OnlineTAS: An Online Baseline for Temporal Action Segmentation

Qing Zhong[1,2*] **Guodong Ding**[2*] **Angela Yao**[2]
[1]University of Adelaide [2]National University of Singapore
qing.zhong@adelaide.edu.au {dinggd,ayao}@comp.nus.edu.sg

Abstract

Temporal context plays a significant role in temporal action segmentation. In an offline setting, the context is typically captured by the segmentation network after observing the entire sequence. However, capturing and using such context information in an online setting remains an under-explored problem. This work presents the an online framework for temporal action segmentation. At the core of the framework is an adaptive memory designed to accommodate dynamic changes in context over time, alongside a feature augmentation module that enhances the frames with the memory. In addition, we propose a post-processing approach to mitigate the severe over-segmentation in the online setting. On three common segmentation benchmarks, our approach achieves state-of-the-art performance.

1 Introduction

This work addresses online temporal action segmentation (TAS) of untrimmed videos. Such videos typically feature procedural activities consisting of multiple actions or steps in a loose temporal sequence to achieve a goal [6]. For example, "making coffee" has actions: 'take cup', 'pour coffee', 'pour water', 'pour milk', 'pour sugar' and 'stir coffee'. Standard TAS models [11, 45, 25, 37] are offline and segment-only videos of complete procedural activities. An *online* TAS model, in contrast, segments only up to the current time point and does not have access to the entire video and, therefore, the entire activity.

Online TAS faces challenges similar to other online tasks [44, 46] in establishing a scalable network that can retain useful information from an ever-increasing volume of data and facilitate effective retrieval when required. Additionally, over-segmentation is a common issue for offline TAS, where the segmentation model divides an action into many discontinuous sub-segments, leading to fragmented outputs. This issue is exacerbated in the online setting, as partial data at the onset of an action may lead to erratic predictions and increased over-segmentation.

Most relevant to our task is online temporal action detection (TAD) [43]. Online TAD aims to identify whether an action is taking place and the action category. TAD targets datasets like THUMOS [16], TVSeries [5], and HACS Segment [48]. Among these, 90.8% videos of THUMOS [16] feature only multiple instances of the same action, while TVSeries [5] comprises diverse yet independent actions (*e.g.*, 'open door', 'wave' and 'write') in one video. These actions do not necessarily correlate with one another or impose specific temporal constraints. As such, a direct adaptation of popular online TAD approaches like LSTR [44] and MAT [41] to online segmentation is non-ideal. For instance, these models encode temporal context with a fixed set of tokens, which may limit their capability to handle the relations of procedural videos. Furthermore, these models are typically trained to prioritize frame-level accuracy while neglecting temporal continuity, which invariably leads to over-segmentation.

*Equal Contribution. The work was done when Qing Zhong was an intern in National University of Singapore.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

To address the online action segmentation task, this work proposes a novel framework centered on a context-aware feature augmentation module and an adaptive memory bank . The memory bank, per-video, tracks short-term and long-term context information. The augmentation module uses an attention mechanism to allow frame features to interact with context information from the memory bank and integrate temporal information into standard frame representations. Finally, we introduce a post-processing technique for online boundary adjustment that imposes duration and prediction confidence constraints to mitigate over-segmentation.

Summarizing our contributions, **1)** We establish an *online* framework for TAS; **2)** We propose a feature augmentation module that generates context-aware representations by incorporating an adaptive memory, which accumulates temporal context collectively. The module operates on frame features independently of model architecture, enabling flexible integration; **3)** We present a simple post-processing technique for online prediction adjustment, which can effectively mitigate the over-segmentation problem; and **4)** Our framework achieves the state-of-the-art online segmentation performance on three TAS benchmarks.

2 Related Work

Online Action Understanding. Many video understanding tasks, such as action detection [44, 42, 5, 2] and video instance segmentation [15, 46, 47], have been explored in online contexts. For online action detection, videos are classified frame by frame without access to future frames. Specifically, LSTR [44] employs a novel memory mechanism to model long- and short-term temporal dependencies by encoding them as query tokens. Follow-up works feature a segment-based long-term memory compression [41] and fusing short- and long-term histories via attention [2].

However, the videos in the datasets commonly used in online TAD contain independent actions [5] or sequences of limited actions [16], thus lacking temporal relations between the actions. In contrast, TAS deals with untrimmed procedural videos, where such relations are more prominent and may span over long temporal durations. There is also a growing trend in online TAD models to use action anticipation as an auxiliary task to enhance action modeling [2, 14]. In our online segmentation task, we do not assume the availability of such information.

Temporal Action Segmentation. In TAS [6], methods vary by their level of supervision, including fully [11, 20, 27], semi-supervised [8, 36], weakly [9, 13, 22, 30, 29, 31, 21], and unsupervised [19, 34, 32, 10, 9] setups. An emerging direction is to learn TAS incrementally [7] where procedural activities are learned sequentially. However, all existing works are offline, and complete video sequences can be used for inference. In contrast, our approach functions within an online setup. The most related work [13] investigates online TAS in a multi-view setup and leverages the offline model to assist online model learning. Furthermore, it uses the frame-wise multi-view correspondence to generate pseudo-labels for action segments. In contrast, we do not assume the availability of multi-view videos nor require assistance from a pre-trained offline model.

Post-processing for Action Segmentation. Post-processing methods are either rule-based or leverage graphical modelling. Rule-based approaches [28, 35, 40, 38] apply predefined rules to smooth out short-duration predictions that are unlikely given the context. Graphical modelling approaches use Conditional Random Fields (CRFs) [17, 31, 30] to model the relationships and transitions between consecutive actions. Online methods need post-processing to alleviate over-segmentation and help locate the action boundaries.

3 Online Action Segmentation

Previous studies [1, 33] have demonstrated that the scope of the temporal context significantly influences the performance of (offline) TAS models. This motivates our two lines of inquiry for our online setting: 1) how to consolidate temporal context over an extended period, and 2) how to enrich the frame representations with context to benefit the segmentation. This work introduces a context-aware feature augmentation module (Sec. 3.2) alongside an adaptive context memory (Sec. 3.3) to tackle the above questions.

