Markov Game Video Augmentation for Action Segmentation

Nicolas Aziere
Oregon State University
azieren@oregonstate.edu

Sinisa Todorovic
Oregon State University
sinisa@oregonstate.edu

Abstract

This paper addresses data augmentation for action segmentation. Our key novelty is that we augment the original training videos in the deep feature space, not in the visual spatiotemporal domain as done by previous work. For augmentation, we modify original deep features of video frames such that the resulting embeddings fall closer to the class decision boundaries. Also, we edit action sequences of the original training videos (a.k.a. transcripts) by inserting, deleting, and replacing actions such that the resulting transcripts are close in edit distance to the ground-truth ones. For our data augmentation we resort to reinforcement learning, instead of more common supervised learning, since we do not have access to reliable oracles which would provide supervision about the optimal data modifications in the deep feature space. For modifying frame embeddings, we use a meta-model formulated as a Markov Game with multiple self-interested agents. Also, new transcripts are generated using a fast, parameter-free Monte Carlo tree search. Our experiments show that the proposed data augmentation of the Breakfast, GTEA, and 50Salads datasets leads to significant performance gains of several state of the art action segmenters.

1. Introduction

This paper presents a new data augmentation framework for fully supervised action segmentation of untrimmed videos. Action segmentation is a basic vision problem. Despite recent tremendous advances in terms of new action segmenters and learning strategies, there is relatively slow progress in increasing the size of existing benchmark datasets. In comparison with peer datasets for action recognition or image classification, available benchmarks for action segmentation are significantly smaller. This presents challenges in training of recent action segmenters which show tendency to overfit on small datasets [11, 43, 40]. However, compiling large datasets is difficult, due to, in part, high costs of manual annotation of action segments.

We propose to augment existing datasets with newly generated video sequences, such that the resulting data augmentation enables more robust training and hence improves performance of action segmenters. Our approach is agnostic of a particular model for action segmentation, and expects that the segmenter has been pre-trained on the original training dataset to predict action classes of video frames.

Augmentation of video data has been mostly considered in the spatiotemporal, visual domain [38, 25, 19], where a human expert would heuristically specify the amount and type of data manipulation (e.g., subsampling, cropping, flipping of video frames) that are useful in training. While these approaches show great success, it is hard to formalize them in a principled manner. Others learn to generate new videos [9, 8, 42, 41], but the results are not sufficiently realistic yet, and hence would require domain adaptation if used for data augmentation in action segmentation.

Our key novelty is that we augment the original training videos directly in the deep feature space, unlike most previous work. As shown in Fig. 1 (top left), for augmentation, we modify original deep features of video frames at the input, such that the resulting embeddings fall closer to the class decision boundaries. Thus, by construction, we enforce that the augmented features be more challenging for learning, and in this way subsequently enable more robust training of the action segmenter. In addition, Fig. 1 (top right) illustrates that we also edit action sequences of the original training videos (a.k.a. transcripts) by inserting, deleting, and replacing actions, such that the resulting transcripts are close in edit distance to the ground-truth ones. Since the generated transcripts are kept similar to the originals, they are expected to be meaningful (i.e., legal) and provide a greater variety of legal action sequences than seen in the original training set. This is especially important for those application domains where some transcripts of interest are naturally rare and hence underrepresented in the original training dataset.

For the proposed augmentation of frame embeddings in the deep feature space, we specify a deep residual meta-model, as shown in Fig. 1 (bottom left). The meta-model takes deep features of frames at the input and predicts an optimal amount of feature modifications – i.e., offset fea-

