Conformal Cache: Reliable Proxy-Discrepancy Caching for Fast Generative Inference

Wenxiao Wu[1,2,*] **Yukang Feng**[2,3,*] **Yachen Gao**[2,4] **Chengming Xu**[4✉] **Moran Li**[5]
Jingyu Li[2,4] **Ming Xie**[2,4] **Xiaobin Hu**[6] **Xinwei Sun**[4] **Jing-Hao Xue**[7]
Nong Sang[1✉] **Yanwei Fu**[2,4]
[1] Huazhong University of Science and Technology [2] Shanghai Innovation Institute
[3] Nankai University [4] Fudan University [5] ByteDance
[6] National University of Singapore [7] University College London
{wenxiaowu, nsang}@hust.edu.cn yukangfeng@nankai.edu.cn
moramli.aca@gmail.com xbhunanu@126.com jinghao.xue@ucl.ac.uk
{ycgao23, cmxu18, jingyuli24, mxie20, sunxinwei, yanweifu}@fudan.edu.cn

Abstract

While diffusion and flow-based generative models have achieved impressive performance in high-fidelity image and video synthesis, this capability comes at the cost of substantial inference overhead during iterative sampling. Cache-based acceleration alleviates this cost by reusing intermediate computations across adjacent denoising or transport steps, typically relying on lightweight proxy discrepancies to decide when cached outputs should be refreshed. However, such proxy-discrepancy rules are inherently unreliable: the proxy is only a point estimate of the inaccessible oracle discrepancy and can misalign across prompts, timesteps, and generation dynamics. This mismatch induces asymmetric failures: proxy underestimation may cause unsafe reuse of stale cached computations and degrade generation quality, while proxy overestimation leads to overly conservative refreshes and reduced acceleration. To address these issues, this paper introduces Conformal Cache, dubbed CCache, a training-free conformal calibration framework for reliable cache-based generation acceleration. Our key insight is to formulate cache reuse as a one-sided risk-control problem and adaptively calibrate a timestep-wise upper correction for the signed proxy-to-oracle gap based on split conformal prediction. CCache is plug-and-play and can be integrated with different proxy-discrepancy caching baselines without structural modifications or parameter updates to the generative model. Extensive experiments on image and video generation demonstrate that CCache improves generation quality and cache reliability while preserving favorable speed-quality trade-offs. The source code is available in https://github.com/Wu-Wenxiao/CCache.

1 Introduction

Diffusion models and flow-based generative models [1, 2, 3] have recently achieved remarkable progress in high-fidelity image and video generation. Their strong generative capability, however, comes at the cost of substantial inference overhead, as producing a single sample typically requires repeatedly performing full forward passes through a large neural network across many denoising or transport steps. Consequently, reducing the number of expensive network evaluations without fine-tuning the model has emerged as a critical problem for practical deployment.

*Equal contributions.
✉Co-corresponding authors.

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

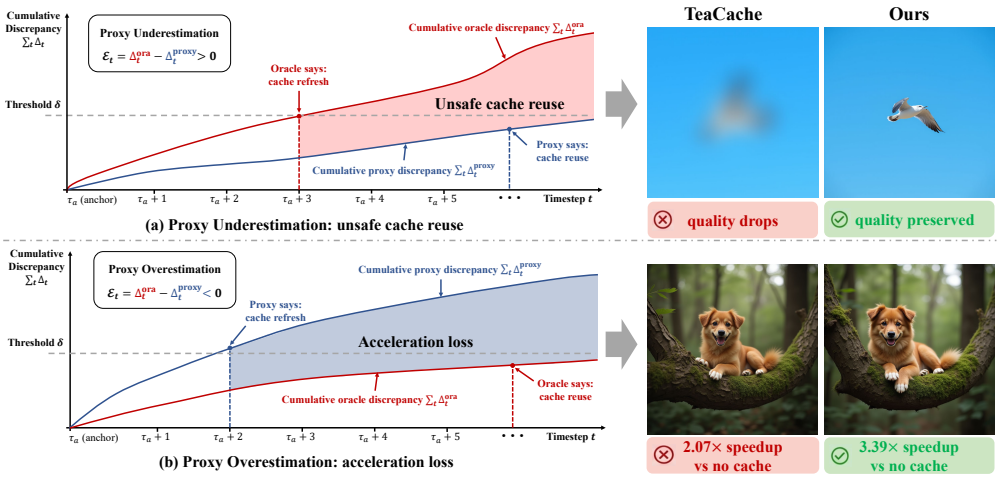


Figure 1: Asymmetric failure modes of proxy-discrepancy caching. These asymmetric consequences motivate CCache to prioritize a reliable upper correction for the signed proxy-to-oracle gap, preventing unsafe underestimation while avoiding excessive conservatism.

A prominent line of recent work [4, 5, 6] accelerates diffusion inference through cache-based computation reuse. These methods exploit the strong temporal redundancy between adjacent denoising or transport steps, allowing intermediate features, attention outputs, residuals, or other model outputs from previous steps. Since directly verifying the safety of cache reuse would require additional full network evaluations, existing methods [7, 8] usually rely on lightweight proxy discrepancies to determine cache refreshes. These proxies may be instantiated as timestep-modulated input differences, shallow-layer feature variations, transformation-vector changes, or residual-magnitude estimates. Such proxy-discrepancy caching methods are appealing because they are training-free, efficient, and compatible with any pretrained generative backbones.

Despite their empirical success, proxy-discrepancy cache rules suffer from a fundamental reliability issue: the cheap proxy is only a point estimate of the inaccessible oracle discrepancy. Since proxy-to-oracle alignment inevitably shifts across prompts, timesteps, and generation dynamics, the proxy cannot remain uniformly accurate in practice. This unavoidable mismatch gives rise to two asymmetric failure modes: **(1)** ***The proxy underestimation.*** When the proxy underestimates the oracle discrepancy, the model may falsely regard stale cached computations as still reliable. As demonstrated in Figure 1(a), cache reuse can continue even after the true oracle discrepancy has exceeded the tolerated threshold. Since generative inference is inherently iterative, such uncorrected errors may propagate and accumulate across subsequent steps, eventually leading to visible artifacts, temporal inconsistency, or even generation collapse. Therefore, reliable cache acceleration requires not only estimating oracle discrepancy, but also controlling the unsafe one-sided underestimation risk. **(2)** ***The proxy overestimation.*** When the proxy overestimates the oracle discrepancy, the cache-refresh condition is violated earlier than necessary. This behavior is generally safe from the perspective of generation quality, because stale cached computations are not overused. However, as shown in Figuer 1(b), it reduces the achievable acceleration ratio by triggering redundant full network evaluations. Therefore, robust proxy-discrepancy caching should prioritize a trustworthy upper bound on the oracle discrepancy to prevent unsafe underestimation, while avoiding excessive conservatism that unnecessarily sacrifices acceleration.

To this end, we propose **C**onformal **C**ache, dubbed **CCache**, a calibration framework that adaptively upper-bounds proxy estimation risk for robust cache-based generation acceleration. Our key observation is that cache-refresh decisions can be formulated as a one-sided risk-control problem. This motivates calibrating an upper correction for the signed proxy-to-oracle gap, which directly targets the unsafe underestimation tail. Built on this principle, CCache addresses the two failure modes discussed above through the following major improvements: (1) *Timestep-wise Conformal Correction for Safe Cache Reuse.* Given a fixed proxy-discrepancy function inherited from an existing cache baseline, CCache uses a small calibration set to run full inference and measure the residual between the oracle discrepancy and the proxy discrepancy at each timestep. Split conformal prediction is then used to estimate timestep-wise upper quantiles of this signed residual. During inference, these calibrated corrections are added to the original proxy discrepancies before applying the cache-

refresh rule. In this way, CCache transforms heuristic proxy reuse into calibrated proxy-discrepancy reuse, reducing the risk of unsafe underestimation without modifying or retraining the generative model. (2) *Online Miscoverage Adaptation for Efficient Cache Reuse.* To mitigate the conservatism and efficiency degradation caused by fixed high-coverage correction, we further introduce an online segment-level miscoverage adaptation strategy. Whenever the cache is refreshed, the full model outputs at the anchor and refresh timesteps are available, allowing CCache to assess whether the previous cache segment has exceeded the tolerated oracle discrepancy. If a violation is observed, CCache decreases the miscoverage level to produce a more conservative correction; otherwise, it increases the miscoverage level to relax the correction and encourage more cache reuse. This adaptive strategy improves acceleration under reliable cache reuse, while avoiding the excessive conservatism of a fixed high-coverage setting. Above all, CCache is fully training-free and broadly applicable. It only calibrates the residual behavior of an existing proxy and inserts the resulting correction into the cache-refresh criterion. Therefore, CCache can serve as a plug-in reliability layer for different proxy-discrepancy caching methods.

