

# DynaTokens: Controlling Token Dynamics for Continual Video–Language Understanding

Toan Nguyen<sup>1</sup> Yang Liu<sup>1</sup> Celso De Melo<sup>2</sup> Flora D. Salim<sup>1</sup>

<sup>1</sup> School of Computer Science and Engineering, University of New South Wales, Australia

<sup>2</sup> DEVCOM Army Research Laboratory, USA

{toan.nguyen, flora.salim}@unsw.edu.au

yang.liu39@student.unsw.edu.au, celso.miguel.de.melo@gmail.com

Correspondence: flora.salim@unsw.edu.au

## Abstract

Continual VideoQA with multimodal LLMs remains challenging because sequential adaptation induces task interference, while storing task-specific prompts becomes impractical as task sequences grow. We introduce **DYNATOKENS**, a transformer-based token generator that dynamically produces fine-tuning tokens on demand, enabling task-adaptive prompt updates through shared generation weights. To mitigate forgetting, we introduce meta-learning-inspired regularisers that look ahead to avoid task-specific sharp update directions while anchoring the evolving generator to prior-task behaviours. We theoretically connect this objective to sharpness-aware optimisation, showing how it favours flatter cross-task minima and improves retention. **DYNATOKENS** combines gradient-free routing based on robust pretrained token and visual embeddings with lightweight auxiliary multimodal supervision, reducing router drift during continual adaptation. Across standard continual VideoQA benchmarks, **DYNATOKENS** achieves higher average accuracy and substantially lower forgetting than strong baselines. It also improves zero-shot generalisation and remains effective in longer domain-incremental sequences with extended task shifts. Finally, we introduce a challenging ImageQA→VideoQA protocol and show that **DYNATOKENS** enables robust cross-modal continual transfer. Our code is available at [this link](#).

## 1 Introduction

Modern multimodal LLMs have demonstrated remarkable capacity for jointly reasoning over video, images, audio, and language, achieving strong performance on tasks such as VideoQA (Shu et al., 2025; Maaz et al., 2024; Qian et al., 2024). However, the prevailing *train-then-deploy* paradigm (Liu et al., 2024; Team et al., 2024; Zhao et al., 2024; Liu et al., 2025b) rests on a stationarity assumption that breaks down in the face of

evolving task distributions, domain shifts, and previously unseen concepts. Naïvely fine-tuning on incoming tasks leads to *catastrophic forgetting* (McCloskey and Cohen, 1989; Cai et al., 2024), and full-parameter updates quickly become prohibitive at scale. Parameter-efficient adaptation (PEA) (Cai et al., 2024; Adhikari et al., 2025; Razdaibiedina et al., 2023) provides an appealing alternative by introducing a small number of learnable parameters atop a frozen backbone, yet applying it to the continual setting remains an open challenge.

The core difficulty is the tension between *plasticity* and *stability*: sharply shifting task distributions cause sequential prompt updates to interfere and erode prior knowledge (Tan et al., 2025). Allocating separate adapter prompts per task (Razdaibiedina et al., 2023; Zeng et al., 2025) preserves task-specific knowledge but scales poorly, while sharing prompt parameters across tasks (Tan et al., 2025; Cai et al., 2024; Wang et al., 2022b) improves efficiency yet exposes shared components to conflicting gradients (Nguyen et al., 2026). Alternatives such as LoRA and prefix tuning (Adhikari et al., 2025; Yu et al., 2025) also fall short of simultaneously (i) scaling gracefully to many tasks, (ii) preserving fine-grained, task-specific control without cross-task interference, and (iii) generalising reliably to multimodal temporal settings such as VideoQA.

To address these challenges, we draw inspiration from hypernetworks (Ha et al., 2017) and propose **DYNATOKENS**, a transformer-based universal token generator that synthesises adapter prompts *on demand* from compact task codes. Since the generator weights are shared across tasks, its size remains fixed regardless of the task sequence length, so *memory grows only minimally* with the number of tasks. Figure 1 illustrates the overall design.

A central challenge in our hypernetwork-based prompt control is twofold: obtaining informative task codes for routing, and updating the gener-

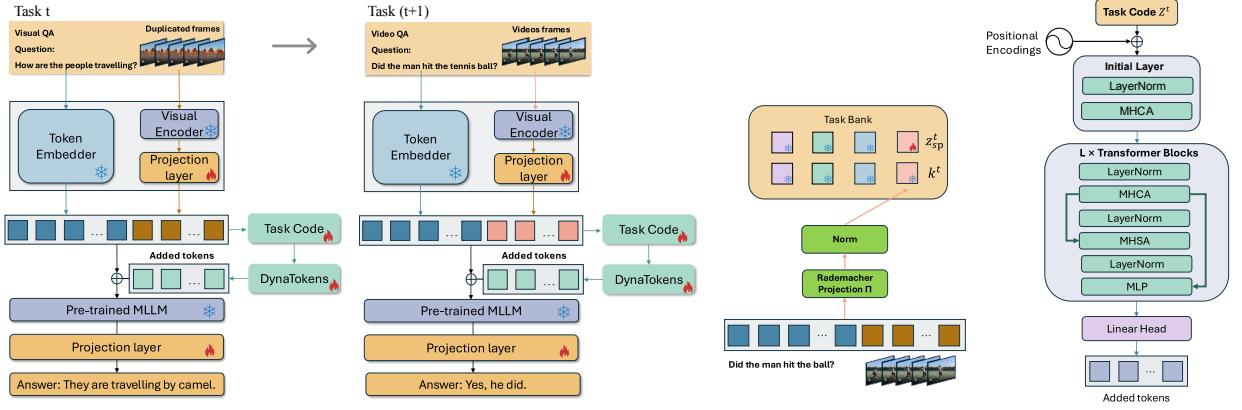


Figure 1: **DYNATOKENS overview.** (Left) Continual adaptation with DYNATOKENS for VideoQA and cross-modal transfer VisualQA→VideoQA. A *fixed-size* generator synthesises *task-specific* fine-tuning tokens. (Middle) Task bank as key-value memory: multimodal embeddings are projected via  $\Pi$  and normalised to form retrieval keys  $k^t$ , indexing learnable task codes  $z_{sp}^t$  for the token generator. (Right) Transformer-based token-generator architecture.

ator without forgetting. For routing, we adopt a gradient-free key-value memory (Geva et al., 2021) with no learnable parameters. The keys are token-level embeddings from the frozen pretrained backbone, already discriminative enough for reliable retrieval without further training. The values are lightweight task codes, each composed of task-specific and task-invariant learnable modules. Since the routing mechanism introduces no additional trainable parameters, it is *immune to drift by construction*: retention of task knowledge is handled by the generator and its regularisation, not by the router. At inference, retrieved task codes are fed into the shared generator to produce task-adapted prompts on the fly.

To preserve the generator under continual updates, we train it with a meta-learning-inspired objective that first *looks ahead* to avoid sharp, task-specific overfitting directions, and then *looks back* to regularise the update by matching an anchored behaviour on a small set of past task codes. We theoretically connect this objective to sharpness-aware minimisation (SAM) (Foret et al., 2021), showing that it favours flatter cross-task minima and improves robustness under continual adaptation. Beyond regularisation, the shared generation weights enable DYNATOKENS to leverage lightweight auxiliary multimodal objectives effectively, encouraging richer cross-modal token learning.

Empirically, DYNATOKENS achieves strong accuracy with substantially reduced forgetting on standard continual VideoQA benchmarks, while also improving zero-shot generalisation and remaining effective on continual dataset with ex-

tended domain shift sequences. To further stress-test continual adaptation, we introduce a challenging ImageQA→VideoQA protocol in which the model must shift from static image understanding to temporal video reasoning, inducing both a *modality mismatch* and a *domain mismatch* between single-frame recognition and temporal reasoning. The strongest recent PEA baseline degrades sharply under this regime, whereas DYNATOKENS remains robust and outperforms it across both stages, highlighting the effectiveness of token-hypernetwork generation for multimodal continual learning.

## 2 Related Work

**Video Question Answering (VideoQA)** requires answering questions grounded in video inputs, demanding joint reasoning over visual and linguistic context. Existing approaches either build specialised video vision-language models (Deitke et al., 2025; Alayrac et al., 2022; Yu et al., 2023) or adapt large pretrained LLMs to better capture temporal visual context (Yang et al., 2022; Zhang et al., 2024a,b). Yet they implicitly assume stationary data and are not designed for the streaming setting in which VideoQA datasets arrive sequentially with substantial distribution shifts.

**Continual Learning for VideoQA** considers a streaming setting where a model is trained sequentially on multiple VideoQA datasets, with catastrophic forgetting as the central challenge. Replay-based methods rehearse past samples (Buzzege et al., 2020; Nguyen et al., 2024) but face prohibitive storage costs for raw videos.

Regularisation- and architecture-based approaches have been explored mainly for multimodal classification (Ke et al., 2020; Jha et al., 2024; Liu et al., 2025a) and do not transfer cleanly to next-token LLM-style VideoQA, motivating recent shifts toward parameter-efficient adaptation atop frozen backbones.

**Parameter-Efficient Adaptation (PEA)** (Zhang et al., 2024b; Li and Liang, 2021) enables continual adaptation by training lightweight task-specific modules atop a frozen backbone. We highlight three differentiating properties of DYNATOKENS against representative baselines:

(i) **Bank size.** Prompt-pool methods (Wang et al., 2022b,a; Smith et al., 2023) and per-task prompt-storage methods (Razdaibiedina et al., 2023; Zeng et al., 2025) maintain prompt collections that grow linearly with the task stream. DYNATOKENS instead synthesises prompts on demand from compact task codes via a *fixed-size* generator, decoupling memory cost from the number of tasks.

(ii) **Routing dependency.** ModalPrompt (Zeng et al., 2025) requires auxiliary CLIP encoders for prompt selection, adding external modules and inference cost. DYNATOKENS routes directly through the frozen backbone’s own token embeddings, requiring no external encoders.

(iii) **Look-ahead regularisation.** The closest baseline, AdaPrefix++ (Adhikari et al., 2025), also generates fine-tuning parameters with a hypernetwork (Ha et al., 2017) from stored per-task inputs, but regularises against past behaviour without look-ahead and offers no theoretical analysis of its regulariser. DYNATOKENS adds a look-ahead regulariser whose gains grow with more look-ahead steps and which admits the sharpness-based characterisation in Sec. 4.2; Appendix D details the remaining differences in setting, routing, and architecture. Crucially, existing PEA methods are evaluated primarily in unimodal or static image-text settings, leaving continual VideoQA with its temporal reasoning demands largely unexplored. We provide an extended discussion of prompt-based methods and hypernetwork-based continual learning in Appendix D.

### 3 Background

**Problem Setup.** We study continual VideoQA over a sequence of tasks  $t = 1, \dots, T$ . Task  $t$  provides a dataset  $\mathcal{D}_t = \{(V_i^t, Q_i^t, A_i^t)\}_{i=1}^{N_t}$ , where  $V_i^t \in \mathbb{R}^{L_v \times C}$ ,  $Q_i^t \in \mathbb{R}^{L_q \times C}$ , and  $A_i^t \in \mathbb{R}^{L_a \times C}$  are

the embedded video, question, and answer token sequences, respectively. The goal is to adapt a pre-trained multimodal LLM  $f_\theta$  to each incoming task while preserving performance on all previous ones. Denoting the causal context  $\mathcal{C}_{i,j}^t = (V_i^t, Q_i^t, A_{i,\leq j}^t)$ , the per-task training objective is the standard next-token negative log-likelihood:

$$\mathcal{L}_{\text{NLL}}^t = -\mathbb{E}_{i \sim \mathcal{D}_t} \sum_{j=1}^{L_a} \log p_\theta(a_{i,j}^t | \mathcal{C}_{i,j-1}^t). \quad (1)$$

**Forgetting.** Sequentially minimising (1) across tasks induces *catastrophic forgetting*: updates fitted to  $\mathcal{D}_t$  may overwrite knowledge required for earlier tasks, especially in the replay-free regime where past data and gradients are unavailable. Parameter-efficient adaptation (PEA) partially mitigates this by freezing the backbone and training only lightweight modules (e.g., adapters, prompts, or LoRA weights), though these modules must still be stored and correctly routed at inference, which becomes increasingly difficult as the task stream grows.

## 4 Method

### 4.1 DynaTokens

We introduce **DYNATOKENS**, a hypernetwork-based method that synthesises task-specific prompt tokens *on demand* from compact task codes. Each task  $t$  is represented by two lightweight objects: a generative code  $z^t$  for prompt synthesis, and a retrieval key  $k^t$  for identifying the task at inference. Decoupling *what to generate* from *how to route* enables scalable continual prompt control without storing per-task prompts.

