Training-free Detection of Generated Videos via Spatial-Temporal Likelihoods - Supplementary Material

Omer Ben Hayun, Roy Betser, Meir Yossef Levi, Levi Kassel, Guy Gilboa
Viterbi Faculty of Electrical and Computer Engineering
Technion – Israel Institute of Technology, Haifa, Israel

{omerben,roybe,me.levi,kassellevi}@campus.technion.ac.il; guy.gilboa@ee.technion.ac.il

Abstract

In this supplementary material document, we provide additional implementation details to ensure the full reproducibility of STALL. We also present extended explanations of the statistical tests used to assess the normality of embeddings and the uniformity of the temporal representation features. Furthermore, we include additional experiments and experimental details on all experiments. Next, we provide further details on the newly introduced synthetic dataset, ComGenVid*, and important details on the other used benchmarks. We conclude with an efficiency analysis, comparing our method to other zero-shot and supervised methods. The source code, dataset, and pre-computed whitening parameters are publicly available here.*

A. Reproducibility

A.1. Detailed algorithms

To ensure complete reproducibility, we provide full implementation details, including detailed algorithms for whitening (Algorithm 1), scores (Algorithm 2), (Algorithm 3), calibration (Algorithm 4), and inference (Algorithm 5).

A.1.1. Notation

- $\mathcal{C} = \{c^{(i)}\}_{i=1}^{N_c}$: Calibration set of N_c real videos
- $c^{(i)} = \{f_t^{(i)}\}_{t=1}^{T}$: Calibration video i consisting of T frames.
- $v = \{f_t\}_{t=1}^{T}$: query video with T frames.
- $E : \mathbb{R}^{H\times W\times 3} \to \mathbb{R}^d$: Image vision encoder.
- $x_t = E(f_t) \in \mathbb{R}^d$: Embedding of frame f_t.
- $\Delta_t = x_{t+1} - x_t$: Temporal difference between consecutive embeddings.
- $\tilde{\Delta}_t = \Delta_t / \|\Delta_t\|_2$: ℓ_2-normalized temporal difference.
- μ, W: Mean and whitening matrix for spatial embeddings
- μ_Δ, W_Δ: Mean and whitening matrix for temporal embeddings
- $s_{\text{spat}}, s_{\text{temp}}$ spatial and temporal scores.

A.1.2. Algorithms

Algorithm 1 Compute Whitening Transform

Require: Embeddings $\mathcal{X} = \{x_i\}_{i=1}^N \subset \mathbb{R}^d$

$\mu \leftarrow \frac{1}{N}\sum_{i=1}^N x_i$

$\hat{x}_i \leftarrow x_i - \mu$ for $i = 1, \ldots, N$

$\hat{\mathbf{X}} \leftarrow [\hat{x}_1, \ldots, \hat{x}_N]$

$\mathbf{\Sigma} \leftarrow \frac{1}{N}\hat{\mathbf{X}}\hat{\mathbf{X}}^\top$

Eigendecomposition: $\mathbf{\Sigma} = V\Lambda V^\top$

$W \leftarrow \Lambda^{-1/2}V^\top$

return μ, W

Algorithm 2 Compute Spatial Score

Require: Frame embeddings $\{x_t\}_{t=1}^T$, parameters (μ, W)

$y_t \leftarrow W(x_t - \mu)$ for $t = 1, \ldots, T$

$\ell_{\text{spat}}(t) \leftarrow -\frac{1}{2}(d\log(2\pi) + \|y_t\|_2^2)$ for $t = 1, \ldots, T$

$s_{\text{spat}} \leftarrow \max\{\ell_{\text{spat}}(t)\}_{t=1}^T$

return s_{spat}

Algorithm 3 Compute Temporal Score

Require: Frame embeddings $\{x_t\}_{t=1}^T$, parameters (μ_Δ, W_Δ)

$\Delta_t \leftarrow x_{t+1} - x_t$ for $t = 1, \ldots, T-1$ ▷ Temporal differences

$\tilde{\Delta}_t \leftarrow \frac{\Delta_t}{\|\Delta_t\|_2}$ for $t = 1, \ldots, T-1$ ▷ Normalization

$z_t \leftarrow W_\Delta(\tilde{\Delta}_t - \mu_\Delta)$ for $t = 1, \ldots, T-1$

$\ell_{\text{temp}}(t) \leftarrow -\frac{1}{2}(d\log(2\pi) + \|z_t\|_2^2)$ for $t = 1, \ldots, T-1$

$s_{\text{temp}} \leftarrow \min\{\ell_{\text{temp}}(t)\}_{t=1}^{T-1}$

return s_{temp}

Algorithm 4 STALL Calibration

Require: Calibration set $\mathcal{C} = \{c^{(i)}\}_{i=1}^{N_c}$ of N_c videos, each with T frames, encoder E

Encode all frames from calibration set:
for $i = 1$ to N_c **do**
  $x_t^{(i)} \leftarrow E(f_t^{(i)})$ for $t = 1, \ldots, T_i$
end for
Compute spatial whitening parameters:
$\mathcal{X}_{\text{spat}} \leftarrow \emptyset$
for $i = 1$ to N_c **do**
  Sample one frame: $x \sim \text{Uniform}(\{x_t^{(i)}\}_{t=1}^{T_i})$
  $\mathcal{X}_{\text{spat}} \leftarrow \mathcal{X}_{\text{spat}} \cup \{x\}$
end for
$(\mu, W) \leftarrow$ COMPUTE WHITENING TRANSFORM$(\mathcal{X}_{\text{spat}})$ ▷ Algorithm 1
Compute temporal whitening parameters:
$\mathcal{X}_{\text{temp}} \leftarrow \emptyset$
for $i = 1$ to N_c **do**
  for $t = 1$ to $T_i - 1$ **do**
    $\Delta_t \leftarrow x_{t+1}^{(i)} - x_t^{(i)}$
    $\tilde{\Delta}_t \leftarrow \frac{\Delta_t}{\|\Delta_t\|_2}$
    $\mathcal{X}_{\text{temp}} \leftarrow \mathcal{X}_{\text{temp}} \cup \{\tilde{\Delta}_t\}$
  end for
end for
$(\mu_\Delta, W_\Delta) \leftarrow$ COMPUTE WHITENING TRANSFORM$(\mathcal{X}_{\text{temp}})$ ▷ Algorithm 1
Compute calibration score distributions:
for $i = 1$ to N_c **do**
  $s_{\text{spat}}^{(i)} \leftarrow$ COMPUTE SPATIAL SCORE$(\{x_t^{(i)}\}_{t=1}^{T_i}, \mu, W)$ ▷ Algorithm 2
  $s_{\text{temp}}^{(i)} \leftarrow$ COMPUTE TEMPORAL SCORE$(\{x_t^{(i)}\}_{t=1}^{T_i}, \mu_\Delta, W_\Delta)$ ▷ Algorithm 3
end for
$\mathcal{S}_{\text{spat}} \leftarrow \{s_{\text{spat}}^{(i)}\}_{i=1}^{N_c}$
$\mathcal{S}_{\text{temp}} \leftarrow \{s_{\text{temp}}^{(i)}\}_{i=1}^{N_c}$
return $\mu, W, \mu_\Delta, W_\Delta, \mathcal{S}_{\text{spat}}, \mathcal{S}_{\text{temp}}$

Algorithm 5 STALL Inference

Require: Test video $v = \{f_t\}_{t=1}^{T}$, encoder E, calibration parameters $\mu, W, \mu_\Delta, W_\Delta, \mathcal{S}_{\text{spat}}, \mathcal{S}_{\text{temp}}$

// Encode all frames from test video
$x_t \leftarrow E(f_t)$ for $t = 1, \ldots, T$
Compute spatial and temporal scores:
$s_{\text{spat}} \leftarrow$ COMPUTE SPATIAL SCORE$(\{x_t\}_{t=1}^{T}, \mu, W)$ ▷ Algorithm 2
$s_{\text{temp}} \leftarrow$ COMPUTE TEMPORAL SCORE$(\{x_t\}_{t=1}^{T}, \mu_\Delta, W_\Delta)$ ▷ Algorithm 3
Compute percentile ranks from calibration distributions:
$\text{perc}_{\text{spat}} \leftarrow \frac{1}{N_c}|\{s \in \mathcal{S}_{\text{spat}} : s \leq s_{\text{spat}}\}|$
$\text{perc}_{\text{temp}} \leftarrow \frac{1}{N_c}|\{s \in \mathcal{S}_{\text{temp}} : s \leq s_{\text{temp}}\}|$
Combine percentiles for final detection score:
$s_{\text{video}} \leftarrow \frac{1}{2}(\text{perc}_{\text{spat}} + \text{perc}_{\text{temp}})$
return s_{video}

A.2. Implementation details

All of our experiments are conducted using Intel Core i9-7940X CPU and NVIDIA GeForce RTX 3090 GPU. For our method we use DINOv3 [38] as our encoder, available at `DINOv3 repository`. For all competing methods, we rely on the official

implementations when available, otherwise, we implement the corresponding baselines.

For AEROBLADE [37], we use the official implementation available at AEROBLADE repository. Since no official implementation is available for RIGID [26], we implement the method using the DINO [9] model. As there is no official implementation for ZED [16] either, we follow the lossless compression setup based on SReC [8]. ZED introduces four separate criteria, and we report results for the Δ^{01} criterion. For AIGVDet [4] and T2VE [1], we use the official code and pretrained weights released by the authors, available at AIGVDet repository and T2VE repository, respectively. For D3 [51], we rely on the official implementation at D3 repository, and use DINOv3 as the encoder.

A.3. Comparison Methodology

A.3.1. Video Preprocessing

In all of our evaluations, we filter out videos that are shorter than 2 seconds or have a frame rate below 8 FPS to ensure sufficient temporal coverage and quality. After filtering, we subsample frames to achieve a target frame rate of 8 FPS. Given an original frame rate f_{orig} and target rate $f_{\text{target}} = 8$ FPS, we compute the sampling ratio $r = f_{\text{orig}}/f_{\text{target}}$ and select every r-th frame (approximately).

Specifically, we maintain a continuous position that advances by r at each step, selecting the frame at the rounded position:

$$i_0 = 0, \quad i_j = \text{round}(r \cdot j), \quad j = 1, 2, \ldots$$

subject to $i_j < N$, where N is the total number of frames of the original video.

