OFFLINE MAKES ONLINE BETTER: CALIBRATED CACHE COMPRESSION FOR VIDEO GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Token merging is an effective approach to reducing KV-cache memory consumption in autoregressive video generation by clustering temporally redundant tokens and retaining only a small set of representatives. Existing methods, however, typically apply a uniform merging ratio across all attention layers and heads. We observe that different layers and heads exhibit substantially different yet jointly correlated sensitivities to token merging, and that these sensitivity patterns remain consistent across prompts. Based on this observation, we introduce CALM, a Calibrated Allocation for Layer-Head Merging that uses offline calibration to improve online cache compression. Exhaustively evaluating all possible layer-head allocations is prohibitively expensive. To enable efficient calibration, we show that attention-output error strongly correlates with end-to-end video quality and can therefore serve as a lightweight proxy for allocation risk assessment. We evaluate different allocation strategies using this proxy and construct a fixed layer-head risk map offline. Given a cache budget at inference time, CALM uses dynamic programming to determine the best allocation of merging ratios. Experiments across three different autoregressive video models demonstrate that CALM consistently outperforms uniform token merging under the same memory budget, by reducing the LPIPS to original videos by around 21% with higher video-quality scores.

1 INTRODUCTION

Autoregressive video diffusion models generate the frame-chunks sequentially, keeping a key-value (KV) cache that grows with every chunk (Wan et al., 2025a; Yin et al., 2025; Huang et al., 2025; Zhuang et al., 2026). Compressing this KV cache can improve inference efficiency by reducing both memory usage and computational cost. Token Merge has emerged as an effective training-free approach for improving model efficiency and compressing KV caches (Bolya et al., 2023; Li et al., 2024a; Wu et al., 2025; Hu et al., 2026). In autoregressive video generation, it merges redundant tokens along the temporal dimension while retaining only a few representatives at each spatial position. However, existing token merging methods typically keep the same number of representatives uniformly across all transformer layers and attention heads. We argue that this uniform strategy is suboptimal and focus on addressing this limitation to enable a more fine-grained merging strategy.

To start with, we conduct a sensitivity analysis of token merging across different transformer layers and attention heads and make three key observations. First, different layers and heads exhibit different sensitivities to merging, as measured using LPIPS against the original unmerged video. Second, layer and head sensitivities are jointly dependent, for example, the relative sensitivity ordering of two head indices can differ across layers. Third, these sensitivity patterns are consistent across different prompts. The first two patterns indicate that the layer-head merging policy cannot be reduced to a simple heuristic. Fortunately, the third observation shows that we can obtain such a policy through offline calibration and apply it across different prompts.

These observations motivate CALM, an offline-calibrated allocation of temporal KV representatives across layer–head pairs. To avoid generating videos for every candidate allocation, we measure attention-output error using dense queries and shadow KV caches, building a distortion table over candidate representative counts. An offline dynamic program minimizes the summed calibrated distortion under a fixed budget. The resulting allocation is frozen and reused across unseen prompts without retraining or per-video allocation search.

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

Our contributions can be summarized as follows:

- We reveal that sensitivity to temporal KV merging varies across layers and heads, with coupled layer–head effects that remain consistent across prompts.
- We develop a lightweight calibration proxy based on attention-output error that predicts end-to-end per-frame LPIPS without regenerating videos.
- We formulate temporal KV merging as a fixed-budget layer–head allocation problem and solve it exactly with dynamic programming.
- We show that the resulting frozen policy outperforms uniform allocation at equal budget without per-video search during inference.

2 Related Work

Autoregressive video generation. Recent large-scale diffusion transformers, such as Sora (Brooks et al., 2024), CogVideoX (Yang et al., 2025), and Wan (Wan et al., 2025a), have achieved remarkable success in video generation. Nowadays, autoregressive video diffusion has emerged as a mainstream paradigm for long-form generation, producing future frames or chunks conditioned on previously generated content. Diffusion Forcing trains causal sequence models using independently sampled noise levels for different frames (Chen et al., 2024). CausVid distills a bidirectional Wan teacher into a few-step causal generator through distribution-matching distillation (Wan et al., 2025a; Yin et al., 2024; 2025). Self-Forcing trains on self-generated rollouts to reduce the train–test discrepancy (Huang et al., 2025), while Self Gradient Forcing propagates future supervision through historical KV states (Zhuang et al., 2026). These causal models retain past KV states during generation, making cache efficiency important. Sparse Forcing learns sparse attention (Xu et al., 2026), whereas Forcing-KV performs head-wise cache pruning (Ji et al., 2026). Our method instead merges redundant cache tokens and calibrates the per-(layer, head) merge budget offline.

Token merging and KV-cache compression. Token merging reduces transformer computation by combining redundant tokens into shared representatives (Bolya et al., 2023). Attention heads also exhibit substantial specialization and differences in pruning sensitivity (Voita et al., 2019). For LLMs, KV-cache eviction methods select entries by position or attention statistics: StreamingLLM retains initial attention sinks (Xiao et al., 2024), H2O retains heavy hitters (Zhang et al., 2023), and SnapKV uses an end-of-prompt observation window (Li et al., 2024b). Merge-based methods instead aggregate cached states. KVMerger clusters similar keys (Wang et al., 2024); CaM and D2O merge evicted states into retained states (Zhang et al., 2024; Wan et al., 2025b); MiniCache shares KV states across adjacent layers (Liu et al., 2024); and WeightedKV applies attention-weighted fusion (Yuan et al., 2025). Token merging has also been applied to vision transformers and diffusion models (Bolya & Hoffman, 2023; Liang et al., 2022; Chen et al., 2023; Zeng et al., 2022; Wu et al., 2025), video transformers (Choi et al., 2024; Li et al., 2024a; Choudhury et al., 2024), and video diffusion transformers (Sun et al., 2025; Lu et al., 2025). Our work studies temporal KV-cache merging with an offline-calibrated per-(layer, head) budget.

3 Preliminaries

KV Cache in Autoregressive Video Generative Models. A key-value (KV) cache stores previously computed attention keys and values for reuse in subsequent autoregressive steps. Take Self-Forcing (Huang et al., 2025) as an example. Consider a causal video transformer with L layers, token dimension D, and H attention heads of dimension $d = D/H$. For chunk c, let F_c denote its number of latent frames and S the number of spatial tokens per frame, giving $N_c = F_c S$ input tokens $\boldsymbol{X}_l^c \in \mathbb{R}^{B\times N_c\times D}$ at layer l. At each denoising step, the layer computes

$$\boldsymbol{Q}_l^c = \mathrm{RoPE}\Big(\mathrm{Norm}_q(\boldsymbol{X}_l^c \boldsymbol{W}_l^Q)\Big), \quad \boldsymbol{K}_l^c = \mathrm{RoPE}\big(\mathrm{Norm}_k(\boldsymbol{X}_l^c \boldsymbol{W}_l^K)\big), \quad \boldsymbol{V}_l^c = \boldsymbol{X}_l^c \boldsymbol{W}_l^V, \quad (1)$$

where each tensor is reshaped to $\mathbb{R}^{B\times N_c\times H\times d}$. Before processing chunk c, layer l retains $\boldsymbol{K}_l^{<c}$ and $\boldsymbol{V}_l^{<c}$ in $\mathbb{R}^{B\times N_{<c}\times H\times d}$, where $N_{<c} = \sum_{j<c} N_j$. The current queries attend jointly to the cached

Algorithm 1 Temporal KV Token Merging

```
def temporal_kv_merge(K, V, count, R):
    # K, V: [T, S, d]; count: [T, S], ones for new tokens
    # R: representatives; attention adds log(count) to logits
    T, S, d = K.shape
    if T <= R:
        return K, V, count # keep existing counts

    K_out = K.new_empty(R, S, d)
    V_out = V.new_empty(R, S, d)
    count_out = count.new_empty(R, S)

    for s in range(S):
        K_norm = normalize(K[:, s], dim=-1)
        rep = farthest_point_sample(K_norm, R)
        sim = K_norm @ K_norm[rep].T
        group = sim.argmax(dim=-1)
        group[rep] = arange(R) # each representative owns itself

        for j in range(R):
            members = group == j
            w = count[members, s]
            K_out[j, s] = K[rep[j], s]
            V_out[j, s] = (w[:, None] * V[members, s]).sum(0) / w.sum()
            count_out[j, s] = w.sum()

    return K_out, V_out, count_out
```

context and the current chunk:

$$\boldsymbol{O}_l^c = \mathrm{Softmax}\left(\frac{\boldsymbol{Q}_l^c[\boldsymbol{K}_l^{<c}; \boldsymbol{K}_l^c]^\top}{\sqrt{d}}\right)[\boldsymbol{V}_l^{<c}; \boldsymbol{V}_l^c], \tag{2}$$

where $[\,;\,]$ denotes concatenation along the token dimension and the head dimension is omitted for clarity. During chunk denoising, the historical cache is reused while the current K/V slots are overwritten at each step. After the chunk is denoised, Self-Forcing performs one additional context forward at timestep zero and stores the resulting clean K/V for subsequent chunks. Thus, after chunk c, each layer holds a cache of shape $B \times N_{\le c} \times H \times d$ for both keys and values; the Wan-1.3B configuration used by Self-Forcing has $L = 30$, $H = 12$, $d = 128$, and $S = 1560$.

TokenMerge for KV Cache Compression. In sequential token merging, a KV cache $\boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{T \times S \times d}$ is compressed to at most r temporal representatives independently at each spatial position. We apply farthest-point sampling to the normalized keys to select r representatives, then assign each temporal token to its most similar representative by cosine similarity. Each cache entry carries the number c_i of original tokens it represents, with $c_i = 1$ for new tokens. On each merge, we sum these counts within each cluster G_j, retain the selected key, and average values weighted by c_i. To account for cluster size in attention, we add $\log g_j$ to each representative's logit:

$$\widetilde{\boldsymbol{O}} = \mathrm{Softmax}\left(\frac{\boldsymbol{Q}\widetilde{\boldsymbol{K}}^\top}{\sqrt{d}} + \log \boldsymbol{g}^\top\right)\widetilde{\boldsymbol{V}}, \qquad g_j = \sum_{i \in G_j} c_i, \tag{3}$$

where $\log \boldsymbol{g}^\top$ is broadcast across queries. Algorithm 1 summarizes the procedure.