We summarize our contributions as four-folds: (1) We introduce a new formulation of proxy-discrepancy caching as a one-side reliability problem, showing that proxy underestimation can induce cache-related generation degradation, whereas proxy overestimation causes conservative refreshes and reduced acceleration; (2) We propose a split conformal prediction-based correction strategy to calibrate the signed proxy-to-oracle gap, thereby transforming heuristic proxy reuse into reliable cache reuse . The resulting method-agnostic, plug-and-play reliability layer accommodates different proxy mechanisms without modifying or retraining the generative model; (3) We introduce an online adaptive miscoverage strategy to mitigate the conservatism of fixed high-coverage correction, dynamically balancing cache reliability and acceleration efficiency; (4) Extensive experiments across methods, generation tasks, and model scales demonstrate that our CCache can substantially improve the robustness of proxy-discrepancy caching methods, achieving better generation quality and cache reliability while preserving favorable speed-quality trade-offs.

2 Related work

Cache-based Acceleration for Generative Models and Proxy-Discrepancy Caching. Cache-based acceleration methods exploit temporal redundancy in diffusion inference by reusing intermediate computations across adjacent denoising steps, often without modifying or retraining the pretrained backbone. DeepCache [4] reuses cached high-level features while updating low-level ones cheaply. Block Caching [5] extends this idea to layer-wise reuse by caching block outputs according to their temporal variation. For video diffusion transformers, Pyramid Attention Broadcast [6] broadcasts attention outputs across timesteps by exploiting their U-shaped temporal variation. Beyond architecture-specific reuse, recent methods increasingly adopt proxy-discrepancy rules, where lightweight proxy signals are used to determine when to refresh cached computations. Among these methods, TeaCache [7] estimates model-output differences from timestep-modulated input differences. EasyCache [9] dynamically reuses previously computed transformation vectors without offline profiling or pre-computation, and MagCache [10] exploits a magnitude law for adaptive timestep skipping. DiCache [8] uses shallow-layer probes for sample-specific cache decisions. Beyond direct reuse, GOC [11] compensates caching errors using temporal feature gradients and inflection-aware correction. TaylorSeer [12] forecasts future features through higher-order Taylor expansions. ERTACache [13] combines residual error rectification with trajectory-aware timestep adjustment to mitigate feature shift and step amplification. Rather than improving feature approximations or designing a new discrepancy proxy, we revisit proxy-discrepancy caching from a reliability perspective and formulate cache reuse as a one-sided risk-control problem. By directly controlling the unsafe underestimation tail, CCache improves the reliability of cache reuse.

Conformal Prediction and Adaptive Conformal Inference. Conformal prediction [14, 15, 16] provides distribution-free uncertainty quantification with theoretically guaranteed coverage probabilities under exchangeability. Conformal prediction has attracted increasing attention in tasks that require rigorous uncertainty quantification, including multi-class prediction [17], In-context learning [18], LLM benchmarking [19], and robotic trajectory prediction [20]. Split conformal prediction (SCP) [21, 22] is especially relevant to our setting because it calibrates a fixed prediction rule on held-out data without retraining the underlying model. Extensions further study split conformal prediction under certain non-exchangeable processes via explicit coverage penalties [22]. Beyond static calibration, adaptive conformal inference updates the miscoverage level online to handle distribu-

tion shifts over time [23]. It is also related to conformal risk-control frameworks, which calibrate black-box predictors to satisfy user-specified finite-sample risk bounds [24]. Motivated by adaptive conformal inference, CCache dynamically updates the miscoverage level of the SCP-calibrated upper correction using an online cache-reliability signal, effectively reducing excessive conservatism while maintaining reliable cache reuse.

3 Methodology

3.1 Preliminary: a general formulation of Proxy-Discrepancy Caching

Oracle Discrepancy and Ideal Caching Condition. In diffusion models and flow matching, generation is performed by discretizing a continuous denoising or transport trajectory into T timesteps. At each timestep t, the model evaluates a full neural network Φ to obtain the velocity, residual, or feature output. We denote the oracle output at timestep t as $\mathbf{O}_t = \Phi(\mathbf{x}_t, t, c)$, where $\mathbf{x}_t$ is the current latent state and c denotes the conditioning signal like a text prompt. Since evaluating Φ at every timestep dominates the inference cost, caching-based acceleration methods aim to skip redundant full forward passes by reusing a cached output from a previous anchor timestep t_a. Given a distance function $d(\cdot, \cdot)$, such as relative L_1 distance, the oracle discrepancy between consecutive timesteps is defined as $d(\mathbf{O}_t, \mathbf{O}_{t+1})$. Starting from an anchor timestep τ_a, cached computation can be safely reused as long as the cumulative oracle discrepancy remains below a predefined threshold δ:

$$\sum_{t=\tau_a}^{\tau-1} d(\mathbf{O}_t, \mathbf{O}_{t+1}) \leq \delta. \tag{1}$$

Once the cumulative discrepancy exceeds δ, the cache should be refreshed by performing a full network evaluation and resetting the anchor. However, directly computing $d(\mathbf{O}_t, \mathbf{O}_{t+1})$ requires evaluating Φ at both timesteps, which defeats the purpose of caching.

Proxy-Discrepancy Function. Proxy-Discrepancy Caching introduces a lightweight proxy function $g(\cdot, \cdot)$ to estimate the oracle discrepancy without running the full network. Since the oracle discrepancy is defined between consecutive model outputs, the proxy is generally computed from the corresponding low-cost feature variation between adjacent timesteps. Let $\mathbf{F}_t$ denote an easily accessible feature at timestep τ. The proxy estimates the single-step oracle discrepancy as

$$g(\mathbf{F}_t, \mathbf{F}_{t+1}) \approx d(\mathbf{O}_t, \mathbf{O}_{t+1}). \tag{2}$$

Different caching methods instantiate g in different ways. For example, TeaCache [7] uses a polynomially corrected difference of timestep-modulated inputs, while DiCache uses shallow-layer feature variation as an online proxy for output variation. Despite using different proxy signals, both methods share the same underlying principle: replacing the expensive oracle discrepancy with a cheap error predictor.

During inference, Proxy-Discrepancy Caching avoids full network evaluation at every timestep. Instead, it accumulates the predicted discrepancy given by the proxy function and reuses the cache while $\sum_{t=\tau_a}^{\tau-1} g(\mathbf{F}_t, \mathbf{F}_{t+1}) \leq \delta$. A cache refresh is triggered at the first timestep τ_b where

$$\sum_{t=\tau_a}^{\tau_b-1} g(\mathbf{F}_t, \mathbf{F}_{t+1}) \leq \delta < \sum_{t=\tau_a}^{\tau_b} g(\mathbf{F}_t, \mathbf{F}_{t+1}). \tag{3}$$

At this point, the model performs a full forward pass, updates the cached output, clears the accumulated error, and resets the anchor as $\tau_a \leftarrow \tau_b$.

Motivation: The Proxy-to-Oracle Gap. Although Proxy-Discrepancy Caching provides a general and efficient formulation for generation acceleration, it also exposes a fundamental limitation. The proxy g is typically a deterministic point estimate calibrated from observed data. Therefore, it may fail to precisely predict the true oracle discrepancy under diverse prompts, complex motion patterns, rare visual concepts, or out-of-distribution inputs. We define the proxy-to-oracle gap as

$$\mathcal{E}_t = d(\mathbf{O}_{t-1}, \mathbf{O}_t) - g(\mathbf{F}_{t-1}, \mathbf{F}_t). \tag{4}$$

Analysis. This generalization failure can manifest in two directions, creating highly asymmetric consequences that demand distinct strategies:

- **Severe Underestimation ($\mathcal{E}_t \gg 0$):** The proxy substantially underestimates the oracle discrepancy, causing the model to falsely regard the cached output as still reliable. Consequently, The accumulated proxy discrepancy may remain below δ even after the true oracle discrepancy has exceeded the tolerated range, leading to prolonged cache reuse. Since the reverse generative trajectory is iterative, such uncorrected deviations can propagate and amplify across successive timesteps, eventually leading to even catastrophic generation collapse. This motivates moving beyond heuristic point estimation for robust quality preservation. *To safely constrain this risk, we establish a reliable upper bound on $\mathcal{E}_\tau$ using Split Conformal Prediction (detailed in Section 3.2).*
- **Conservative Overestimation ($\mathcal{E}_t < 0$):** The proxy overestimates the oracle discrepancy, causing the caching condition to be violated prematurely. While this is generally safe and preserves generation quality, it leads to more frequent cache refreshes and thus a suboptimal acceleration ratio. *To recover efficiency after conservative correction, we further adapt the calibration miscoverage level online using a segment-level reliability signal (detailed in Section 3.3)).*

3.2 Timestep-wise conformal correction for safe cache reuse

This section details the construction of a reliable timestep-wise correction for Proxy-Discrepancy Caching, which upper-bounds the proxy-to-oracle gap $\mathcal{E}_t$ with a user-specified significance level α, or equivalently a coverage level $1-\alpha$, guided by the theory of split conformal prediction.