When f_{orig} is perfectly divisible by f_{target} (i.e., r is an integer), this reduces to uniform sampling of every r-th frame. For non-integer ratios, this approach selects frames with approximately uniform temporal spacing that best approximates the target frame rate. The corresponding Python implementation is provided below:

```python
def downsample_frames(num_frames, current_fps, target_fps=8):
    """Downsample frame indices to achieve target fps."""
    ratio = current_fps / target_fps
    indices = []
    j = 0

    while True:
        frame_idx = round(ratio * j)
        if frame_idx >= num_frames:
            break
        indices.append(frame_idx)
        j += 1

    return indices
```

Unless stated otherwise, all videos in our experiments were sampled at 8 FPS and truncated to 2 seconds, yielding 16 frames per video.

A.3.2. Pairwise Comparison Protocol

We conduct systematic pairwise comparisons between synthetic videos from each generative model and authentic videos. To ensure fair metric comparisons and to address data imbalance, we implement a balanced sampling procedure where we use equal numbers of real and generated videos, determined by the smaller class in each split. For each generative model M_i with N_i synthetic videos, we sample exactly N_i authentic videos from our real video datasets. Specifically, in the Videofeedback benchmark [24], the real videos are drawn from two datasets [3, 15]. To prevent bias from over-representation of any specific real dataset source, we sample an equal number of videos from both sources such that their total equals N_i.

A.4. D3 Baseline Evaluation

We identified two systematic differences between D3's [51] official evaluation protocol and ours that explain the performance gap reported in the main paper.

A.4.1. Unbalanced test set

When fewer than 1000 synthetic videos are available, 1000 real videos are compared against fewer synthetic samples (on GenVideo [13], this is most notably the case with Sora [34], which contains only 56 samples). Average Precision (AP) is sensitive to class imbalance and tends to inflate. Our pairwise comparison protocol (Section A.3.2) uses equal numbers of real and generated videos to ensure unbiased evaluation.

A.4.2. FPS upsampling by frame duplication

In the GenVideo [13] benchmark, the MSR-VTT [48] real videos provided on ModelScope are stored at only 3 FPS. D3's official code brings these to 8 FPS by duplicating frames. This duplication applies exclusively to real videos, because the generated videos in GenVideo are already at 8 FPS or higher. The resulting temporal redundancy inflates detection scores for methods that rely on inter-frame differences. In our evaluation, we download high-frame-rate MSR-VTT videos and uniformly downsample them to 8 FPS, which removes this artifact. for more details, see Sections A.3.1 and C.2.

A.4.3. D3 Ablation Results

In our main experiments, we evaluated D3 using the same embedder as our method (DinoV3 [38]) for consistency. For completeness, we now report D3 results using X-CLIP16 [33] as embedder, as conducted in the original D3 implementation, which yields a small performance increase. Table 1 shows how D3 mean AP on GenVideo varies across evaluation settings: choice of embedder, test-set balancing and sampling. Each protocol difference individually inflates AP, and their combination produces a large gap relative to our evaluation.

Table 1. D3 [51] mean AP across evaluation settings on GenVideo. The two protocol differences (class imbalance and FPS duplication) each inflate AP; together they fully explain the gap between D3's reported numbers and our evaluation.

Real video Sampling FPS	**Downsample** $\sim 27 \rightarrow 8$		**Upsample (Duplication)** $3 \rightarrow 8$	
Embedder	**Balanced**	**Unbalanced**	**Balanced**	**Unbalanced**
X-CLIP16 [33]	0.78	0.85	0.97	0.98
DinoV3 [38]	0.74	0.83	0.94	0.96

B. Normality measures and tests

B.1. Normality and uniformity tests

To assess whether the embeddings are approximately Gaussian, we apply two classical normality tests, Anderson–Darlin [2] and D'Agostino–Pearson [17], performed on each coordinate of the embeddings independently. Each test measures the normality of the one-dimensional vector input. Below we elaborate on each test and present results.

B.1.1. Anderson-Darling normality test

The Anderson-Darling (AD) test [2] can be viewed as a refinement of the classical Kolmogorov-Smirnov (KS) goodness-of-fit test [39], designed to put more weight on discrepancies in the tails of the distribution. Given a sample $X = \{x_1, x_2, \ldots, x_n\}$ and a target cumulative distribution function (CDF) of F (in our case, a normal CDF with parameters estimated from the data), the AD statistic is defined as

$$A^2 = -n - \sum_{i=1}^{n} \frac{2i-1}{n} \big[\ln F(x_i) + \ln\big(1 - F(x_{n+1-i})\big)\big], \tag{1}$$

where $x_1 \leq \cdots \leq x_n$ are the ordered sample values, $F(x)$ is the CDF of the reference normal distribution, and n is the sample size. Larger values of A^2 indicate stronger deviations from normality. these values are compared to tabulated critical values to decide whether to reject the Gaussian assumption.In our setting, we follow the conventional threshold $A^2 < 0.752$ as evidence to accept normality. To preform this test, we used `stats.anderson` function from `scipy` python package.

B.1.2. D'Agostino-Pearson Test

The D'Agostino-Pearson (DP) test [17] evaluates departures from normality by combining information about sample skewness and kurtosis. Let $X = \{x_1, x_2, \ldots, x_n\}$ be a univariate sample and let $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$ denote its mean. Let $m_i = \frac{1}{n}\sum_{j=1}^{n}(x_j - \mu)^i$ be the i-th centeral moment. The skewness g_1 and kurtosis g_2 are defined as:

$$g_1 = \frac{m_3}{m_2^{3/2}} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^3}{\big[\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2\big]^{3/2}} \tag{2}$$

$$g_2 = \frac{m_4}{m_2^2} - 3 = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^4}{\big[\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2\big]^2} - 3 \tag{3}$$

These two statistics are then transformed into approximately standard normal variables Z_1 and Z_2, and the DP test statistic is

$$K^2 = Z_1^2 + Z_2^2. \tag{4}$$

Under the null hypothesis of normality, K^2 approximately follows a chi square distribution with two degrees of freedom, so the p-value is

$$p = 1 - F_{\chi_2^2}(K^2), \tag{5}$$

where $F_{\chi_2^2}$ is the cumulative distribution function of χ_2^2. Positive g_1 indicates right-skewed data and negative g_1 left-skewed data, while positive g_2 corresponds to heavy tails and negative g_2 to light tails. As g_1 and g_2 approach zero, the test statistic K^2 decreases and the p-value increases, which is consistent with normality; in practice we treat $p > 0.05$ as compatible with a Gaussian distribution. We performed this test using the `stats.normaltest` function from the `scipy` Python package.

B.1.3. Results

To obtain stable estimates of normality, we randomly sample 40 independent groups of 250 embeddings each from the VATEX [44] calibration set, for both frame embeddings and frame embedding differences. As described and used in the main paper, we employ DINOv3 [38] as the frame level embedder, whose embedding space is 1024 dimensional. For every group and every coordinate, we apply the Anderson Darling (AD) and D'Agostino Pearson (DP) normality tests. We then aggregate the outcomes in two separate ways: (i) we average the test statistics across all coordinates and groups, and (ii) we compute the fraction of coordinates whose average statistic satisfies the normality thresholds ($A^2 < 0.752$ for AD, $p > 0.05$ for DP). As summarized in Table 2, raw frame embeddings x_t show high proportions of coordinates passing both tests, and whitening further increases these fractions. In contrast, raw temporal differences Δ_t are strongly non-Gaussian, with essentially no coordinates passing either criterion. After ℓ_2 normalization, temporal differences $\tilde{\Delta}_t$ exhibit high pass rates, and an additional whitening step z_t yields almost all coordinates satisfying both normality thresholds.

Table 2. **Normality tests.** Results of Anderson Darling (AD) and D'Agostino Pearson (DP) normality tests for different representations on the VATEX [44] calibration set. "Avg Score" is the mean test statistic across embedding coordinates and groups, "Threshold" specifies the acceptance condition for normality, and "Normal Features" is the percentage of coordinates satisfying that condition.

Representation	Test	Avg Score	Threshold	Normal Features (↑)	approximately gaussian?
Raw embeddings x_t	AD	0.4750	< 0.752	96.3%	✓
	DP	0.3931	> 0.05	99.2%	✓
Whitened embeddings y_t	AD	0.5034	< 0.752	98.2%	✓
	DP	0.3207	> 0.05	99.3%	✓
Transition vector $\Delta_t = x_{t+1} - x_t$	AD	3.3992	< 0.752	0.0%	✗
	DP	0.0093	> 0.05	0.0%	✗
ℓ_2 Normalized transition vector $\tilde{\Delta}_t$	AD	0.4134	< 0.752	98.4%	✓
	DP	0.4752	> 0.05	99.9%	✓
Whitened ℓ_2 normalized transition vector z_t	AD	0.4119	< 0.752	99.6%	✓
	DP	0.4648	> 0.05	100.0%	✓

B.2. Histogram comparisons

B.2.1. Raw vs. normalized temporal differences

To qualitatively illustrate these findings, Figure 1 presents histograms of temporal differences for the first four embedding dimensions of DINOv3 [38], computed from all adjacent frame pairs in the VATEX [44] calibration set. The left column shows raw temporal differences (Raw Δ_t), which exhibit significant deviations from the overlaid Gaussian distributions. In contrast, the right column shows ℓ_2-normalized temporal differences (Normalized Δ_t), which align closely with their corresponding Gaussian fits. This visual demonstration confirms the quantitative results in Table 2: ℓ_2 normalization transforms non-Gaussian temporal differences into approximately Gaussian distributions.

B.2.2. Real and AI-generated videos embeddings

To further illustrate the statistical properties of the embedding space, we visualize the univariate feature distributions of embeddings extracted from real and generated videos. Figure 2 shows histograms for the first four dimensions of DINOv3 [38] embeddings, computed from randomly sampled frames from real and generated videos in the GenVideo [13] dataset.

For each dimension, we overlay a Gaussian distribution fitted using the empirical mean and variance of the data (i.e., a moment-matched Gaussian). The empirical distributions closely follow the corresponding Gaussian curves for both real and generated samples. While the means and variances differ slightly between real and generated content, the overall shapes remain approximately Gaussian. These visualizations provide additional qualitative support for modeling the embedding dimensions using Gaussian statistics, which underlies the likelihood formulation used in our method.

B.3. Maxwell Poincare Lemma

Lemma 1 (Maxwell-Poincaré [18])**.** *Let U_d be uniform on $\mathbb{S}^{d-1}$ and fix $k \in \mathbb{N}$. Then*

$$\sqrt{d}(U_{d,1}, \ldots, U_{d,k}) \;\Rightarrow\; \mathcal{N}(0, I_k) \qquad (d \to \infty). \tag{6}$$

The rate at which this convergence occurs was quantified by Diaconis and Freedman [19]:

Theorem 2 (Maxwell-Poincaré convergence rate[19])**.** *If $1 \le k \le d-4$, then*

$$d_{\mathrm{TV}}\big(\sqrt{d}(U_{d,1}, \ldots, U_{d,k}), Z\big) \;\le\; \frac{2(k+3)}{d-k-3}\,, \tag{7}$$

where $Z \sim \mathcal{N}(0, I_k)$.