4 Why Offline Could Help Online?

Different layers and heads have different sensitivity. To determine whether sensitivity to temporal merging differs across layers and heads, we conduct an analysis along both axes. At a fixed merge budget of $R = 6$, each run merges only one unit: either all 12 heads in one layer, yielding 30 layer configurations; or one head index across all 30 layers, yielding 12 head configurations. We evaluate each configuration on 100 prompts using mean per-frame LPIPS against the unmerged video. Layer sensitivity varies substantially: layer 0 is among the most fragile at LPIPS 0.12, whereas layer 27 is the most robust at 0.069. Layer 18 is a notable exception to this overall trend, reaching 0.148 and producing a 0.079 range across layers. Head sensitivity also varies, with mean LPIPS ranging from 0.141 for head 1 to 0.107 for head 8, a spread of 0.034. Compared with layers, the variation across heads is less pronounced but still worth noticing. Figure 1 presents both sweeps.

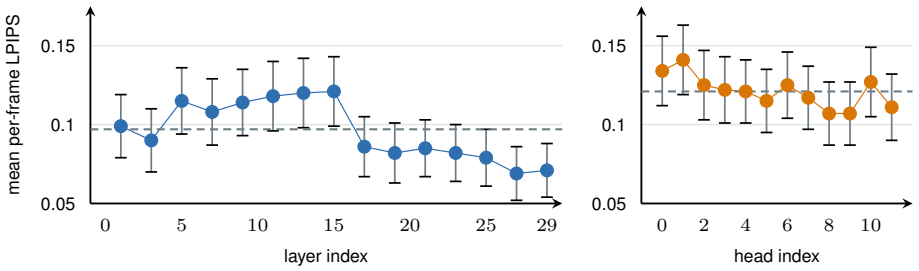


Figure 1: Mean per-frame LPIPS relative to the unmerged video with R=6. **(left)** All heads within each evaluated layer are merged; for clarity, we show every second layer. **(right)** Each head is merged across all layers. Robustness to merging varies across both layers and heads.

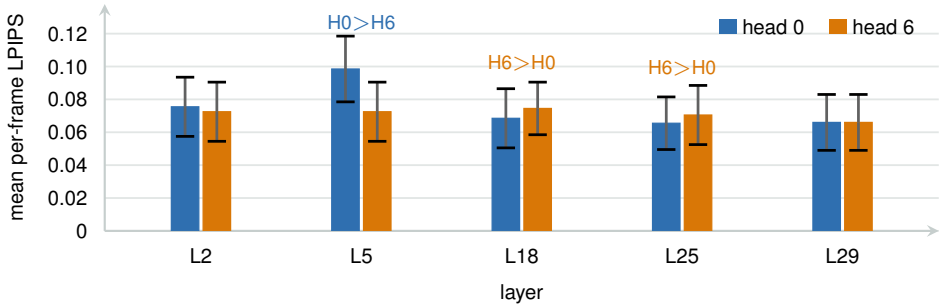


Figure 2: Sensitivity of two head indices across selected layers when one layer and head pair is merged. Points report mean LPIPS, and error bars report confidence intervals. The sensitivity ordering reverses across layers, showing that layer and head effects are jointly dependent.

Layer and head sensitivities are jointly dependent. We next examine whether layer and head sensitivities are jointly dependent rather than separable across the two axes. This distinction is important because, if they were separable, one could estimate their effects independently and combine them into a simple allocation heuristic. We disprove the independent assumption by identifying counterexamples. For each layer, we merge the same two head indices, 0 and 6, one at a time and compare their sensitivities. For example, in layer 5, head 0 is more sensitive than head 6, with LPIPS 0.098 versus 0.072. The ordering reverses in layers 18 and 25, where head 6 is more sensitive, with LPIPS 0.074 versus 0.069 and 0.070 versus 0.065, respectively. Thus, the effects of merging are interdependent and complex across layers and heads, motivating us to calibrate the entire allocation map directly. Figure 2 illustrates this joint dependency.

The patterns are consistent across prompts. Finally, we examine whether the same layers and heads exhibit similar sensitivity patterns across different prompts. We split the 100 prompts equally into calibration and test sets, obtain layer-wise and head-wise sensitivity rankings from the calibration set, and measure their Spearman correlation with the rankings of individual test prompts. We use a shuffled calibration ranking as the control. The rankings remain consistent across prompts, with median correlations of 0.69 for layers and 0.41 for heads, compared with approximately 0 for the shuffled control. This consistency allows us to estimate layer and head sensitivities in advance through offline calibration and use them to allocate the token budget more effectively during online token merging on unseen prompts. Figure 3 reports the per-prompt correlations.

5 Offline Calibration for Online Merging

CALM builds a distortion table $D \in \mathbb{R}^{L\times H\times|\mathcal{R}|}$, where $D_{l,h}(r)$ measures attention-output error when a layer–head pair retains r temporal representatives. We use this lightweight proxy instead of generating videos for every candidate allocation, then solve a fixed-budget allocation problem offline and freeze the resulting map for inference.

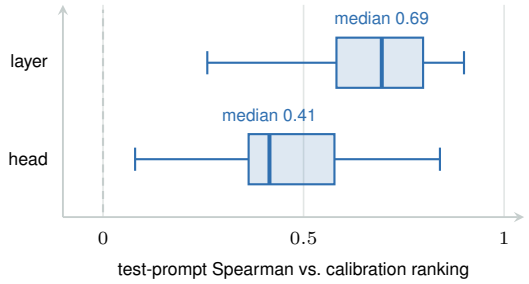


Figure 3: Sensitivity consistency across prompts. Boxes show the correlation between each test prompt and the calibration ranking, with quartiles reported for layers and heads. Both rankings remain consistent across prompts.

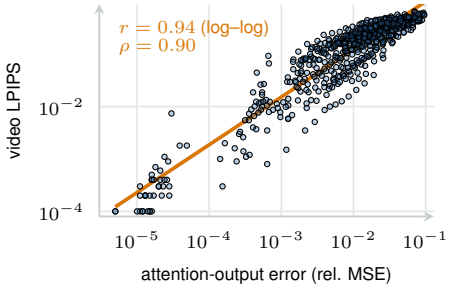


Figure 4: Validation of attention output error as a calibration proxy. The plot compares attention output error with video LPIPS across representative counts. Their strong correlation shows that the proxy predicts video distortion.

Algorithm 2 Distortion-table calibration

```
# Input: model; prompts C with fixed seeds; increasing grid R
# Output: processed distortion table D with shape [L, H, |R|]
# sum_sq reduces batch, token, channel axes, retaining heads
sq_err, sq_full = zeros(|C|, L, H, |R|), zeros(|C|, L, H, |R|)
for x_id, x in enumerate(C):
   for r_id, r in enumerate(R):
      shadow_cache = [empty_cache() for l in range(L)] # K, V, count
      for b in dense_rollout(model, x, seed=seed(x)): # merging off
         for p in passes: # 4 denoising steps + clean-context pass
            for l in range(L):
               Q, K_b, V_b, O_full = dense_attention(model, x, b, p, l)
               K, V, count = concat(shadow_cache[l], (K_b, V_b, ones))
               O_r = attention(Q, K, V, bias=log(count))
               sq_err[x_id, l, :, r_id] += sum_sq(O_r - O_full)
               sq_full[x_id, l, :, r_id] += sum_sq(O_full)
         for l in range(L): # clean K/V of chunk b persist
            K_b, V_b = clean_kv(b, l)
            K, V, count = concat(shadow_cache[l], (K_b, V_b, ones))
            shadow_cache[l] = temporal_kv_merge(K, V, count, r) # per head
D_raw = mean(sq_err / sq_full, dim=0) # per-prompt ratio, then mean
D = cummin(D_raw, dim=-1) # running minimum along R
return D
```

Attention relative MSE is the right proxy. We first verify that attention output error is strongly correlated with final video LPIPS and can therefore serve as a faithful calibration proxy. For each layer l, head h, and representative count r, we directly compute the attention output relative MSE as

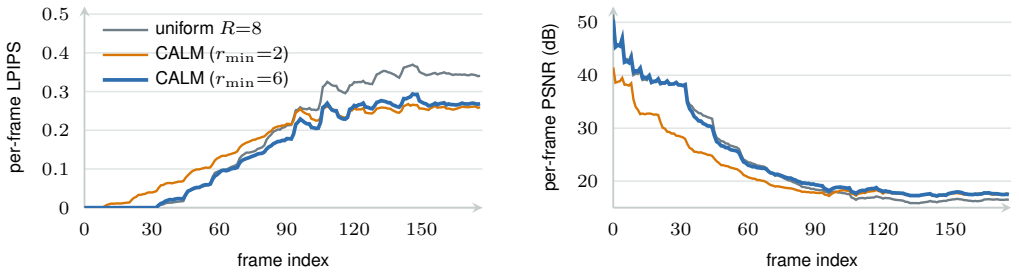
$$D^{\text{raw}}_{l,h}(r) = \mathbb{E}_x \left[\frac{\|\boldsymbol{O}^{(r)}_{l,h}(x) - \boldsymbol{O}^{\text{full}}_{l,h}(x)\|_F^2}{\|\boldsymbol{O}^{\text{full}}_{l,h}(x)\|_F^2} \right], \qquad D^{\text{raw}} \in \mathbb{R}^{L\times H\times|\mathcal{R}|}, \tag{4}$$

where the norms span the full rollout, including denoising and clean-context passes, and ratios are averaged over prompts. Across 17 representative counts on each of 40 prompts, attention-output error correlates with video LPIPS (Spearman ρ=0.90; log-space Pearson r=0.94), supporting its use as a calibration proxy (Fig. 4 and Section D).

Risk map calibration. We build the distortion table from dense-generation rollouts. For each prompt and $r \in \mathcal{R}$, dense queries attend to a shadow cache recursively merged from clean K/V, plus the current chunk's K/V; we compare outputs with dense attention. Pooling squared errors and dense-output energies over each rollout, then averaging per-prompt ratios, gives $D^{\text{raw}}_{l,h}(r)$. Each $D_{l,h}(r)$ estimates the distortion from merging head h in layer l to r representatives.

