Rethinking Learning Approaches for Long-Term Action Anticipation

Megha Nawhal[1], Akash Abdu Jyothi[1], and Greg Mori[1,2]

[1] Simon Fraser University, Burnaby, Canada
[2] Borealis AI, Vancouver, Canada

A Appendix

In this document, we provide additional quantitative and qualitative analyses, and additional details of the implementation of our approach. Specifically, this document contains the following items.

- Sec. A.1: Technical details of the implementation and evaluation of our proposed approach
  - Sec. A.1.1: Architecture details (network architectures and loss function)
  - Sec. A.1.2: Implementation details (input representations and hyperparameters)
  - Sec. A.1.3: Evaluation details
- Sec. A.2: Additional ablation analysis
- Sec. A.3: Additional visualizations and qualitative analysis
- Sec. A.4 Additional discussion

A.1 Technical Details

In this section, we provide additional details for implementation of our proposed approach ANTICIPATR to supplement Sec. 3 in the main paper.

A.1.1 Architecture Details

We propose ANTICIPATR that uses a two-stage learning approach to train a transformer-based model for the task of long-term action anticipation. The model comprises three networks: *segment encoder*, *video encoder* and *anticipation decoder*. Fig. F1 shows the architecture of the three networks.

In the first stage, we train a *segment encoder* that receives a segment (sequence of frames from a video) as input and predicts the set of action labels that would occur at any future time instant after the occurrence of the segment in the video.

In the second stage, we train a *video encoder* and an *anticipation decoder* to be used along with the segment encoder for long-term action anticipation. The video encoder encodes the observed video to a video-level representation. The segment encoder (trained in the first stage) is fed with a sequence of segments

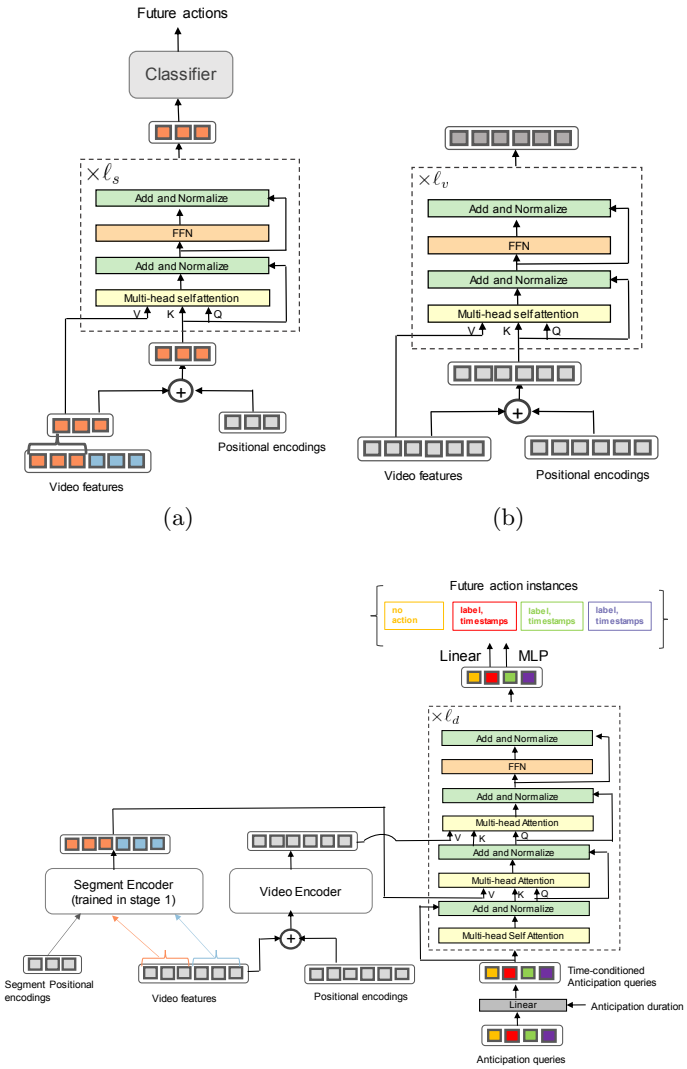


Fig. F1. Detailed Architecture. Architecture overview of (a) Segment encoder, (b) Video encoder, and (c) Anticipation Decoder. Refer to Sec. A.1.1 for details. 'Q','K','V' are query, key and value to the self-attention layer as described in [10].

from the observed video as input to obtain a segment-level representation of the video. The anticipation decoder receives the two representations along with the anticipation duration to predict a set of future action instances over the given anticipation duration in a single pass. The video encoder and anticipation decoder are trained using classification losses on the action labels and two temporal losses (L_1 loss and temporal IoU loss) on the timestamps while the segment encoder is kept unchanged.