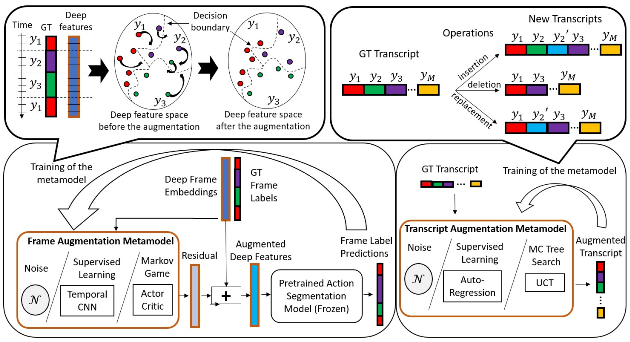

Figure 1. The proposed two-pronged data augmentation: (top and bottom left) Original deep features of frames at the input of a meta-model are modified to fall closer to class decision boundaries of the pretrained action segmenter. Alternative frameworks are considered for training the meta-model, including Gaussian noise, or supervised learning of a Temporal CNN, or reinforcement learning of an Actor-Critic network. (top and bottom right) The original action transcripts of training videos are modified by inserting, replacing, and removing actions, while ensuring a small edit distance between the newly generated transcripts and the originals. For training the meta-model for transcript augmentation, we consider supervised learning and alternatively a Monte Carlo tree search.

tures which are added to the original frame features, resulting in augmented embeddings. For training the meta-model, we consider both RL (Fig. 1 (bottom left)) and supervised learning. From our experiments, RL proves to be a more reliable paradigm.

Supervised learning of the proposed meta-model faces a fundamental challenge. As a direct consequence of performing data augmentation in the deep feature space, there is no reliable oracle which would provide supervision for the optimal amount of feature modifications. The only available oracle is the pre-trained action segmentation model which could predict action labels of the augmented frame embeddings, and the incurred loss could be used for training the meta-model. The action segmentation model provides only pseudo-labels for generated data – not ground-truth labels – and hence a fully-supervised training of our meta-model could suffer from noisy pseudo-labels.

An alternative is to resort to reinforcement learning (RL) due to the following advantages. First, it allows us to sequentially modify frame features, where previous modifications define a state in which RL estimates an optimal modification of the next frames. Learning of the proposed sequential feature augmentation is expected to be more reliable, especially for long video sequences, than learning how to optimally modify all frame embeddings at once. RL produces an optimal policy which would account for temporal dependence between frame embeddings, and hence make them suitable for action segmentation models with large temporal receptive fields (like MS-TCN or ASFormer). Second, RL is expected to provide for a more stable training of our meta-model by optimizing the *expected* reward over a policy of feature modifications, in comparison with the aforementioned minimization of the unreliable loss of the pretrained action segmenter on *particular* feature modifications. Third, RL is known to be very effective for problems with a large, continuous, output space, as is our case of predicting offset features in the deep feature space. Finally, RL is known to successfully address non-stationary environments with the distribution shift between training and test sets [37], which exactly characterizes our problem statement where data augmentation is aimed at bridging the distribution shift.

Within the RL framework, for modifying frame embeddings, we formulate the meta-model as a Deep Actor-Critic Network for learning policies of two self-interested agents in a Markov Game. Also, for generating new transcripts, we use a fast, parameter-free Monte Carlo tree search. We call our approach Markov Game Video Augmentation (MVGA).

Our experimental evaluation shows significant performance gains of recent convolutional and transformer-based action segmenters when our MVGA is used to augment the Breakfast, GTEA, and 50Salads datasets. Interestingly, MVGA enables the convolutional model MS-TCN [11, 27] achieve close performance to that of the significantly more complex (and more recent) ASFormer [43].

In the following, Sec. 2 reviews closely related work, Sec. 3 gives an overview of MVGA, Sec. 4 specifies our transcript augmentation, Sec. 5 formalizes our frame-feature augmentation, and Sec. 6 presents our results.

2. Related Work

This section reviews closely related work on fully supervised action segmentation and video data augmentation.

A family of temporal convolution models [10, 23, 24, 11, 40, 27, 34] have been studied for action segmentation. Some of these models use gradual temporal pooling for overcoming oversegmentation [10, 23, 24, 34], and others integrate reasoning about action boundaries [40, 18]. As representatives of this family, in our experiments, we consider MS-TCN [11, 27] and boundary-aware BCN [40] both of which consist of multiple stages of temporal convolution layers. Recent transformer-based models [43, 31, 39, 2] outperform temporal convolution models. Among these transformers, for evaluation, we consider ASFormer [43]. Both MS-TCN and ASFormer use standard I3D deep features [5]) of video frames as input. We also study a version of ASFormer with more informative frame embeddings [26].

