliminaries

Prompt-Based Continual Learning. Prompt-based continual learning exploits the representational capacity of pretrained ViT [25] by introducing a small set of learnable parameters (prompts) to adapt the model to new tasks while keeping the backbone frozen. This strategy avoids the computational cost and catastrophic forgetting associated with fine-tuning the entire network.

Formally, consider a ViT encoder $f_{\boldsymbol{\theta}}$ with fixed parameters $\boldsymbol{\theta}$. The input to the l-th layer is denoted as $\mathbf{z}^{(l)} \in \mathbb{R}^{L \times D}$, where L is the sequence length and D is the embedding dimension. To adapt the representation, standard approaches employ Prefix-Tuning [55]. This mechanism prepends learnable vectors to the internal attention layers. Specifically, learnable prompt matrices $\mathbf{P}_K, \mathbf{P}_V \in \mathbb{R}^{L_p \times D}$ (where L_p is the prompt length) are defined and concatenated with the standard Key and Value projections:

$$\begin{aligned} \mathbf{Q} &= \mathbf{z}^{(l)}\mathbf{W}_Q, \\ \mathbf{K}' &= \left[\mathbf{P}_K; \mathbf{z}^{(l)}\mathbf{W}_K\right], \\ \mathbf{V}' &= \left[\mathbf{P}_V; \mathbf{z}^{(l)}\mathbf{W}_V\right], \\ \mathbf{z}^{(l+1)} &= \mathrm{MSA}(\mathbf{Q}, \mathbf{K}', \mathbf{V}'), \end{aligned} \tag{5}$$

where MSA denotes Multi-Head Self-Attention, $[;]$ represents concatenation along the sequence dimension, and $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$ are the frozen projection matrices. In this framework, only the prompts $\mathbf{P}_K, \mathbf{P}_V$ and the final classification head are optimized, allowing the system to adapt to non-stationary data streams efficiently.

Prompt Selection and Memory Structure. A central challenge in task-agnostic continual learning is selecting appropriate prompts without access to task identifiers at test time. To address this, methods like Learning to Prompt (L2P) [8] maintain a shared prompt pool $\mathcal{P} = \{(\mathbf{k}_i, \mathbf{p}_i)\}_{i=1}^{M}$, where each learnable prompt $\mathbf{p}_i$ is paired with a key $\mathbf{k}_i \in \mathbb{R}^D$.

The inference process operates via a content-based retrieval mechanism. First, the input $\mathbf{x}$ is passed through the frozen encoder to generate a query vector $\mathbf{q}(\mathbf{x})$ (typically the `[CLS]` embedding). This query is matched against the stored keys $\{\mathbf{k}_i\}$ using cosine similarity to identify the subset of the K most relevant prompts. These selected prompts are then explicitly concatenated with the input embeddings or attention prefixes, effectively conditioning the frozen backbone on the retrieved context. During training, optimization is sparse; gradients are applied only to the selected prompt-key pairs, allowing the remaining pool contents to preserve knowledge from disparate tasks.

Optimization Objective. Since the discrete retrieval of keys (Top-K) is non-differentiable, gradients cannot backpropagate to the keys directly. Consequently, prompt-based methods typically minimize a composite objective $\mathcal{L}_{\text{Total}} = \mathcal{L}_{\text{task}} + \lambda \mathcal{L}_{\text{match}}$. Here, $\mathcal{L}_{\text{task}}$ (e.g., Cross-Entropy) updates the prompts and classifier head, while the surrogate loss $\mathcal{L}_{\text{match}}$ pulls the keys $\{\mathbf{k}_i\}$ closer to the queries of their assigned inputs, ensuring that the retrieval mechanism aligns with the data distribution.

Hierarchical and Decomposed Prompting. DualPrompt [9] decouples plasticity and stability by partitioning the prompts into distinct functional groups: G-Prompts ($\mathbf{g}$) for task-invariant features and E-Prompts ($\mathbf{e}_t$) for task-specific experts. These are inserted into complementary depths of the backbone:

$$\mathbf{z}^{(l+1)} = \begin{cases} \mathrm{MSA}(\mathbf{z}^{(l)}, \mathbf{g}) & \text{if } l \leq L_g \quad \text{(General Layers)} \\ \mathrm{MSA}(\mathbf{z}^{(l)}, \mathbf{e}_t) & \text{if } L_g < l \leq L_e \quad \text{(Expert Layers)} \end{cases} \tag{6}$$

where $\mathbf{e}_t$ is retrieved via a task-specific key $\mathbf{k}_t$, and the remaining layers are not prompted.

CODA-Prompt [19] generalizes this by replacing discrete selection with a soft, weighted summation of components. The input query $\mathbf{q}(\mathbf{x})$ attends to the entire pool of prompt components $\{\mathbf{P}_m\}_{m=1}^{M}$, each paired with a key $\mathbf{k}_m$ and an attention vector $\mathbf{A}_m$, via a learnable attention mechanism:

$$\alpha_m(\mathbf{x}) = \cos\left(\mathbf{q}(\mathbf{x}) \odot \mathbf{A}_m, \mathbf{k}_m\right), \quad \mathbf{p}(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m(\mathbf{x}) \mathbf{P}_m. \tag{7}$$

Here, $\cos(\cdot,\cdot)$ denotes the cosine similarity, $\mathbf{A}_m$ acts as a query-modulation vector, and $\odot$ denotes the Hadamard product. This differentiable formulation allows the prompt $\mathbf{p}(\mathbf{x})$ to form complex linear combinations of existing knowledge, stabilized by an orthogonality penalty on the components, keys, and attention vectors to minimize inter-component redundancy.

C Implementation Details

Architecture and Training. All prompt-based experiments use a ViT-B/16 backbone [25] pre-trained on ImageNet-21k, with the exact weights depending on the underlying codebase. Experiments based on KA-Prompt, including the main results and occupancy analysis, use the original ImageNet-21k weights without ImageNet-1k fine-tuning. Experiments based on the CODA-Prompt codebase, including ablations, sensitivity analyses, and comparisons with other prompt methods, use weights additionally fine-tuned on ImageNet-1k [26]. L2P + InTAct and DualPrompt + InTAct follow the latter configuration. In all experiments, the backbone remains frozen, and adaptation is performed by optimizing the prompts and shared classification head. Since we consider DIL, all tasks share the same label space and classifier output dimension. The model must therefore adapt to domain shifts while preserving consistent decision boundaries across tasks.

Datasets and Protocols. We evaluate InTAct on four benchmarks designed to assess robustness against varying degrees of domain shift: **DomainNet** [56] offers the most severe visual shifts, consisting of six distinct domains (Clipart, Infograph, Painting, Quickdraw, Real, Sketch). In our DIL setting, each domain constitutes a separate task, processed sequentially. **ImageNet-R** [57] contains 200 classes rendered in diverse artistic styles (e.g., cartoons, embroidery). To construct a DIL sequence, we partition the dataset into 15 sequential tasks, ensuring all classes are present in every task but represented by different samples/styles. **ImageNet-C** [58] evaluates robustness against corruption. Following the protocol of [20], we utilize the 200 classes overlapping with ImageNet-R, subjecting them to 15 types of algorithmic perturbations (e.g., blur, noise) to form the task sequence. Finally, **ImageNet-Mix** [20] combines ImageNet-R styles with ImageNet-C corruptions, resulting in a 30-task benchmark that rigorously tests stability under compound distributional shifts. All datasets are normalized and transformed during both training and evaluation, following the KA-Prompt method [21].

Code and Reproducibility. For the main experiments (Tab. 1), we integrate InTAct into the official KA-Prompt implementation [21], adopting its task splits and evaluation protocol. For experiments with L2P, DualPrompt, and CODA-Prompt, we use the official CODA-Prompt implementation and its corresponding task splits and evaluation protocol. In both cases, InTAct is integrated without modifying the original prompt selection or training procedures.

Experiments were conducted on NVIDIA H100 (80 GB), A100 (40 GB), and V100 (32 GB) GPUs within a DGX cluster, as well as standalone RTX 4090 (24 GB) machines. Training pipelines, configuration files, and reproduction scripts are available in our GitHub repository.

D Hyperparameter Configuration

For KA-Prompt + InTAct, we keep all hyperparameters of KA-Prompt, including its EMA classifier, fix $\lambda_{\text{Align}} = 1$, and select the remaining coefficients via grid search over $\lambda_{\text{Var}} \in \{10^{-2}, 10^{-1}, 1\}$, $\lambda_{\text{IntDrift}} \in \{10^{-4}, 10^{-3}, 10^{-2}\}$, and $\lambda_{\text{Feat}} \in \{10^{-2}, 10^{-1}\}$. For CODA-Prompt + InTAct, we use identical backbone and prompt configurations as the baseline, fix $\lambda_{\text{Align}} = 1$, and search over $\lambda_{\text{Var}} \in \{10^{-3}, \ldots, 1.0\}$ and $\lambda_{\text{IntDrift}}, \lambda_{\text{Feat}} \in \{10^{-6}, \ldots, 10^{-1}\}$. The specific values selected for the final evaluation are detailed in Table 3.

