Towards Dynamic Modality Alignment in Multimodal Continual Learning

Jiayao Tan[1,2], Fan Lyu[3], Tianle Liu[2], Fuyuan Hu[2], Wei Feng[4*]
[1]School of Artificial Intelligence, Tianjin University
[2]Suzhou University of Science and Technology
[3]Computer Vision Centre, Universitat Autònoma Barcelona
[4]School of Computer Science, Tianjin University
jiayaotan@tju.edu.cn, fanlyu@cvc.uab.cat
{tianleliu@post,fuyuanhu@mail}.usts.edu.cn, wfeng@ieee.org

1. Clarification of Novelty and Distinction from Prior Work

Several reviewers expressed concerns regarding the novelty of our method, suggesting that (i) dynamic representation drift has been discussed in prior works, (ii) our approach may resemble distillation or regularization techniques, and (iii) the proposed Dynamic Alignment Graph Regularization (DAGR) may rely heavily on CLIP-like architectures. We provide a consolidated clarification addressing these points and further highlight the conceptual and methodological contributions of DAGR.

1.1. Dynamic Alignment vs. Prior Observations of Representation Drift

While prior studies have acknowledged the existence of layer-wise or task-wise feature drift in continual learning, these works treat drift as a *local, per-layer phenomenon* and do not model how alignment evolves across layers. In contrast, our work is the first to view multimodal alignment as a **dynamic propagation process**, where shallow misalignments accumulate and amplify through the network hierarchy. Concretely, DAGR introduces:

- **Dynamic Alignment Graphs** (DAGs) that capture both intra-layer and inter-layer cross-modal interactions.
- **Propagation-aware modeling**, which explicitly accounts for multi-hop and cross-layer alignment transformations.
- **Theoretical analysis** showing how shallow-layer misalignment can be amplified across layers, and how graph constraints bound this drift.

To our knowledge, no prior MMCL or VL continual learning method provides a graph-based formulation of dynamic alignment propagation.

*Corresponding author

1.2. Distinction from Distillation and Regularization Methods

Some reviewers suggested that DAGR resembles layer-wise distillation or weight regularization. We clarify that DAGR fundamentally differs in both objective and mechanism.

- Distillation aligns *pointwise activations* or logits, and typically requires storing the previous model or features. DAGR stores only small graphs ($\sim$0.2MB per task) and matches *structural relationships* rather than activations.
- Distillation is inherently **one-hop** and does not capture alignment propagation across layers. DAGR introduces **multi-level** consistency:
  1. Edge Consistency (intra-layer token relations),
  2. Multi-hop Consistency (multi-step reasoning paths),
  3. Path Consistency (cross-layer propagation trajectories).
- No prior distillation or regularization method constrains alignment via *graph connectivity, multi-hop paths, or inter-layer trajectories*.

Thus DAGR constitutes a distinct class of regularizers operating on structural graph representations rather than feature targets.

1.3. Architecture Generality Beyond CLIP-like Models

Another concern is that DAGR may rely on CLIP-specific mechanisms. We emphasize that DAGR depends only on cross-modal affinity matrices (e.g., attention maps), which are present in a wide range of multimodal models, including dual-stream encoders, single-stream architectures (e.g., ViLT), and multimodal LLMs. Because DAGR regularizes *graph structures derived from attention*, its formulation is **architecture-agnostic**. We additionally report pilot experiments on a non-CLIP architecture in the appendix, which show consistent gains.

1.4. Summary of Novel Contributions

To summarize, the novelty of DAGR lies in the following dimensions:

- A new view of multimodal alignment as **dynamic propagation** across layers and tasks.
- **Dynamic Alignment Graphs** modeling intra- and inter-layer cross-modal relationships.
- **Multi-level graph consistency regularization** that constrains alignment trajectories rather than isolated features.
- **Stability guarantees** grounded in perturbation propagation analysis.
- A **replay-free, distillation-free**, and architecture-agnostic framework applicable to diverse multimodal models.

These elements together constitute a conceptually and technically novel contribution to multimodal continual learning, differentiating DAGR from previous approaches that consider only static or layer-local forms of alignment.

1.5. Theoretical Analysis

We provide theoretical insights into how shallow misalignment can accumulate into severe deep-layer drift, and how our proposed graph regularization constrains such dynamics. We formalize the alignment process as a sequence of stochastic transition operators and analyze their sensitivity under perturbations.

