Gated Probabilistic Diffusion for Temporal Action Segmentation

Yun Li*, Hanmin Li † and Kin-Man Lam*
* The Hong Kong Polytechnic University, Hong Kong
† Sun Yat-sen University, China

***Abstract*—Temporal action segmentation is a fundamental task in video understanding, involving the identification and classification of human actions in long, untrimmed videos. Existing methods often suffer from over-segmentation errors and struggle to model complex temporal dependencies. Inspired by denoising diffusion models, we propose Gated Probabilistic Diffusion Action Segmentation (GPDAS), a novel framework that formulates action segmentation as a conditional sequence generation task. GPDAS iteratively refines frame-wise action labels through a denoising process conditioned on video features, implicitly modeling action priors and domain-specific behavioral knowledge. Our approach includes (1) a gated probabilistic decoder with adaptive temporal convolutions to enhance boundary accuracy and action continuity, (2) dual boundary-aware and action-dependent loss functions to capture chronological dependencies and improve temporal localization, and (3) masked conditioning strategies to improve robustness. Evaluated on the GTEA, 50Salads, and Breakfast benchmarks, GPDAS achieves state-of-the-art performance, outperforming existing methods in edit score and segmental F1 scores while effectively mitigating over-segmentation. The gated decoder demonstrates strong performance in modeling long-range, complex action dynamics.**

I. Introduction

Temporal action segmentation is a key task in video understanding, involving the identification and classification of human actions in long, untrimmed videos. It plays a critical role in applications such as video surveillance, human-computer interaction, and autonomous systems. Existing methods often employ an iterative refinement strategy, utilizing multistage models to progressively improve segmentation accuracy [1]–[3]. These methods aim to capture long-range temporal dependencies and mitigate over-segmentation errors, which are common in single-pass approaches.

Inspired by the success of denoising diffusion models in generative tasks [4]–[7], we formulate action segmentation as a conditional sequence generation problem. Our proposed method, Gated Probabilistic Diffusion Action Segmentation (GPDAS), generates frame-wise action label sequences through an iterative denoising process conditioned on input video features. This formulation not only learns the mapping from video frames to actions, but also implicitly models the prior distribution of action sequences, enabling the incorporation of domain-specific knowledge about human behaviors.

To further enhance the model's ability to capture the nuances of human actions, we introduce two types of loss functions: (1) boundary loss, which accounts for the temporal locations where actions are likely to occur; and (2) action loss, which captures the chronological dependencies between actions. To better control uncertainty and temporal relations, we introduce a gated decoder structure, which further improves the model performance. Through extensive experiments on benchmark datasets, including GTEA, 50Salads, and Breakfast, we demonstrate that GPDAS achieves competitive or superior performance compared to existing state-of-the-art methods. The contributions of this paper are summarized as follows:

- We propose GPDAS, a gated probabilistic diffusion framework for action segmentation that jointly balances action continuity and boundary accuracy
- The proposed gated decoder structure proves effective across multiple benchmark datasets, delivering strong performance in modeling complex temporal dynamics.

II. Related Works

A. Action Segmentation

Segmentation tasks have a vast range of applications [8]–[12]. Action segmentation aims to divide a video into temporal segments, each associated with a distinct action. Recurrent Neural Networks (RNNs) were among the first approaches to model temporal dependencies, leveraging LSTM/GRU units to capture short-range context. However, they struggle with long videos due to vanishing gradients and limited receptive fields. To address these limitations, Temporal Convolutional Networks (TCNs) emerged as a dominant solution. Early TCNs used dilated convolutions to expand receptive fields [13], while multi-stage architectures (e.g., MS-TCN [14]) iteratively refined predictions across stages to reduce over-segmentation. Subsequent variants, such as BCN [1] introduced boundary-aware pooling, and G2L [15] optimized global-to-local feature interactions.

The Diffact [4] introduced a novel generative approach to the action segmentation task. Unlike traditional discriminative methods, it employs denoising diffusion models to generate action label sequences conditioned on video features. This method explicitly models action priors, such as position, boundary, and relational dependencies, through a conditional masking strategy, achieving state-of-the-art performance. However, it still suffers from over-segmentation in certain scenarios. By introducing a gated decoder into the diffusion model, we successfully improve segmentation performance and mitigate over-segmentation.

979-8-3315-7206-8/25/$31.00

B. Diffusion models

Diffusion models [5] have emerged as a powerful paradigm for generative modeling, achieving significant success across multiple domains due to their stable training dynamics and independence from adversarial mechanisms. These models, theoretically linked to score-based generative models [16], operate by progressively adding noise to data in a forward process and learning to reverse the corruption to reconstruct the original distribution. In the realm of image generation [17], diffusion models have established new benchmarks for quality and realism [18]. In particular, Rombach et al. [19] developed latent diffusion models, enabling high-resolution image synthesis and demonstrating the scalability and expressive power of this approach. Beyond image generation, diffusion models have been successfully adapted to diverse applications, including natural language generation [20], text-to-image synthesis [21], image restoration [22] [23], super-resolution [24] and audio generation [25].

