Adaptive Latent Trajectory Anchoring for Action Segmentation Dataset Condensation

Arthème Gauthier-Villars[1,2*], Guodong Ding[1*,†], and Angela Yao[1]

[1] National University of Singapore, Singapore
[2] ETH Zurich, Switzerland
agauthier@ethz.ch {dinggd,ayao}@comp.nus.edu.sg

Abstract. Dataset condensation for action segmentation synthesizes compact, informative representations of long, untrimmed video datasets. The existing approach relies on Variational Autoencoders and an iterative latent optimization; it is computationally expensive and suffers from over-smoothed reconstructions and rigid temporal constraints. This paper proposes to shift the condensation paradigm from optimization-based inversion to deterministic latent mapping. By leveraging Denoising Diffusion Implicit Models, we represent action segments as continuous trajectories anchored by sparse latent points in the noise manifold. To maximize representational efficiency, we introduce an adaptive allocation mechanism that dynamically redistributes the anchoring budget based on segment-wise reconstruction difficulty. Extensive experiments demonstrate that our framework significantly outperforms state-of-the-art methods in segmentation performance across common datasets. Notably, our approach achieves performance parity with real data training while maintaining a condensation ratio of 2.4% on Breakfast dataset.

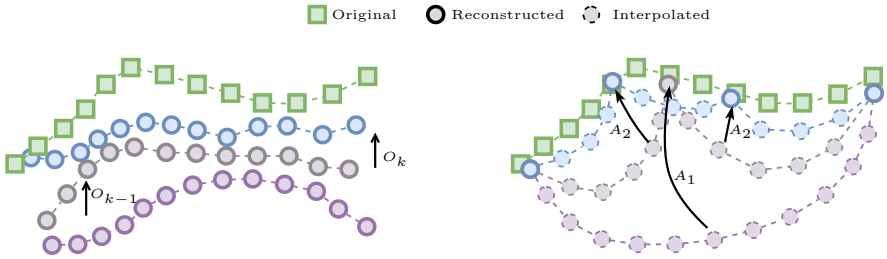
Keywords: Dataset Condensation · Temporal Action Segmentation · Diffusion Models

1 Introduction

Temporal action segmentation (TAS) [7] targets the task of predicting a semantic label for every frame in a long, untrimmed video. Despite the success of modern TAS architectures [10, 17, 31], their performance remains heavily dependent on the availability of massive, densely-labeled datasets, which introduces significant bottlenecks in terms of storage and training efficiency. To address this, dataset condensation [29] has emerged as a promising direction, seeking to synthesize a compact, highly informative representation of the original data.

A recent pioneering study formulates this problem as generative network inversion [5]. A conditional Variational Autoencoder (cVAE) [14] is first trained

* Equal contribution.
† Corresponding author and project lead.



(a) Direct Optimization [5] (b) Latent Anchoring (Ours)

Fig. 1: Comparison of TAS condensation paradigms. (a) Direct Optimization: An existing method uses iterative optimizations (O_k, O_{k-1}) to find optimal latent codes for condensation and rely on these fixed codes for reconstruction, which can cause low reconstruction fidelity. (b) Latent Anchoring: Our method adaptively selects latent anchors (A_1, A_2) per segment for condensation and uses latent trajectory interpolation to enable reconstruction of fine grained action dynamics.

to model action priors in the feature space and each action segment is then reconstructed by optimizing latent variables to minimize reconstruction error. While effective, this strategy inherits structural constraints: reconstruction fidelity depends on the expressiveness of a variational latent space trained under a unimodal Gaussian prior, and condensation requires iterative per-segment optimization. As a result, fine temporal variations may be over-smoothed. More importantly, the same latent code is reused across consecutive frames during reconstruction, imposing a block-wise temporal structure that may even obscure the continuous evolution of action dynamics.

In this work, we depart from the inversion perspective and revisit TAS condensation through the lens of generative dynamics. Our central observation is that action segments are not arbitrary collections of feature vectors; they exhibit structured, smooth trajectories in representation space. A desirable condensation mechanism should therefore preserve this trajectory structure rather than compress segments into isolated random codes. This requirement calls for a generative model whose latent space admits a deterministic and structure-preserving mapping between data and noise. Deterministic diffusion models, in particular DDIMs [25], are a natural fit. Unlike cVAEs, DDIM defines a deterministic reverse process that establishes an almost bijective mapping between a data sample and its corresponding noise. As a result, any input can be mapped to its corresponding latent, enabling high-fidelity reconstruction without the need for per-instance optimization.

Building on this property, we propose a diffusion-based TAS condensation framework that reformulates compression as a latent trajectory anchoring problem. We represent action segments as continuous trajectories in diffusion latent space, anchored by sparse, informative points. Specifically, each action segment is encoded into the latent trajectory induced by the deterministic DDIM reverse process. To achieve condensation, we store each segment as a sparse collection of

latent anchors that characterizes the trajectory. During reconstruction, the full feature sequence can be recovered by interpolating between latent anchors.

Going a step further, we introduce an adaptive anchoring strategy that allocates anchors according to segment-specific reconstruction difficulty by dynamically adjusting anchor density along the trajectory. In particular, segments exhibiting more complex temporal variations are assigned higher anchor density to better preserve reconstruction fidelity, while simpler segments are represented using fewer anchors for better efficiency.

Our proposed framework offers several advantages. First, by leveraging deterministic diffusion trajectories, the condensation process avoids expensive per-segment latent optimization while still supporting faithful sequence reconstruction. Second, interpolation between the latent anchors helps reconstruct fine-grained temporal variations as oppose to decoding from repeated latent codes. Third, the adaptive anchoring strategy further improves efficiency by allocating higher anchor density to segments with higher reconstruction difficulty. A conceptual comparison between the existing optimization-based paradigm [5] and our proposed latent anchoring framework is provided in Fig. 1.