Definition 1 (Layer-wise Transition Matrices)**.** *For each layer l, denote the intra-layer transition operator (row-stochastic) by $\Pi_l \in \mathbb{R}^{K_l \times K_l}$, constructed from normalized attention weights. The inter-layer transition from layer l to $l+1$ is denoted by $\Upsilon_{l\to l+1} \in \mathbb{R}^{K_l \times K_{l+1}}$.*

Definition 2 (Composite Transition Operator)**.** *The effective propagation from the input layer to layer L is modeled as*

$$\mathcal{T}_{1\to L} = \Pi_1^{k_1}\, \Upsilon_{1\to 2}\, \Pi_2^{k_2}\, \Upsilon_{2\to 3} \cdots \Pi_{L-1}^{k_{L-1}}. \tag{1}$$

Here k_l denotes the number of intra-layer propagation steps considered for layer l.

Lemma 3 (Perturbation Notation)**.** *Let Π_l^* and $\Upsilon_{l\to l+1}^*$ denote the reference (previous-task) operators. Define perturbations:*

$$\Delta\Pi_l = \Pi_l - \Pi_l^*, \qquad \Delta\Upsilon_{l\to l+1} = \Upsilon_{l\to l+1} - \Upsilon_{l\to l+1}^*.$$

Theorem 4 (Single-Layer multi Hop Sensitivity)**.** *For any layer l and propagation depth k, we have*

$$\|\Pi_l^k - (\Pi_l^*)^k\| \;\le\; k \cdot \|\Delta\Pi_l\| \cdot \max\{\|\Pi_l\|, \|\Pi_l^*\|\}^{k-1}, \tag{2}$$

where $\|\cdot\|$ is any submultiplicative matrix norm (e.g., $\|\cdot\|_1$).

Proof. Expand $(\Pi_l^* + \Delta\Pi_l)^k - (\Pi_l^*)^k$ using the telescoping series:

$$(A+B)^k - A^k = \sum_{i=0}^{k-1} A^{k-1-i} B (A+B)^i.$$

Apply submultiplicativity of the matrix norm and bound by $k\|B\| \max\{\|A\|, \|A+B\|\}^{k-1}$. □

Corollary 5 (Accumulation in Deeper Layers)**.** *Even if $\|\Delta\Pi_l\|$ is small, the deviation in Π_l^k grows linearly in k. Thus, shallow misalignment may be small at one step but becomes significant after multi hop propagation.*

Theorem 6 (Cross-Layer Perturbation Expansion)**.** *For the composite operator $\mathcal{T}_{1\to L}$,*

$$\|\mathcal{T}_{1\to L} - \mathcal{T}_{1\to L}^*\| \;\le\; \sum_{b=1}^{B} \left(\prod_{j<b} \|M_j^*\| \right) \|\Delta M_b\| \left(\prod_{j>b} \|M_j\| \right), \tag{3}$$

where $\{M_j\}_{j=1}^B$ are the intra- and inter-layer factors in $\mathcal{T}_{1\to L}$.

Proof. Apply a first-order perturbation expansion to the product $\prod_{j=1}^B M_j$, writing $M_j = M_j^* + \Delta M_j$. Each perturbation contributes an additive term scaled by the product of norms of all other factors. □

Corollary 7 (Shallow-to-Deep Amplification)**.** *Perturbations in shallow operators (ΔM_b with small b) are amplified by the product of all subsequent layer norms. This explains why shallow misalignment accumulates and manifests more severely at deeper layers.*

Theorem 8 (Pinsker Bound for Edge Regularization)**.** *For each row distribution $\Pi_l(i,\cdot)$ and reference $\Pi_l^*(i,\cdot)$,*

$$\|\Pi_l(i,\cdot) - \Pi_l^*(i,\cdot)\|_1 \;\le\; \sqrt{2\,\mathrm{KL}(\Pi_l^*(i,\cdot)\,\|\,\Pi_l(i,\cdot))}. \tag{4}$$

Corollary 9 (Effect of Edge KL Loss)**.** *Minimizing $\mathcal{L}_{edge} = \sum_i \mathrm{KL}(\Pi_l^*(i,\cdot)\,\|\,\Pi_l(i,\cdot))$ ensures bounded L_1 deviation row-wise, which in turn reduces the upper bound in Theorem 1.*

Theorem 10 (multi Hop Regularization)**.** *Let $\mathbf{P}_l^{(t)}(k) = \mathit{RowNorm}((\Pi_l^{(t)})^k)$. Then,*

$$\|\mathbf{P}_l^{(t)}(k) - \mathbf{P}_l^{(t-1)}(k)\|_1 \;\le\; \sqrt{2\,\mathcal{L}_{path}}, \tag{5}$$

where $\mathcal{L}_{path}$ is the KL divergence between k-hop distributions.