We begin by randomly sampling a minuscule calibration subset of prompts from the target evaluation dataset $\mathcal{X}_{test}$, denoted as $\mathcal{X}_{cal} = \{c_i\}_{i=1}^{N} \subset \mathcal{X}_{test}$, where $N \ll |\mathcal{X}_{test}|$. The remaining prompts $\mathcal{X}_{eval} = \mathcal{X}_{test} \setminus \mathcal{X}_{cal}$ are reserved for evaluation. Please note that we do not need extra data to train a predictive model, because the proxy-discrepancy function $g(\cdot,\cdot)$ is inherited from existing caching baselines, such as the *poly1d* function from TeaCache [7], and is used as a fixed discrepancy estimator. $\mathcal{X}_{cal}$ is only used to calibrate the proxy-to-oracle gap defined in Eq. (4). For each prompt $c_i \in \mathcal{X}_{cal}$, we run the full T-step generation process without caching. This gives the oracle outputs $\{\mathbf{O}_{i,t}\}_{t=1}^{T}$ and the corresponding lightweight proxy features $\{\mathbf{F}_{i,t}\}_{t=1}^{T}$. For each timestep $t \in \{2,\dots,T\}$, we compute the oracle discrepancy and proxy discrepancy between the current timestep and its previous timestep as

$$\Delta_{i,t}^{\mathrm{ora}} = d(\mathbf{O}_{i,t-1}, \mathbf{O}_{i,t}), \qquad \Delta_{i,t}^{\mathrm{proxy}} = g(\mathbf{F}_{i,t-1}, \mathbf{F}_{i,t}). \tag{5}$$

Based on the above definitions, the unconformity score $s_{i,t}$ for each calibration prompt c_i at timestep t is calculated as the proxy-to-oracle gap:

$$\mathcal{E}_{i,t} = \Delta_{i,t}^{\mathrm{ora}} - \Delta_{i,t}^{\mathrm{proxy}}. \tag{6}$$

We use the signed proxy-to-oracle gap $\mathcal{E}_{i,t}$, rather than its absolute value, as the nonconformity score because proxy errors have asymmetric consequences in cache reuse. A positive gap indicates proxy underestimation, which is unsafe since stale cached outputs may be reused beyond the tolerated range. In contrast, a negative gap indicates proxy overestimation, which is conservative buys safe, and mainly leads to earlier refreshes and reduced acceleration. Therefore, instead of pursuing a two-sided accurate proxy, our objective is to prioritize a reliable upper correction for the unsafe underestimation direction. This procedure is repeated for every $c_i \in \mathcal{X}_{cal}$ at each timestep t. Then, for each timestep t, a set including all N unconformity scores can be denoted by

$$\mathcal{S}_t = \{\mathcal{E}_{i,t}\}_{i=1}^{N} \cup \{+\infty\}. \tag{7}$$

Subsequently, the corresponding upper $(1-\alpha)$-quantile $q_{t,1-\alpha}$, which serves as the timestep-specific reliability correction, is determined from the empirical distribution of unconformity scores in $\mathcal{S}_t$:

$$q_{t,1-\alpha} = \text{the } \lceil (1-\alpha)(N+1) \rceil \text{-th smallest value in } \mathcal{S}_t. \tag{8}$$

Under the exchangeability assumption between calibration and evaluation prompts, split conformal prediction guarantees that, for each timestep t,

$$\mathbb{P}\left[\mathcal{E}_t \le q_{t,1-\alpha}\right] = \mathbb{P}\left[\Delta_t^{\mathrm{ora}} \le \Delta_t^{\mathrm{proxy}} + q_{t,1-\alpha}\right] \ge 1-\alpha. \tag{9}$$

This conformal guarantee is timestep-wise and marginal, and thus should not be interpreted as a formal trajectory-level coverage guarantee. Therefore, by adding the conformal correction $q_{t,1-\alpha}$ to the original proxy discrepancy, we obtain a high-confidence upper estimate of the oracle discrepancy:

$$\widetilde{\Delta}_t^{\mathrm{proxy}} = \Delta_t^{\mathrm{proxy}} + q_{t,1-\alpha}. \tag{10}$$

After calibration, all timestep-wise corrections constitute a vector

$$\mathbf{q}_{1-\alpha} = [q_{2,1-\alpha}, q_{3,1-\alpha}, \ldots, q_{T,1-\alpha}], \tag{11}$$

which characterizes the timestep-dependent proxy-to-oracle gap along the generation trajectory. During inference, for any prompt $c \in \mathcal{X}_{eval}$, we directly use this correction vector to replace the original proxy discrepancy Δ_t^{proxy} with its corrected version $\widetilde{\Delta}_t^{\text{proxy}}$ in the cache reuse criterion. Specifically, starting from an anchor timestep τ_a, a cache refresh is triggered at the first timestep τ_b such that

$$\sum_{t=\tau_a+1}^{\tau_b-1} \left[g(\mathbf{F}_{t-1}, \mathbf{F}_t) + q_{t,1-\alpha} \right] \leq \delta < \sum_{t=\tau_a+1}^{\tau_b} \left[g(\mathbf{F}_{t-1}, \mathbf{F}_t) + q_{t,1-\alpha} \right]. \tag{12}$$

Compared with the original proxy-only caching rule, our conformally corrected rule explicitly accounts for the calibrated proxy-to-oracle gap, thereby reducing the risk of severe proxy underestimation while retaining the efficiency of lightweight proxy accumulation.

Remark. The conformal correction provides a timestep-wise marginal upper bound on the proxy-to-oracle gap. Since the cache-refresh rule accumulates corrected discrepancies over a segment, CCache should be viewed as a conformally calibrated surrogate for safer cumulative cache.

3.3 Online miscoverage adaptation for efficient cache reuse

The conformal correction above effectively reduces the risk of unsafe proxy underestimation. However, under the same caching threshold δ, adding $q_{t,1-\alpha}$ to the original proxy discrepancy inevitably makes the cache-refresh rule more conservative than the original proxy-discrepancy caching method. This effect becomes more pronounced when α is small, since a smaller significance level corresponds to a larger coverage level $1-\alpha$, and thus leads to a larger conformal correction. Consequently, although a fixed high-coverage correction improves reliability, it may also trigger overly frequent cache refreshes and reduce the acceleration ratio. Inspired by adaptive conformal inference, we further introduce an online adaptation strategy to better balance generation quality and inference efficiency, which updates the coverage level on the fly based on the observed discrepancy at cache refresh points.

Let α_m denote the adaptive miscoverage level used for the m-th cache segment. In the fixed conformal correction described above, α_m remains unchanged throughout inference. In contrast, our adaptive conformal inference updates α_m online according to the observed reliability of each cache segment. Specifically, given an anchor timestep τ_a^m, suppose the cache is refreshed at timestep τ_b^m. Since full model outputs are available at both τ_a^m and τ_b^m, we compute the observed oracle discrepancy as $D_m^{\text{ora}} = d(\mathbf{O}_{\tau_a^m}, \mathbf{O}_{\tau_b^m})$. If $D_m^{\text{ora}} > \delta$, the true discrepancy has exceeded the tolerance threshold, indicating that the current correction is still too optimistic. In this case, we decrease α_m, which increases the coverage level $1-\alpha_m$ and yields a more conservative correction. Otherwise, if $D_m^{\text{ora}} \leq \delta$, the current correction is sufficiently safe, and we increase α_m to reduce the correction and encourage more cache reuse. Formally, we define the segment-level violation indicator as

$$v_m = \mathbb{1}\left[D_m^{\text{ora}} > \delta \right]. \tag{13}$$

Given a target miscoverage level α_0 and an update step size η, we update the adaptive miscoverage level by

$$\alpha_{m+1} = \Pi_{[\alpha_{\min}, \alpha_{\max}]} \left(\alpha_m + \eta(\alpha_0 - v_m) \right), \tag{14}$$

where $\Pi_{[\alpha_{\min}, \alpha_{\max}]}$ denotes clipping to a predefined valid range. When a violation occurs, i.e., $v_m = 1$, Eq. (14) decreases α_m, thereby increasing the coverage level $1-\alpha_m$ and producing a larger conformal correction in the next segment. When no violation occurs, i.e., $v_m = 0$, Eq. (14) increases α_m, thereby reducing the coverage level and allowing the method to reuse cached computation more aggressively. In this way, Eq. (14) adaptively tightens or relaxes the correction according to the observed cache reliability, improving the acceleration ratio while preserving the safety benefit of conformal calibration.