III. Methodology

In this section, we first present the preliminary concepts of diffusion models, including the diffusion forward process and the reverse (backward) diffusion process. We then provide a detailed introduction to our proposed model, Gated Probabilistic Diffusion Action Segmentation (GPDAS). GPDAS comprises three main components: a condition generation encoder, a feature generation based on the concatenated cross-layer feature with masking, and a gated decoder for probabilistic diffusion.

A. Preliminary

Diffusion models are generative frameworks that transform complex data distributions into simple noise through a forward process and then reverse this process to generate new samples. They excel at capturing uncertainty and generating diverse outputs, making them ideal for tasks involving ambiguity. Diffusion models [5] [26] are designed to estimate the noise distribution during training. In this formulation, the objective is to estimate the ground truth data distribution $g(x_0)$ under the model-generated data distribution $p_\theta(x_0)$. To achieve this, the diffusion process consists of two key components: the forward process and the backward process.

Forward Process: The forward process is a Markov chain that incrementally adds Gaussian noise to an original data sample over a series of steps. Starting with a clean data sample distribution $x_0 \sim g(x_0)$, noise is introduced at each step t according to a predefined variance schedule β_t, where $t = 1, 2, ..., T$. This process can be formally defined as:

$$g(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I), \tag{1}$$

where $\mathcal{N}$ represents a Gaussian distribution, x_t is the noisy data at step t, and I is the identity matrix. The parameter β_t controls the noise level at each step t, with $\beta_t \in (0, 1)$. As step t increases, x_t becomes progressively noisy, eventually approximating a standard Gaussian distribution $\mathcal{N}(0, I)$ when $t = T$. This gradual corruption ensures that the original data structure is systematically dismantled, providing a smooth transition to pure noise.

In [5], the process was reformulated to allow direct computation from the initial timestep 0 to any timestep t. The formulation is expressed as follows:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \tag{2}$$

where $\bar{\alpha}_t = \prod_{t=1}^{T}(1-\beta_t)$, and $\epsilon \sim \mathcal{N}(0, I)$ is a random noise sample. This approach is computationally efficient and widely adopted in practice.

Reverse Process: The reverse process, or denoising process, aims to reconstruct the original data by iteratively removing noise from a sample drawn from a Gaussian distribution. Starting from pure noise x_T, the model learns a Markov chain that approximates the posterior distribution $g(x_{t-1}|x_t)$. This process is parameterized by a neural network denoted as p_θ, and can be formulated as follows:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2 I), \tag{3}$$

where $\mu_\theta(x_t, t)$ is the mean predicted by the network, and σ_t^2 is a variance term controlled by the variance schedule in the forward process (e.g., $\sigma_t^2 = \beta_t$). The network is usually trained to predict the noise component ϵ added at each step, rather than directly predicting x_{t-1}. The training loss is typically the mean squared error between predicted noise $\epsilon_\theta(x_t, t)$ and actual noise ϵ:

$$\mathcal{L} = E_{t,x_0,\epsilon}\left[||\epsilon - \epsilon_\theta(x_t, t)||^2\right]. \tag{4}$$

During inference, the reverse process starts with a Gaussian noise sample $\mathcal{N}(0, I)$ and iteratively applies the learned transitions to generate a predicted sample $\widetilde{x}_0 \sim p_\theta(x_0)$ that approximates the true data distribution $g(x_0)$. The denoised output at each step can be expressed as follows:

$$\begin{aligned}\mathbf{x}_{t-1} =& \sqrt{\bar{\alpha}_{t-1}}\widetilde{x}_0 + \\ & \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\widetilde{x}_0}{\sqrt{1-\bar{\alpha}_t}} + \sigma_t\boldsymbol{\epsilon},\end{aligned} \tag{5}$$

$$\widetilde{x}_0 = F_\theta\left(\mathbf{x}_t, t\right), \tag{6}$$

where $F_\theta(\cdot)$ represents the diffusion model process, and $\widetilde{x}_0$ is the predicted sample from the model $F_\theta\left(\mathbf{x}_t, t\right)$ at step t.

B. The proposed method

In our proposed method, action segmentation is formulated as a diffusion process, as illustrated in Fig.1. In this formulation, video frame features served as the condition for the diffusion process. Unlike conventional diffusion models designed for image generation, our method is specifically tailored for recognition tasks. Similar to standard diffusion models, the input to the diffusion process is Gaussian noise, while the output consists of segmentation labels conditioned on the input video features.