Budget Allocation with Dynamic Programming. Given the distortion map above, we aim to obtain the optimal allocation under the given budget B before inference. Formally, we solve

$$P^\star = \arg\min_P \sum_{l,h} D_{l,h}(P_{l,h}) \quad \text{s.t.} \quad \sum_{l,h} P_{l,h} = B, \qquad P_{l,h} \in \mathcal{R}_{\text{search}}, \tag{5}$$



(a) per-frame LPIPS (↓) (b) per-frame PSNR (↑)

Figure 5: Temporal quality under a fixed cache budget relative to the unmerged video. Panel (a) reports per frame LPIPS, and panel (b) reports per frame PSNR. The curves compare uniform allocation, calibrated allocation at the grid-minimum floor ($r_{\min}$=2), and calibrated allocation with $r_{\min}$=6. The lower bound preserves early frame quality while calibrated allocation improves later frames.

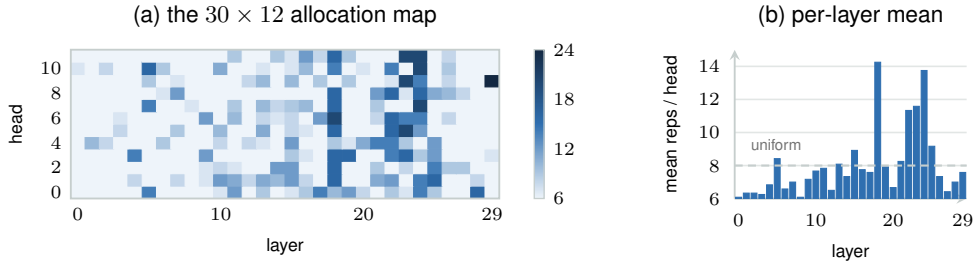


Figure 6: Calibrated allocation across layers and attention heads. Panel (a) shows the representative count for each layer and head, with darker colors indicating larger allocations. Panel (b) shows the mean representative count for each layer, with the dashed line indicating uniform allocation. The resulting allocation follows no simple ordering across layers or heads.

where D is the processed table and $\mathcal{R}_{\text{search}} = \{r \in \mathcal{R} : r_{\min} \le r \le r_{\max}\}$ keeps the measured counts within the bounds. We solve this discrete allocation problem exactly with dynamic programming. The resulting map P minimizes the predicted distortion under the given budget, as shown in Algorithm 3. We also require every unit to retain at least $r_{\min}$ representatives. This lower bound prevents overly aggressive compression, which otherwise degrades the early frames, as analyzed in Section 6.3.

6 Experiments

6.1 Setup

We evaluate on VBench prompts (Huang et al., 2024) that are separate from the 50 calibration prompts. We use R=8 and B=2880. This uniform setting reduces the logical temporal representative capacity by 5.6×. The current implementation preallocates the full KV buffer, so realized memory usage is unchanged (Section H). We use one fixed seed for each prompt across all methods and compare every result with the corresponding Dense video. We report mean LPIPS over the full clip, the first 60 frames, and the last 12 frames, together with PSNR and SSIM. We compare allocation methods using our merge operator (Table 1) and compare merge operators under a fixed uniform budget (Table 4).

6.2 Main results

Calibrated allocation vs. uniform (same operator, same budget). We fix the merge operator to FPS and use the same total budget for all allocation methods. At R=8 and B=2880, uniform merging obtains a full clip LPIPS of 0.149 against the corresponding Dense videos on 400 test prompts. CALM obtains an LPIPS of 0.117 with a lower bound of $r_{\min}$=6 and improves the result

Allocation	LPIPS	early	tail	PSNR	SSIM
Uniform (R=8)	0.149	0.015	0.258	19.3	0.778
Random	0.156	0.022	0.261	19.1	0.769
Head-only	0.146	0.014	0.255	19.4	0.781
Layer-only	0.141	0.018	0.238	19.4	0.781
Sensitivity	0.124	0.018	0.194	20.0	0.792
Greedy	0.118	0.015	0.191	20.1	0.797
CALM (ours)	**0.117**	0.015	**0.192**	**20.2**	**0.798**
CALM-warm (ours)	0.131	0.015	0.211	19.8	0.789

Table 1: Comparison of allocation strategies under the same KV budget and merge operator. LPIPS measures perceptual error over the full clip, while early and tail report the first and last temporal windows. Lower LPIPS and higher PSNR and SSIM indicate better quality. CALM achieves the best overall quality, with most of the improvement in later frames.

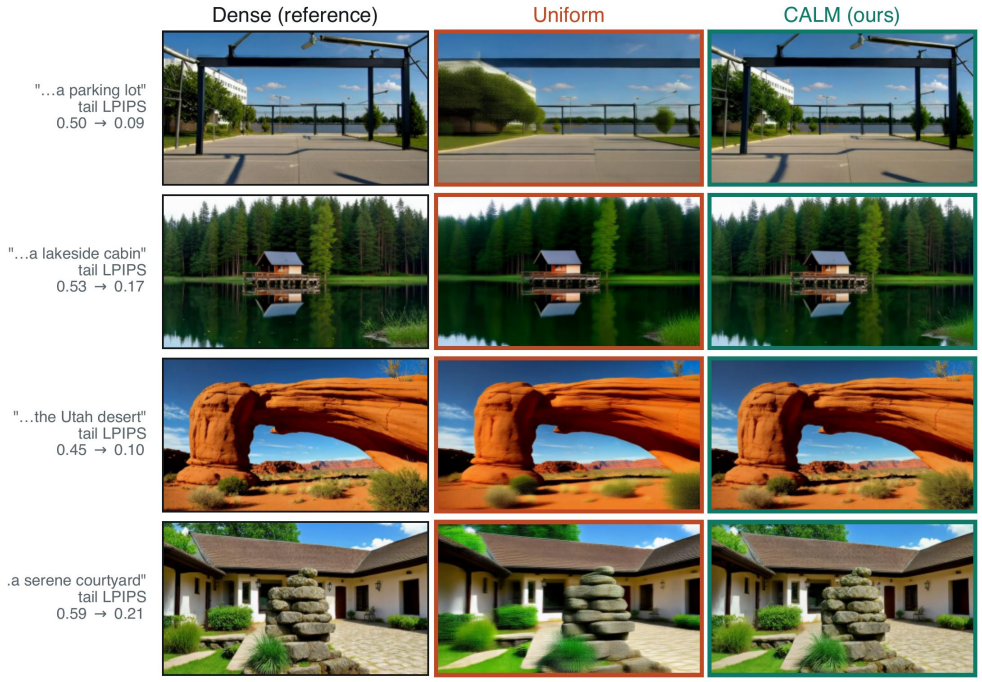


Figure 7: **Tail frame ($\approx 170/177$): uniform drifts from Dense; CALM stays faithful at equal budget.** Late-rollout examples from VBench comparing Dense, uniform allocation, and CALM; the compressed methods share matched budgets and generation settings, and each row is annotated with its tail LPIPS.

on 356 prompts (Table 1). Most of the improvement occurs in the later frames, as the qualitative comparison in Fig. 7 shows: uniform allocation visibly drifts in the late rollout while CALM stays closer to Dense (further examples in Section A). The LPIPS over the first 60 frames is 0.015 for both CALM and uniform merging. Greedy allocation obtains 0.118, and the remaining allocation methods are compared in Section 6.3.

VBench quality. On six compression-relevant VBench dimensions, CALM matches or improves on uniform merging and remains close to Dense (Table 2). Each dimension uses its designated prompt subset, with matched prompts, seeds, and generation settings across methods; sample counts are reported in the table. Five additional dimensions appear in Section G.

Cross model generalization. We apply CALM to CausVid (Yin et al., 2025), Causal Forcing (Zhu et al., 2026), and a Self Gradient Forcing variant (Zhuang et al., 2026) on the Wan back-

Dimension ↑	n	Dense	Uniform	CALM
Subject Consistency	72	0.948	0.928	**0.941**
Background Consistency	85	0.955	0.947	**0.952**
Temporal Flickering	75	0.990	0.989	**0.990**
Motion Smoothness	72	0.984	0.983	**0.984**
Imaging Quality	92	0.698	0.688	**0.697**
Aesthetic Quality	92	0.628	0.613	**0.622**
VRAM save	–	0×	5.6×	5.6×

Table 2: VBench comparison on six dimensions relevant to cache compression. n is the number of matched samples, and higher scores indicate better quality. VRAM save reports the reduction in logical cache capacity. CALM matches or outperforms uniform allocation on every dimension while remaining close to Dense.