When it comes to implementation, the calibration scores can be sorted once to support efficient quantile extraction. For each candidate miscoverage level α, we directly extract the corresponding timestep-wise quantiles from the sorted calibration scores, yielding a correction vector $\mathbf{q}_{1-\alpha}$. During inference, the adaptive strategy simply selects the correction vector associated with the current α_m. This lookup-based design incurs negligible additional overhead while enabling online adjustment of the conformal correction.

Table 1: Quantitative comparison on HPSV3 for text-to-image generation. "+ CCache" denotes our calibrated version of each caching baseline.

Model	Method	Speedup↑	Latency↓	HPSv3 Characters	Arts	Design	Architecture	Animals	Natural Scenery
FLUX.1-dev	Vanilla (100% steps)	1.00×	6.45s	-	-	-			-
	Uniform Cache ($I = 2$)	1.90×	3.40s	11.43	10.05	8.33	10.76	9.78	9.38
	TeaCache ($\delta = 0.40$)	1.98×	3.25s	11.55	10.16	8.93	10.58	10.05	9.34
	+ CCache	2.05×	3.14s	11.59	10.13	9.04	10.57	10.14	9.50
	TeaCache ($\delta = 0.80$)	2.99×	2.16s	10.20	9.02	7.57	9.36	8.48	7.71
	+ CCache	3.14×	2.06s	11.08	9.20	8.35	9.88	9.70	8.54
	DiCache ($\delta = 0.40$)	2.82×	2.29s	11.50	9.72	8.73	10.91	10.62	10.00
	+CCache	2.78×	2.32s	11.91	10.06	9.01	11.47	11.15	10.39
Model	**Method**	**Speedup↑**	**Latency↓**	**HPSv3 Transportation**	**Products**	**Others**	**Plants**	**Food**	**Science**
FLUX.1-dev	Vanilla (100% steps)	1.00×	6.45s	-	-	-			-
	Uniform Cache ($I = 2$)	1.90×	3.40s	10.58	10.05	9.75	8.99	10.41	8.30
	TeaCache ($\delta = 0.40$)	1.98×	3.25s	10.46	10.10	9.80	9.56	10.64	8.68
	+ CCache	2.05×	3.14s	10.60	10.30	9.95	10.03	10.96	8.77
	TeaCache ($\delta = 0.80$)	2.99×	2.16s	8.95	8.53	8.45	6.46	8.61	7.26
	+ CCache	3.14×	2.06s	10.23	9.92	9.48	9.45	10.81	7.89
	DiCache ($\delta = 0.40$)	2.82×	2.29s	10.91	10.80	10.37	10.48	11.90	8.57
	+ CCache	2.78×	2.32s	11.56	11.39	10.85	11.03	12.33	8.93

4 Experiments

We give the setups, and evaluate CCache on two state-of-the-art proxy-discrepancy caching baselines, TeaCache [7] and DiCache [8], across both image and video generation tasks.

Experimental Settings. We conduct experiments on Wan2.1-1.3B [2] and Wan2.1-14B [2] for video generation, and FLUX.1-dev [1] for image generation. Since the conformal upper correction makes the CCache-calibrated rule more conservative under the same cache threshold, directly using the same threshold would typically lead to lower speedup. For a fair comparison, we adjust the caching threshold of each CCache-enhanced variant to achieve a comparable acceleration ratio to its corresponding baseline. Since reliability is our primary objective, we fix the target miscoverage level to $\alpha_0 = 0.05$. We use $N = 100$ calibration samples, yielding an empirical quantile resolution of approximately $1/(N+1) \approx 0.01$. Accordingly, we set $\eta = 0.2$, with $\alpha_{min} = 0.05$ and $\alpha_{max} = 0.5$. Under the proposed update rule, each non-violating segment increases the miscoverage level by 0.01, whereas a violating segment decreases it by 0.19. Therefore, the correction is relaxed at a practically meaningful resolution but rapidly returns toward the conservative 95% coverage baseline once a violation is observed. We evaluate generation quality and inference efficiency, reporting VMBench [25] and HPSv3 [26] scores together with speedup and latency to characterize the overall speed-quality trade-off. All experiments are conducted on 8 NVIDIA H100 GPUs. More experimental details can be seen in Appendix B.1.

Evaluation Benchmarks. For video generation, we employ VMBench [25] to assess motion quality. VMBench is a perception-aligned benchmark designed for evaluating video motion generation, with a prompt suite covering diverse motion patterns and six major dynamic scene dimensions. Following VMBench, we report five perception-driven motion metrics: *Perceptible Amplitude Score (PAS), Object Integrity Score (OIS), Temporal Coherence Score (TCS), Motion Smoothness Score (MSS), and Commonsense Adherence Score (CAS)*. For image generation, we adopt HPSv3 [26] to evaluate synthesis quality. HPSv3 is a human-preference-based image evaluation model designed to assess generated images in terms of perceptual quality and text-image alignment. For benchmarking, HPSv3 samples 1,000 prompts from each of the 12 HPDv3 categories, resulting in 12,000 prompts. We report the HPSv3 score as the image-quality metric.

4.1 Main results

Quantitative Analysis. The quantitative results for image and video generation are presented in Table 1 and Table 2, respectively. Here are several observations. First, CCache consistently improves generation quality under comparable acceleration across both video and image generation tasks. On Wan2.1-1.3B with TeaCache at $\delta = 0.08$, CCache maintains the same speedup and latency, i.e., $2.07\times$ and 46s, while improving the average VMBench score from 59.77 to 60.32. A consistent improvement is observed on FLUX.1-dev for image generation, demonstrating the gen-

Table 2: Quantitative comparison on VMBench for text-to-video generation. "+CCache" denotes our calibrated version of each caching baseline.

Model	Method	Speedup↑	Latency↓	VMBench					
				PAS	OIS	TCS	MSS	CAS	Avg.
Wan2.1-1.3B	Vanilla (100% steps)	1.00×	95s	-	-	-	-	-	-
	Uniform Cache ($I = 2$)	2.11×	45s	10.17	66.03	94.07	71.09	55.98	59.46
	TeaCache ($\delta = 0.08$)	2.07×	46s	8.78	67.91	94.08	71.62	56.44	59.77
	+ CCache	2.07×	46s	8.79	68.65	94.41	72.54	57.23	60.32
	TeaCache ($\delta = 0.20$)	3.52×	27s	8.17	67.14	93.38	65.93	53.78	57.68
	+ CCache	3.39×	28s	8.42	68.39	94.06	68.99	55.36	59.04
	DiCache ($\delta = 0.20$)	1.76×	54s	13.46	66.71	94.42	72.16	52.87	59.92
	+ CCache	1.73×	55s	14.19	67.07	94.63	75.63	54.48	61.20
Wan2.1-14B	Vanilla (100% steps)	1.00×	1794s	-	-	-	-	-	-
	Uniform Cache ($I = 2$)	1.99×	903s	-	-	-	-	-	-
	TeaCache ($\delta = 0.20$)	2.06×	872s	15.95	71.39	94.72	78.99	63.79	64.97
	+ CCache	2.05×	874s	17.55	74.56	95.63	82.82	65.91	67.29