Note that both Lemma 1 and Theorem 2 extend naturally to arbitrary coordinate selections, or more generally to any k-dimensional orthonormal projection of U_d. When norms exhibit concentration behavior, an analogous result can be established:

Lemma 3 (Maxwell–Poincaré with norm concentration)**.** *Let U_d be as in Lemma 1 and fix $k \in \mathbb{N}$. Let $z_d = r_d U_d$ with $r_d \ge 0$. If $r_d \xrightarrow[d\to\infty]{\mathsf{P}} r_0$, then*

$$r_d \sqrt{d}\,(U_{d,1}, \ldots, U_{d,k}) \;\Rightarrow\; \mathcal{N}\big(0,\, r_0^2 I_k\big) \qquad (d \to \infty). \tag{8}$$

This is obtained by combining the Maxwell–Poincaré limit with radial concentration and applying Slutsky's theorem [41]. Figure 3 visualizes the distribution of the first coordinate of U_d, a random vector uniformly distributed on $\mathbb{S}^{d-1}$, for several dimensions d. As d increases, this distribution approaches the standard normal distribution.

Figure 4 shows histograms of cosine similarities over all unordered pairs of $3k0$ randomly sampled normalized temporal differences $\tilde{\Delta}_t \subset \mathbb{R}^{1024}$ from the VATEX [44] calibration set and of $3k$ points drawn uniformly from $\mathbb{S}^{1023}$.

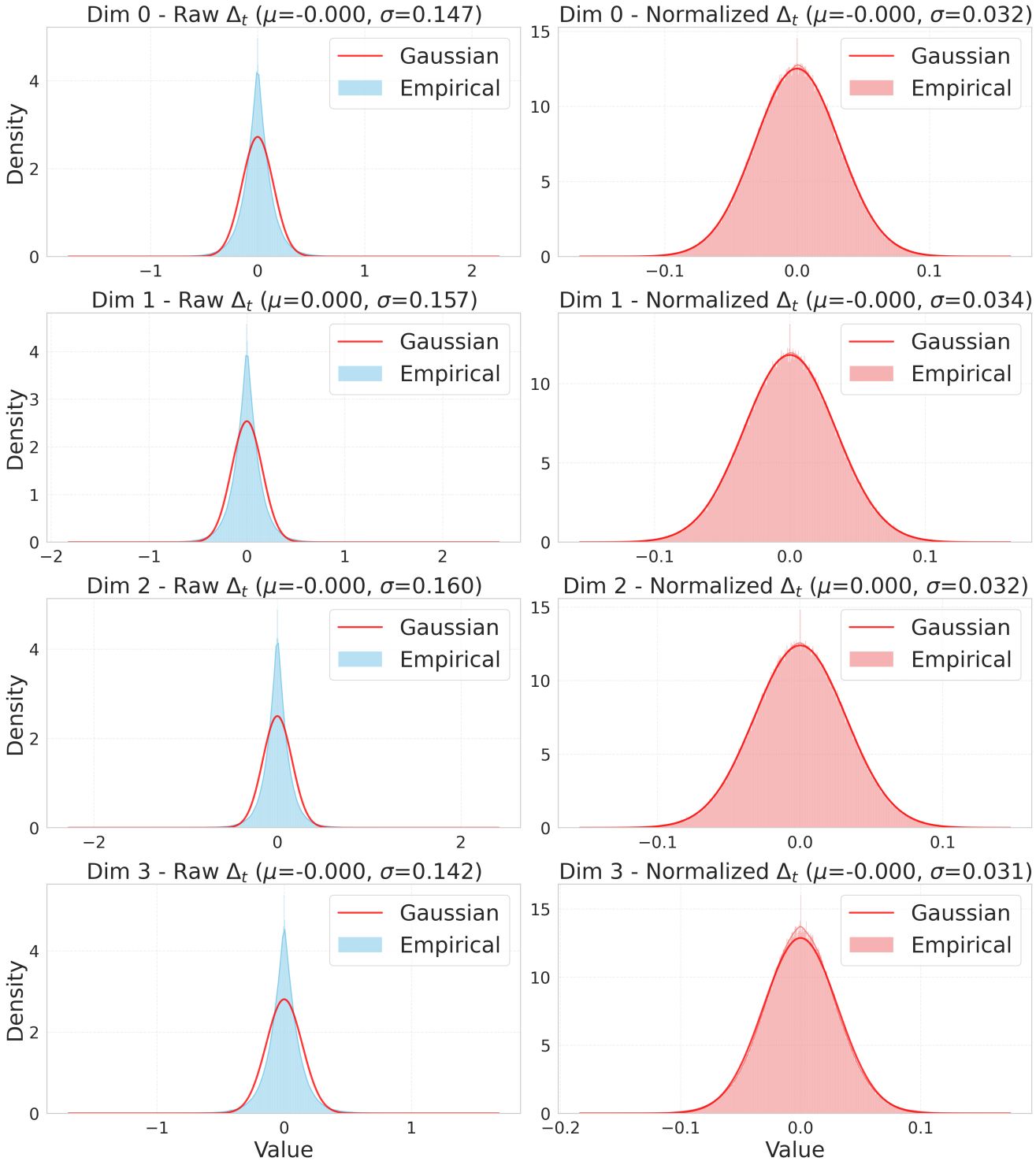


Figure 1. **Raw vs. Normalized Temporal Difference Histograms.** Histogram comparison of raw temporal differences (Raw Δ_t, left) versus normalized temporal differences (Normalized Δ_t, right) for the first four dimensions of DINOv3 [38] embeddings, computed from all adjacent frame pairs in the VATEX [44] calibration set. Red curves show Gaussian distributions fitted using the empirical mean and variance of the data (i.e., a moment-matched Gaussian). Even under this moment-matched fit, raw differences exhibit clear deviations from Gaussianity, while normalized differences closely match Gaussian distributions across all dimensions.

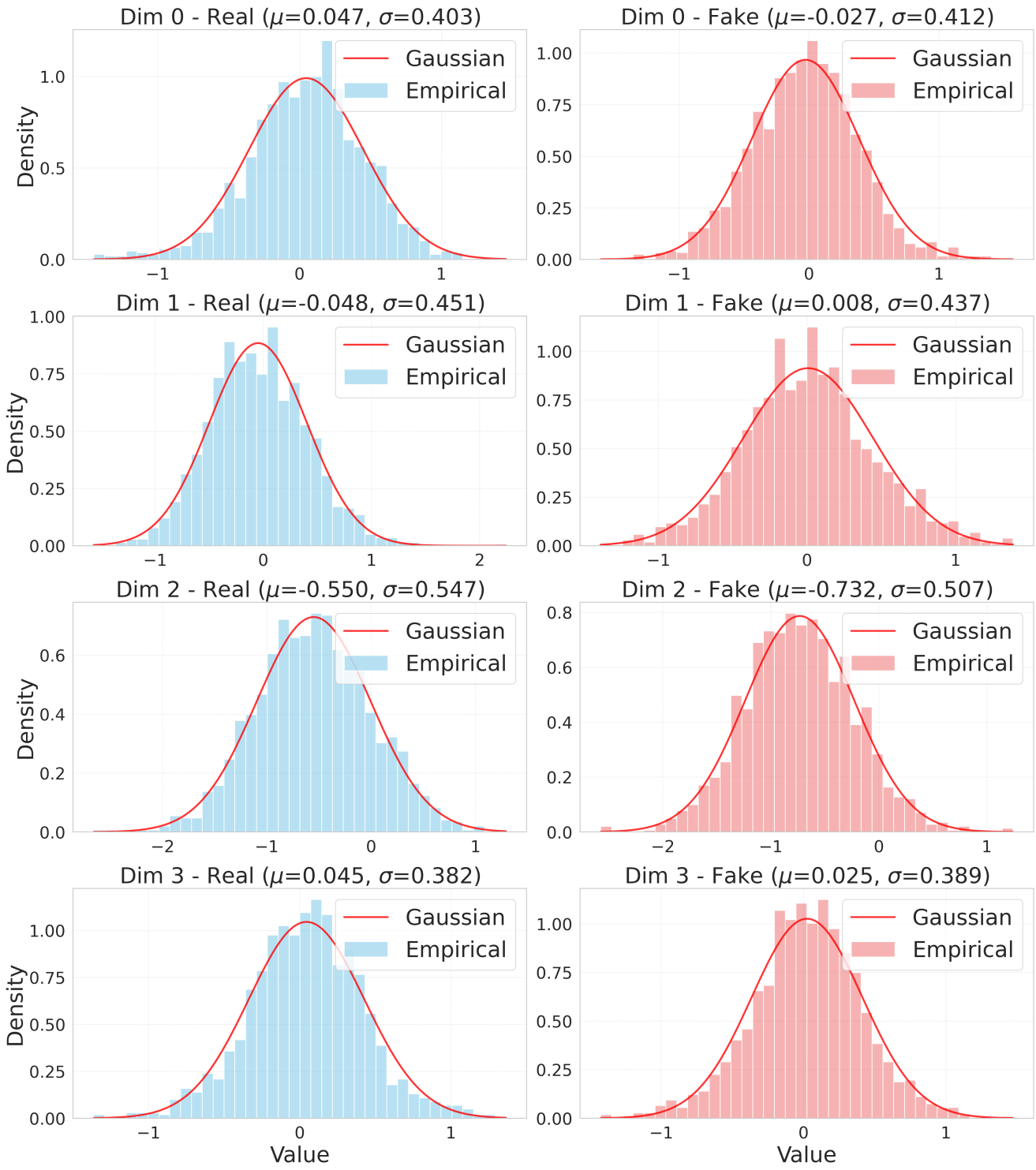


Figure 2. **Real and AI-generated videos embeddings.** Histograms of selected embedding dimensions from DINOv3 [38] features extracted from randomly sampled frames in the GenVideo [13] dataset. The left column corresponds to real videos, and the right column corresponds to generated videos. Red curves show Gaussian distributions fitted using the empirical mean and variance of each feature dimension.

