MS-ADD: MLLM-SUPERVISED ASYNCHRONOUS DUAL-BRANCH DIFFUSION FOR TEMPORAL ACTION SEGMENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Weak supervision makes temporal action segmentation cheaper to annotate than frame-wise labels, yet every format of it still requires a person to watch and annotate every video. We use a multimodal large language model (MLLM) instead of a person to generate the supervision, which eliminates the human labour cost. Because the labels an MLLM generates inevitably contain errors, we select sparse timestamps and action sets, two formats of supervision that are easy for the MLLM to generate and that tolerate its errors. Neither format is perfect, so we use both: when one contains an error, the other may still provide valid supervision. The two formats differ in the computation they require, so under a fixed budget an allocation plan decides which share of the videos receives each format. Training on such supervision poses two challenges: a format may be missing from a video, and on average the timestamps of a video cover less than 1% of its frames. We propose MLLM-Supervised Asynchronous Dual-Branch Diffusion (MS-ADD). Its two-branch architecture takes the two formats of supervision as separate inputs and lets its branches exchange information. Its asynchronous diffusion assigns an independent noise level to every frame and every action, and thereby unifies pseudo-label generation, training and inference with the same forward pass. On CrossTask, Breakfast and GTEA, our model outperforms each baseline on nearly every segmental metric under the same single format of supervision, and a suitable allocation enables it to outperform the single-format state of the art at a lower budget. These results yield a general allocation plan.

1 INTRODUCTION

Recognising the action a person performs at each moment of a video is essential for assistive technologies, video surveillance and human–robot interaction (Ding et al., 2024). The videos in these applications are long and untrimmed, which has motivated a large body of work on temporal action segmentation (Abu Farha & Gall, 2019; Yi et al., 2021; Liu et al., 2023; Lu & Elhamifar, 2024). Temporal action segmentation assigns an action label to every frame, and thereby determines when each action of a procedure starts and how far the procedure has progressed. Training a segmentation model requires supervision, and its standard format, frame-wise labels, is costly: labelling every frame of a video takes longer than watching it (Ding et al., 2020). Weaker formats of supervision were introduced in response: an ordered *transcript* of the actions (Bojanowski et al., 2014; Richard et al., 2018b), an unordered *action set* naming the actions that occur (Richard et al., 2018a; Fayyaz & Gall, 2020), or *timestamps*, one labelled frame inside each segment (Li et al., 2021). Yet one timestamp per segment still requires about 26% of the annotation time of frame-wise labelling, and an action set about 21% (Ding et al., 2020). Each of these formats still requires a person to watch every video and annotate it.

The human labour cost of every format therefore grows with the number of videos. Multimodal large language models (MLLMs) are increasingly used to label videos (Zhang et al., 2025; Hyun et al., 2025; Chen et al., 2026). We use an MLLM instead of a person to generate the supervision, which eliminates the human labour cost (Figure 1).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

The labels an MLLM generates inevitably contain errors. We address these errors in two ways. First, we select only formats of supervision that are easy for the MLLM to generate and that tolerate its errors. A format is easy for the MLLM to generate when the MLLM can omit the labels it is uncertain about without making the remaining supervision wrong. A format tolerates errors when an error remains local rather than corrupting other labels. To choose such supervision, we examine the common formats against the two properties. Transcript supervision has neither property: omitting an uncertain action, or including a wrong one, shifts every subsequent entry. Sparse timestamp supervision, unlike standard timestamp supervision, does not require a timestamp in every segment. It has both properties: the MLLM can omit the timestamps it is uncertain about, and each timestamp is independent of the others. Action-set supervision also has both properties: the MLLM can omit the actions it is uncertain about, and a missed action leaves the set incomplete rather than wrong, so the model loses one positive label but is not trained towards a wrong action. Second, whichever supervision is chosen, its quality is never perfect. Because the MLLM generates the two formats separately, when one of them contains an error, the other may still be correct and provide valid supervision for training the model. We therefore use both sparse timestamp and action-set supervision rather than only one of them.

Using an MLLM eliminates the human labour cost, but generating the supervision still consumes computation, and the two formats differ in the computation they require. We take a step further and aim to use a limited computational budget as efficiently as possible, that is, to find the best allocation plan. An allocation specifies the share of the videos that receives each format. Training on such supervision poses two challenges. First, the videos for each format are drawn at random, so a format may be missing: a video receives both formats, timestamps only, an action set only, or neither (Figure 1), and the model must be trained on all four cases jointly. Second, the supervision of a video is incomplete: on average, its timestamps cover less than 1% of its frames.

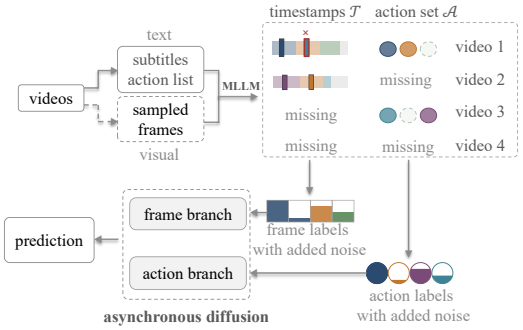


Figure 1: The pipeline, on four videos. The MLLM receives sampled frames only when it generates timestamps, and returns only the timestamps it is confident about. Each format is assigned to a random share of the videos, so the four videos have both formats, timestamps only, an action set only, and neither. Colour denotes the action and grey denotes background. Under $\mathcal{T}$, a pale block shows the ground truth, which is never observed, and a narrow bar shows a timestamp returned by the MLLM; the one circled in red is labelled with the wrong action. Under $\mathcal{A}$, a filled circle is an action in the set, and a dashed circle is an action that occurs but is missing from the set. Each format then enters its own branch, and the fill of a mark indicates how well it is known.

Contributions. To address these two challenges, we propose MLLM-Supervised Asynchronous Dual-Branch Diffusion (MS-ADD), whose model, ADD, combines a two-branch architecture with asynchronous diffusion. The *two-branch architecture* takes the two formats of supervision as separate inputs and lets its branches interact throughout processing, so that the formats do not interfere with each other and yet information flows between them. *Asynchronous diffusion* assigns an independent noise level to every frame and every action, and thereby unifies pseudo-label generation for the frames without a timestamp, training and inference with the same forward pass.

Two-branch architecture. Following FACT (Lu & Elhamifar, 2024), a frame branch processes the frame features and the timestamp supervision, and an action branch processes the action-set supervision. Processing the two formats in separate branches ensures that a missing format never enters the input of the other branch. The two branches exchange information repeatedly, so that each draws on the other (§3.2.2).

Asynchronous diffusion. The two-branch architecture structurally accommodates a missing format of supervision, but it cannot recover the information that the missing format would have provided. In addition, the sparse timestamps of a video cover almost none of its frames. We therefore

use the model to assign pseudo-labels to the frames without a timestamp and train it on them. The quality of these pseudo-labels determines how well the model is trained, and depends largely on how the available supervision is exploited. A common way to refine pseudo-labels is post-processing: the model first predicts from the video alone, and the supervision then corrects the prediction, for instance by propagating each timestamp's label to neighbouring frames (Li et al., 2021) or by removing the actions that are not in the action set (Li & Todorovic, 2020). Post-processing can correct a prediction after it is made, but cannot guide how it is made.