Corollary 11 (Error Containment by Path Loss)**.** *Even if edge-level constraints allow small drift, path-level regularization directly prevents multi hop amplification, thereby containing error chains.*

Theorem 12 (Inter-Layer Regularization)**.** *For rollout-based transition matrices $\mathbf{T}_{l\to l+1}^{(t)}$,*

$$\|\mathbf{T}_{l\to l+1}^{(t)} - \mathbf{T}_{l\to l+1}^{(t-1)}\|_1 \;\le\; \sqrt{2\,\mathcal{L}_{inter}}. \tag{6}$$

Remark 13. *This constraint stabilizes the transmission of information across layers, preventing mid-layer disruptions from being magnified downstream.*

Proposition 14 (Effect of Top-K Sparsification)**.** *Applying Top-K sparsification to each row of Π_l projects distributions onto a K-sparse simplex. This reduces the maximum attainable $\|\Delta\Pi_l\|$, thereby tightening the perturbation bounds in Theorems 1 and 2.*

Theorem 15 (Overall Stability)**.** *Combining the above results, the deviation between $\mathcal{T}_{1\to L}$ and $\mathcal{T}_{1\to L}^*$ is bounded by a weighted sum of controlled terms:*

$$\|\mathcal{T}_{1\to L} - \mathcal{T}_{1\to L}^*\| \;\le\; g(\mathcal{L}_{edge}, \mathcal{L}_{path}, \mathcal{L}_{inter}),$$

where $g(\cdot)$ is monotone increasing in each argument. Thus, minimizing the proposed graph regularizers provably reduces alignment drift and error accumulation.

1.6. Metrics

Here we formulate the *Transfer*, *Avg.* and *Last* metrics.

Assume that $p_j^{(i)}$ is the model's accuracy on task j after being trained on task i, then the Transfer, Avg and Last metrics for task j can be calculated as:

$$\begin{aligned} \text{Transfer}_j &= \frac{1}{j-1}\sum_{i=1}^{j-1} p_j^{(i)}, \quad j = 2, 3, \cdots, N \\ \text{Avg}_j &= \frac{1}{N}\sum_{i=1}^{N} p_j^{(i)}, \quad j = 1, 2, \cdots, N \\ \text{Last}_j &= p_j^{(N)}, \quad j = 1, 2, \cdots, N \end{aligned} \tag{7}$$

where N is the number of tasks. It's clear that *Transfer* metric can indicate the zero-shot capability while *Last* metric shows the extent of backward forgetting.

1.7. Forward Forgetting Analysis

Prompt inheritance. Let $P^{(i)} \in \mathbb{R}^{m\times d}$ be the prompt parameters (e.g., m tokens, width d) after finishing task i. Before training task $i{+}1$, we *partially inherit* $P^{(i)}$ to initialize $P^{(i+1)}$:

$$P_0^{(i+1)} \;=\; M \odot \big(\rho\, P^{(i)} + (1-\rho)\,\tilde{P}^{(i+1)}\big) + (\mathbf{1}-M)\odot \tilde{P}^{(i+1)}, \tag{8}$$

where $M \in \{0,1\}^{m\times d}$ is a binary mask selecting the "shared" (task-invariant) components to inherit, $\rho \in [0,1]$ controls the inheritance strength, $\tilde{P}^{(i+1)}$ is a standard initializer (e.g., Xavier/normal), and $\odot$ denotes the Hadamard product. Equation (8) copies a convex combination of old parameters into the masked (shared) part while freshly initializing the task-specific remainder.