3.1 Preliminaries

Consider an untrimmed video $v = \{x_t\}_{t=1}^{T}$ of T frames, where $x_t \in \mathbb{R}^D$ is the per-extracted frame feature at time t and D is the feature dimension. An action segmentation model partitions v into N contiguous and non-overlapping temporal segments $\{s_i = (y_i, \ell_i)\}_{i=1}^{N}$ corresponding to actions $y \in \mathcal{Y}$ present in the video, where ℓ indicates the segment length and $\mathcal{Y}$ defines the action space [6]. A widely adopted strategy for TAS is to design a segmentation model $\mathcal{M}$ that predicts the action label y_t for each frame x_t, akin to a frame-wise classification. In the offline setting [11, 45, 25], the per-frame prediction $\hat{y}_t$ is based on the entire video sequence. The online setting uses only frames up to the current prediction time t, without access to future frames. Comparatively:

$$\hat{y}_t^{\text{offline}} = \arg\max_{y \in \mathcal{Y}} p_t(y_t | x_t; x_{1:T}) \quad \text{and} \quad \hat{y}_t^{\text{online}} = \arg\max_{y \in \mathcal{Y}} p_t(y_t | x_t; x_{1:t}). \tag{1}$$

where $p_t \in \mathbb{R}^{|\mathcal{Y}|}$ is the estimated action probability for frame x_t. Most existing offline TAS works [11, 45, 23] train the segmentation model $\mathcal{M}$ using a cross-entropy loss ($\mathcal{L}_{\text{cls}}$) for frame-wise classification which is balanced by a smoothing loss ($\mathcal{L}_{\text{sm}}$) that encourages smooth transitions between consecutive frames:

$$\mathcal{L} = \underbrace{\frac{1}{T} \sum_{t} -\log(p_t(y_t))}_{\mathcal{L}_{\text{cls}}} + \lambda \cdot \underbrace{\frac{1}{T|\mathcal{Y}|} \sum_{t,y} \tilde{\Delta}_{t,y}^2}_{\mathcal{L}_{\text{sm}}}, \tag{2}$$

$$\text{where} \quad \tilde{\Delta}_{t,y} = \begin{cases} \Delta_{t,y} : \Delta_{t,y} \leq \tau \\ \tau \quad : \text{otherwise} \end{cases} \quad \text{and} \quad \Delta_{t,y} = \left| \log p_t(y) - \log p_{t-1}(y) \right|.$$

In this paper, we opt for the widely recognized convolution-based architecture MS-TCN [11] as our foundational framework. This choice is driven by its relatively lower computational requirements than the attention- or diffusion-based models [45, 25]. A straightforward transition from offline mode to an online mode of the segmentation model $\mathcal{M}$ is to substitute standard convolutions with causal convolutions. Causal and standard convolutions differ in their receptive field in that causal convolutions consider only past and present inputs while standard convolutions may incorporate both past and future inputs within a kernel. Mathematical details and illustrations of the two are shown in the Appendix.

3.2 Context-aware Feature Augmentation

The context-aware feature augmentation (CFA) module generates enhanced clip-wise features through interactions with temporal context captured by an adaptive memory bank. The module operates on a clip-wise basis. During training, video v is split into K non-overlapping clips $\{c_k\}_{k=1}^{K}$. Each clip has a window size w and is sampled from v with a stride of $\delta = w$, where the final clip c_K is padded if $|c_K| < w$. The CFA module integrates the original pre-extracted frame features $c_k = \{x_t\}_{t=(k-1)w+1}^{kw}$ with temporal context to produce a context-enhanced version of representations $\tilde{c}_k = \{\tilde{x}_t\}_{t=(k-1)w+1}^{kw}$. Like [44, 41], our CFA module is also equipped with a simultaneously updated memory bank M_k as a context resource for feature augmentation. The memory bank is further described in Sec. 3.3.

At each step k, context is accumulated by feeding c_k through a lightweight GRU [4] to obtain c_k^{GRU}. The GRU is reliable in capturing information over long video sequences [24]. The clip is then passed through a context aggregation block to be augmented. The context aggregation block incorporates the GRU features c_k^{GRU} with the memory state M_{k-1} from the previous step for I iterations. Concretely, we pass c_k^{GRU} through a self-attention (SA) block to encourage information exchange with the local clip window. Additionally, we leverage a Transformer decoder [39] to achieve a more effective memory encoding $\tilde{M}_{k-1}^{\text{TD}}$, *i.e.*,

$$c_k^{\text{SA}} = \text{SelfAttn}(c_k^{\text{GRU}}) \quad \text{and} \quad \tilde{M}_{k-1}^{\text{TD}} = \text{TransDecoder}(M_{k-1}, c_k^{\text{GRU}}, c_k^{\text{GRU}}). \tag{3}$$

The outputs from the self-attention module (c_k^{SA}) and the transformer decoder (c_k^{TD}) are then combined with cross attention (CA) and merged with c_k^{GRU} to produce the context-augmented features:

$$\tilde{c}_k = \text{CrossAttn}(c_k^{\text{SA}}, \tilde{M}_{k-1}^{\text{TD}}, \tilde{M}_{k-1}^{\text{TD}}) + c_k^{\text{GRU}}. \tag{4}$$

The detailed formulas of SelfAttn(), TransDecoder(), CrossAttn() are given in the Appendix. An illustration of our CFA module is provided in Fig. 1.

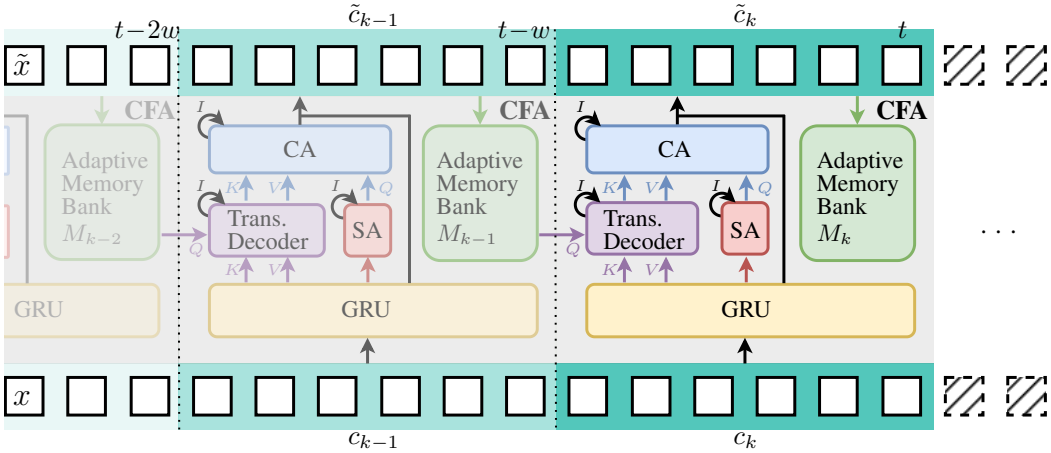


Figure 1: Context-aware Feature Augmentation (CFA) module. CFA takes as input a video clip c_k of length w, augments it with temporal information captured in an adaptive memory bank M_k, and outputs an enhanced clip feature $\tilde{c}_k$. I is the number of iterations of SA, TransDecoder, and CA.