Positional Encoding for Segment Encoder. The input to the segment encoder is a video segment. We represent the segment as a sequence of features. As the encoder is permutation-invariant, we provide temporal information in the segment using the sinusoidal positional encodings (c.f. Vaswani *et al.* [10]) based on timestamps corresponding to the features of input segment. Specifically, for each input feature of each embedding we independently use sine and cosine functions with different frequencies. We then concatenate them along the channel dimension to get the final positional encoding. In our implementation, the embedding size is same as that of the segment feature so that they can be combined by simple addition of the positional encodings and segment features.

Positional Encoding for Video Encoder. The input to the video encoder is a video. We represent the video as a sequence of features. As the transformer encoder is permutation-invariant, we provide temporal information in the input video using the sinusoidal positional encodings (c.f. Vaswani *et al.* [10]) based on timestamps corresponding to the features of input video. Specifically, for each input feature of each embedding we independently use sine and cosine functions with different frequencies. We then concatenate them along the channel dimension to to get the final positional encoding. In our implementation, the embedding size is same as that of the video feature so that they can be combined by simple addition of the positional encodings and video features.

Anticipation Queries (Anticipation Decoder). The anticipation queries are learnable positional encoding designed as a learnable embedding layer. The positional encoding layer receives integer index i as input corresponding to $i-th$ anticipation query and provides an embedding $\mathbf{q}_0^i$ where $i \in \{1, \ldots, N_a\}$. In our implementation, we use `torch.nn.Embedding` in Pytorch to implement this. The weights of the layer are learnable during training, thus, the positional encoding layer is also learnable. The initialization of this layer requires maximum possible value of the index, *i.e.*, N_a in our case.

The anticipation queries $\mathbf{q}_0$ are then combined with anticipation duration T_a using a simple neural network to create time-conditioned anticipation queries $\mathbf{q}_a$. These time-conditioned queries enable the model to predict actions over any specified anticipation duration.

Training. We provide supplemental details about computation of loss function used to train the networks in the second stage (*i.e.*, action anticipation stage) of our ANTICIPATR approach. The training involves aligning groundtruth and predicted set of action instances followed by computing the anticipation loss over all aligned pairs.

Greedy Set Correspondence. Given an observed video, the groundtruth set of future action instances varies based on input whereas our anticipation decoder predicts a set of fixed size (larger than maximum size of groundtruth sets in the dataset). Therefore, there is no prior correspondence between the groundtruth and predicted set. We derive this correspondence using a greedy algorithm based on temporal overlap among instances. Intuitively, the objective is to correctly align actions at as many future time instants as possible. We first sort the action instances groundtruth set based on the descending order of the duration of the instances. We begin the alignment process with the groundtruth instance having the maximum duration. We lookup the predicted set to find the predicted instance that has maximum temporal overlap with this groundtruth instance. Since the predicted set is designed to represent a single action instance, the alignment between groundtruth and predicted set is one-to-one. Thus, to continue the alignment process, the matched groundtruth instance and predicted instance are removed from the corresponding sets. In this way, this process is repeated until the groundtruth set is empty. As the predicted set is of size larger than groundtruth set, the remaining predicted instances are mapped to $\varnothing$ denoting no action. In Sec. A.2, we also evaluate anticipation results of models trained using another set correspondence algorithm, namely, Hungarian matcher (see Table T9 and Table T10).

Loss function. We compute loss $\mathcal{L}$ (defined in Eq. (4) in the main paper) over all the matched pairs as a weighted combination of cross-entropy loss for classification and two temporal losses ($L1$ loss and IoU loss $\mathcal{L}_{iou}$) for prediction of segment timestamps. Here, we provide our motivation behind temporal loss and provide additional description.