Given input video features $V \in \mathbb{R}^{L\times D}$, where L is the number of video frames and D is the feature dimension, our objective is to estimate the one-hot representation of the ground-truth action label sequence $A \in \{0, 1\}^{L,C}$, where C denotes the number of action classes.

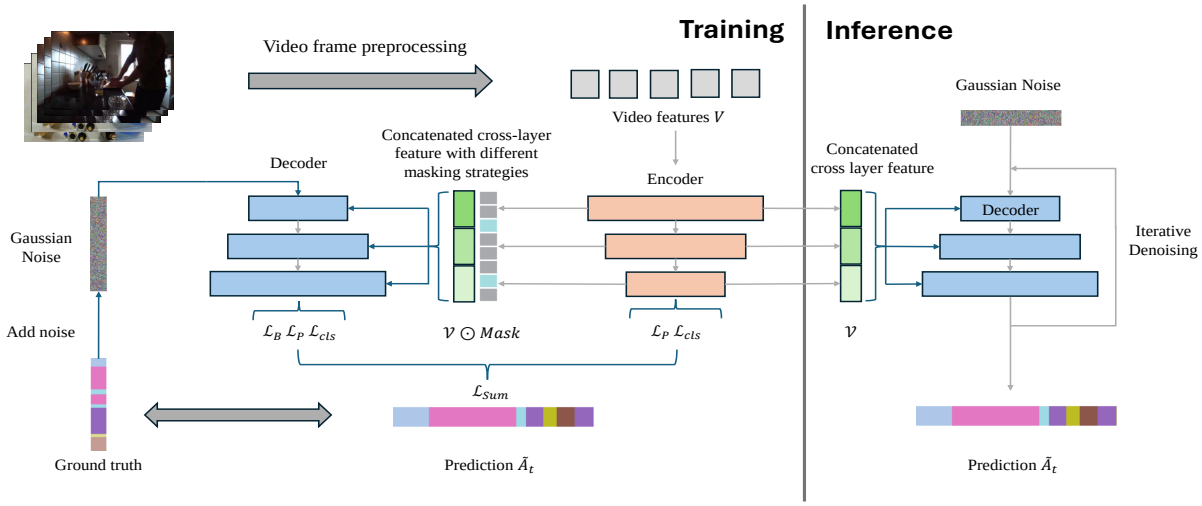

Fig. 1. Structure overview. The GPDAS framework comprises two distinct phases: training and inference. During the training phase, pre-extracted video features are input into an encoder network. This encoder generates a conditioning signal by processing and concatenating features from three distinct layers. These aggregated features are then masked using various masking strategies. Simultaneously, the ground truth action sequence labels undergo a diffusion process, progressively corrupted by the addition of Gaussian noise. A denoising model is trained to reverse this corruption, learning to reconstruct the original labels from the noisy input conditioned on the masked encoder output. In the inference phase, the same encoder and denoising model are utilized. However, the encoder processes video features to generate the unmasked conditioning signal. Starting from pure Gaussian noise, the denoising model iteratively generates the predicted action sequence, conditioned on the unmasked encoder output.

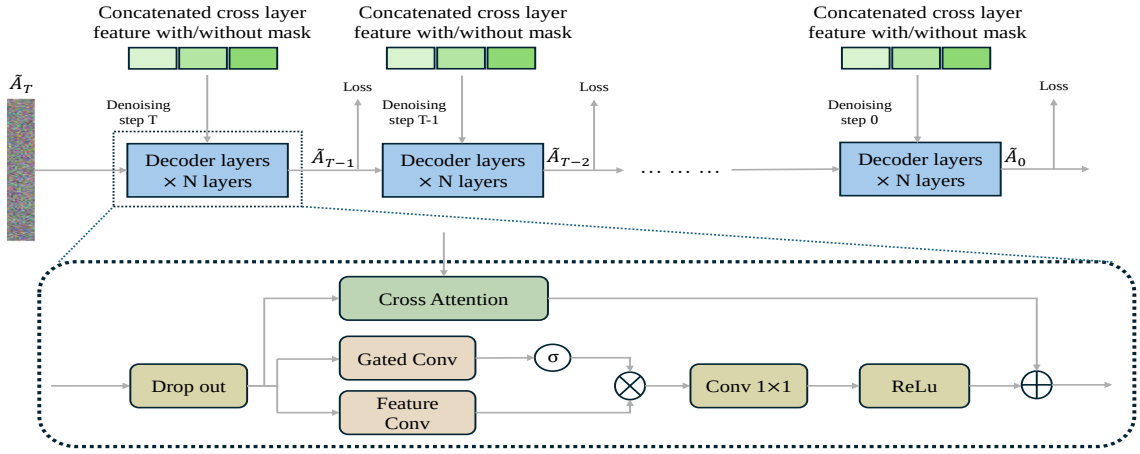

Fig. 2. Structure of the proposed gated decoder. The proposed decoder employs a gated structure comprising N layers. During the denoising process, the input traverses all N layers sequentially to complete a single denoising step. It is important to distinguish between the training and inference phases in terms of denoising steps. During training, a step number t is sampled uniformly from the range $[0, T]$ for each denoising operation. In contrast, inference requires the execution of the full sequence of T denoising steps.