3.3 Adaptive Memory Bank

In a similar spirit with [44], our memory is designed to account for both short- and long-term context, *i.e.*, $M = [M^{\text{long}}, M^{\text{short}}]$. Short-term memory helps capture the local action dynamics while long-term memory retains information across extended durations important for TAS [1, 33].

Short Memory M^{short}. Given that our enhancement module works on a per-clip basis with temporal stride w, we directly regard the enhanced feature $\tilde{c}_{k-1}$ from the last clip as the short-term memory, *i.e.*, $M_k^{\text{short}} = \tilde{c}_{k-1} \in \mathbb{R}^{Dw}$.

Long Memory M^{long}. We update our long-term memory with information from processed clips. Specifically, we apply a convolutional layer on top of our context-enhanced representation $\tilde{c}_k$, where $\tilde{c}_k \in \mathbb{R}^{Dw}$, to collapse the temporal dimension and yield a memory token $m_k = \text{Conv1D}(\tilde{c}_k) \in \mathbb{R}^D$. This memory token is then appended to the current long-term memory.

Algorithm 1 Adaptive Memory Update

Require: $\{c_k\}_{k=1}^K$, w

Initialize $M_0^{\text{short}} \leftarrow c_1$, $M_0^{\text{long}} \leftarrow \varnothing$
for $k \in [1...K]$ **do**
  $\tilde{c}_k = \text{CFA}(c_k, M_{k-1})$
  $m_k = \text{Conv1D}(\tilde{c}_k)$
  if $\text{len}(M_{k-1}^{\text{long}}) \leq \frac{2}{3}w$ **then**
    $M_k^{\text{long}} = \text{concat}(M_{k-1}^{\text{long}}, m_k)$
  else
    $M_k^{\text{long}} = \text{concat}(M_{k-1}^{\text{long}}[1:], m_k)$
  end if
  $M_k^{\text{short}} = \tilde{c}_{k-1}[\text{len}(M_k^{\text{long}}):]$
  $M_k = [M_k^{\text{long}}, M_k^{\text{short}}]$
end for

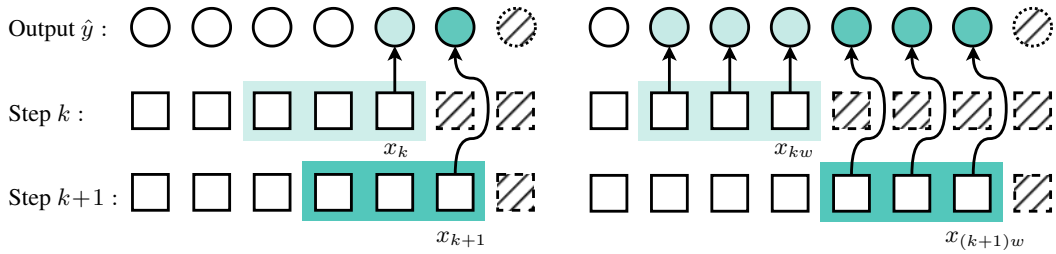
Adaptive Memory Update. The memory is updated whenever a new clip is processed. In practice, we constrain the total footprint of both short- and long-term memory to match the size of the processing clip, *i.e.*, $M \in \mathbb{R}^{Dw}$. At the beginning of each sequence, the memory bank is initialized with short-term information only, *i.e.*, $M = M_0^{\text{short}} = c_1$ and $M_0^{\text{long}} = \varnothing$. As more clips are processed, we gradually increase the budget to accommodate longer-term information. However, in anticipation of M^{long} draining the entire budget in prolonged sequences, we cap its utilization at a maximum of two-thirds of the total budget. In instances where this threshold is exceeded, the earliest token is discarded, *i.e.*,:

$$M_k^{\text{long}} = \begin{cases} \text{concat}(M_{k-1}^{\text{long}}, m_k) & : \text{len}(M_{k-1}^{\text{long}}) \leq \frac{2}{3}w \\ \text{concat}(M_{k-1}^{\text{long}}[1:], m_k) & : \text{otherwise} \end{cases}. \tag{5}$$

The remaining budget is allocated for the short-term information accordingly:

$$M_k^{\text{short}} = \tilde{c}_{k-1}[\text{len}(M_k^{\text{long}}):] \tag{6}$$

As video progresses, our feature augmentation module (Sec. 3.2) receives more longer-term context while emphasizing only shorter and more relevant short-term information. This adaptive approach



(a) Online Inference ($\delta = 1$) (b) Semi-online Inference ($\delta = w$)

Figure 2: Two inference types. a) Online inference samples clips with stride $\delta = 1$ and only preserves the last frame prediction, while b) Semi-online inference samples non-overlapping clips with stride $\delta = w$ and all predictions are preserved.

enables the context memory to flexibly shift its attention between short and long-term information as the video progresses. Algorithm 1 summarizes the update mechanism.

Discussion. Our module integrates context memory on top of a GRU layer. While the GRU captures temporal dependencies, it may struggle, especially in thousands-frame long sequences common in TAS. The context memory supplements the GRU by allowing selective access and updates, thereby enabling the retrieval and manipulation of long-term information. Such a design is supported by our empirical study that the explicit memory can extend the capacity of the GRU's internal state. Supporting ablations are found in Sec. 4.1.

3.4 Training and Inference

Training. Our final online segmentation model is constructed by combining our CFA module with a single-stage TCN [11] with causal convolutions. Specifically, TCN takes as input the enhanced representations $\tilde{c}_k$ and maps them to the same labels for c_k. We train the framework end-to-end with the loss function formulated in Eq. (2), but on a clip basis with T replaced by w:

$$\mathcal{L}^{\text{clip}} = \frac{1}{w} \sum_{t} -\log(p_t(y_t)) + \lambda \cdot \sum_{t,y} \tilde{\Delta}^2_{t,y}, \quad (7)$$

we set $\lambda = 0.15$ following [11]. Like [44], training on a clip basis provides better efficiency.

Algorithm 2 Post-processing for Online TAS

```
Compute ℓ_min = σ × T_max
Initialize ℓ = 0
for each frame t do
    if q_t < θ and ℓ < ℓ_min then
        ŷ*_t = ŷ*_{t−1}
        ℓ = ℓ + 1
    else
        ŷ*_t = ŷ_t
        ℓ = 0
    end if
end for
```

Inference. We present two distinct inference approaches by manipulating the clip sampling stride parameter δ. The first mode of inference, referred to as *online*, is characterized by setting $\delta = 1$. In such a setting, a video clip of w frames is processed, with emphasis placed solely on the prediction derived from the final frame and rest are discarded. This facilitates the scenario when frame-by-frame prediction is preferred. The alternate mode of inference, termed *semi-online*, adheres to the training regime by setting $\delta = w$. In this mode, video clips are processed, and the dense predictions generated across all w frames are preserved as final output. An illustration of two modes of inference is provided in Fig. 2.