We instead provide the supervision to the model as input, so that it conditions the prediction. Diffusion enables this. DiffAct (Liu et al., 2023) generates the frame labels by denoising: its input consists of the frame features and the labels corrupted with noise, and it learns to recover the labels. The noise level of a label determines how heavily it is corrupted. DiffAct, however, corrupts all labels with the same noise level, so it cannot keep the known labels at a lower noise level to condition the denoising of the unknown ones. Our model must accept *any subset* of the frames and actions as a condition, since a timestamp covers a single frame, an action set covers the whole video, and a video may have both, one or neither.

Asynchronous diffusion meets this requirement by assigning an independent noise level to every frame and every action, motivated by diffusion forcing (Chen et al., 2024a). During pseudo-label generation, the known labels are kept at a low noise level and condition the denoising of the unknown ones. Independent noise levels also benefit training on the pseudo-labels, which are the model's own estimates and may be wrong. During training, the noise level of each pseudo-label is drawn from a distribution that shifts towards higher noise levels as the model's confidence in the pseudo-label decreases. A pseudo-label with low confidence is therefore less likely to be presented at a low noise level, which discourages the model from conditioning on it (§3.2.3). At inference, an *asynchronous schedule* denoises the easier frames and actions first and uses them to condition the rest (§S1). Pseudo-label generation, training and inference thus share the same forward pass and differ only in how the noise levels are set.

We evaluate ADD on CrossTask, Breakfast and GTEA. Not only does it outperform each baseline on nearly every segmental metric under the same single format of supervision, which shows the effectiveness of the model itself, but a suitable allocation also enables it to outperform the single-format state of the art at a lower budget, which shows the value of the approach as a whole (§4.2).

2 Related Work

Weak supervision for segmentation. Transcript supervision reduces segmentation to alignment (Bojanowski et al., 2014; Richard et al., 2018b; Li et al., 2019), but a single missing or extra entry shifts every subsequent entry. Discarding the order yields action-set supervision (Richard et al., 2018a; Fayyaz & Gall, 2020; Li & Todorovic, 2020; Lu & Elhamifar, 2022). Timestamp supervision (Li et al., 2021; Behrmann et al., 2022; Rahaman et al., 2022; Souri et al., 2022a; Du et al., 2023) places a timestamp in every segment. Some of these works relax this requirement (Rahaman et al., 2022; Souri et al., 2022a), but the reported settings still place a timestamp in most segments, down to 70% (Souri et al., 2022a). The timestamps generated by the MLLM, however, cover far fewer than 70% of the segments before their accuracy drops (§4.3). Formats of supervision have also been combined, with frame-wise labels on some videos and transcripts on the rest (Souri et al., 2022b), and compared for image segmentation under a fixed annotation budget (Bearman et al., 2016), but only for human-annotated formats and without allocating a budget between formats of weak supervision.

MLLM-generated supervision. MLLMs are increasingly used to generate labels for video understanding (Zhang et al., 2025; Hyun et al., 2025; Chen et al., 2026) and for zero-shot action segmentation (Unmesh et al., 2026). Closest to our work, Chen et al. (2024b) and Li et al. (2024) use an LLM to convert narration into procedure steps, align the steps temporally, and train on the alignments that exceed a confidence threshold as pseudo-labels. We instead study which formats of weak supervision an MLLM can generate reliably, and at what cost. To our knowledge, no prior work on action segmentation generates timestamp or action-set supervision with an MLLM, measures its computational cost, or allocates a budget between formats of weak supervision.

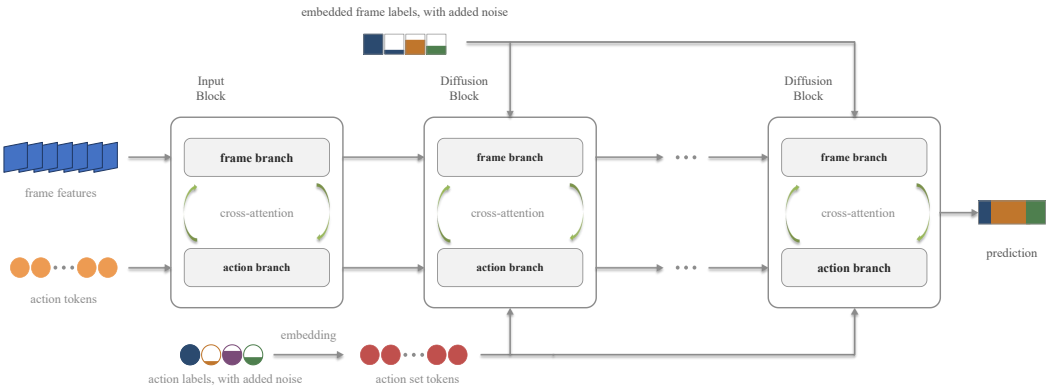


Figure 2: The model. Frame features and M learnable action tokens enter an input block. Every diffusion block receives the corrupted labels: frame labels are embedded and added to the frame features, and the action set is embedded into action-set tokens that are concatenated with the action tokens. The fill of a mark indicates how well it is known.

3 METHODOLOGY

Given a video with T frames and pre-extracted features $\mathbf{x} = (x_1, \dots, x_T) \in \mathbb{R}^{T \times D}$, our goal is to predict its frame-wise labels $\mathbf{y}^\star \in \mathcal{C}^T$ over the set of actions $\mathcal{C} = \{1, \dots, K\}$. The labels $\mathbf{y}^\star$ are used only for evaluation.

Because no supervision is available, we generate it automatically with an MLLM (§3.1) and propose a model trained on the generated supervision (§3.2).

3.1 GENERATING THE SUPERVISION

The two formats of supervision require different inputs. The input for the **action set** is text alone: the complete subtitle track and the set of actions the video's task may contain, denoted as $\mathcal{C}_{\text{task}} \subset \mathcal{C}$. The dataset defines $\mathcal{C}_{\text{task}}$ once per task, so it does not indicate which of these actions a given video contains. We ask the MLLM for the actions most likely performed, so an action it is uncertain about is omitted rather than guessed. The MLLM returns an action set $\mathcal{A} \subseteq \mathcal{C}_{\text{task}}$, which lists the actions it identifies in the video.

A timestamp specifies the action at a single frame, so the input for the **timestamps** consists of the same text and frames sampled at uniform intervals, one per segment on average. We ask the MLLM to label only the share η of these frames it is most confident about. The MLLM returns the timestamps $\mathcal{T} = \{(t_j, c_j)\}_{j=1}^{n_\tau}$, which assign an action c_j to each of $n_\tau \ll T$ frames t_j.

Both prompts are given in §S3, and Table 2 reports the quality of both formats on CrossTask.