Table 3: **Optimal Hyperparameters.** Selected hyperparameters for InTAct integrated into KA-Prompt and into CODA-Prompt across different benchmarks.

	KA-Prompt + InTAct			CODA-Prompt + InTAct		
Dataset	λ_{Feat}	$\lambda_{\text{IntDrift}}$	λ_{Var}	λ_{Feat}	$\lambda_{\text{IntDrift}}$	λ_{Var}
DomainNet	10^{-2}	10^{-4}	10^{-1}	10^{-4}	10^{-3}	1.0
ImageNet-R	10^{-1}	10^{-3}	1.0	10^{-1}	10^{-3}	10^{-1}
ImageNet-C	10^{-2}	10^{-4}	1.0	10^{-1}	10^{-4}	10^{-1}
ImageNet-Mix	10^{-2}	10^{-4}	1.0	10^{-1}	10^{-4}	10^{-1}

E Interval Arithmetic

IA, introduced in its modern form by Ramon E. Moore [23], provides a method for performing computations on ranges of real numbers rather than single point values. An interval x is a closed, bounded set of real numbers defined by its endpoints, $x = [\underline{x}, \overline{x}]$, such that $\underline{x} \leq \overline{x}$. This framework is essential for validated numerics, as it allows for the rigorous containment of numerical errors, including rounding errors and uncertainties in initial data.

E.1 Fundamental Operations

The fundamental principle of IA is the *inclusion property*: the resulting interval of an operation must contain all possible results from applying the same operation to any real numbers within the operand intervals. Let $x = [\underline{x}, \overline{x}]$ and $y = [\underline{y}, \overline{y}]$ be two intervals. The elementary arithmetic operations are defined as follows:

Addition. The sum is obtained by adding the respective endpoints:

$$x + y = [\underline{x} + \underline{y}, \overline{x} + \overline{y}]$$

Subtraction. The difference is computed by cross-adding the endpoints:

$$x - y = [\underline{x} - \overline{y}, \overline{x} - \underline{y}]$$

Multiplication. The product is the interval spanning the minimum and maximum of the four products of the endpoints.

$$x \cdot y = [\underline{p}, \overline{p}],$$

where

$$\underline{p} = \min(\underline{x}\,\underline{y}, \underline{x}\,\overline{y}, \overline{x}\,\underline{y}, \overline{x}\,\overline{y}),$$
$$\overline{p} = \max(\underline{x}\,\underline{y}, \underline{x}\,\overline{y}, \overline{x}\,\underline{y}, \overline{x}\,\overline{y}).$$

Division. Division is defined as multiplication by the reciprocal of y, provided that $0 \notin y$.

$$\frac{x}{y} = x \cdot \left[\frac{1}{\overline{y}}, \frac{1}{\underline{y}}\right], \quad \text{if } 0 \notin [\underline{y}, \overline{y}]$$

Suppose the divisor interval y contains zero. In that case, the operation is typically undefined or results in extended intervals (e.g., a union of two intervals), depending on the specific framework being used [23].

E.2 Interval Matrix Multiplication

The operations of IA extend naturally to linear algebra, enabling computations with interval matrices. An interval matrix $\mathbf{A}$ is a matrix whose elements are intervals. Given two compatible interval matrices, $\mathbf{A}$ and $\mathbf{B}$, their product $\mathbf{C} = \mathbf{AB}$ is an interval matrix whose elements c_{ij} are defined by the interval extension of the standard dot product:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}.$$

Here, each multiplication $a_{ik}b_{kj}$ is an interval multiplication, and the summation $\sum$ represents a sequence of interval additions. This operation is fundamental for solving interval linear systems, which are crucial for analyzing the effects of bounded uncertainties in models. However, this definition is highly susceptible to the dependency problem; if the same interval variable appears in multiple entries of $\mathbf{A}$ or $\mathbf{B}$, the resulting bounds on $\mathbf{C}$ can be much wider than the true range.

E.3 Interval Convolutions

IA can be extended to operations like convolutions, which are fundamental in areas such as signal processing and image analysis. By representing an input signal $\mathbf{X}$ and a filter kernel $\mathbf{W}$ as intervals, one can analyze systems where inputs or parameters are subject to bounded uncertainties.

A key operation is the interval convolution. For a 2D input $\mathbf{X}$, an interval kernel (filter) $\mathbf{W}$, and an optional interval bias $\mathbf{b}$, the resulting output feature map $\mathbf{Y}$ is computed as:

$$\mathbf{Y}_{ij} = \left(\sum_{k,l} \mathbf{X}_{i+k,j+l} \cdot \mathbf{W}_{k,l} \right) + \mathbf{b}.$$

This computation propagates the interval bounds through the filtering operation, producing a rigorous enclosure for the true output. This is particularly useful in robust state estimation and control, where system parameters or measurements are known to lie within certain bounds [59]. In InTAct, only the layer inputs are intervals, whereas all weights (e.g., $\mathbf{W}$ and $\Delta\mathbf{W}$) are real-valued.

E.4 Properties and Practical Considerations

While IA provides a mechanism for bounding activations, two characteristics are notable for implementation. First, theoretically, IA requires *directed rounding*, expanding the bounds slightly outward, to strictly guarantee that no precision errors are excluded. However, in the context of neural networks, these microscopic floating-point errors are negligible compared to the inherent high variance and noise of the feature activations.

Second, IA is subject to the *dependency problem*. When a variable appears multiple times in an expression, each occurrence is treated independently, leading to an overestimation of the true range. For example, computing $x - x$ for $x \in [0, 1]$ yields $[-1, 1]$ rather than the true set $\{0\}$. A related source of overestimation is the *wrapping effect*: when intervals are propagated through successive layers, the image of a box is repeatedly enclosed in a larger axis-aligned box. InTAct avoids both effects, as it bounds the drift of one affine layer at a time, in which each coordinate of the input enters every output only once; the resulting drift bounds are therefore exact (Appendix G).

F Metrics

We assess continual learning performance using AA. Let T be the total number of tasks, and $R_{i,j}$ denote the test accuracy on task i after the model has finished training on task j. Average Accuracy (AA) reports the mean performance across all tasks after the final training session:

$$AA = \frac{1}{T} \sum_{i=1}^{T} R_{i,T}.$$

G Detailed Derivation of Internal Representation Drift Loss

In this section, we provide the step-by-step derivation of the regularization term $\mathcal{L}_{\text{IntDrift}}$. Our goal is to ensure that the parameter updates for a layer do not alter the layer's output for any input belonging to the *established activation region* of previous tasks.

The Stability Condition. Consider the l-th affine layer of the network with weights $\mathbf{W}_l$ and bias $\mathbf{b}_l$. Let $\mathbf{z} \in \mathbb{R}^{D_{in}}$ denote the input to this layer (i.e., the output of layer $l-1$). The preactivation output is given by:

$$f(\mathbf{z}; \boldsymbol{\theta}_l) = \mathbf{W}_l \mathbf{z} + \mathbf{b}_l.$$

After training on a new task, the parameters update to $\mathbf{W}_l + \Delta\mathbf{W}_l$ and $\mathbf{b}_l + \Delta\mathbf{b}_l$. The drift in the preactivation for a given input $\mathbf{z}$ is:

$$\Delta f(\mathbf{z}) = f(\mathbf{z}; \boldsymbol{\theta}_l + \Delta\boldsymbol{\theta}_l) - f(\mathbf{z}; \boldsymbol{\theta}_l) = \Delta\mathbf{W}_l \mathbf{z} + \Delta\mathbf{b}_l.$$

To prevent catastrophic forgetting, we require this drift to be zero for all inputs residing within the hypercube $\mathcal{H}$ defined by the previous tasks. Let this hypercube be denoted by the interval vector $[\underline{\mathbf{z}}, \overline{\mathbf{z}}]$. The condition is:

$$\Delta\mathbf{W}_l \mathbf{z} + \Delta\mathbf{b}_l = \mathbf{0}, \quad \forall \mathbf{z} \in [\underline{\mathbf{z}}, \overline{\mathbf{z}}].$$

Interval Extension. Since checking every point $\mathbf{z}$ is impossible, we use IA to bound the drift $\Delta f(\mathbf{z})$. The interval extension of the linear map $\Delta \mathbf{W}_l \mathbf{z}$ computes the tightest box containing the image of the input hypercube. For the i-th output coordinate, the drift bounds $[\underline{\delta}_i, \overline{\delta}_i]$ are computed as:

$$[\underline{\delta}_i, \overline{\delta}_i] = \Delta \mathbf{W}_{l,i} \cdot [\underline{\mathbf{z}}, \overline{\mathbf{z}}] + \Delta b_{l,i},$$

where $\Delta \mathbf{W}_{l,i}$ is the i-th row of the update matrix.