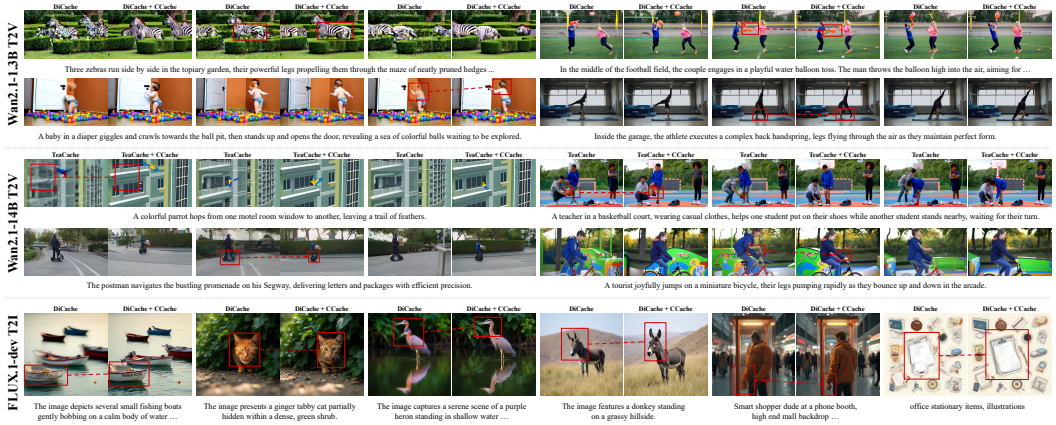


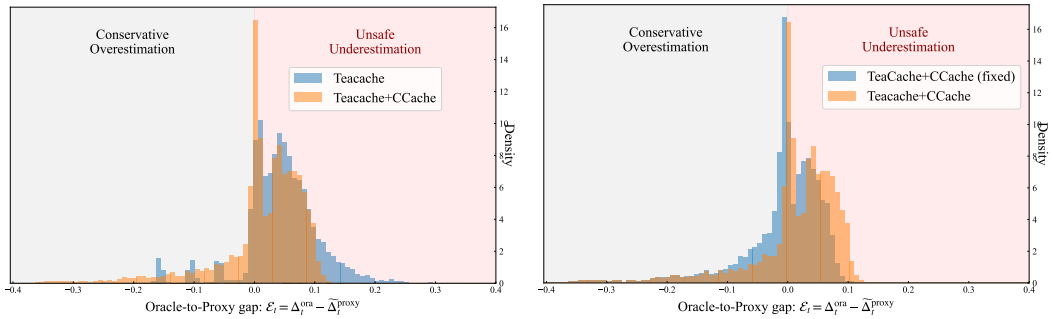
Figure 2: Qualitative comparison of baseline caching methods with and without CCache. Red boxes highlight representative differences. CCache reduces visible structural distortions, including merged subjects, malformed object boundaries, and loss of fine detail, under comparable acceleration.

eralization capability of CCache. Second, the benefit of CCache becomes more pronounced under more aggressive caching. On Wan2.1-1.3B, the average VMBench gain over TeaCache increases from +0.55 at $\delta = 0.08$ to +1.36 at $\delta = 0.20$. Similarly, on image generation, the average HPSv3 improvement under TeaCache is larger at $\delta = 0.80$, where the score increases from 8.38 to 9.54. This supports our motivation that *conformal correction is especially beneficial when the baseline proxy is more likely to misestimate the oracle discrepancy*. Third, the improvement is more substantial on the stronger Wan2.1-14B model. At the same TeaCache setting $\delta = 0.20$, CCache improves the average VMBench score by +1.36 on Wan2.1-1.3B, while the gain increases to +2.32 on Wan2.1-14B. This suggests that CCache can better preserve and exploit the stronger generative capability of larger models once unsafe proxy underestimation is constrained.

Qualitative Analysis. Figure 2 shows that CCache mitigates localized structural artifacts in the outputs of the underlying caching methods. On Wan2.1-1.3B, DiCache produces visibly fragmented zebra bodies, discontinuous stripe patterns, and poorly defined legs. Adding CCache makes the individual animals easier to distinguish and preserves more coherent body contours across the sampled frames. In the other video examples, the improvement is evident around the infant's face and arms, the water-balloon interaction, and the gymnast's extended limbs, where errors in shape or articulation are particularly noticeable. On Wan2.1-14B, TeaCache distorts the window geometry surrounding the moving parrot, while CCache retains a more regular building structure. In the basketball-court example, the CCache result also depicts the requested shoe-fitting interaction more clearly, rather than obscuring it with an inconsistent object. The FLUX.1-dev examples exhibit a similar pattern: CCache yields better-defined boat hulls and stationery outlines, and avoids the conspicuous duplicated head in the heron image. Overall, these examples complement the quantitative results by illustrating that CCache improves perceptual quality across diverse scenes, model scales,

Table 3: Ablation study on conformal correction and online adaptation. We compare TeaCache, full CCache under matched acceleration, and a fixed-miscoverage variant under the same CCache threshold. Better quality and lower latency are highlighted in bold.

Method	δ	α	Speedup↑	Latency↓	VMBench PAS	OIS	TCS	MSS	CAS	Avg.
Vanilla	-	-	1.00×	95s	-	-	-			-
TeaCache	0.08	-	**2.07**×	**46s**	8.78	67.91	94.08	71.62	56.44	59.77
+ CCache	0.13	0.05	**2.07**×	**46s**	**8.79**	**68.65**	94.41	72.54	57.23	**60.32**
+ CCache (Fixed)	0.13	0.05	1.72×	55s	8.62	68.26	**94.49**	**72.94**	**57.29**	**60.32**



(a) TeaCache vs. CCache (b) Fixed vs. adaptive miscoverage

Figure 3: Distributional analysis of conformal correction and online adaptation.

and generation tasks, while maintaining acceleration comparable to the underlying caching methods. Additional qualitative results are provided in Appendix B.3.

4.2 Ablation study

Effects of conformal correction and online adaptation. To disentangle the effects of conformal correction and online miscoverage adaptation, we conduct an ablation study based on TeaCache with $\delta = 0.08$. We first compare TeaCache with our full CCache-enhanced version under comparable acceleration. Since conformal correction makes cache reuse more conservative under the same threshold, we adjust the cache threshold of CCache to $\delta = 0.13$ and initialize the miscoverage level as $\alpha_0 = 0.05$, yielding the same $2.07\times$ speedup and 46s latency as TeaCache. This setting evaluates whether the calibrated correction improves generation quality when the practical acceleration is matched. To isolate the effect of online adaptation, we further keep the same CCache threshold $\delta = 0.13$ and compare the full adaptive version with a fixed-miscoverage variant, where the coverage level is fixed at $1 - \alpha = 95\%$ throughout inference. Since both variants use the same threshold and the same conformal correction mechanism, their latency difference mainly reflects the effect of online miscoverage adaptation. The results in Table 3 show that, under matched acceleration, CCache yields performance improvements over the baseline TeaCache, confirming the effectiveness of conformal correction for safer cache reuse. Moreover, the fixed-miscoverage variant achieves a similar average quality but is substantially slower, with speedup dropping from $2.07\times$ to $1.72\times$. This verifies that online miscoverage adaptation effectively mitigates excessive conservatism and recovers acceleration without sacrificing the quality benefit of conformal calibration.

Distributional effects of conformal correction and online adaptation. To further understand the behavior of CCache, we visualize the distribution of the signed oracle-to-proxy gap $\mathcal{E}_t = \Delta_t^{\text{ora}} - \widetilde{\Delta}_t^{\text{proxy}}$ in Figure 3. Figure 3(a) compares TeaCache with its CCache-calibrated variant. The original TeaCache exhibits a positive long tail, indicating substantial proxy underestimation of the oracle discrepancy. Applying CCache shortens this unsafe tail and concentrates the distribution near the decision boundary, confirming that conformal correction targets unsafe underestimation rather than minimizing two-sided error uniformly. Figure 3(b) compares fixed high-coverage correction with full CCache. The fixed variant is conservative, producing more negative gaps and frequent cache refreshes. Online adaptation relaxes correction when segments are reliable, thus achieving similar quality with faster execution.

Sensitivity to Calibration Set Size. We vary the number of calibration prompts from $N = 25$ to $N = 100$, while maintaining comparable acceleration. As shown in Table 4, CCache already

Table 4: Performance comparison with different calibration-set sizes.

Method	Speedup	Latency	PAS	OIS	TCS	MSS	CAS	Avg.
DiCache ($N = 0$)	$1.76\times$	54s	13.46	66.71	94.42	72.16	52.87	59.92
+CCache ($N = 25$)	$1.70\times$	56s	13.50	66.87	94.61	75.17	54.72	60.97
+CCache ($N = 50$)	$1.73\times$	55s	13.57	67.12	94.56	75.43	54.51	61.04
+CCache ($N = 75$)	$1.76\times$	54s	13.86	67.18	94.53	75.39	54.46	61.08
+CCache ($N = 100$)	$1.73\times$	55s	14.19	67.07	94.63	75.63	54.48	61.20

provides a substantial improvement with only 25 calibration prompts. Performance then improves gradually and remains stable as the calibration set grows. These results indicate that CCache does not require a large calibration set and is relatively insensitive to the precise calibration-set size.