The computational cost of a training set depends on how many videos receive each format. We measure computation in tokens: let $\text{cost}(\mathcal{T})$ and $\text{cost}(\mathcal{A})$ denote the numbers of tokens the MLLM consumes to generate the timestamps and the action set of one video, and let ρ_τ and ρ_α denote the percentages of videos for which the MLLM generates each format. The videos that receive timestamps and those that receive an action set are drawn at random and independently; a video without timestamps has $\mathcal{T} = \emptyset$, and one without an action set $\mathcal{A} = \emptyset$. The budget B is the cost of a training set as a percentage of its cost with both formats on every video,

$$r = \frac{\text{cost}(\mathcal{T})}{\text{cost}(\mathcal{A})}, \qquad B(\rho_\tau, \rho_\alpha) = \frac{r\,\rho_\tau + \rho_\alpha}{r+1}. \tag{1}$$

B is 100 with both formats on every video, $100r/(r+1)$ with timestamps on every video and no action set, and $100/(r+1)$ with an action set on every video and no timestamps. The values of η and r are given in §4.1.

3.2 THE PROPOSED MODEL

Training on the generated supervision poses two challenges: a format of supervision may be missing from a video, and the supervision of a video is incomplete. The model (Figure 2) addresses both by combining a two-branch architecture, which takes the two formats of supervision as separate inputs (§3.2.2), with asynchronous diffusion, which assigns an independent noise level to every frame and every action (§3.2.1), so that the available supervision enters as a condition when pseudo-labels are generated (§3.2.3).

3.2.1 INDEPENDENT NOISE LEVELS

Each frame t has an independent noise level $\ell_t \in \{0, \dots, N\}$, where N is the maximum noise level (Figure 3a). Let y_t denote the label of the frame, either a timestamp or a pseudo-label, $e(y_t)$ its one-hot encoding over the K actions, $\epsilon_t \sim \mathcal{N}(0, I)$ Gaussian noise of the same dimension, and $\bar{\alpha}_0, \dots, \bar{\alpha}_N$ a fixed schedule decreasing from $\bar{\alpha}_0 = 1$ to $\bar{\alpha}_N \approx 0$. As in standard forward diffusion (Ho et al., 2020), the model receives the label corrupted by noise,

$$\tilde{y}_t = \sqrt{\bar{\alpha}_{\ell_t}}\, e(y_t) + \sqrt{1 - \bar{\alpha}_{\ell_t}}\, \epsilon_t, \qquad (2)$$

so the higher ℓ_t, the less of the label is preserved: at $\ell_t = 0$ the model receives the label itself, and at $\ell_t = N$ essentially pure noise, which is the input for an unlabelled frame. The noise level is provided alongside $\tilde{y}_t$, and the features are never corrupted.

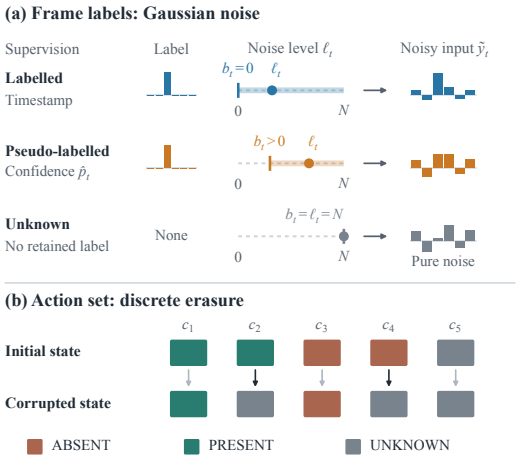


Figure 3: **(a)** A frame's base noise level (§3.2.3) indicates how well its label is known, and its noise level is drawn upwards from it. **(b)** Each action is independently erased to UNKNOWN.

Each action also has an independent noise level, and its supervision is the action set. The set $\mathcal{A}$ provides positives, as the actions it contains are assumed to occur, but a model trained only on positives can satisfy them by predicting every action as present. We draw negatives from $\mathcal{C} \setminus \mathcal{C}_{\text{task}}$. Every action the video's task may contain lies in $\mathcal{C}_{\text{task}}$, so an action outside $\mathcal{C}_{\text{task}}$ cannot occur in the video, and every negative is certain. Each action k then has a state z_k: PRESENT if $\mathcal{A}$ contains k, ABSENT if k lies outside $\mathcal{C}_{\text{task}}$, and UNKNOWN otherwise. Unlike a label, a discrete state cannot be blurred, so corruption erases it, as in discrete diffusion with an absorbing state (Austin et al., 2021): independently for each action, the state $\tilde{z}_k$ the model receives is UNKNOWN with probability γ_k and z_k otherwise,

$$\tilde{z}_k = \begin{cases} z_k, & \text{with probability } 1 - \gamma_k, \\ \text{UNKNOWN}, & \text{with probability } \gamma_k, \end{cases} \qquad (3)$$

which makes γ_k the noise level of action k: at $\gamma_k = 0$ the model receives what is known about the action, and at $\gamma_k = 1$ it receives nothing. An action in $\mathcal{C}_{\text{task}}$ that $\mathcal{A}$ does not contain is already UNKNOWN, just as an unlabelled frame is already at $\ell_t = N$ (Figure 3b).

3.2.2 MODEL ARCHITECTURE

To separate the two formats of supervision, we design the model with *two branches* following FACT (Lu & Elhamifar, 2024): a *frame branch*, which processes the frame features and the timestamp supervision, and an *action branch*, which processes learnable action tokens and the action-set supervision. The two branches exchange information only through cross-attention.

Input block. The input block feeds the pre-extracted frame features $\mathbf{x}$ into the frame branch and initialises M learnable *action tokens* $a_1, \dots, a_M$ in the action branch. An action token is not assigned to any frame or action in advance; it gradually encodes the action of one segment and its approximate location, and is refined from block to block. We index the blocks in order by i, with

$i = 0$ for the input block and $i = 1, 2, \ldots$ for the diffusion blocks that follow it. Let $\mathbf{f}^{(i)}$ and $\mathbf{a}^{(i)}$ denote the frame features and the action tokens that block i outputs and passes to the next block, so that $(\mathbf{f}^{(0)}, \mathbf{a}^{(0)}) = \mathrm{InputBlock}(\mathbf{x}, [a_1, \ldots, a_M])$.

Diffusion block. Each diffusion block i takes the corrupted labels as input and denoises them, conditioned on $\mathbf{f}^{(i-1)}$ and $\mathbf{a}^{(i-1)}$ from the preceding block. A frame label and a frame feature share the same index, the frame, so they can be summed: with $\tilde{\mathbf{y}}$ and $\boldsymbol{\ell}$ the corrupted labels and their noise levels at all T frames, and emb the layer that maps one frame's label and noise level to a vector of the frame-feature dimension, block i computes $\mathbf{f}^{(i)} = \mathbf{f}^{(i-1)} + \mathrm{emb}(\tilde{\mathbf{y}}, \boldsymbol{\ell})$. The action set and the action tokens share no index, so we convert the corrupted action set into K *action-set tokens*, one per action, $s_k = E_{\mathrm{act}}(k) + E_{\mathrm{state}}(\tilde{z}_k)$, where the learned embedding E_{act} encodes which action the token represents and E_{state} which of the three states is currently known for it. Block i concatenates them with the action tokens into one action stream $\mathbf{u}^{(i)} = [\mathbf{a}^{(i-1)}, s_1, \ldots, s_K]$ of $M + K$ tokens.