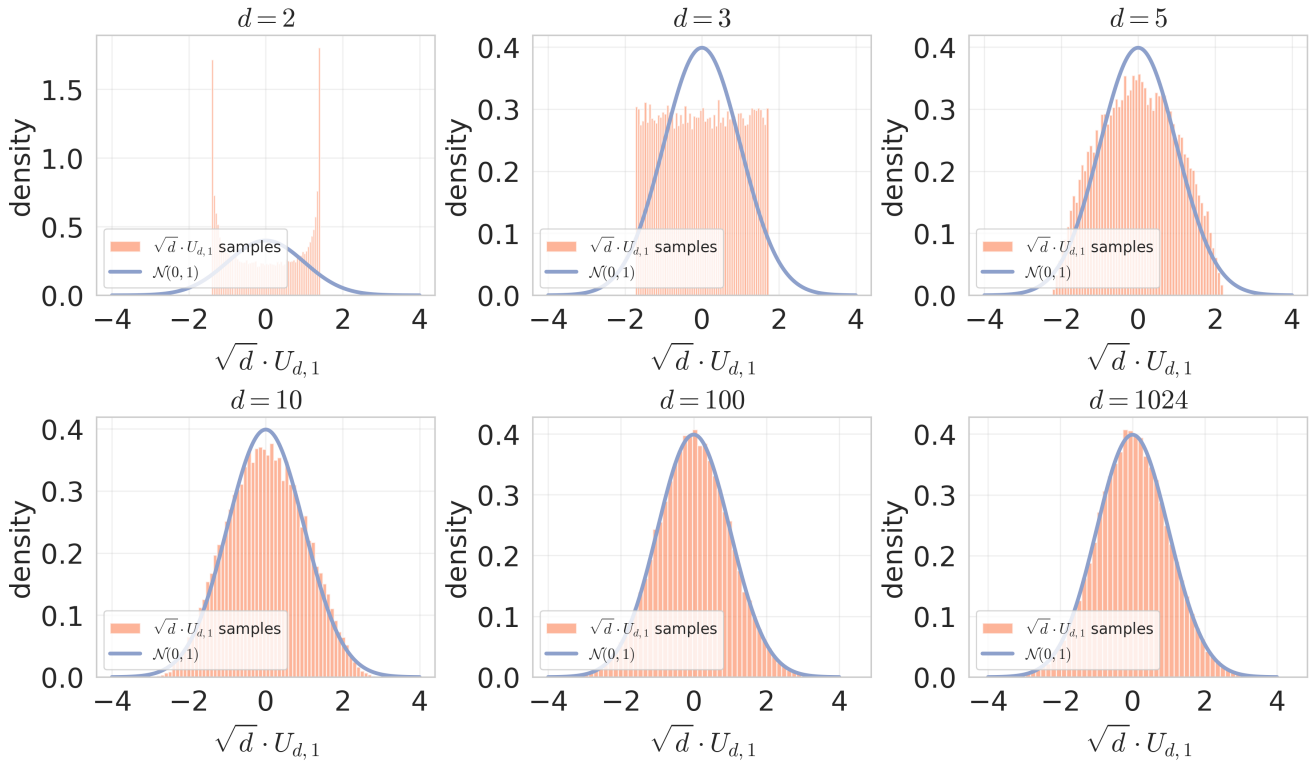


Figure 3. **Uniformity on the sphere and Gaussian distribution.** Histograms of the first coordinate of U_d, where U_d is uniformly distributed on the d-dimensional unit sphere $\mathbb{S}^{d-1}$, shown for several values of d and overlaid with the $\mathcal{N}(0,1)$ density.

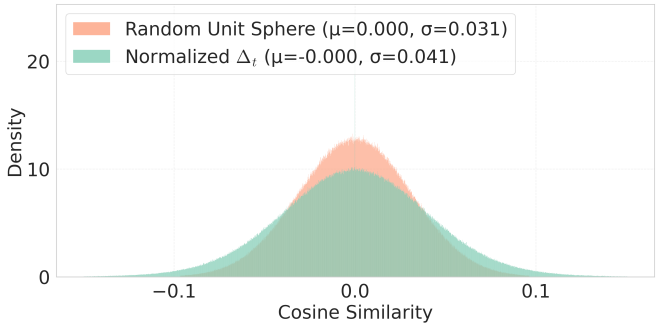


Figure 4. **Uniformity on the sphere.** Cosine similarity distributions over all unordered pairs of $3k$ randomly sampled normalized temporal differences $\tilde{\Delta}_t$ from the VATEX [44] calibration set and $3k$ points drawn uniformly from the unit sphere.

C. Datasets

C.1. ComGenVid benchmark

for full statistics of our *ComGenVid* benchmark, refer to table 3

MSVD Dataset. We obtained the complete MSVD dataset [11] from the sarthakjain004 Kaggle repository using the kaggle command line interface:

```
kaggle datasets download -d sarthakjain004/msvd-clips --unzip
```

We sampled 1.7k videos with 2 seconds length from this dataset.

The complete set of sampled videos is listed in `msvd_sampled_videos.csv`.

Generative Model Videos. To ensure fair comparison, we sampled 1.7k videos from both Sora [7] and Veo3 [21] to match the MSVD dataset size.

Veo3 Sampling. We randomly selected 1.7k videos from the ShareVeo3 dataset [43], available on the WenhaoWang Hugging Face repository, and can be download with the following python script:

```python
# pip install huggingface_hub[hf_xet]
from huggingface_hub import hf_hub_download

for i in range(1, 51):
    hf_hub_download(
        repo_id="WenhaoWang/ShareVeo3",
        filename=f"generated_videos_veo3_tar/veo3_videos_{i}.tar",
        repo_type="dataset"
    )
```

The complete list of sampled videos is provided in `veo3_sampled_videos.csv`.

Sora Sampling. We manually collected 1.7k videos from distinct users on the OpenAI Sora public explore feed. The complete list of sampled videos is provided in `sora_sampled_videos.csv`.

Table 3. **Overview of video sources and characteristics in the *ComGenVid* dataset.**

Video Source	Type	Length Range	Length (Mean±Std)	Resolution	Number of Pixels (Mean±Std)	FPS (Mean±Std)	**Total Count**
MSVD [11]	Real	2-60s	9.68±6.27s	160×112-1920×1080	0.29±0.35M	29.1±8.6	1700
Sora [34]	Fake	4-20s	6.01±2.26s	480×480-720×1080	0.36±0.05M	30.0±0.0	1700
VEO3 [21]	Fake	8s	8.00±0.00s	1280×720	0.92±0.00M	24.0±0.0	1700
Total Count	-	-	-	-	-	-	5100

C.2. GenVideo

We obtained the AI-generated videos for GenVideo [13] from the official ModelScope collection. Because the MSR-VTT [48] real videos available on ModelScope are limited to 3 FPS, we downloaded the original MSR-VTT dataset from the khoahunhtngng Kaggle repository to access higher-frame-rate versions.

All evaluated videos are uniformly sampled at 8 FPS for a 2-second duration (16 frames). The only exceptions are HotShot-XL and MoonValley [27, 31], which produce clips shorter than 2 seconds, these are compared against real 1-second videos at 8 FPS (8 frames). Generative models that produce less than 2-second clips are excluded from the ablation study. Because MSR-VTT contains far more real videos (10k) than any generative model, we selected 1.4k videos from it, following our pairwise comparison protocol (Section A.3.2). Comprehensive GenVideo benchmark statistics are provided in Table 4.

C.3. VideoFeedback

We gather the Videofeedback [24] dataset from the official Hugging Face repository. We evaluate only videos that are at least 2 seconds long, except for HotShot-XL [27], which generates 1 s clips and is therefore compared against real 1 s videos at 8 FPS (8 frames). in the original paper [24], each clip is assigned a dynamic-degree score (1–4) indicating how clearly its

Table 4. **Comprehensive statistics of the GenVideo [13] dataset used in our evaluations.**

Video Source	Type	Length Range	Length (Mean±Std)	Resolution	Number of Pixels (Mean±Std)	FPS Range	FPS (Mean±Std)	**Total Count**
MSR-VTT [48]	Real	10-30s	14.80±5.04s	320×240	0.08±0.00M	10-30	27.3±3.3	1400
Crafter [12]	Fake	2s	2.00±0.00s	1024×576	0.59±0.00M	8	8.0±0.0	188
Gen2 [20]	Fake	4s	4.00±0.00s	896×504-1408×768	0.77±0.31M	24	24.0±0.0	1380
Lavie [46]	Fake	2-3s	2.27±0.27s	512×320	0.16±0.00M	8-24	16.0±8.0	1400
ModelScope [42]	Fake	4s	4.00±0.00s	448×256-1280×720	0.70±0.36M	8	8.0±0.0	700
MorphStudio [32]	Fake	2s	2.00±0.00s	1024×576	0.59±0.00M	8	8.0±0.0	700
Show_1 [50]	Fake	4s	3.62±0.00s	576×320	0.18±0.00M	8	8.0±0.0	700
Sora [34]	Fake	9-60s	16.78±10.96s	512×512-1920×1088	1.43±0.65M	30	30.0±0.0	56
WildScrape [47]	Fake	2-251s	7.41±18.79s	256×256-2286×1120	0.48±0.41M	8-45	16.9±9.7	529
HotShot [27]	Fake	1s	1.00±0.00s	672×384	0.26±0.00M	8	8.0±0.0	700
MoonValley [31]	Fake	1.82s	1.82±0.00s	1184×672	0.80±0.00M	50	50.0±0.0	626
Total Count	-	-	-	-	-	-	-	8379

motion can be distinguished from a static image. We retain only the highest-scoring videos (levels 3–4). for the full statistics of the VideoFeedback benchmark, refer to Table 5.