OOD Transferability of Conformal Corrections. We further calibrate CCache using 100 prompts from a separate benchmark VBench rather than from the target VMBench evaluation pool. As shown in Table 5, despite this prompt-distribution shift, the transferred correction achieves an average VMBench score of 61.18, which is nearly identical to the 61.20 obtained using in-domain VMBench calibration prompts. We hypothesize that this transferability partly benefits from the shared semantic representation induced by the same pretrained text encoder.

Table 5: Comparison of DiCache with out-of-distribution calibration set.

Method	Speedup	Latency	PAS	OIS	TCS	MSS	CAS	Avg.
DiCache	$1.76\times$	54s	13.46	66.71	94.42	72.16	52.87	59.92
+CCache (VMBench)	$1.73\times$	55s	14.19	67.07	94.63	75.63	54.48	61.20
+CCache (VBench)	$1.79\times$	53s	14.15	66.95	94.68	74.87	55.26	61.18

Latency–Quality Trade-off across Cache Thresholds We provide a quantitative evaluation across a broader range of caching aggressiveness levels in Table 6. We sweep the cache threshold δ for DiCache and CCache on Wan2.1-1.3B. CCache consistently achieves higher VMBench scores at comparable acceleration, with the advantage increasing under more aggressive caching.

Table 6: Latency–quality comparison between DiCache and CCache across acceleration levels.

Speedup Level	DiCache			+ CCache		
	δ	Speedup	Avg.	δ	Speedup	Avg.
$\approx 1.50\times$	0.12	$1.50\times$	61.21	0.22	$1.54\times$	61.59
$\approx 1.75\times$	0.20	$1.76\times$	59.93	0.35	$1.73\times$	61.20
$\approx 1.90\times$	0.45	$1.89\times$	55.68	0.50	$1.88\times$	60.03
$\approx 2.00\times$	0.55	$1.97\times$	54.11	0.70	$1.95\times$	57.98

5 Conclusion

In this paper, we presented CCache, a training-free conformal calibration framework for reliable cache-based acceleration in diffusion and flow-based generative models. Rather than designing a stronger proxy discrepancy estimator, we revisited proxy-discrepancy caching from a reliability perspective and formulated cache reuse as a one-sided risk-control problem. By calibrating timestep-wise upper corrections for the signed proxy-to-oracle gap, CCache directly targets unsafe proxy underestimation, which is the main source of cache-induced quality degradation. To further mitigate the conservatism introduced by fixed high-coverage correction, we introduced an online miscoverage adaptation strategy that adjusts the calibration strength according to segment-level cache reliability. CCache is plug-and-play and can be integrated with existing proxy-discrepancy caching methods without modifying or retraining the generative backbone. Extensive experiments on both image and video generation demonstrate that CCache improves generation quality and cache reliability while preserving favorable speed-quality trade-offs. We hope this work encourages future research on statistically calibrated and reliability-aware acceleration mechanisms for generative model inference.

References

[1] Black Forest Labs. Flux. `https://github.com/black-forest-labs/flux`, 2024.

[2] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[3] Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. Cogvideo: Large-scale pretraining for text-to-video generation via transformers. *arXiv preprint arXiv:2205.15868*, 2022.

[4] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15762–15772, 2024.

[5] Felix Wimbauer, Bichen Wu, Edgar Schoenfeld, Xiaoliang Dai, Ji Hou, Zijian He, Artsiom Sanakoyeu, Peizhao Zhang, Sam Tsai, Jonas Kohler, et al. Cache me if you can: Accelerating diffusion models through block caching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6211–6220, 2024.

[6] Xuanlei Zhao, Xiaolong Jin, Kai Wang, and Yang You. Real-time video generation with pyramid attention broadcast. *arXiv preprint arXiv:2408.12588*, 2024.

[7] Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It's time to cache for video diffusion model. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 7353–7363, 2025.

[8] Jiazi Bu, Pengyang Ling, Yujie Zhou, Yibin Wang, Yuhang Zang, Dahua Lin, and Jiaqi Wang. Dicache: Let diffusion model determine its own cache. *arXiv preprint arXiv:2508.17356*, 2025.

[9] Xin Zhou, Dingkang Liang, Kaijin Chen, Tianrui Feng, Xiwu Chen, Hongkai Lin, Yikang Ding, Feiyang Tan, Hengshuang Zhao, and Xiang Bai. Less is enough: Training-free video diffusion acceleration via runtime-adaptive caching. *arXiv preprint arXiv:2507.02860*, 2025.

[10] Zehong Ma, Longhui Wei, Feng Wang, Shiliang Zhang, and Qi Tian. Magcache: Fast video generation with magnitude-aware cache. *arXiv preprint arXiv:2506.09045*, 2025.

[11] Junxiang Qiu, Lin Liu, Shuo Wang, Jinda Lu, Kezhou Chen, and Yanbin Hao. Accelerating diffusion transformer via gradient-optimized cache. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 17608–17617. IEEE, 2025.

[12] Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with taylorseers. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15853–15863. IEEE, 2025.

[13] Xurui Peng, Chenqian Yan, Hong Liu, Rui Ma, Fangmin Chen, Xing Wang, Zhihua Wu, Songwei Liu, and Mingbao Lin. Ertacache: Error rectification and timesteps adjustment for efficient diffusion. In *International Conference on Learning Representations*, volume 2026, pages 82104–82120, 2026.

[14] Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of machine learning research*, 9(3), 2008.

[15] Xiaofan Zhou, Baiting Chen, Yu Gui, and Lu Cheng. Conformal prediction: A data perspective. *arXiv preprint arXiv:2410.06494*, 2024.

[16] Sophia Sun. Conformal methods for quantifying uncertainty in spatiotemporal data: A survey. *arXiv preprint arXiv:2209.03580*, 2022.

[17] Tiffany Ding, Anastasios Angelopoulos, Stephen Bates, Michael Jordan, and Ryan J Tibshirani. Class-conditional conformal prediction with many classes. *Advances in neural information processing systems*, 36:64555–64576, 2023.

[18] Wenxiao Wu, Jing-Hao Xue, Chengming Xu, Chen Liu, Xinwei Sun, Changxin Gao, Nong Sang, and Yanwei Fu. Towards reliable and holistic visual in-context learning prompt selection. *Advances in Neural Information Processing Systems*, 38:111285–111305, 2026.

[19] Fanghua Ye, Mingming Yang, Jianhui Pang, Longyue Wang, Derek Wong, Emine Yilmaz, Shuming Shi, and Zhaopeng Tu. Benchmarking llms via uncertainty quantification. *Advances in Neural Information Processing Systems*, 37:15356–15385, 2024.

[20] Jiankai Sun, Yiqi Jiang, Jianing Qiu, Parth Nobel, Mykel J Kochenderfer, and Mac Schwager. Conformal prediction for uncertainty-aware planning with diffusion dynamics model. *Advances in Neural Information Processing Systems*, 36:80324–80337, 2023.

[21] Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J Tibshirani, and Larry Wasserman. Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113(523):1094–1111, 2018.

[22] Roberto I Oliveira, Paulo Orenstein, Thiago Ramos, and Joao Vitor Romano. Split conformal prediction and non-exchangeable data. *Journal of Machine Learning Research*, 25(225):1–38, 2024.

[23] Isaac Gibbs and Emmanuel Candes. Adaptive conformal inference under distribution shift. *Advances in Neural Information Processing Systems*, 34:1660–1672, 2021.

[24] Stephen Bates, Anastasios Angelopoulos, Lihua Lei, Jitendra Malik, and Michael Jordan. Distribution-free, risk-controlling prediction sets. *Journal of the ACM (JACM)*, 68(6):1–34, 2021.

[25] Xinran Ling, Chen Zhu, Meiqi Wu, Hangyu Li, Xiaokun Feng, Cundian Yang, Aiming Hao, Jiashu Zhu, Jiahong Wu, and Xiangxiang Chu. Vmbench: A benchmark for perception-aligned video motion generation. *arXiv preprint arXiv:2503.10076*, 2025.

