Supplement: Markov Game Video Augmentation for Action Segmentation

Nicolas Aziere
Oregon State University
azieren@oregonstate.edu

Sinisa Todorovic
Oregon State University
sinisa@oregonstate.edu

1. Introduction

In the following, Sec. 2 depicts the algorithm used to generate features from new transcripts. Sec. 3 details the architecture of agent 2's proposal module. Sec. 4 provide an ablation study on the reward functions and Sec. **??** a study on the amount of transitions introduced by new transcript.

2. Feature Generation from New Transcript

When a new transcript is generated using the Monte-Carlo Tree Search, the next step is to find the associated deep features representing it. In Algorithm. 1, we show the pseudo-code describing the process of selecting the features for a new transcript π from the dataset $\mathcal{D}$. The features are copy and paste from videos having a similar transcript to π. We use the edit distance function to sort all videos in terms of edit distance, then iteratively select matching segments between π and the transcript of the current video. The segments with no current match are kept empty until being match to a different video. We first generate the same number of new transcripts as there are ground-truth ones, with zero intersection between them. Then, for our experiments, we select a subset (maximum 20%) of the generated transcripts that are closest in edit distance to the ground-truth transcripts.

3. Segment Proposal network Architecture

We detail in this section the architecture of agent 2's head, responsible of selecting the frames to be augmented. The head representing agent μ^2 is shown in Fig. 1. The state is first fed to the backbone network common to both agents 1 and 2. The resulting latent state features are passed to a single stage TCN turned into a proposal network inspired from [1]. The output for each frame is composed of three values: left offset, right offset and a score for each frame $c_{t,k}$. To select intervals of interest and remove unwanted selections, we apply the non-maximum suppression algorithm filtering out overlapping and low-score segment proposals. After the segment proposal selection, we can construct the mask a_k^2 with 1's where time indices fall in the segment proposals and 0's otherwise.

Algorithm 1 Feature generation from new Transcripts

procedure FEATGEN($\pi, \mathcal{D}$) ▷ Input: transcript, dataset
  $X \leftarrow \{a : \{\}$ for $a = 1...|\pi|\}$
    ▷ Initialize hashmap of frame features
  $Y \leftarrow \{a : \{\}$ for $a = 1...|\pi|\}$ ▷ Initialize labels
  $\hat{\mathcal{D}} \leftarrow$ EditSort $(\pi, \mathcal{D})$ ▷ Sort the dataset by edit distance with π from closer to further
  for $\hat{\pi}, \hat{X}, \hat{Y} \leftarrow \hat{\mathcal{D}}$ **do** ▷ Get transcript, features and labels
    $A, \hat{A} \leftarrow$ Match(π, $\hat{\pi}$)
    ▷ Find set of corresponding indices of matching actions
    for $(a, \hat{a}) \leftarrow (A, \hat{A})$ **do**
      if $X[a]$ is empty **then**
        $X[a] \leftarrow \hat{X}[\hat{a}]$
        $Y[a] \leftarrow \hat{Y}[\hat{a}]$
    if X is completed **then**
      return π, X, Y
  return Error

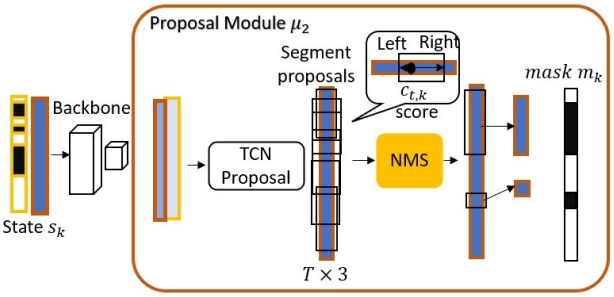

Figure 1. Details of our segment proposal agent. First a latent representation of state s_k is extracted by a backbone network. The deep representation is then fed to the proposal head outputting 3 values per frame: Right offset, Left offset and segment score. The NMS algorithm is applied to filter redundant and low score segment proposals. Finally the agent's action represented by a_k^2 is selected and represent the selection of frames to modify by the other agent.