The two branches then exchange information through three attention layers: the frames attend to the action stream, a self-attention over $\mathbf{u}^{(i)}$ lets the action tokens and the action-set tokens attend to each other, and the action stream attends back to the frames. We write the three layers as one update of both branches, $\mathbf{f}^{(i)}, \mathbf{u}^{(i)} \leftarrow \text{Cross-Attention}(\mathbf{f}^{(i)}, \mathbf{u}^{(i)})$. Each branch then denoises its own supervision: a *frame head* on $\mathbf{f}^{(i)}$ outputs the frame prediction $\hat{\mathbf{y}}^{(i)}$ of the block, one action per frame, and an *existence head* on the K action-set tokens of $\mathbf{u}^{(i)}$ outputs the action-set prediction $\hat{\mathbf{z}}^{(i)}$, which indicates whether each action occurs. The next block receives $\mathbf{f}^{(i)}$ and the first M tokens of $\mathbf{u}^{(i)}$, the updated action tokens $\mathbf{a}^{(i)}$.

Matching and loss. Each action token must be matched to a part of the video before it can be supervised. Our videos have no ground-truth segments, unlike those FACT matches its tokens to, so we partition the frames into fixed-length chunks and greedily assign each chunk to one token, using the timestamps, the action set, the cross-attention and the predictions of both branches, and supervise each token with the action of its chunks (§S4).

The training objective sums, over all blocks including the input block, a frame loss $\mathcal{L}_{\mathrm{f}}$ on the timestamps and the pseudo-labels, which weights each pseudo-labelled frame by its confidence (§3.2.3), and four terms on the action tokens, the action-set prediction, the cross-attention and smoothness (§S5). Each block thus learns to refine the output of the preceding one.

3.2.3 Model training

The model is trained on the supervision generated by the MLLM, which is highly incomplete. Early in training, the model cannot yet produce reliable pseudo-labels, so during a warm-up phase it is trained on the MLLM's supervision alone. After the warm-up, asynchronous diffusion is used both to generate pseudo-labels and to train on them.

Pseudo-label generation. Independent noise levels allow the supervision to enter as a condition. Generation initialises each frame and each action with what is known about it — noise level 0 at the timestamped frames and N elsewhere; PRESENT for the actions in $\mathcal{A}$, ABSENT for those outside $\mathcal{C}_{\mathrm{task}}$ and UNKNOWN for the rest, or UNKNOWN for every action when the video has no action set — and re-imposes the timestamps at every step, as in RePaint (Lugmayr et al., 2022), so that the frames around them are generated conditioned on them. Algorithm S1 denoises the frames without a timestamp from pure noise to clean labels in R steps, and a frame retains its pseudo-label only if the model is confident about it.

Uncertainty-aware noise sampling. Independent noise levels also benefit training on the pseudo-labels. Pseudo-labels cover far more frames than the timestamps, and some are wrong. A wrong pseudo-label misleads the model in two places: in the loss, where the model is fitted to it, and in the input, where it conditions the prediction at every other frame. When presented nearly clean, it is relied upon as if it were a timestamp. We address both through one quantity per frame, its base noise level.

For each pseudo-label, generation records its confidence $\hat{p}_t \in [0, 1]$, the probability that the frame head assigns to the predicted action. The confidence determines the frame's *base noise level* b_t, the

lowest noise level at which the frame is presented to the model. A constant $\kappa \in (0, 1]$ bounds it from above, so the least confident pseudo-label is placed just below κN and the most confident just above 0,

$$b_t \;=\; \min\big(\max\big((1-\hat{p}_t)\,\kappa N,\; 1\big),\; \kappa N - 1\big), \tag{4}$$

rounded to an integer. During training, a frame's noise level is drawn upwards from its base noise level, $\ell_t \;=\; \min(b_t + \delta_t,\; N)$, with $\delta_t \sim \mathcal{U}\{0, \ldots, N\}$ resampled per frame at every iteration. A low-confidence pseudo-label thus has a high b_t and rarely conditions other frames as a clean label would, whereas a high-confidence one often does; a timestamp has $b_t = 0$ and an unlabelled frame $b_t = N$ (Figure 3a).

The base noise level also determines w_t, the weight of the frame in $\mathcal{L}_{\mathrm{f}}$, which decreases linearly from 1 at $b_t = 0$ to 0 at $b_t = \kappa N$ and is clipped at 0 beyond,

$$w_t \;=\; \max\big(1 - b_t/(\kappa N),\; 0\big), \tag{5}$$

so a timestamp contributes fully, a pseudo-label in proportion to its confidence, and an unlabelled frame not at all.

4 Experiments

4.1 Experimental setup

Datasets and metrics. We use CrossTask (Zhukov et al., 2019), Breakfast (Kuehne et al., 2014) and GTEA (Fathi et al., 2011). CrossTask provides a complete subtitle track for every video and $\mathcal{C}_{\mathrm{task}}$ for every task, and its supervision is generated by the MLLM (§3.1), including its errors. On Breakfast and GTEA, we derive the timestamps and the action set from the ground truth, with timestamps on the same share η of the sampled frames as on CrossTask, so the supervision is equally sparse but free of label errors. The model is thus tested both with and without label errors. Following Abu Farha & Gall (2019), we report frame accuracy (Acc, equivalently MoF), segmental F1 at IoU $\{0.10, 0.25, 0.50\}$, and segmental Edit.

Implementation. The MLLM, GPT-5.5 (OpenAI, 2026), labels the share $\eta = 30\%$ of the sampled frames it is most confident about (§3.1). Following the price ratio of the MLLM, we count each output token, including reasoning tokens, as eight input tokens. The timestamps then cost $28{,}228$ tokens per video and the action set $5{,}817$, which gives $r = 4.9$ in equation 1. The model consists of one input block followed by three diffusion blocks, with a maximum noise level $N = 1000$ with a cosine schedule (Nichol & Dhariwal, 2021) and an upper bound $\kappa = 0.8$ on the base noise level. During training, γ_k in equation 3 is $1/3$ for an action in the action set and $2/3$ for an action outside $\mathcal{C}_{\mathrm{task}}$. All results are obtained with the asynchronous schedule at inference (§S1). The denoising steps, confidence thresholds, features, optimiser and training schedule are given in §S6.

4.2 Main results

For action-set supervision, we compare with POC (Lu & Elhamifar, 2022), the current state of the art. For timestamp supervision, we compare with the state of the art among the methods that do not require a timestamp in every segment, namely TS-SEG (Li et al., 2021), run without its label expansion, which assumes one timestamp per segment, and EM-GEN (Rahaman et al., 2022), because other methods (Behrmann et al., 2022; Du et al., 2023) rely on a timestamp in every segment for training.