Table 5. **Comprehensive statistics for the VideoFeedback [24] benchmark used in our evaluations.**

Video Source	Type	Length Range	Length (Mean±Std)	Resolution	Number of Pixels (Mean±Std)	FPS Range	FPS (Mean±Std)	**Total Count**
DiDeMo [3]	Real	3s	3.00±0.00s	352×288-640×1138	0.27±0.09M	8	8.0±0.0	1861
Panda70M [15]	Real	2-3s	2.38±0.48s	384×288-640×360	0.23±0.01M	8	8.0±0.0	1861
AnimateDiff [22]	Fake	2s	2.00±0.00s	512×512	0.26±0.00M	8	8.0±0.0	992
Fast-SVD [5]	Fake	3s	3.00±0.00s	768×432	0.33±0.00M	8	8.0±0.0	959
LVDM [25]	Fake	2s	2.00±0.00s	256×256	0.07±0.00M	8	8.0±0.0	2973
LaVie [46]	Fake	2s	2.00±0.00s	256×160-512×320	0.13±0.06M	8	8.0±0.0	2789
ModelScope [42]	Fake	2s	2.00±0.00s	256×256	0.07±0.00M	8	8.0±0.0	3722
Pika [36]	Fake	3s	3.00±0.00s	768×640	0.49±0.00M	8	8.0±0.0	1906
Sora [34]	Fake	2-3s	2.73±0.27s	512×512-1920×1088	1.25±0.58M	8	8.0±0.0	898
Text2Video [30]	Fake	2s	2.00±0.00s	256×256	0.07±0.00M	8	8.0±0.0	3722
VideoCrafter2 [14]	Fake	2s	2.00±0.00s	512×320	0.16±0.00M	8	8.0±0.0	3543
ZeroScope [10]	Fake	3s	3.00±0.00s	256×256	0.07±0.00M	8	8.0±0.0	2022
Hotshot-XL [27]	Fake	1s	1.00±0.00s	512×512	0.26±0.00M	8	8.0±0.0	2736
Total Count	-	-	-	-	-	-	-	29984

C.4. VATEX (Calibration set)

We obtained the VATEX dataset [44] from the khaledatef1's Kaggle repositories. The dataset is distributed across three parts: Vatex 1, Vatex 2, and Vatex 3. We downloaded all parts using the Kaggle command-line interface:

```
kaggle datasets download -d khaledatef1/vatex0110 --unzip
kaggle datasets download -d khaledatef1/vatex01101 --unzip
kaggle datasets download -d khaledatef1/vatex011011 --unzip
```

For the complete statistics of VATEX [44] calibaration set, see Table 6.

Table 6. **Comprehensive statistics of the VATEX [44] calibration set.**

Video Source	Type	Length Range	Length (Mean±Std)	Resolution	Number of Pixels (Mean±Std)	FPS Range	FPS (Mean±Std)	**Total Count**
VATEX [44]	Real	2-10s	9.68±1.10s	128×88-720×1280	0.44±0.37M	8-30	27.2±5.0	33976
Total Count	-	-	-	-	-	-	-	33976

C.5. Other datasets

We collected approximately 1.5k real videos from Kinetics400 [29] and PE [6]. The Kinetics400 clips were taken from the test split available in the cvdfoundation repository, while the PE videos were downloaded from the Facebook PE Hugging Face page. These datasets were used exclusively for the calibration set ablation experiment reported in Fig. 6(c) in the main paper. A detailed specification of this calibration set configuration is provided in Table 7. A complete list of the sampled videos is provided in `pe_kinetics400_sampled_videos.csv`.

Table 7. **Comprehensive statistics of the Kinetics+PE calibration set.**

Video Source	Type	Length Range	Length (Mean±Std)	Resolution	Number of Pixels (Mean±Std)	FPS Range	FPS (Mean±Std)	**Total Count**
Kinetics400 [29]	Real	2-10s	9.62±1.19s	128×96-1280×720	0.55±0.38M	24-30	26.9±3.0	1496
PE [6]	Real	5-60s	16.49±9.37s	608×254-608×1152	0.21±0.03M	24-60	37.7±14.5	1500
Total Count	-	-	-	-	-	-	-	2996

D. Additional results and experimental details

Unless stated otherwise, all ablation studies are conducted on the GenVideo [13] benchmark, restricted to generative-model videos sampled at 8 FPS and 2 seconds duration (16 frames). The only exceptions are HotShot-XL and MoonValley [27, 31], which generate videos shorter than 2 seconds and are therefore omitted from our ablation experiments. We use DINOv3 [38] as the frame-level embedder and the VATEX [44] dataset as the calibration set.

D.1. Derivative order ablations

We isolate the effect of higher-order temporal differences by keeping the entire STALL pipeline fixed and changing only the temporal derivative order $D \in \{1, 2, 3, 4\}$. For each video $v = \{f_t\}_{t=1}^{T}$, we extract frame-wise embeddings $E(v) \in \mathbb{R}^{T \times d}$, compute the D-th order finite-difference trajectory along time, and apply frame-wise ℓ_2 normalization. Importantly, we fit a separate whitening transform for each temporal derivative order, yielding parameters $(\mu_{\Delta=i}, W_{\Delta=i})$ for every $i \in D$, estimated on the corresponding derivative trajectories from the VATEX [44] calibration set. The temporal log-likelihood sequence for each video is aggregated using the same statistic as in the main STALL score, and we then average this temporal percentile with the spatial log-likelihood percentile to obtain a single score for each derivative order. An example implementation of the derivative and normalization computation is shown below:

```
def differences_vec(features):
    return features[:, 1:, :] - features[:, :-1, :]

def temporal_diff_with_order(features, derivative_order: int):
    """
    Apply a temporal finite-difference operator of the given order
    to frame-wise embeddings and L2-normalize the result.

    Args:
        frame_embeddings: array of shape [N, T, d]
        derivative_order: positive integer order of the derivative

    Returns:
        Array of shape [N, T - derivative_order, d].
    """
    if derivative_order < 1:
        raise ValueError("derivative_order must be positive")
    for _ in range(derivative_order):
        features = differences_vec(features)
    norms = np.linalg.norm(features, axis=-1, keepdims=True) + 1e-8
    return features / norms
```

Ablation results across all derivative orders, including AUC, Pearson correlation, and Spearman correlation, are summarized in Figure 5. All derivative orders are strongly correlated and produce very similar performance.

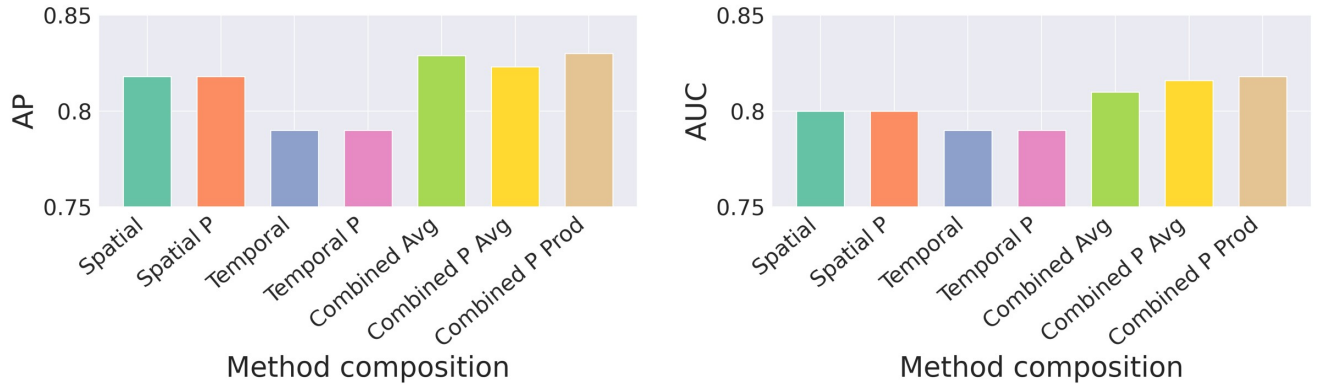
D.2. spatial and temporal aggregation methods

D.2.1. Components ablation

We analyze the contribution of the spatial and temporal branches by evaluating three detector variants: a spatial-only model, a temporal-only model, and a combined model that fuses both branches. For each variant, we consider raw scores and percentile-ranked scores, and for the combined model we also test mean-based and product-based fusions of the spatial and

(a) AUC - different derivative orders. (b) Pearson correlation. (c) Spearman correlation.

Figure 5. **Ablations on temporal derivative order.** STALL performance for different temporal derivative orders $D \in \{1, 2, 3, 4\}$: (a) AUC across orders, (b) Pearson correlation between scores from different orders, and (c) Spearman correlation between scores from different orders.



(a) AP - different components (b) AUC - different components

Figure 6. **Components ablation.** Bar plots of spatial-only, temporal-only, and combined detectors, using both raw scores and percentile-ranked scores. Spatial and Temporal denote single-branch detectors, and Combined refers to the fused Spatial and Temporal detector. Configurations labeled with "P" use percentile-ranked scores, where "P" stands for percentile. "Avg" stands for average of the two scores and "Prod" stands for the product of the two scores.

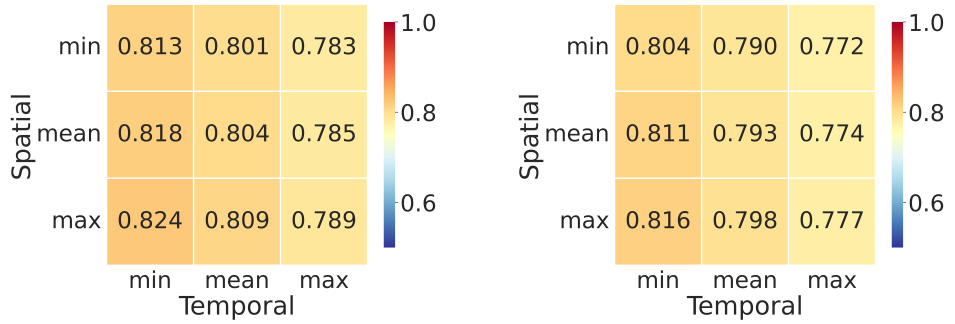
temporal percentile scores. For every configuration, we report the average AP and AUC across all three benchmarks. See results in Fig. 6.

D.2.2. Aggregation ablation

To analyze the effect of the frame-level aggregation operators, we fix our pipeline as defined in the main paper and vary only the frame-level aggregation used within each branch. We sweep over `min`, `mean`, and `max` for both the spatial and temporal components. For each spatial and temporal aggregation pair, we compute the average AUC and AP across all three benchmarks and report the resulting scores as heatmaps in Supp. Fig. 7.

D.3. Sampling a single frame difference for temporal whitening

In this ablation, we test whether temporal whitening requires all frame-to-frame transitions or can be reliably estimated from a single transition per video. We keep the spatial branch (and its whitening transform) fixed, and re-compute the temporal whitening statistics (μ_Δ, W_Δ) twice on the calibration set: (i) using all normalized frame differences from all videos, and (ii) using only one randomly sampled frame difference per video. Re-evaluating the combined score under these two temporal-whitening calibration variants yields essentially the same performance, with average AUC **0.8110** and average AP **0.8046** for (i) and average AUC **0.8105** and average AP **0.8044** for (ii) (Pearson correlation **0.9994**, Spearman correlation **0.9992** between (i) and (ii)), indicating that our temporal whitening is robust to the number of transitions used from each video for its estimation.



(a) AP for different spatial and temporal aggregation pairs.

(b) AUC for different spatial and temporal aggregation pairs.

Figure 7. **Frame-level aggregation ablation.** Heatmaps of AP and AUC, for different combinations of spatial and temporal frame-level aggregation operators. Rows correspond to the spatial aggregation and columns to the temporal aggregation, while all other parts of the pipeline follow the default configuration described in the main paper.