3.5 Post-processing

Our intuition is that a valid action segment should not fall below a minimum length threshold unless there is high confidence in the prediction to justify a change in the action class. Specifically, we consider the maximum softmax probability of a prediction as its confidence measure, denoted as $q_t = \max(p_t)$, for frame x_t, as q_t to some extent indicates its reliability. A prediction is considered "unreliable" if its confidence measure scores below a certain threshold θ, *i.e.*, $q_t < \theta$. For the frame with "unreliable" prediction, we disregard its current prediction and assign the action label of its proceeding frame $\hat{y}^*_{t-1}$, when the previous action segment is shorter than the minimum length.

	p.p.	GTEA [12]					50Salads [38]					Breakfast [18]				
		Acc	Edit	F1 @ {10, 25, 50}			Acc	Edit	F1 @ {10, 25, 50}			Acc	Edit	F1 @ {10, 25, 50}		
Online																
TCN	-	74.4	66.6	73.9	70.3	57.2	75.2	19.6	26.8	24.4	19.6	55.3	18.7	15.1	11.7	8.3
TCN	✓	72.1	**71.9**	**79.2**	**77.4**	64.1	75.1	68.5	74.1	70.6	60.4	52.3	54.7	52.0	43.2	29.8
Ours	-	**76.2**	63.5	72.6	68.3	58.8	**80.9**	28.8	36.1	31.0	23.3	**56.7**	19.3	16.8	13.9	9.3
Ours	✓	74.4	70.3	78.5	76.4	**67.7**	77.7	**71.5**	**77.7**	**74.6**	**64.1**	52.9	**55.7**	**54.8**	**45.8**	**30.5**
Semi-online																
TCN	-	75.8	66.8	74.3	71.5	60.3	79.1	29.0	38.5	35.5	28.3	55.7	18.6	15.4	12.7	9.0
TCN	✓	73.5	75.4	80.3	76.9	66.6	76.7	69.2	73.1	70.5	62.8	52.5	54.0	53.1	44.5	29.6
Ours	-	**77.1**	68.1	76.7	73.5	63.9	**82.4**	32.8	43.0	41.1	34.7	**57.4**	19.6	17.8	14.8	10.1
Ours	✓	76.0	**79.7**	**84.9**	**81.4**	**69.2**	79.4	**75.0**	**82.5**	**80.2**	**68.0**	53.8	**57.5**	**56.4**	**47.3**	**31.4**

Table 1: Performance of our approach on three TAS benchmarks under two inference mode, *i.e.*, online and semi-online. Post-processing is indicated by p.p..

Otherwise, we retain the original prediction. We set the length threshold $\ell_{\text{min}} = \sigma \times T_{\text{max}}$ with $\sigma \in (0, 1)$, in proportion to to the longest video duration T_{max} in training set.

Our post-processing mitigates the over-segmentation by adjusting action boundaries according to network predictions and action length, which is very efficient compared to [32, 10] that calculates frame similarities. The procedure for post-processing is illustrated in Algorithm 2.

4 Experiments

Datasets: We evaluate our model on three common TAS datasets. **Breakfast** [18] comprises in total 1,712 videos performing ten different activities with 48 actions. On average, each video contains six action instances. **50Salads** [38] has 50 videos with 17 action classes. **GTEA** [12] contains 28 videos of seven kitchen activities composing 11 different actions. We use common I3D features [3] as input.

Evaluation Metrics: Standard evaluation metrics for TAS are reported for our online setting, which includes frame-wise accuracy (Acc), segmental edit score (Edit), and segmental F1 scores with varying overlap thresholds 10%, 25%, and 50%.

Implementation Details. In CFA, we stack 2 Transformer decoder layer with 8 heads, 2 Swin [26] self- and cross attention with 4 heads. We use a single-stage TCN as segmentation backbone and sample non-overlapping clips *i.e.*, $\delta = w$ for efficiency. We train the model end-to-end with a learning rate of $5e^{-4}$ of total 50 epochs. Detailed hyperparameter settings can be found in Appendix.

4.1 Experiment Results

Effectiveness. We report the overall performance for *online* and *semi-online* inference (see Sec. 3.4) in in Table 1. Our baseline is a single-stage causal TCN and we build our framework on top of it. Across all three datasets, the integration of our CFA module leads to a consistent boost in the segmentation performance. Specifically, our approach gained 5.7% (75.2% vs. 80.9%) in Acc and 9.2% (19.6% vs. 28.8%) in Edit on 50Salads. While the improvements on other datasets are not as significant, they still show effectiveness, with a margin of about 2%. Generally, semi-online inference achieves better performance over online across all metrics. Such improvement is likely because clip-wise prediction better preserves the local temporal continuity of labels compared to step-by-step single frame prediction.

Comparing across the metrics, segmental scores appear to be significantly low. On breakfast [18] with our online inference, a frame-wise accuracy of 56.7% only corresponds to a 9.3% F1 score with 50% IoU. Such score indicates a severe over-segmentation issue and necessitates an effective post-processing. However, a significant performance increase is observed on Edit and F1 scores after our proposed pose-processing. For example, the same F1 score increases to 30.5%, tripling its original value. Although post-processing could lead to a slight decrease in accuracy, it demonstrates great effectiveness in mitigating the over-segmentation problem.

GRU	CFA	Mem.	Acc	Edit	F1 @ {10, 25, 50}		
-	-	-	75.2	19.6	26.8	24.4	19.6
✓	-	-	78.1	27.1	37.9	34.7	26.7
-	✓	-	76.2	22.3	30.1	27.0	21.9
✓	✓	-	79.1	29.0	38.5	35.5	28.3
-	✓	✓	78.9	29.2	38.7	35.1	28.8
✓	✓	✓	**82.4**	**32.8**	**43.0**	**41.1**	**34.7**

Table 2: Ablation study of module components on 50Salads [38].

I	Acc	Edit	F1 @ {10, 25, 50}		
1	79.1	29.0	38.5	35.5	28.3
2	**79.6**	30.7	**40.7**	**38.2**	**31.4**
3	79.5	28.5	37.2	36.1	29.0
4	79.1	29.2	39.1	36.3	30.5
5	79.2	**30.8**	39.1	37.7	30.6

Table 3: Effect of interactions I.

Ablation study. Table 2 evaluates the components in our CFA module. The first row is our single-layer causal TCN baseline with strong frame-wise accuracy but poor segmental metrics. The GRU boosts segment metrics (7-11%) over the baseline, showing its ability to accumulate context information. While CFA using the current clip as pseudo memory predictably leads to a performance drop (5%) compared to GRU due to lack of any context information. Combining either GRU or our adaptive memory with our CFA achieves very close performance (rows 4 and 5), highlighting the importance of the context information for TAS. The complete model yields the best performance and boosts Acc by 7% and average segmental scores by 15.3%. This validates the complementary effect of GRU's internal state and our explicit memory design.