#### Prompt Generation via a Token Hypernetwork.

We split the generative code into a shared, task-invariant component and a task-specific one,  $z^t = [z_{\text{inv}}, z_{\text{sp}}^t] \in \mathbb{R}^{N_z \times C_z}$  with  $N_z = N_{\text{inv}} + N_{\text{sp}}$ . To produce layer-specific prompts, the hypernetwork  $H_\phi$  is additionally conditioned on a learnable layer embedding  $e_\ell \in \mathbb{R}^d$  for each prompted layer  $\ell \in \{1, \dots, L_p\}$ , where  $d$  is its internal width:

$$\begin{aligned} P_\ell^t &= H_\phi(z^t, e_\ell) \in \mathbb{R}^{N_p \times C}, \\ P^t &= [P_1^t; \dots; P_{L_p}^t] \in \mathbb{R}^{L_p \times N_p \times C}, \end{aligned} \quad (2)$$

where  $N_p$  is the number of prompt tokens per layer and  $C$  is the LLM hidden dimension. Layer conditioning lets a single generator specialise prompts to the role of each transformer layer (low-level

features vs. higher-level semantics) while keeping per-task storage small. For example, on LLaMA2-7B ( $C=4096$ ) with  $N_{\text{sp}}=25$ ,  $C_z=256$ ,  $L_p=32$ ,  $N_p=10$ , each task stores only its code  $z_{\text{sp}}^t$  and a  $D_k=256$ -dimensional key,  $N_{\text{sp}}C_z + D_k \approx 6.7 \times 10^3$  values ( $\approx 26$  KB), whereas the materialised prompt tensor has  $L_p N_p C \approx 1.3 \times 10^6$  entries, a roughly  $200\times$  reduction. Each new task therefore adds only kilobytes of storage rather than full prompt blocks.

We instantiate  $H_\phi$  as a transformer because prompt synthesis is naturally a structured *set-to-set* mapping: a compact task code must be expanded into many coordinated prompts across layers. The code is projected to the generator width,  $X^t = z^t W_z + E_z \in \mathbb{R}^{N_z \times d}$ , and the layer identity  $e_\ell$  is injected by feature-wise modulation (FiLM) of the internal features. A lightweight pooling head derives  $N_p$  initial query tokens  $U^{(0)}$  from  $X^t$ , and  $L_h$  pre-norm blocks of cross-attention (MHCA), self-attention (MHSA), and MLP then let the queries repeatedly read from the task code and interact with each other to specialise:

$$\begin{aligned}\bar{U}^{(\ell)} &= U^{(\ell)} + \text{MHCA}(\text{LN}(U^{(\ell)}), X^t), \\ \tilde{U}^{(\ell)} &= \bar{U}^{(\ell)} + \text{MHSA}(\text{LN}(\bar{U}^{(\ell)})), \\ U^{(\ell+1)} &= \tilde{U}^{(\ell)} + \text{MLP}(\text{LN}(\tilde{U}^{(\ell)})).\end{aligned}\quad (3)$$

A shared head  $W_o \in \mathbb{R}^{d \times C}$  maps the output states to the LLM width,  $P_\ell^t = U^{(L_h)} W_o$ ; the  $L_p$  layers are generated in one batched call and stacked into  $P^t$ . Full architectural details are given in Appendix C.1.

**Decoupled Retrieval Keys.** Codes optimised for generation need not be good retrieval prototypes, so we maintain a separate key  $k^t \in \mathbb{R}^{D_k}$  per task, grounded directly in the frozen input embeddings  $(V_i^t, Q_i^t)$ . For each sample we form

$$r_i^t = \text{Norm}(\Pi \text{Pool}([V_i^t; Q_i^t])), \quad (4)$$

where  $\Pi \in \mathbb{R}^{D_k \times C}$  is a fixed Rademacher random projection that approximately preserves distances under the Johnson–Lindenstrauss lemma (Johnson et al., 1984). The task key is its normalised mean,

$$k^t = \text{Norm}\left(\frac{1}{N_t} \sum_{i=1}^{N_t} r_i^t\right). \quad (5)$$

Since  $\Pi$  is frozen and the underlying embeddings come from frozen modules, all task keys live in a stable, drift-free space. Routing thus adds *no trainable parameters* and requires no rehearsal or anti-forgetting on the router itself, sidestepping a

major failure mode of learned routers in continual learning. In practice,  $k^t$  is accumulated with a moving average over minibatches, and pooling  $Q_i^t$  alone already yields discriminative keys.

**Task-Code Initialisation.** We set  $N_{\text{inv}} = \frac{1}{5} N_z$ , reserving a small portion of the code budget for shared structure and the rest for task-specific capacity.  $z_{\text{inv}}$  is initialised once with  $\mathcal{N}(0, \sigma^2 I)$ ,  $\sigma = 0.02$ , and updated across tasks. The first task’s code is sampled the same way; for  $t > 1$ , we warm-start  $z_{\text{sp}}^t$  from previously learned codes weighted by key similarity:

$$z_{\text{sp}}^t = \sum_{\tau < t} w_\tau^t z_{\text{sp}}^\tau + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2 I), \quad (6)$$

$$w_\tau^t = \frac{\exp(\cos(k^t, k^\tau)/\rho_z)}{\sum_{j < t} \exp(\cos(k^t, k^j)/\rho_z)}. \quad (7)$$

This is a soft retrieval in the key space defined above: semantically related past tasks contribute more, providing positive transfer rather than optimising from scratch. With a small temperature ( $\rho_z = 0.1$ ), the weighting concentrates on the most similar past tasks, approximating soft nearest-neighbour retrieval (Oord et al., 2018).

**Training Objective and LA-Reg.** The base objective is the autoregressive negative log-likelihood conditioned on the generated prompts:

$$\mathcal{L}_{\text{NLL}}^t = -\mathbb{E}_i \sum_{j=0}^{N_{i,a}^t - 1} \log p(a_{i,j+1}^t | \mathcal{C}_{i,j}^t), \quad (8)$$

where  $\mathcal{C}_{i,j}^t = (V_i^t, Q_i^t, A_{i,\leq j}^t, P^t)$ . Since  $H_\phi$  is shared across tasks, naïvely updating it on  $\mathcal{D}_t$  can drift the prompts produced for previous task codes. We prevent this with a *look-ahead regularisation* (**LA-Reg**) that penalises such drift on stored codes. Let  $\phi^*$  be the generator snapshot before task  $t$  and  $\Delta\phi$  the update from  $M$  inner-loop steps on  $\mathcal{L}_{\text{NLL}}^t$ :

$$\mathcal{L}_{\text{LA-Reg}}^t = \sum_{\tau < t} \|H_{\phi^*}(z^\tau) - H_{\phi + \Delta\phi}(z^\tau)\|_2^2. \quad (9)$$

LA-Reg requires only the stored *codes*, not raw data: previous prompt behaviour is reconstructed on the fly through  $H_{\phi^*}$ . We additionally anchor the shared code with a quadratic penalty  $\|z_{\text{inv}} - z_{\text{inv}}^*\|_2^2$ , where  $z_{\text{inv}}^*$  is its pre-task snapshot; in practice this term remains small throughout training and acts as a mild, non-restrictive prior on the shared capacity.

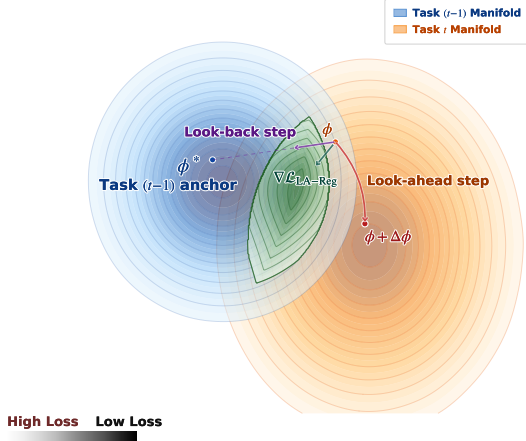


Figure 2: **Geometry of LA-Reg.** LA-Reg balances progress along the task- $t$  gradient with alignment to the task- $(t-1)$  anchor, steering optimisation into the shared low-loss basin (green) and yielding flatter cross-task minima. Regularisation acts in prompt-output space, not parameter space.

**Auxiliary Multimodal Supervision.** Following recent VideoQA work (Tan et al., 2025; Cai et al., 2024), we augment  $\mathcal{L}_{\text{NLL}}^t$  with two cross-modal auxiliary objectives,  $\mathcal{L}_{\text{VAQ}}^t$  and  $\mathcal{L}_{\text{QAV}}^t$ : the next-token NLL of  $Q$  given  $(V, A, P)$  and of  $V$  given  $(Q, A, P)$ , respectively. A *causal view* (Pearl, 2009; Nguyen et al., 2023) explains their relative weighting. Under the VideoQA graph  $V \rightarrow Q$ ,  $V \rightarrow A$ ,  $Q \rightarrow A$ , the conditional  $p(Q | V, A)$  follows a feasible causal direction and yields an identifiable signal;  $p(V | Q, A)$ , in contrast, is *anti-causal* and underdetermined, as many videos can map to the same  $(Q, A)$ . Yet the anti-causal loss is not discarded: even as a poor generative target, it still pushes the prompts and the QA-side representation to stay *visually grounded*. We therefore keep  $\mathcal{L}_{\text{QAV}}^t$  in the same next-token form but give it a small weight, an asymmetry that ablations in Sec. E confirm: removing it weakens token learning, while up-weighting it hurts performance.

**Full Objective.** The task- $t$  objective combines the base, regularisation, and auxiliary terms:

$$\begin{aligned} \mathcal{L}_{\text{Final}}^t = & \mathcal{L}_{\text{NLL}}^t + \lambda_{\text{LA-Reg}} \mathcal{L}_{\text{LA-Reg}}^t \\ & + \lambda_{\text{inv}} \|z_{\text{inv}} - z_{\text{inv}}^*\|_2^2 \\ & + \lambda_{\text{VAQ}} \mathcal{L}_{\text{VAQ}}^t + \lambda_{\text{QAV}} \mathcal{L}_{\text{QAV}}^t. \end{aligned} \quad (10)$$

Unless stated otherwise, we fix  $\lambda_{\text{LA-Reg}} = 0.5$ ,  $\lambda_{\text{inv}} = 0.1$ ,  $\lambda_{\text{VAQ}} = 1.0$ , and  $\lambda_{\text{QAV}} = 0.1$  across all benchmarks, following the causal weighting above. The small  $\lambda_{\text{inv}}$  reflects that its underlying penalty stays naturally small across tasks, yet ablations (Sec. 5) show it still contributes meaningfully to retention. This single default leaves

DYNATOKENS tuning-free at deployment. Since  $\mathcal{L}_{\text{LA-Reg}}^t$  sums over the  $t-1$  stored codes, its magnitude grows along the stream under a fixed weight; averaging over stored codes, equivalently scaling  $\lambda_{\text{LA-Reg}} \propto \frac{1}{t-1}$ , keeps the per-task strength constant. Our fixed setting nevertheless remains stable over the 13-task stream.

**Task Bank and Unknown-Task Inference.** After task  $t$ , we cache only  $(k^t, z_{\text{sp}}^t)$  in a compact bank  $\mathcal{B}_t = \mathcal{B}_{t-1} \cup \{(k^t, z_{\text{sp}}^t)\}$ . At test time, given  $(V_{\text{test}}, Q_{\text{test}})$ , we compute  $r_{\text{test}}$  via Eq. (4) and retrieve the nearest task,

$$t^* = \arg \max_{\tau \leq t} \cos(r_{\text{test}}, k^\tau), \quad (11)$$

reconstruct  $z^{t^*} = [z_{\text{inv}}, z_{\text{sp}}^{t^*}]$ , and synthesise  $P_{\text{test}} = H_\phi(z^{t^*})$  on the fly. Inference requires *no task identifier*, extending DYNATOKENS to the more challenging *task-agnostic* continual learning setting where task identity is hidden at test time. Routing accuracy, when nearest-key retrieval errs, and its bounded downstream impact are analysed in Appendix E.5.

## 4.2 Theoretical Analysis

We give two complementary views on why DYNATOKENS resists forgetting: (i) LA-Reg acts as a task-wise sharpness penalty, and (ii) the shared generator confines adaptation to a low-dimensional prompt manifold. Full proofs are in Appendix B.

**LA-Reg as a Sharpness Penalty.** Let  $J_\phi(z) := \partial H_\phi(z)/\partial \phi$  and  $g(\phi) := \nabla_\phi \mathcal{L}_{\text{NLL}}^t(\phi)$ , so the look-ahead update in Eq. 9 is  $\Delta \phi = -\eta g(\phi)$ .