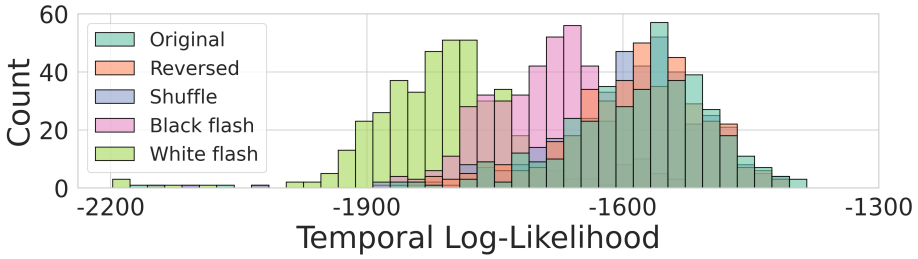


Figure 8. **Temporal scores under temporal perturbations.** Each condition applies a different perturbation to 400 real MSR-VTT videos: *Original* (no perturbation), *Reversed* (frames in reverse order), *Shuffle* (consecutive frames shuffled), *Black flash* (one black frame inserted mid-video), and *White flash* (one white frame inserted mid-video). The temporal likelihood is largely unaffected by reversal and shuffling, since adjacent-frame difference statistics are preserved; however, abrupt flash frames cause a strong drop in the score, with white flashes producing a larger degradation than black flashes.

D.4. Temporal likelihood under temporal perturbations

We analyze the temporal likelihood under realistic perturbations: reversing frame order (rewind), shuffling consecutive frames, and inserting black or white frames, representing data transmission issues [49]. As shown in Figure 8, the method is robust to reversal and shuffling, since the score depends only on statistics of adjacent-frame differences, which are largely preserved under these operations. In contrast, inserting a flash frame introduces an abrupt temporal inconsistency that strongly reduces the likelihood. The experiment is conducted on 400 real videos from MSR-VTT [48]. The temporal likelihood is stable under common temporal distortions but responds strongly to abrupt temporal anomalies.

D.5. Image perturbation experiment details

To explore robustness to perturbations, we apply four standard image corruptions to GenVideo [13] frames at inference time only, keeping the calibration set fixed. We use a reduced GenVideo subset with 250 videos per generative model and corrupt every video with each perturbation type at all five predefined severity levels. We then measure the impact of each corruption setting on detection performance. We additionally include level 0, which corresponds to uncorrupted frames (no perturbation). As shown in Fig. 7(b) of the main paper, STALL maintains strong separation across all perturbation types and severities.

The specific parameter settings for each perturbation type and level are summarized in Tab. 8. Gaussian blur severity is controlled by the blur radius r, with larger r producing stronger smoothing. JPEG compression reduces the image quality parameter, where lower quality introduces stronger compression artifacts. The random resized crop perturbation is parameterized by scale ranges, where a random crop is sampled at each application: sometimes removing more content and sometimes less. Higher severity levels use wider ranges with smaller minimum scales, increasing the chance of more aggressive crops. Gaussian noise severity is adjusted by increasing the standard deviation of zero-mean noise added to each frame, which

Table 8. **Image perturbation details.** These perturbations are used in the robustness ablation (Sec. D.5). Levels 1–5 span the full range of corruption strength for each type; level 0 corresponds to no perturbation.

Perturbation	Implementation	Severity levels (1→5)
Gaussian blur	`TF.gaussian_blur`	$(r, \sigma, k) =$ $\{(1, 0.5, 3), (2, 1.0, 5),$ $(3, 1.5, 7), (5, 2.5, 11),$ $(10, 5.0, 21)\}$
JPEG compression	`PIL.save(..., format="JPEG", quality)`	JPEG quality $q \in \{80, 50, 30, 10, 1\}$
Random resized crop	`transforms.RandomResizedCrop`	scale ranges $\{(0.85, 0.9), (0.7, 0.85),$ $(0.5, 0.8), (0.3, 0.9),$ $(0.08, 1.0)\}$
Gaussian noise	`torch.randn_like`, `torch.clamp`	$\sigma \in \{0.02, 0.05, 0.1, 0.2, 0.5\}$

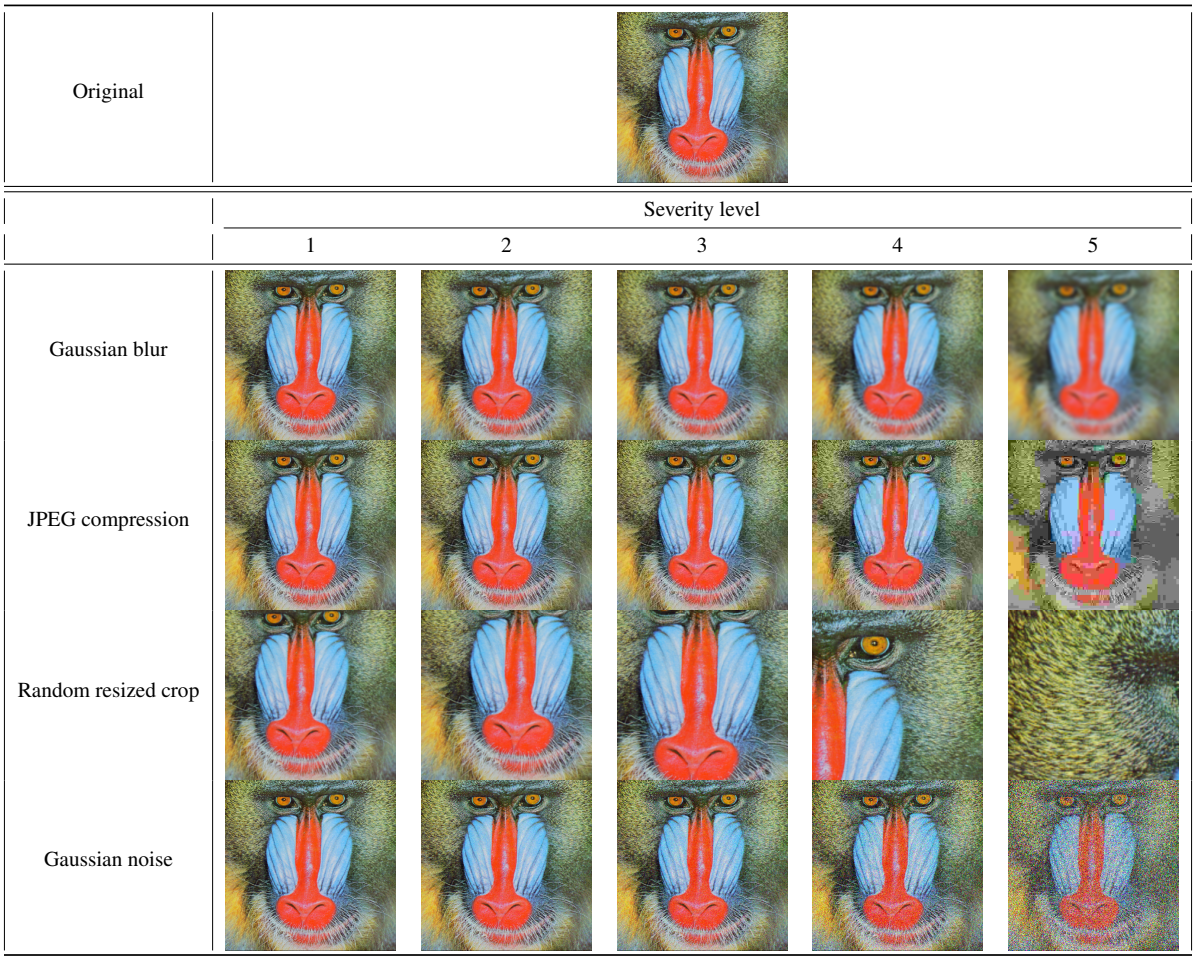


Figure 9. **Perturbations example.** mandrill.

progressively degrades fine texture while preserving global structure.

Perturbation examples can be found in Figs. 9 and 10.

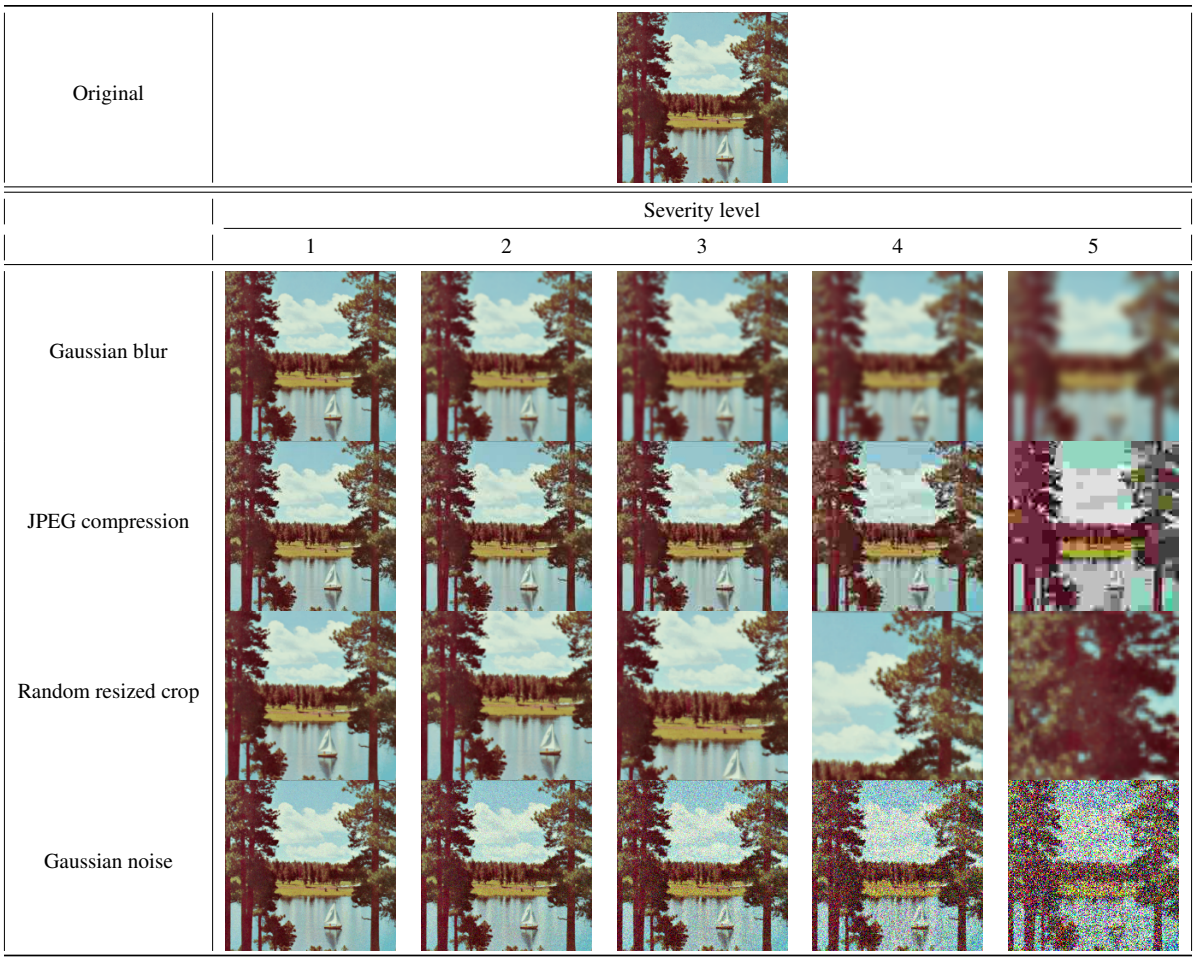


Figure 10. **Perturbations example.** Sailboat on lake.