As shown in Fig. 1, the ensemble structure comprises two main components: an encoder $E_\phi(\cdot)$, which generates condition features for the diffusion process, and a decoder $D_\omega(\cdot)$, which transforms Gaussian noise through the diffusion process to predict the action sequence $\tilde{A}$. The overall process includes two phases: a training phase, during which the encoder and decoder are optimized to segment corrupted inputs, and an inference stage, where Gaussian noise is denoised based on the conditions derived from the input video.

The proposed method, GPDAS, is built upon the diffusion process. Let $A_0, A_1, ..., A_T$ denote the sequence of ground-truth labels progressively corrupted into Gaussian noise. According to the basic diffusion theorem, the forward process can be formulated as follows:

$$A_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, \tag{7}$$

where ϵ is Gaussian white noise.

The encoder $E_\phi(V)$ projects the input video features into a higher dimensional space $\mathcal{V} \in \mathbb{R}^{L \times D'}$, where D' is the hidden dimension. To generate condition features, we use not only the output from the last layer of the encoder, but also splice outputs from different layers to form a cross-layer feature representation. Inspired by [27] [4], we apply a masking strategy to the extracted features to generate diffusion conditions. The spliced cross-layer feature is multiplied element-wise (Hadamard product) with a mask to produce masked encoding: $\mathcal{V} \odot Mask$. Following [26], we adopt three masking strategies for masked encoding: (1) Full Masking: Random video segments are masked to leverage the benefits of temporal context. Only location embeddings are used. (2) Transition Masking: Regions near action boundaries are masked with a small probability to enhance robustness against ambiguity.

(3) Sequence Masking: The entire action duration of a single action feature segment is masked to enable inference under the absence of temporal dependencies. In each training iteration, only one masking strategy is randomly selected, generating a mask $Mask \in \{0,1\}^{T\times D'}$. The masked features are then input into the decoder as conditions, promoting robustness prior learning.

Based on cross-layer features with a masking strategy $\mathcal{V}\odot Mask$, the decoder uses this as a condition for the diffusion process. It combines this condition with the diffusion time steps and the corrupted feature A_T to perform the denoising process: $D_\omega(A_T, t, \mathcal{V}\odot Mask)$. The denoised sequence is represented as: $\widetilde{A}_T, \widetilde{A}_{T-1}, ..., \widetilde{A}_0$, from the noisy feature $\widetilde{A}_T$ to the prediction $\widetilde{A}_0$ during inference. Following [5], we can directly sample the noise corresponding to a given timestep t without simulating the entire forward diffusion process. Consequently, for each training iteration, we randomly select a denoising step $t \sim Uniform\{1,2,...,T\}$. The backward process can be formulated as follows:

$$\begin{aligned}\widetilde{A}_{t-1} =& \sqrt{\bar{\alpha}_{t-1}} D_\omega(A_T, t, \mathcal{V}\odot Mask) + \\ & \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\,\frac{\widetilde{A}_t - \sqrt{\bar{\alpha}_t} D_\omega(A_T, t, \mathcal{V}\odot Mask)}{\sqrt{1-\bar{\alpha}_t}} + \sigma_t \boldsymbol{\epsilon}.\end{aligned} \tag{8}$$

where $\widetilde{A}_{t-1}$ and $\widetilde{A}_t$ are the denoised action sequences at the respective steps. At the end of the denoising step, we obtain the final predicted action sequence $\widetilde{A}_0$.

C. Gated Decoder for Probabilistic Diffusion

In this section, we elaborate on the novel structure of our proposed decoder: the gated probabilistic diffusion network. This generator backbone is designed to enhance the gated temporal diffusion model by effectively modeling temporal dependencies and managing the distinction between observed and future frames. As the core component of the reverse diffusion process, the gated temporal diffusion model predicts denoised action labels, enabling the generation of multiple plausible future action sequences conditioned on partially observed video data.

The decoder structure of GPDAS is shown in Fig. 2. It is worth noting that the decoder employs a shared structure during both training and inference stages, although the input conditions differ. It is organized as a multi-layer network, with each layer containing multiple blocks of gated temporal convolution. Within these blocks, a gating mechanism adaptively controls the information flow between the observed and future frames. Crucially, by learning the degree of mixing at different network layers, this gating mechanism ensures the model effectively leverages observed data while generating coherent predictions for unseen future classes.