**Theorem 1** (Task-wise sharpness penalty). *Assume  $H_\phi$  is twice differentiable with bounded Jacobian and remainder,  $\|H_{\phi^*}(z^\tau) - H_\phi(z^\tau)\|_2 \leq \varepsilon$ , and  $\|g(\phi)\|_2 \leq G$ . Then for small  $\eta$ , there exists  $C > 0$  such that*

$$\begin{aligned} & |\mathcal{L}_{\text{LA-Reg}}^t(\phi) - \eta^2 \sum_{\tau < t} \|J_\phi(z^\tau) g(\phi)\|_2^2| \\ & \leq C(\varepsilon^2 + \varepsilon \eta + \eta^3). \end{aligned} \quad (12)$$

The leading term  $\eta^2 \sum_{\tau < t} \|J_\phi(z^\tau) g(\phi)\|_2^2$  flags “sharp” directions where fitting task  $t$  erodes past behaviour. LA-Reg penalises them (Fig. 2), encouraging flatter cross-task basins. Theorem 1 is a *first-order characterisation* of what LA-Reg penalises, a cross-task sharpness term on the generator parameters  $\phi$ , rather than an anti-forgetting bound. The connection to sharpness-aware minimisation (Foret

et al., 2021) is correspondingly scoped: both penalise sensitivity to a gradient-aligned parameter perturbation, but SAM perturbs all model parameters with respect to the training loss, whereas LA-Reg perturbs only  $\phi$  and measures the induced drift of past-task prompt outputs.

**Low-Rank Adaptation on a Shared Manifold.** Local updates to a generated prompt  $H_\phi(z^t)$  lie in the range of the Jacobian  $J_z H_\phi(z^t)$ , whose dimension is bounded by  $N_z C_z$ , the task-code budget. Since  $N_z C_z \ll L_p N_p C$ , DYNATOKENS implicitly restricts adaptation to a low-dimensional shared manifold inside the much larger prompt space, with task codes providing task-specific control and the shared generator biasing prompts toward reusable cross-task structure. A formal statement and proof are deferred to Appendix B.2.

## 5 Experiments

### 5.1 Continual VideoQA

#### 5.1.1 Setup

**Datasets and Settings.** We evaluate on two continual VideoQA benchmarks. Following (Tan et al., 2025), NExT-QA (Xiao et al., 2021) is split into eight tasks by question type (TP, CW, DC, TC, DL, DO, TN, CH), spanning causal, descriptive, and temporal categories; following (Cai et al., 2024), DramaQA (Choi et al., 2021) is split in the order What  $\rightarrow$  Who  $\rightarrow$  Where  $\rightarrow$  How  $\rightarrow$  Why. We further introduce a *continual-dataset* setting that chains the two benchmarks as a longer *domain-incremental* stream, and report *zero-shot performance* on unseen datasets to probe robustness under test-time distribution shift.

**Baselines.** We compare against representative PEA methods for LLM-based continual VideoQA: Bisele (Tan et al., 2025), DAM (Cheng et al., 2025), ColPro (Cai et al., 2024), ProgPrompt (Razdaibiedina et al., 2023), LAE (Gao et al., 2023), DualPrompt (Wang et al., 2022a), and L2P (Wang et al., 2022b), plus a naïve LLaMA-Adapter (Zhang et al., 2024b) reference.

**Metrics.** We report two standard metrics: average accuracy **Acc** (mean top-1 across tasks after the sequence) and average forgetting **Fog** (mean drop from each task’s peak to its final accuracy).

**Implementation.** All methods share a frozen backbone: LLaMA-2-7B (Touvron et al., 2023) as the LLM and CLIP ViT-L/14 (Radford et al.,

**Table 1:** *Continual VideoQA* results on **NExT-QA** and **DramaQA** with per-dataset accuracy and forgetting. **Bold** and underline denote the best and second-best results.

| Method (Venue)           | NExT-QA            |                      | DramaQA            |                      |
|--------------------------|--------------------|----------------------|--------------------|----------------------|
|                          | Acc ( $\uparrow$ ) | Fog ( $\downarrow$ ) | Acc ( $\uparrow$ ) | Fog ( $\downarrow$ ) |
| LLaMA-Adapter (ICLR’24)  | 46.58              | 13.83                | 60.99              | 24.39                |
| L2P (CVPR’22)            | 48.82              | 12.25                | 62.50              | 20.67                |
| DualPrompt (ECCV’22)     | 50.62              | 11.74                | 65.89              | 17.93                |
| LAE (ICCV’23)            | 49.38              | 11.47                | 65.82              | 17.35                |
| ProgPrompt (ICLR’23)     | 53.95              | 10.69                | 67.92              | 14.95                |
| ColPro (EMNLP’24)        | 55.14              | 7.43                 | 71.24              | 12.64                |
| DAM (WACV’25)            | 53.88              | 9.99                 | 67.37              | 15.19                |
| Bisele (NeurIPS’25)      | <u>62.37</u>       | <u>5.34</u>          | <u>71.49</u>       | <u>10.37</u>         |
| <b>DYNATOKENS (ours)</b> | <b>64.11</b>       | <b>4.76</b>          | <b>72.52</b>       | <b>9.76</b>          |

2021; Dosovitskiy et al., 2021) as the visual encoder. Unless stated otherwise, we use LLaMA-Adapter (Zhang et al., 2024b) with 32 adapter layers. Following (Tan et al., 2025; Cai et al., 2024), each video is sampled to 10 frames at 224×224; text is padded/truncated to 128 tokens on NExT-QA and 280 on DramaQA. We train five epochs on NExT-QA and six on DramaQA, with two look-ahead steps for LA-Reg. Full details are in Appendix C.

#### 5.1.2 Results

**Performance Comparison.** Table 1 shows that DYNATOKENS sets a new state of the art on both benchmarks, achieving the highest **Acc** and lowest **Fog** across the board. On NExT-QA, DYNATOKENS lifts **Acc** by +1.74 over the previous best (Bisele) while cutting **Fog** by 0.58, a non-trivial gain when both metrics are averaged over eight sequential tasks. The advantage carries over to DramaQA, a domain with markedly different question distributions, where DYNATOKENS surpasses Bisele by +1.03 **Acc** and −0.61 **Fog** and outperforms strong prompt-based baselines such as ProgPrompt and ColPro by even larger margins. These consistent gains underscore the effectiveness of controlled prompt synthesis via the shared token hypernetwork.

**Acc and Fog Changes.** Fig. 3 tracks **Acc** and **Fog** after each task on NExT-QA. DYNATOKENS consistently leads Bisele throughout the sequence, with the gap widening as tasks accumulate. The largest single-task gain reaches +5.07 **Acc** (DC) and −2.75 **Fog** (TN), confirming that DYNATOKENS not only learns new tasks better but also forgets less along the way.

**Table 2:** *Continual-dataset* results on the joint NExT-QA  $\rightarrow$  DramaQA stream (13 tasks total). We report **Acc** and **Fog** averaged across all tasks. **Bold** and underline denote the best and second-best results.

| Method (Venue)       | Acc ( $\uparrow$ ) | Fog ( $\downarrow$ ) |
|----------------------|--------------------|----------------------|
| ProgPrompt (ICLR'23) | 48.43              | <u>11.88</u>         |
| ColPro (EMNLP'24)    | 54.18              | 12.49                |
| Bisecle (NeurIPS'25) | <u>55.31</u>       | 13.30                |
| DYNATOKENS (ours)    | <b>61.96</b>       | <b>11.72</b>         |

**Table 3:** *Zero-shot inference* results across NExT-QA and DramaQA: train on one benchmark, evaluate on the other. We report top-1 accuracy. **Bold** and underline denote the best and second-best results.

| Method (Venue)       | OOD on NExT-QA | OOD on DramaQA |
|----------------------|----------------|----------------|
| ProgPrompt (ICLR'23) | <u>30.55</u>   | 39.24          |
| ColPro (EMNLP'24)    | 27.73          | <u>51.47</u>   |
| Bisecle (NeurIPS'25) | 27.86          | 49.07          |
| DYNATOKENS (ours)    | <b>36.29</b>   | <b>52.83</b>   |

**Continual-dataset Setting.** We further evaluate a more demanding setting that chains NExT-QA and DramaQA into a single 13-task stream, introducing both a *longer sequence* and a substantial *domain shift* (real-world  $\rightarrow$  movie videos). Table 2 reports **Acc** and **Fog** averaged over all 13 tasks: DYNATOKENS outperforms all baselines by a clear margin, confirming its robustness under extended task streams and cross-domain transitions.

**Zero-shot Inference** We probe robustness on *unseen* datasets at test time, which also tests our routing mechanism under out-of-distribution inputs. Each method’s checkpoint trained on one benchmark is evaluated zero-shot on the other (NExT-QA  $\leftrightarrow$  DramaQA). From Table 3, DYNATOKENS generalises most robustly against baselines.

### 5.1.3 Analysis and Ablation Studies

**Loss contribution.** Table 4 ablates the three training terms of DYNATOKENS on NExT-QA. The largest gain comes from  $\mathcal{L}_{LA}$ : adding it alone (row 3) lifts **Acc** from 21.28 to 62.12 and cuts **Fog** by over  $3\times$ , confirming look-ahead regularisation as the principal driver of anti-forgetting. The auxiliary objective  $\mathcal{L}_{Aux}$  adds a complementary  $+1.67$  **Acc** and  $-0.58$  **Fog**, pushing the model into the state-of-the-art regime by enriching cross-modal grounding. The invariance term  $\mathcal{L}_{Inv}$  contributes a smaller but consistent improvement that stabilises the shared code. The full objective is best on both metrics, in-

**Table 4:** *Contribution of loss terms* on NExT-QA.  $\mathcal{L}_{Inv}$  denotes invariance regularisation and  $\mathcal{L}_{Aux}$  denotes the weighted sum of auxiliary objectives  $\mathcal{L}_{VAQ}$  and  $\mathcal{L}_{QAV}$ .

| $\mathcal{L}_{LA-Reg}$ | $\mathcal{L}_{Inv}$ | $\mathcal{L}_{Aux}$ | Acc ( $\uparrow$ ) | Fog ( $\downarrow$ ) |
|------------------------|---------------------|---------------------|--------------------|----------------------|
| $\times$               | $\times$            | $\times$            | 21.28              | 16.46                |
| $\times$               | $\times$            | $\checkmark$        | 54.54              | 8.83                 |
| $\checkmark$           | $\times$            | $\times$            | 62.12              | 5.46                 |
| $\checkmark$           | $\checkmark$        | $\times$            | 62.41              | 5.19                 |
| $\checkmark$           | $\times$            | $\checkmark$        | <u>63.79</u>       | <u>4.88</u>          |
| $\checkmark$           | $\checkmark$        | $\checkmark$        | <b>64.11</b>       | <b>4.76</b>          |

|                    | Bisecle         | ProgPrompt     | DYNATOKENS      |
|--------------------|-----------------|----------------|-----------------|
| Storage / task     | $\approx 16$ KB | $\approx 5$ MB | $\approx 26$ KB |
| Latency (ms/query) | 47.7            | 48.4           | 49.3            |

**Table 5: Deployment cost.** Per-task storage and inference latency (batch = 1). Training cost is analysed in Appendix E.7.

dicating the three components are complementary rather than redundant.

**Effect of look-ahead steps.** Table 6 varies look-ahead steps in Eq. 9 from 0 to 2. Both **Acc** and **Fog** improve monotonically with more steps ( $54.54 \rightarrow 62.30 \rightarrow 63.92 \rightarrow 64.11$  **Acc**), confirming that look-ahead updates strengthen anti-forgetting by anticipating current-task drift on past behaviour.

**Compute, memory, and inference cost.** Table 5 reports the two costs that recur at deployment. Per-task storage is a compact code of  $\sim 26$  KB and stays *constant* along the stream, whereas ProgPrompt materialises a prompt per task and its memory grows with the sequence ( $21.16 \rightarrow 25.43$  GB on NExT-QA; Table 16). Inference latency is essentially identical across methods (within  $\sim 3\%$ ), since prompt synthesis is a single forward pass through the small generator. Training cost is higher for DYNATOKENS, owing to the look-ahead inner loop, but this overhead is confined to training and does not recur at deployment (Appendix E.7); moreover, the 1-step variant halves training time ( $5.55 \rightarrow 2.46$  h), trims peak memory by  $\sim 20\%$ , and still surpasses Bisecle ( $63.92$  vs  $62.37$  **Acc**,  $5.02$  vs  $5.34$  **Fog**), making it a practical default.

**Token Analysis.** Figure 4 (middle/right) visualises task-specific prompt tokens at the silhouette-optimal mid layers ( $i = 12$  on NExT-QA,  $i = 9$  on DramaQA). Across both datasets, the layout suggests a *skill-oriented organisation* that does not strictly follow the dataset-defined taxonomies: tasks demanding similar reasoning mechanisms

tend to cluster together (e.g., DL/TC for spatial-temporal grounding on NExT-QA; DO/DL for visual grounding on DramaQA), while semantically distinct tasks (CW, TP) appear as relative outliers. The clusters remain reasonably compact and well separated, consistent with DYNATOKENS retaining discriminative task representations across the continual training trajectory and providing qualitative evidence of mitigated catastrophic forgetting. A more detailed cluster-level analysis is provided in Appendix E.1.