Decomposition into Positive and Negative Parts. In IA, the product of a scalar w and an interval $[a, b]$ depends on the sign of w:

$$w \cdot [a, b] = \begin{cases} [wa, wb] & \text{if } w \geq 0 \\ [wb, wa] & \text{if } w < 0 \end{cases}.$$

To vectorize this operation, we decompose the weight row $\Delta \mathbf{W}_{l,i}$ into its positive and negative components:

$$\Delta \mathbf{W}_{l,i}^{+} = \max(0, \Delta \mathbf{W}_{l,i}), \quad \Delta \mathbf{W}_{l,i}^{-} = \max(0, -\Delta \mathbf{W}_{l,i}).$$

Thus, $\Delta \mathbf{W}_{l,i} = \Delta \mathbf{W}_{l,i}^{+} - \Delta \mathbf{W}_{l,i}^{-}$. Applying interval arithmetic, we obtain:

$$\begin{aligned} \Delta \mathbf{W}_{l,i} \cdot [\underline{\mathbf{z}}, \overline{\mathbf{z}}] &= (\Delta \mathbf{W}_{l,i}^{+} - \Delta \mathbf{W}_{l,i}^{-}) \cdot [\underline{\mathbf{z}}, \overline{\mathbf{z}}] \\ &= [\Delta \mathbf{W}_{l,i}^{+} \underline{\mathbf{z}}, \Delta \mathbf{W}_{l,i}^{+} \overline{\mathbf{z}}] - [\Delta \mathbf{W}_{l,i}^{-} \underline{\mathbf{z}}, \Delta \mathbf{W}_{l,i}^{-} \overline{\mathbf{z}}] \\ &= [\Delta \mathbf{W}_{l,i}^{+} \underline{\mathbf{z}} - \Delta \mathbf{W}_{l,i}^{-} \overline{\mathbf{z}}, \Delta \mathbf{W}_{l,i}^{+} \overline{\mathbf{z}} - \Delta \mathbf{W}_{l,i}^{-} \underline{\mathbf{z}}]. \end{aligned}$$

The second equality is exact because $\Delta \mathbf{W}_{l,i}^{+}$ and $\Delta \mathbf{W}_{l,i}^{-}$ have disjoint supports, so each coordinate of $\mathbf{z}$ enters only one of the two terms. Using the interval subtraction rule $[a, b] - [c, d] = [a - d, b - c]$, we derive the bounds for the drift:

$$\begin{aligned} \underline{\delta}_i &= \Delta \mathbf{W}_{l,i}^{+} \underline{\mathbf{z}} - \Delta \mathbf{W}_{l,i}^{-} \overline{\mathbf{z}} + \Delta b_{l,i}, \\ \overline{\delta}_i &= \Delta \mathbf{W}_{l,i}^{+} \overline{\mathbf{z}} - \Delta \mathbf{W}_{l,i}^{-} \underline{\mathbf{z}} + \Delta b_{l,i}. \end{aligned}$$

Loss Formulation. The condition $\Delta f(\mathbf{z}) = 0$ requires the interval $[\underline{\delta}_i, \overline{\delta}_i]$ to collapse to the singleton set $[0, 0]$. To enforce this behavior, we minimize the squared magnitude of the interval bounds across all protected layers, denoted by $\mathcal{S}$:

$$\mathcal{L}_{\text{IntDrift}} = \sum_{l \in \mathcal{S}} \sum_{i} \left((\underline{\delta}_{l,i})^2 + (\overline{\delta}_{l,i})^2 \right).$$

Here, $\underline{\delta}_{l,i}$ and $\overline{\delta}_{l,i}$ denote the bounds $\underline{\delta}_i$ and $\overline{\delta}_i$ of layer l. For clarity, constant normalization factors are omitted from the formulation above. Minimizing this objective progressively shrinks the admissible drift interval, thereby enforcing functional consistency between the current and previous model within the protected regions of the feature space.

H Internal Representation Drift for Convolutional Layers

Convolutional Layer. We extend the derivation of $\mathcal{L}_{\text{IntDrift}}$ to 2D convolutional layers. Let $f(\mathbf{z}; \boldsymbol{\theta}_l) = \text{Conv2d}(\mathbf{z}, \mathbf{W}_l) + \mathbf{b}_l$ denote the layer operation. Since convolution is a linear operator applied to local spatial patches, it can be reformulated as an affine transformation on unfolded input vectors.

Consider a single output channel i. The preactivation at a specific spatial location is computed by the dot product of the flattened filter kernel $\mathbf{k}_i$ (derived from $\mathbf{W}_l$) and the corresponding flattened input patch $\mathbf{z}_p$. Let $\Delta \mathbf{k}_i$ and Δb_i represent the parameter updates. The stability condition requires the drift to be zero for any patch $\mathbf{z}_p$ within the established bounds $[\underline{\mathbf{z}}_p, \overline{\mathbf{z}}_p]$:

$$\Delta \mathbf{k}_i^{\top} \mathbf{z}_p + \Delta b_i = 0, \quad \forall \mathbf{z}_p \in [\underline{\mathbf{z}}_p, \overline{\mathbf{z}}_p].$$

By treating the flattened patch $\mathbf{z}_p$ as the input vector, this formulation becomes mathematically identical to the affine case derived in Appendix G. We apply the same IA decomposition. Let $\Delta \mathbf{k}_i^{+}$

and $\Delta\mathbf{k}_i^-$ denote the positive and negative components of the filter update. The bounds on the drift $[\underline{\delta}_i, \overline{\delta}_i]$ for this channel are:

$$\underline{\delta}_i = (\Delta\mathbf{k}_i^+)^\top \underline{\mathbf{z}}_p - (\Delta\mathbf{k}_i^-)^\top \overline{\mathbf{z}}_p + \Delta b_i,$$
$$\overline{\delta}_i = (\Delta\mathbf{k}_i^+)^\top \overline{\mathbf{z}}_p - (\Delta\mathbf{k}_i^-)^\top \underline{\mathbf{z}}_p + \Delta b_i.$$

The loss $\mathcal{L}_{\text{IntDrift}}$ is then computed by minimizing the squared L_2 norm of these bounds, summed across all channels and spatial locations. This ensures that the convolutional filters remain functionally invariant within the protected local geometric regions of past tasks.

Batch Normalization Layer. During inference, a Batch Normalization (BN) layer operates as a channel-wise affine transformation. Let $\boldsymbol{\gamma}, \boldsymbol{\beta}$ be the learnable parameters and $\boldsymbol{\mu}, \boldsymbol{\sigma}^2$ be the frozen running statistics. The layer operation is:

$$f(\mathbf{z}; \boldsymbol{\theta}) = \underbrace{\frac{\boldsymbol{\gamma}}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}}}_{\mathbf{W}_{\text{eff}}} \mathbf{z} + \underbrace{\left(\boldsymbol{\beta} - \frac{\boldsymbol{\gamma}\boldsymbol{\mu}}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}}\right)}_{\mathbf{b}_{\text{eff}}}.$$

When the parameters update by $(\Delta\boldsymbol{\gamma}, \Delta\boldsymbol{\beta})$, the resulting functional drift is equivalent to an affine update with effective parameters:

$$\Delta\mathbf{W}_{\text{eff}} = \frac{\Delta\boldsymbol{\gamma}}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}}, \quad \Delta\mathbf{b}_{\text{eff}} = \Delta\boldsymbol{\beta} - \frac{\Delta\boldsymbol{\gamma}\boldsymbol{\mu}}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}}.$$

Since the drift form $\Delta\mathbf{W}_{\text{eff}}\mathbf{z} + \Delta\mathbf{b}_{\text{eff}}$ is identical to the standard affine case, we compute $\mathcal{L}_{\text{IntDrift}}$ directly using the interval bounds derived in Appendix G, substituting $\Delta\mathbf{W}_l$ and $\Delta\mathbf{b}_l$ with $\Delta\mathbf{W}_{\text{eff}}$ and $\Delta\mathbf{b}_{\text{eff}}$.

I Facts and Proofs

In this section, we provide the full mathematical formalization and proofs supporting Theorem 1 from the main text.

Assumptions. Let $f, f' : \mathcal{Z} \to \mathcal{Y}$ be linear mappings from the latent representation space $\mathcal{Z} \subseteq \mathbb{R}^d$ to the output space $\mathcal{Y} \subseteq \mathbb{R}^N$, where d and N denote the respective dimensions. Thus, for all $\mathbf{x}, \mathbf{y} \in \mathcal{Z}$ and $a, b \in \mathbb{R}$, they satisfy $f(a\mathbf{x} + b\mathbf{y}) = af(\mathbf{x}) + bf(\mathbf{y})$, and analogously for f'. We define the functional perturbation as $\Delta f \triangleq f' - f$.