Number of layers in CFA. Table 3 explores the interaction iterations I in CFA. The results indicate that the performance is not significantly affected by the number of iterations. In practice, we set the number of iterations to 2, as it achieves a good balance between performance and efficiency.

M^{short}	M^{long}	Acc	Seg.
✓	-	80.3	36.7
-	✓	80.4	36.4
✓	✓	**82.4**	**37.9**

Table 4: Effect of memory composition.

w / len(M)	16	32	64	128	256
Acc	79.8	81.4	81.6	**82.4**	80.7
Seg.	35.1	36.5	37.6	37.9	**38.2**

Table 5: Effect of clip size and memory length. "Seg." indicates the mean of Edit and F1 scores.

Memory composition (M^{short}/M^{long}). We assess the impact of memory types and present results in Table 4. It shows comparable performances for each memory type when considered individually. However, the combination of both yields a 2% improvement in Acc and 1.5% for averaged segmental metric, suggesting the significance of incorporating diverse memory for TAS.

Clip size w **/ Memory size** len(M). In our implementation, we set clip size and memory size to be equal and we report its influence on performance in Table 5. It shows a larger clip size leads to better segmental results; this is because temporal continuity can be better modeled with longer clips for learning. However, the information of short actions could be diluted when compressed to form the memory token if the window size is too large.

Post-processing hyperparameters. Two hyperparameters are defined in our post-processing: confidence threshold θ and the minimum segment length ℓ_{min}. We vary its scaling factor σ to assess ℓ_{min}. In Table 6, increasing θ greatly enhances the segment results, with a 18.1% increase observed when $\theta = 0.7$. Although the accuracy tends to decrease as θ becomes larger, the drop is not as substantial (3.1%) compared to the improvements in segmental results. While Table 7 shows the segmental performance stops increasing and stays stable when $\sigma > \frac{1}{16}$ with a fixed confidence score $\theta = 0.9$. In conclusion, employing a higher confidence threshold can help better mitigate the over-segmentation because it makes more sense to prioritize preserving the continuity of a segment that includes frames with highly confident predictions given a fixed length budget.

4.2 Comparison with State-of-the-Art Methods

Tables 8 and 9 compare our approach against the state-of-the-art TAS approaches on all three benchmarks. Due to the absence of dedicated online TAS methods, we benchmark against the online

θ	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Acc	82.5	**82.6**	82.6	82.6	81.2	81.1	79.4
Seg.	50.9	51.8	51.9	51.9	69.0	73.3	**76.4**

Table 6: Effect of confidence threshold θ ($\sigma = \frac{1}{16}$).

σ	$\frac{1}{64}$	$\frac{1}{32}$	$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{4}$
Acc	**79.6**	79.4	79.4	79.4	79.4
Seg.	70.9	74.7	**76.4**	76.4	76.4

Table 7: Effect of minimum length factor σ with $\theta = 0.9$.

	Method	GTEA [12]					50Salads [38]				
		Acc	Edit	F1 @ {10, 25, 50}			Acc	Edit	F1 @ {10, 25, 50}		
offline	MS-TCN [11]	78.7	84.0	88.3	86.6	72.8	81.2	65.8	72.8	70.4	61.7
	MS-TCN + p.p.	78.7	85.2	89.6	88.3	73.3	80.4	74.1	82.0	79.2	70.2
	ASFormer [45]	79.7	84.6	90.1	88.8	79.2	85.6	79.6	85.1	83.4	76.0
	DiffAct [25]	82.2	89.6	92.5	91.5	84.7	87.4	88.9	90.1	89.2	83.7
online	LSTR [44]	63.7	33.2	41.5	37.7	25.0	60.5	5.0	8.2	6.6	4.1
	Causal TCN	74.4	66.6	73.9	70.3	57.2	75.2	19.6	26.8	24.4	19.6
	Ours$^{\text{online}}$	75.8	66.8	74.3	71.5	60.3	79.1	29.0	38.5	35.5	28.3
	Ours$^{\text{online}}$ + p.p.	73.5	75.4	80.3	76.9	66.6	76.7	69.2	73.1	70.5	62.8
	Ours$^{\text{semi}}$	**77.1**	68.1	76.7	73.5	63.9	**82.4**	32.8	43.0	41.1	34.7
	Ours$^{\text{semi}}$ + p.p.	76.0	**79.7**	**84.9**	**81.4**	**69.2**	79.4	**75.0**	**82.5**	**80.2**	**68.0**

Table 8: Comparison with the state-of-the-art methods on GTEA and 50Salads.

TAD approach LSTR [44]. We train LSTR on TAS datasets using the official code implementation[2]. To ensure a fair comparison, we configure their working (short-term) memory to be the same as ours (w). Additionally, we adjust its long memory accordingly to provide access to the entire past sequence. As evident from Tables 8 and 9, LSTR [44] consistently achieves relatively low performance, particularly with Edit scores of 5.0% and 4.9% on 50Salads and Breakfast datasets, respectively. This suggests severe over-segmentation in their predictions. Moreover, these performances are inferior even to those of our baseline model (casual TCN), indicating that a direct adoption of online detection models for the segmentation task is not ideal.

Amongst all datasets, Breakfast is the most challenging, with a significant performance gap between offline and online models, particularly on segmental metrics. Notably, the F1@50 score on Breakfast experiences a drastic drop of 4/5, from 47.5% to 8.3%, highlighting the difficulty of the online segmentation task with videos that are more complex. Nonetheless, we still achieve a modest absolute performance improvement of 2%. Furthermore, our post-processing technique, significantly boosts segmental performance, nearly tripling the original performance, albeit with a slight decrease in Acc. This underscores the effectiveness of our post-processing technique in mitigating the over-segmentation. MV-TAS [13] tackles online segmentation but under a multi-view setting. It leverages multi-view information and an offline model as a reference for online segmentation. Despite this, even our baseline model, depicted in the third-to-last row of Table 9, showcases a notable performance improvement (55.3% vs. 41.6%) over MV-TAS [13]. This considerable margin emphasizes the competitiveness of our baseline model.

When compared to offline models, our semi-online inference with post-processing manages to surpass the offline model MS-TCN [11] on 50Salads dataset across the segmental metrics and reaches around 90% of the accuracy of the best-performing DiffAct [25]. On Breakfast, our approach lags behind the offline model in both frame-wise accuracy and segmental metrics.