[26] Yuhang Ma, Xiaoshi Wu, Keqiang Sun, and Hongsheng Li. Hpsv3: Towards wide-spectrum human preference score, 2025.

A Limitations and broader impacts

Limitations. While CCache improves the reliability of proxy-discrepancy caching, its effectiveness still depends on the quality of the underlying proxy signal. If the original proxy is severely misaligned with the oracle discrepancy across most timesteps, conformal calibration can reduce unsafe underestimation but may require larger corrections, potentially limiting the achievable acceleration. In addition, the validity of split conformal prediction relies on the exchangeability between calibration and evaluation prompts. Although we construct the calibration subset from the target evaluation prompt pool to better satisfy this assumption, substantial distribution shifts may still weaken the reliability of the calibrated correction. Moreover, our conformal guarantee is timestep-wise and marginal, and should not be interpreted as a formal trajectory-level coverage guarantee. We believe that stronger proxy designs, larger and more diverse calibration sets, and more advanced trajectory-level risk-control strategies could further improve the reliability and efficiency of CCache.

Broader impacts. Our work aims to improve the efficiency and reliability of generative model inference. By reducing unnecessary full network evaluations, CCache may help lower computational cost and energy consumption when deploying large image and video generation models. However, as with other acceleration techniques for generative models, improving inference efficiency may also make it easier to produce large volumes of synthetic content, including potentially misleading, biased, or harmful media. In addition, CCache inherits the social risks of the underlying generative models, such as biased visual outputs, unsafe content generation, and possible misuse in deepfake creation. Addressing these risks requires complementary safeguards, including dataset auditing, content moderation, watermarking, and responsible deployment policies.

B Appendix

B.1 Additional experimental details

Calibration Cost. CCache requires only a single standard no-cache generation run for each calibration prompt. During this run, the full-model outputs and the corresponding proxy features are recorded, allowing the oracle–proxy residuals to be computed. For Wan2.1-1.3B, collecting $N = 100$ calibration trajectories takes approximately 20 minutes on 8 NVIDIA H200 GPUs. Once these trajectories have been collected, sorting the residual scores and pre-extracting the timestep-wise quantiles for all considered coverage levels takes only a few seconds. This is just a one-time offline cost.

Tuning burden for matched-speed comparisons. The reported matched-speed comparisons require only limited tuning: the adaptive-update hyperparameters are fixed across all experiments, and only the cache threshold δ is adjusted to match the target runtime. Since reliability is our primary objective, we fix the target miscoverage level to $\alpha_0 = 0.05$, rather than tuning it separately for individual models, datasets, or baselines. We use $N = 100$ calibration samples, yielding an empirical quantile resolution of approximately $1/(N + 1) \approx 0.01$. Accordingly, we set $\eta = 0.2$, with $\alpha_{min} = 0.05$ and $\alpha_{max} = 0.5$ throughout all experiments. Under the proposed update rule, each non-violating segment increases the miscoverage level by 0.01, corresponding approximately to one empirical quantile level, whereas a violating segment decreases it by 0.19. Therefore, the correction is relaxed at a practically meaningful resolution but rapidly returns toward the conservative 95% coverage baseline once a violation is observed.

The only parameter adjusted for the matched-speed comparisons is the cache threshold δ. Increasing δ generally permits longer cache reuse and therefore increases the speedup, making runtime matching a one-dimensional search problem. Because cache refreshes are discrete threshold-crossing decisions, the exact latency may change in a piecewise manner rather than perfectly smoothly; nevertheless, the operating point can be localized reliably from the calibration statistics. **Specifically, we initialize the CCache threshold by increasing the baseline threshold by approximately $(\mathrm{Speedup} - 1)\bar{q}_{0.95}$, where $\bar{q}_{0.95}$ is the mean timestep-wise 95%-quantile correction.** The term $Speedup - 1$ approximates the expected number of cache-reuse steps per full model evaluation, so this initialization compensates for the additional conformal corrections accumulated during cache reuse. We then evaluate only one or two nearby threshold values on a small fixed pilot set, five prompts in our experiments, and select the value whose latency most closely matches that of the corresponding baseline.

An illustrative example is shown below:

Table 7: Illustration of cache-threshold adjustment for matched-speed comparison.

Method	δ	**Speedup**	**Latency**	**Method**	δ	**Speedup**	**Latency**	$\bar{q}_{0.95}$	**Estimated** δ
TeaCache	0.08	2.07×	46s	+ CCache	0.13	2.07×	46s	0.028	≈ 0.11
TeaCache	0.20	3.52×	27s	+ CCache	0.25	3.39×	28s	0.028	≈ 0.27

Transferability across resolutions and aspect ratios. Base on the T2I generation task and FLUX model, we apply the timestep-wise quantile corrections calibrated at the default 1024×1024 resolution directly to all seven resolution configurations, ranging from 512×512 to 1536×640, without any resolution-specific recalibration.

Table 8: Quality and efficiency comparison across resolutions.

	HPSv3 ↑				Latency (s) ↓			Speedup ↑	
Resolution	No-cache	DiCache	+CCache	Δ	No-cache	DiCache	+CCache	DiCache	+CCache
512×512	10.25	9.87	**10.17**	**+0.30**	2.39	1.07	**0.94**	2.24×	2.54×
768×768	10.35	10.51	**10.75**	**+0.25**	3.99	1.67	**1.42**	2.40×	2.81×
1024×1024	10.61	10.37	**10.84**	**+0.46**	5.66	**2.29**	2.32	2.47×	2.44×
1216×832	10.85	10.21	**10.61**	**+0.40**	6.82	2.74	**2.25**	2.49×	3.03×
1152×896	10.75	10.21	**10.62**	**+0.40**	8.30	2.77	**2.28**	3.00×	3.63×
1344×768	11.01	10.34	**10.72**	**+0.38**	6.95	2.76	**2.28**	2.52×	3.04×
1536×640	11.06	10.49	**10.85**	**+0.36**	8.84	2.64	**2.19**	3.35×	4.03×

As presented in Table 8, CCache consistently improves HPSv3 over DiCache across all tested resolutions and aspect ratios, with gains ranging from +0.25 to +0.46. These results indicate that the calibrated residual corrections are robust to substantial changes in spatial resolution and aspect ratio. In contrast, we do not recommend directly transferring the same correction vector across substantially different models, tasks, samplers, or sampling schedules. The validity of the calibrated correction relies on approximate exchangeability between the calibration and evaluation nonconformity scores. Such changes may alter the distribution of the proxy-to-oracle residuals and therefore violate the exchangeability assumption underlying the conformal calibration. Moreover, CCache uses timestep-indexed corrections, and different generation settings may have different trajectory lengths and timestep semantics.

B.2 Additional reliability analysis

Timestep- and Segment-Level Reliability Although the formal conformal guarantee of CCache is timestep-wise and marginal, this reflects a fundamental theory–practice trade-off in adaptive cache-based inference. A trajectory-level statement is substantially more challenging because the segment length is determined by a threshold-crossing stopping rule, discrepancies are dependent across timesteps, and cache reuse itself may alter the subsequent latent trajectory. Our current analysis therefore makes the more limited claim that, at each timestep, the conformal correction controls the one-sided proxy-underestimation risk under the stated exchangeability assumption.

Moreover, the refresh action is a discrete threshold decision. Small residual estimation errors that do not move the accumulated discrepancy across the refresh threshold leave the resulting cache schedule unchanged. This provides a degree of practical robustness away from the decision boundary.

We wish to emphasize that despite this marginal guarantee, our proposed CCache remains a powerful and generally applicable contribution. The consistent improvements across caching baselines, model scales, and image and video generation tasks provide empirical evidence that timestep-wise one-sided calibration is practically effective. This indicates that while the conformal guarantee is timestep-wise and marginal, the overall value and utility of our framework across different model scales and generation tasks are robust.

To directly quantify the practical risk, we conduct an oracle-instrumented on-policy evaluation on 100 held-out prompts using Wan2.1-T2V-1.3B in Table 9. We preserve the exact cached trajectory and refresh schedule, while additionally running the full Transformer at every timestep solely for measurement. For each skipped timestep, we record the adjacent-step oracle discrepancy and accumulate it within the current cache segment. A segment violation is counted only when this

discrepancy exceeds $\delta = 0.13$ before the actual refresh, meaning that the cache is genuinely reused after the oracle criterion has been violated.

Table 9: Empirical reliability of CCache under on-policy cached inference.