Although stated for linear mappings, all results below also extend to affine mappings of the form $f(\mathbf{z}) = \mathbf{W}\mathbf{z} + \mathbf{b}$ and $f'(\mathbf{z}) = \mathbf{W}'\mathbf{z} + \mathbf{b}'$. In this case, $\Delta f_i(\mathbf{e}_j)$ is interpreted as the corresponding entry ΔW_{ij} of $\Delta\mathbf{W} = \mathbf{W}' - \mathbf{W}$. This extension follows because Fact 1 does not require linearity, while the bias perturbation cancels in the differences $\overline{\delta}_i - \underline{\delta}_i$ and $\Delta f_i(\mathbf{z}^*) - \Delta f_i(\tilde{\mathbf{z}})$ used in the proofs of Facts 2 and 3.

Let $\mathcal{H} = \prod_{j=1}^d [\underline{z}_j, \overline{z}_j] \subseteq \mathcal{Z}$ denote the protected hypercube, where $\underline{z}_j$ and $\overline{z}_j$ are the lower and upper bounds of historical activations along the j-th coordinate, respectively. Importantly, $\mathcal{H}$ need not contain every historical representation. We define its diameter along dimension j as $\text{diam}_j(\mathcal{H}) \triangleq \overline{z}_j - \underline{z}_j$ and assume $\text{diam}_j(\mathcal{H}) > 0$ for all $j \in \{1, \ldots, d\}$.

We further consider the internal drift loss:

$$\mathcal{L}_{\text{IntDrift}} \triangleq \frac{\lambda}{N} \sum_{i=1}^{N} \left[\left(\sup_{\mathbf{z} \in \mathcal{H}} \Delta f_i(\mathbf{z}) \right)^2 + \left(\inf_{\mathbf{z} \in \mathcal{H}} \Delta f_i(\mathbf{z}) \right)^2 \right],$$

where $\lambda > 0$ is the regularization weight (corresponding to $\lambda_{\text{IntDrift}}$ in Eq. (1)), and $\Delta f_i(\mathbf{z})$ denotes the i-th component of the output perturbation $\Delta f(\mathbf{z})$. We assume that, for some $\varepsilon > 0$, this loss satisfies $\mathcal{L}_{\text{IntDrift}} \leq \varepsilon$.

Formal Statements. Under the above assumptions, we establish three bounds on the functional perturbation:

- **Fact 1 (Drift within the protected region).** For any $\mathbf{z} \in \mathcal{H}$, the absolute functional perturbation is uniformly bounded:

$$\|\Delta f(\mathbf{z})\|_\infty \leq \sqrt{\frac{N\varepsilon}{\lambda}} \triangleq \kappa.$$

- **Fact 2 (Sensitivity along coordinate directions).** For any output dimension i and coordinate direction $\mathbf{e}_j$, the functional perturbation satisfies:

$$|\Delta f_i(\mathbf{e}_j)| \leq \frac{1}{\mathrm{diam}_j(\mathcal{H})} \sqrt{\frac{2N\varepsilon}{\lambda}}.$$

- **Fact 3 (Drift outside the protected region).** For any $\mathbf{z}^* \notin \mathcal{H}$, define its coordinate-wise distance from the hypercube as

$$d_j(\mathbf{z}^*) \triangleq \min_{\mathbf{z}' \in \mathcal{H}} |z_j^* - z_j'|.$$

  The functional perturbation is then bounded by:

$$|\Delta f_i(\mathbf{z}^*)| \leq \kappa + \sum_{j=1}^{d} \frac{d_j(\mathbf{z}^*)}{\mathrm{diam}_j(\mathcal{H})} \sqrt{\frac{2N\varepsilon}{\lambda}}.$$

Proof of Fact 1. For each output dimension i, let

$$\overline{\delta}_i \triangleq \sup_{\mathbf{z} \in \mathcal{H}} \Delta f_i(\mathbf{z}),$$
$$\underline{\delta}_i \triangleq \inf_{\mathbf{z} \in \mathcal{H}} \Delta f_i(\mathbf{z}).$$

Since $\mathcal{H}$ is compact and Δf_i is continuous, both extrema are finite and attained. Thus, for any $\mathbf{z} \in \mathcal{H}$,

$$\underline{\delta}_i \leq \Delta f_i(\mathbf{z}) \leq \overline{\delta}_i,$$

which implies

$$|\Delta f_i(\mathbf{z})|^2 \leq \max\left\{(\underline{\delta}_i)^2, (\overline{\delta}_i)^2\right\} \leq (\underline{\delta}_i)^2 + (\overline{\delta}_i)^2.$$

By the assumption $\mathcal{L}_{\mathrm{IntDrift}} \leq \varepsilon$,

$$\frac{\lambda}{N} \sum_{k=1}^{N} \left[(\overline{\delta}_k)^2 + (\underline{\delta}_k)^2\right] \leq \varepsilon.$$

Since all terms are non-negative, each individual term satisfies

$$(\overline{\delta}_i)^2 + (\underline{\delta}_i)^2 \leq \frac{N\varepsilon}{\lambda}.$$

Combining the above inequalities, we obtain, for every $\mathbf{z} \in \mathcal{H}$ and output dimension i,

$$|\Delta f_i(\mathbf{z})| \leq \sqrt{\frac{N\varepsilon}{\lambda}}.$$

Taking the maximum over all output dimensions yields

$$\|\Delta f(\mathbf{z})\|_\infty \leq \sqrt{\frac{N\varepsilon}{\lambda}} \triangleq \kappa,$$

which completes the proof. □

Proof of Fact 2. By linearity, any $\mathbf{z} \in \mathcal{Z}$ can be expressed in the canonical basis as $\mathbf{z} = \sum_{j=1}^{d} z_j \mathbf{e}_j$, giving

$$\Delta f_i(\mathbf{z}) = \sum_{j=1}^{d} z_j \Delta f_i(\mathbf{e}_j).$$

Since $\mathcal{H}$ is a Cartesian product of independent coordinate intervals, the supremum and infimum of this linear expression decompose across dimensions:

$$\overline{\delta}_i = \sup_{\mathbf{z}\in\mathcal{H}} \Delta f_i(\mathbf{z}) = \sum_{j=1}^{d} \max_{z_j \in [\underline{z}_j, \overline{z}_j]} z_j \Delta f_i(\mathbf{e}_j),$$

$$\underline{\delta}_i = \inf_{\mathbf{z}\in\mathcal{H}} \Delta f_i(\mathbf{z}) = \sum_{j=1}^{d} \min_{z_j \in [\underline{z}_j, \overline{z}_j]} z_j \Delta f_i(\mathbf{e}_j).$$

Using the identity $\max_{x\in[a,b]} cx - \min_{x\in[a,b]} cx = |c|(b-a)$, we obtain

$$\overline{\delta}_i - \underline{\delta}_i = \sum_{j=1}^{d} |\Delta f_i(\mathbf{e}_j)| \mathrm{diam}_j(\mathcal{H}).$$

Next, applying $a^2 + b^2 \geq \frac{1}{2}(a-b)^2$ yields

$$\begin{aligned}(\overline{\delta}_i)^2 + (\underline{\delta}_i)^2 &\geq \frac{1}{2}(\overline{\delta}_i - \underline{\delta}_i)^2 \\ &= \frac{1}{2}\left(\sum_{j=1}^{d} |\Delta f_i(\mathbf{e}_j)| \mathrm{diam}_j(\mathcal{H})\right)^2.\end{aligned}$$

By the assumption $\mathcal{L}_{\mathrm{IntDrift}} \leq \varepsilon$ and the non-negativity of all terms in its definition, each output dimension satisfies

$$(\overline{\delta}_i)^2 + (\underline{\delta}_i)^2 \leq \frac{N\varepsilon}{\lambda}.$$

Combining the above inequalities gives

$$\frac{1}{2}\left(\sum_{j=1}^{d} |\Delta f_i(\mathbf{e}_j)| \mathrm{diam}_j(\mathcal{H})\right)^2 \leq \frac{N\varepsilon}{\lambda}.$$

Since every summand is non-negative, the sum dominates each individual term. Therefore, for every j,

$$\frac{1}{2}\left(|\Delta f_i(\mathbf{e}_j)| \mathrm{diam}_j(\mathcal{H})\right)^2 \leq \frac{N\varepsilon}{\lambda}.$$

Finally, using $\mathrm{diam}_j(\mathcal{H}) > 0$ and taking square roots, we obtain

$$|\Delta f_i(\mathbf{e}_j)| \leq \frac{1}{\mathrm{diam}_j(\mathcal{H})} \sqrt{\frac{2N\varepsilon}{\lambda}},$$

which completes the proof. □

Proof of Fact 3. To demonstrate that this directional bound preserves functional plasticity for novel tasks, we evaluate the permissible functional drift for a $\mathbf{z}^* \notin \mathcal{H}$.