At each step t of the denoising process, the decoder receives an input that combines the current noisy action labels $\widetilde{A}_t$ and the feature sequence of conditions. This sequence may be either $\mathcal{V}$ or $\mathcal{V}\odot Mask$, depending on the phase (training of inference). Denoting $Z_{n,t}$ as the features in decoder layer n at denoising step t, the computation within each block proceeds as follows:

Gated convolution: The gated convolution branch is responsible for learning the soft mask to dynamically select and emphasize feature information that is more important to the current task, while suppressing irrelevant or noisy information. A dilated temporal convolution computes gating values that determine the relevance of input features from the previous layer $Z_{n-1,t}$. This process can be formulated as follows:

$$G_{n,t} = \sigma(W^G_{n,t} \cdot Z_{n-1,t} + b^G_{n,t}), \tag{9}$$

where $W^G_{n,t}$ and $b^G_{n,t}$ are the weight and bias for the n-th decoder layer at the t-th denoising step, σ denotes the sigmoid activation function, and $G_{n,t}$ is the gated feature output for the n-th decoder layer at denoising step t.

Feature Convolution: The feature convolution branch is responsible for extracting potentially relevant features in the action sequence and further computing with the mask generated by the gate convolution branch. Parallel to the branch of gated convolution, the branch of feature convolution operates as follows:

$$F_{n,t} = W^F_{n,t} \cdot Z_{n-1,t} + b^F_{n,t}, \tag{10}$$

where $W^F_{n,t}$ and $b^F_{n,t}$ are the weights and bias of the feature convolution branch, and $F_{n,t}$ is the feature convolution output for the n-th decoder layer at denoising step t. This branch does not employ any activation function. The main reason is that the gated convolution branch has generated mask features which can be regarded as an activation function. Therefore, the feature convolution branch must retain the original features to avoid conflicts between two branches. After obtaining results from both branches, the final output is computed via element-wise multiplication: $Y_{n,t} = G_{n,t} \odot F_{n,t}$, where $Y_{n,t}$ is the fused output of the gated- and feature-convolution branches. The output $Y_{n,t}$ is then passed through a 1×1 convolution followed by the *ReLU* activation function to produce the final result. Subsequently, the addition is produced in element-wise order between the cross-attention branch and $Y_{n,t}$ to produce the output for the n-th decoder layer at denoising step t: $Z_{n,t} = Y_{n,t} + CrossAtten$.

D. Loss Functions:

The loss function of our framework is divided into two components: action loss and boundary loss. As shown in Fig. 1, we use both the encoder's and decoder's outputs to generate losses to further balance the condition generation and prediction process. We denote the encoder features as Y^{en} and the decoder features as Y^{de}.

Action Loss: Action classification is treated as a standard classification task. The loss function used is the standard cross-entropy loss for classification, which minimizes the negative log-likelihood of the ground-truth action class for each frame:

$$\mathcal{L}_{\text{cls}} = -\frac{1}{LC}\sum_{l=1}^{L}\sum_{c=1}^{C} A_{l,c} \log Y^{de/en}_{l,c}, \tag{11}$$

where $logY^{de/en}_{l,c}$ represents the logits from both the decoder and encoder for frame l and class c, and $A_{l,c}$ is an element of the action label for frame l and class c. We use logits from

TABLE I
COMPARISON WITH STAT-OF-THE-ART METHODS. THE BEST PERFORMANCE IS HIGHLIGHTED IN **BOLD**.