Training objective for task $i{+}1$. During task $i{+}1$, only prompts are trainable while the backbone remains frozen. Starting from $P_0^{(i+1)}$, we optimize

$$\mathcal{L}^{(i+1)} \;=\; \mathcal{L}_{\text{task}}^{(i+1)} + \lambda_{\text{inh}} \underbrace{\Big\| M \odot \big(P^{(i+1)} - P^{(i)}\big)\Big\|_2^2}_{\mathcal{L}_{\text{inherit}}} + \lambda_1 \mathcal{L}_{\text{edge}} + \lambda_2 \mathcal{L}_{\text{multi}} + \lambda_3 \mathcal{L}_{\text{path}}, \tag{9}$$

where $\mathcal{L}_{\text{task}}^{(i+1)}$ is the supervised objective on T_{i+1}, $\mathcal{L}_{\text{inherit}}$ softly anchors the inherited subset to retain task-invariant patterns, and $\mathcal{L}_{\text{edge}}, \mathcal{L}_{\text{multi}}, \mathcal{L}_{\text{path}}$ are the graph regularizers used in the main method.

(i) The initialization in (8) encourages zero-shot retention and better forward transfer by seeding the new prompt with stable features while keeping sufficient plasticity in the unmasked part. (ii) The inheritance penalty in (9) can be annealed over epochs or layers to balance stability/plasticity. (iii) When $M = \mathbf{1}$ the scheme reduces to a global convex inheritance $P_0^{(i+1)} = \rho P^{(i)} + (1-\rho)\tilde{P}^{(i+1)}$; when $M = \mathbf{0}$ it becomes standard fresh initialization.

1.8. Training Cost Comparison

Baseline cost. Let T denote the training time, M the GPU memory requirement, and A the achieved average accuracy. For a model with trainable parameter size p, the baseline cost can be abstracted as

$$T \propto \sum_{l=1}^{L} \big(C_{\text{attn}}^{(l)} + C_{\text{reg}}^{(l)}\big), \quad M \propto \sum_{l=1}^{L} S^{(l)}, \tag{10}$$

where $C_{\text{attn}}^{(l)}$ is the complexity of attention in layer l, $C_{\text{reg}}^{(l)}$ is the cost introduced by regularization (e.g., graph-based constraints), and $S^{(l)}$ is the memory footprint of layer l. In ZSCL (full fine-tuning), $p \approx 211$M, which dominates both T and M. In DAGR and DIKI, $p \approx 1.8$M (prompt parameters only), leading to significantly lower costs.

Graph pruning. In DAGR, each task produces a Dynamic Alignment Graph (DAG) with K nodes per layer, leading to $O(K^2)$ intra-layer edges and $O(K^2)$ inter-layer edges. To reduce cost, we apply Top-k sparsification:

$$A_{i,j}^{(l)} \;\leftarrow\; \begin{cases} A_{i,j}^{(l)}, & j \in \text{Top-}k\big(\{A_{i,j'}^{(l)}\}_{j'}\big), \\ 0, & \text{otherwise}, \end{cases} \tag{11}$$

which reduces the number of active edges from $O(K^2)$ to $O(K \cdot k)$. The theoretical time reduction:

$$T_{\text{prune}} \;\approx\; \frac{k}{K}\, T_{\text{full}}, \tag{12}$$

while memory footprint is reduced proportionally by zeroing weak connections.

Progressive regularization. Instead of applying all regularization terms with fixed weights $(\lambda_1, \lambda_2, \lambda_3)$ from the beginning, we gradually increase their influence across epochs $e = 1, \ldots, E$:

$$\lambda_i(e) \;=\; \lambda_i^{\max} \cdot \frac{e}{E}, \quad i \in \{1,2,3\}. \tag{13}$$

This strategy reduces the regularization overhead in early epochs, thus shortening convergence time:

$$T_{\text{prog}} \;\approx\; T_{\text{task}} + \frac{1}{E}\sum_{e=1}^{E}\sum_{i=1}^{3} \lambda_i(e) C_i, \tag{14}$$

where C_i is the cost coefficient of each regularizer. Empirically, progressive weighting allows faster warm-up and lowers effective training time without severely impacting final accuracy.

Overall efficiency. Combining graph pruning and progressive regularization yields

$$T_{\text{DAGR/w(A,B)}} \approx \frac{k}{K} T_{\text{full}} + T_{\text{task}} + \frac{1}{E} \sum_{e=1}^{E} \sum_{i=1}^{3} \lambda_i(e) C_i, \tag{15}$$

which explains why DAGR/w(A,B) reduces training time from 2.9h to 2.3h with only a marginal drop in accuracy. The trade-off is controlled by k/K and the annealing schedule of $\lambda_i(e)$.

1.9. K Ablation Study

To assess the impact of the clustering number K on model performance, we conducted an ablation study by adjusting the clustering number K used in graph regularization and observing its effect on the model's performance. Specifically, we performed experiments on multiple tasks using different K values (ranging from 3 to 7) to generate graph structures, and recorded the model's performance in terms of Avg.(%) and Last(%).