Let $\mathbf{z}^*$ possess features in the complement of the historical intervals. We define its distance to hypercube $\mathcal{H}$ along the j-th dimension as $d_j(\mathbf{z}^*) \triangleq \min_{\mathbf{z}'\in\mathcal{H}} |z_j^* - z_j'|$.

Let $\tilde{\mathbf{z}} \in \mathcal{H}$ be the closest boundary point to $\mathbf{z}^*$, i.e., $\tilde{z}_j = \min(\max(z_j^*, \underline{z}_j), \overline{z}_j)$ for all j. By the properties of vector spaces, $\mathbf{z}^* = \tilde{\mathbf{z}} + (\mathbf{z}^* - \tilde{\mathbf{z}})$. Due to the linearity of the operator, we can perfectly decompose the functional drift:

$$\Delta f_i(\mathbf{z}^*) = \Delta f_i(\tilde{\mathbf{z}}) + \Delta f_i(\mathbf{z}^* - \tilde{\mathbf{z}}) = \Delta f_i(\tilde{\mathbf{z}}) + \sum_{j=1}^{d} \Delta f_i(\mathbf{e}_j)(z_j^* - \tilde{z}_j)$$

Taking the absolute magnitude and applying the triangle inequality:

$$|\Delta f_i(\mathbf{z}^*)| \leq |\Delta f_i(\tilde{\mathbf{z}})| + \sum_{j=1}^{d} |\Delta f_i(\mathbf{e}_j)||z_j^* - \tilde{z}_j|$$

By the definition of the closest point $\tilde{\mathbf{z}}$, the term $|z_j^* - \tilde{z}_j|$ is exactly the minimum distance $d_j(\mathbf{z}^*)$. From Fact 1, since $\tilde{\mathbf{z}} \in \mathcal{H}$, we know that $|\Delta f_i(\tilde{\mathbf{z}})| \leq \kappa$. Substituting our bound from Fact 2 yields the maximum permissible functional plasticity for the novel representation:

$$|\Delta f_i(\mathbf{z}^*)| \leq \kappa + \sum_{j=1}^{d} \left(\frac{d_j(\mathbf{z}^*)}{\mathrm{diam}_j(\mathcal{H})} \right) \sqrt{\frac{2N\varepsilon}{\lambda}}$$

□

I.1 Proof of Theorem 1

Proof. The theorem follows directly from Facts 1 and 3. Consider an arbitrary representation $\mathbf{z} \in \mathcal{Z}$.

Case 1: $\mathbf{z} \in \mathcal{H}$. In this case, $d_j(\mathbf{z}) = 0$ for all j. By Fact 1,

$$|\Delta f_i(\mathbf{z})| \leq \kappa = \sqrt{\frac{N\varepsilon}{\lambda}} = \mathcal{O}(\sqrt{\varepsilon}),$$

which recovers the theorem's bound, as the distance-dependent term vanishes.

Case 2: $\mathbf{z} \notin \mathcal{H}$. Applying Fact 3 directly gives

$$|\Delta f_i(\mathbf{z})| \leq \sqrt{\frac{N\varepsilon}{\lambda}} + \sum_{j=1}^{d} \frac{d_j(\mathbf{z})}{\mathrm{diam}_j(\mathcal{H})} \sqrt{\frac{2N\varepsilon}{\lambda}}.$$

Since N and λ are fixed, both coefficients scale as $\mathcal{O}(\sqrt{\varepsilon})$. Consequently,

$$|\Delta f_i(\mathbf{z})| \leq \mathcal{O}(\sqrt{\varepsilon}) + \sum_{j=1}^{d} \frac{d_j(\mathbf{z})}{\mathrm{diam}_j(\mathcal{H})} \mathcal{O}(\sqrt{\varepsilon}),$$

which establishes the claimed bound in Theorem 1. □

I.2 Extension to Lipschitz Activation Functions

Theorem 1 bounds the drift of the pre-activation outputs $f(\mathbf{z})$ of a protected layer. Consider an element-wise activation function σ that is L_σ-Lipschitz, i.e., $|\sigma(a) - \sigma(b)| \leq L_\sigma |a - b|$ for all $a, b \in \mathbb{R}$. Then, for every output unit i,

$$\left|\sigma\big(f_i'(\mathbf{z})\big) - \sigma\big(f_i(\mathbf{z})\big)\right| \leq L_\sigma \, |\Delta f_i(\mathbf{z})|.$$

Consequently, the bounds in Fact 1, Fact 3, and Theorem 1 also apply to the post-activation outputs, with their right-hand sides scaled by L_σ. For ReLU and leaky ReLU, $L_\sigma = 1$, so these activations do not amplify functional drift. For GELU, $L_\sigma \approx 1.13$.

J Training Algorithm

The complete training procedure for InTAct is detailed in Algorithm 1.

Optimization Setup. We optimize the set of learnable parameters $\mathbf{\Psi} = \{\boldsymbol{\phi}, \mathbf{w}\}$, comprising the prompt pool parameters $\boldsymbol{\phi}$ and the classifier weights $\mathbf{w}$, while keeping the backbone $f_{\boldsymbol{\theta}}$ frozen. The training process incorporates a standard prompt-query mechanism QUERYPROMPTS that retrieves task-relevant prompts and computes any method-specific auxiliary losses (e.g., orthogonality penalties), denoted as $\mathcal{L}_{\text{aux}}$.

Training Dynamics. The optimization strategy gradually balances stability and adaptability across tasks. During the first task ($t = 1$), the model optimizes the classification loss together with the variance regularization $\mathcal{L}_{\text{Var}}$, which is crucial for compressing early task activations into compact, well-defined regions. For subsequent tasks ($t \geq 2$), the full regularization is applied by incorporating $\mathcal{L}_{\text{IntDrift}}$, $\mathcal{L}_{\text{Feat}}$, and $\mathcal{L}_{\text{Align}}$. These terms encourage the model to produce consistent outputs for previously seen data, ensuring that new learning does not interfere with the predictions established for earlier tasks.

Algorithm 1: InTAct Training Procedure

Require: Sequence of T tasks $\{\mathcal{D}_t\}_{t=1}^{T}$, Frozen backbone $f_{\boldsymbol{\theta}}$.
Require: Learnable classification head $g(\cdot; \mathbf{w})$ and prompts $\boldsymbol{\phi}$.
Require: Learning rate η, number of epochs E, Hyperparameters $\{\lambda_{\text{IntDrift}}, \lambda_{\text{Feat}}, \lambda_{\text{Var}}, \lambda_{\text{Align}}, \lambda_{\text{Prompt}}, \alpha\}$.
Ensure: Optimized parameters $\boldsymbol{\Psi} = \{\boldsymbol{\phi}, \mathbf{w}\}$.

Initialize: $\mathcal{S}$ as the subset of layers for hypercube tracking; $\mathcal{H}_l^{(\leq 0)} \leftarrow \emptyset$ for all $l \in \mathcal{S}$.
for task $t = 1$ to T **do**
  for epoch $e = 1$ to E **do**
    for batch $(\mathbf{x}, y) \sim \mathcal{D}_t$ **do**
      $\mathbf{x}_p, \mathcal{L}_{\text{aux}} \leftarrow \text{QUERYPROMPTS}(\mathbf{x})$ {Retrieve task-specific prompts}
      $\mathbf{z} \leftarrow f_{\boldsymbol{\theta}}(\mathbf{x}_p)$ {Representation from backbone}
      $\hat{\mathbf{y}} \leftarrow g(\mathbf{z}; \mathbf{w})$ {Forward pass through classification head}
      $\mathcal{L}_{\text{Total}} \leftarrow \mathcal{L}_{\text{CE}}(y, \hat{\mathbf{y}}) + \lambda_{\text{Prompt}}\mathcal{L}_{\text{aux}} + \mathcal{L}_{\text{Var}}$
      if $t > 1$ **then**
        $\mathcal{L}_{\text{stab}} \leftarrow \mathcal{L}_{\text{Feat}} + \mathcal{L}_{\text{IntDrift}} + \mathcal{L}_{\text{Align}}$
        $\mathcal{L}_{\text{Total}} \leftarrow \mathcal{L}_{\text{Total}} + \mathcal{L}_{\text{stab}}$
      end if
      $\boldsymbol{\Psi} \leftarrow \boldsymbol{\Psi} - \eta \nabla_{\boldsymbol{\Psi}} \mathcal{L}_{\text{Total}}$
    end for
  end for
  Update Cumulative Hypercube:
  for each layer $l \in \mathcal{S}$ **do**
    $\mathcal{H}_{t,l} \leftarrow \text{PERCENTILEBOUNDS}(\mathcal{Z}_{t,l}, \alpha)$ {$\mathcal{Z}_{t,l}$: representations of task t at layer l}
    if $\mathcal{H}_l^{(\leq t-1)} = \emptyset$ **then**
      $\mathcal{H}_l^{(\leq t)} \leftarrow \mathcal{H}_{t,l}$
    else
      $\mathcal{H}_l^{(\leq t)} \leftarrow \mathcal{H}_l^{(\leq t-1)} \sqcup \mathcal{H}_{t,l}$ {Expand hypercubes via interval hull}
    end if
  end for
end for

K InTAct in Other Prompt-based Methods

Table 4 reports the results of integrating InTAct into L2P, DualPrompt, and CODA-Prompt on DomainNet and ImageNet-R. We use the optimal hyperparameter settings for class-incremental learning (CIL) identified in the comprehensive CODA-Prompt study [19], applying them consistently to both the baseline methods and their InTAct-augmented variants.