Qualitative Result. Fig. 3 qualitatively compares the segmentation results from different approaches. It is clear to see that LSTR [44] suffers from the most prominent over-segmentation issue, which remains significant after the post-processing. Under the same configuration, our semi-online achieves slightly better results compared to the frame-by-frame online inference. On the other hand, our post-processing, when applied, successfully removes the short fragments (black boxes) in the raw

[2] `https://github.com/amazon-science/long-short-term-transformer`

	Method	Breakfast [18]				
		Acc	Edit	F1 @ {10, 25, 50}		
offline	MS-TCN [11]	69.3	67.3	64.7	59.6	47.5
	ASFormer [45]	73.5	75.0	76.0	70.6	57.4
	DiffAct [25]	75.1	76.4	80.3	75.9	75.1
online	MV-TAS [13]	41.6	-	-	-	-
	LSTR [44]	24.2	4.9	5.5	3.9	1.7
	Causal TCN	55.3	18.7	15.1	11.7	8.3
	Ours$^{\text{online}}$	56.7	19.3	16.8	13.9	9.3
	Ours$^{\text{online}}$ + p.p.	52.9	55.7	54.8	45.8	30.5
	Ours$^{\text{semi}}$	**57.4**	19.6	17.8	14.8	10.1
	Ours$^{\text{semi}}$ + p.p.	53.8	**57.5**	**56.4**	**47.3**	**31.4**

Table 9: Comparison with the state-of-the-art methods on Breakfast.

prediction and refines the segmentation output. In the meantime, it may reduce Acc, particularly at action boundaries (red boxes).

For failure cases, we have the following two observations: Action start often delays due to the need for more frame information to predict new actions, especially when facing semantic ambiguities at action boundaries. Persistent over-segmentation happens when the network makes incorrect but confident predictions, which could be improved with a stronger backbone or better temporal context modeling.

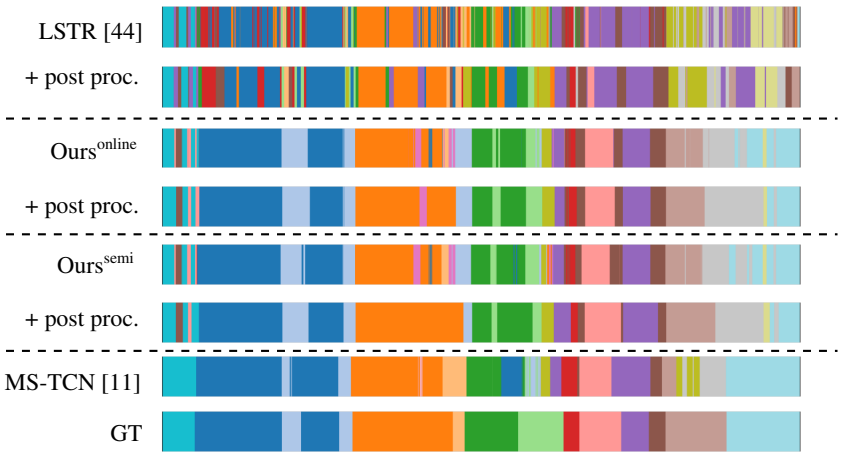


Figure 3: Visualization of segmentation outputs for sequence "rgb-01-1" from 50Salads [38].

4.3 Runtime Analysis

We evaluate the runtime performance of our approach using an Nvidia A40 GPU with both pre-computed I3D features and raw streaming RGB inputs, and present the inference times in Table 10. As shown, our approach can achieve up to 238.1 FPS when using pre-computed I3D features. To calculate the runtime for the entire segmentation pipeline, we take into consideration of the computational overheads of optical flow calculation and the I3D feature extraction. By leveraging a GPU backend for optical flow calculation, our full framework is able to achieve a runtime of 33.8 FPS.

Inference Latency. The inference speed presented above is identical for both online and semi-online inference modes since their input sizes are the same. However, the latency can differ. In the online mode, inference is performed on a per-frame basis, meaning its latency is only dependent on the inference speed. In contrast, the semi-online mode incurs additional latency as it requires gathering

frames up to the clip lengths (128 frames at a standard 25 FPS corresponds to 5.12 seconds) before forming inputs.

Online inference offers better real-time responsiveness compared to semi-online inference, but the latter achieves superior performance as we discussed in Sec. 4.1. The choice between these two modes depends on the application's priorities: if the real-time inference is critical, online inference is preferable; however, if accuracy is more important and the task is less time-sensitive, semi-online inference is recommended.

	Ours (I3D)	OF Comp.	I3D Comp.	Ours (raw)
Time (ms)	4.2	4.8	20.5	29.5
FPS	238.1	208.3	48.8	33.8

Table 10: Runtime profile (in ms and FPS).

Limitation. Handling diverse and real-world videos presents several challenges. One common scenario involves interrupted actions, where a subject abruptly switches to a different action, leaving the ongoing action unfinished. These interruptions can be challenging for the model to handle effectively. Additionally, the extended length of the video poses another challenge. Streaming videos can be infinitely long, so effectively managing and preserving long-form history within a fixed memory budget becomes a critical issue.

5 Conclusion

This paper presents the first framework for the online segmentation of actions in procedural videos. Specifically, we propose an adaptive memory bank designed to accumulate and condense temporal context, alongside a feature augmentation module capable of injecting context information into inputs and producing enhanced representations. In addition, we propose a fast and effective post-processing technique aimed at mitigating the over-segmentation problem. Extensive experiments on common benchmarks have shown the effectiveness of our approach in addressing the online segmentation task.

Acknowledgments and Disclosure of Funding

This research is supported by the National Research Foundation, Singapore under its NRF Fellowship for AI (NRF-NRFFAI1-2019-0001). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

References

[1] E. Bahrami, G. Francesca, and J. Gall. How much temporal long-term context is needed for action segmentation? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10351–10361, 2023.

[2] S. Cao, W. Luo, B. Wang, W. Zhang, and L. Ma. E2e-load: end-to-end long-form online action detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10422–10432, 2023.

[3] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[4] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[5] R. De Geest, E. Gavves, A. Ghodrati, Z. Li, C. Snoek, and T. Tuytelaars. Online action detection. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14*, pages 269–284. Springer, 2016.

[6] G. Ding, S. Fadime, and A. Yao. Temporal action segmentation: An analysis of modern techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[7] G. Ding, H. Golong, and A. Yao. Coherent temporal synthesis for incremental action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28485–28494, 2024.

[8] G. Ding and A. Yao. Leveraging action affinity and continuity for semi-supervised temporal action segmentation. In *European Conference on Computer Vision*, pages 17–32. Springer, 2022.

[9] G. Ding and A. Yao. Temporal action segmentation with high-level complex activity labels. *IEEE Transactions on Multimedia*, 25:1928–1939, 2022.