	GTEA			50Salads			Breakfast		
Method	F1@{10,,25,50}	Edit	Acc	F1@{10,25,50}	Edit	Acc	F1@{10,25,50}	Edit	Acc
MS-TCN++ [14]	88.8/85.7/76.0	83.5	80.1	80.7/78.5/70.1	74.3	83.7	64.1/58.6/45.9	65.6	67.6
SSTDA [28]	90.0/89.1/78.0	86.2	79.8	83.0/81.5/73.8	75.8	83.2	75.0/69.1/55.2	73.7	70.2
GTRM [29]	-/-/-	-	-	75.4/72.8/63.9	67.5	82.6	57.5/54.0/43.3	58.7	65.0
BCN [1]	88.5/87.1/77.3	84.4	79.8	82.3/81.3/74.0	74.3	84.4	68.7/65.5/55.0	66.2	70.4
MTDA [30]	90.5/88.4/76.2	85.8	80.0	82.0/80.1/72.5	75.2	83.2	74.2/68.6/56.5	73.6	71.0
G2L [15]	89.9/87.3/75.8	84.6	78.5	80.3/78.0/69.8	73.4	82.2	74.9/69.0/55.2	73.3	70.7
HASR [31]	90.9/88.6/76.4	87.5	78.7	86.6/85.7/78.5	81.0	83.9	74.7/69.5/57.0	71.9	69.4
ASRF [32]	89.4/87.8/79.8	83.7	77.3	84.9/83.5/77.3	79.3	84.5	74.3/68.9/56.1	72.4	67.6
ASFormer [33]	90.1/88.8/79.2	84.6	79.7	85.1/83.4/76.0	79.6	85.6	76.0/70.6/57.4	75.0	73.5
UARL [34]	92.7/91.5/82.8	88.1	79.6	85.3/83.5/77.8	78.2	84.1	65.2/59.4/47.4	66.2	67.8
DPRN [35]	92.9/92.0/82.9	90.9	82.0	87.8/86.3/79.4	82.0	87.2	75.6/70.5/57.6	75.1	71.7
SEDT [36]	93.7/92.4/84.0	91.3	81.3	89.9/88.7/81.1	84.7	86.5	-/-/-	-	-
TCTR [3]	91.3/90.1/80.0	87.9	81.1	87.5/86.1/80.2	83.4	86.6	76.6/71.1/58.5	76.1	**77.5**
FAMMSDTN [37]	91.6/90.9/80.9	88.3	80.7	86.2/84.4/77.9	79.9	86.4	78.5/72.9/60.2	77.5	74.8
DTL [38]	-/-/-	-	-	87.1/85.7/78.5	80.5	86.9	78.8/74.5/62.9	77.7	75.8
UVAST [39]	92.7/91.3/81.0	**92.1**	80.2	89.1/87.6/81.7	83.9	87.4	76.9/71.5/58.0	77.1	69.7
Diffact [4]	92.5/91.5/84.7	89.6	**82.2**	90.1/89.2/83.7	85.0	88.9	80.3/75.9/64.6	78.4	76.4
GPDAS, Ours	**93.3/92.6/86.7**	91.3	81.8	**91.3/90.6/85.8**	**86.7**	**89.1**	**80.8/76.2/64.8**	**79.2**	76.8

both the decoder and encoder simultaneously to compute the loss.

Boundary Loss: Boundary loss consists of two components: boundary regression probability loss and boundary classification loss. To promote local similarity along the temporal dimension, the boundary regression loss is computed as the mean squared error of the log-likelihoods between adjacent video frames. This encourages consistency in predictions over time. This loss is formulated as follows:

$$\mathcal{L}_P = -\frac{1}{(L-1)C}\sum_{l=1}^{L-1}\sum_{c=1}^{C}\left\|\log Y_{l,c}^{de/en} - \log Y_{l+1,c}^{de/en}\right\|_2, \tag{12}$$

where $\log Y_{l,c}^{de/en}$ and $\log Y_{l+1,c}^{de/en}$ are logits from the decoder and encoder for frames l and $l+1$, and C represents the number of classes. L2 norm is used in this loss function.

For the boundary classification loss, boundary labels are required for training. For a ground-truth boundary label $AB \in \{0,1\}^{L-1}$, we use a softness calculation based on Gaussian kernel to generate $\widehat{AB}$. The boundary classification can be regarded as binary classification, which can be expressed as follows:

$$\begin{aligned}\mathcal{L}_B = -\frac{1}{(L-1)}\sum_{l=1}^{L-1}\Big(\widehat{AB}\log\left(1-Y_l^{de}\cdot Y_{l+1}^{de}\right)+\\ \left(1-\widehat{AB}\right)\log\left(Y_l^{de}\cdot Y_{l+1}^{de}\right)\Big),\end{aligned} \tag{13}$$

where $\log Y_l^{de}$ and $\log Y_{l+1}^{de}$ represents the logit output only from the decoder for frames l and $l+1$.

The overall loss function used for training is a weighted combination of the three components:

$$\mathcal{L} = \alpha\mathcal{L}_P + \beta\mathcal{L}_B + \gamma\mathcal{L}_{cls}, \tag{14}$$

where α, β, and γ are manually selected hyperparameters that balance the contributions of the three losses.

IV. EXPERIMENTS

A. Datasets.

GTEA: The Georgia Tech Egocentric Activity (GTEA) dataset consists of first-person perspective videos capturing daily activities, such as cooking and cutting. Each video is annotated at the frame level with actions such as "open", "close", and "pour". Each video is a clip with a duration of one minute. **50 Salads:** The 50 Salads dataset comprises third-person perspective videos of individuals preparing various salads. Each video captures a distinct salad-making process, featuring frame-level annotated actions such as "cut tomato", "peel cucumber", and "mix ingredients". This dataset serves as a standard benchmark for action segmentation and anticipation tasks, offering a controlled environment for studying sequential cooking actions. **Breakfast:** The Breakfast dataset is a large-scale collection of third-person perspective videos depicting the preparation of breakfast meals. It features a diverse range of frame-level annotated actions, including "pour milk", "stir coffee", and "fry egg" across numerous videos. All three datasets are divided into four subsets. We perform training and testing on each subsection and report the results in terms of three evaluation metrics.