Contributions. Our contributions are summarized as follows: 1) We offer a novel perspective for TAS condensation by recasting the problem as a generative latent trajectory reconstruction task through the lens of diffusion models. This shift enables high-fidelity feature synthesis while bypassing iterative inversion. 2) We propose latent trajectory anchoring with latent-space interpolation as a replacement for discrete code instantiation, modeling action segments as continuous trajectories that preserve fine-grained temporal dynamics. 3) We develop an adaptive budgeting strategy that redistributes representation capacity based on reconstruction difficulty, ensuring scalability and flexibility across diverse action lengths. 4) We demonstrate that our framework achieves significant dataset condensation, with state-of-the-art performance across major benchmarks.

2 Related Work

Temporal Action Segmentation. TAS requires the frame-level labeling of action sequences in untrimmed videos [7]. The field primarily focuses on architecture, supervision, and training paradigms. Architectural designs prioritize long-range temporal modeling, spanning dilated convolutional networks (MSTCN [10]), transformer-based architectures (ASFormer [31]), and diffusion-based generative frameworks (DiffAct [17]). To mitigate annotation costs, diverse supervision regimes have been studied ranging from fully supervised [10, 17, 19, 24, 31], semi-supervised [8, 23] to weakly supervised [9, 21] and unsupervised learning [1, 16]. Furthermore, the scope extends to specialized settings including active learning [27], incremental learning [6], and online learning [22, 33]. Despite this breadth, the challenge of TAS condensation [5] remains an emerging frontier. Unlike previous work, we replace the optimization-based inversion paradigm with deterministic latent mapping to condense action segments into sparse latent anchors and restore through latent trajectory interpolation.

Dataset Condensation. DC aims to synthesize a small set of informative samples that represent the knowledge of a large-scale dataset [29]. While initial methods focused on image classification [3, 4, 18, 28, 32] using gradient matching, trajectory matching [3] or distribution matching [32]. Extending DC to the video domain is challenging due to the high dimensionality and temporal redundancy of video features. Effective condensation must therefore preserve temporal coherence and transition structure and directly adapting image-based strategies is thus suboptimal. Furthermore, TAS represents a structured video setting that further requires fine-grained, frame-wise temporal modeling, making action recognition condensation techniques [30] undesirable. Existing TAS condensation methods [5] rely on iterative inversion of generative models like cVAEs. However, these are often computationally expensive and struggle to capture continuous temporal dynamics. In contrast, our approach leverages the deterministic flow of DDIM to achieve efficient, trajectory-based condensation without the need for costly per-sequence optimization.

3 Preliminaries

3.1 Temporal Action Segmentation

Temporal Action Segmentation (TAS) [7] aims to assign a semantic category to every frame in an untrimmed video. Formally, a video is represented as a sequence of L frame-level features $V = \{x_i\}_{i=1}^{L}$, where $x_i \in \mathbb{R}^D$ represents the feature vector of the i-th frame from a pretrained visual backbone, *e.g.*, I3D [2]. The goal is to map this sequence to a corresponding set of semantic labels $Y = \{y_i\}_{i=1}^{L}$, where each $y_i \in [1, \ldots, A]$ denotes the action class.

To learn a TAS model $\mathcal{M}$ [10, 31], a composite objective function is typically employed to balance the frame-wise accuracy with temporal coherence. The primary component is the classification loss, formulated as a frame-wise cross-entropy:

$$\mathcal{L}_{\text{cls}}(x, y) = \frac{1}{L} \sum_{i=1}^{L} -\log(\hat{y}_{i,a}), \tag{1}$$

where $\hat{y}_{i,a}$ represents the predicted probability for the ground-truth class a for frame i. In addition, to address the over-segmentation issue where the model predicts frequent and erratic action transition, a smoothing loss is integrated:

$$\mathcal{L}_{\text{sm}}(x) = \frac{1}{LA} \sum_{i,a} \tilde{\Delta}_{i,a}^2, \;\; \tilde{\Delta}_{i,a} = \begin{cases} \Delta_{i,a} : \Delta_{i,a} \leq \tau \\ \tau \quad\;\; : \text{otherwise} \end{cases}, \Delta_{i,a} = \left|\log(\hat{y}_{i,a}) - \log(\hat{y}_{i-1,a})\right|, \tag{2}$$

with the truncation parameter τ set to 4, following [10]. By minimizing the total loss with a trade–off parameter λ:

$$\mathcal{L}_{\text{tas}} = \mathcal{L}_{\text{cls}}(x, y) + \lambda \mathcal{L}_{\text{sm}}(x), \tag{3}$$

the model is encouraged to produce semantically accurate yet temporally stable segmentation. Complementing the frame-wise view, the segmentation can also be represented homogeneously as a sequence of N action segments, *i.e.*,:

$$S = \{s_1, \ldots, s_N\}, \quad \text{where} \quad s_n = (a_n, t_n, \ell_n) \quad \text{and} \quad t_{n+1} = t_n + \ell_n, \quad (4)$$

where each segment s_n is characterized by its action category a_n, its starting timestamp t_n, and its temporal duration ℓ_n.

3.2 TAS Condensation

The main objective of TAS condensation is to construct a compressed representation of a TAS dataset while preserving its task-relevant and temporal properties. The condensed dataset should retain sufficient semantic and sequential information such that a TAS model trained on it achieves performance comparable to that obtained when trained on the full, uncompressed dataset.

Formally, given an original TAS dataset $\mathcal{D} = \{(V_i, Y_i)\}_{i=1}^{N_v}$, the goal of TAS condensation is to construct a compressed dataset $\mathcal{D}^* = \{(V_i^*, L_i)\}_{i=1}^{N_v}$ that enables the generation of a reconstructed proxy dataset $\hat{\mathcal{D}} = \{(\hat{V}_i, L_i)\}_{i=1}^{N_v}$ such that a model trained on $\hat{\mathcal{D}}$ achieves performance close to that obtained when trained on the original dataset $\mathcal{D}$. Each compressed representation is defined as $V_i^* = \{v_{i,1}^*, \ldots, v_{i,T_i^*}^*\}$, with $v_{i,j}^* \in \mathbb{R}^D$ and $T_i^* \ll T_i$. We also follow [5] and define the compression ratio as $\rho = |\hat{\mathcal{D}}|/|\mathcal{D}|$. Given that frame features are pre-extracted, temporal redundancy represents the most prominent overhead in TAS; we thus focus on compression along the temporal axis in this paper.