Model	R	Uniform ↓	CALM ↓	rel.	Δ ($CALM-$Uniform)	n
CausVid	2	0.135	**0.114**	16%	$\mathbf{-0.021}$ $[-0.025, -0.018]$	30
	3	0.107	**0.100**	6%	$\mathbf{-0.007}$ $[-0.012, -0.003]$	30
	4	0.096	**0.088**	9%	$\mathbf{-0.008}$ $[-0.011, -0.005]$	30
	5	0.083	0.084	−1%	$+0.001$ $[-0.001, +0.003]$	30
	6	0.077	0.078	−1%	$+0.001$ $[-0.001, +0.003]$	30
Causal Forcing	2	0.186	0.223	−20%	$\mathbf{+0.037}$ $[+0.012, +0.065]$	29
	3	0.186	0.189	−2%	$+0.003$ $[-0.024, +0.027]$	29
	4	0.212	**0.176**	17%	$\mathbf{-0.036}$ $[-0.061, -0.011]$	29
	6	0.196	**0.171**	13%	$\mathbf{-0.025}$ $[-0.043, -0.010]$	29
SGF	2	0.105	0.107	−2%	$+0.002$ $[-0.004, +0.007]$	29
	3	0.087	0.084	3%	-0.003 $[-0.008, +0.003]$	29
	4	0.091	**0.081**	11%	$\mathbf{-0.010}$ $[-0.016, -0.004]$	29
	6	0.085	**0.079**	7%	$\mathbf{-0.006}$ $[-0.010, -0.002]$	29

Table 3: Comparison of uniform and calibrated allocation across autoregressive video generators after separate calibration for each model. R is the representative count, rel. is the relative LPIPS reduction, Δ is the paired LPIPS difference, and n is the number of prompts. Bold values indicate that the confidence interval for Δ excludes zero. The effect of calibrated allocation depends on the model and compression level.

bone. We calibrate a new distortion table and DP allocation for each model. Table 3 compares uniform and calibrated allocation under the same budget within each model. On CausVid, CALM significantly reduces LPIPS at R=2, 3, 4, with an improvement of up to 16% on all 30 prompts at R=2, but provides no significant gain at R=5, 6. For Causal Forcing and Self Gradient Forcing, significant gains occur at both R=4 and R=6. At R=2, the gain is not significant for Self Gradient Forcing; we do not read the apparent Causal Forcing result at R=2 as a genuine regression, because its uniform baseline is itself non-monotonic in R (Section F). The proxy error decreases for every model and budget, but the LPIPS improvement depends on the model and compression level (Figs. 11 and 12). We therefore report the complete budget sweep for each model without averaging across models. The same value of R does not represent the same cache size or compression difficulty across models. We also evaluate transfer of the Self Forcing allocation map to CausVid without recalibration in Section F.

6.3 Ablation and analysis

Allocation strategies. We fix the merge operator and compare six allocation strategies with uniform allocation in Table 1. The strategies are Random, Layer only, Head only, Sensitivity, Greedy, and the exact DP used by CALM. Layer only and Head only use one sensitivity axis, while Sensitivity uses one point from each distortion curve. Greedy and CALM use the complete distortion curves. Greedy obtains an LPIPS of 0.118, which is within 0.001 of CALM. These results show that the measured joint sensitivity across layers and heads is more important than the choice of optimizer.

Operator (at R=8)	LPIPS	early	tail	PSNR	SSIM
Random merging	0.339	0.082	0.534	13.2	0.582
Average pooling	0.305	0.042	0.520	13.6	0.621
AsymRnR-style (ported)	0.190	0.019	0.333	17.1	0.726
ToMe	0.188	0.018	0.331	17.2	0.728
DPC-kNN	0.187	0.017	0.336	17.3	0.731
FPS-based merging @ Uniform	**0.149**	0.015	**0.258**	**19.3**	**0.778**
Allocation map transfers to ToMe without recalibration					
ToMe @ uniform	0.188	0.018	0.331	17.2	0.728
ToMe @ CALM map	0.213	0.025	0.358	16.3	0.701

Table 4: Comparison of temporal KV merge operators under the same uniform budget. LPIPS measures perceptual error over the full clip, while early and tail report the first and last temporal windows. The bottom rows evaluate direct transfer of the calibrated allocation map to ToMe. FPS achieves the best quality among the uniform operators, while direct transfer to ToMe does not improve its results.

Comparison with adapted merge implementations. Table 4 compares our method with adapted implementations of other merge operators under the same uniform budget. These implementations use our pipeline rather than the original systems. They differ from our method in grouping and key aggregation. At the uniform budget, FPS obtains an LPIPS of 0.149, ToMe obtains 0.188 (Bolya et al., 2023), DPC kNN obtains 0.187, and the adapted AsymRnR method obtains 0.190 (Sun et al., 2025). Average pooling and random merging produce larger errors. Full CALM combines FPS with calibrated allocation and obtains 0.117. This comparison does not isolate the allocation gain because both the operator and allocation are different.

Proxy validation. CALM selects the allocation map by minimizing the summed attention error $S(P)=\sum_{l,h} D_{l,h}(P_{l,h})$. Since the DP optimizes $S(P)$, the LPIPS results in Table 1 provide the main evidence for the allocation benefit; Section D confirms that $S(P)$ nonetheless tracks video LPIPS across compression levels and across same-budget allocation maps.

Floors, temporal behavior, and budget sensitivity. The lower bound $r_{\min}$ specifies the minimum number of temporal representatives per layer–head. Increasing it restricts allocation flexibility without increasing the total budget. We compare $r_{\min}$=2 and $r_{\min}$=6 on the same held-out subset, using the same merge operator and calibration table and re-solving the DP at B=2880. With $r_{\min}$=2, CALM improves full-clip LPIPS over uniform allocation (0.058 vs. 0.066) but increases first-60-frame LPIPS (0.020 vs. 0.002). Raising the floor to 6 substantially reduces this early degradation (0.003) and further lowers full-clip LPIPS to 0.043 (Fig. 5). The budget sweep in Section E.1 examines how allocation gains vary with compression strength.

The calibrated map is irregular. The calibrated map is not monotonic across layer or head indices (Fig. 6). The remaining budget follows no simple ordering across layers or heads. This irregular map further demonstrates the need for our method because an optimal budget allocation is complex and difficult to explain with simple heuristics. It therefore requires direct calibration of the entire layer and head map.

7 Conclusion

We introduce CALM, which calibrates layer–head sensitivity using attention-output error and solves a fixed-budget allocation offline. The frozen allocation improves fidelity over uniform merging on unseen prompts without per-video search. Cross-model experiments show that its benefits depend on the generator and compression level. The compression is logical: our implementation keeps B representatives but pre-allocates the full KV buffer, so the memory ratio is a bound rather than a realized latency or memory win (Table 9).

AI USE STATEMENT

Generative AI assisted with experimental coding, idea exploration, and edits to the writing. The tools used included GPT-Codex and Claude. We checked the resulting code and text for correctness, accuracy, and suitability. The authors are fully responsible for the final manuscript, its claims, and all accompanying artifacts, including material developed with AI assistance.

REPRODUCIBILITY STATEMENT

Our experiments are reproducible, and we will release the code upon acceptance.

REFERENCES

Daniel Bolya and Judy Hoffman. Token merging for fast stable diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2023.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In *International Conference on Learning Representations*, 2023.

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. `https://openai.com/index/video-generation-models-as-world-simulators/`, 2024.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. In *Advances in Neural Information Processing Systems*, 2024.

Mengzhao Chen, Wenqi Shao, Peng Xu, Mingbao Lin, Kaipeng Zhang, Fei Chao, Rongrong Ji, Yu Qiao, and Ping Luo. DiffRate: Differentiable compression rate for efficient vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Joonmyung Choi, Sanghyeok Lee, Jaewon Chu, Minhyuk Choi, and Hyunwoo J. Kim. vid-TLDR: Training free token merging for light-weight video transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Rohan Choudhury, Guanglei Zhu, Sihan Liu, Koichiro Niinuma, Kris M. Kitani, and László Jeni. Don't look twice: Faster video transformers with run-length tokenization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Jie Hu, Shengnan Wang, Yutong He, Ping Gong, Jiawei Yi, Juncheng Zhang, Youhui Bai, Renhai Chen, Gong Zhang, Cheng Li, and Kun Yuan. CentroidKV: Efficient long-context LLM inference via KV cache clustering. *Transactions on Machine Learning Research*, 2026.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. In *Advances in Neural Information Processing Systems*, 2025.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Yicheng Ji, Zhizhou Zhong, Jun Zhang, Qin Yang, XiTai Jin, Ying Qin, Wenhan Luo, Shuiyang Mao, Wei Liu, and Huan Li. Forcing-kv: Hybrid kv cache compression for efficient autoregressive video diffusion models. *arXiv preprint arXiv:2605.09681*, 2026.

Xirui Li, Chao Ma, Xiaokang Yang, and Ming-Hsuan Yang. VidToMe: Video token merging for zero-shot video editing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024a.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*, 2024b.

Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not all patches are what you need: Expediting vision transformers via token reorganizations. In *International Conference on Learning Representations (ICLR)*, 2022.

Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. MiniCache: KV cache compression in depth dimension for large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Wenbo Lu, Shaoyi Zheng, Yuxuan Xia, and Shengjie Wang. ToMA: Token merge with attention for diffusion models. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

Wenhao Sun, Rong-Cheng Tu, Jingyi Liao, Zhao Jin, and Dacheng Tao. AsymRnR: Video diffusion transformers acceleration with asymmetric reduction and restoration. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*, 2019.

Team Wan, Ang Wang, Baole Ai, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025a.

Zhongwei Wan, Xinjian Wu, Yu Zhang, Yi Xin, Chaofan Tao, Zhihong Zhu, Xin Wang, Siqi Luo, Jing Xiong, Longyue Wang, and Mi Zhang. D2O: Dynamic discriminative operations for efficient long-context inference of large language models. In *International Conference on Learning Representations (ICLR)*, 2025b.

Zheng Wang, Boxiao Jin, Zhongzhi Yu, and Minjia Zhang. Model tells you where to merge: Adaptive KV cache merging for LLMs on long-context tasks, 2024.

Haoyu Wu, Jingyi Xu, Hieu Le, and Dimitris Samaras. Importance-based token merging for efficient image and video generation. In *IEEE/CVF International Conference on Computer Vision*, 2025.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024.

Boxun Xu, Yuming Du, Zichang Liu, Siyu Yang, Ziyang Jiang, Siqi Yan, Rajasi Saha, Albert Pumarola, Wenchen Wang, and Peng Li. Sparse forcing: Native trainable sparse attention for real-time autoregressive diffusion video generation. *arXiv preprint arXiv:2604.21221*, 2026.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Yuxuan Zhang, Weihan Wang, Yean Cheng, Bin Xu, Xiaotao Gu, Yuxiao Dong, and Jie Tang. CogVideoX: Text-to-video diffusion models with an expert transformer. In *International Conference on Learning Representations*, 2025.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Tianwei Yin, Qiang Zhang, Richard Zhang, William T Freeman, Frédo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast autoregressive video diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Jian Yuan, Ziwei He, Haoli Bai, Jingwen Leng, and Bo Jiang. WeightedKV: Attention scores weighted key-value cache merging for large language models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025.