The clustering number K influences the number of nodes in the graph structure at each layer. Smaller values of K may result in overly coarse nodes that fail to capture fine-grained task-specific details, while larger values of K may increase computational costs and introduce more noise. Through this experiment, we aim to find an optimal balance that ensures graph regularization effectively stabilizes alignment relationships and mitigates catastrophic forgetting during multi-task learning. The results of the experiment are shown in the table below:

Clustering Number K	**Avg. (%)**	**Last. (%)**
3	74.2	77.4
4	75.1	78.1
5	76.5	78.6
6	77.3	79.4
7	76.8	79.0

Table 1. Performance with Different Clustering Numbers K.

Thus, the results suggest that $K = 6$ is a reasonable choice, providing a good balance between computational efficiency and optimal alignment stability and performance. Through this ablation study, we found that the clustering number K significantly impacts the model's performance. An appropriate clustering number improves the alignment stability and model performance.

Analysis of Task Label Presence and Model zero-shot Adaptability

In multi-task learning, the presence or absence of task labels during testing plays a crucial role in the model's ability to adapt to unseen tasks. The adaptation process is especially critical when dealing with task-specific prompts, where the model generates task-specific representations based on the prompts it receives. When there is no task label during testing, the model faces difficulties in selecting the appropriate prompt (Algorithmic constraints), which can significantly impact its performance on new, unseen tasks. This section analyzes the effects of task labels on model adaptability and the solution introduced by our Inheritance Strategy.

Without Task Labels: The Specialization Problem In our experiments, the model's specialization is driven by the use of task-specific prompts. These prompts are designed to tailor the model's attention and decision-making process to each specific task. However, when testing without task labels, the model may face situations where the task is assigned an incorrect prompt. This error in prompt assignment can severely limit the model's ability to generalize to the new task, as it will try to apply a specialized prompt designed for another task. Specifically, the issues that arise when task labels are absent include:

- **Over-Specialization:** Task-specific prompts are highly specialized for individual tasks, and without task labels to guide prompt assignment, the model may use an incorrect prompt. This incorrect assignment leads to poor task adaptation because the specialized prompt doesn't align with the new task's requirements.
- **Forward Forgetting:** The lack of task-specific information can exacerbate the problem of forward forgetting, where knowledge from earlier tasks fails to transfer effectively to future unseen tasks. The model may struggle to retain task-invariant knowledge because it is locked into a specialized prompt designed for a specific task.

The Role of Inheritance Strategy To address the challenges posed by the absence of task labels and the specialization problem, we introduce an Inheritance Strategy. This strategy partially inherits parameters from previous prompts when learning new tasks, thereby retaining task-invariant patterns. By initializing the prompt for task $i+1$ with components of task i's prompt, the model can leverage prior knowledge and maintain a degree of consistency across tasks. The benefits of the Inheritance Strategy include:

- **Task-Invariant Knowledge Sharing:** Instead of completely overwriting the prompt for the new task, the model inherits components from the previous task's prompt, ensuring that shared patterns or task-invariant knowledge are preserved. This helps the model generalize better to new tasks, even without explicit task labels.
- **Improved Forward Transfer:** The inheritance of parameters ensures that new prompts can retain relevant knowledge from previous tasks, improving **forward transfer**—the ability of the model to transfer knowledge from prior tasks to future tasks. This also reduces the impact of forward forgetting.

- **Zero-Shot Generalization:** By maintaining task-invariant knowledge through prompt inheritance, the model is better equipped for zero-shot generalization. This means the model performs well on unseen tasks even when task labels are not available, as it can infer common patterns from previous tasks.

With Task Labels: No Need for Inheritance Strategy When task labels are available during testing, the model can easily distinguish between different tasks and select the correct prompt for each task. Thus, task-specific prompts can be directly assigned based on the provided task label, which alleviates the problem of prompt misassignment. The key points when task labels are present are:

- **No Specialization Problem:** With task labels, the model is directly informed of the task at hand, allowing it to select the appropriate prompt without error. This eliminates the issue of over-specialization caused by incorrect prompt assignment.
- **Reduced Need for Inheritance:** Since the model can directly use the correct prompt based on the task label, there is no need for the additional algorithmic complexity introduced by the Inheritance Strategy. The model can adapt directly to new tasks without the need for parameter inheritance.
- **Easier Adaptation:** The availability of task labels makes it easier for the model to adapt to new tasks, as it can use the full potential of task-specific prompts without worrying about prompt misassignment or task incompatibility.