[10] Z. Du, X. Wang, G. Zhou, and Q. Wang. Fast and unsupervised action boundary detection for action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3323–3332, 2022.

[11] Y. A. Farha and J. Gall. Ms-tcn: Multi-stage temporal convolutional network for action segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3575–3584, 2019.

[12] A. Fathi, X. Ren, and J. M. Rehg. Learning to recognize objects in egocentric activities. In *CVPR 2011*, pages 3281–3288. IEEE, 2011.

[13] R. Ghoddoosian, I. Dwivedi, N. Agarwal, C. Choi, and B. Dariush. Weakly-supervised online action segmentation in multi-view instructional videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13780–13790, 2022.

[14] M. Guermal, A. Ali, R. Dai, and F. Brémond. Joadaa: joint online action detection and action anticipation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6889–6898, 2024.

[15] D.-A. Huang, Z. Yu, and A. Anandkumar. Minvis: A minimal video instance segmentation framework without video-based training. *Advances in Neural Information Processing Systems*, 35:31265–31277, 2022.

[16] H. Idrees, A. R. Zamir, Y.-G. Jiang, A. Gorban, I. Laptev, R. Sukthankar, and M. Shah. The thumos challenge on action recognition for videos "in the wild". *Computer Vision and Image Understanding*, 155:1–23, 2017.

[17] Y. Kong and Y. Fu. Human action recognition and prediction: A survey. *International Journal of Computer Vision*, 130(5):1366–1401, 2022.

[18] H. Kuehne, A. Arslan, and T. Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 780–787, 2014.

[19] S. Kumar, S. Haresh, A. Ahmed, A. Konin, M. Z. Zia, and Q.-H. Tran. Unsupervised action segmentation by joint representation learning and online clustering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20174–20185, 2022.

[20] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager. Temporal convolutional networks for action segmentation and detection. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 156–165, 2017.

[21] J. Li, P. Lei, and S. Todorovic. Weakly supervised energy-based learning for action segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6243–6251, 2019.

[22] J. Li and S. Todorovic. Action shuffle alternating learning for unsupervised action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12628–12636, 2021.

[23] S. Li, Y. A. Farha, Y. Liu, M.-M. Cheng, and J. Gall. Ms-tcn++: Multi-stage temporal convolutional network for action segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 45(6):6647–6658, 2020.

[24] S. Lin, L. Yang, I. Saleemi, and S. Sengupta. Robust high-resolution video matting with temporal guidance. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 238–247, 2022.

[25] D. Liu, Q. Li, A.-D. Dinh, T. Jiang, M. Shah, and C. Xu. Diffusion action segmentation. In *International Conference on Computer Vision (ICCV)*, 2023.

[26] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[27] D. Moltisanti, S. Fidler, and D. Damen. Action recognition from single timestamp supervision in untrimmed videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9915–9924, 2019.

[28] M. K. Myers, N. Wright, A. S. McGough, and N. Martin. O-talc: Steps towards combating oversegmentation within online action segmentation. *arXiv preprint arXiv:2404.06894*, 2024.

[29] R. Rahaman, D. Singhania, A. Thiery, and A. Yao. A generalized and robust framework for timestamp supervision in temporal action segmentation. In *European Conference on Computer Vision*, pages 279–296. Springer, 2022.

[30] A. Richard, H. Kuehne, and J. Gall. Action sets: Weakly supervised action segmentation without ordering constraints. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 5987–5996, 2018.

[31] A. Richard, H. Kuehne, A. Iqbal, and J. Gall. Neuralnetwork-viterbi: A framework for weakly supervised video learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7386–7395, 2018.

[32] S. Sarfraz, N. Murray, V. Sharma, A. Diba, L. Van Gool, and R. Stiefelhagen. Temporally-weighted hierarchical clustering for unsupervised action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11225–11234, 2021.

[33] F. Sener, D. Singhania, and A. Yao. Temporal aggregate representations for long-range video understanding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*, pages 154–171. Springer, 2020.

[34] F. Sener and A. Yao. Unsupervised learning and segmentation of complex activities from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8368–8376, 2018.

[35] Z. Shou, D. Wang, and S.-F. Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1049–1058, 2016.

[36] D. Singhania, R. Rahaman, and A. Yao. Iterative contrast-classify for semi-supervised temporal action segmentation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(2):2262–2270, Jul 2022.

[37] D. Singhania, R. Rahaman, and A. Yao. C2f-tcn: A framework for semi-and fully-supervised temporal action segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[38] S. Stein and S. J. McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 729–738, 2013.

[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[40] S. Venkatesh, D. Moffat, and E. R. Miranda. Investigating the effects of training set synthesis for audio segmentation of radio broadcast. *Electronics*, 10(7):827, 2021.

[41] J. Wang, G. Chen, Y. Huang, L. Wang, and T. Lu. Memory-and-anticipation transformer for online action understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13824–13835, 2023.

[42] X. Wang, S. Zhang, Z. Qing, Y. Shao, Z. Zuo, C. Gao, and N. Sang. Oadtr: Online action detection with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7565–7575, 2021.

[43] M. Xu, M. Gao, Y.-T. Chen, L. S. Davis, and D. J. Crandall. Temporal recurrent networks for online action detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5532–5541, 2019.

[44] M. Xu, Y. Xiong, H. Chen, X. Li, W. Xia, Z. Tu, and S. Soatto. Long short-term transformer for online action detection. *Advances in Neural Information Processing Systems*, 34:1086–1099, 2021.

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

[46] K. Ying, Q. Zhong, W. Mao, Z. Wang, H. Chen, L. Y. Wu, Y. Liu, C. Fan, Y. Zhuge, and C. Shen. Ctvis: Consistent training for online video instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 899–908, 2023.

[47] T. Zhang, X. Tian, Y. Wu, S. Ji, X. Wang, Y. Zhang, and P. Wan. Dvis: Decoupled video instance segmentation framework. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1282–1291, 2023.

[48] H. Zhao, A. Torralba, L. Torresani, and Z. Yan. Hacs: Human action clips and segments dataset for recognition and temporal localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8668–8678, 2019.

A Appendix / supplemental material

A.1 Standard vs. Causal Convolution

Standard convolution. As shown in Fig. 4 (left), the receptive field of a standard convolution includes both past and future inputs. Mathematically, for an input sequence $x(t)$ and a filter $h(k)$, the output $y(t)$ at time t is given by:

$$y(t) = \sum_{k=-K}^{K} h(k) \cdot x(t-k) \tag{8}$$

where K is the size of the filter. This means the output at time t depends on inputs from $t-K$ to $t+K$.