R_k^1	R_k^2	F1@10,25,50			Edit	MoF
No Cond.	-	86.1	82.7	72.4	81.4	76.2
Cond.	-	88.5	85.0	74.7	83.8	78.2
No Cond.	No Cond.	86.6	83.8	74.4	80.7	77.5
No Cond.	Cond. 1	86.1	84.3	74.8	81.5	77.4
No Cond.	Cond. 2	87.3	83.9	75.0	83.0	77.2
No Cond.	Cond. 1&2	87.5	84.9	75.1	82.7	77.7
Cond.	No Cond.	88.8	87.4	77.0	86.7	79.1
Cond.	Cond. 1	89.6	87.3	77.8	87.4	79.0
Cond.	Cond. 2	89.5	88.0	78.8	87.9	79.4
Cond.	Cond. 1&2	**90.9**	**88.2**	**79.2**	**88.8**	**79.6**

Table 1. MS-TCN performance on GTEA for different definition of the reward functions: gradually, the reward function are added new elements in the form of a conditional statement with the goal of removing unwanted behaviors. First in the top 2 rows, only R_k^1 is evaluated on the single-agent case. In the multi-agent setting we show the boost in performance when including new constraints as part of the reward R_k^2.

4. Reward Ablation

In this section, we vary the definition of our rewards for training the policy of agent 1 and 2. Since the definition of the rewards is heuristic, we experimentally demonstrate that our choices gives the best results. For R_k^1, we have only 2 variants. The first, depicted below in Eq. 1, is translating our main motivation which is enforcing frame prediction ambiguity, but without the conditional statement enforcing correct classification. The second version is the proposed reward function referred to as Eq.(6) in the main paper.

$$R_k^1 = \frac{1}{\|a_k^2\|} \sum_{t \in \mathcal{T}(a_k^2)} 1 - [p(\hat{y}_t | x_{t,k}; \theta) - p(\hat{y}'_t | x_{t,k}; \theta)] \quad (1)$$

In the top 2 rows of Tab. 1, 'No Cond.' corresponds to the reward function depicted in Eq. 1 and 'Cond.' to Eq.(6) in the main paper. Results are improving when the conditional statement is added to the reward function because we avoid feature augmentation resulting in wrong classification by the action segmentation model. We also have different version of the reward function R_k^2. We show that adding the conditional statements also increases the performances. From Eq.(7) in the main paper, we define by 'Cond. 1' the first condition $\sum_t a_{t,k}^2 m_{t,k} < \alpha$, and by 'Cond. 2' the second condition $\|a_k^2\| < \frac{T}{2}$. The 'No Cond.' corresponds to a reward function R_k^2 equals to only the positive part of Eq.(7) with no conditional statement at all.

As reflected in Tab. 1, the performances are lower when R_k^1 does not punish wrong classification by the model with a negative reward. It proves that this heuristic choice is determinant for the performance of the feature augmentation policy. Regarding R_k^2, we observe that adding either the first or second conditional statement by themselves leads to similar performances. Combining both conditions leads to a boost in performance. The best results are achieved when all conditional statement are present for both reward function.

Method	F1@10,25,50			Edit	MoF
Random	82.3	80.0	75.2	81.7	75.1
Constrained	**90.9**	**88.2**	**79.2**	**88.8**	**79.6**

Table 2. MS-TCN performance on GTAE for the random and constrained methods of constructing new training videos from generated transcripts, for 1.2× for the number of transcripts and 2× for the total number of videos in training

5. Construction of new training videos

Tab. 2 shows MS-TCN performance on GTA for two alternative methods of constructing new training videos from generated transcripts. For each action class visited sequentially along a given artificial transcript, the Random method randomly selects an instance of that action class from a real video, and copies all frames of the selected interval into the new video at the appropriate temporal location. MVGA uses the Constrained method to select the real video whose ground-truth transcript has the smallest edit distance to the new transcript, and copies all of the matched action intervals from the selected video to the new video; any remaining action classes present in the artificial transcript are constructed using the next closest video. Tab. 2 shows that our Constrained method used in MVGA is better.

References

[1] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019.