The results demonstrate that InTAct provides a modular enhancement to existing prompt-based frameworks, yielding substantial performance gains. On ImageNet-R, it improves the AA of L2P by $+\mathbf{8.0}$ p.p., DualPrompt by $+\mathbf{8.5}$ p.p., and CODA-Prompt by $+\mathbf{13.0}$ p.p.; on DomainNet, the gains are smaller ($+4.1$ to $+6.0$ p.p.). These improvements indicate that InTAct helps existing methods better balance stability and plasticity. Notably, the gains are larger than those obtained when integrating InTAct into KA-Prompt. A possible explanation is that KA-Prompt maintains an exponential moving average (EMA) of its classifier weights, which helps mitigate classifier drift, whereas L2P, DualPrompt, and CODA-Prompt do not include a comparable mechanism. We also use CODA-Prompt + InTAct for the ablation and sensitivity analyses in Sec. 4.3 and 4.4. On ImageNet-C and ImageNet-Mix, this configuration achieves 85.85 ± 0.34 and 74.20 ± 0.28, respectively.

Finally, the results in this table are not directly comparable in absolute terms with those in Table 1, as the experiments use different backbone configurations. All methods in this table use ViT-B/16 weights fine-tuned on ImageNet-1k.

Table 4: **Performance results of integrating InTAct with L2P, DualPrompt, and CODA-Prompt.** We report AA for ImageNet-R and DomainNet in the DIL setting. Incorporating InTAct consistently yields performance gains over prompt-based baselines. Results are averaged over 5 independent runs.

Method	DomainNet	ImageNet-R	**Average**
L2P	48.44 ± 0.09	50.34 ± 0.26	49.39 ± 0.18
+ InTAct	$\mathbf{53.44 \pm 0.02}$	$\mathbf{58.34 \pm 0.24}$	$\mathbf{55.89 \pm 0.13}$
DualPrompt	50.87 ± 0.31	53.31 ± 0.36	52.09 ± 0.34
+ InTAct	$\mathbf{56.83 \pm 0.28}$	$\mathbf{61.85 \pm 0.45}$	$\mathbf{59.34 \pm 0.37}$
CODA-Prompt	52.52 ± 0.22	61.21 ± 0.49	56.87 ± 0.36
+ InTAct	$\mathbf{56.66 \pm 0.39}$	$\mathbf{74.23 \pm 0.16}$	$\mathbf{65.45 \pm 0.28}$

L Additional Results

We provide additional results and analyses to further assess the effectiveness of InTAct.

Pairwise Ablation. Tab. 5 extends the ablation in Tab. 2 by evaluating all six pairwise removals of loss components on ImageNet-R. The strongest interaction occurs between $\mathcal{L}_{\text{IntDrift}}$ and $\mathcal{L}_{\text{Var}}$: removing both reduces AA by 8.43 p.p., nearly twice the combined drop from removing each individually (4.43 p.p.). This suggests that the two components reinforce each other, with $\mathcal{L}_{\text{Var}}$ shaping the intervals whose functional drift is controlled by $\mathcal{L}_{\text{IntDrift}}$. The other pairwise removals result in drops close to or smaller than the sum of the corresponding individual drops, suggesting that these components play complementary, but partially overlapping, roles.

Table 5: **Pairwise ablation on ImageNet-R.** Extension of Tab. 2 with the joint removal of every pair of loss components. We report the performance drop (Δ) relative to the full InTAct.

$\mathcal{L}_{\text{IntDrift}}$	$\mathcal{L}_{\text{Var}}$	$\mathcal{L}_{\text{Align}}$	$\mathcal{L}_{\text{Feat}}$	**AA (%)**	Δ
✓	✓	✓	✓	**74.23**	—
×	✓	✓	✓	70.75	-3.48
✓	×	✓	✓	73.28	-0.95
✓	✓	×	✓	73.76	-0.47
✓	✓	✓	×	72.44	-1.79
✓	×	×	✓	73.08	-1.15
✓	×	✓	×	72.03	-2.20
✓	✓	×	×	72.92	-1.31
×	×	✓	✓	65.80	-8.43
×	✓	×	✓	70.18	-4.05
×	✓	✓	×	70.72	-3.51

Gradient Interaction between $\mathcal{L}_{\text{IntDrift}}$ and $\mathcal{L}_{\text{Align}}$. To examine whether the alignment objective conflicts with the interval drift constraint, we measure the cosine similarity between the gradients of $\mathcal{L}_{\text{IntDrift}}$ and $\mathcal{L}_{\text{Align}}$ with respect to the trainable parameters at every training iteration of CODA-Prompt + InTAct on ImageNet-R. For the first three tasks in which both objectives are active ($t = 2, 3, 4$), the average cosine similarities are 0.007, 0.012, and 0.007, respectively. Although occasional negative values occur, they remain small (minimum -0.067). These results indicate that the two objectives are largely orthogonal and do not systematically oppose each other. In particular, $\mathcal{L}_{\text{Align}}$ acts as a mild geometric regularizer without consistently driving new representations toward regions strongly constrained by $\mathcal{L}_{\text{IntDrift}}$. This interpretation is consistent with the pairwise ablation results, where removing both terms leads to a performance drop close to the sum of their individual effects (Tab. 5).

Comparison with HINT. We additionally compare InTAct with HINT [7], an interval-based continual learning method. We focus on HINT rather than InterContiNet [6], as HINT outperforms it

across the benchmarks considered in [7]. To ensure a controlled comparison, we implement HINT within the CODA-Prompt framework, keeping all other components unchanged. Following the original formulation, a universal embedding is passed through a hypernetwork (a single-layer MLP with 50 hidden neurons) to generate the classifier weights. A perturbation radius defines a hypercube around each task embedding. The hypercubes associated with the current and previous tasks are then intersected, and the universal embedding is set to the center of this intersection.

We grid-search the embedding dimension ($\{48, 96\}$), perturbation radius ($\{0.05, 0.1\}$), and hypernetwork regularization weight ($\{0.01, 0.1\}$). The best configurations are $(48, 0.1, 0.01)$ for ImageNet-R and $(48, 0.1, 0.1)$ for DomainNet. As reported in Tab. 6, InTAct outperforms HINT by 32.8 p.p. on ImageNet-R and 4.6 p.p. on DomainNet.

Table 6: **Comparison with HINT.** AA of HINT and InTAct, both implemented in the CODA-Prompt framework.

Method	ImageNet-R	DomainNet
HINT [7]	41.42 ± 1.26	52.02 ± 0.08
InTAct	$\mathbf{74.23 \pm 0.16}$	$\mathbf{56.66 \pm 0.39}$

Class-Incremental Learning. Tab. 7 reports results on Split MNIST, Split Fashion-MNIST (Split FMNIST), and Split CIFAR-10. To enable controlled comparisons within a reasonable computational budget, we evaluate all methods using the same two-layer MLP architecture, with 400 neurons per layer. Across all three benchmarks, InTAct consistently outperforms standard parameter-space regularization and feature-distillation baselines (EWC, LwF, MAS, and SI).

The gains are particularly pronounced on Split MNIST and Split FMNIST, where InTAct surpasses the strongest baseline, LwF, by 6.64 and 8.33 p.p., respectively. On the more challenging Split CIFAR-10 benchmark, InTAct also achieves the highest AA of 31.28%, outperforming the strongest baseline, EWC. These results demonstrate that activation-space constraints can effectively mitigate catastrophic forgetting beyond the domain-incremental setting.

Table 7: AA in CIL settings. All results are averaged over 5 seeds.

Method	Split MNIST	Split FMNIST	Split CIFAR-10
EWC	20.57	19.92	30.43
LwF	28.63	25.92	24.26
MAS	17.91	18.74	21.75
SI	18.40	17.30	20.18
InTAct (Ours)	**35.27**	**34.25**	**31.28**

Hypercube Coverage and Hyperparameter α. The hyperparameter α controls the coverage of the cumulative hypercube $\mathcal{H}$ by determining how many extreme activations are excluded from each side of the coordinate intervals. Smaller values of α yield tighter bounds around the core task distribution but leave more outliers outside $\mathcal{H}$, introducing a trade-off between coverage and compactness. Importantly, Theorem 1 also bounds functional drift outside the hypercube, with the bound increasing proportionally to the distance from $\mathcal{H}$.