Allocation plan. We compare the allocations and the baselines in terms of F1 at IoU 0.1. On every dataset, an allocation that combines the two formats of supervision outperforms, at a lower budget, the state of the art under a single format (the filled diamonds in Figure 4). At the smallest budget on GTEA, timestamps cannot supervise the model effectively: GTEA has only 21 short training videos, so the timestamps cover very few frames of the training set. We treat this case as an exception. Excluding this exception, Figure 4 shows that the best allocation of every budget spends between 70% and 90% of it on timestamps (the shaded strip), and that spending the entire budget on timestamps is never optimal. Our allocation plan is therefore to spend about 80% of a budget on timestamps.

Table 1: Comparison with the state of the art. ρ_τ/ρ_α is the percentage of videos receiving each format of supervision, and Cost is B in equation 1. Our rows between the two single-format allocations use $\rho_\tau = \rho_\alpha$ rather than the best allocation at each budget (Figure 4). Shaded bands group rows with the same supervision. Bold marks the best result in each column.

			CrossTask (MLLM)					**Breakfast** (ground truth)					**GTEA** (ground truth)				
Method	ρ_τ/ρ_α	**Cost**	F1@{10, 25, 50}			Edit	Acc	F1@{10, 25, 50}			Edit	Acc	F1@{10, 25, 50}			Edit	Acc
POC	0/100	17	6.9	4.0	1.4	13.8	45.9	14.4	10.1	4.8	27.6	23.9	16.0	7.7	5.5	30.1	20.0
Ours	0/100	17	9.4	3.8	2.1	19.3	39.2	16.5	11.7	5.8	29.4	22.8	22.5	11.3	6.3	31.2	13.0
Ours	20/20	20	29.1	19.7	8.7	34.7	62.7	23.8	19.4	10.6	34.7	29.5	13.4	8.3	2.5	10.6	14.2
Ours	40/40	40	34.6	25.1	11.4	38.5	57.4	25.1	20.2	12.2	36.9	37.3	21.5	17.2	8.6	23.3	28.7
Ours	60/60	60	37.6	27.0	13.2	**42.7**	59.6	26.0	22.6	13.3	38.3	47.0	39.9	31.8	14.9	40.6	37.8
TS-SEG	100/0	83	33.6	28.1	14.2	30.2	**74.3**	39.9	34.0	22.7	41.0	53.7	27.0	21.4	13.9	40.6	29.5
EM-GEN	100/0	83	32.8	29.0	13.5	31.7	73.4	41.5	36.2	23.2	41.2	51.9	50.8	36.0	16.9	41.7	33.0
Ours	100/0	83	36.4	29.1	**15.7**	36.7	72.3	42.2	38.3	24.1	40.9	50.7	47.9	41.4	20.7	47.4	34.6
Ours	100/100	100	**39.4**	**30.2**	14.9	40.0	59.3	**44.2**	**39.3**	**28.6**	**47.4**	**58.6**	**57.6**	**53.5**	**28.0**	**49.5**	**47.0**

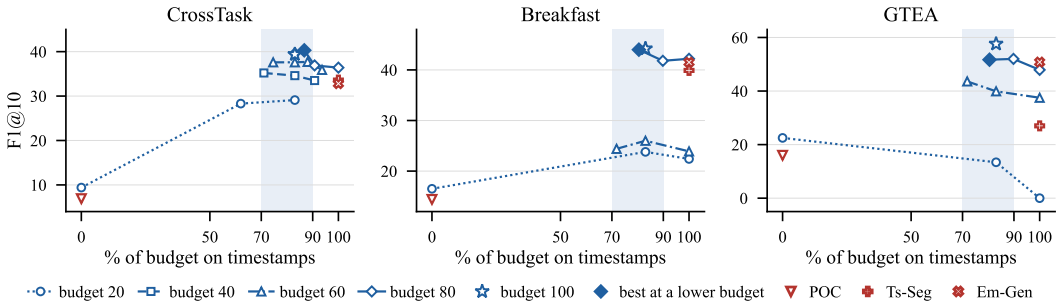


Figure 4: Best allocation of a budget. The x axis is the share of the budget spent on timestamps, $r\rho_\tau/(r\rho_\tau + \rho_\alpha)$, and each curve joins allocations of the same budget. The shaded strip marks the 70–90% band. A filled diamond marks the best allocation that costs less than TS-SEG and EM-GEN.

Our model outperforms the baselines under identical supervision. With supervision identical to that of each baseline, our model outperforms all three baselines on nearly every segmental metric on all three datasets (Table 1). We base this comparison on the segmental metrics, since over-segmentation has little effect on frame accuracy (Abu Farha & Gall, 2019). The gain therefore stems from the model itself, not from access to two formats of supervision.

Our model benefits from a larger budget. With both formats of supervision on every video (the last row of Table 1), our model improves on its results with either format alone in most segmental metrics, whereas a model designed for one format cannot use the other. Although the last row is compared with the baselines at a larger budget, it shows the potential of our model when the budget allows more supervision.

Table 2: Quality of the CrossTask supervision against the ground truth (%). Columns give η. Coverage is the share of segments that contain a timestamp, and accuracy is the share of timestamps with the correct label. The action set does not depend on η.

η	20	30	40
timestamps			
action coverage	12.9	27.4	36.8
background coverage	23.1	33.7	42.3
action accuracy	75	74	55
background accuracy	73	73	71
overall accuracy	73	74	63
action set			
precision		90.7	
recall		74.8	
F1		81.9	

4.3 ABLATIONS AND ANALYSIS

Timestamp coverage and accuracy. We vary η, the share of the sampled frames given to the MLLM that it labels as timestamps, and measure the quality of the returned timestamps (Table 2). Considering both coverage and accuracy, $\eta = 30\%$ is the best choice: below it, the timestamps cover far fewer segments, and above it, their accuracy drops markedly.

Table 3: Pseudo-label quality on CrossTask at $\rho_\tau/\rho_\alpha = 80/60$ at two checkpoints (%). Coverage is the share of frames with a pseudo-label.

	ours		post-processing	
	45k	90k	45k	90k
coverage	30.0	48.1	22.5	46.9
precision	79.4	75.4	76.2	68.3
recall	23.8	36.2	17.1	32.0
F1	36.6	48.9	27.9	43.6

Table 4: Ablation of the base noise level on CrossTask: the noise level of every pseudo-label is drawn as for a timestamp.

	ours		ablated	
	F1@10	Edit	F1@10	Edit
$\rho_\tau/\rho_\alpha = 20/20$	29.1	34.7	26.3	29.8
$\rho_\tau/\rho_\alpha = 40/40$	34.6	38.5	30.0	36.6
$\rho_\tau/\rho_\alpha = 80/80$	39.6	41.1	38.2	38.4



Figure 5: Erroneous supervision on the CrossTask training video `76400_DRyTaSflfqo` (*Make French Toast*) at $\rho_\tau/\rho_\alpha = 80/60$. Dashed boxes mark where our prediction departs from the supervision and agrees with the ground truth.

Pseudo-label generation. We compare two ways of obtaining pseudo-labels: generation with asynchronous diffusion (§3.2.3) and post-processing, which propagates each timestamp's label to neighbouring frames (Li et al., 2021) and removes the actions that are not in the action set (Li & Todorovic, 2020). At both checkpoints, asynchronous diffusion yields higher coverage, precision, recall and F1 (Table 3), which shows that our pseudo-label generation is effective.