3.3 Diffusion and Deterministic DDIM Flow

Diffusion models [13, 25] are generative models that learn data distributions by reversing a gradual stochastic corruption process. Given a data sample x_0, the forward diffusion process progressively adds Gaussian noise to obtain a noisy latent representation x_t according to:

$$x_t = \sqrt{\alpha_t} x_0 + \sqrt{1 - \alpha_t}\epsilon, \quad (5)$$

where $\epsilon \sim \mathcal{N}(0, I)$ and α_t is a predefined noise schedule controlling the signal-to-noise ratio across timestamps t.

In particular, Denoising Diffusion Implicit Models (DDIM) [25] introduce an efficient sampling formulation by introducing a non-Markovian inference process that supports deterministic trajectories. Let $x_T \sim \mathcal{N}(0, I)$ denote Gaussian noise, the inversion and sampling trajectories of DDIM by the forward and reverse flow mappings, respectively:

$$\Phi_{0\to T} : x_0 \to \{x_t\}_{t=1}^{T}, \quad \text{and} \quad \Phi_{T\to 0} : x_T \to \{x_t\}_{t=T-1}^{0}. \quad (6)$$

The trajectory transition dynamics follow the update rule:

$$x_{t-1} = \sqrt{\alpha_{t-1}}\hat{x}_0(x_t) + \sqrt{1 - \alpha_{t-1}}\epsilon_\theta(x_t, t), \quad (7)$$

where ϵ_θ is the neural network prediction of the noise component, and $\hat{x_0}(x_t)$ denotes the model's prediction of the clean signal, given by:

$$\hat{x_0}(x_t) = \frac{x_t - \sqrt{1-\alpha_t}\epsilon_\theta(x_t, t)}{\sqrt{\alpha_t}}. \tag{8}$$

In practice, the model parameters are trained using a denoising objective that encourages accurate prediction of the noise component:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{x_0,\epsilon,t}\left[|\epsilon - \epsilon_\theta(x_t, t)|^2\right], \tag{9}$$

where t is uniformly sampled from the diffusion timesteps during training.

Since DDIM admits a non-Markovian formulation, the inversion and sampling trajectories are deterministic once the model parameters and noise schedule are fixed according to Eqs. (6) and (7). This deterministic property is particularly desirable for dataset condensation, as it enables stable trajectory reconstruction and controlled generation of condensed representations for TAS datasets.

4 Method

4.1 Action Modeling with Diffusion

For action modeling, we adopt a diffusion-based generative modeling strategy as opposed to the cVAE used in [5]. Diffusion models are chosen due to their stronger distribution modeling capacity and more flexible latent learning without requiring explicit prior regularization.

Specifically, we train a diffusion model Φ to learn the distribution of frame-level features conditioned on the action label and the frame's relative position within its action sequence, similar to [5]. For a frame x_i belonging to action a of length l with normalized sequence index $c_i = (i-1)/(\ell-1)$, where $c_i \in [0, 1]$, the noise prediction network is therefore parametrized as $\epsilon_\theta(x_{i,t}, a, c_i, t)$ at every step t. The action model is trained using the same denoising reconstruction objective as Eq. (9) with additional conditioning variables a and c:

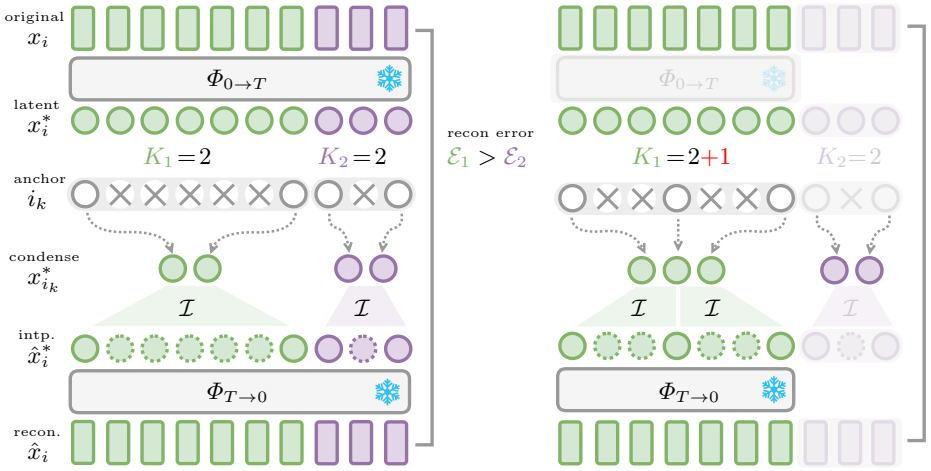
$$\mathcal{L}_{\text{act}} = \mathbb{E}_{x_i,a,c_i,\epsilon,t}\left[|\epsilon - \epsilon_\theta(x_{i,t}, a, c_i, t)|^2\right], \tag{10}$$

where $x_{i,t}$ denotes the latent representation along the DDIM forward flow trajectory $\Phi_{0\to T}(x_i)$ for x_i.

Latent Encoding and Generation. After training, latent representations for any frame can be obtained by directly applying the DDIM forward flow trajectory. Formally, the latent representation x_i^* is obtained by mapping the frame feature x_i through the trained diffusion model:

$$x_{i,T}^* = \Phi_{0\to T}(x_i, a, c_i), \tag{11}$$

with $\Phi_{0\to T}(\cdot)$ being the forward process.