The L_1 temporal loss is sensitive to the absolute value of the duration of the segments. The IoU loss $\mathcal{L}_{iou}$ is invariant to the duration of the segments. Thus, these two losses together are designed to incorporate different aspects of segment prediction. For completeness, we describe $\mathcal{L}_{iou}$ as follows.

$$\mathcal{L}_{iou}(s^i, \hat{s}^{\gamma(i)}) = 1 - \frac{|s^i \cap \hat{s}^{\gamma(i)}|}{|s^i \cup \hat{s}^{\gamma(i)}|}, \tag{1}$$

where $|.|$ is the duration of the instance, *i.e.*, difference between end and start timestamp.

A.1.2 Training Details

For training of first stage, we use dropout probability of 0.1. For the segment encoder, we use base model dimension as 2048 and set the number of encoder layers as 3 with 8 attention heads. We use an effective batch size of 64 for training segment encoder on this dataset. For training in the second stage, we use base model dimension in the video encoder and anticipation decoder as 2048 and set the number of encoder and decoder layers as 3 with 8 heads.

We use four datasets – Breakfast, 50Salads, EPIC-Kitchens-55, EGTEA Gaze+ – to evaluate our model on long-term action anticipation. We provide dataset-specific hyperparameters as follows.

We train all our models using AdamW [6] optimizer on 4 Nvidia V100 32GB GPUs. We initialize all the learnable weights using Xavier initialization.

Breakfast. We represent input videos as I3D features provided by [1]. We choose N_a (anticipation queries) to be 150. We use an effective batch size of 16 for training the video encoder and anticipation decoder on this dataset on the long-term anticipation task. We train our models with a learning rate of 1e-4 and a weight decay of 0. The model is trained for 4000k steps. We use a dropout probability of 0.1. We set $\lambda_{L1} = 3$ and $\lambda_{iou} = 5$. To obtain segment-level representation of the observed video during action anticipation, we use a temporal window of length $k = 16$.

50Salads. We represent input videos as Fisher vectors computed using [2]. We choose N_a (anticipation queries) to be 80. We use an effective batch size of 16 for training the video encoder and anticipation decoder on this dataset on the long-term anticipation task. We use a learning rate of 1e-5 and a weight decay of 1e-5. We train the model for 3000k steps and reduce the learning rate by factor of 10 after 1500k steps. We don't use dropout for this dataset. We set $\lambda_{L1} = 3$ and $\lambda_{iou} = 5$. To obtain segment-level representation of the observed video during action anticipation, we use a temporal window of length $k = 48$.

EPIC-Kitchens-55. We represent input videos as I3D features provided by [3, 7]. We use an effective batch size of 16 for training the video encoder and anticipation decoder in the second stage. We choose N_a (anticipation queries) to be 900. We use a learning rate of 1e-4 and a weight decay of 1e-5. We train the model for 6000k steps and reduce the learning rate by factor of 10 after 4000k steps. We use a dropout probability of 0.1. We set $\lambda_{L1} = 5$ and $\lambda_{iou} = 8$. To obtain segment-level representation of the observed video during action anticipation, we use a temporal window of length $k = 32$.

EGTEA Gaze+. We represent input videos as I3D features provided by [3, 7]. We use an effective batch size of 16 for training the video encoder and anticipation decoder in the second stage. We choose N_a to be 600. We use a learning rate of 1e-5 and a weight decay of 1e-5. We train the model for 4000k steps and reduce the learning rate by factor of 10 after 3000k steps. We use a dropout probability of 0.1. We set $\lambda_{L1} = 3$ and $\lambda_{iou} = 5$. To obtain segment-level representation of the observed video during action anticipation, we use a temporal window of length $k = 24$.

A.1.3 Evaluation Details

Note that our model predicts a set of action instances, wherein, each action instance is of the form (label, start time, end time). To evaluate the model outputs as per the benchmarks, we do the following postprocessing.