Uncertainty-aware noise sampling. The base noise level ties the noise level of each pseudo-label to the model's confidence in it (§3.2.3). We ablate this design by drawing the noise level of every pseudo-label uniformly across the whole range, as for a timestamp, while its loss weight w_t in equation 5 is still set by the base noise level, as in the full model. Both F1 at IoU 0.1 and Edit decrease at every allocation tested (Table 4), so uncertainty-aware noise sampling is effective.

Qualitative results. We examine whether the model corrects the errors in the supervision generated by the MLLM, which are never marked as wrong during training. On the training video in Figure 5, the model corrects some of the wrong timestamps and part of the wrong action set, which illustrates that our design has the intended effect.

5 Conclusion

We use an MLLM instead of a person to generate the supervision for temporal action segmentation, which eliminates the human labour cost. Because the labels an MLLM generates inevitably contain errors, we request only sparse timestamps and action sets, two formats of supervision that are easy for the MLLM to generate and that tolerate its errors. Neither format is perfect, so we use both: when one contains an error, the other may still provide valid supervision. The two formats differ in the computation they require, so under a fixed budget an allocation plan decides which share of the videos receives each format. Training on such supervision poses two challenges: a format may be missing from a video, and on average the timestamps of a video cover less than 1% of its frames. MS-ADD addresses both. Its two-branch architecture takes the two formats of supervision as separate inputs, and its asynchronous diffusion assigns an independent noise level to every frame and every action. The available supervision thus conditions pseudo-label generation, which shares the same forward pass with training and inference. Under the same single format of supervision, our model outperforms each baseline on nearly every segmental metric, and a suitable allocation enables it to outperform the single-format state of the art at a lower budget. These results yield a general allocation plan. Modelling the errors in the MLLM's labels explicitly is a natural next step.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods, to give feedback on the research methodology and the experiments, to propose and refine hypotheses, to convert the data sets and the supervision into the input formats of the baselines, to help interpret results, and to translate the authors' notes into English. We also used an MLLM, GPT-5.5, to generate synthetic data, namely the training supervision on CrossTask; generating the supervision with an MLLM is the setting this work studies (§3.1). From the subtitle track of a training video and the set of actions its task may contain, GPT-5.5 generates an action set; from the same text and frames sampled from the video, it generates timestamps on the share of these frames it is most confident about. GPT-5.5 generates no other data: the videos and the ground-truth annotations used for evaluation are those released with each dataset, and the supervision on Breakfast and GTEA is derived from their ground-truth annotations. We have not used generative AI tools to develop theoretical models or conceptual frameworks, to formulate mathematical claims, to prove them, or to support qualitative and thematic data analysis. Additionally, we used generative AI tools to create and edit software code, to draft and edit parts of the paper, to create some of the figures, to identify relevant literature, to format references, and to suggest the structure of the paper. We have reviewed all AI-assisted work: one of the authors reviewed and tested the code behind every reported result, checked every number against the experiment logs, checked every citation against the cited work, and revised all AI-assisted text and figures. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Yazan Abu Farha and Juergen Gall. MS-TCN: Multi-stage temporal convolutional network for action segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3575–3584, 2019.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Amy Bearman, Olga Russakovsky, Vittorio Ferrari, and Li Fei-Fei. What's the point: Semantic segmentation with point supervision. In *European Conference on Computer Vision (ECCV)*, pp. 549–565, 2016.

Nadine Behrmann, S. Alireza Golestaneh, Zico Kolter, Juergen Gall, and Mehdi Noroozi. Unified fully and timestamp supervised temporal action segmentation via sequence to sequence translation. In *European Conference on Computer Vision (ECCV)*, 2022.

Piotr Bojanowski, Rémi Lajugie, Francis Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, and Josef Sivic. Weakly supervised action labeling in videos under ordering constraints. In *European Conference on Computer Vision (ECCV)*, 2014.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. MaskGIT: Masked generative image transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11315–11325, 2022.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a.

Delong Chen, Tejaswi Kasarla, Yejin Bang, Mustafa Shukor, Willy Chung, Jade Yu, Allen Bolourchi, Theo Moutakanni, and Pascale Fung. Action100M: A large-scale video action dataset. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2026.

Yuxiao Chen, Kai Li, Wentao Bao, Deep Patel, Yu Kong, Martin Renqiang Min, and Dimitris N. Metaxas. Learning to localize actions in instructional videos with LLM-based multi-pathway text-video alignment. In *European Conference on Computer Vision (ECCV)*, 2024b.

Guodong Ding, Fadime Sener, and Angela Yao. Temporal action segmentation: An analysis of modern techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2024.

Xinpeng Ding, Nannan Wang, Xinbo Gao, Jie Li, Xiaoyu Wang, and Tongliang Liu. Weakly supervised temporal action localization with segment-level labels. *arXiv preprint arXiv:2007.01598*, 2020. A shortened version appears in PRCV 2021, LNCS 13019, pp. 42–54.

Dazhao Du, Enhan Li, Lingyu Si, Fanjiang Xu, and Fuchun Sun. Timestamp-supervised action segmentation from the perspective of clustering. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 690–698, 2023.

Alireza Fathi, Xiaofeng Ren, and James M. Rehg. Learning to recognize objects in egocentric activities. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3281–3288, 2011.

Mohsen Fayyaz and Juergen Gall. SCT: Set constrained temporal transformer for set supervised action segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 498–507, 2020.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Jeongseok Hyun, Su Ho Han, Hyolim Kang, Joon-Young Lee, and Seon Joo Kim. Exploring scalability of self-training for open-vocabulary temporal action localization. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2025.

Hilde Kuehne, Ali Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 780–787, 2014.

Jun Li and Sinisa Todorovic. Set-constrained viterbi for set-supervised action segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Jun Li, Peng Lei, and Sinisa Todorovic. Weakly supervised energy-based learning for action segmentation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6242–6250, 2019.

Zeqian Li, Qirui Chen, Tengda Han, Ya Zhang, Yanfeng Wang, and Weidi Xie. Multi-sentence grounding for long-term instructional video. In *European Conference on Computer Vision (ECCV)*, pp. 200–216, 2024.

Zhe Li, Yazan Abu Farha, and Juergen Gall. Temporal action segmentation from timestamp supervision. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Daochang Liu, Qiyue Li, AnhDung Dinh, Tingting Jiang, Mubarak Shah, and Chang Xu. Diffusion action segmentation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Zijia Lu and Ehsan Elhamifar. Set-supervised action learning in procedural task videos via pairwise order consistency. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19871–19881, 2022.

Zijia Lu and Ehsan Elhamifar. FACT: Frame-action cross-attention temporal modeling for efficient action segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18175–18185, 2024.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. RePaint: Inpainting using denoising diffusion probabilistic models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11461–11471, 2022.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8162–8171, 2021.

OpenAI. GPT-5.5, 2026.