Wang Zeng, Sheng Jin, Wentao Liu, Chen Qian, Ping Luo, Wanli Ouyang, and Xiaogang Wang. Not all tokens are equal: Human-centric visual analysis via token clustering transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Yuxin Zhang, Yuxuan Du, Gen Luo, Yunshan Zhong, Zhenyu Zhang, Shiwei Liu, and Rongrong Ji. CaM: Cache merging for memory-efficient LLMs inference. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, 2023.

Hongzhou Zhu, Min Zhao, Guande He, Hang Su, Chongxuan Li, and Jun Zhu. Causal forcing: Autoregressive diffusion distillation done right for high-quality real-time interactive video generation. In *International Conference on Machine Learning*, 2026.

Junhao Zhuang, Shiyi Zhang, Yuxuan Bian, Yaowei Li, Yawen Luo, Yijun Liu, Weiyang Jin, Songchun Zhang, Xianglong He, Xuying Zhang, Haoran Li, Haoyang Huang, Zeyue Xue, and Nan Duan. Self gradient forcing: Native long video extrapolation. *arXiv preprint arXiv:2607.20368*, 2026.

A QUALITATIVE RESULTS

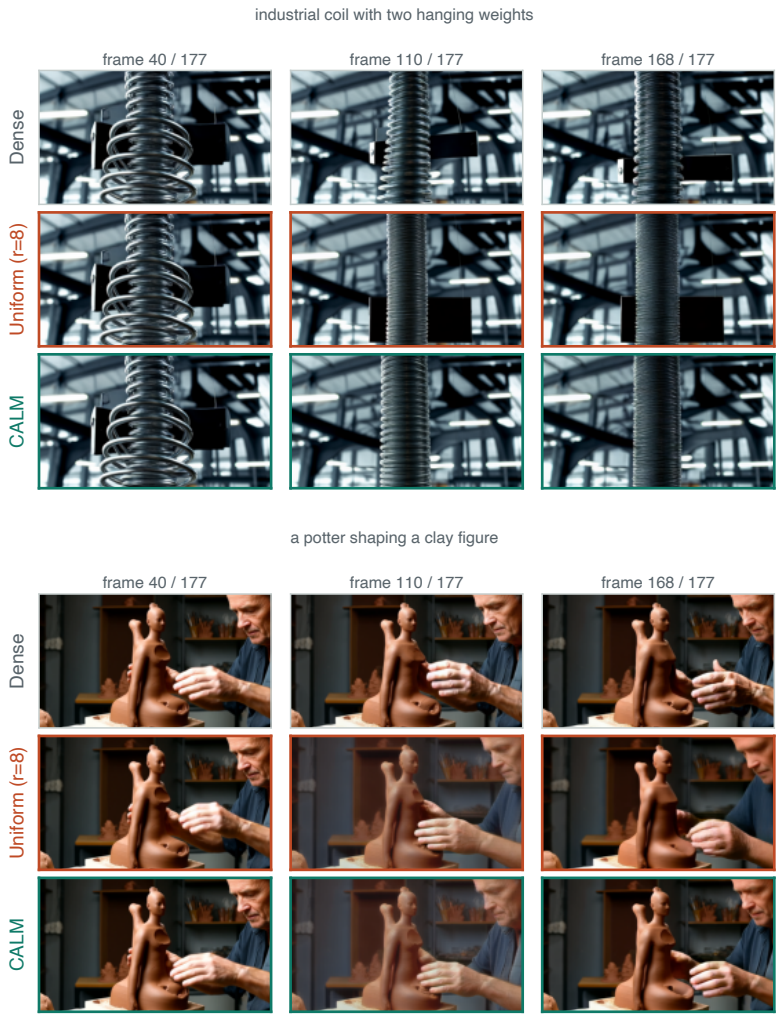


Figure 8: Where the difference emerges: two held-out prompts (top and bottom blocks), each shown at early, middle, and late rollout frames (columns). Within a block the rows are the Dense reference, uniform allocation, and CALM; all three share the prompt and seed, and the two compressed rows share the representative budget. The arms are indistinguishable early and separate in the later frames, where CALM stays closer to Dense.

We collect additional video comparisons in this section, extending the qualitative result shown in the main text (Fig. 7). In every panel the three arms are the uncompressed Dense reference, uniform allocation, and CALM. The two compressed arms share the prompt, seed, generation settings, and *representative budget*, so any difference between them reflects how that budget is distributed; Dense is the no-compression reference and is not budget-matched. Frames are sampled at the rollout positions marked in each figure, and because the differences are temporal and accumulate over the rollout, still frames understate the tail-LPIPS gap that the numbers report. The rows are illustrative examples from the held-out and VBench-400 test prompts, not a uniform random sample.

Figure 7 shows late-rollout frames on VBench-400 test prompts (Table 1) annotated with each row's tail LPIPS; Fig. 8 tracks two held-out prompts from early to late frames to show when the gap emerges; Fig. 9 extends the test-set comparison to nine further prompts; and Fig. 10 adds eight further held-out prompts (disjoint from the calibration set) at a late frame.



Figure 9: Additional prompts from the VBench-400 test set at a late rollout frame, spanning objects, animals, architecture, and landscapes. Columns are the Dense reference, uniform allocation, and CALM; the two compressed columns share the representative budget and seed, and Dense is the uncompressed reference. Rows are annotated with the tail-LPIPS gap between uniform and CALM (per row) and span clear improvements to near-ties.

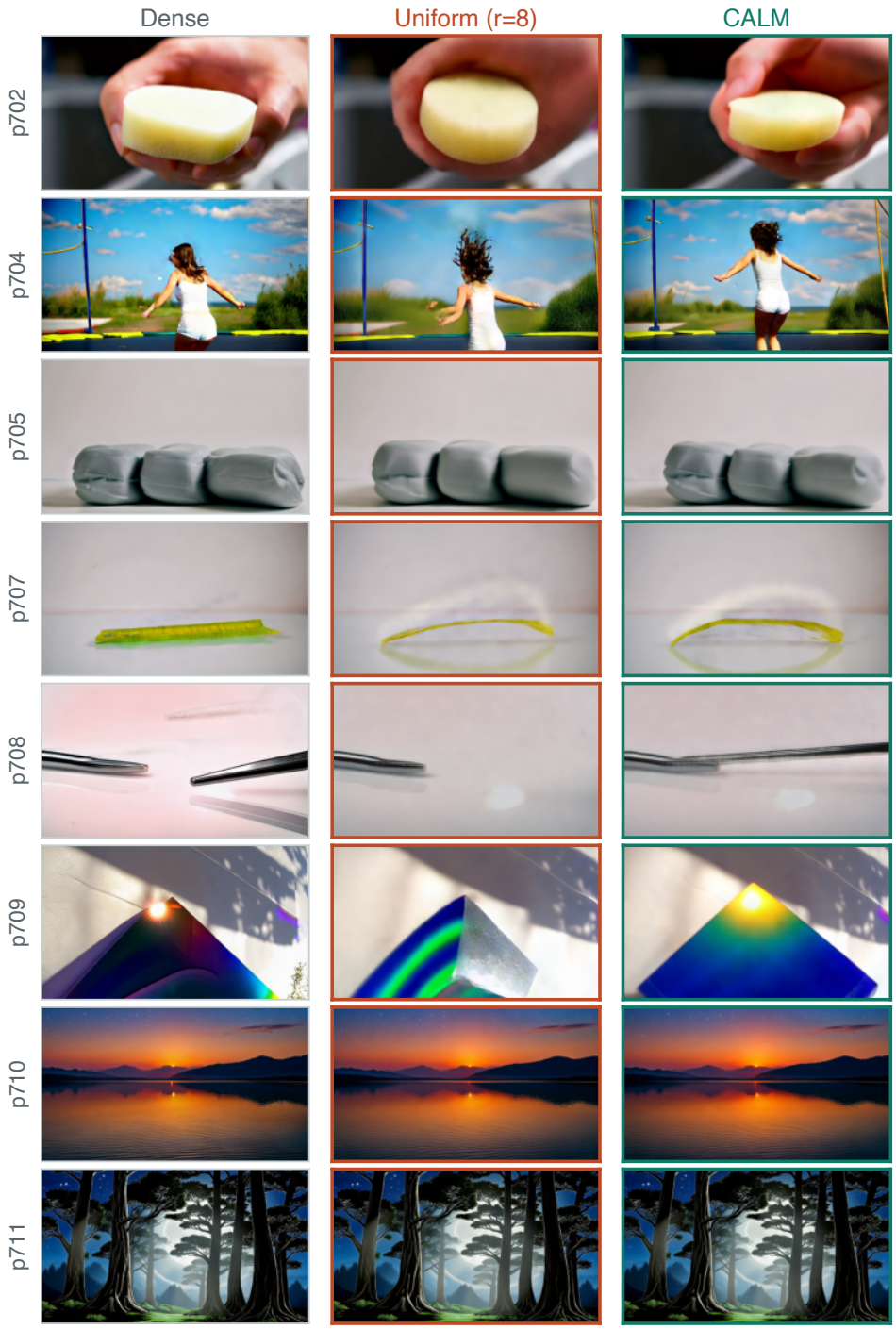


Figure 10: Eight further held-out prompts (disjoint from the calibration set) at a late rollout frame, covering close-up objects, human motion, landscapes, and stylized scenes. Columns are the Dense reference, uniform allocation, and CALM; the two compressed columns share the representative budget and seed, and Dense is the uncompressed reference. These are illustrative cases where uniform allocation degrades visibly; on them CALM stays closer to Dense.

B EXPERIMENTAL PROTOCOL

Models and generation. Our main experiments use Self-Forcing on the Wan2.1-T2V-1.3B backbone; the cross-model study (Section F) adds CausVid, Causal Forcing, and a Self-Gradient-Forcing variant. Within any comparison all arms share the prompt, seed, and sampler settings and use the same mask-aware SDPA attention backend, and each comparison varies a single factor—the allocation map, or the merge operator (which also changes grouping and key aggregation)—while holding the others fixed. Dense denotes the uncompressed reference.