For Breakfast and 50Salads, following the benchmark [9], we evaluate the action anticipation outputs over a dense timeline. Our proposed ANTICIPATR predicts a set of action instances. During evaluation, we process this set of action

Table T1. Ablation: Loss function (Breakfast and 50Salads). We report the mean over classes accuracy for different observation/anticipation durations. Higher values indicate better performance. ✓ and ✗ indicate whether the component of the temporal loss is used or not respectively.

	Method	$\beta_o \rightarrow$	20%				30%			
		$\beta_a \rightarrow$	10%	20%	30%	50%	10%	20%	30%	50%
Breakfast	L_1: ✗; $\mathcal{L}_{iou}$:✓		36.2	30.7	28.6	26.4	38.7	33.9	31.0	27.3
	L_1: ✓; $\mathcal{L}_{iou}$:✗		36.5	31.1	29.1	28.2	39.2	34.2	31.7	28.1
	L_1: ✓; $\mathcal{L}_{iou}$:✓		**37.4**	**32.0**	**30.3**	**28.6**	**39.9**	**35.7**	**32.1**	**29.4**
50Salads	L_1: ✗; $\mathcal{L}_{iou}$:✓		40.2	33.9	26.8	26.0	41.9	41.4	27.6	23.3
	L_1: ✓; $\mathcal{L}_{iou}$:✗		40.8	34.5	27.1	26.8	42.1	41.6	27.9	23.4
	L_1: ✓; $\mathcal{L}_{iou}$:✓		**41.1**	**35.0**	**27.6**	**27.3**	**42.8**	**42.3**	**28.5**	**23.6**

Table T2. Ablation: Loss function (EK-55 and EGTEA+). We report mAP values for ALL classes, FREQUENT classes (> 100 action instances) and RARE class (< 10 action instances). Following [7], we report the mAP values averaged over different observation durations. Higher values implies better performance. ✓ and ✗ indicate whether the component of the temporal loss is used or not respectively.

Method	**EK-55**			**EGTEA+**		
	ALL	FREQ	RARE	ALL	FREQ	RARE
L_1: ✗; $\mathcal{L}_{iou}$:✓	34.9	56.4	27.3	75.2	82.1	53.8
L_1: ✓; $\mathcal{L}_{iou}$:✗	37.7	57.8	28.4	76.0	82.7	54.6
L_1: ✓; $\mathcal{L}_{iou}$:✓	**39.1**	**58.1**	**29.1**	**76.8**	**83.3**	**55.1**

instances to construct a timeline corresponding to the anticipation duration. We refer to the timeline as a sequence of action labels for time instants in the anticipation duration, *i.e.*, between $T_o + 1, \ldots, T_o + T_a$. In the benchmarks, the timeline contains a single action class corresponding to each time instant. We iterate over the predicted set to assign class labels to this timeline. Specifically, for each action instance in the predicted set, we assign the predicted action class to the time instants that are within the predicted segment (determined by predicted start and end timestamp). When predicted action instances overlap at certain time instants, we assign the action class with highest probability score among the overlapping predictions. Once the timeline is constructed, we compute mean over classes accuracy [9] to evaluate the model performance. Note that we are constructing this timeline only during evaluation to follow the benchmark evaluation protocols.

For EPIC-Kitchens-55 and EGTEA Gaze+, we perform a union over the action classes in the predicted set of instances to obtain a set of future action classes. We remove $\varnothing$ class from this set and use this set to compute mAP as described in benchmark [7].

A.2 Additional Ablation Analysis

In this section, we report our findings from additional ablation experiments.

Ablation: Loss function. The training loss function defined in Eq. (4) in the main paper contains three components (cross-entropy loss and two temporal losses). We conduct ablation experiments by removing one of the temporal losses.

Table T3. Ablation: Anticipation Queries (Breakfast and 50Salads). We report the mean over classes accuracy for different observation/anticipation durations. Higher values indicate better performance.