Rahul Rahaman, Dipika Singhania, Alexandre Thiery, and Angela Yao. A generalized and robust framework for timestamp supervision in temporal action segmentation. In *European Conference on Computer Vision (ECCV)*, pp. 279–296, 2022.

Alexander Richard, Hilde Kuehne, and Juergen Gall. Action sets: Weakly supervised action segmentation without ordering constraints. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018a.

Alexander Richard, Hilde Kuehne, Ahsan Iqbal, and Juergen Gall. NeuralNetwork-Viterbi: A framework for weakly supervised video learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018b.

Yaser Souri, Yazan Abu Farha, Emad Bahrami, Gianpiero Francesca, and Juergen Gall. Robust action segmentation from timestamp supervision. In *British Machine Vision Conference (BMVC)*, 2022a.

Yaser Souri, Mohsen Fayyaz, Luca Minciullo, Gianpiero Francesca, and Juergen Gall. Fast weakly supervised action segmentation using mutual consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):6196–6208, 2022b.

Asim Unmesh, Kaki Ramesh, Mayank Patel, Rahul Jain, and Karthik Ramani. Exploring vision-language models for open-vocabulary zero-shot action segmentation, 2026. arXiv:2602.21406.

Fangqiu Yi, Hongyu Wen, and Tingting Jiang. ASFormer: Transformer for action segmentation. In *British Machine Vision Conference (BMVC)*, 2021.

Quan Zhang, Jinwei Fang, Rui Yuan, Xi Tang, Yuxin Qi, Ke Zhang, and Chun Yuan. Weakly supervised temporal action localization via dual-prior collaborative learning guided by multimodal large language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24139–24148, 2025.

Dimitri Zhukov, Jean-Baptiste Alayrac, Ramazan Gokberk Cinbis, David F. Fouhey, Ivan Laptev, and Josef Sivic. Cross-task weakly supervised learning from instructional videos. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3532–3540, 2019.

SUPPLEMENTARY MATERIAL

This supplementary material first gives the parts of the method and the results that do not fit in the main paper (§S1–§S2), then the details the main paper refers to (§S3–§S6), and finally the details of the cost, the datasets and the baselines (§S7–§S8). Section, figure, table and equation numbers without the prefix S refer to the main paper.

S1 INFERENCE

All results in the main paper are obtained with the schedule described here. During training, the model receives inputs in which every frame and every action has an independent noise level. Inference can therefore follow an *asynchronous schedule*, which denoises the easier frames and actions first and uses them to condition the rest (Figure S1). The asynchronous schedule consists of two parts, each evaluated in Table S1.

The segmentation is obtained from the last block as in FACT (Lu & Elhamifar, 2024): each action token predicts an action, and at every frame the prediction of the token that receives the highest attention from the frame is combined with the frame prediction using a fixed weight.

Action set first. Which actions a video contains is easier to determine than where they occur: the actions can be identified from evidence anywhere in the video, whereas locating them requires identifying the right frames. The action set is therefore determined before the frame labels. Every action starts as UNKNOWN; halfway through denoising, the actions to which the existence head assigns a probability above a threshold are marked present and remain so for the remaining steps, and the frame branch denoises the frame labels conditioned on these actions. In Figure S1, k_1 and k_2 are marked present at the halfway step, and k_3 remains undecided until the last step.

Easier frames first. At each step, every frame whose confidence exceeds a threshold has its noise level set to 0 and fixed thereafter, as MaskGIT retains its most confident tokens (Chang et al., 2022). From then on, the frame acts as a timestamp and conditions the remaining frames. The frames below the threshold continue to be denoised step by step. In Figure S1, f_1 exceeds the threshold almost immediately and f_2 only late, while f_3 and f_4 never exceed it and darken gradually until the last step.

Ablation. We remove each part in turn, or both, which gives a synchronous schedule, and compare with the full schedule (Table S1). Removing either part lowers both F1 at IoU 0.1 and Edit, so each part is effective and their combination performs best.

S2 PSEUDO-LABEL GENERATION

Algorithm S1 gives the generation of pseudo-labels described in §3.2.3 of the main paper.

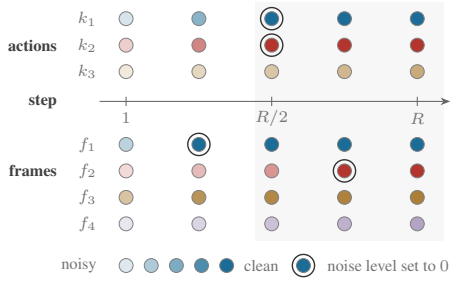


Figure S1: Each row darkens as its frame or action is denoised; a ring marks the step at which its noise level is set to 0.

Table S1: The asynchronous schedule on CrossTask at $\rho_\tau/\rho_\alpha = 80/60$, against one of its parts alone and a synchronous schedule that denoises every frame and action at the same rate.

	F1@10	Edit
asynchronous schedule	40.3	41.8
action set first only	37.4	41.2
easier frames first only	39.9	40.7
synchronous	36.8	40.5

Algorithm S1 Pseudo-label generation for one video.

1: $\ell_t \leftarrow 0$ if $(t, c) \in \mathcal{T}$ else N; $\gamma_k \leftarrow 0$ if $\mathcal{A} \neq \emptyset$ else 1
2: $\hat{y}_t \leftarrow c$ if $(t, c) \in \mathcal{T}$ else undefined
3: **for** each of R steps **do**
4: $\tilde{\mathbf{y}}, \tilde{\mathbf{z}} \leftarrow \mathrm{corrupt}(\hat{\mathbf{y}}, \boldsymbol{\ell}),\ \mathrm{corrupt}(\mathbf{z}, \boldsymbol{\gamma})$ ▷ equation 2, equation 3; known labels remain clean
5: $\hat{\mathbf{y}} \leftarrow \mathrm{Model}(\mathbf{x}, \tilde{\mathbf{y}}, \boldsymbol{\ell}, \tilde{\mathbf{z}})$
6: $\hat{y}_t \leftarrow c$ for every $(t, c) \in \mathcal{T}$ ▷ the timestamps overwrite the prediction
7: $\ell_t \leftarrow 0$ if $(t, c) \in \mathcal{T}$ else one step lower
8: **end for**
9: **return** $\hat{y}_t$ and its confidence $\hat{p}_t$ at every frame without a timestamp

S3 PROMPTS

Bracketed italics mark the fields substituted for each video.

Action set.

```
You are analyzing a cooking/tutorial video based on its
YouTube auto-generated subtitles.
Task steps (from recipe/template):
[numbered list of the actions in C_task]
Video subtitles (may contain ASR errors and irrelevant
chatter):
--- [complete subtitle track] ---
Instructions:
- Your goal is to produce a HIGH-PRECISION set of steps that
are DEFINITELY performed in this video.
- Only mark a step as "present" if the subtitle EXPLICITLY
describes the action being performed (e.g. "now I'm adding
flour", "let me crack the egg", "pour the milk in").
- Do NOT mark a step as present if it is only mentioned,
referenced, or implied (e.g. "I'll add flour later", "you
could also add sugar", "the recipe calls for eggs").
- When in doubt, put the step in "absent", NOT "uncertain".
Only use "uncertain" for steps where the subtitle strongly
suggests the action happened but you're not 100% sure.
- It is perfectly fine to have very few or zero "present"
steps. Precision is more important than recall.
- Return ONLY a JSON object with this exact format:
{"present": ["step_name_1"], "absent": ["step_name_2",
"step_name_3"], "uncertain": ["step_name_4"], "reasoning":
"brief explanation"}
- Use the EXACT step names from the list above.
```

The returned `present` list is the action set $\mathcal{A}$. The actions listed as `absent` or `uncertain` are discarded.