(a) Anchor Initialization (b) Adaptive Anchoring

Fig. 2: Overview of adaptive latent trajectory anchoring. (a) Anchor initialization: Video frames x_i are mapped to latent space x_i^* with the DDIM forward process $\Phi_{0\to T}$. Initial anchors $x_{i_k}^*$ are sampled to represent each segment, which are then interpolated by $\mathcal{I}$ to reconstruct the original frames with $\Phi_{T\to 0}$. (b) Adaptive Anchoring: The framework dynamically reallocates anchors based on reconstruction error $\mathcal{E}$. Segments with higher error (*e.g.*, $\mathcal{E}_1 > \mathcal{E}_2$) are assigned additional anchors (+1) to capture complex temporal variations, while simpler segments remain intact. This process is then repeated until the anchor budget is exhausted. We omit the conditional variables (a, c) from this figure for simplicity.

Importantly, since DDIM inference adopts a deterministic formulation, obtaining latent representations with high correspondence to the original frame does not require solving an additional optimization problem as in [5]. Consequently, the reconstructed frame feature $\hat{x}_i$ from $x_{i,T}^*$ is expected to closely approximate x_i:

$$\hat{x_i} = \Phi_{T\to 0}(x_{i,T}^*, a, c_i), \quad \text{s.t.,} \quad \hat{x}_i \approx x_i, \tag{12}$$

where $\Phi_{T\to 0}(\cdot)$ is the reverse flow.

4.2 Adaptive Latent Trajectory Anchoring

The primary redundancy in TAS datasets is concentrated along the temporal axis, where consecutive frames within a local neighborhood exhibit high feature affinity. To exploit this, we seek to represent each action segment as a trajectory anchored by a set of sparse latent representations. Fig. 2 depicts the overall procedure of our adaptive latent trajectory anchoring framework.

Condensing Actions into Latent Anchors. For a segment $s = \{x_i\}_{i=1}^{\ell}$ of action class a and duration ℓ, we identify a compact set of K representative frames $\{x_{i_k}\}_{k=1}^{K}$, which serve as the anchors for the entire sequence. By storing

only these anchors, we reduce the storage footprint of action segments by a factor of ℓ/K compared to the original sequence since $K \ll \ell$. A straightforward way to initialize this condensation is to determine the temporal indices i_k for the k-th anchor via linear spacing:

$$i_k = \lceil 1 + \frac{k-1}{K-1}(\ell - 1) \rfloor, \quad k = 1, \ldots, K, \tag{13}$$

where $\lceil \cdot \rfloor$ denotes rounding to integer. These anchors are then mapped into the latent noise manifold using the DDIM forward process $\Phi_{0 \to T}$:

$$x^*_{i_k} = \Phi_{0 \to T}(x_{i_k}, a, c_{i_k}). \tag{14}$$

Notably, our diffusion action model enables direct latent mapping via a single forward pass, providing a significant efficiency gain over GNI [5] that relies on expensive iterative optimization for network inversion.

Latent Temporal Interpolation. In order to restore temporal resolution during reconstruction for TAS training, we model the action as a continuous latent trajectory to approximate the latent in between the anchors. Specifically, we define an interpolation operator, $\mathcal{I}$, which estimates the latent representations x^*_j for any frames situated between two anchors, *i.e.*, $i_k < j < i_{k+1}$. In contrast to the nearest-neighbor "inflation" used in [5], we have:

$$x^*_j = \mathcal{I}(x^*_{i_k}, x^*_{i_{k+1}}; \lambda_j) \quad \text{where} \quad \lambda_j = \frac{c_j - c_{i_k}}{c_{i_{k+1}} - c_{i_k}}. \tag{15}$$

Interpolating in latent space allows each frame to be assigned a unique latent representation at reconstruction time. This helps to preserve the fine-grained dynamics of action progression in cases where conditioning variables c alone may not suffice.

Adaptive Anchor Allocation. The anchor allocation strategy described above assigns an equal number of anchors to each action segment. While straightforward, enforcing a uniform distribution across segments may be suboptimal at the video level, as action segments can vary substantially in temporal duration and structural complexity. For a video containing N action segments, allocating K anchors per segment yields a total anchor budget of $B = N \times K$. To better utilize this fixed budget, we introduce an adaptive anchor allocation strategy that redistributes anchors across action sequences according to their reconstruction difficulty.

Let K_n denote the number of anchors assigned to segment s_n, we initialize each action segment with a minimal interpolation capacity $K_n = 2$, which ensures that latent interpolation between anchors is well-defined. To guide adaptive allocation, we measure the reconstruction difficulty of each segment using the mean reconstruction error:

$$\mathcal{E}_n = \mathbb{E}_{x_i \in s_n}[|x_i - \Phi_{T \to 0}(x^*_i, a, c_i)|^2] \tag{16}$$

where x^*_i denotes the interpolated latent representations obtained by Eq. (15) under the current anchor configuration. Reconstruction objectives are widely used

Algorithm 1 Adaptive Latent Trajectory Condensation

Input: A video $S = \{s_1, \ldots, s_N\}$, global anchor budget B
Output: Condensed anchors $\{K_n\}_{n=1}^{N}$,
for each segment $s_n = \{x_i\}_{i=1}^{\ell_n}$ **do** ▷ Anchor Initialization (Fig. 2(a))
  Initialize minimal anchors $K_n = 2$
  Sample anchor indices $\{i_k\}_{k=1}^{K_n}$ ▷ Eq. (13)
  Encode anchors $\{x^*_{i_k}\}_{k=1}^{K_n}$ ▷ Eq. (14)
  Interpolate latent trajectory x^*_j ▷ Eq. (15)
  Reconstruct frames $\hat{x}_i$ ▷ Eq. (12)
  Compute reconstruction error $\mathcal{E}_n$ ▷ Eq. (16)
end for
Build max-priority queue $\mathcal{Q}$ over $\{\mathcal{E}_n\}_{n=1}^{N}$
while $\sum_{n=1}^{N} K_n < B$ **do** ▷ Adaptive Anchoring (Fig. 2(b))
  Select worst segment n^* from the queue ▷ Eq. (17)
  Update anchor allocation K_{n^*} ▷ Eq. (18)
  Sample anchors for s_{n^*} with new K_{n^*} ▷ Eq. (13)
  Interpolate, reconstruct and compute error $\mathcal{E}_{n^*}$ ▷ Eqs. (12) and (14) to (16)
  Insert updated $(n^*, \mathcal{E}_{n^*})$ into $\mathcal{Q}$
end while

in representation learning as a proxy to indicate that the underlying structure of the data is well captured [12]. Here, we interpret the reconstruction residual $\mathcal{E}_n$ as a measure of how well the current set of anchors capture the temporal dynamics of the sequence.