Video augmentation in the visible space-time domain, such as, e.g., temporal cropping of frames [38, 25] or random cropping/removing/flipping of frames in a training mini-batch [19] have become standard practice in action recognition, person re-ID, and hand gesture recognition. Other data augmentation methods have also been used, including pooling frame features within a window of variable length [34]. For video augmentation, some approaches generate simulated videos with video-game engines [9, 8], or GANs [14, 42, 41, 33]. However, there is still a large domain gap between such simulated and real videos, which limits the utility of these methods for data augmentation.

Reinforcement learning has been used for addressing various vision problems, including 3D image segmentation [28, 32], image classification [29], object detection [3, 4], and tracking [6, 16]. To the best of our knowledge, video data augmentation for action segmentation has never been formulated within the reinforcement learning framework.

3. Overview of MVGA

In this and following two sections, we focus on our reinforcement learning formulation of the proposed data augmentation. The supervised learning formulation is described in Sec. 6. Fig. 2 shows an overview of our approach which consists of the following four steps. The first step pre-trains an action segmenter on a given training dataset $\mathcal{D}$. A training video of length T in $\mathcal{D}$ is given by its deep features for every frame $\mathcal{X} = \{x_t : t = 1, ..., T\}$, where $x_t \in \mathbb{R}^{d_{\text{in}}}$ ($d_{\text{in}} = 2048$ for standard I3D features [5]), and the ground-truth action classes $\mathcal{Y} = \{y_t : t = 1, ..., T\}$, where $y_t \in \mathbb{Y}$ and $\mathbb{Y}$ is a set of action classes. After pre-training, the action segmenter f_θ can be used to predict action classes of video frames $\hat{\mathcal{Y}} = f(\mathcal{X}; \theta)$.

The second step generates new transcripts and their corresponding videos. We use the UCT algorithm [20] to first efficiently construct a tree, whose paths from the root to leaves represent legal transcripts, and then select optimal paths. For every new transcript, a new video is constructed by copying appropriate action segments from real videos in $\mathcal{D}$ to the new video following the action sequence of the new transcript.

The third step augments features of the original and new videos with the Actor and Critic networks, which gives the augmented dataset $\mathcal{D}'$. The Actor predicts how much and where to modify features in the input video. The Critic estimates the expected rewards for the Actor. Both Actor and Critic are learned using the pre-trained f_θ as oracle.

The fourth step fine-tunes f_θ such that every mini-batch consists of videos from both $\mathcal{D}$ and $\mathcal{D}'$.

In the following, we specify the second and third steps.

4. New Transcript and Video Generation

From $\mathcal{D}$, we generate new transcripts and their corresponding new videos. The new transcripts should be semantically meaningful, i.e., legal. This is enforced by requiring that the new transcripts have: (i) similar lengths as the original transcripts; and (ii) high likelihoods of consecutive pairs of action classes. Our experiments suggest that auto-regressive models – e.g., a recurrent-neural network (RNN) or Transformer network [35] – provide poor transcript generation, since they require large training datasets and hence have limited utility for our target settings where data augmentation is needed to address lack of data. Therefore, in this section, we focus on an alternative framework – parameter-free Monte-Carlo Tree Search (MTCS), where the space of transcripts is efficiently represented by a tree. We first construct the tree, and then identify its optimal path.