	Method	$\beta_o \rightarrow$	20%				30%			
		$\beta_a \rightarrow$	10%	20%	30%	50%	10%	20%	30%	50%
Breakfast	$N_a = 50$		32.6	28.2	26.4	24.3	35.8	31.4	28.7	25.3
	$N_a = 150$		**37.4**	**32.0**	**30.3**	**28.6**	**39.9**	**35.7**	**32.1**	**29.4**
	$N_a = 500$		36.6	31.5	29.4	27.3	38.5	34.4	31.3	28.3
50Salads	$N_a = 20$		38.4	33.2	24.2	23.6	39.1	35.6	25.5	24.2
	$N_a = 80$		**41.1**	**35.0**	**27.6**	**27.3**	**42.8**	**42.3**	**28.5**	**23.6**
	$N_a = 320$		40.5	34.2	26.0	25.6	41.3	40.9	27.4	23.3

Table T4. Ablation: Anticipation Queries (EK-55 and EGTEA+). We report mAP values for ALL classes, FREQUENT classes ($>$ 100 action instances) and RARE class ($<$ 10 action instances). Following [7], we report the mAP values averaged over different observation durations. Higher values implies better performance.

Dataset		ALL	FREQ	RARE
EK-55	$N_a = 300$	34.3	55.6	24.2
	$N_a = 900$	**39.1**	**58.1**	**29.1**
	$N_a = 2700$	38.2	56.9	28.3
EGTEA+	$N_a = 200$	70.2	79.5	49.7
	$N_a = 600$	**76.8**	**83.3**	**55.1**
	$N_a = 1800$	75.3	82.4	53.3

Note that we always need cross entropy loss for the classification task. Results in Table T1 and Table T2 show that models trained with overall loss perform better than the ones trained with the ablated versions. Moreover, the models trained with only L_1 temporal loss perform better than the ones trained with only $\mathcal{L}_{iou}$.

Ablation: Anticipation queries. The number of anticipation queries discerns the maximum number of action instances the model is supposed to predict. Results in Table T3 and Table T4 shows the performance of our model with different number of anticipation queries. The results suggest minor improvement with higher number of anticipation queries, however, the models with more number of queries require longer training times. Intuitively, a very large number of anticipation queries implies the model will require more time to learn the non-maximal suppression of the irrelevant predictions. On the other hand, when the number of anticipation queries is reduced, the anticipation performance of our model degrades. A very small number of anticipation queries implies less number of action are anticipated. Thus, for very complex video with many future action instances, the model would miss several action instances resulting in poor anticipation performance. Additionally, as shown in Table T3, the anticipation error increases over time. This is because there are more actions to be anticipated and the model is limited by the number of anticipation queries.

Ablation: Segment window length. Results in Table T5 and Table T6 shows the performance of our model with different values of temporal window lengths used to extract segment-level representations during action anticipation. The results suggest that neither a very small window length nor a very large window is helpful. The segment encoder is trained to predict future actions given a video

Table T5. Ablation: Segment window length (Breakfast and 50Salads). We report the mean over classes accuracy for different observation/anticipation durations. Higher values indicate better performance.

	Method $\beta_o \rightarrow$		20%				30%			
		$\beta_a \rightarrow$	10%	20%	30%	50%	10%	20%	30%	50%
Breakfast	$k = 4$		35.9	30.6	26.3	26.1	38.4	33.6	30.8	28.2
	$k = 16$		**37.4**	**32.0**	**30.3**	**28.6**	**39.9**	**35.7**	**32.1**	**29.4**
	$k = 64$		37.4	31.7	29.9	28.1	39.1	35.0	31.7	28.7
50Salads	$k = 12$		39.0	33.5	25.8	25.4	39.6	38.4	26.4	21.5
	$k = 48$		**41.1**	**35.0**	**27.6**	**27.3**	**42.8**	**42.3**	**28.5**	**23.6**
	$k = 192$		41.0	34.8	27.2	26.8	42.6	42.1	27.5	22.8

Table T6. Ablation: Segment window length (EK-55 and EGTEA+). We report mAP values for ALL classes, FREQUENT classes (> 100 action instances) and RARE class (< 10 action instances). Following [7], we report the mAP values averaged over different observation durations. Higher values implies better performance.