## 5.2 ImageQA → VideoQA

### 5.2.1 Setup

**Dataset.** We use Visual7W (Zhu et al., 2016) (28K images, 140K QA pairs) as the ImageQA source and pair it with NExT-QA to form an ImageQA → VideoQA continual sequence: we first pretrain offline on Visual7W, then continue under the original 8-task NExT-QA protocol, probing how image-level pretraining affects downstream video reasoning under continual adaptation.

**Baselines & Metrics.** LLM-based PEA baselines have not been evaluated in this setting to our knowledge. We compare DYNATOKENS against the next-best baseline Bisecle, treating each image as a single-frame video by duplicating its visual tokens to match the video token length, which yields a unified continual fine-tuning protocol across image and video tasks.

**Implementation.** We retain the backbone and hyperparameters from the NExT-QA setup. On Visual7W, we fine-tune for 10 epochs with  $\mathcal{L}_{\text{NLL}}$  and  $\mathcal{L}_{\text{VAQ}}$  only, disabling video-specific auxiliary losses since static images require no temporal reasoning. We use an 80/20 train-validation split.

### 5.2.2 Results

Table 7 shows that DYNATOKENS outperforms Bisecle on Visual7W → NExT-QA in both **Acc** and **Fog**, achieving 43.44 vs 41.90 on Visual7W and 62.31 vs 58.24 on NExT-QA, with consistently lower forgetting. More importantly, continual ImageQA → VideoQA is markedly harder than VideoQA alone and tends to induce *slight negative transfer*: Bisecle’s NExT-QA accuracy collapses from 62.37 (VideoQA-only) to 58.24 (−4.13), whereas DYNATOKENS degrades only mildly (64.11 → 62.31, −1.80), retaining over half of the gap as a clear advantage.

**Table 7:** *Continual ImageQA → VideoQA* results on **Visual7W** → **NExT-QA**, with accuracy and forgetting. **Bold** and underline denote the best and second-best results.

| Method (Venue)           | Visual7W     |              | NExT-QA      |             |
|--------------------------|--------------|--------------|--------------|-------------|
|                          | Acc (↑)      | Fog (↓)      | Acc (↑)      | Fog (↓)     |
| Bisecle (NeurIPS’25)     | 41.90        | 28.89        | 58.24        | 5.93        |
| <b>DYNATOKENS (ours)</b> | <b>43.44</b> | <b>27.92</b> | <b>62.31</b> | <b>5.02</b> |

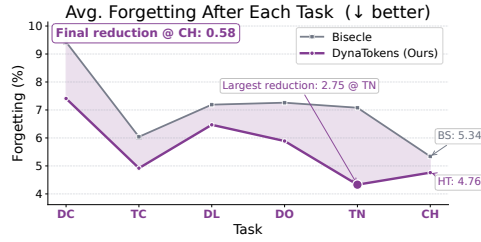
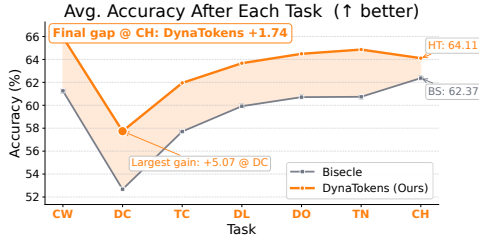
We attribute this robustness to a depth-wise specialisation mismatch revealed in Fig. 4 (left). Prompt similarity between ImageQA and VideoQA tasks peaks around layers 5–11 (mean  $\approx 0.7$ ) and then steadily declines toward the final layers (mean  $\approx 0.3$ ), indicating that mid-layer semantics are largely shared while *late layers specialise for temporal reasoning*. Strictly anchoring late-layer behaviour to the static-scene prior learned from ImageQA would therefore freeze the very plasticity that VideoQA most needs. Guided by this observation, we relax LA-Reg on the ImageQA task during the ImageQA → VideoQA transition, letting the generator reshape temporal-specific layers while the shared task code preserves reusable mid-layer structure. This selective relaxation is uniquely enabled by DYNATOKENS’s layer-conditioned generation and underlies its consistent advantage under cross-modal continual transfer.

## 6 Conclusion

We presented DYNATOKENS, a transformer-based token hyper-generator for continual VideoQA that synthesises fine-tuning tokens on demand, enabling *explicit, memory-bounded* adaptation with a fixed budget. To mitigate forgetting, we introduced a look-ahead regulariser that anchors the generator to prior tasks, with a sharpness-aware interpretation explaining its improved retention via flatter cross-task minima. A causal view of VideoQA further guided the weighting of auxiliary objectives to strengthen cross-modal alignment without anti-causal drift. Across standard continual VideoQA benchmarks, DYNATOKENS consistently improves accuracy and reduces forgetting, providing a strong foundation for lifelong multimodal learning.

### Limitations

**No positive cross-modal transfer.** On ImageQA → VideoQA, DYNATOKENS suffers far less negative transfer than the strongest



| Steps | Acc. ↑ | For. ↓ |
|-------|--------|--------|
| X     | 54.54  | 8.83   |
| 0     | 62.30  | 5.40   |
| 1     | 63.92  | 5.02   |
| 2     | 64.11  | 4.76   |

Figure 3: DYNATOKENS consistently surpasses Bisecle in average accuracy and forgetting across tasks. TP is excluded from the accuracy plot (uniformly low at start) and TP-CW from the forgetting plot (forgetting starts at DC).

Table 6: Effect of look-ahead steps on NExT-QA.

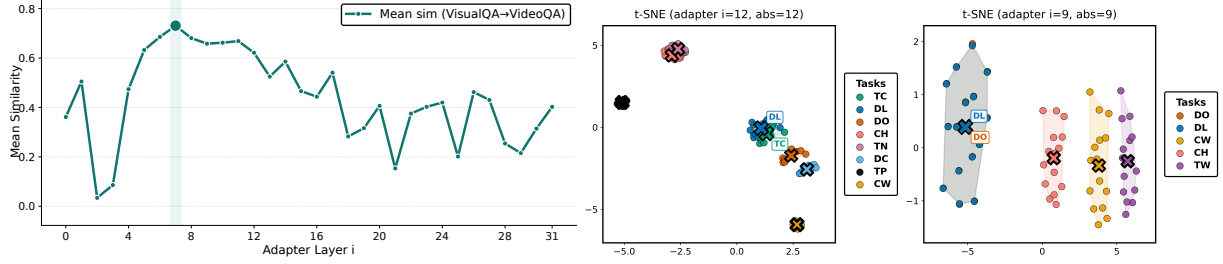


Figure 4: **Token analysis.** **Left:** Mean image-video token similarity across adapter layers after Visual7W→NExT-QA continual training. **Middle/Right:** t-SNE visualisations of task-specific prompt tokens at the semantically most-discriminative mid layer ( $i = 12$  for NExT-QA,  $i = 9$  for DramaQA).

baseline ( $-1.80$  vs  $-4.13$  Acc), but image pretraining does not yet yield a net gain on video reasoning, which remains open for any PEA framework.

**Training cost.** The look-ahead inner loop increases training time and peak memory relative to simpler PEA baselines (Table 16). The overhead does not recur at deployment and the 1-step variant recovers most of it, but training remains more expensive.

**Scope of distribution shift.** The standard NExT-QA and DramaQA splits are question-type-incremental. Our domain- and modality-incremental protocols go further, but streams with broader visual and temporal drift remain to be evaluated.

**Routing and horizon.** Nearest-key retrieval is less accurate on visually homogeneous streams (e.g., 41.7% on DramaQA); its downstream effect is small by design (0.50 Acc below oracle routing) but not eliminated. LA-Reg scales linearly with the number of stored codes, which is negligible at our scale and can be held constant by subsampling codes (Appendix E.7); our streams span up to 13 tasks, and longer horizons are left for future work.

**Future work.** Extensions include audio-visual reasoning, embodied perception, and long-horizon multi-task agents, where compact task-code memory and on-demand prompt synthesis suit heterogeneous task streams.

## Acknowledgements

This work was supported by the US Army International Technology Center Pacific (ITC-IPAC) under Contract No. FA520923C0020. This research is also supported by the ARC Training Centre for Whole Life Design of Carbon Neutral Infrastructure (IC230100015). The authors acknowledge ResetData for computational support, the National Computational Infrastructure (NCI Australia), an NCRIS-enabled capability supported by the Australian Government, for the computational resources used in this study, and the Katana computational cluster, supported by Research Technology Services at UNSW Sydney.

## References

- Sayanta Adhikari, Dupati Srikar Chandra, PK Srijith, Pankaj Wasnik, and Naoyuki Oneo. 2025. Adapre-fix++: Integrating adapters, prefixes and hypernetwork for continual learning. In *WACV*, pages 7298–7307. IEEE.
- Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, and 1 others. 2022. Flamingo: a visual language model for few-shot learning. *NeurIPS*, 35:23716–23736.
- Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. 2020. Dark experience for general continual learning: a strong, simple baseline. *NeurIPS*, 33:15920–15930.
- Chen Cai, Zheng Wang, Jianjun Gao, Wenyang Liu, Ye Lu, Runzhong Zhang, and Kim-Hui Yap. 2024. Empowering large language model for continual video question answering with collaborative prompting. In *EMNLP*, pages 3921–3932.
- Dupati Srikar Chandra, Sakshi Varshney, PK Srijith, and Sunil Gupta. 2023. Continual learning with dependency preserving hypernetworks. In *WACV*, pages 2338–2347. IEEE.
- Rujikorn Charakorn, Edoardo Cetin, Yujin Tang, and Robert Tjarko Lange. 2025. Text-to-loRA: Instant transformer adaption. In *ICML*.
- Feng Cheng, Ziyang Wang, Yi-Lin Sung, Yan-Bo Lin, Mohit Bansal, and Gedas Bertasius. 2025. Dam: Dynamic adapter merging for continual video qa learning. In *WACV*, pages 6805–6817. IEEE.
- Seongho Choi, Kyoung-Woon On, Yu-Jung Heo, Ahjeong Seo, Youwon Jang, Minsu Lee, and Byoung-Tak Zhang. 2021. Dramaqa: Character-centered video story understanding with hierarchical qa. In *AAAI*, volume 35, pages 1166–1174.
- Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, and 1 others. 2025. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models. In *CVPR*, pages 91–104. IEEE.
- 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. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*.
- Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2021. Sharpness-aware minimization for efficiently improving generalization. In *ICLR*.
- Qiankun Gao, Chen Zhao, Yifan Sun, Teng Xi, Gang Zhang, Bernard Ghanem, and Jian Zhang. 2023. A unified continual learning framework with general parameter-efficient tuning. In *ICCV*.
- Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *EMNLP*, pages 5484–5495.
- Gunshi Gupta, Karmesh Yadav, and Liam Paull. 2020. Look-ahead meta learning for continual learning. *NeurIPS*, 33:11588–11598.
- David Ha, Andrew M Dai, and Quoc V Le. 2017. Hypernetworks. In *ICLR*.
- Hamed Hemati, Vincenzo Lomonaco, Davide Bacciu, and Damian Borth. 2023. Partial hypernetworks for continual learning. In *CoLLAs*, pages 318–336. PMLR.
- Saurav Jha, Dong Gong, and Lina Yao. 2024. Clap4clip: Continual learning with probabilistic finetuning for vision-language models. *NeurIPS*, 37:129146–129186.
- William B Johnson, Joram Lindenstrauss, and 1 others. 1984. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1.
- Zixuan Ke, Bing Liu, and Xingchang Huang. 2020. Continual learning of a mixed sequence of similar and dissimilar tasks. *NeurIPS*, 33:18493–18504.
- Vishnuprasadh Kumaravelu, PK Srijith, and Sunil Gupta. 2025. Evocl: Continual learning over evolving domains. In *WACV*, pages 7533–7541.
- Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *EMNLP*.
- Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*.
- Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.
- Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *NeurIPS*, 36:34892–34916.
- Wenzhuo Liu, Fei Zhu, Longhui Wei, and Qi Tian. 2025a. C-clip: Multimodal continual learning for vision-language model. In *ICLR*.
- Yuyang Liu, Qiuhe Hong, Linlan Huang, Alexandra Gomez-Villa, Dipam Goswami, Tiantian Peng, Xialei Liu, Joost van de Weijer, and Yonghong Tian. 2025b. Continual learning for vlms: A survey and taxonomy beyond forgetting. *arXiv preprint arXiv:2508.04227*.

- Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Khan. 2024. Video-chatgpt: Towards detailed video understanding via large vision and language models. In *ACL*, pages 12585–12602.
- Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.
- Toan Nguyen, Kien Do, Duc Thanh Nguyen, Bao Duong, and Thin Nguyen. 2023. Causal inference via style transfer for out-of-distribution generalisation. In *KDD*, pages 1746–1757.
- Toan Nguyen, Duc Kieu, Bao Duong, Tung Kieu, Kien Do, Thin Nguyen, and Bac Le. 2024. Class-incremental learning with causal relational replay. *Expert Systems with Applications*, 250.
- Toan Nguyen, Yang Liu, Trung Le, Celso De Melo, and Flora D Salim. 2026. Fogo: Forgetting-aware orthogonalization optimizer. *arXiv preprint arXiv:2606.10406*.
- Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*.
- Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.
- Judea Pearl. 2009. *Causality*. Cambridge university press.
- Rui Qian, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Shuangrui Ding, Dahua Lin, and Jiaqi Wang. 2024. Streaming long video understanding with large language models. *NeurIPS*, 37:119336–119360.
- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763. PMLR.
- Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabza, Mike Lewis, and Amjad Almahairi. 2023. Progressive prompts: Continual learning for language models. In *ICLR*.
- Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauero. 2019. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *ICLR*.
- Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671*.
- Fangxun Shu, Lei Zhang, Hao Jiang, and Cihang Xie. 2025. Audio-visual llm for video understanding. In *ICCVW*, pages 4305–4314. IEEE.
- James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. 2023. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *CVPR*, pages 11909–11919. IEEE.
- Yue Tan, Xiaoqian Hu, Hao Xue, Celso De Melo, and Flora D Salim. 2025. Bisecl: Binding and separation in continual learning for video language understanding. *NeurIPS*.
- Zhaoxuan Tan, Zixuan Zhang, Haoyang Wen, Zheng Li, Rongzhi Zhang, Pei Chen, Fengran Mo, Zheyuan Liu, Qingkai Zeng, Qingyu Yin, and 1 others. 2026. Instant personalized large language model adaptation via hypernetwork. In *ACL*, pages 23557–23580.
- Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.
- Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, and 1 others. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- Johannes von Oswald, Christian Henning, Benjamin F Grewe, and João Sacramento. 2020. Continual learning with hypernetworks. In *ICLR*.
- Liyuan Wang, Jingyi Xie, Xingxing Zhang, Mingyi Huang, Hang Su, and Jun Zhu. 2023. Hierarchical decomposition of prompt-based continual learning: Rethinking obscured sub-optimality. *NeurIPS*.
- Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and 1 others. 2022a. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *ECCV*.
- Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. 2022b. Learning to prompt for continual learning. In *CVPR*.
- Junbin Xiao, Xindi Shang, Angela Yao, and Tat-Seng Chua. 2021. Next-qa: Next phase of question-answering to explaining temporal actions. In *CVPR*, pages 9777–9786.

Antoine Yang, Antoine Miech, Josef Sivic, Ivan Laptev, and Cordelia Schmid. 2022. Zero-shot video question answering via frozen bidirectional language models. *NeurIPS*, 35:124–141.

Jiazuo Yu, Zichen Huang, Yunzhi Zhuge, Lu Zhang, Ping Hu, Dong Wang, Huchuan Lu, and You He. 2025. Moe-adapters++: Towards more efficient continual learning of vision-language models via dynamic mixture-of-experts adapters. *TPAMI*.

Shoubin Yu, Jaemin Cho, Prateek Yadav, and Mohit Bansal. 2023. Self-chained image-language model for video localization and question answering. *NeurIPS*, 36:76749–76771.

Fanhu Zeng, Fei Zhu, Haiyang Guo, Xu-Yao Zhang, and Cheng-Lin Liu. 2025. Modalprompt: Towards efficient multimodal continual instruction tuning with dual-modality guided prompt. In *EMNLP*.

Ce Zhang, Taixi Lu, Md Mohaiminul Islam, Ziyang Wang, Shoubin Yu, Mohit Bansal, and Gedas Bertasius. 2024a. A simple llm framework for long-range video question-answering. In *EMNLP*, pages 21715–21737.

Renrui Zhang, Jiaming Han, Chris Liu, Aojun Zhou, Pan Lu, Yu Qiao, Hongsheng Li, and Peng Gao. 2024b. LLaMA-adapter: Efficient fine-tuning of large language models with zero-initialized attention. In *ICLR*.

Hongbo Zhao, Bolin Ni, Junsong Fan, Yuxi Wang, Yuntao Chen, Gaofeng Meng, and Zhaoxiang Zhang. 2024. Continual forgetting for pre-trained vision models. In *CVPR*, pages 28631–28642. IEEE.

Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. 2016. Visual7w: Grounded question answering in images. In *CVPR*, pages 4995–5004.

## A Extended Preliminaries

### A.1 LLaMA-Adapter

Following prior work (Tan et al., 2025; Cai et al., 2024), we adopt *LLaMA-Adapter* (Zhang et al., 2024b) for parameter-efficient continual adaptation of a frozen multimodal LLM on VideoQA. For each task  $t$ , we introduce a small set of adapter tokens at selected transformer layers,

$$P_l^t \in \mathbb{R}^{N_p \times C}, \quad l = 1, \dots, L_p, \quad (13)$$

injected into the frozen backbone and optimised under the objective in Eq. 8. In DYNATOKENS, the token generator  $H_\phi$  synthesises layer-wise adapter tokens  $\{P_l^t\}$  from compact task codes, so continual fine-tuning reduces to updating  $\phi$  rather than maintaining per-task adapter banks.

**Gated prefix attention.** At layer  $l$ , let  $X_l \in \mathbb{R}^{N \times C}$  be the hidden sequence of the frozen backbone (text tokens, preceded by visual tokens from a projector  $W_{\text{vis}}$  shared across tasks in the multimodal case). Each token in  $X_l$  attends to two sources: the other tokens of  $X_l$ , as in the original model, and the  $N_p$  adapter tokens  $P_l^t$ . Writing  $\text{Attn}(A, B)$  for attention with queries from  $A$  and keys/values from  $B$ , the layer computes

$$\begin{aligned} X'_l &= X_l + \text{Attn}(X_l, X_l) \\ &\quad + \tanh(g_l) \odot \text{Attn}(X_l, P_l^t), \\ X_{l+1} &= X'_l + \text{FFN}(X'_l). \end{aligned} \quad (14)$$

The two attention terms are normalised by separate softmaxes, so the first term is exactly the frozen model’s own self-attention and is never altered by the prefix. The second term injects the adapter tokens, scaled by a learnable per-head gate  $g_l \in \mathbb{R}^H$  passed through  $\tanh$ . Following LLaMA-Adapter,  $g_l$  is initialised to zero: training starts from the unmodified backbone and the influence of the prompts grows only as the gates open. Setting  $g_l = 0$  for all  $l$  therefore removes the generated prompts entirely, which we use as an ablation in Appendix E.6.

## B Theoretical Results

### B.1 Proof of Theorem 1

Before presenting the proof, we briefly recall the intuition behind *sharpness-aware minimisation* (SAM) (Foret et al., 2021).

**SAM and flat minima.** Given a loss  $\mathcal{L}(\theta)$ , SAM minimises the robust objective

$$\min_{\theta} \max_{\|\epsilon\| \leq \rho} \mathcal{L}(\theta + \epsilon), \quad (15)$$

seeking parameters whose loss stays small under worst-case perturbations within a ball of radius  $\rho$ . A first-order analysis gives the inner maximiser  $\epsilon^* = \rho \nabla_{\theta} \mathcal{L}(\theta) / \|\nabla_{\theta} \mathcal{L}(\theta)\|$ , so SAM penalises high-curvature directions and favours flat minima, which generalise better under distribution shift. This robustness perspective motivates LA-Reg, which can be read as enforcing prediction consistency on past task codes under a look-ahead parameter perturbation.

**Proof setup.** Fix  $\tau < t$  and write

$$H_{\phi}^{\tau} := H_{\phi}(z^{\tau}), \quad J_{\phi}^{\tau} := J_{\phi}(z^{\tau}), \quad (16)$$

$$a_{\tau} := H_{\phi}^{\tau} - H_{\phi}^{\tau*}, \quad b_{\tau} := J_{\phi}^{\tau} \Delta\phi. \quad (17)$$

*Proof.* By assumption (A1),  $H_{\phi}^{\tau}$  admits the second-order expansion

$$H_{\phi+\Delta\phi}^{\tau} = H_{\phi}^{\tau} + J_{\phi}^{\tau} \Delta\phi + r_{\tau}, \quad (18)$$

with  $\|r_{\tau}\|_2 \leq C_H \|\Delta\phi\|_2^2$ . Hence

$$H_{\phi+\Delta\phi}^{\tau} - H_{\phi}^{\tau} = a_{\tau} + b_{\tau} + r_{\tau}, \quad (19)$$

and substituting into Eq. (9) gives

$$\mathcal{L}_{\text{LA-Reg}}^t(\phi) = \sum_{\tau=1}^{t-1} \|a_{\tau} + b_{\tau} + r_{\tau}\|_2^2. \quad (20)$$

**Leading term.** Since  $\Delta\phi = -\eta g(\phi)$ , we have  $b_{\tau} = -\eta J_{\phi}^{\tau} g(\phi)$ , so

$$\sum_{\tau=1}^{t-1} \|b_{\tau}\|_2^2 = \eta^2 \sum_{\tau=1}^{t-1} \|J_{\phi}^{\tau} g(\phi)\|_2^2. \quad (21)$$

**Residual bound.** Expanding the squared norm of  $a_{\tau} + b_{\tau} + r_{\tau}$  and subtracting  $\|b_{\tau}\|_2^2$ , Cauchy-Schwarz yields

$$\begin{aligned} & \left| \mathcal{L}_{\text{LA-Reg}}^t(\phi) - \sum_{\tau < t} \|b_{\tau}\|_2^2 \right| \\ & \leq \sum_{\tau < t} \left( \|a_{\tau}\|_2^2 + \|r_{\tau}\|_2^2 \right. \\ & \quad + 2\|a_{\tau}\|_2 \|b_{\tau}\|_2 \\ & \quad + 2\|a_{\tau}\|_2 \|r_{\tau}\|_2 \\ & \quad \left. + 2\|b_{\tau}\|_2 \|r_{\tau}\|_2 \right). \end{aligned} \quad (22)$$

Assumptions (A1)–(A3) give the per-term bounds

$$\begin{aligned} \|a_{\tau}\|_2 & \leq \varepsilon, \\ \|b_{\tau}\|_2 & \leq L_J \eta G, \\ \|r_{\tau}\|_2 & \leq C_H \eta^2 G^2. \end{aligned} \quad (23)$$

Plugging these into Eq. (22),

$$\begin{aligned} & \left| \mathcal{L}_{\text{LA-Reg}}^t(\phi) - \sum_{\tau < t} \|b_{\tau}\|_2^2 \right| \\ & \leq (t-1) \left( \varepsilon^2 + 2L_J G \varepsilon \eta + 2C_H G^2 \varepsilon \eta^2 \right. \\ & \quad \left. + 2C_H L_J G^3 \eta^3 + C_H^2 G^4 \eta^4 \right). \end{aligned} \quad (24)$$

**Final simplification.** For sufficiently small  $\eta$  (say  $\eta \leq 1$ ), we have  $\eta^2 \leq \eta$  and  $\eta^4 \leq \eta^3$ . Hence Eq. (24) simplifies to

$$\left| \mathcal{L}_{\text{LA-Reg}}^t(\phi) - \sum_{\tau < t} \|b_{\tau}\|_2^2 \right| \leq C(\varepsilon^2 + \varepsilon \eta + \eta^3), \quad (25)$$

with

$$\begin{aligned} C & := (t-1) \max\{1, \\ & \quad 2L_J G + 2C_H G^2, \\ & \quad 2C_H L_J G^3 + C_H^2 G^4\}. \end{aligned} \quad (26)$$

which is independent of  $\eta$ . Substituting Eq. (21) into Eq. (25) yields Eq. (12).  $\square$

## B.2 Adaptation on a Shared Prompt Manifold

We formalise the claim in Sec. 4.2 that the token generator  $H_{\phi}$  confines local task-specific adaptation to a low-dimensional subspace of the full prompt space.

**Setup.** For clarity, we view the output of the hypernetwork as a vector in  $\mathbb{R}^{d_P}$ , where  $d_P := L_P N_P C$ , and similarly flatten task codes to  $\mathbb{R}^{d_z}$  with  $d_z := N_z C_z$ . The hypernetwork is then a map

$$H_{\phi} : \mathbb{R}^{d_z} \rightarrow \mathbb{R}^{d_P}. \quad (27)$$

Let  $J_z H_{\phi}(z) \in \mathbb{R}^{d_P \times d_z}$  denote its Jacobian with respect to the task code, and define the reachable prompt set

$$\mathcal{M}_{\phi} := \{H_{\phi}(z) : z \in \mathbb{R}^{d_z}\} \subseteq \mathbb{R}^{d_P}. \quad (28)$$

**Proposition 2** (Low-dimensional adaptation subspace). *Assume  $H_{\phi}$  is continuously differentiable. For any task code  $z^t$  and perturbation  $\delta z^t$ , the induced prompt change  $\delta P^t := H_{\phi}(z^t + \delta z^t) - H_{\phi}(z^t)$  satisfies*

$$\delta P^t = J_z H_{\phi}(z^t) \delta z^t + R(\delta z^t), \quad (29)$$

with  $\|R(\delta z^t)\|_2 = O(\|\delta z^t\|_2^2)$ . The first-order term lies in  $\text{Range}(J_z H_\phi(z^t))$ , a subspace of dimension at most  $d_z = N_z C_z$ , strictly smaller than  $d_P = L_p N_p C$  whenever  $d_z < d_P$ .