4.1. MTCS for New Transcript Generation

In the tree of transcripts, the root represents the dummy "start" of action sequences. The root's descendants sequentially add action classes to the transcripts until leaf nodes, which represent the dummy "end". A node v represents the last action class of the path π_v from the root to v, $\pi_v = \{\text{"start"}, y_1, ..., y_v\}$, where $y_v \in \mathbb{Y}$ or $y_v = \text{"end"}$ if v is a leaf node. Each node is assigned a weight, $w(v)$, specified as the joint likelihood of the corresponding path:

$$w(v) = p(|\pi_v|) \prod_{(u,u')\in\pi_v} p(y_{u'}|y_u), \tag{1}$$

where $p(|\pi_v|)$ is a prior of the transcript length, u' is a child of u along π_v, and $p(y_{u'}|y_u)$ denotes the transition probability of consecutive actions. In the special case, $p(y_{u'}|\text{"start"})$ and $p(\text{"end"}|y_u)$ represent the priors that the transcript begins and ends with classes $y_{u'}, y_u \in \mathbb{Y}$. Both $p(|\pi_v|)$ and $p(y_{u'}|y_u)$ are estimated from $\mathcal{D}$. For $p(|\pi_v|)$ we learn the Poisson distribution, and for $p(y_{u'}|y_u)$ we estimate the frequency of the corresponding class transitions in $\mathcal{D}$.

The tree is constructed iteratively using the well-known UCT algorithm [20] which balances a trade-off between exploitation and exploration. In each iteration, the root is sequentially expanded with a path of optimal descendants until the dummy "end" leaf or the maximum tree depth is reached. We do not allow illegal expansions, i.e., a path is guaranteed to consist of class transitions seen in $\mathcal{D}$. Suppose a path has reached node u without meeting the stopping criterion. Then, UCT adds to the path the optimal node u' from a subset of children, $\text{ch}(u) \subset \text{Ch}(u) = \{u' : p(y_{u'}|y_u) > 0\}$:

$$u' = \arg\max_{v\in\text{ch}(u)} \left[w(v) + c\sqrt{\frac{2\log n(u)}{n(v)}} \right] \tag{2}$$

where $\text{ch}(u)$ is randomly sampled 50% of $\text{Ch}(u)$ to enable exploring alternative paths; $w(v)$ is given by (1); $c = \frac{1}{\sqrt{2}}$ is the exploration-exploitation trade-off parameter; and $n(v)$ is the number of paths in the current tree that include node v. As the tree iteratively grows, the value of $n(v)$ keeps changing for every node, which enables exploring less visited nodes in the space of transcripts even if $w(v)$ is small.

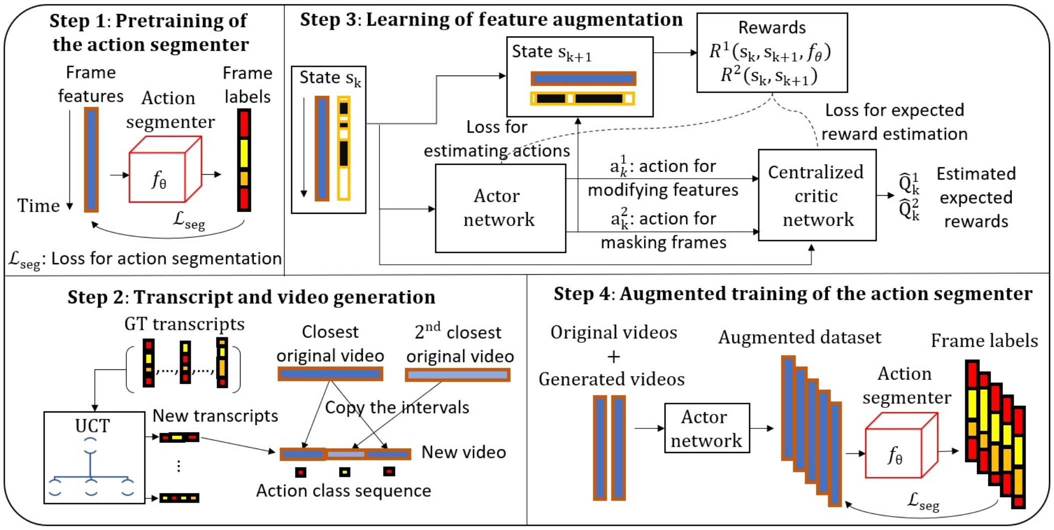

Figure 2. Our MVGA consists of four steps. 1–An action segmentation model is pre-trained using the original training set. 2–New transcripts are generated by selecting optimal paths in a UCT tree [20] of legal transcripts. For every new transcript, a new video is constructed by copying instances of action classes in the new transcript from real videos. 3- Features of the original and constructed new videos are augmented with the Actor and Critic networks. The Actor predicts the amount and location of feature modifications in the video. The Critic estimates the expected rewards for the Actor's two predictions. Both Actor and Critic are learned using the pre-trained action segmenter as oracle. 4- The action segmenter is fine-tuned on the original and augmented training videos.