Causal convolution. As shown in Fig. 4 (right), the receptive field of a causal convolution includes only the past and current inputs, ensuring that the output at time t does not depend on future inputs. Mathematically, the output $y(t)$ is given by:

$$y(t) = \sum_{k=0}^{K} h(k) \cdot x(t-k) \tag{9}$$

where K is the size of the filter. This means the output at time t depends only on inputs from t to $t-K$.

A.2 CFA Formula

The attention mechanism [39] is written as:

$$\text{Attention}(Q, K, V) = \text{SoftMax}\left(\frac{Q \times K^T}{\sqrt{d}}\right) \times V \tag{10}$$

where Q, K, V represents query, key and value, respectively, and d is the hidden dimension.

We use a Transformer decoder [39] to obtain the memory encoding, and TransDecoder() is formulated as follows:

$$\tilde{M}_{k-1}^{\text{TD}} = \text{SelfAttn}(M_{k-1}, M_{k-1}, M_{k-1}) + \text{CrossAttn}(M_{k-1}, c_k^{\text{GRU}}, c_k^{\text{GRU}}) + \text{FFN} \tag{11}$$

Here, the FFN (Feed-Forward Network) is a two-layer fully connected network, M_{k-1} is the memory bank, which first undergoes self-attention. The output of the self-attention mechanism is used as the query for cross-attention, where c_k^{GRU} serves as the key and value. This interaction results in a more effective memory encoding $\tilde{M}_{k-1}^{\text{TD}}$.

We split the input feature of size $C \times T$ to 2 windows with size $C \times \frac{T}{2}$, and perform Swin [26] self attention within each local window independently, and Cross attention for every two consecutive local windows. The Swin attention mechanism can be formulated as:

$$\text{Swin Attention}(Q, K, V) = \text{SoftMax}\left(\frac{Q \times K^T}{\sqrt{d}} + B\right) \times V \tag{12}$$

Where the B is the relative position of the window. Then, our method produces context-augmented features $\tilde{c}_k$ using Eq. (3) and Eq. (4)

Our self-attention based on Eq. (12):

$$\text{SelfAttn}(c_k^{GRU}, c_k^{GRU}, c_k^{GRU}) = \text{SoftMax}\left(\frac{c_k^{GRU} \times (c_k^{GRU})^T}{\sqrt{d}} + B\right) \times c_k^{GRU} \tag{13}$$

where c as the clip features and k as the current step, c_k passes through a GRU to obtain c_k^{GRU}. Next, we use cross-attention to interact with the output of the self-attention c_k^{SA} with the output memory bank of the Transformer decoder, $\tilde{M}_{k-1}^{\text{TD}}$:

$$\text{CrossAttn}(c_k^{\text{SA}}, \tilde{M}_{k-1}^{\text{TD}}, \tilde{M}_{k-1}^{\text{TD}}) = \text{SoftMax}\left(\frac{c_k^{\text{SA}} \times (\tilde{M}_{k-1}^{\text{TD}})^T}{\sqrt{d}} + B\right) \times \tilde{M}_{k-1}^{\text{TD}} \tag{14}$$

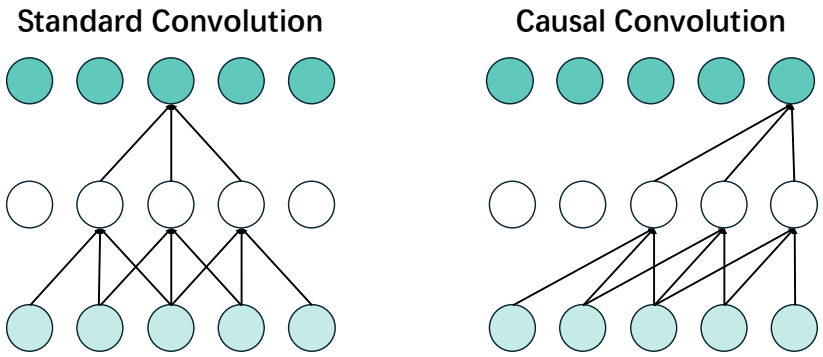


Figure 4: Standard vs. Causal Convolution

A.3 Implementation

Hyper-parameters. As shown in Table 11, the hyper-parameter settings are generally the same for each dataset. GTEA uses a shorter window size because the longest video is only about 2000 frames, whereas the other datasets all use a window size of 128. The optimal confidence threshold for segmental metrics varies for each dataset: for GTEA is 0.6; for 50Salads is 0.9; and for Breakfast is 0.8.

Hyper-parameters	**GTEA**	**50Salads**	**Breakfast**
Learning rate	$5e^4$	$5e^4$	$5e^4$
Epochs	50	50	50
No. GRU layers	1	1	1
w	64	128	128
σ	1/16	1/16	1/16
θ	0.6	0.9	0.8
iteration I	2	2	2
TD heads	8	8	8
SwinAttn heads	4	4	4
No. Causal TCN stages	1	1	1
No. Causal dilated Conv layers	10	10	10

Table 11: Hyper-parameter settings for GTEA, 50Salads, and Breakfast datasets.

A.4 AsFormer Performance

We conduct experiments on three common TAS datasets, where we replace the MS-TCN backbone with AsFormer. In MS-TCN, the transition to an online method is relatively straightforward, as it only requires replacing all the standard convolution layers with causal convolution layers. However, in AsFormer, the transformation involves more extensive modifications. In addition to replacing the convolution layers with causal convolutions, we also modify the standard attention layers into causal attention layers. Furthermore, we incorporate our proposed GRU, CFA, Memory Bank, and a Post-processing module to ensure that AsFormer transitions from an offline method to an online method. Our approach remains highly effective in boosting online segmentation performance while maintaining the strengths of the AsFormer architecture.

		GTEA					50Salads					Breakfast				
	p.p.	Acc	Edit	F1 @ {10, 25, 50}			Acc	Edit	F1 @ {10, 25, 50}			Acc	Edit	F1 @ {10, 25, 50}		
Offline	-	79.7	84.6	90.1	88.8	79.2	85.6	79.6	85.1	83.4	76.0	73.5	75.0	76.0	70.6	57.4
Online	-	75.0	69.7	77.7	74.0	62.0	77.5	29.1	37.9	35.3	28.6	64.9	32.1	31.2	27.4	20.2
	✓	72.8	77.8	84.5	80.8	64.2	69.4	36.4	70.6	65.7	52.3	63.1	60.1	61.6	54.3	39.2
Semi	-	76.5	71.3	79.0	76.7	63.1	78.5	29.7	38.5	36.2	30.4	64.8	37.0	33.9	30.0	22.6
	✓	74.7	79.6	86.3	82.8	67.0	71.0	64.9	72.2	67.2	53.8	64.0	63.2	64.9	57.5	43.0

Table 12: Performance of our approach when using ASFormer as the segmentation backbone.