B. Evaluation Metrics.

We evaluated performance using three standard metrics commonly adopted in prior work: (1) Frame-wise Accuracy (Acc) measures the proportion of correctly classified video frames. (2) Edit Score (Edit) computes the minimum number of editing operations required to transform the predicted action sequence into the ground truth sequence by using Levenshtein distance. It strictly measures the action sequence between the prediction and the ground truth at the frame level. (3) Segmental F1 Scores (F1 @ 10, 25, 50) calculate the harmonic mean of precision and recall for detection of action segments at three temporal overlap thresholds: $10\%, 25\%$ and 50%.

C. Implementation Details.

We use I3D-pretrained features [40], following the same pre-processing steps as in prior work. The input video feature of the encoder V has a dimensionality of 2048. Both the encoder $E_\phi(\cdot)$ and decoder $D_\omega(\cdot)$ are implemented using the ASFormer architecture [33]. Encoder and decoder configurations vary across datasets. For the 50 Salads and GTEA datasets, the encoder consists of 10 layers with hidden dimension of 64, while

the decoder contains 8 layers with hidden feature dimension of 24. For the Breakfast dataset, the encoder has 10 layers with 256-dimensional hidden features, while the decoder uses 8 layers with 128-dimensional hidden features.

As illustrated in Fig. 1, we extract multiscale features from multiple encoder layers. The features are concatenated, and subsequently masking is applied to condition the decoder. Specifically, we use features from the 5th, 7th, and 9th encoder layers for GTEA and 50 Salads, and from the 7th, 8th, and 9th layers for Breakfast.

D. Comparison to State-of-the-art:

Our proposed GPDAS framework demonstrates strong performance compared to state-of-the-art methods across three benchmark datasets: GTEA, 50 Salads, and Breakfast. As shown in Tab. I, GPDAS achieved outstanding results in 50 Salads and Breakfast, while delivering competitive performance in GTEA. In 50 Salads, our method improves the Edit metric to 86.7, surpassing recent approaches such as Diffact (85.0) and SEDT (84.7). In Breakfast, GPDAS increases the Edit metric to 79.2, outperforming strong baselines like DTL (77.7) and Diffact (78.4). In GTEA, GPDAS achieved competitive results, particularly excelling in frame-wise accuracy. The gated decoder in GPDAS performs especially well in larger and more complex datasets (Breakfast, 50Salads), demonstrating its effectiveness in modeling action dynamics. It serves as a powerful paradigm for diffusion-based action segmentation, effectively capturing temporal dependencies and mitigating over-segmentation errors through iterative denoising.

V. Conclusions

In this work, we have introduced an improved framework for temporal action segmentation that employs denoising diffusion models to iteratively generate action sequences conditioned on video features. Our approach, named Gated Probabilistic Diffusion Action Segmentation (GPDAS), formulates action segmentation within a generative framework. This enables the model to capture the prior distribution of human actions while iteratively refining predictions through a denoising process. The gated decoder architecture further enhances the system by dynamically selecting features to generate refined segmentation outputs. This improves the model's ability to handle complex temporal dynamics and long-range dependencies inherent in video data.

References

[1] Z. Wang, Z. Gao, L. Wang, Z. Li, and G. Wu, "Boundary-aware cascade networks for temporal action segmentation," in *ECCV*, 2020.

[2] D. Wang, Y. Yuan, and Q. Wang, "Gated forward refinement network for action segmentation," *Neurocomputing*, 2020.

[3] N. Aziere and S. Todorovic, "Multistage temporal convolution transformer for action segmentation," *Image and Vision Computing*, 2022.

[4] D. Liu, Q. Li, A.-D. Dinh, T. Jiang, M. Shah, and C. Xu, "Diffusion action segmentation," in *CVPR*, 2023.

[5] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, 2020.

[6] C. Zhang, T. Liu, J. Xiao, K.-M. Lam, and Q. Wang, "Boosting object detectors via strong-classification weak-localization pretraining in remote sensing imagery," *IEEE Transactions on Instrumentation and Measurement*,

[7] C. Zhang, J. Xiao, C. Yang, J. Zhou, K.-M. Lam, and Q. Wang, "Integrally mixing pyramid representations for anchor-free object detection in aerial imagery," *IEEE Geoscience and Remote Sensing Letters*, 2024.

[8] Y. Li, H. Xie, J. Xiao, C. Zhang, T. Liu, and K.-M. Lam, "Hierarchical vertex-wise intensification graph convolution for skeleton-based activity recognition," in *IEEE ICIP*, 2024.

[9] H. Xie, Z. Huang, F. H. Leung, Y. Ju, Y.-P. Zheng, and S. H. Ling, "A structure-affinity dual attention-based network to segment spine for scoliosis assessment," in *Conference on BIBM*, 2023.