*Proof.* By continuous differentiability of  $H_\phi$  in  $z$ , Taylor’s theorem at  $z^t$  yields Eq. (29) with the stated remainder bound. The first-order term  $J_z H_\phi(z^t) \delta z^t$  lies in  $\text{Range}(J_z H_\phi(z^t))$  by definition, and the dimension of this range equals the rank of the Jacobian, which is bounded by the dimension of its domain:

$$\begin{aligned} \dim \text{Range}(J_z H_\phi(z^t)) &= \text{rank}(J_z H_\phi(z^t)) \\ &\leq \min(d_P, d_z) = d_z, \end{aligned} \quad (30)$$

where the last equality uses  $d_z \leq d_P$  in our configuration.  $\square$

**Interpretation.** Proposition 2 formalises why  $H_\phi$  acts as an implicit structural regulariser. Although the ambient prompt space has  $d_P$  degrees of freedom, task-specific updates through the compact code  $z^t$  can only move, to first order, within a subspace of dimension at most  $d_z \ll d_P$ . DYNATOKENS therefore biases task prompts toward a shared low-dimensional manifold while still permitting fine-grained task-specific control through the task code, discouraging task-specific overfitting and reducing the cross-task interference inherent in unconstrained prompt tuning. In our default LLaMA-2-7B configuration,  $d_z \approx 7.7 \times 10^3$  and  $d_P \approx 1.3 \times 10^6$ , corresponding to roughly a  $170\times$  compression of the local adaptation space (the per-task *storage* is smaller still,  $\approx 200\times$  below the prompt tensor, since  $z_{\text{inv}}$  is shared across tasks; Sec. 4).

### B.3 Empirical Verification in a Controlled Setting

Theorem 1 predicts that LA-Reg steers  $H_\phi$  toward regions where past-task outputs are insensitive to gradient-aligned perturbations, i.e., flatter cross-task solutions. To test this prediction directly, isolated from the LLM and the VideoQA objective, we construct a controlled five-task continual-learning problem on the generator alone.

**Setup.** Each task  $\tau$  is a fixed pair  $(z^\tau, y^\tau)$  with  $z^\tau \in \mathbb{R}^{16}$  and  $y^\tau \in \mathbb{R}^{32}$  drawn from a standard Gaussian, learned sequentially with an MSE objective.  $H_\phi$  is a three-layer MLP ( $16 \rightarrow 64 \rightarrow$

$64 \rightarrow 32$ , tanh activations,  $\sim 7\text{K}$  parameters). Using matched initialisation we compare (i) the task loss alone and (ii) the task loss plus LA-Reg,  $\sum_{\tau < t} \|H_\phi(z^\tau) - H_{\phi^*}(z^\tau)\|_2^2$ , evaluated at a two-step look-ahead point exactly as in Eq. 9. We report two quantities.  $\Delta\mathcal{L}@p$  is the increase in total-task loss after perturbing  $\phi$  by  $p=0.05$  along the gradient direction; lower values indicate a flatter solution. *Forgetting* is the mean loss on all earlier tasks at the end of the sequence; since each task is trained to convergence before the next begins, this equals the loss increase caused by subsequent training.

| Method     | $\Delta\mathcal{L}@p$ ( $\downarrow$ ) | Forgetting ( $\downarrow$ ) |
|------------|----------------------------------------|-----------------------------|
| w/o LA-Reg | $1.44 \pm 0.31$                        | $26.0 \pm 7.2$              |
| w/ LA-Reg  | $0.02 \pm 0.003$                       | $\approx 0$                 |

**Table 8: Controlled verification of Theorem 1** (mean  $\pm$  std over 10 seeds). LA-Reg reduces the sharpness measure by over  $60\times$  and nearly eliminates forgetting.

**Result.** Table 8 shows that LA-Reg reduces the sharpness measure by over  $60\times$  and drives forgetting to nearly zero, with both quantities decreasing consistently across all 10 seeds. This directly supports the link between flatter cross-task solutions and improved retention, and mirrors the trend on NExT-QA, where forgetting falls from 8.83 to 4.76 and decreases further with look-ahead depth (Table 6). The script for this experiment is included in our code release.

## C Additional Experimental Details

### C.1 Architecture

**$H_\phi$  implementation.** We use the same generator across all three datasets (NExT-QA, DramaQA, Visual7W). Its computation proceeds in four steps. (i) *Code projection.* The task code  $z^t \in \mathbb{R}^{N_z \times C_z}$ , with  $N_z = 30$  slots (5 invariant, 25 task-specific) of width  $C_z = 256$ , is projected to the generator’s internal width  $d = 384$  and given a positional embedding, yielding  $X^t$ . (ii) *Layer conditioning.* A table of  $L_p = 32$  learnable layer embeddings  $\{e_\ell\}$ , one per adapter layer, predicts a FiLM scale and shift,  $X \odot (1 + \tanh \gamma_\ell) + \beta_\ell$ , which is applied to  $X^t$  and to the query tokens below; this is how a single generator produces different prompts for different layers. (iii) *Query initialisation.* The  $N_p$  initial query tokens are derived from the code: an attention-pooling head summarises  $X^t$  into  $N_p$

vectors, each passes through a rank- $r=32$  bottleneck onto a shared basis, and a learned position bias is added; a first cross-attention from these queries to  $X^t$  gives  $U^{(0)}$ . (iv) *Refinement and output.*  $U^{(0)}$  is refined by 2 pre-norm blocks of cross-attention (to  $X^t$ ), self-attention, and MLP, each with 8 heads and dropout 0.2, and a shared linear head  $W_o$  maps the result from  $d = 384$  to the LLaMA-2-7B hidden dimension  $C = 4096$ , producing the  $N_p$  prompt tokens for layer  $\ell$ .

## C.2 Hyper-parameters and Continual Fine-tuning Details

**Routing.** The task-key dimension is set to  $D_k = 256$ . For efficiency, we maintain an exponential moving average of the task key updated batch-wise with momentum 0.95, which closely approximates the dataset-level mean (Eq. 5) at a fraction of the cost. The EMA introduces neither learnable parameters nor gradient updates; computing the exact key requires an additional full pass over the training set, whereas the same batch embeddings are already available during training. Table 9 confirms the approximation is essentially lossless, as expected since the embeddings come from frozen modules and do not evolve during training.

| Key computation     | Routing Acc | Acc   | Fog  |
|---------------------|-------------|-------|------|
| Exact mean (Eq. 5)  | 69.76%      | 64.14 | 4.69 |
| EMA (momentum 0.95) | 69.00%      | 64.11 | 4.76 |

**Table 9: EMA vs. exact task keys on NExT-QA.** The online EMA matches the exact dataset-level mean within 0.03 Acc.

**Training schedule.** We fine-tune on NVIDIA H200 GPUs with a two-epoch warm-up. Epoch budgets are matched across methods: five epochs for all methods on NExT-QA, and six epochs for all methods in the continual-dataset setting (NExT-QA→DramaQA), which inherits the DramaQA configuration end-to-end. The only exception is the single-dataset DramaQA setting, where DYNATOKENS uses six epochs while the baselines are reported at their published five-epoch results (Tan et al., 2025; Cai et al., 2024); to verify this does not tilt the comparison, we re-ran the baselines for six epochs and observed no material change in accuracy, as they had already converged by epoch five. The NExT-QA stage of the Visual7W→NExT-QA transfer reuses the standard NExT-QA configuration.

**Pretraining on Visual7W.** We pretrain both Bisele and DYNATOKENS for ten epochs with a four-epoch warm-up, using AdamW with batch size 16 and gradient accumulation over 64 steps; all remaining hyperparameters follow the protocol above. We select the checkpoint with the best validation accuracy as the starting point for continual adaptation on NExT-QA.

**Summary of hyperparameters.** Table 10 lists every setting used. The DYNATOKENS architecture, loss weights, and look-ahead configuration are identical across datasets; the backbone-side training recipe (learning rate, weight decay, adapter length, sequence length, and epochs) follows the published per-dataset recipe of Bisele (Tan et al., 2025) so that all methods share the same schedule. The *base learning rate* applies to the backbone-side trainable modules of LLaMA-Adapter, namely the visual projector (which maps CLIP features into the LLaMA hidden space) and the adapter gates; the DYNATOKENS modules use their own smaller rates.

|                                                                                                | NExT-QA                                   | DramaQA              |
|------------------------------------------------------------------------------------------------|-------------------------------------------|----------------------|
| <i>Shared across datasets</i>                                                                  |                                           |                      |
| Backbone / visual encoder                                                                      | LLaMA-2-7B / CLIP ViT-L/14                |                      |
| Adapter layers $L_p$                                                                           | 32                                        |                      |
| Task code $N_z (N_{\text{inv}} + N_{\text{sp}}) / C_z$                                         | 30 (5+25) / 256                           |                      |
| Generator $H_\phi$                                                                             | 2 blocks, width 384, 8 heads, dropout 0.2 |                      |
| Key dim. $D_k$ / EMA momentum                                                                  | 256 / 0.95                                |                      |
| $\lambda_{\text{LA-Reg}} / \lambda_{\text{inv}} / \lambda_{\text{VAQ}} / \lambda_{\text{QAV}}$ | 0.5 / 0.1 / 1.0 / 0.1                     |                      |
| Look-ahead steps / inner step size                                                             | 2 / 0.5 $\times$ base LR                  |                      |
| Optimiser / $(\beta_1, \beta_2)$                                                               | AdamW / (0.9, 0.95)                       |                      |
| LR: generator $H_\phi$ / $z_{\text{sp}}$                                                       | $10^{-3}$ / $10^{-3}$                     |                      |
| Effective batch size (4 per GPU)                                                               | 128                                       |                      |
| <i>Per-dataset (following Biselec)</i>                                                         |                                           |                      |
| Epochs                                                                                         | 5                                         | 6                    |
| Adapter tokens per layer $N_p$                                                                 | 10                                        | 15                   |
| Max text length (tokens)                                                                       | 128                                       | 280                  |
| Base LR (projector, gates)                                                                     | $5 \times 10^{-3}$                        | $7.5 \times 10^{-3}$ |
| LR: $z_{\text{inv}}$                                                                           | $10^{-3}$                                 | $5 \times 10^{-4}$   |
| Weight decay                                                                                   | 0.14                                      | 0.10                 |

**Table 10: Complete hyperparameter settings for DYNATOKENS.** The continual-dataset stream inherits the DramaQA column; the Visual7W→NExT-QA stage reuses the NExT-QA column.

## D Extended Related Work

**Prompt-based continual learning.** L2P (Wang et al., 2022b) pioneers prompt retrieval from a fixed pool via prompt tuning (Lester et al., 2021) but suffers from interference due to limited prompt capacity. DualPrompt (Wang et al., 2022a) separates task-invariant (G-Prompt) from task-specific (E-Prompt) components, and CODA-Prompt (Smith et al., 2023) refines this with attention-based prompting. LAE (Gao et al., 2023) couples an online PEA

module with an offline one via an adaptation-speed calibrator, while HiDe-Prompt (Wang et al., 2023) adopts hierarchical prompts with inference-time ensembling. A different line stores per-task prompts to avoid interference: ProgPrompt (Razdaibiedina et al., 2023) concatenates frozen task prompts in the spirit of progressive networks (Rusu et al., 2016), and ModalPrompt (Zeng et al., 2025) extends this to multimodal continual instruction tuning by using CLIP-based dual-modality similarity for prompt fusion and top- $k$  selection. While effective, these methods maintain prompt banks that grow linearly with the task stream, and ModalPrompt additionally depends on auxiliary CLIP encoders for routing.