After 1000 tree-growing iterations, the node v^* with the highest likelihood $v^* = \arg\max_v w(v)$ in the tree uniquely identifies the newly generated transcript π_{v^*}. For generating another transcript, we construct another tree anew.

4.2. Generating New Videos of New Transcripts

Given a new transcript, π, we sequentially construct a new video with deep features (e.g., I3D [5]) following the ordering of action classes in π. It is worth emphasizing that our video generation occurs directly in the deep feature space. We begin by selecting a video $\mathcal{X}_0 \in \mathcal{D}$ whose ground-truth transcript has the smallest edit distance to π. From $\mathcal{X}_0$, we remove all action segments that are not represented in π, resulting in our initial new video $\mathcal{X}'_0$. For action classes that are present in π but missing in $\mathcal{X}'_0$, we identify the second closest video $\mathcal{X}_1 \in \mathcal{D}$ with these missing actions, and copy their respective temporal intervals to the appropriate locations in the new video, resulting in $\mathcal{X}'_1$. This is repeated until the entire π is fully represented by $\mathcal{X}'_g$, $g = 0, 1, 2, \ldots$. Note that $\mathcal{X}'_g$ keeps the original lengths of action instances found in real videos, which ensures temporal coherence of every action in $\mathcal{X}'_g$. A pseudo-code of the proposed video generation is given in the supplement.

5. Augmentation of Frame Features

Features of both original and new videos are augmented by a Markov Game (MG) [15, 30, 37]. Since MG is a well-studied framework and we do not claim novelty in our particular formulation, below we rely that the reader is already familiar with the motivation and main concepts of MG. Our MG consists of two agents which sequentially take independent actions causing state changes of a fully observable environment. The environment at step k is defined by state $s_k = (\mathcal{X}_k, \mathcal{M}_k)$, where $\mathcal{X}_k = \{x_{k,t} : t = 1, \ldots, T\}$ is the set of current video frame features, and $\mathcal{M}_k = \{m_{k,t} : t = 1, \ldots, T\}$ is a binary mask assigned to the video frames, $m_{k,t} \in \{0, 1\}$, for keeping the record which frames have been already modified. Given s_k, the two agents follow their respective policies, μ^1 and μ^2, to take actions $a_k^1 = \mu^1(s_k) \in \mathcal{A}^1$ and $a_k^2 = \mu^2(s_k) \in \mathcal{A}^2$. $\mathcal{A}^1$ is a continuous action space of agent 1, where $a_k^1 = \{a_{t,k}^1 \in \mathbb{R}^{d_{\text{in}}} : t = 1, \ldots, T\}$ represents the amount of feature augmentation, i.e., offset features. $\mathcal{A}^2$ is a discrete action space of agent 2 for selecting video frames for augmentation, $a_k^2 \in \{0, 1\}^T$, where $a_{t,k}^2 = 0$ means that the frame t will not be augmented in step k. a_k^1 and a_k^2 cause the environment to change to next state s_{k+1}, which incurs the respective two rewards $R_k^1 = R^1(s_k, s_{k+1}, f_\theta)$ and $R_k^2 = R^2(s_k, a_2^k)$. Our goal is to learn μ^1 and μ^2 so as to maximize the agents' action-value functions given by:

$$Q^i(s, a) = \mathbb{E}\Big[\sum_{k \geq 0} (\gamma)^k R_k^i \Big| \mu^i, a_0^i{=}a, s_0{=}s\Big],\ i{=}\{1, 2\} \tag{3}$$

where $\mathbb{E}[\cdot]$ denotes expected value, $\gamma = 0.99$ is the discount factor raised to the power of k, $s \in \mathcal{S}$, and $a \in \mathcal{A}^i$.