Table 8: Sensitivity analysis of the hyperparameter α for CODA-Prompt + InTAct on ImageNet-R.

α	0.75	0.80	0.85	0.90	0.95	0.99	1.00
AA (%)	74.66	74.51	74.49	74.63	74.23	73.12	71.37

As shown in Tab. 8, performance on ImageNet-R remains stable across $\alpha \in [0.75, 0.95]$, but declines as α approaches 1.00. Including extreme outliers in the hypercube can substantially expand its bounds into sparsely occupied feature regions. This, in turn, imposes overly restrictive constraints on the model, limiting the plasticity needed to adapt to new tasks.

Cumulative Hypercube Growth. To assess whether the protected region expands uncontrollably over long task sequences, we track the average coordinate diameter of the cumulative hypercube, $\frac{1}{d}\sum_{j=1}^{d}\mathrm{diam}_j(\mathcal{H})$, throughout the 15-task ImageNet-R benchmark using CODA-Prompt + InTAct.

Table 9: Growth of the cumulative hypercube's average coordinate diameter across ImageNet-R.

Task Interval	$\frac{1}{d}\sum \mathrm{diam}_j(\mathcal{H})$
Tasks 1–5	$3.280 \rightarrow 4.095$
Tasks 6–10	$4.096 \rightarrow 4.196$
Tasks 11–15	$4.197 \rightarrow 4.202$

Tab. 9 shows that most of the expansion occurs during the first five tasks, after which the hypercube rapidly approaches saturation. Its average coordinate diameter increases by only 0.107 over the remaining ten tasks. This indicates that the protected region stops growing as new tasks arrive, rather than continuing to expand into increasingly unoccupied feature space.

To determine whether this behavior persists over longer task sequences, we repeat the analysis on ImageNet-Mix, which comprises 30 tasks. As reported in Tab. 10, most of the expansion again occurs early: the average coordinate diameter increases by 0.653 over the first ten tasks, but by only 0.040 over the subsequent twenty tasks.

Table 10: Growth of the cumulative hypercube's average coordinate diameter across ImageNet-Mix (30 tasks), reported as mean $\pm$ standard deviation over runs.

Task Interval	$\frac{1}{d}\sum \mathrm{diam}_j(\mathcal{H})$
Tasks 1–10	$2.991 \pm 0.005 \rightarrow 3.644 \pm 0.037$
Tasks 11–20	$3.644 \pm 0.037 \rightarrow 3.662 \pm 0.035$
Tasks 21–30	$3.664 \pm 0.035 \rightarrow 3.684 \pm 0.035$

Occupancy and Out-of-Bounds Rate. To assess whether the cumulative hypercube becomes excessively loose in the high-dimensional activation space, we compare it with the task-specific hypercubes $\mathcal{H}_t$ defined in Sec. 3.1. We conduct this analysis on ImageNet-R using KA-Prompt with InTAct. We define occupancy as the ratio between the volume of the union of task-specific hypercubes and that of the cumulative hypercube $\mathcal{H}^{(\le T)}$, whose bounds are given by the coordinate-wise minima and maxima across tasks. Since volume ratios scale exponentially with the dimensionality d, we report the dimension-normalized occupancy:

$$R_{\mathrm{norm}} = \left(\frac{\mathrm{Vol}\big(\bigcup_{t=1}^{T}\mathcal{H}_t\big)}{\mathrm{Vol}\big(\mathcal{H}^{(\le T)}\big)}\right)^{1/d}, \qquad \mathrm{Vol}(\mathcal{H}) = \prod_{j=1}^{d}\mathrm{diam}_j(\mathcal{H}).$$

This normalization expresses occupancy as a side-length ratio, directly relating it to the coordinate-wise bounds used in Sec. 3.2. Since computing the exact union volume is costly, we replace $\mathrm{Vol}\big(\bigcup_{t=1}^{T}\mathcal{H}_t\big)$ with $\sum_{t=1}^{T}\mathrm{Vol}(\mathcal{H}_t)$, yielding an upper bound on R_{norm}. Indeed, $\mathrm{Vol}\big(\bigcup_t \mathcal{H}_t\big) \ge \frac{1}{T}\sum_t \mathrm{Vol}(\mathcal{H}_t)$, so the bound exceeds the exact value by at most a factor of $T^{1/d}$. For the pre-classifier activations ($d = 768$, $T = 15$), this factor is below 1.004. The resulting upper bound is 88.8%, placing the exact occupancy between 88.5% and 88.8%. Thus, the cumulative hypercube is only about 1.13 times wider per dimension than the region occupied by the task-specific hypercubes.

Complementing the geometric occupancy measure, the out-of-bounds rate quantifies how often activations from new tasks fall outside the cumulative bounds established by previous tasks. For a training batch $\{\mathbf{z}_n\}_{n=1}^{N_B}$ from task t, evaluated at the protected layer, we define

$$R_{\mathrm{OOB}} = \frac{1}{N_B\, d}\sum_{n=1}^{N_B}\sum_{j=1}^{d}\mathbb{I}\Big[z_{n,j} < \underline{z}_j^{(\le t-1)} \;\vee\; z_{n,j} > \overline{z}_j^{(\le t-1)}\Big],$$

where $\underline{z}_j^{(\le t-1)}$ and $\overline{z}_j^{(\le t-1)}$ denote the cumulative lower and upper bounds before observing task t. We compute this rate over all training batches and average it across tasks $t = 2, \ldots, T$. On

ImageNet-R, the resulting average is $R_{\mathrm{OOB}} = 5.8\%$, meaning that 94.2% of activation coordinates from new tasks fall within the existing bounds. Importantly, this is a coordinate-wise statistic: although most coordinates remain within the protected region, a typical sample still falls outside it along approximately 45 of its 768 coordinates. Thus, new tasks largely reuse existing per-coordinate ranges, limiting cumulative hypercube expansion, while individual representations can still extend beyond the protected region in a small number of dimensions.

Coverage of Previous-Task Activations. To assess how well the trained model preserves the coverage assumed in Theorem 1, we measure the proportion of previous-task activations contained within the cumulative hypercube on ImageNet-R using CODA-Prompt + InTAct. After training task t, we pass the evaluation data of all tasks $s \leq t$ through the current model and compute the percentage of activation coordinates z_j lying within the corresponding intervals $[\underline{z}_j, \overline{z}_j]$ of $\mathcal{H}^{(\leq t)}$. Equivalently, these are the coordinates for which $d_j(\mathbf{z}) = 0$ in Theorem 1.

Tab. 11 reports the resulting coverage. Immediately after learning a task, at least approximately 90% of its activation coordinates lie within the cumulative bounds, consistent with the quantile-based construction using $\alpha = 0.95$. Coverage of previously learned tasks generally increases as training progresses, despite temporary fluctuations. For task 1, it rises from 90.3% to 97.0%, while final coverage across tasks ranges from 96.1% to 97.0%. Task 14 is the only exception, with a slight decrease from 96.7% to 96.4%. Overall, no entry falls below 90%.

This improvement initially reflects the expansion of the cumulative hypercube. However, hypercube growth largely saturates after task 10: its average coordinate diameter increases by only 0.006 between tasks 10 and 15 (Tab. 9). Despite this limited expansion, task 1's coverage continues to rise from 95.5% to 97.0%, suggesting that its representations move further inside the protected region. The largest temporary decrease in coverage occurs after task 9, coinciding with the transition at which baseline performance drops sharply (Fig. 4).

Since coordinates contained within their intervals have $d_j(\mathbf{z}) = 0$, they contribute nothing to the distance-dependent term in Theorem 1. Consequently, for previous-task representations, this term depends only on the remaining 3–10% of coordinates lying outside the cumulative bounds.