At each iteration, an additional anchor is assigned to the segment with the largest reconstruction error, *i.e.*,:

$$n^* = \arg\max_n \mathcal{E}_n, \tag{17}$$

followed by updating

$$K_{n^*} \leftarrow K_{n^*} + 1. \tag{18}$$

After each update, anchor positions for segment s_{n^*} are re-sampled via Eq. (13) under the updated K_{n^*}, and the corresponding latent representations are recomputed. The interpolation operator is then re-applied to obtain updated latent approximations for that segment before the next allocation step.

This iterative procedure continues until the global anchor budget B is exhausted. By allocating anchors according to segment-wise reconstruction difficulty, the proposed strategy adaptively matches representation capacity to segment-wise reconstruction difficulty, assigning more anchors to segments that require finer temporal modeling. The overall adaptive anchor allocation is summarized in Algorithm 1.

4.3 Computational Complexity

Our framework achieves significantly lower latency by leveraging the deterministic flow of DDIM. For a video of N segments, our standard anchoring strategy of inversion and reconstruction process with T diffusion timesteps requires:

$\mathcal{O}(NT)$. For the adaptive variant, anchoring is repeated until the anchor budget $B = N \times \bar{K}$ is exhausted, yielding $\mathcal{O}(N\bar{K}T)$, where $\bar{K}$ is the average number of anchors per segment. While with GNI [5], for the same video, the latent codes are optimized via S optimization steps, yielding a total complexity of $\mathcal{O}(NS)$.

In practice, T is typically set to small values (*e.g.*, 10 or 50), whereas GNI [5] requires thousands of iterations ($S = 10,000$) of optimization. This ensures that even with the iterative nature of adaptive anchor allocation, the cumulative cost $N\bar{K}T$ remains orders of magnitude smaller than the $\mathcal{O}(NS)$ cost of GNI while achieving greater temporal expressiveness through latent anchoring.

4.4 Decoding for TAS Training

The condensed latent representation obtained from the adaptive trajectory condensation process is used to construct a compact surrogate dataset for TAS training. After condensation, each action segment is represented by a sparse set of anchor latent codes $\{x^*_{i_k}\}_{k=1}^{K}$. Formally, the reconstructed frame-level representation of a video sequence is given by:

$$\hat{x}_i = \Phi_{T \to 0}(\mathcal{I}(x^*_{i_k}, x^*_{i_{k+1}}; \lambda_i), a, c_i). \tag{19}$$

The TAS model is then trained on the reconstructed dataset with the standard loss functions introduced in Eq. (3) by substituting x with $\hat{x}$:

$$\mathcal{L}_{\text{tas}} = \mathcal{L}_{\text{cls}}(\hat{x}, y) + \lambda \mathcal{L}_{\text{sm}}(\hat{x}). \tag{20}$$

5 Experiment

5.1 Datasets and Evaluation Metrics

Datasets. We evaluate our method on three widely used TAS benchmarks that vary in scale and storage requirements: GTEA [11], 50Salads [26], and Breakfast [15]. **GTEA** [11] consists of 28 egocentric kitchen videos spanning 7 high-level activities and 11 action classes. The videos are relatively short and exhibit limited intra-class variation. **50Salads** [26] contains 50 long videos of salad preparation annotated with 19 action classes. It features extended temporal durations and more complex action transitions. **Breakfast** [15] is a large-scale benchmark comprising 1,712 videos covering 10 breakfast preparation activities and 48 fine-grained action classes. Each video contains 5 to 14 action segments on average, with substantial variability in duration and ordering.

For all datasets, we use pre-extracted I3D features [2] and follow the standard evaluation splits. While I3D compresses spatial information by mapping RGB frames into a feature space, the original temporal resolution of the video is preserved.

Evaluation Metrics. We follow the standard TAS evaluation protocol and report three metrics: frame-wise accuracy (Acc), segmental edit score (Edit), and F1 score at overlap thresholds of 10%, 25%, and 50%.

5.2 Implementation

Our diffusion model was implemented as an ϵ-predictor with a lightweight architecture design. The timestep is encoded using a single linear layer, then concatenated with the other inputs and passed through a two-layer linear module with a bottleneck compression ratio of 16. We used a learning rate of 0.001 and trained the model for 2K epochs. We set the average anchor number per segment to be $\bar{K} = 8$.

Baselines. Following [5], we implement the following baselines for comparison:
– **Mean:** A simple condensation baseline where frame features are averaged within each action segment. The averaged feature is then temporally expanded during reconstruction to match the original segment length.
– **Intp.:** This method stores the first and last frame of each action segment and reconstructs intermediate frames through linear interpolation between them.
– **Coreset:** This approach uses herding to select the frame feature closest to the mean feature of each segment. The selected features are then temporally upsampled to restore the original temporal resolution.
– **GNI [5]:** This method employs a time coherent VAE. Each action sequence is divided into temporal chunks, and a common latent representation is optimized for each chunk. During reconstruction, each latent is expanded to the corresponding chunk length. We optimize latent codes for 10K steps.
– **Original:** This is the standard setup where full original frame features are used, which we consider as the upper bound.