Prompt sets. We use 50 prompts for calibration (Section C) and evaluate the main allocation table (Table 1) on a 400-prompt VBench subset disjoint from the calibration set. The proxy preregistration (Section D) uses a separate set of 20 prompts, split into 12 for choosing the error definition and 8 for confirmation. The qualitative galleries (Section A) use held-out prompts as noted per figure. Cross-model evaluations use each model's own held-out prompts (30 for CausVid, 29 for Causal Forcing and SGF).

Metrics. Unless stated otherwise, quality is per-frame LPIPS against the Dense reference over the full clip, with the first- and last-window (early/tail) LPIPS reported separately because the error is end-loaded; we also report PSNR and SSIM. VBench dimensions are scored with the official VBench scorers on their respective prompt subsets (Section G).

Statistics. Cross-arm comparisons are paired over shared prompts. We report the mean paired difference against uniform merging, its 95% confidence interval from a paired bootstrap over prompts, and the number of prompts on which the calibrated map wins; a difference is called significant when its confidence interval excludes zero.

C CALIBRATION AND ALLOCATION DETAILS

We give the exact procedure that turns a set of calibration prompts and a frozen model into the 30×12 allocation map; Algorithms 2 and 3 give the pseudocode. The calibration fixes the prompts, seeds, and implementation settings and sweeps a fixed grid of representative counts.

Probing one representative count. For a single representative count r, we generate a calibration prompt *densely* (merging off) and, alongside the dense cache, maintain a shadow KV cache: a recursively merged copy of the dense history, built with our operator (FPS on the keys, count-weighted mean value, $\log$-count bias). After each chunk's clean-context pass, that chunk's clean K/V join the shadow KV cache with unit counts and it is merged again to r, so counts carry across chunks exactly as in merged generation. At each attention call—15 blocks $\times$ 5 passes (four denoising steps plus the clean-context pass)—we re-run the block's own query against the shadow KV cache plus the current chunk's dense K/V and compare the resulting attention output $O^{(r)}_{l,h}$ against the dense output $O^{\text{full}}_{l,h}$. Because the shadow KV cache holds all $L \times H$ heads, one dense generation measures the merge error of all 360 heads at that r simultaneously. No budget enters here: each (l, h, r) is probed independently; the budget B only enters the allocation step.

Filling one prompt's table. The error of a head is its *relative* MSE, Eq. (4): the squared attention-output error summed over batch, tokens, channels, and the whole rollout (every block and pass), divided by the squared dense output summed the same way. The implementation clamps the denominator to at least 10^{-30} for numerical stability; this safeguard is omitted from Eq. (4). Sweeping r over the grid $\mathcal{R}$ (17 values, 2 to 41, denser near the uniform r=8) fills one prompt's table $E[l, h, r]$ at a cost of $|\mathcal{R}|$ dense generations per prompt.

Integrating prompts and solving. We average the per-prompt relative MSE over the calibration set, $D^{\text{raw}}[l, h, r] = \frac{1}{|\mathcal{C}|} \sum_x E_x[l, h, r]$, and take a running minimum along the full measured grid $\mathcal{R}$ so the *table used for optimization*, D, is monotone in r (a smoothing of the calibration table, not a guarantee about the error of any individual generation). Averaging ratios rather than pooling numerators and denominators weights every prompt equally. We use 50 calibration prompts; the

Algorithm 3 Budget Allocation

```
# Input: processed distortion table D with shape [L, H, |R|] (Alg. 2);
# measured grid R; bounds r_min, r_max; budget B
# Output: integer allocation map P with shape [L, H], sum(P) = B

# Step 1: keep admissible counts, flatten layer-head pairs
keep = [j for j, r in enumerate(R) if r_min <= r <= r_max]
R = [R[j] for j in keep] # R_search
L, H = D.shape[:2]
N = L * H
D = reshape(D[:, :, keep], (N, len(R)))
# Step 2: fill the DP table
DP = full((N + 1, B + 1), infinity)
DP[0, 0] = 0
choice = zeros((N + 1, B + 1), dtype=int)
for i in range(1, N + 1):
   for t in range(min(R), B + 1):
      DP[i, t], choice[i, t] = min(
         (DP[i-1, t-r] + D[i-1, j], r)
         for j, r in enumerate(R) if r <= t)
# Step 3: recover the allocation map
P = zeros(N, dtype=int)
t = B
for i in reversed(range(1, N + 1)):
   P[i-1] = choice[i, t]
   t -= P[i-1]

return reshape(P, (L, H))
```

resulting DP allocation reproduces our deployed map, and the sensitivity ranking is stable across prompts (Fig. 3). The dynamic program of Algorithm 3 then keeps only the columns of D in $\mathcal{R}_{\text{search}} = \{r \in \mathcal{R} : r_{\min} \le r \le r_{\max}\}$, after the running minimum, and allocates the budget B over them; the frozen map is applied to all unseen prompts.

D PROXY EVALUATION

Re-generating a full video to score every candidate merge is the bottleneck of any allocation search: across many layers, heads, and admissible representative counts, the space of same-budget allocation maps is far too large to explore one video at a time. If instead a cheap, per-(layer, head) attention-output error ranks merge configurations in the same order that full-video LPIPS does, the search can be run entirely offline, evaluating each candidate from cached attention statistics rather than a generated video. We use the per-(layer, head) relative mean-squared error between the merged and dense attention outputs, Eq. (4), so that a single scalar per (layer, head) summarizes a candidate.

We fixed the analysis before inspecting any correlation. The decision rule was pre-registered as a per-prompt Spearman correlation of at least 0.8 between attention error and LPIPS. We partitioned the 20 prompts into 12 "choosing" prompts, used only to select the error definition, and 8 held-out "confirmation" prompts, reserved to test that choice. We also pre-registered 17 admissible representative counts, spaced more densely near the uniform setting $r{=}8$.

Two analyses probe the proxy, and they answer different questions on different samples and units; we report them separately rather than pooling them into one summary. The first, within-prompt analysis correlates, for each of the 20 pre-registered prompts, the attention error against video LPIPS across the 17 *uniform* representative counts; because the total budget varies with the count, it tests only whether the proxy tracks LPIPS as overall compression strength changes, not whether it ranks allocations at a fixed budget. The median per-prompt Spearman correlation is 1.00, the minimum is 0.96, and all 20 prompts clear the pre-registered 0.8 threshold. Separately, Fig. 4 pools 680 prompt–count pairs into a single log–log correlation (Spearman $\rho{=}0.90$, Pearson $r{=}0.94$); this pooled value is lower than the per-prompt median; the pooled and per-prompt statistics use different scopes and are not directly comparable. The second, same-budget analysis is the one that bears directly on the allocation search: it fixes the total budget at $B{=}2880$ and asks whether the offline score orders *different allocations of the same budget* the way measured LPIPS does. Across seven static, equal-budget allocations—uniform, the DP map, and five variants, all at $B{=}2880$ with the same merge operator and bounds, evaluated on the same 102 common prompts—the summed score $S(P){=}\sum_{l,h} D_{l,h}(P_{l,h})$ correlates strongly with mean video LPIPS (Spearman $\rho{=}0.96$, Pear-

son r=0.98; Table 5). The single rank inversion is the near-tie between the DP and greedy maps, whose scores differ by about 10^{-4}. The 102 prompts are those on which all seven maps completed generation (the intersection of the per-map evaluation sets). The distortion table and its running-minimum processing, the bounds, and all seven candidate maps were fixed by the main calibration before these generations were scored, so no hyperparameter was selected on this comparison; it is a post-hoc validation of the proxy rather than a tuning step.

Allocation	$S(P)$	mean LPIPS ↓
CALM (DP)	4.4550	0.1684
Greedy	4.4551	0.1683
Sensitivity	4.8466	0.1789
Layer-only	5.6131	0.1999
Head-only	6.7252	0.2095
Uniform (R=8)	6.7315	0.2143
Random	7.5905	0.2223

Table 5: Offline calibration score $S(P) = \sum_{l,h} D_{l,h}(P_{l,h})$ versus end-to-end video LPIPS for seven static, equal-budget allocations of the same budget (B=2880), evaluated with the same merge operator and bounds on the same 102 common prompts. The score orders the allocations in nearly the same order as measured LPIPS (Spearman ρ=0.96, Pearson r=0.98). The single inversion is the near-tie between the CALM (DP) and Greedy maps, whose scores differ by about 10^{-4}; extra decimals are shown to expose it.

On the within-prompt uniform sweep, all 18 candidate error definitions — relative MSE, L_1, or cosine distance, under three poolings, computed per head or pooled across heads — yield the same verdict, so that conclusion does not hinge on a lucky choice of "error"; we did not repeat the same-budget test for all 18.

Two complications warrant control. First, a kernel-noise floor: a control cell that passes through the merge mask with every cluster size equal to one has a $\log 1$=0 bias and should therefore reproduce the un-merged video exactly, yet a small residual LPIPS remains. We attribute this residual to nondeterminism and kernel/backend differences along the merge path rather than treating it as a single global measurement floor; we align the attention backend across arms and read correlations only where the measured differences exceed this residual. Second, saturation: under very aggressive merging LPIPS stops discriminating between configurations, so we read correlations only in the regime where LPIPS remains informative. Neither complication alters the conclusion that attention-output error is strongly correlated with video LPIPS on the evaluated budgets and candidate maps, making it a usable offline proxy for the allocation search.

E Additional ablations and comparisons

All baselines share the locked budget B=2880 (R=8); every comparison therefore redistributes the same KV tokens rather than spending more cache.

Candidate counts. The distortion table is measured on a grid of 17 representative counts $\mathcal{R}_{\text{meas}} = \{2, 3, \ldots, 12, 14, 16, 20, 24, 32, 41\}$, denser near the uniform r=8. The main-table allocators search the sub-grid $\mathcal{R}_{\text{search}} = \{r \in \mathcal{R}_{\text{meas}} : r_{\min} \le r \le r_{\max}\}$ with $r_{\min}$=6 and $r_{\max}$=24. The cap binds—CALM reaches r=24—so 32 and 41 are measured but unused at this budget; whether relaxing the cap would help is left open. Continuous or off-grid proposals (Sensitivity, Random) are rounded onto $\mathcal{R}_{\text{search}}$ and repaired to $\sum P$=B.

Allocation baselines (merge operator fixed to ours).