Timestamps.

```
[frames sampled at uniform intervals, one per segment on
average]
```

```
You are watching frames sampled from a video of the task
"[task title]".
Task steps:
[indexed list of the actions in C_task]
Video subtitles (may contain ASR errors and irrelevant
chatter):
--- [complete subtitle track] ---
Frames:
[Frame i (t=...s) for each frame]
For each frame, determine which single task step is being
performed, or whether it is background.  Label only the [η]
of the frames you are most confident about, and leave every
other frame out of your answer.
Reply with ONLY a JSON object mapping frame numbers to step
indices:  {"1":  0, "3":  -1, "4":  2, ...}
Rules:
- Use the step index from the list above; use -1 for
background (no action is being performed).
- Choose ONLY ONE step per frame --- the one that BEST
matches what you see.
- Pay attention to the DIFFERENCE between similar steps
(e.g.  "add flour" vs "add sugar").
- If a frame shows an action but you cannot determine WHICH
specific step, leave it out.
```

The labelled frames, including those labelled as background, are the timestamps $\mathcal{T}$.

S4 MATCHING

The frames of a video are partitioned into chunks of a fixed length (Table S2), and each chunk is assigned to at most one action token (§3.2.2). A chunk can be assigned to an action of the candidate set, which contains the actions in the action set, the actions of the timestamps and background. For a video without an action set, an action also enters the candidate set when the existence head assigns it a probability of at least 0.5. A chunk is assignable to an action when the frame probability of that action reaches 0.35 somewhere in the chunk. A chunk that contains a timestamp receives a score bonus of 0.3 towards the action of the timestamp, and all its frames are supervised. A chunk without a timestamp needs a peak probability of 0.6, its attention target is restricted to the frames where the probability of the action is at least 0.3, and at most five tokens per video are spent on such chunks. A chunk is assigned to background only when every action fails, with a peak probability of 0.75 and at most eight background tokens per video. Each chunk goes to the token with the highest score, which weights the frame prediction and the prediction of the token equally, so an action with several confident chunks claims several tokens. The token is then supervised with the action of its chunks.

S5 LOSS

Every diffusion block is trained with

$$\mathcal{L} \;=\; 10\,\mathcal{L}_{\mathrm{f}} \;+\; \mathcal{L}_{\mathrm{a}} \;+\; 0.5\,\mathcal{L}_{\mathrm{s}} \;+\; \mathcal{L}_{\mathrm{ca}} \;+\; 5\,\mathcal{L}_{\mathrm{smooth}}. \tag{S1}$$

$\mathcal{L}_{\mathrm{f}}$ is the cross-entropy between the frame prediction and the label of each frame, the timestamp where one exists and the pseudo-label otherwise, with each frame weighted by w_t in equation 5. $\mathcal{L}_{\mathrm{a}}$ is the cross-entropy between the action predicted by each action token and the action assigned to it by the matching (§S4); a token without a chunk is trained towards a null class with weight 0.08, and background targets have weight 0.5. $\mathcal{L}_{\mathrm{s}}$ is the binary cross-entropy between the prediction of the existence head and the state of every action that is not UNKNOWN. $\mathcal{L}_{\mathrm{ca}}$ supervises the attention from the frames to the action tokens and from the action tokens to the frames with the matching as the target. $\mathcal{L}_{\mathrm{smooth}}$ penalises temporal changes in the frame prediction and in both attention maps. The input block receives no corrupted supervision and is trained with $10\,\mathcal{L}_{\mathrm{f}} + \mathcal{L}_{\mathrm{a}} + 5\,\mathcal{L}_{\mathrm{smooth}}$, the smoothness term applying to its frame prediction only.

Table S2: Settings on CrossTask.

blocks	one input block and three diffusion blocks
frame branch	10 temporal layers
action branch	6 attention layers with 8 heads in the input block
noise levels	$N = 1000$, cosine schedule; $\kappa = 0.8$
action noise level γ_k	$1/3$ in the action set, $2/3$ outside $\mathcal{C}_{\text{task}}$, in training
self-conditioning	with probability 0.5 in training
pseudo-label generation	$R = 10$ steps; a pseudo-label is kept above confidence 0.7
pseudo-label schedule	after a 30-epoch warm-up, every 30 epochs
inference	10 denoising steps (§S1)
inference thresholds	0.7 on the existence probability and the frame confidence
optimiser	Adam, learning rate 2×10^{-4}, decayed $10\times$ after epoch 30
batch	1 video; gradients accumulated over 4 videos, clipped at norm 10
action tokens M	60
chunk length	12 frames
readout weight of the frame prediction	0.9

Table S3: Tokens per video consumed by the MLLM to generate each format of supervision on CrossTask.

	action set	timestamps
instructions	299	310
candidate actions	38	38
subtitles	1,040	1,040
sampled frames (12.3 on average, 1,105 tokens each)	—	13,592
input tokens	1,377	14,980
output tokens	75	56
reasoning tokens	480	1,600
cost, output and reasoning tokens counted $8\times$	5,817	28,228

S6 Implementation details

Table S2 lists the settings on CrossTask.

The features are pre-extracted; on CrossTask they have 512 dimensions per frame.

S7 Cost of the supervision

Table S3 breaks down the tokens behind $r = 4.9$ in equation 1. Both inputs contain instructions of similar length, the candidate actions and the subtitles; the input for the timestamps also contains the sampled frames, which account for most of its tokens. Following the price ratio of the MLLM, each output token, including reasoning tokens, counts as eight input tokens. With $r = 4.9$, the budget B is 83 with timestamps on every video and no action set, and 17 with an action set on every video and no timestamps.

S8 Datasets and baselines

Datasets. We use split 1 of each dataset (Table S4). On CrossTask, the MLLM generates the supervision of every training video, and background makes up 74.1% of the test frames. On Breakfast and GTEA, the timestamps and the action set are derived from the ground truth, with timestamps on the same share η of the sampled frames as on CrossTask.

Baselines. Each baseline receives exactly the supervision our model receives at the corresponding single-format allocation, and all methods are scored with the same evaluation code. TS-SEG

Table S4: Datasets, split 1.

	CrossTask	Breakfast	GTEA
training / test videos	1,503 / 500	1,460 / 252	21 / 7
actions	105 and background	48	11

(Li et al., 2021) is run without its label expansion, which requires a timestamp in every segment. EM-GEN (Rahaman et al., 2022) models at most one missing segment between two consecutive timestamps and was evaluated with up to 20% of the segments missing.