5.3 Effectiveness

Tab. 1 compares different strategies for TAS condensation across common benchmarks. Since GNI additionally applies an 8× compression along the feature dimension, we reproduce and report GNI under two settings: GNI with $\bar{K} = 8$, which matches our temporal compression factor $\bar{K}$, and GNI with $\bar{K} = 64$, which matches our overall compression ratio ρ. Our method is evaluated in two variants: the primary adaptive latent trajectory anchoring (Ours) and a variant using standard anchor allocation (Ours†). As we can see, both of our variants surpass the strongest baseline GNI [5] by significant margins across all metrics. For instance, on Breakfast with the ASFormer backbone, our primary variant achieves 68.0% accuracy, representing a 5.2% absolute improvement over GNI (62.8%) and an 18.1% improvement over the Coreset baseline (49.9%). The two variants also exhibit complementary strengths. The standard version (Ours†) can achieve higher Acc in some cases, such as 75.1% accuracy on 50Salads. In contrast, the adaptive variant (Ours) shows better long-term temporal consistency, reflected by higher values on segmental metrics. This suggests that the adaptive strategy is effective in modeling the action evolution of long and complex actions where more anchors are allocated.

Last but not least, our method remarkably narrows the performance gap relative to the original full dataset. For example, on 50Salads with ASFormer, our method reaches 75.1% accuracy, coming within 2% of the performance achieved using the full dataset, while using only 1.4% of storage.

	$\bar{K}$	GTEA (256MB) Acc	Edit	F1@10/25/50	50Salads (4.7GB) Acc	Edit	F1@10/25/50	Breakfast (27.4GB) Acc	Edit	F1@10/25/50
MSTCN [10]										
Mean	1	69.4	67.5	72.6/68.6/52.8	70.0	46.7	54.2/49.5/40.2	48.0	33.2	29.6/25.5/17.6
Intp.	2	61.1	69.6	74.6/64.4/46.8	49.9	53.9	54.4/49.6/34.6	33.6	42.9	38.7/31.8/19.3
Coreset	1	60.3	63.4	65.4/59.2/44.2	69.6	62.5	65.7/63.1/56.6	50.2	41.3	36.7/31.6/22.3
GNI [5]	8	66.7	71.9	73.9/68.9/45.9	**73.5**	43.8	49.0/46.6/38.6	37.9	44.5	38.1/33.5/24.7
GNI [5]	64	66.6	66.9	74.2/69.9/48.8	72.8	44.2	49.3/45.0/40.2	47.8	54.0	46.9/41.9/30.9
Ours†	8	71.2	**80.6**	**86.5**/**80.5**/58.6	72.7	**65.1**	**71.9**/**68.6**/**57.9**	54.8	**67.2**	**63.8**/57.6/45.1
Ours	8	**72.1**	79.1	85.9/78.0/**61.4**	70.1	62.3	68.8/65.3/53.5	**63.4**	65.6	**63.8**/**58.4**/**46.1**
Original	ℓ	72.7	73.8	79.9/73.7/59.4	74.6	60.0	66.9/64.7/56.2	67.9	67.9	67.7/61.8/49.5
ASFormer [31]										
Mean	1	70.7	73.3	78.2/76.9/67.3	62.2	40.7	47.9/42.2/33.4	51.8	46.5	44.9/39.8/28.1
Intp.	2	64.1	72.7	76.5/71.9/54.9	54.8	48.2	55.4/49.5/31.9	43.1	48.3	47.3/40.1/25.6
Coreset	1	67.2	69.4	73.6/72.3/55.3	66.8	45.3	53.6/48.4/38.2	49.9	51.0	47.1/41.6/30.2
GNI [5]	8	71.6	76.4	80.8/79.4/63.4	70.5	51.4	61.6/57.6/47.3	62.8	66.3	65.8/59.4/47.2
GNI [5]	64	70.8	73.5	81.2/77.8/62.1	72.1	49.9	59.7/56.1/46.2	61.1	64.6	63.5/58.4/45.9
Ours†	8	**73.7**	**81.5**	84.3/81.5/71.8	**75.1**	**64.9**	**71.7**/**68.8**/**57.6**	65.3	**71.5**	70.1/65.2/48.4
Ours	8	73.4	80.5	**85.2**/**83.1**/**73.9**	71.2	60.7	69.2/64.9/54.4	**68.0**	71.3	**71.4**/**66.3**/**52.9**
Original	ℓ	74.3	77.4	82.9/80.1/73.1	77.1	67.6	77.1/73.1/61.7	70.7	72.0	73.6/69.0/56.1

Table 1: Comparison of dataset condensation performances on three TAS benchmarks using MSTCN and ASFormer backbones. † indicates fixed anchor allocation. Our trajectory-based condensation variations consistently outperform existing baselines and shows competitive performance when training with original data.

5.4 Ablation Study and Analysis

Anchor budget. Table 2 reports the effect of the mean anchor budget $\bar{K}$ on condensation performance. As we can see, increasing $\bar{K}$ tends to improve the segmentation performance across all metrics, reflecting the benefit of denser temporal coverage. However, returns diminish beyond $\bar{K} = 8$: increasing to $\bar{K} = 10$ yields only a marginal gain in accuracy (72.1% → 73.3%) at the cost of a 21% increase in storage ($\rho : 0.24 \rightarrow 0.29$). We therefore set $\bar{K} = 8$ as our default.

On the other hand, our adaptive anchoring strategy (Ours) generally outperforms the uniform baseline (Ours†) across various budget constraints ($\bar{K}$). This performance gain is particularly evident at lower anchor budgets, where the adaptive mechanism adaptively redistributes resources to segments that are most challenging to reconstruct.