**Hypernetworks for continual adaptation.** Hypernetworks (Ha et al., 2017) produce the parameters of another network conditioned on an auxiliary signal. In continual learning, prior work typically conditions on task identity to synthesise full or partial model weights (von Oswald et al., 2020; Hemati et al., 2023; Chandra et al., 2023; Kumar-avelu et al., 2025), which is computationally costly and poorly suited to LLM-scale models. AdaPrefix++ (Adhikari et al., 2025) mitigates this cost by generating only prefix parameters from trainable per-task embeddings through a shared hypernetwork. It is conceptually the closest method to DYNATOKENS, and we detail the differences here. *Setting.* AdaPrefix++ targets image classification with a fixed vision backbone, whereas DYNATOKENS addresses generative VideoQA with temporal inputs and is evaluated under cross-dataset and cross-modality shifts. *Routing.* Both can operate without task identity but by different mechanisms: AdaPrefix++ infers the task by running a forward pass through every task’s head and selecting the lowest-entropy prediction, an  $O(T)$  cost that grows with the number of tasks and depends on model confidence; DYNATOKENS routes by a single cosine lookup in a frozen embedding space ( $O(1)$ ), which does not drift as tasks accumulate. *Architecture.* Both store a compact per-task input to a shared generator (task embeddings vs. task codes), but AdaPrefix++ additionally attaches a task-specific adapter to the backbone for every task, whereas DYNATOKENS adds no per-task backbone modules; its only per-task state is a routing key and code. *Regularisation.* AdaPrefix++ regularises against stored past behaviour without look-ahead and provides no theoretical account of its regulariser; DYNATOKENS adds LA-Reg, characterised in Theorem 1

as a cross-task sharpness penalty. Table 6 isolates the contribution of look-ahead alone (+1.81 Acc, −0.64 Fog from 0 to 2 steps). AdaPrefix++ has no public implementation, so we compare along these design dimensions rather than empirically. Recent hypernetwork-based LLM adaptation methods (Charakorn et al., 2025; Tan et al., 2026) target different settings (efficient fine-tuning, in-context adaptation) and lack continual-learning properties.

**Image Question Answering (ImageQA).** ImageQA is a core multimodal task requiring models to ground visual content in natural-language answers, and recent multimodal LLMs have substantially improved its performance (Team et al., 2023; Liu et al., 2023). Yet these systems are predominantly trained offline on stationary data distributions and are not designed for continual learning from evolving multimodal streams. Whether and how they transfer to the more demanding continual VideoQA setting, which adds temporal reasoning on top of distribution shift, remains largely unexplored. We provide one of the first systematic evaluations of this transfer in our ImageQA→VideoQA protocol (Sec. 5).

**Meta-Learning for Continual Learning.** Meta-learning mitigates forgetting by optimising parameters, or update rules, for rapid adaptation with minimal interference. Existing meta-CL methods focus primarily on shaping the optimisation trajectory: MER aligns gradients across replayed experiences via Reptile-style updates (Nichol et al., 2018; Riemer et al., 2019), and La-MAML extends MAML with a look-ahead objective and per-parameter learning-rate modulation for online continual learning (Gupta et al., 2020). We target a substantially harder regime, prompt-based continual adaptation of multimodal LLMs for video reasoning, and depart from prior meta-CL work in two ways. First, our look-ahead term regularises the *prompt-output space* rather than the parameter or gradient space, making it directly compatible with frozen-backbone PEA. Second, Theorem 1 gives a first-order characterisation of our look-ahead term as a cross-task sharpness penalty, connecting it to sharpness-aware minimisation.

## E More Results and Analyses

### E.1 Detailed Token Analysis

**Layer selection.** We select visualisation layers by silhouette score across all 32 adapter layers,

picking the middle-layer maximum:  $i = 12$  for NExT-QA,  $i = 9$  for DramaQA. Early layers process inputs; late layers commit to output generation; middle layers tend to encode the most task-discriminative representations.

**NExT-QA (Fig. 4, middle).** Three regions emerge: (i) a *sequential-reasoning* cluster merging TN (Temporal-Next: predicting what happens next) and CH (Causal-How: explaining how an event unfolds), both involving temporal event prediction; (ii) a *descriptive continuum* where DL (Location), TC (Count), DO (Object), and DC (Description) arrange from spatial-temporal grounding to cardinality counting; and (iii) two outliers, TP (Temporal-Previous) and CW (Causal-Why), closer to static-state and abstract-causal reasoning respectively. Interestingly, this organisation does not align with the official NExT-QA taxonomy (Causal/Temporal/Descriptive), suggesting DYNATOKENS groups tasks more by reasoning *skill* than by question type.

**DramaQA (Fig. 4, right).** A similar skill-oriented pattern appears. DO (Descriptive-Object) and DL (Descriptive-Location) form a spatial-visual grouping (locating an entity arguably requires identifying it first), while CH (Causal-How), CW (Causal-Why), and TW (Temporal-When) occupy distinct strips associated with procedural, abstract-causal, and temporal reasoning. CH and CW are also relatively well separated despite sharing the “Causal” taxonomy label, hinting that DYNATOKENS captures some distinction between *how* and *why* at the representational level.

**Cross-dataset consistency.** This skill-oriented organisation appears across both datasets, which are trained independently with different question types and orderings. While the visualisations are exploratory, they are consistent with the hypothesis that DYNATOKENS captures transferable reasoning structure rather than purely dataset-specific patterns.

## E.2 Impact of $\lambda_{\text{LA-Reg}}$

We sweep  $\lambda_{\text{LA-Reg}}$  over  $\{0, 0.25, 0.5, 0.75, 1.0, 1.5\}$  on NExT-QA. Table 11 shows that LA-Reg substantially improves over the no-regularisation baseline, peaks at 0.5–0.75, and saturates beyond 1.0, confirming robustness to a wide range of weights.

**Table 11: Effect of  $\lambda_{\text{LA-Reg}}$  on NExT-QA.** We use  $\lambda_{\text{LA-Reg}} = 0.5$  as default.

| $\lambda_{\text{LA-Reg}}$ | Acc $\uparrow$ | Fog $\downarrow$ |
|---------------------------|----------------|------------------|
| 0.00                      | 54.91          | 8.39             |
| 0.25                      | 63.46          | 5.01             |
| <b>0.50</b>               | <b>64.11</b>   | <b>4.76</b>      |
| 0.75                      | <u>63.92</u>   | <u>4.89</u>      |
| 1.00                      | 63.70          | 4.98             |
| 1.50                      | 63.59          | 5.07             |

**Table 12: Effect of look-ahead step size  $\eta_{\text{LA}}$  on NExT-QA.** Performance is robust for  $\eta_{\text{LA}} < 1.0$  and degrades for larger values, which over-fit the look-ahead probe to the current task. Best in **bold**, second-best underlined.

| $\eta_{\text{LA}}$ | Acc $\uparrow$ | Fog $\downarrow$ |
|--------------------|----------------|------------------|
| 0.10               | 63.74          | 4.92             |
| 0.25               | <u>63.96</u>   | <u>4.84</u>      |
| <b>0.50</b>        | <b>64.11</b>   | <b>4.76</b>      |
| 0.75               | 63.39          | 5.07             |
| 1.00               | 62.72          | 5.23             |
| 1.50               | 61.81          | 6.03             |

## E.3 Impact of Look-ahead learning rate

Table 12 sweeps the look-ahead step size  $\eta_{\text{LA}}$  on NExT-QA. DYNATOKENS achieves the best **Acc** (64.11) and lowest **Fog** (4.76) at  $\eta_{\text{LA}} = 0.50$ , and stays within 0.72 **Acc** for  $\eta_{\text{LA}} \leq 0.75$ , indicating robustness to fine-grained tuning. Beyond  $\eta_{\text{LA}} \geq 1.0$ , both metrics degrade monotonically (up to  $-2.30$  **Acc** and  $+1.27$  **Fog** at  $\eta_{\text{LA}} = 1.5$ ): large inner steps push the probe into the current task’s local minimum, so the regulariser encodes *overfitted directions* rather than local flatness, matching our SAM-based motivation. We use  $\eta_{\text{LA}} = 0.50$  as default.

## E.4 Impact of Video Auxiliary Loss

Our causal analysis (Sec. 4) treats  $\mathcal{L}_{\text{QAV}}$  as an anti-causal but useful grounding signal, prescribing a small  $\lambda_{\text{QAV}}$ . Table 13 confirms the predicted *inverted-U*: removing QAV ( $\lambda_{\text{QAV}} = 0$ ) drops **Acc** to 63.75 as visual grounding is lost, the optimum at  $\lambda_{\text{QAV}} = 0.10$  reaches 64.11 **Acc** and 4.76 **Fog**, and up-weighting progressively degrades to 62.21 **Acc** at  $\lambda_{\text{QAV}} = 1.50$  ( $-1.90$  **Acc**,  $+0.73$  **Fog**) as the underdetermined  $p(V | Q, A)$  pulls the model along anti-causal paths. Even the  $\lambda_{\text{QAV}} = 0$  result highlights the importance of the causal  $\mathcal{L}_{\text{VAQ}}$ , which alone lifts **Acc** from 62.41 (Table 4, no auxiliary) to 63.75. The same trend is reported in concurrent VideoQA work (Tan et al., 2025). We adopt  $\lambda_{\text{QAV}} = 0.10$  throughout.

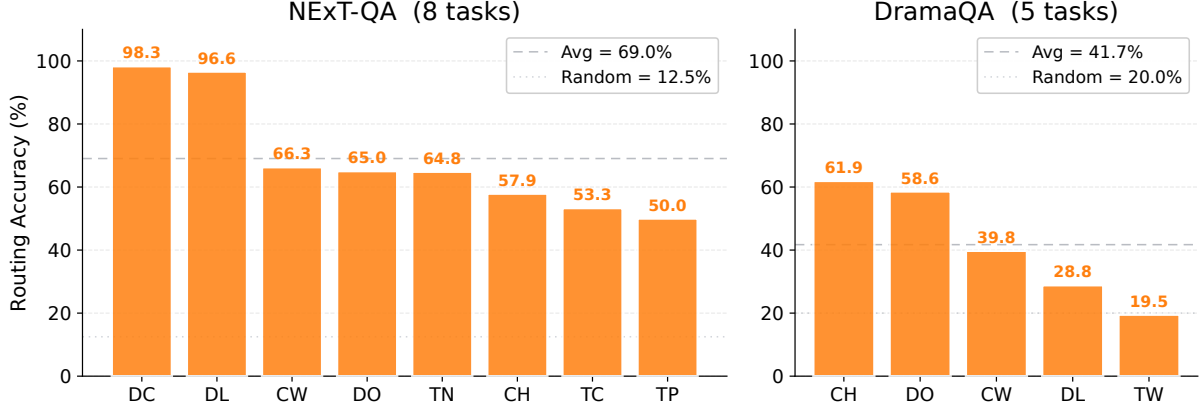


Figure 5: **Per-task accuracy of our training-free routing** on NExT-QA (left, 8 tasks) and DramaQA (right, 5 tasks). Bars show retrieval accuracy via cosine similarity in the frozen embedding space; dashed and dotted lines mark the dataset-wise average and the random baseline.

**Table 13: Effect of QAV auxiliary loss weight  $\lambda_{QAV}$  on NExT-QA.** Small weights ( $\lambda_{QAV} \leq 0.25$ ) yield best performance; disabling QAV ( $\lambda_{QAV} = 0$ ) loses cross-modal supervision, while large weights induce anti-causal drift as predicted by our causal analysis. Best in **bold**.

| $\lambda_{QAV}$ | Acc $\uparrow$ | Fog $\downarrow$ |
|-----------------|----------------|------------------|
| 0.00            | 63.75          | 4.98             |
| <b>0.10</b>     | <b>64.11</b>   | <b>4.76</b>      |
| 0.25            | 64.02          | 4.77             |
| 0.50            | 63.42          | 5.10             |
| 0.75            | 63.58          | 5.03             |
| 1.00            | 62.68          | 5.24             |
| 1.50            | 62.21          | 5.49             |

**Table 14: Training-free vs oracle routing** for DYNATOKENS. Minor gaps on both benchmarks confirm that downstream performance is largely decoupled from routing accuracy.

| Routing              | NExT-QA        |                  | DramaQA        |                  |
|----------------------|----------------|------------------|----------------|------------------|
|                      | Acc $\uparrow$ | Fog $\downarrow$ | Acc $\uparrow$ | Fog $\downarrow$ |
| Training-free (ours) | 64.11          | 4.76             | 72.52          | 9.76             |
| Oracle               | 64.61          | 4.14             | 72.90          | 9.01             |
| $\Delta$             | -0.50          | +0.62            | -0.38          | +0.75            |

### E.5 Routing Accuracy.

Fig. 5 reports per-task accuracy of our gradient-free routing. NExT-QA reaches 69.0% on average ( $5.5\times$  random); DramaQA drops to 41.7% ( $2\times$  random) due to heavy overlap among its five Wh-categories sharing a single TV-series visual stream. Nearest-key errors are thus systematic rather than random: they concentrate among tasks that share visual context, so a misrouted query lands on a se-

mantically adjacent task whose code produces a similar prompt. Crucially, this gap does not translate to downstream loss: against an *oracle* variant with ground-truth task identity (Table 14), DYNATOKENS loses only 0.50 Acc on NExT-QA and 0.38 on DramaQA. The reason is that discriminability lives in *token space*, not in pretrained key space: although DramaQA’s Wh-categories are barely separable for frozen embeddings, the prompts generated by  $H_\phi$  form well-separated clusters in the LLM hidden space (see token analysis in Sec. 5). Combined with the shared  $z_{inv}$ , this allows neighbouring task codes to produce visually appropriate prompts under imperfect routing, so end performance is largely insensitive to routing accuracy by design. Consistent with this, retrieval errors are systematic rather than random: when the router errs, it selects a semantically adjacent task rather than an arbitrary one, so a misrouted query receives a code that produces a similar prompt. The next subsection bounds the worst case directly.