D.6. Additional experimental details

D.6.1. Step size ablation

To assess sensitivity to temporal sampling rate, we vary the frame step size applied to videos already sampled at 8 FPS. For step size $s \in \{1, 2, 3, 4\}$, we subsample every s-th frame from all videos, reducing the effective frame rate by a factor of s (yielding 8, 4, 2.67, and 2 FPS respectively). Specifically, this subsampling is applied uniformly to both the VATEX [44] calibration set used to estimate the whitening transform and to the GenVideo [13] test videos being evaluated. We then compute the combined spatial and temporal score using these subsampled sequences. From another perspective, the setup can be viewed as using a larger temporal stride of s between successive differences, without overlap between stride segments in the original video. This ablation tests whether the detector remains robust when operating at lower effective frame rates, which is relevant for computational efficiency. The average AUC results are shown in Fig. 8(a) of the main paper.

D.6.2. FPS ablation

To evaluate robustness to different frame rates, we subsample only the inference videos while keeping the whitening transform fixed (estimated on VATEX [44] at 8 FPS as in all other experiments). We select videos from GenVideo [13] that are originally at 24 FPS with at least 2 seconds duration (425 videos from Gen2 [20], 425 from Lavie [46], and 42 from WildScrape [47], and a matching number of real videos from MSR-VTT [48]), isolating the effect of frame rate from other factors. In contrast to our standard 8 FPS setup, in this experiment we downsample the videos to target frame rates in $\{2, 4, 8, 12, 24\}$ FPS using

exact subsampling. The downsampling factor and target frame rate are chosen such that

$$\frac{\text{current\_fps}}{\text{target\_fps}} = n \in \mathbb{N},$$

so the downsampled sequence is obtained by retaining every n-th frame. This ensures deterministic frame selection without approximations. The score is computed on the downsampled sequences. Results, summarized in Fig. 8(c) of the main paper, show that performance is essentially unchanged across this range of frame rates, indicating that our method is robust to frame rate variation and that calibrating the whitening transform at 8 FPS does not degrade inference at other frame rates.

D.6.3. Length of video ablation

To evaluate robustness to video length, we follow a procedure similar in nature to the FPS ablation. We select videos from GenVideo [13] that were originally 4 seconds in duration (1380 videos from Gen2 [20], 700 from ModelScope [42], 214 from WildScrape [47], 56 from Sora [34], and 1400 real videos from MSR-VTT [48]), and then truncate each to $\{1, 2, 3, 4\}$ seconds. The whitening transform remains fixed throughout (estimated on VATEX [44] at 2 seconds as in all other experiments). The score is then computed on these truncated videos. Results, shown in Fig. 8(b) of the main paper, demonstrate that our method remains robust across this range of video durations, indicating that calibrating at 2 seconds does not degrade performance on shorter or longer clips.

D.6.4. Backbone encoders ablation

To assess the impact of the feature extractor, we evaluate STALL with five different backbone encoders: three image-based encoders and two video-based encoders. For image encoders, we use DINOv3 [38] (`dinov3_vitl16`), the lightweight MobileNetV3 [28] (`mobilenetv3_large_100` from `timm` python pacagke), and ResNet-18 [23] (from `torchvision.models`). For video encoders, we test VideoMAE [40] (`MCG-NJU/videomae-base`) and ViCLIP [45] (`OpenGVLab/ViCLIP-L-14-hf`), both from HuggingFace. All encoders are used with pretrained weights, and for image encoders we extract per-frame features and apply them independently to each frame in the video sequence. Results are presented in Table 2 of the main paper.

D.6.5. Calibration set size

To evaluate the sensitivity of our method to the size of the calibration set used for estimating the whitening transform, we systematically vary the number of videos sampled from VATEX [44]. We test calibration set sizes ranging from 1,000 to the full VATEX dataset (33,976 videos) in increments of 1,000 videos, evaluating each configuration across 5 random seeds to ensure statistical reliability. For each calibration set size, we randomly sample the specified number of videos from VATEX, estimate the whitening parameters on this subset, and then evaluate the resulting detector on the GenVideo [13] benchmark. All other pipeline components remain fixed, results are presented in Fig. 7(a) of the main paper.

D.6.6. Calibration set sources

To assess the impact of calibration set composition on detection performance, we evaluate STALL using five different calibration sets drawn from diverse real video sources. We test: (1) VATEX [44] (33,976 videos), (2) GenVideo [24] real subset from MSR-VTT [48] (8,584 videos, corresponding to all MSR-VTT clips excluded from the test set), (3) VideoFeedback [13] combining DiDeMo [3] (1k videos) and Panda70M [15] (1k videos), (4) a combination of Kinetics400 (1,496 videos) and PE [6] (around 1,500 videos from each, see section C.5 for more details), and (5) a balanced hybrid set sampling 1k videos from each of the six sources: MSR-VTT, PE, Panda70M, DiDeMo, VATEX, and Kinetics400. For each calibration set, we estimate the whitening transform using only videos from that set and evaluate the resulting detector on the GenVideo [13] benchmark. All other pipeline components remain fixed. Results are presented in Fig. 6(c) of the main paper.

Table 9 further reports average AUC across all three benchmarks for each calibration source (2K videos each), confirming stable performance across calibration choices.

Benchmark	VATEX [44]	Kinetics400 [29]	DiDeMo [3]	Panda-70M [15]	MSR-VTT [48]
VideoFeedback [24]	0.82	0.82	0.86	0.76	0.73
GenVideo [13]	0.78	0.77	0.76	0.83	0.83
ComGenVid	0.82	0.81	0.87	0.75	0.76

Table 9. Average AUC for different calibration datasets across all three benchmarks.

More broadly, the calibration set defines the reference feature statistics of the detector. If the test domain is not represented in the calibration set (e.g., surveillance or aerial videos), performance may degrade, which is a limitation of the method. Conversely, this also enables domain adaptiveness, as the detector can be adapted to a target domain by choosing an appropriate calibration set. Within the general video regime studied here, we observe stable behavior under calibration and test variation.

D.7. Additional qualitative results

Figure 11. **Qualitative examples.** Each row shows sampled frames from a video clip, with indicators marking whether its spatial and temporal behavior appears natural or unnatural.

Figure 12. **Qualitative examples.** Each row shows sampled frames from a video clip, with indicators marking whether its spatial and temporal behavior appears natural or unnatural.

E. Efficiency analysis

We conducted a comprehensive efficiency analysis to evaluate the computational performance of all detection methods, measuring model inference time and memory usage under controlled conditions.

E.1. Infrence time analysis

This experiment was performed on a fixed set of 20 videos. Each method ran inference on each video separately (without batching), and we repeated this process 5 times over the same video set to account for performance variability, yielding 100 total inference evaluations per method.

All methods were initialized before timing measurements to ensure a fair comparison. We used Python's `timeit.repeat` function to measure execution times, defining the inference time of each method to include video loading. This design ensures that the measured times reflect realistic end-to-end performance, covering both data loading and inference. The complete inference time analysis is provided in Table 10.

Table 10. **Inference time comparison for all methods.**

Domain	Method	Mean [sec]	Std [sec]
Zero shot images	AEROBLADE [37]	2.5266	0.0243
	ZED [16]	1.1394	0.0067
	RIGID [26]	0.4363	0.0024
Supervised video	T2VE [1]	1.9950	0.0102
	AIGVdet [4]	5.4216	0.0787
Zero-shot video	D3 cos [51]	0.2157	0.0043
	D3 L2 [51]	0.2220	0.0015
	STALL (ours)	0.2230	0.0010

E.2. memory analysis

To comprehensively evaluate the memory requirements of all methods, we conducted a profiling study that measures both model loading and inference memory consumption. Each method was executed in complete isolation within separate processes to eliminate any potential memory pollution or interference between measurements. We distinguished between two critical phases: (1) model loading memory, which captures the one-time cost of initializing model parameters and loading them onto CPU and GPU, and (2) inference memory, which measures the runtime memory footprint during actual video processing. For each method, we repeated measurements across multiple videos (10 videos × 3 repetitions) to ensure statistical reliability. Memory measurements were captured at two levels: CPU memory was tracked using `psutil` to monitor RAM consumption, while GPU memory was measured using PyTorch's [35] CUDA memory tracking facilities to capture peak memory usage. To ensure measurement accuracy, we performed garbage collection and CUDA cache clearing between measurements (`gc.collect()`) and GPU cache clearing (`torch.cuda.empty_cache()` and `torch.cuda.reset_peak_memory_stats()`) performed between each measurement to ensure clean memory states , with deliberate delays to allow the system to stabilize. A detailed analysis of memory consumption appears in Table 11.

Table 11. **Memory usage comparison for all methods**

domain	method	model loading cpu (MB)	model loading gpu (MB)	inference cpu peak (MB)	inference gpu peak (MB)
zero shot images	AEROBLADE [37]	7875.93	141.02	9711.18	2624.43
	ZED [16]	101.11	16.06	1707.72	310.08
	RIGID [26]	142.41	327.30	1373.67	567.57
supervised video	T2VE [1]	1852.16	1271.43	2795.96	140.96
	AIGVdet [4]	488.77	182.59	1849.03	673.18
zero-shot video	D3 cos [51]	315.64	1157.72	1470.08	160.30
	D3 l2 [51]	315.61	1157.72	1416.19	160.30
	STALL (ours)	321.21	1166.77	1647.60	160.30

The memory profiling results reveal significant variations in resource consumption across different detection approaches. Among zero-shot image methods, AEROBLADE [37] demonstrates substantially higher memory footprint during both loading and inference phases, while ZED [16] achieves the most efficient performance. In the supervised video category, T2VE [1] requires notably more GPU memory for model loading compared to AIGVdet [4], though the latter exhibits higher inference-time consumption. Our proposed STALL method, alongside D3 [51], maintains a balanced and efficient memory profile with moderate CPU and GPU usage across both phases, demonstrating comparable efficiency to existing temporal consistency zero-shot method while requiring significantly fewer resources than supervised alternatives. We note that all methods receive video data through CPU memory, which contributes to the observed inference CPU peak measurements across all approaches.