Expressiveness of action models. The results reported in Tab. 3 provide an empirical investigation into the representational capacity of different action models. A critical observation for the GNI baseline is that increasing the number of sub-segments ($\bar{K}$) from 8 to 64, the model exhibits a clear performance plateau, while Edit score degrades from 71.9% to 66.9%. A similar trend is also reported by [5] suggesting that the expressiveness of the underlying VAE framework is

		Ours†			Ours		
$\bar{K}$	ρ	Acc	Edit	F1@10/25/50	Acc	Edit	F1@10/25/50
2	0.06	64.4	79.5	78.2/68.4/51.1	-	-	- / - / -
4	0.12	68.6	**81.9**	84.8/76.5/57.6	70.5	78.8	81.4/72.8/59.3
6	0.18	71.2	79.4	85.9/76.3/57.8	72.5	**80.0**	84.8/76.8/58.7
8	0.24	71.2	80.6	**86.5**/**80.5**/58.6	72.1	79.1	**85.9**/**78.0**/61.4
10	0.29	**73.3**	78.4	84.7/78.8/**63.5**	**73.8**	77.2	83.3/75.5/**61.5**
12	0.32	71.5	81.2	82.8/75.5/61.5	71.8	75.5	82.4/76.7/60.2

Table 2: Performance comparison on GTEA dataset under MSTCN backbone of our methods with different anchor budget.

	$\bar{K}$	ρ	Acc	Edit	F1@10/25/50
GNI [5]	8	0.03	66.7	71.9	73.9/68.9/45.9
	64	0.24	66.6	66.9	74.2/69.9/48.8
Ours†	8	0.24	71.2	80.6	86.5/80.5/58.6
Ours	8	0.24	72.1	79.1	85.9/78.0/61.4

Table 3: The expressiveness comparison of action models on GTEA. Increasing the number of latent codes for GNI leads to performance plateau, likely due to an inherent information bottleneck. While our approach achieves significant boost with same compression ratio.

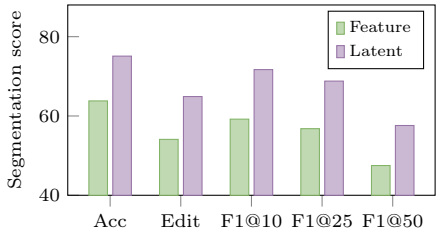


Fig. 3: Comparison between interpolation in the feature space and interpolation in the latent space on 50salads. Latent interpolation shows consistent performance gain over direct feature interpolation.

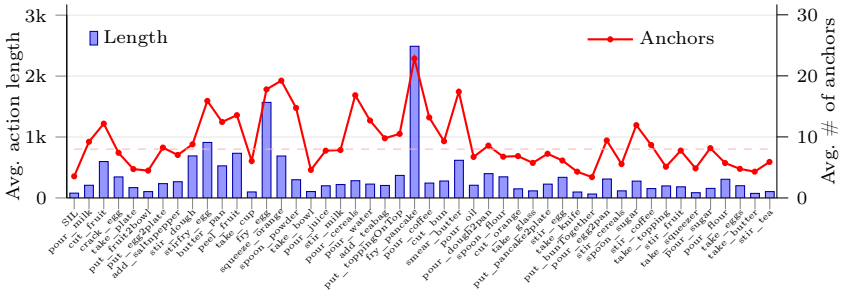


Fig. 4: Analysis of dynamic anchor allocation on Breakfast. Actions are ordered by decreasing frequency. Longer actions generally receive more anchors.

inherently bounded. In contrast, our proposed method consistently outperforms them significantly with the same compression ratio ρ.

Interpolation space. Fig. 3 compares interpolation in feature space and latent space on 50Salads dataset. Overall, latent space interpolation consistently yields better segmentation performance across all metrics, suggesting that tra-

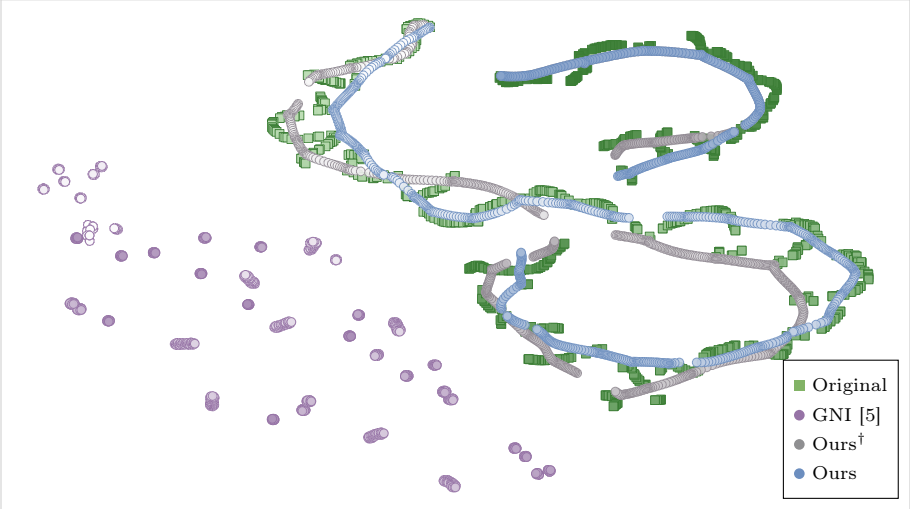


Fig. 5: Comparative t-SNE visualizations of temporal feature trajectories for action `take_bowl` in video `P03_cam01_P03_cereals` from the Breakfast dataset. We evaluate the alignment between the Original features (green squares) and three models: GNI [5] (purple) , Ours† with fixed anchors (gray), and Ours with adaptive anchors (blue). Variation in color intensity indicates temporal progression. Our adaptive approach produces feature trajectories that most closely approximate the original. (Best viewed when zoomed in.)

jectory modeling in the latent representation better preserves temporal semantic structure than direct interpolation in raw feature space. This supports our design choice of performing condensation through anchor interpolation in a learned latent space.

Dynamic Anchor Distribution Analysis. We next study how these dynamic anchors are distributed across action segments. Fig. 4 shows the average number of anchors assigned to each action category under the dynamic anchor allocation setting. While actions are ordered by frequency, no clear correlation between frequency and anchor count is observed. Instead, categories with longer average durations tend to receive more anchors, suggesting that the allocation adapts primarily to temporal scale. This adaptive behavior enables the model to provide finer duration coverage for long actions while avoiding unnecessary anchors for short actions.