- **Uniform.** $P_{l,h}$=8 everywhere.
- **Random.** From uniform, apply budget-preserving ±1 grid swaps (snapped to $\mathcal{R}_{\text{search}}$ and re-paired to budget) to produce a mixed random reallocation, averaged over random seeds.

- **Layer-only.** All heads in a layer share one count R_l, obtained by the same fixed-budget DP over the 30 head-averaged curves $\bar{D}_l(r) = \frac{1}{H}\sum_h D_{l,h}(r)$, spending $\sum_l HR_l{=}B$.
- **Head-only.** Symmetric: each head index shares one count R_h across layers, from the DP over the 12 layer-averaged curves.
- **Sensitivity.** $P_{l,h} \propto D_{l,h}(8)$ (single-point distortion at the uniform count), renormalized to $\sum P{=}B$ and snapped to the grid; a single-point heuristic that ignores curve shape.
- **Greedy.** Start every head at $r_{\min}$ and repeatedly advance the head with the largest marginal distortion drop *per added representative* to its next grid point, until B is spent; converges near the DP optimum.
- **CALM / CALM-warm.** Algorithms 2 and 3; CALM-warm additionally gates on rollout position (Section E.2).

Merge operators (budget fixed to uniform $R{=}8$). Every operator merges each head's T temporal tokens to R representatives and keeps the same count-weighted mean *value* and $\log$-count attention bias; they differ in the grouping and, for the baselines, the *key* aggregation (ours keeps a representative key, the baselines average keys). Table 4 is thus a comparison of full operators, not of grouping alone.

- **FPS-based merging (ours).** Farthest-point sampling on L_2-normalized keys picks R representatives; each token joins its nearest.
- **Average pooling.** Contiguous strided groups of $\lceil T/R \rceil$ over the actual input length T.
- **Random merging.** A fixed random assignment of tokens to R groups (ToMe's own control).
- **ToMe.** Bipartite soft matching (Bolya et al., 2023): tokens split into alternating source/destination sets, each source scored against destinations by key similarity, and the highest-scoring source→destination edges merged into their destination until R tokens remain.
- **DPC-kNN.** Density-peak clustering (Zeng et al., 2022): each token's density ρ is its k-NN key similarity and δ its distance to the nearest higher-density token; the R tokens with largest $\rho{\cdot}\delta$ are the peaks, and every token joins its nearest peak.
- **AsymRnR-style.** We port the reduction rule of AsymRnR (Sun et al., 2025): $R/2$ fixed evenly-spaced destination anchors plus $R/2$ adaptively kept, redundancy-ranked source tokens. Its official code targets diffusers video DiTs (CogVideoX, Mochi, HunyuanVideo) and does not run on the autoregressive Wan cache, so we re-implement the rule in our pipeline and label it *AsymRnR-style* to distinguish it from the official method.

Allocation-map transfer to another operator. *ToMe@CALM* applies the ToMe grouping under the frozen CALM budget P—each head (l, h) merges to $P_{l,h}$ ToMe representatives, surplus slots masked out—*transferring* the FPS-calibrated map without re-fitting it to ToMe. It tests whether this map transfers directly to another operator. It does not: under the faithful ToMe operator the transferred map hurts both full-clip (0.213 vs. 0.188) and tail (0.358 vs. 0.331). We conclude only that this FPS-calibrated map does not transfer directly to ToMe; whether a ToMe-calibrated map would help is a separate question we do not address.

E.1 BUDGET SWEEP: WHERE CALIBRATION PAYS OFF

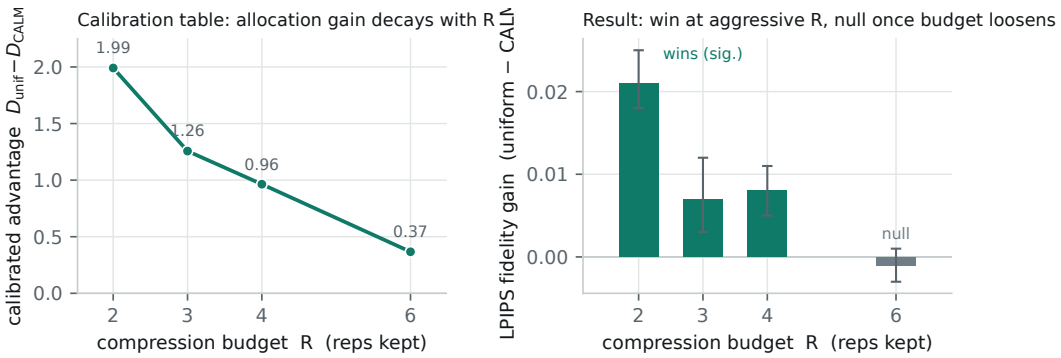


Figure 11: Comparison between calibrated attention error and video LPIPS across cache budgets on CausVid. The left panel shows the attention error reduction of CALM relative to uniform allocation. The right panel shows the corresponding LPIPS reduction with confidence intervals. The proxy improves across all budgets, while the video quality benefit depends on compression strength.

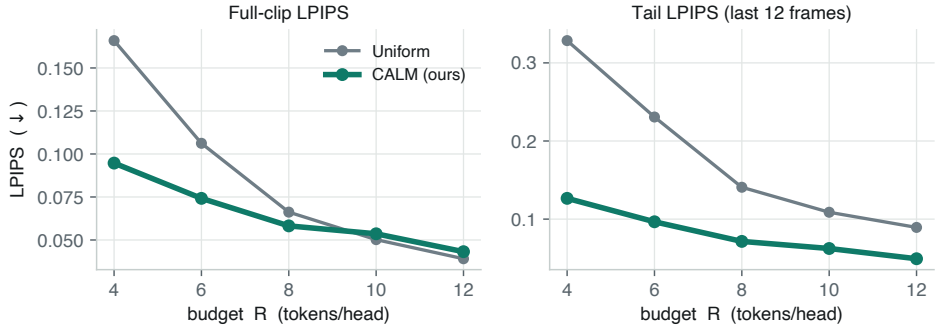


Figure 12: Comparison of uniform and calibrated allocation across representative budgets. The left panel shows full clip LPIPS, and the right panel shows LPIPS over the final temporal window. Lower values indicate better quality. The benefit of calibrated allocation is larger under tighter budgets and remains visible in later frames.

We sweep the per-head budget $R \in \{4, 6, 8, 10, 12\}$ for both uniform and CALM allocation (merge operator fixed to ours) on a 30-prompt held-out subset (Fig. 12); because the sweep includes budgets as low as R=4, it uses the grid-minimum floor $r_{\min}$=2 and the full measured grid up to $r_{\max}$=41, rather than the main table's $r_{\min}$=6 and $r_{\max}$=24; the R=8 point uses this same setting. CALM's advantage is concentrated where it matters for deployment. At tight budgets the full-clip gap is large (R=4: 0.095 vs. 0.166) and it closes as the cache loosens; on the tail CALM wins at *every* budget, cutting the error by more than half at R=4 (0.127 vs. 0.329). At loose budgets ($R \geq 10$) uniform is already near-lossless on full-clip and the two differ by only a small mean gap, but the long-rollout benefit of calibration persists—consistent with the main-text finding that calibration pays off in the constrained, long-rollout regime.

E.2 CALM-WARM: TEMPORAL GATING

CALM-warm merges uniformly (R=8) while the rollout is inside the model's native 81-pixel (21-latent) window, then applies the calibrated map. The gate is keyed on the cumulative *raw* latent-frame position, not the compressed cache length—uniform merging pins the latter near 8, so a "compressed length" trigger would never fire. Merging happens at each block's end, after its clean KV is written: blocks whose start position is < 21 merge uniformly; the block at position 21 (frames 21–23) is still generated against uniform history, the first calibrated merge fires immediately after it, and the first block that *reads* the calibrated cache begins at latent frame 24. The early representatives

are not frozen—they keep participating in later merges under the calibrated quota—so CALM-warm is a genuinely dynamic policy, not the global map applied from frame one.

F CROSS-MODEL EVALUATION DETAILS

F.1 NATIVE CALIBRATION ON EACH MODEL

The cross-model table in the main text (Table 3) calibrates CALM *natively* on each generator: for every model we build a model-specific distortion table from its own dense rollouts, solve the budget-constrained dynamic program, freeze the resulting map, and compare it against uniform merging on that model's own held-out prompts. Table 6 summarizes the protocol. CausVid (Yin et al., 2025) uses the released distilled checkpoint with 48-frame rollouts and budgets $R \in \{2, 3, 4, 5, 6\}$ evaluated on 30 prompts. Causal Forcing (Zhu et al., 2026) and the Self-Gradient-Forcing variant (Zhuang et al., 2026) both use the framewise autoregressive architecture with a 21-latent KV window and 4 sink frames, 45-frame rollouts, and budgets $R \in \{2, 3, 4, 6\}$, and both are evaluated on the same 29-prompt set, so n=29 for each. Causal Forcing uses the framewise `ar_diffusion` initialization checkpoint, whereas SGF (marked † in Table 3) uses the self-gradient-forcing-trained EMA checkpoint. Paired 95% confidence intervals are bootstraps over the shared prompts. With this initialization checkpoint the Causal Forcing uniform baseline is non-monotonic in R (0.186, 0.186, 0.212, 0.196 at R=2, 3, 4, 6): raising the budget from R=2 to R=4 does not lower LPIPS, which cannot hold for a stable generator at fixed prompts and seed and instead reflects the higher generation variance of this checkpoint at n=29. We therefore do not treat the R=2 Causal Forcing comparison as a reliable regression, and read the better-behaved R=4, 6 points—where uniform LPIPS decreases with budget as expected—as the more indicative result for this model. For each target model we re-derive the candidate representative grid and bounds from its own KV history rather than copying Self-Forcing's; the three framewise generators (CausVid, Causal Forcing, and SGF) share the grid $\{1, 2, 3, 4, 6, 8, 11\}$ with $r_{\min}$=1 and $r_{\max}$=11. The budget is the summed representative count over the 30×12 units—720, 1080, 1440, 2160, and 2880 for $R \in \{2, 3, 4, 6, 8\}$, each equal to $R \times 360$—while the sink, anchor, and recent frames are kept separately and are not counted against it. Because $r_{\min}$=1, an average of R=2 still admits a non-uniform map: the dynamic program drops some heads to a single representative and spends the freed budget on more sensitive ones, so the native R=2 maps range from 1 up to 8–11 representatives with mean exactly 2; for $R \geq 3$ a per-budget floor keeps the minimum from exceeding the average.