Table 11: **Coverage of previous-task activations on ImageNet-R.** Entry (s, t) reports the percentage of activation coordinates of task s's evaluation samples that fall within the cumulative hypercube $\mathcal{H}^{(\leq t)}$, evaluated using the model after training task t. Shaded entries correspond to evaluations performed immediately after learning task s.

	after training task t														
task s	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	90.3	92.3	93.7	94.4	95.6	95.9	95.6	96.5	94.5	95.5	96.4	96.4	96.9	97.0	97.0
2		90.2	91.4	92.3	93.5	93.7	93.3	94.4	93.8	94.5	95.6	95.5	96.0	96.3	96.6
3			92.8	94.0	95.4	95.5	95.2	96.0	94.4	95.2	96.4	96.3	96.8	97.1	97.0
4				92.5	93.6	94.2	93.8	95.0	94.8	95.5	96.2	96.2	96.6	96.7	96.7
5					94.2	94.6	94.1	95.5	94.1	95.3	96.1	95.7	96.5	96.7	96.6
6						95.0	94.6	95.4	93.9	94.9	96.0	95.8	96.2	96.6	96.7
7							96.0	96.7	94.6	95.4	96.3	95.8	96.7	96.9	96.8
8								96.7	94.6	95.6	96.6	96.2	96.8	96.9	97.0
9									93.2	94.2	95.4	94.8	95.6	96.0	96.1
10										95.1	96.1	95.6	96.2	96.4	96.5
11											96.0	95.8	96.6	96.7	96.6
12												96.2	96.6	96.9	96.9
13													96.4	96.9	96.6
14														96.7	96.4
15															97.0

M Additional Visualization

Internal Representation Stability. We analyze the stability of internal representations using the Split-MNIST benchmark in the DIL scenario. For this experiment, we apply InTAct to a standard MLP. To align with our framework, we treat the first hidden layer as the fixed feature extractor

(applying $\mathcal{L}_{\text{Feat}}$ to its outputs) and apply the rest of regularization terms ($\mathcal{L}_{\text{IntDrift}}, \mathcal{L}_{\text{Var}}, \mathcal{L}_{\text{Align}}$) to the subsequent trainable layers.

Fig. 6 quantifies the *internal representation drift* experienced by historical data as the model adapts to new tasks. We track the normalized L_1 difference between the activations of reference samples at the end of their initial training task and their activations after subsequent tasks. While our baseline (LwF) exhibits progressive drift, indicating that the internal features of old data are being distorted by new plasticity, InTAct effectively bounds these changes, maintaining the geometric integrity of the latent space throughout the learning sequence.

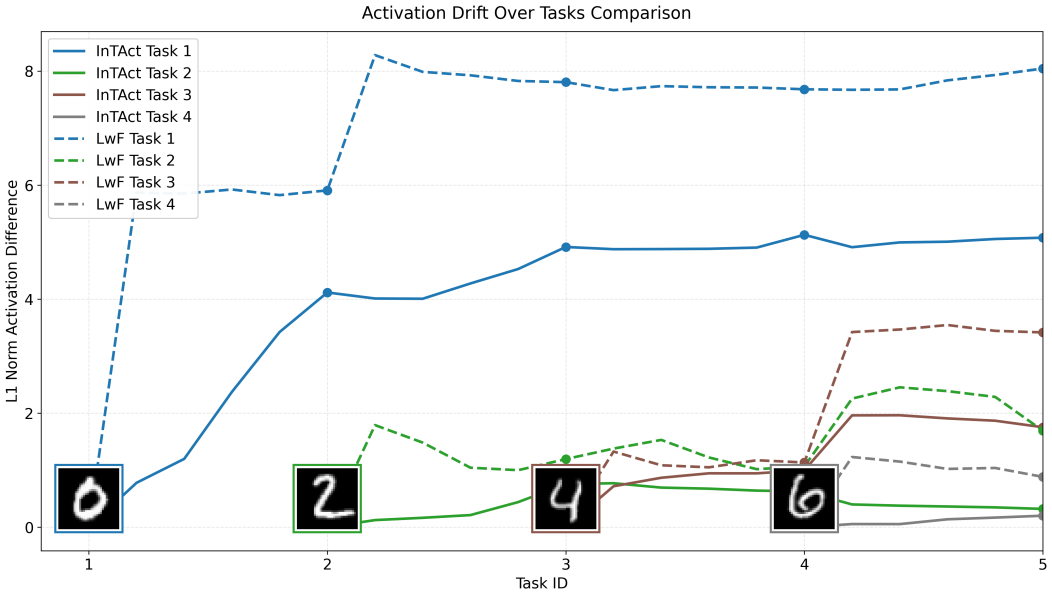


Figure 6: **Evolution of Internal Drift.** Activation stability on Split-MNIST (DIL). The plot tracks the cumulative L_1 shift in hidden activations for reference images from early tasks as the model trains on later tasks. InTAct successfully suppresses drift compared to LwF, validating the effectiveness of the proposed regularization.

Conclusions. The comparison highlights a fundamental difference in how the two methods balance stability and plasticity. Under LwF, representational drift accumulates over the task sequence, progressively altering the features of earlier tasks and eroding previously acquired knowledge. In contrast, InTAct effectively limits this drift. Following a brief initial adaptation phase, during which the shared decision boundaries adjust to new tasks, activation shifts largely plateau. This suggests that InTAct stabilizes the network's functional behavior on previously learned tasks, mitigating the representational changes that contribute to catastrophic forgetting.

N Parameters and Overhead Comparison

Table 12 compares computational efficiency on ImageNet-R, using reference measurements from [21] for the baselines. For KA-Prompt and InTAct + KA-Prompt, we measure GPU memory and batch time ourselves under identical conditions: a single NVIDIA H100 (80 GB) and batch size of 128. We report peak reserved GPU memory and mean training time per batch, including data loading, on the second task, when all InTAct terms are active.

InTAct introduces **no additional learnable parameters** beyond the underlying prompt-based method while substantially improving performance. Compared with KA-Prompt, it increases peak GPU memory by only 4% (from 18.87 GB to 19.64 GB) and training time per batch by 21% (from 0.314 s to 0.381 s). The additional time is mainly due to $\mathcal{L}_{\text{Feat}}$, which requires an extra forward pass through the frozen model. Inference remains unaffected, with nearly identical evaluation times per batch of 512 images (1.82 s for KA-Prompt and 1.81 s for InTAct). Thus, InTAct provides substantial accuracy gains at a modest training cost, without increasing inference overhead.

Table 12: **Efficiency comparison on ImageNet-R.** We report total parameters, trainable parameters, GPU memory usage, batch time, and AA. Baseline measurements are taken from [21]. ‡Measured by us under identical conditions on a single NVIDIA H100 (80 GB) with batch size 128. All other GPU memory and batch time measurements are from [21] and were obtained on different hardware; therefore, the two groups are not directly comparable.

Methods	Total Params	Trainable Params	GPU Memory (GB)	Batch time (s)	AA
L2P	86,263,843	311,387	19.33	0.66	56.55±0.33
S-Prompts	92,964,094	1,637,910	14.90	0.39	27.23±0.16
DualPrompt	86,514,211	561,755	19.45	0.66	59.47±1.00
CODA-Prompt	88,437,580	2,638,926	19.18	0.80	55.21±0.33
CPrompt	92,485,224	2,689,168	41.64	1.18	59.48±1.09
C-Prompt	89,180,505	3,381,851	11.99	0.42	62.43±0.49
KA-Prompt	89,180,505	3,381,851	18.87‡	0.31‡	66.51±0.36
InTAct (Ours)	89,180,505	3,381,851	19.64‡	0.38‡	**67.90±0.44**

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our approach is described in Section 3. All derivation details are provided in Appendix.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: This discussion is provided in Section 5.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be

used reliably to provide closed captions for online lectures because it fails to handle technical jargon.

- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Yes, we present one theorem in the main paper and provide its complete proof, together with additional supporting lemmas, in Appendix I.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide all hyperparameter configurations and practical implementation details in Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide the code as a .zip archive, along with a README file that includes instructions on how to run the individual experiments.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All relevant details are provided in Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report the number of independent runs for each experiment and present the mean and standard deviation of the results where applicable.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide detailed information about the computational resources used, including hardware specifications and runtime considerations, in Appendix C, sufficient to enable reproducibility.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We follow the code of ethics.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: By eliminating the need for replay buffers, InTAct reduces both the memory footprint and privacy risks associated with storing historical data, making it suitable for privacy-sensitive applications such as healthcare or on-device learning. More broadly, efficient continual learning methods may help reduce the computational cost of repeatedly retraining large-scale models. However, strong stability constraints may also preserve undesirable representations learned during earlier tasks, potentially making harmful biases more difficult to correct in later stages.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The proposed method does not introduce or release a general-purpose generative model or dataset that poses a high risk of misuse. Our approach focuses on limiting representation drift in prompt-based models within the DIL setting. Therefore, no specific safeguards are required.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We use standard publicly available datasets (e.g., DomainNet, ImageNet-R, and ImageNet-C) and build upon existing methods (e.g., CODA-Prompt), all of which are properly cited in the paper. We follow their standard usage protocols as defined in prior work. No new datasets are introduced, and we do not modify or redistribute existing assets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not introduce or release new datasets, models, or other assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing, human subjects, or any form of human data collection. All experiments are conducted on standard publicly available datasets.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve human subjects, crowdsourcing, or any form of human data collection. Therefore, IRB approval or equivalent review is not required.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: Large language models were not used as part of the proposed method, experiments, or analysis. Any use was limited to minor writing and editing assistance, which does not affect the scientific content of the work.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.