For a given video, MG starts from the initial state $s_0 = (\mathcal{X}_0, \mathcal{M}_0)$, with the original frame features and all-zero mask, $\mathcal{X}_0 = \mathcal{X}$ and $\mathcal{M}_0 = \{0\}^T$. In state s_k, the two agents take their respective actions, which gives s_{k+1} specified as

$$\mathcal{X}_{k+1} = \{x_{t,k} + a^2_{t,k}(a^1_{t,k} + \epsilon) : t = 1, ..., T\}, \quad (4)$$

$$\mathcal{M}_{k+1} = \mathcal{M}_k \vee a^2_k, \quad (5)$$

where $\vee$ is the logical OR operator, and $\epsilon \in \mathbb{R}^{d_{\text{in}}}$ is noise sampled from the zero-mean and unit-variance Gaussian distribution. ϵ enables an exploration of $\mathcal{A}^1$ and provides a way to generate multiple, distinct, augmented features from the same initial state s_0. From (4), only frames selected by a^2_k get updated with the corresponding offset features $(a^1_k + \epsilon)$. Also, from (5), mask $\mathcal{M}_k$ keeps the record of previously selected frames for augmentation.

MG stops as soon as one of the following happens (experimentally optimized): after $k = 10$ steps, or when 95% of the video has been augmented, $\sum_{t=1}^{T} m_{t,k} \geq 0.95T$.

5.1. Two Rewards for Feature Augmentation

The policy of agent 1, μ^1, is learned to augment original features, such that they become more challenging for the pre-trained action segmenter, f_θ, and thus provide for a more robust subsequent training of f_θ on the augmented training dataset. This is enforced by specifying the following reward $R^1_k = R^1(s_k, s_{k+1}, f_\theta)$. Let $\hat{y}_t$ and $\hat{y}'_t$ denote the top two scoring class predictions for $x_{t,k}$ by f_θ, $\hat{y}_t = \arg\max_{y\in\mathbb{Y}} p(y|x_{t,k};\theta)$ and $\hat{y}'_t = \arg\max_{y\in\mathbb{Y}\setminus\{\hat{y}_t\}} p(y|x_{t,k};\theta)$. We penalize agent 1 with a negative reward whenever a^1_k causes f_θ to make a wrong prediction, $\hat{y}_t \neq y_t$. We assign a positive reward to agent 1 when f_θ's prediction is equal to the ground truth, $\hat{y}_t = y_t$, and reduce this positive reward if the feature augmentation is not challenging enough for f_θ. The positive reward reduction is proportional to f_θ's confidence in its prediction, specified as a difference between its top two scoring predictions, $\kappa = [p(\hat{y}_t|x_{t,k};\theta) - p(\hat{y}'_t|x_{j,k};\theta)]$, $0 < \kappa \leq 1$. Hence, when confidence κ is large and close to 1, the positive reward for agent 1 is maximally reduced as

$$R^1_k = \frac{1}{\|a^2_k\|} \sum_{t\in\mathcal{T}(a^2_k)} r^1(x_{t,k+1}, f_\theta),$$

$$r^1(x_{t,k}, f_\theta) = \begin{cases} 1-[p(\hat{y}_t|x_{t,k};\theta)-p(\hat{y}'_t|x_{t,k};\theta)] \text{ , if } \hat{y}_t = y_t \\ -1 \qquad\qquad\qquad\qquad\qquad\qquad \text{ , if } \hat{y}_t \neq y_t \end{cases} \quad (6)$$

where $\mathcal{T}(a^2_k)$ returns the set of frames where $a^2_{t,k}$=1.