	CausVid	Causal Forcing	SGF†
Checkpoint	released distilled	framewise `ar_diffusion`	framewise self-grad. (EMA)
Rollout (latent frames)	48	45	45
KV window / sink	persistent	21 / 4	21 / 4
Budgets R	2–6	2, 3, 4, 6	2, 3, 4, 6
Eval prompts n	30	29	29
CI	paired bootstrap	paired bootstrap	paired bootstrap

Table 6: Native cross-model calibration protocol; results in Table 3. Each model is calibrated and evaluated on its own dense rollouts and held-out prompts with the same merge operator as the main experiments. Causal Forcing and SGF share the framewise autoregressive architecture and the same 29-prompt evaluation set (hence n=29 for both); *SGF*† denotes the self-gradient-forcing EMA-weight variant, distinct from the `ar_diffusion` initialization used for Causal Forcing.

F.2 TRANSFER WITHOUT TARGET-MODEL CALIBRATION

Our main experiments evaluate Self-Forcing on the Wan2.1-T2V-1.3B backbone. To test whether the allocation gains transfer beyond the calibrated generator, we port CALM to **CausVid** (Yin et al., 2025)—an independently distilled autoregressive generator that shares the Wan backbone but a different training procedure and persistent-history KV path. All comparison arms use the same mask-aware SDPA attention backend, including the Dense reference; this reference differs from the released Flash-attention execution path. On the tested prompt and seed, enabling the merge hook with a budget large enough to bypass reduction reproduces this backend-matched Dense output exactly

(measured MSE 0). This checks the no-reduction path, rather than equivalence to the released execution backend. We then compare Uniform with the transferred allocation at three matched budgets on 30 held-out prompts with matched noise and sampler settings.

Crucially this is a *transfer*, not a re-calibration: we take the per-(layer,head) allocation *ranking* calibrated on Self-Forcing and water-fill it into the adapter's budget grid, holding the total representative budget equal to Uniform's. The transferred allocations use a maximum quota of 11 representatives per head; this is an allocation constraint, not CausVid's native history length. No CausVid distortion table is used to construct these maps.

R	budget B	Uniform ↓	CALM ↓	Δ (CALM − Uniform)	wins
8	2880	0.0100	0.0105	$+0.0005$ $[-0.0002, +0.0011]$	$12/30$
4	1440	0.0204	0.0212	$+0.0008$ $[-0.00003, +0.00192]$	$12/30$
2	720	0.0545	**0.0457**	$\mathbf{-0.0088}$ $[-0.0124, -0.0054]$	$\mathbf{28/30}$

Table 7: Transfer of the Self Forcing allocation map to CausVid **without target model calibration**. R and B specify the cache budget, Δ is the paired LPIPS difference from uniform allocation, and wins counts improved prompts. Lower LPIPS is better. The transferred map improves quality only under the strongest compression setting.

Table 7 reports all three budgets. At $R{=}8, 4$, Uniform incurs relatively low LPIPS (0.0100 and 0.0204). Transfer yields slightly higher point estimates, with paired 95% confidence intervals that include zero and $12/30$ prompt wins at each budget. These results do not establish an improvement or statistical equivalence. At $R{=}2$, Uniform's LPIPS rises to 0.0545, and the transferred map reduces it to 0.0457 (approximately 16%), winning $28/30$ prompts; the paired confidence interval excludes zero. Thus, transfer helps at the tightest tested budget, supporting the usefulness of the Self-Forcing allocation ranking on this second generator without target-model calibration.

Equal representative counts need not produce equal distortion across generators and rollout lengths. For example, Uniform at $R{=}8$ has LPIPS 0.149 on our Self-Forcing evaluation but 0.0100 on CausVid. This motivates interpreting gains alongside each baseline's distortion, rather than assuming that equal R defines equal compression difficulty. These experiments do not match distortion across models: even CausVid at $R{=}2$ has lower baseline LPIPS than Self-Forcing at $R{=}8$, and the rollout protocols differ. The same caveat applies within CausVid: this subsection's transferred-map Uniform LPIPS (0.0545 at $R{=}2$) is lower than the native CausVid Uniform in Table 3 (0.135 at $R{=}2$). The two use different rollout protocols, so their absolute LPIPS are not directly comparable across the two tables. Low baseline error limits the available absolute reduction but does not rule out a gain at a less aggressive budget. The observed budget dependence is empirical evidence, not a proof that calibration can help only in a constrained regime.

The *transfer* evidence here is limited to CausVid (30 prompts, with a positive result at $R{=}2$); native calibration on three generators is reported separately in Table 3. Transferring a source allocation ranking and recalibrating on the target model answer different questions. The former tests whether the allocation pattern transfers; the latter requires building a CausVid-native distortion table and solving its budget-constrained allocation problem. A comparison of Uniform, the transferred map, and a native map under the same target-model protocol would assess both; native calibration results (including CausVid) are reported in Table 3 and Table 6. Even so, native calibration is not guaranteed to improve video LPIPS over transfer, because dynamic programming minimizes the calibrated additive proxy within its candidate set, rather than end-to-end video distortion.

Full-sequence diffusion requires a separate extension: compressing the K/V used by each attention call differs from compressing the accumulated history of an autoregressive generator. Such an extension needs a model-specific adapter, an explicit grouping domain and budget definition, and independent validation. Transfer across backbone architectures and this full-sequence extension remain future work.

G ADDITIONAL VBENCH RESULTS

Dimension ↑	n	Dense	Uniform	CALM
Subject Consistency	72	0.948	0.928	**0.941**
Background Consistency	85	0.955	0.947	**0.952**
Temporal Flickering	75	0.990	0.989	**0.990**
Motion Smoothness	72	0.984	0.983	**0.984**
Imaging Quality	92	0.698	0.688	**0.697**
Aesthetic Quality	92	0.628	0.613	**0.622**
Dynamic Degree	72	0.653	0.611	0.597
Human Action	100	0.810	0.810	0.810
Appearance Style	90	0.199	0.202	0.200
Temporal Style	94	0.231	0.233	0.232
Overall Consistency	92	0.241	0.242	0.241

Table 8: VBench comparison across all available scoreable dimensions. n is the number of matched samples, and higher scores indicate better quality. The unavailable object detection dimensions are omitted. CALM matches or outperforms uniform allocation on the quality and consistency dimensions, while the other dimensions remain similar.

For completeness we report every VBench dimension whose scorer was available to us—eleven of the sixteen; the five GRIT-based object-detection dimensions are omitted because the VBench GRIT checkpoint was not available in this evaluation (Table 8). Beyond the six compression-relevant dimensions of Table 2, the style and alignment dimensions (Human Action, Appearance Style, Temporal Style, Overall Consistency) change little under compression in this evaluation: the three arms differ only by small margins. Dynamic Degree is reported but read with care: VBench itself notes the metric is easily gamed—it scores raw motion magnitude, not fidelity, so a lower value is not necessarily better or worse. Here CALM (0.597) and uniform (0.611) both fall below Dense (0.653); we do not read this as a quality difference on its own, and it should be interpreted alongside the consistency and fidelity dimensions.

H EFFICIENCY AND CALIBRATION COST

	Dense	Uniform (R=8)	CALM
Theoretical logical KV (GB) ↓	12.9	2.3	2.3
Realized KV buffer (GB)	12.9	12.9	12.9
Peak GPU memory (GB)	30.0	30.1	30.1
Per-video generation time (s)	31.0	30.2	34.7
Offline calibration (one-time)	—	—	50×17 shadow-probed rollouts

Table 9: Efficiency accounting, separating three distinct memory quantities. *Theoretical logical KV* is the ideal compact cache at the representative budget (B=2880 for the compressed arms); *realized KV buffer* is the resident cache our implementation actually holds (the full 45-frame buffer, merged slots marked dead rather than freed), so it is identical across methods; *peak GPU memory* is the measured device peak. Peak memory and per-video generation time are measured on a single RTX PRO 6000 Blackwell over the evaluation prompts (per-video means; CALM's median is 35.6 s). The frozen allocation adds no online allocation search at inference (the dynamic program runs once, offline). In this measured setting CALM runs about 15% slower than Uniform per video; the per-head padding of non-uniform maps (each layer's heads padded to its widest head) may contribute this overhead, which we have not profiled further. Calibration is a one-time offline cost of 50×17 dense rollouts, each instrumented with shadow-cache probing, plus a negligible dynamic program.

We keep three quantities distinct (Table 9). *(i) Logical cache.* At budget B a compressed arm holds $\sum_{l,h} P_{l,h}$=B temporal representatives; each representative is one key/value over S=1560

spatial positions per frame, so the ideal compact size is $2\,b\,S\,d_h\,q\,B$ bytes (K and V; batch b, q bytes/element). At $B{=}2880$ this is ≈ 2.3 GB versus Dense's ≈ 12.9 GB, the $45/8 \approx 5.6\times$ ratio—a *theoretical* bound. *(ii) Realized cache.* Our implementation pre-allocates the KV buffer for the full 45-frame rollout and marks merged slots dead rather than freeing them, and heads are padded to their layer's widest head; the *resident* buffer is therefore ≈ 12.9 GB for every method regardless of B. *(iii) Latency.* The frozen allocation adds no online allocation search at inference—the dynamic program runs once, offline. In this measured setting CALM generates video about 15% slower than uniform in our current implementation (per-video means 30.2 vs. 34.7 s; Table 9). A non-uniform map pads each layer's heads to its widest head, which may contribute this overhead; we have not profiled it further. We accordingly report compression as a logical/theoretical quantity: the paper's claim is *quality at a fixed representative budget*, not a realized memory or speed win. Whether compact storage and a matching ragged or compaction-aware attention kernel recover the $5.6\times$ and remove this overhead is left to verify.