[10] H. Xie, Z. Huang, F. H. Leung, *et al.*, "Satr: A structure-affinity attention-based transformer encoder for spine segmentation," in *IEEE ISBI*, 2024.

[11] C. Zhang, T. Liu, Y. Ju, and K.-M. Lam, "Pyramid masked image modeling for transformer-based aerial object detection," in *IEEE ICIP*, 2023.

[12] C. Zhang, Q. Wang, and X. Li, "Iq-stan: Image quality guided spatio-temporal attention network for license plate recognition," in *IEEE ICASSP*, 2020.

[13] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *CVPR*, 2017.

[14] Y. A. Farha and J. Gall, "Ms-tcn: Multi-stage temporal convolutional network for action segmentation," in *CVPR*, 2019.

[15] S.-H. Gao, Q. Han, Z.-Y. Li, P. Peng, L. Wang, and M.-M. Cheng, "Global2local: Efficient structure search for video action segmentation," in *CVPR*, 2021.

[16] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *ICLR*, 2021.

[17] Z. Lyu, J. Xiao, C. Zhang, and K.-M. Lam, "Ai-generated image detectors are surprisingly easy to mislead… for now," in *APSIPA ASC*, IEEE, 2024.

[18] A. K. Bhunia, S. Khan, H. Cholakkal, *et al.*, "Person image synthesis via denoising diffusion model," in *CVPR*, 2023.

[19] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *CVPR*, 2022.

[20] P. Yu, S. Xie, X. Ma, *et al.*, "Latent diffusion energy-based model for interpretable text modeling," *ICML*, 2022.

[21] G. Kim and J. C. Ye, "Diffusionclip: Text-guided image manipulation using diffusion models," in *CVPR*, 2022.

[22] J. Xiao, C. Shui, Z.-S. Liu, Q. Ye, and K.-M. Lam, "Learning equilibrium transformation for gamut expansion and color restoration," in *ECCV*, 2024.

[23] J. Xiao, Z. Lyu, C. Zhang, Y. Ju, C. Shui, and K.-M. Lam, "Towards progressive multi-frequency representation for image warping," in *CVPR*, 2024.

[24] J. Xiao, X. Jiang, N. Zheng, *et al.*, "Online video super-resolution with convolutional kernel bypass grafts," *IEEE Transactions on Multimedia*, 2023.

[25] Y. Leng, Z. Chen, J. Guo, *et al.*, "Binauralgrad: A two-stage conditional diffusion probabilistic model for binaural audio synthesis," *Advances in Neural Information Processing Systems*, 2022.

[26] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *ICLR*, 2021.

[27] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *CVPR*, 2022.

[28] M.-H. Chen, B. Li, Y. Bao, G. AlRegib, and Z. Kira, "Action segmentation with joint self-supervised temporal domain adaptation," in *CVPR*, 2020.

[29] Y. Huang, Y. Sugano, and Y. Sato, "Improving action segmentation via graph-based temporal reasoning," in *CVPR*, 2020.

[30] M.-H. Chen, B. Li, Y. Bao, and G. AlRegib, "Action segmentation with mixed temporal domain adaptation," in *WACV*, 2020.

[31] H. Ahn and D. Lee, "Refining action segmentation with hierarchical video representations," in *CVPR*, 2021.

[32] Y. Ishikawa, S. Kasai, Y. Aoki, and H. Kataoka, "Alleviating over-segmentation errors by detecting action boundaries," in *WACV*, 2021.

[33] F. Yi, H. Wen, and T. Jiang, "Asformer: Transformer for action segmentation," in *BMVC*, 2021.

[34] L. Chen, M. Li, Y. Duan, J. Zhou, and J. Lu, "Uncertainty-aware representation learning for action segmentation.," in *IJCAI*, 2022.

[35] J. Park, D. Kim, S. Huh, and S. Jo, "Maximization and restoration: Action segmentation through dilation passing and temporal reconstruction," *Pattern Recognition*,

[36] G.-h. Kim and E. Kim, "Stacked encoder–decoder transformer with boundary smoothing for action segmentation," *Electronics Letters*, 2022.

[37] Z. Du and Q. Wang, "Dilated transformer with feature aggregation module for action segmentation," *Neural Processing Letters*, 2023.

[38] Z. Xu, Y. Rawat, Y. Wong, M. S. Kankanhalli, and M. Shah, "Don't pour cereal into coffee: Differentiable temporal logic for temporal action segmentation," *Advances in Neural Information Processing Systems*, 2022.

[39] N. Behrmann, S. A. Golestaneh, Z. Kolter, J. Gall, and M. Noroozi, "Unified fully and timestamp supervised temporal action segmentation via sequence to sequence translation," in *ECCV*.

[40] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *CVPR*, 2017.