Empirical reliability metric	**Result**
Evaluated skipped Cond/Uncond passes	5,140
Passes satisfying the upper bound	4,580
Fixed-$q_{0.95}$ coverage on skipped passes	89.11%
Evaluated cache segments	1,366
Segment validity rate	93.70%
Unsafe reuse steps per violating segment	1.00

For the timestep-level diagnostic, 4,580 of the 5,140 skipped Cond/Uncond passes satisfy$\Delta_t^{\mathrm{ora}} \leq \Delta_t^{\mathrm{proxy}} + q_{t,0.95}$, yielding a fixed-$q_{0.95}$ conditional coverage of 89.11%. At the segment level, 93.70% of cache segments satisfy the cumulative oracle criterion throughout reuse. Moreover, when a violation occurs, the method refreshes after only one additional reuse step on average, indicating that the severity of the observed violations is limited. Together with the distributional analysis in Fig. 3, which shows a substantial reduction in the unsafe positive residual tail, this experiment provides complementary evidence at both the timestep and segment levels.

Trajectory Shift under Cached Inference. Calibration is performed on no-cache trajectories, whereas cache reuse may perturb the latent state encountered during accelerated inference. Consequently, the residual distribution encountered during accelerated inference may differ from that observed during no-cache calibration.

In practice, however, this shift is mitigated by the periodic refresh mechanism. Cache reuse is restricted to relatively short segments under the local temporal redundancy assumption. At each refresh timestep, the model performs a full network evaluation of the noise/velocity prediction at the current accelerated latent state and resets the cache anchor. This re-anchors the cached approximation to the current on-policy trajectory and prevents errors from an increasingly stale model output from accumulating indefinitely.

CCache further reacts to residual shift through its online adaptation. At each refresh point, full-model outputs are available for evaluating the reliability of the preceding cache segment. When the observed discrepancy exceeds the tolerance, CCache decreases the miscoverage level and applies a more conservative correction to subsequent segments; otherwise, it gradually relaxes the correction. Thus, although the initial quantiles are obtained from no-cache trajectories, the online controller can partially compensate when the residual behavior encountered during accelerated inference becomes less reliable.

The empirical results suggest that this periodic re-anchoring and adaptation are effective in practice. In particular, CCache continues to improve generation quality under aggressive caching settings, where cache-induced trajectory deviation is expected to be more pronounced. The distributional analysis in Figure 3 also shows that CCache substantially reduces the positive residual tail associated with unsafe underestimation.

B.3 Additional qualitative results

We provide additional qualitative results to complement the evaluation in Figure 4 and Figure 5, comparing caching baselines with and without CCache across diverse scenes and generation settings. The examples include human interactions, animal motion, and objects with intricate structures, illustrating improvements in subject appearance, object geometry, and fine visual details. For video generation, we present multiple frames sampled at identical temporal indices for both methods, allowing direct comparison at different stages of each sequence. The selected examples show that CCache can alleviate ghosting, structural distortions, and unintended duplication, and in some cases improve alignment with the prompt. Corresponding prompts and highlighted regions facilitate inspection of these differences.

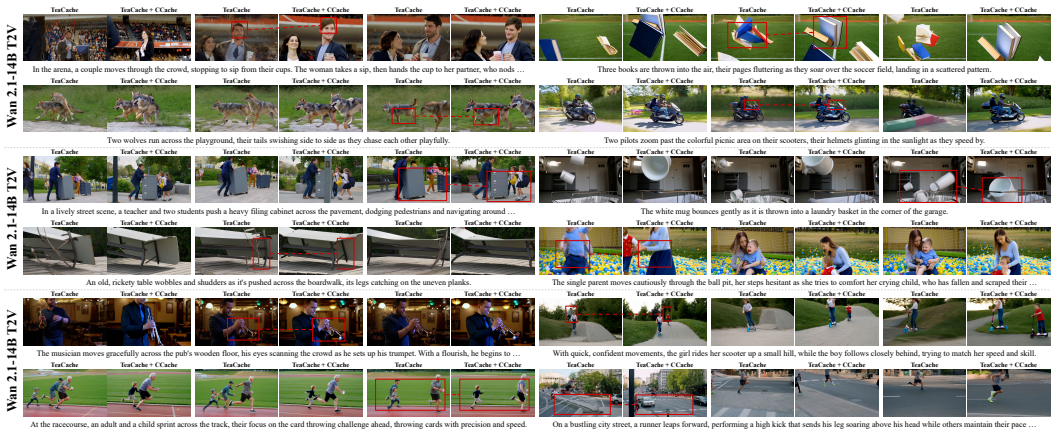


Figure 4: Additional qualitative comparisons of TeaCache with and without CCache on Wan2.1-T2V-14B. Red boxes highlight representative differences.

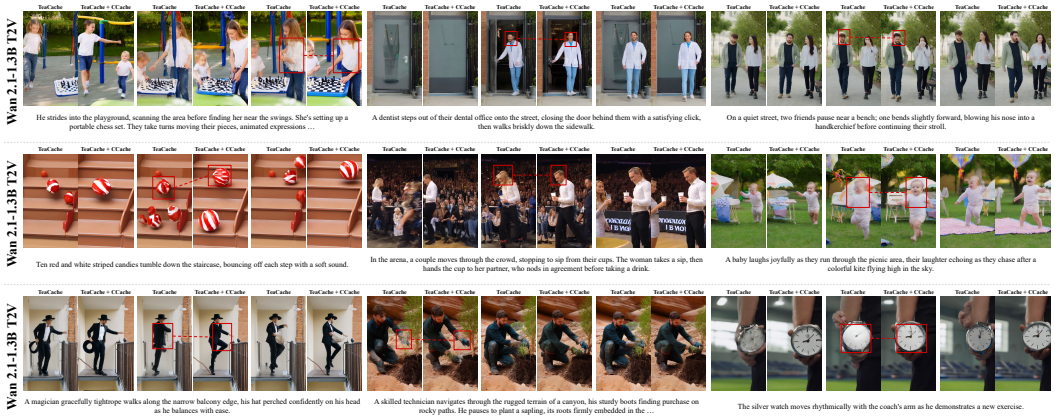


Figure 5: Additional qualitative comparisons of TeaCache with and without CCache on Wan2.1-T2V-1.3B. Red boxes highlight representative differences.

B.4 Additional quality evaluation

Full-Reference Fidelity and Perceptual Metrics To complement HPSv3 and VMBench, we conduct additional evaluations using both full-reference fidelity metrics and reference-free perceptual and text-alignment metrics. For the full-reference evaluation, we treat the output of the original no-cache model as the reference and compare it with the accelerated output. We report LPIPS, SSIM, and PSNR, which directly measure how faithfully each caching method preserves the output of the underlying generative model. As presented in Table 10, we additionally report ImageReward and CLIPScore to evaluate perceptual preference and prompt–image alignment, respectively.

Table 10: Additional fidelity and perceptual quality evaluation for TeaCache and its CCache-enhanced variant.

Method	LPIPS↓	SSIM↑	PSNR↑	HPSv3↑	ImageReward↑	CLIPScore↑
TeaCache ($\delta = 0.40$)	0.2715	0.7680	20.11	9.99	0.7953	0.8024
+ CCache	**0.1694**	**0.8462**	**24.03**	**10.13**	**0.8027**	**0.8026**
TeaCache ($\delta = 0.80$)	0.4370	0.6649	16.89	8.38	0.5819	0.7930
+ CCache	**0.3593**	**0.7267**	**19.89**	**9.54**	**0.6996**	**0.7999**

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction accurately summarize our main theoretical findings and empirical results.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We have discuss limitations in Appendix.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

Justification: We do not have theoretical results.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We provide all necessary details for reproduction, including hyperparameters, model architectures, and dataset configurations.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

Answer: [Yes]

Justification: All open-source models and datasets used in our experiments are properly cited.

Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes]

Justification: We have pecify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results in experimental settings.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

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

Answer: [No]

Justification: We cannot provide error bar for some competitors. Besides, the current results are sufficient to show the efficacy of our method.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We have provide sufficient information on the computer resources in experimental settings.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: Our research focuses on general visual tasks, which totally conforms with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We discuss both potential positive societal impacts and negative societal impacts of the work in the Appendix.

Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We cite all used public datasets and pre-trained models in Section 4.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [N/A]

    Justification: We do not introduce new assets.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: We do not include any crowdsourcing experiments and research with human subjects in our paper.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: We only conduct objective experiments in our paper.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: This work does not incorporate Large Language Models (LLMs) into its core methodology. Consequently, no description of such usage is provided

    Guidelines:

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