For the policy of agent 2, μ^2, we specify the following three requirements: (i) μ^2 should not select frames which have already been augmented in the previous MG steps; (ii) frame selection should be local at each MG step and focus only on a very few action instances in the video (and in this way make learning of μ^1 easier); and (iii) selected temporal intervals should maximally overlap the ground-truth action instances. These three requirements are enforced by specifying the following reward:

$$R^2_k = \begin{cases} \sum_\tau \frac{|\tau \cap a^2_k|}{|\tau|} \text{ , if } \sum_t a^2_{t,k} m_{t,k} < \alpha \text{ and } \|a^2_k\| < \frac{T}{2} \\ -1 \text{ , otherwise} \end{cases} \quad (7)$$

τ is the ground-truth mask of actions over frames, and enforces that each iteration of the video augmentation covers the entire time interval of an action rather than randomly scattered frames. $\alpha = 0.4T$ is an experimentally optimized threshold which allows a flexible frame selection of up to 40% of the past $\mathcal{M}_k$. From (7), agent 2 is penalized when selects more than a half of the video for augmentation.

5.2. Learning the Policies for Feature Augmentation

To learn μ^1 and μ^2, we design an Actor-Critic model with a *decentralized-actor* deep network for computing the agents' actions, and a *centralized-critic* deep network for efficiently estimating action-value functions $\hat{Q}^1$ and $\hat{Q}^2$, as shown in Fig. 3. The Actor-Critic framework has been demonstrated effective for continuous agent-action spaces and when the agents have individual rewards [17], as in our case. In general, decentralizing the actor network helps increase stationarity of environments with self-interested agents [37]. The centralized critic network is suitable because it allows relevant information from all of the agents to be shared toward estimating each agent's expected reward.

As can be seen in Fig. 3, the input to the actor network, $s_k = (\mathcal{X}_k, \mathcal{M}_k)$, is passed through a backbone, specified as a multi-stage temporal convolutional neural network (TCN) [11], in order to estimate the latent (deep) representation of s_k. Then, two distinct 1-stage TCNs take this latent representation as input and compute the frame selection

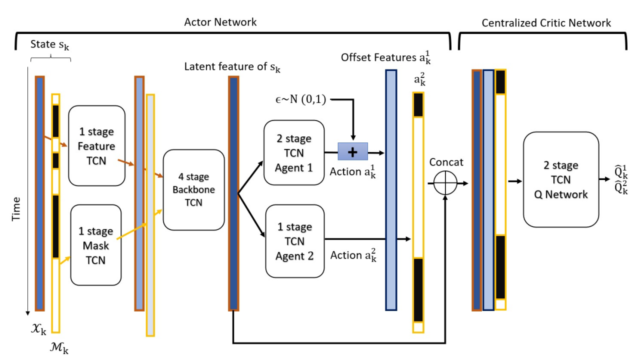

Figure 3. Actor-Critic model for learning policies μ^1 and μ^2 consists of several stages of temporal convolution network (TCN). The actor network predicts feature augmentation, and the centralized critic estimates the expected rewards for each policy.

a_2^k and feature offsets of the selected frames a_k^1. Finally, the predicted a_k^1, a_2^k, and latent feature of s_k are passed to the centralized critic – specifically, a 2-stage TCN – to estimate the expected rewards $\hat{Q}_k^1 = \hat{Q}^1(s_k, a_k^1, a_k^2)$ and $\hat{Q}_k^2 = \hat{Q}^2(s_k, a_k^1, a_k^2)$ of μ^1 and μ^2.

Parameters of the proposed actor-critic deep architecture are learned using the standard temporal difference learning [21]. The critic learns to iteratively simulate the action-value function, $\hat{Q}_k^i, i = \{1, 2\}$, which is later used to update parameters of the actor network. To this end, we minimize the following loss of the critic network:

$$L_C = \sum_{i \in \{1,2\}} \left(R_k^i + \gamma \hat{Q}_{k+1}^i - \hat{Q}_k^i \right)^2, \tag{8}$$

where R_k^i is given by (6) and (7), and γ is the discount factor. The actor network is learned with the following loss:

$$L_A^i = \|a_k^i\|_2^2 \cdot \left(R_k^i + \gamma \hat{Q}_{k+1}^i - \hat{Q}_k^i \right)^2, \; i = \{1, 2\}. \tag{9}$$

A proof in [37] shows that minimizing L_C and L_A in (8) and (9) optimizes μ^1 and μ^2 so they manage to achieve maximal rewards R^1 and R^2 given by (6) and (7), as desired.