Dataset		ALL	FREQ	RARE
EK-55	$k = 8$	37.9	57.2	27.4
	$k = 32$	**39.1**	**58.1**	**29.1**
	$k = 128$	38.8	58.0	28.7
EGTEA+	$k = 6$	75.4	81.7	53.9
	$k = 24$	**76.8**	**83.3**	**55.1**
	$k = 96$	76.3	82.9	54.8

segment depicting a single action. During the action anticipation stage, when the segment encoder is used to extract segment-level representations, the observed video is divided into a series of non-overlapping segment using temporal sliding windows as the action boundaries are not known. Intuitively, when the temporal sliding window is very small, the individual segments do not have enough information to obtain effective representations. On the other hand, when the window is very large, the segments contain more than one action and potentially results in segment-level representations with overlapping semantic content. We observe that the drop in performance with models that use smaller window lengths is larger as compared to the ones with larger window lengths.

Ablation: Sliding Windows for Segment Encoder Training. Instead of using action boundaries we used sliding temporal windows of length=k (same as used during stage 2) to obtain segments for segment-level training. Results in Table T7 and Table T8 show that this approach results in a slightly lower performance than our proposed training approach. This is possibly due to increased noise in the segment-level representations from this training approach.

Table T7. Ablation: Sliding windows for Segment Encoder Training. Mean over classes accuracy for different observation/anticipation durations. Higher is better. [BF: Breakfast; 50SL: 50Salads]

	Observation (β_o) $\rightarrow$	20%				30%			
	Anticipation (β_a) $\rightarrow$	10%	20%	30%	50%	10%	20%	30%	50%
BF	Sliding windows	35.9	30.7	28.0	26.4	37.8	33.5	29.9	25.2
	Anticipatr(Full)	**37.4**	**32.0**	**30.3**	**28.6**	**39.9**	**35.7**	**32.1**	**29.4**
50SL	Sliding windows	37.2	33.5	26.3	25.8	37.9	37.0	26.1	24.5
	Anticipatr(Full)	**41.1**	**35.0**	**27.6**	**27.3**	**42.8**	**42.3**	**28.5**	**23.6**

Table T8. Ablation: Sliding windows for Segment Encoder Training. mAP values for ALL classes, FREQUENT classes (> 100 action instances) and RARE class (< 10 action instances). Higher is better.

Method	EK-55			EGTEA+		
	ALL	FREQ	RARE	ALL	FREQ	RARE
Sliding Windows	37.6	56.5	27.4	74.8	81.2	53.0
No Video Encoder	30.9	51.8	21.2	70.2	79.9	50.1
Anticipatr(Full)	**39.1**	**58.1**	**29.1**	**76.8**	**83.3**	**55.1**

Ablation: Set correspondence. To compute the anticipation loss, we use a greedy algorithm to align groundtruth and predicted set of action instances. Another commonly employed set correspondence algorithm is Hungarian matcher algorithm used in prior works [4, 5]. For completeness, we also conducted experiments with Hungarian matcher optimized over the cost function with all three terms (classification loss and two temporal losses) following [8]. We didn't observe any significant difference in performance of the models trained using either of the two matchers as shown in Table T9 and Table T10.

A.3 Additional Qualitative Analysis

Visualizations in Fig. F2 and Fig. F3 show that our model is generally able to anticipate correct actions at future time instants long anticipation durations for Breakfast and 50Salads benchmarks respectively.

Visualizations in Fig. F4 and Fig. F5 show that our model is able to effectively predict future action classes for EK-55 and EGTEA benchmarks respectively.

Failure Cases. We observe that the action boundaries in some cases are not exactly aligned with the groundtruth even though the class labels are predicted accurately (See Fig. F2 and Fig. F3). We believe this could be because the visual information pertaining to the information is limited or negligible towards the beginning and end of the action instance.

Most classification errors result from the model getting confused among semantically similar classes. Some such cases from our examples are *'take ladle'* and *'pick-up ladle'* in Fig. F4(b)); *'close sandwich'* and *'close hamburger'* in Fig. F4(d)); *'put seasoning'* and *'pour seasoning'* in Fig. F5(a)).

Moreover, our model sometimes misses rare actions during predictions such as *'pour oil'* in Fig F4(a) and *'close fridge'* in Fig. F5(b).