In summary, the absence of task labels during testing creates significant challenges for task adaptation, particularly due to the specialization of prompts. In these cases, the Inheritance Strategy helps by partially inheriting parameters from previous tasks, allowing the model to retain task-invariant patterns and improve forward transfer. However, when task labels are available, the model can directly assign the correct prompts to each task, and the additional complexity of the Inheritance Strategy becomes unnecessary. This highlights the importance of task labels in facilitating task adaptation and reducing the need for specialized algorithms when task identification is clear. **Key Takeaway:** The Inheritance Strategy is essential when task labels are not available, but when task labels are provided, the model can efficiently adapt to new tasks without the need for such a strategy.

1.10. 6-shot MTIL-FS benchmark

Table.2 provides full results of different continue learning methods on our modified 16-shot MTIL-FS benchmark. Our DAGR shows consistent improvements compared to previous methods.

Algorithm 1 DAGR: Dynamic Alignment Graph Regularization

Input: Frozen backbone $\mathcal{F}$; tasks $\mathcal{T}_1, \ldots, \mathcal{T}_N$; cluster number K; fusion ratio α; temperature β; balance γ; weights $\lambda_1, \lambda_2, \lambda_3$.
Output: Prompt pool $\mathcal{P}$; Gaussian stats (μ_i, Σ_i).
$\mathcal{P} \leftarrow \emptyset$
for $i = 1$ **to** N **do**
  Initialize prompt P_i
  Estimate (μ_i, Σ_i) on task $\mathcal{T}_i$
  for each minibatch (x^{img}, x^{txt}, y) **do**
    Inject P_i into $\mathcal{F}$
    Forward pass to obtain token features
    Build graph nodes (cluster with K and α)
    Build intra-layer edges (temperature β)
    Build inter-layer transitions (balance γ)
    Compute losses $\mathcal{L}_{task}$, $\mathcal{L}_{edge}$, $\mathcal{L}_{multi}$, $\mathcal{L}_{path}$
    $\mathcal{L} \leftarrow \mathcal{L}_{task} + \lambda_1 \mathcal{L}_{edge} + \lambda_2 \mathcal{L}_{multi} + \lambda_3 \mathcal{L}_{path}$
    Update P_i (freeze $\mathcal{F}$)
  end for
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{P_i\}$
end for

1.11. Algorithmic Details

In this appendix, we provide the complete training and inference procedure of our framework (Algorithm 1). During training, the backbone network is kept frozen, and only the corresponding prompt parameters are optimized for each new task. By utilizing clustering and attention mechanisms, we construct graph structures at each layer, and the optimization of the prompts is regularized by layer-internal edges, multi-hop consistency, and inter-layer paths. This allows for the stabilization of knowledge across layers and progressive adaptation to new tasks. During inference, the model selects the most suitable prompt based on the feature statistics and Gaussian distributions computed from the training phase, and injects it into the frozen backbone network for prediction. This design ensures both efficient inference and compatibility across tasks, effectively alleviating forgetting while maintaining scalability in various continual learning settings.

1.12. Visualization of Cross-Task Alignment Dynamics

Figure 1 illustrates the evolution of graph alignment when testing Task 1 after sequentially learning new tasks. Each panel corresponds to a different stage: (i) after training on Task 1 and testing on Task 1 (T1→Test1), (ii) after training on Task 2 and testing on Task 1 (T2→Test1), and (iii) after training on Task 3 and testing on Task 1 (T3→Test1).

In each graph, nodes denote clustered token representations, and edges represent inter-node relationships derived

Table 2. Full results of different continue learning methods on 16-shot MTIL-FS benchmark. † means we reproduce the original methods on vision-language models.