Visualization. To assess the fidelity of the condensed representations, we visualize the temporal feature trajectories using t-SNE [20] for a representative sequence from the Breakfast dataset in Fig. 5. As we can see, the optimization-based GNI [5] produces fragmented clusters that fail to capture the sequential manifold of the original features. While our method with a fixed budget recovers the global flow, it tends to over-simplify intricate temporal variations in complex feature regions. In contrast, our full adaptive framework achieves superior

alignment by dynamically concentrating anchors where the trajectory is most volatile. Our diffusion-based approach faithfully preserves both local dynamics and global progression, as evidenced by the reconstructed trajectory's high proximity to the ground truth.

6 Conclusion

In this paper, we present a diffusion-based TAS condensation framework that replaces expensive iterative optimization with deterministic latent trajectory anchoring. By leveraging the bijective properties of DDIM, we represent action segments as trajectories in the latent space, preserving fine-grained temporal dynamics that traditional cVAE-based methods tend to over-smooth. In addition, our adaptive budgeting strategy ensures high-fidelity reconstructions by concentrating representation capacity on complex segments. Results across common TAS benchmarks demonstrate that our approach achieves competitive performance to full real data training while using as little as 1.4% of the original storage. This shift toward optimization-free, trajectory-based compression offers a scalable and efficient path for managing large-scale video data.

Acknowledgment. This research / project is supported by the Ministry of Education, Singapore, under the Academic Research Fund Tier 1 (FY2025).

References

1. Bueno-Benito, E., Dimiccoli, M.: Clot: Closed loop optimal transport for unsupervised action segmentation. In: ICCV (2025)
2. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: CVPR (2017)
3. Cazenavette, G., Wang, T., Torralba, A., Efros, A.A., Zhu, J.Y.: Dataset distillation by matching training trajectories. In: CVPR (2022)
4. Cui, J., Wang, R., Si, S., Hsieh, C.J.: Scaling up dataset distillation to imagenet-1k with constant memory. In: ICML (2023)
5. Ding, G., Chen, R., Yao, A.: Condensing action segmentation datasets via generative network inversion. In: CVPR (2025)
6. Ding, G., Golong, H., Yao, A.: Coherent temporal synthesis for incremental action segmentation. In: CVPR (2024)
7. Ding, G., Sener, F., Yao, A.: Temporal action segmentation: An analysis of modern techniques. IEEE Transactions on Pattern Analysis and Machine Intelligence **46**(2), 1011–1030 (2023)
8. Ding, G., Yao, A.: Leveraging action affinity and continuity for semi-supervised temporal action segmentation. In: ECCV (2022)
9. Ding, G., Yao, A.: Temporal action segmentation with high-level complex activity labels. TMM (2022)
10. Farha, Y.A., Gall, J.: Ms-tcn: Multi-stage temporal convolutional network for action segmentation. In: CVPR (2019)
11. Fathi, A., Ren, X., Rehg, J.M.: Learning to recognize objects in egocentric activities. In: CVPR (2011)

12. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 16000–16009 (2022)
13. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: NeurIPS (2020)
14. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: ICLR (2014)
15. Kuehne, H., Arslan, A., Serre, T.: The language of actions: Recovering the syntax and semantics of goal-directed human activities. In: CVPR (2014)
16. Kumar, S., Haresh, S., Ahmed, A., Konin, A., Zia, M.Z., Tran, Q.H.: Unsupervised action segmentation by joint representation learning and online clustering. In: CVPR (2022)
17. Liu, D., Li, Q., Dinh, A.D., Jiang, T., Shah, M., Xu, C.: Diffusion action segmentation. In: ICCV (2023)
18. Liu, S., Wang, K., Yang, X., Ye, J., Wang, X.: Dataset distillation via factorization. NeurIPS **35** (2022)
19. Lu, Z., Elhamifar, E.: Fact: Frame-action cross-attention temporal modeling for efficient action segmentation. In: CVPR (2024)
20. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(11) (2008)
21. Richard, A., Kuehne, H., Gall, J.: Action sets: Weakly supervised action segmentation without ordering constraints. In: CVPR (2018)
22. Shen, Y., Elhamifar, E.: Progress-aware online action segmentation for egocentric procedural task videos. In: CVPR (2024)
23. Singhania, D., Rahaman, R., Yao, A.: Iterative contrast-classify for semi-supervised temporal action segmentation. In: AAAI (2022)
24. Singhania, D., Rahaman, R., Yao, A.: C2f-tcn: A framework for semi-and fully-supervised temporal action segmentation. IEEE TPAMI (2023)
25. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. In: ICLR (2021)
26. Stein, S., McKenna, S.J.: Combining embedded accelerometers with computer vision for recognizing food preparation activities. In: UbiComp (2013)
27. Su, Y., Elhamifar, E.: Two-stage active learning for efficient temporal action segmentation. In: ECCV (2024)
28. Wang, K., Zhao, B., Peng, X., Zhu, Z., Yang, S., Wang, S., Huang, G., Bilen, H., Wang, X., You, Y.: Cafe: Learning to condense dataset by aligning features. In: CVPR (2022)
29. Wang, T., Zhu, J.Y., Torralba, A., Efros, A.A.: Dataset distillation. arXiv preprint arXiv:1811.10959 (2018)
30. Wang, Z., Xu, Y., Lu, C., Li, Y.L.: Dancing with still images: Video distillation via static-dynamic disentanglement. In: CVPR (2024)
31. Yi, F., Wen, H., Jiang, T.: Asformer: Transformer for action segmentation. In: BMVC (2021)
32. Zhao, B., Bilen, H.: Dataset condensation with distribution matching. In: WACV (2023)
33. Zhong, Q., Ding, G., Yao, A.: Onlinetas: An online baseline for temporal action segmentation. NeurIPS (2024)