Additionally, we also observe that having seen certain objects in the observed video, the model predicts objects that are likely to co-occur with the seen objects. See the scenario in Fig. F3(d). The model doesn't predict *'cut_cheese'* and *'place_cheese_into_bowl'* after the action *'place_cucumber_into_bowl'* and instead predicts *cut_tomato* and *'place_tomato_into_bowl'*. While the prediction is not correct for this specific activity, it is still a reasonable sequence of actions as there are several other salad recipe videos in the dataset that only use *cucumber* and *tomato*. In another scenario in Fig. F5(b), having seen *'pasta'* in the observed video, the model anticipates action classes with *'cheese'* noun. While

Table T9. Ablation: Set correspondence (Breakfast & 50Salads). We report the mean over classes accuracy for different observation/anticipation durations. Higher values indicate better performance.

	Method	$\beta_o \rightarrow$	20%				30%			
		$\beta_a \rightarrow$	10%	20%	30%	50%	10%	20%	30%	50%
Breakfast	Hungarian		36.8	**32.0**	**30.5**	28.4	39.2	35.4	31.9	**29.6**
	Greedy		**37.4**	32.0	**30.3**	**28.6**	**39.9**	**35.7**	**32.1**	29.4
50Salads	Hungarian		**41.3**	**35.1**	27.4	26.8	**42.9**	42.0	28.4	**23.8**
	Greedy		41.1	35.0	**27.6**	**27.3**	**42.8**	**42.3**	**28.5**	**23.6**

Table T10. Ablation: Set correspondence (EK-55 & EGTEA+). We report mAP values for ALL classes, FREQUENT classes ($>$ 100 action instances) and RARE class ($<$ 10 action instances). Following [7], we report the mAP values averaged over different observation durations. Higher values implies better performance.

Method	**EK-55**			**EGTEA+**		
	ALL	FREQ	RARE	ALL	FREQ	RARE
Hungarian	39.0	**58.4**	28.4	76.7	**83.5**	55.0
Greedy	**39.1**	58.1	**29.1**	**76.8**	83.3	**55.1**

'cheese' does not appear in this particular video, it is a reasonable prediction since the nouns *'pasta'* and *'cheese'* often appear together in activity videos in this dataset.

A.4 Additional Discussion

In this work, we demonstrate the effectiveness of our model on minutes-long activity videos. Handling longer videos with durations in hours or days (common in surveillance or monitoring scenarios) would be interesting future work. Furthermore, our approach assumes that the videos have an overall context provided by the ongoing long-term activity. We show that modeling interactions among segments (and, in turn, segment-level representation) is an effective technique for such activity videos as the video segments are indeed related. However, such approaches cannot tackle videos that are just a montage of several unrelated content like videos containing clips from different movies. Our approach focuses on activity videos where contextual information is present and relevant for action anticipation.

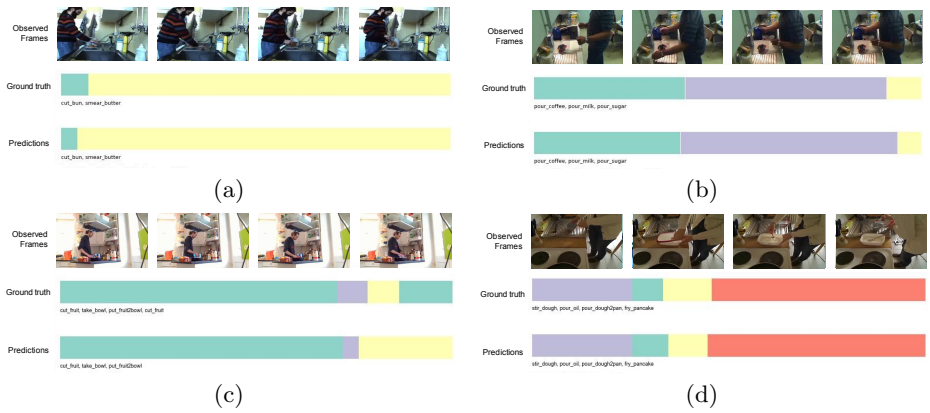


Fig. F2. Visualizations (Breakfast). Examples from Breakfast dataset for the case where observation duration is 20% of the video duration and anticipation duration involves predicting actions for 50% of the remaining video.