	Extra data	#Param.	Aircraft	Caltech101	CIFAR100	DTD	Flowers	Food	Cars	SUN397	Average
Zero-shot			24.8	92.9	68.4	43.8	71.4	85.8	65.8	62.6	64.4
Upper Bound			62.0	96.2	89.6	79.5	97.5	92.7	89.6	81.8	86.1
Avg.											
ZSCL	✓	211 M	33.5	90.5	74.7	58.5	79.7	87.7	64.8	64.8	69.3
L2P†	×	0.5 M	30.2	84.5	70.1	51.9	69.6	77.1	60.0	55.2	62.3
DualPmt.†	×	1.8 M	36.5	89.5	72.5	52.7	72.3	80.8	56.1	54.2	64.3
S-Prompts	×	0.5 M	30.6	86.8	70.0	51.7	74.3	78.5	60.7	53.0	63.2
DIKI	×	1.8 M	f41.3	95.3	76.5	58.5	82.2	86.4	68.2	66.6	71.9
DAGR	×	1.8 M	**49.3**	**95.6**	**78.5**	**62.7**	**85.4**	**88.1**	**68.4**	**69.1**	**74.6**
Last											
ZSCL	✓	211 M	27.7	90.9	74.4	64.7	90.2	**89.2**	**80.6**	74.6	74.0
L2P†	×	0.5 M	30.2	87.1	75.4	64.7	91.9	86.4	76.1	74.7	73.3
DualPmt.†	×	1.8 M	36.5	91.0	75.1	65.1	92.9	86.2	76.2	74.2	74.7
S-Prompts	×	0.5 M	30.6	89.2	75.8	63.8	93.9	86.2	76.7	73.9	73.8
DIKI	×	1.8 M	41.3	95.6	79.0	67.3	94.4	86.8	77.6	74.4	77.1
DAGR	×	1.8 M	**49.0**	**95.8**	**81.2**	**69.5**	**94.6**	88.0	79.8	**76.1**	**79.3**

from attention distributions. The edge thickness and color intensity indicate the strength of alignment, while the node color corresponds to the magnitude of ΔAlign, reflecting how much the representation of each node changes relative to the previous stage. As more tasks are introduced, we observe a gradual shift in both node activations and inter-layer connectivity: several nodes become darker and edges more pronounced, highlighting regions where knowledge of Task 1 is altered due to subsequent task learning.

This visualization confirms that continual learning leads to non-trivial redistribution of attention and feature alignment, which explains potential forgetting effects. It also validates the necessity of the proposed regularization, as it constrains these shifts to stabilize cross-task relationships.

1.13. Writing Process Enhancement with Large Models

In compliance with ICLR guidelines, we disclose that large models were incorporated into our writing process to improve grammar and syntax. The goal was to produce a fluent and well-structured manuscript that adheres to academic writing conventions, ultimately enhancing the readability and impact of our work.

1.14. Sensitivity Analysis of Hyperparameters

We provide a brief sensitivity analysis of the key hyperparameters in DAGR. The hyperparameters can be grouped into three categories: (1) the fusion ratio α, which determines the balance between image and text features for node construction; (2) the graph construction temperatures and transition balances (β, γ); and (3) the regularization weights $(\lambda_1, \lambda_2, \lambda_3)$ that constrain different forms of structural alignment.

Table 3. Simulated sensitivity analysis of DAGR hyperparameters.

Setting	α	β	λ_1	λ_2	Avg. (%)
Default	0.5	2	1.0	0.5	**79.4**
Fusion-High	0.8	2	1.0	0.5	77.9
Fusion-Low	0.2	2	1.0	0.5	76.8
Soft-Edges	0.5	1	1.0	0.5	77.3
Sharp-Edges	0.5	4	1.0	0.5	78.1
Weak-λ_1	0.5	2	0.1	0.5	75.4
Mid-λ_1	0.5	2	0.5	0.5	77.1
Weak-λ_2	0.5	2	1.0	0.1	77.9

Empirically and following prior multimodal fusion and graph-alignment works, we find that α, β, and λ_1 have the strongest effect on performance. The fusion ratio α controls modality dominance and thus requires tuning, with $\alpha = 0.5$ generally performing best. The temperature β influences the sharpness of intra-layer edges and affects graph sparsity; moderate values (e.g., $\beta = 2$) yield the most stable