References

[1] 1129ljc. T2ve: Text-vision embedding for generalized ai-generated video detection. https://github.com/1129ljc/T2VE, 2025. GitHub repository. 4, 21

[2] Theodore W Anderson and Donald A Darling. A test of goodness of fit. *Journal of the American statistical association*, 49(268): 765–769, 1954. 6

[3] Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. Localizing moments in video with natural language. In *Proceedings of the IEEE international conference on computer vision*, pages 5803–5812, 2017. 4, 13, 19

[4] Jianfa Bai, Man Lin, Gang Cao, and Zijie Lou. Ai-generated video detection via spatial-temporal anomaly learning. In *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*, pages 460–470. Springer, 2024. 4, 21

[5] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023. 13

[6] Daniel Bolya, Po-Yao Huang, Peize Sun, Jang Hyun Cho, Andrea Madotto, Chen Wei, Tengyu Ma, Jiale Zhi, Jathushan Rajasegaran, Hanoona Rasheed, et al. Perception encoder: The best visual embeddings are not at the output of the network. *arXiv preprint arXiv:2504.13181*, 2025. 13, 14, 19

[7] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, et al. Video generation models as world simulators. *OpenAI Blog*, 1(8):1, 2024. 12

[8] Sheng Cao, Chao-Yuan Wu, and Philipp Krähenbühl. Lossless image compression through super-resolution, 2020. 4

[9] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 4

[10] Cerspense. Zeroscope v2 (576 w). https://huggingface.co/cerspense/zeroscope_v2_576w, 2024. 13

[11] David Chen and William B Dolan. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pages 190–200, 2011. 12

[12] Haoxin Chen, Menghan Xia, Yingqing He, Yong Zhang, Xiaodong Cun, Shaoshu Yang, Jinbo Xing, Yaofang Liu, Qifeng Chen, Xintao Wang, et al. Videocrafter1: Open diffusion models for high-quality video generation. *arXiv preprint arXiv:2310.19512*, 2023. 13

[13] Haoxing Chen, Yan Hong, Zizheng Huang, Zhuoer Xu, Zhangxuan Gu, Yaohui Li, Jun Lan, Huijia Zhu, Jianfu Zhang, Weiqiang Wang, et al. Demamba: Ai-generated video detection on million-scale genvideo benchmark. *arXiv preprint arXiv:2405.19707*, 2024. 5, 7, 10, 12, 13, 14, 16, 18, 19

[14] Haoxin Chen, Yong Zhang, Xiaodong Cun, Menghan Xia, Xintao Wang, Chao Weng, and Ying Shan. Videocrafter2: Overcoming data limitations for high-quality video diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7310–7320, 2024. 13

[15] Tsai-Shien Chen, Aliaksandr Siarohin, Willi Menapace, Ekaterina Deyneka, Hsiang-wei Chao, Byung Eun Jeon, Yuwei Fang, Hsin-Ying Lee, Jian Ren, Ming-Hsuan Yang, et al. Panda-70m: Captioning 70m videos with multiple cross-modality teachers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13320–13331, 2024. 4, 13, 19

[16] Davide Cozzolino, Giovanni Poggi, Matthias Nießner, and Luisa Verdoliva. Zero-shot detection of ai-generated images. In *European Conference on Computer Vision*, pages 54–72. Springer, 2024. 4, 21

[17] Ralph D'agostino and Egon S Pearson. Tests for departure from normality. empirical results for the distributions of b^2 and $\sqrt{b}$. *Biometrika*, 60(3):613–622, 1973. 6

[18] Persi Diaconis and David Freedman. Asymptotics of graphical projection pursuit. *The annals of statistics*, pages 793–815, 1984. 7

[19] Persi Diaconis and David Freedman. A dozen de finetti-style results in search of a theory. In *Annales de l'IHP Probabilités et statistiques*, pages 397–423, 1987. 7

[20] Patrick Esser, Johnathan Chiu, Parmida Atighehchian, Jonathan Granskog, and Anastasis Germanidis. Structure and content-guided video synthesis with diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7346–7356, 2023. 13, 18, 19

[21] DeepMind / Google. Veo 3: Google deepmind's third-generation text-to-video model. Online technical report, 2025. 12

[22] Yuwei Guo, Ceyuan Yang, Anyi Rao, Zhengyang Liang, Yaohui Wang, Yu Qiao, Maneesh Agrawala, Dahua Lin, and Bo Dai. Animatediff: Animate your personalized text-to-image diffusion models without specific tuning. *arXiv preprint arXiv:2307.04725*, 2023. 13

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 19

[24] Xuan He, Dongfu Jiang, Ge Zhang, Max Ku, Achint Soni, Sherman Siu, Haonan Chen, Abhranil Chandra, Ziyan Jiang, Aaran Arulraj, Kai Wang, Quy Duc Do, Yuansheng Ni, Bohan Lyu, Yaswanth Narsupalli, Rongqi Fan, Zhiheng Lyu, Yuchen Lin, and Wenhu Chen. Videoscore: Building automatic metrics to simulate fine-grained human feedback for video generation. *ArXiv*, abs/2406.15252, 2024. 4, 12, 13, 19

[25] Yingqing He, Tianyu Yang, Yong Zhang, Ying Shan, and Qifeng Chen. Latent video diffusion models for high-fidelity long video generation. *arXiv preprint arXiv:2211.13221*, 2022. 13

[26] Zhiyuan He, Pin-Yu Chen, and Tsung-Yi Ho. Rigid: A training-free and model-agnostic framework for robust ai-generated image detection. *arXiv preprint arXiv:2405.20112*, 2024. 4, 21

[27] HotshotCo. Hotshot-xl. https://github.com/hotshotco/hotshot-xl, 2023. 12, 13, 14

[28] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1314–1324, 2019. 19

[29] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 13, 14, 19

[30] Levon Khachatryan, Andranik Movsisyan, Vahram Tadevosyan, Roberto Henschel, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. Text2video-zero: Text-to-image diffusion models are zero-shot video generators. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15954–15964, 2023. 13

[31] MoonValley. Moonvalley. https://moonvalley.ai/, 2022. 12, 13, 14

[32] MorphStudio. Morphstudio. https://www.morphstudio.com/, 2023. 13

[33] Bolin Ni, Houwen Peng, Minghao Chen, Songyang Zhang, Gaofeng Meng, Jianlong Fu, Shiming Xiang, and Haibin Ling. Expanding language-image pretrained models for general video recognition, 2022. 5

[34] OpenAI. Video generation models as world simulators – introducing sora. Online technical report, 2024. 5, 12, 13, 19

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 21

[36] Pika Labs. Pika. https://pika.art/, 2023. 13

[37] Jonas Ricker, Denis Lukovnikov, and Asja Fischer. Aeroblade: Training-free detection of latent diffusion images using autoencoder reconstruction error. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9130–9140, 2024. 4, 21

[38] Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. Dinov3, 2025. 3, 5, 6, 7, 9, 10, 14, 19

[39] Nikolai V Smirnov. On the estimation of the discrepancy between empirical curves of distribution for two independent samples. *Bull. Math. Univ. Moscou*, 2(2):3–14, 1939. 6

[40] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in neural information processing systems*, 35:10078–10093, 2022. 19

[41] Aad W Van der Vaart. *Asymptotic statistics*. Cambridge university press, 2000. 8

[42] Jiuniu Wang, Hangjie Yuan, Dayou Chen, Yingya Zhang, Xiang Wang, and Shiwei Zhang. Modelscope text-to-video technical report. *arXiv preprint arXiv:2308.06571*, 2023. 13, 19

[43] Wenhao Wang and Yi Yang. Vidprom: A million-scale real prompt-gallery dataset for text-to-video diffusion models. 2024. 12

[44] Xin Wang, Jiawei Wu, Junkun Chen, Lei Li, Yuan-Fang Wang, and William Yang Wang. Vatex: A large-scale, high-quality multilingual dataset for video-and-language research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4581–4591, 2019. 6, 7, 8, 9, 11, 13, 14, 18, 19

[45] Yi Wang, Yinan He, Yizhuo Li, Kunchang Li, Jiashuo Yu, Xin Ma, Xinhao Li, Guo Chen, Xinyuan Chen, Yaohui Wang, et al. Internvid: A large-scale video-text dataset for multimodal understanding and generation. *arXiv preprint arXiv:2307.06942*, 2023. 19

[46] Yaohui Wang, Xinyuan Chen, Xin Ma, Shangchen Zhou, Ziqi Huang, Yi Wang, Ceyuan Yang, Yinan He, Jiashuo Yu, Peiqing Yang, et al. Lavie: High-quality video generation with cascaded latent diffusion models. *International Journal of Computer Vision*, 133(5): 3059–3078, 2025. 13, 18

[47] Yujie Wei, Shiwei Zhang, Zhiwu Qing, Hangjie Yuan, Zhiheng Liu, Yu Liu, Yingya Zhang, Jingren Zhou, and Hongming Shan. Dreamvideo: Composing your dream videos with customized subject and motion, 2023. 13, 18, 19

[48] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. Msr-vtt: A large video description dataset for bridging video and language. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5288–5296, 2016. 5, 12, 13, 16, 18, 19

[49] Runhao Zeng, Xiaoyong Chen, Jiaming Liang, Huisi Wu, Guangzhong Cao, and Yong Guo. Benchmarking the robustness of temporal action detection models against temporal corruptions. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2024. 16

[50] David Junhao Zhang, Jay Zhangjie Wu, Jia-Wei Liu, Rui Zhao, Lingmin Ran, Yuchao Gu, Difei Gao, and Mike Zheng Shou. Show-1: Marrying pixel and latent diffusion models for text-to-video generation. *International Journal of Computer Vision*, 133(4): 1879–1893, 2025. 13

[51] Chende Zheng, Ruiqi Suo, Chenhao Lin, Zhengyu Zhao, Le Yang, Shuai Liu, Minghui Yang, Cong Wang, and Chao Shen. D3: Training-free ai-generated video detection using second-order features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12852–12862, 2025. 4, 5, 21