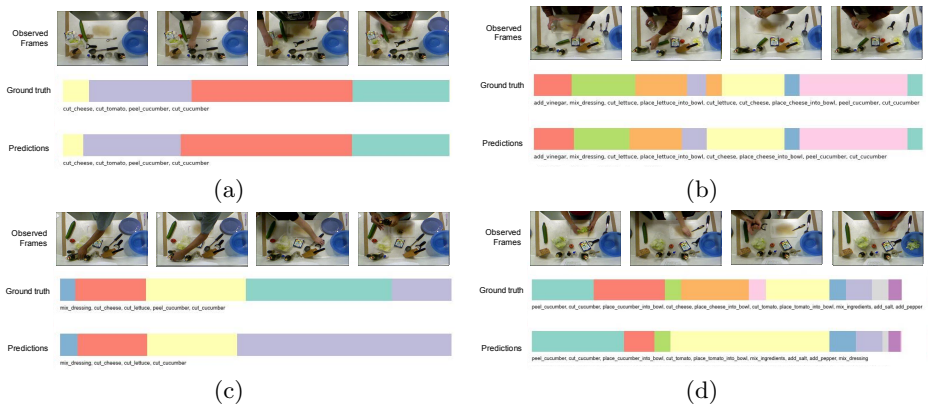


Fig. F3. Visualizations (50Salads). Examples from 50Salads dataset for the case where observation duration is 20% of the video duration and anticipation duration involves predicting actions for 50% of the remaining video.

pour oil
take pan
put-down spoon
put-down fork
take pan

turn-off stove
put-down pan
put egg
lift pan

(a)

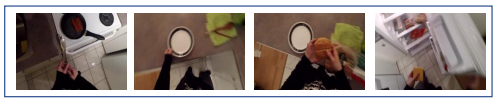
mix sauce
lift pot
put pasta
take ladle
pick-up ladle
stir pasta

mix sauce
lift pot
put spoon
put fork
put cheese

(b)

serve jambalaya
serve food
take fork
pick-up fork
put-down fork
move pot

(c)



put sausage
put cheese
put tomato
close sandwich
close hamburger
put-down tongs

(d)

Fig. F4. Visualizations (EK-55). Examples from Epic-Kitchens-55 dataset for the case where observation duration is 50% of the video duration. We show the predicted action classes in the visualization – classes in green color are correct predictions, classes in red color are wrong predictions, and classes in gray color are missed classes.

mix pasta
put condiment
take condiment-container
put seasoning
pour seasoning
take bowl
move-around pot
put pasta
pour pasta

take bell-pepper
mix bell-pepper
take oil-container
put pot
cut tomato
put cheese
mix cheese
mix seasoning

(a)

put cheese
close fridge
put patty
put lettuce
take lettuce

take condiment-container
take cheese
put tomato
squeeze sandwich
move-around patty

(b)

Fig. F5. Visualizations (EGTEA+). Examples from EGTEA Gaze+ dataset for the case where 50% of the video is observed. We show the predicted action classes in the visualization – classes in green color are correct predictions, classes in red color are wrong predictions, and classes in gray color are missed classes.

References

1. https://github.com/yabufarha/ms-tcn
2. https://bitbucket.org/doneata/fv4a/src/master/
3. https://github.com/facebookresearch/ego-topo
4. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: Proceedings of the European Conference on Computer Vision (ECCV) (2020)
5. Kim, B., Lee, J., Kang, J., Kim, E.S., Kim, H.J.: Hotr: End-to-end human-object interaction detection with transformers. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2021)
6. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization (2017)
7. Nagarajan, T., Li, Y., Feichtenhofer, C., Grauman, K.: Ego-topo: Environment affordances from egocentric video. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
8. Nawhal, M., Mori, G.: Activity graph transformer for temporal action localization. arXiv preprint arXiv:2101.08540 (2021)
9. Sener, F., Singhania, D., Yao, A.: Temporal aggregate representations for long-range video understanding. In: Proceedings of the European Conference on Computer Vision (ECCV) (2020)
10. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems (NIPS) (2017)