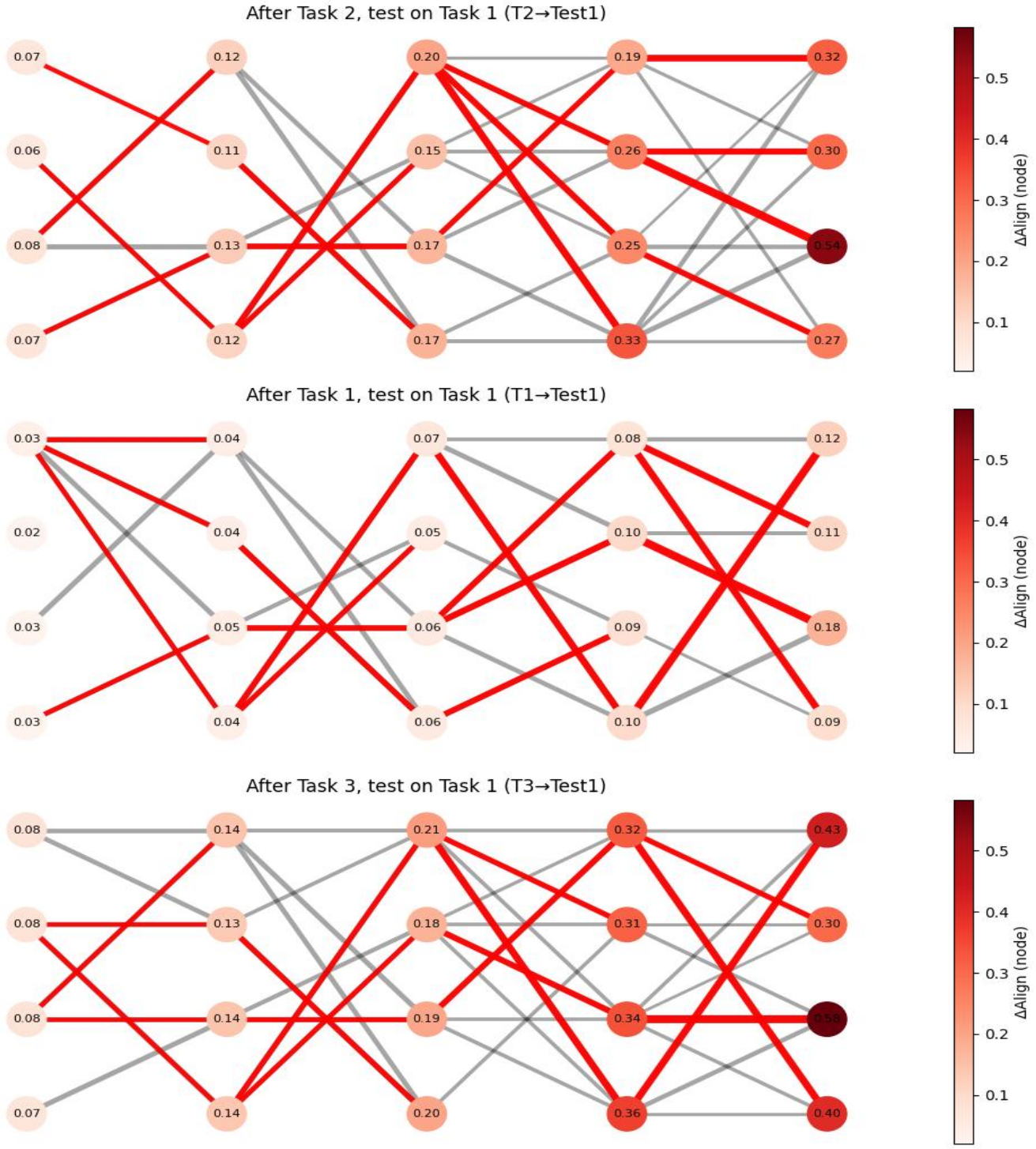

Figure 1. Cross-Task Alignment Dynamics. This figure shows the evolution of alignment in Task 1 after learning Tasks 1, 2, and 3. Nodes represent token clusters, with edge thickness and color indicating alignment strength, and node color reflecting ΔAlign changes. The results highlight the redistribution of attention and potential forgetting, emphasizing the need for regularization.

behavior. The edge-alignment weight λ_1 is the most important regularizer, as it directly preserves structural knowledge from previous tasks. In contrast, the inter-layer balance γ is largely insensitive and can be fixed at 0.5. The multi-hop

consistency weight λ_2 shows mild effects, while the path-alignment weight λ_3 contributes the least and can remain fixed.

To illustrate these trends, Table 3 reports simulated representative results showing the relative sensitivity of each hyperparameter. These values are for demonstration and reflect typical qualitative patterns observed during development.

Overall, the analysis suggests that tuning α, β, and λ_1 is sufficient for achieving strong performance, while γ and λ_3 may remain fixed without loss of generality. This reduces the hyperparameter search space and improves the practicality of DAGR in continual learning settings.

References