### E.6 Task-Code Contribution and the Prompt Pathway

The small oracle gap in Table 14 measures the cost of imperfect routing, not the importance of the task codes. To isolate the contribution of the retrieved code, we replace it at inference with the code of the *most-dissimilar* stored task (lowest key cosine similarity) while keeping the shared component  $z_{inv}$  fixed. Table 15 shows that this far more extreme substitution than routing ever makes lowers Acc from 64.11 to 62.14, confirming that the task-specific component carries measurable task-relevant information; the drop is moderate because

transferable structure is intentionally captured by the generator and  $z_{\text{inv}}$ .

| Code assignment (NExT-QA)             | Acc   |
|---------------------------------------|-------|
| Oracle                                | 64.61 |
| Retrieved (ours)                      | 64.11 |
| Most-dissimilar task code             | 62.14 |
| Prompt pathway disabled ( $g_l = 0$ ) | 25.24 |

**Table 15: Code-swap and prompt-pathway ablation** on NExT-QA. A wrong code costs 1.97 points; removing the generated prompt entirely costs 38.87.

The prompt pathway itself is load-bearing: zeroing the adapter gates ( $g_l = 0$  in Eq. 14), which removes the generated prompt from every layer, collapses accuracy to 25.24, a 38.87-point drop, whereas the most-dissimilar code costs only 1.97 points. This contrast clarifies the division of roles: the generated prompt carries the transferable structure, while the task-specific code refines it. Consistently, the learned gate modulation is selective and concentrated in a small number of mid-to-late-layer heads, with  $\text{mean}(\tanh g_l) \approx 0.01$  and  $\max |\tanh g_l| \approx 0.25$  (the per-layer maximum grows from 0.03 in early layers to 0.25 in the final layer). The small oracle gap therefore reflects robustness to imperfect routing rather than a lack of task-code contribution.

## E.7 Efficiency Analysis

Table 16 reports trainable parameters, training time, and peak GPU memory. We measure efficiency under matched batch size of four on  $2 \times$  H200 GPUs, with  $\mathcal{L}_{\text{VAQ}}$  active for all methods and  $\mathcal{L}_{\text{QAV}}$  omitted since ProgPrompt is incompatible with it. DYNATOKENS uses 9.6M trainable parameters, roughly  $3 \times$  Bisecle and  $2 \times$  ProgPrompt, yet still under 0.15% of the 7B-parameter backbone. Training cost and peak memory are correspondingly higher, reflecting two principled design choices: (i) a transformer-based token generator that synthesises layer-specific prompts on demand, and (ii) the look-ahead inner loop in LA-Reg, which doubles forward passes per step. Crucially, reducing LA-Reg to a single look-ahead step (DYNATOKENS 1-step) more than halves training time ( $5.55 \rightarrow 2.46$ h on NExT-QA,  $15.58 \rightarrow 8.03$ h on the joint stream) and trims peak memory by  $\sim 20\%$ , while still surpassing Bisecle on both metrics (63.92 vs 62.37 Acc, 5.02 vs 5.34 Fog).

ProgPrompt’s lower wall-clock also stems from a classification objective that sidesteps autoregres-

sive next-token prediction. Its limitation, however, is scalability: memory *grows with the task sequence* ( $21.16 \rightarrow 25.43$  on NExT-QA,  $22.98 \rightarrow 29.34$  on the joint stream) since per-task adapters and prompts must be retained, an overhead that compounds over long continual deployments. DYNATOKENS’s footprint, by contrast, remains *constant* across the task stream: each new task adds only kilobytes through the compact task code, independent of stream length.

Overall, DYNATOKENS’s additional compute is a one-time investment in the generator and look-ahead optimisation, repaid on three fronts: (i) consistent SOTA accuracy and forgetting across single-dataset, multi-dataset, and ImageQA  $\rightarrow$  VideoQA settings; (ii) stronger zero-shot generalisation to unseen datasets at test time; and (iii) constant per-task memory growth that scales gracefully to long task streams—a property whose value only increases as deployments accumulate more tasks.

**LA-Reg at long horizons.** LA-Reg scales linearly with the number of stored task codes, but each term requires only a forward pass through the small generator  $H_\phi$  ( $\approx 6.4$ M parameters), not the 7B LLM. Table 17 reports a micro-benchmark of the per-step overhead: at 50 stored codes it is 9.47 ms, i.e., 2.2% of a full training step (438.9 ms). The cost can also be made independent of the horizon by sampling a fixed number of stored codes per step: on the 13-task NExT-QA  $\rightarrow$  DramaQA stream, sampling only 4 codes yields 61.19 Acc / 11.96 Fog versus 61.96 / 11.72 with all codes, a 0.77-point reduction that remains far above Bisecle (55.31 Acc). Regarding plasticity, since the regulariser sums over  $t - 1$  codes, averaging over stored codes (equivalently  $\lambda_{\text{LA-Reg}} \propto \frac{1}{t-1}$ ) keeps its per-task strength constant; our fixed setting nonetheless shows no degradation in new-task learning over the 13-task stream.

**Joint-training upper bound.** To place the gains in context, Table 18 reports joint (non-continual) training on all eight NExT-QA tasks under the same backbone and training budget. DYNATOKENS reaches 93.7% of the joint-training accuracy versus 91.2% for Bisecle; equivalently, it closes about 29% of the gap between Bisecle and the non-continual ceiling.

**Table 16: Computational cost comparison.** Fine-tuned parameters, training time (hours), and peak GPU memory (GB) across continual VideoQA settings. Numbers in parentheses indicate timing without auxiliary losses. For ProgPrompt, peak memory grows across the task stream (initial → final).

| Metric                               | Bisecle | ProgPrompt  | DYNATOKENS | DYNATOKENS (1 step) |
|--------------------------------------|---------|-------------|------------|---------------------|
| Fine-tuned parameters (M)            | 3.2M    | 5.1M        | 9.6M       | 9.6M                |
| Training time on NExT-QA             | 1.48    | 2.13        | 5.55       | 2.46                |
| Peak GPU Mem on NExT-QA (GB)         | 18.9    | 21.16→25.43 | 103.8      | 82.0                |
| Training time on NExT-QA→DramaQA     | 5.77    | 6.42        | 15.58      | 8.03                |
| Peak GPU Mem on NExT-QA→DramaQA (GB) | 18.9    | 22.98→29.34 | 103.9      | 82.1                |

|                               |      |      |      |
|-------------------------------|------|------|------|
| Stored codes                  | 12   | 25   | 50   |
| LA-Reg overhead per step (ms) | 2.74 | 5.05 | 9.47 |

**Table 17: LA-Reg overhead vs. number of stored codes** (per training step; full step = 438.9 ms).

| Method (NExT-QA)             | Acc   |
|------------------------------|-------|
| Joint training (upper bound) | 68.42 |
| DYNATOKENS                   | 64.11 |
| Bisecle                      | 62.37 |

**Table 18: Joint-training upper bound** on NExT-QA.

**Table 19: Performance with different numbers of prompt layers.**

| # P. Layer | Acc (↑) | Fog (↓) |
|------------|---------|---------|
| 8          | 54.32   | 9.02    |
| 16         | 58.30   | 7.13    |
| 24         | 59.04   | 6.87    |
| 32         | 64.11   | 4.76    |

video tasks, DYNATOKENS still predicts the correct answer, whereas Bisecle fails on these Visual7W examples.

## E.8 Sensitivity to the Number of Prompt-Injection Layers

We study how DYNATOKENS varies with the number of adapted adapter layers in Table 19. Overall, increasing the adapted depth consistently improves performance, supporting the effectiveness of parameter-efficient adaptation for continual VideoQA.

## E.9 Task Order

We evaluate DYNATOKENS under different task orders on NExT-QA and compare against Bisecle in Table 20. DYNATOKENS is more robust, outperforming Bisecle on three orders while remaining competitive on the fourth, supporting the effectiveness of multimodal token hyper-generation.

## F Qualitative Results

### F.1 Continual VideoQA

Fig. 6 and Fig. 7 present qualitative comparisons in continual VideoQA. DYNATOKENS consistently yields the correct answer, whereas Bisecle fails in these cases.

### F.2 ImageQA→VideoQA transfer

Fig. 8 show qualitative results for continual ImageQA→VideoQA transfer. After adapting to

**Table 20:** Robustness to task order on NExT-QA: Biseclc versus DYNATOKENS , reporting average accuracy and forgetting across four task permutations.

| Task Order                       | Acc (↑)      |              | Fog (↓) |             |
|----------------------------------|--------------|--------------|---------|-------------|
|                                  | Biseclc      | DYNATOKENS   | Biseclc | DYNATOKENS  |
| <TP, TN, CH, TC, DL, DO, CW, DC> | 57.98        | <b>61.23</b> | 7.16    | <b>6.82</b> |
| <DO, CW, DC, CH, TP, TC, TN, DL> | 57.95        | <b>61.35</b> | 8.93    | <b>6.91</b> |
| <CW, DO, TN, DL, TC, TP, DC, CH> | 62.25        | <b>62.79</b> | 5.70    | <b>4.96</b> |
| <CH, DL, TP, TC, DC, DO, TN, CW> | <b>63.09</b> | 62.96        | 2.87    | <b>2.79</b> |

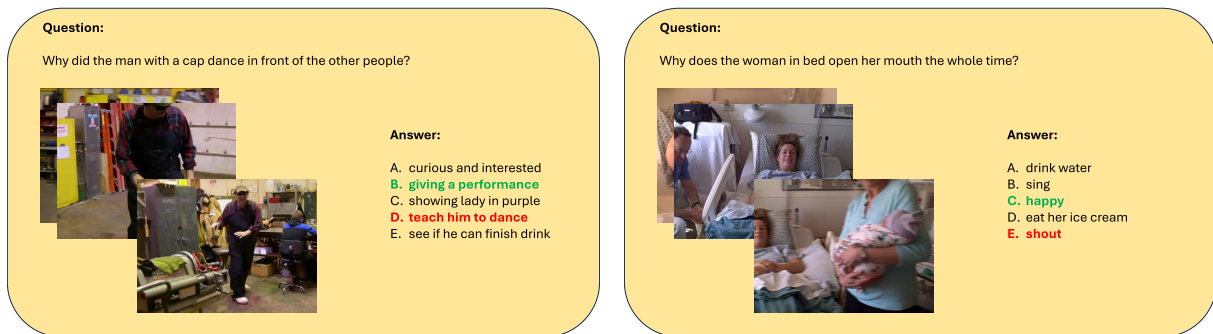


Figure 6: **Qualitative Examples 1–2.** DYNATOKENS predicts the correct answer (green), whereas Biseclc produces an incorrect one (red) in continual VideoQA.



Figure 7: **Qualitative Examples 3–4.** DYNATOKENS predicts the correct answer (green), whereas Biseclc produces an incorrect one (red) in continual VideoQA.

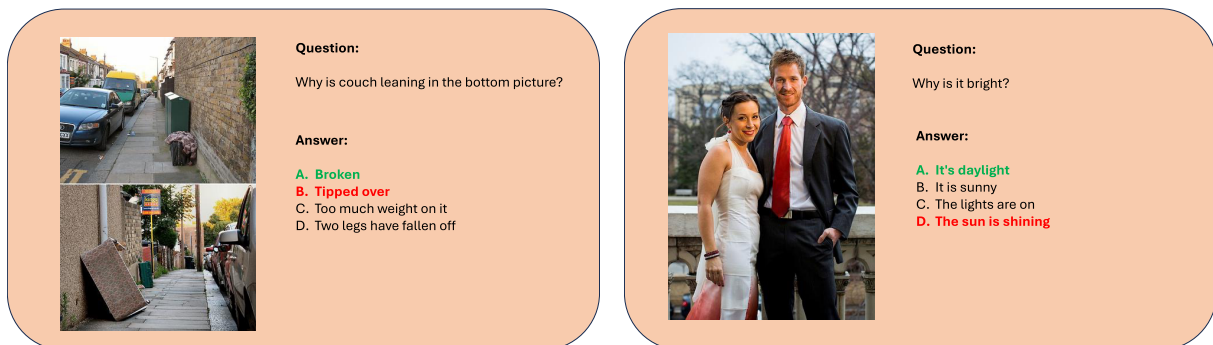


Figure 8: **Qualitative Examples 5–6.** After adapting to video tasks, DYNATOKENS still predicts the correct answer (green), whereas Biseclc fails on these Visual7W examples (red).