6. Results

Datasets include three benchmarks: Breakfast [22], GTEA [12], and 50Salads [36]. For each dataset, we precompute the standard I3D frame features without any self-supervision, fine-tuning, or data augmentation, as in [11]. *Breakfast* has 1712 videos with 48 action classes, *GTEA* shows 7 complex activities, each specified in terms of 11 action classes including the background class. *50Salads* consists of 50 videos with 17 action classes. For Breakfast, GTEA, and 50Salads, we perform the standard 4-fold, 4-fold, and 5-fold cross validation, respectively.

Metrics. Mean-of-Frame (MoF) is the average frame-wise classification accuracy. Edit score counts edit operations to make the predicted and ground-truth sequences equivalent. F1 score counts true positives when a temporal intersection between the predicted and ground-truth segments is 10%, 25% and 50%, denoted as F1@10,25,50.

Implementation details. *The Actor-Critic* has the same layers as MS-TCN [11]. One stage consists of 10 convolution layers with an increasing dilation rate and feature maps with size 128. The mask backbone and feature backbone each represents a 1-stage TCN. Their outputs are, first, concatenated, then, passed to a 4-stage MS-TCN, and, finally, input to the agents' heads. Agent 1's head is a 2-stage MS-TCN. Agent 2's head is a 1-stage TCN. The critic network is a 2-stage MS-TCN. The learning rate for the actor-critic training is 0.0001. Our training of the meta-model is run for 30 epochs. The NMS threshold is set to 0.5, and the minimum IOU is set to 0. The discount factor $\gamma = 0.99$. The policy training on Tesla-V100s for GTEA takes 3 hours. *The MCTS algorithm* runs 1000 node expansions in a couple of seconds. The maximum tree width is 200. The maximum tree depth is $\hat{\lambda} + 2\sqrt{\hat{\lambda}}$ where $\hat{\lambda}$ is the average transcript length of the training set. The number of epochs in the fine-tuning on the original and augmented data is 50 epochs for MS-TCN, 100 epochs for ASFormer, and 50 epochs for BCN. The meta-training of the meta-model increases complexity compared to baseline approaches. It is important to separate the time for training an action segmentation model, and the time for our meta-training. Our meta-training can be viewed as a part of the dataset preparation, which takes significantly less time than manually collecting and annotating real videos.

# Transcripts	Train. set size	F1@10,25,50			Edit	MoF
1×	1×	87.3	84.9	72.4	82.0	78.0
1.1×	1×	85.2	81.1	69.7	80.0	74.9
1×	2×	89.8	88.5	77.6	86.8	79.5
1.1×	2×	90.5	88.6	77.9	87.1	78.9
1.2×	3×	**90.9**	**88.2**	**79.2**	**88.8**	**79.6**
1.2×	4×	89.1	87.7	78.9	89.0	79.7
1.2×	5×	89.0	87.0	75.6	85.7	78.2

Table 1. MS-TCN performance on GTEA as a function of the number of training transcripts, and the number of original and augmented videos used for training –both expressed as the increase factor of the original training set. When our augmentation triples the set of training videos such that there are 20% of newly generated transcripts, MS-TCN achieves the best performance. The top row is without any augmentation, and the second from the top row is for only feature augmentation without new transcripts.

6.1. Ablation Studies of MS-TCN on GTEA

The training dataset size. Tab. 1 reports how MS-TCN performance on GTEA changes as a function of two variables: (i) the number of training transcripts; and (ii) the total number of original and augmented videos – both expressed as the increase factor of the original set. Tab. 1 shows that when the original set of training videos is tripled such that there are 20% of new transcripts, MS-TCN achieves the best performance for all of the metrics, and this setting is used in the sequel. Too many augmentations using the same input video leads to similar data outputs, and consequently to overfitting in training of an action segmentation model.Note that testing the transcript augmentation as a standalone contribution is inappropriate and reduces performance (second row), since new transcripts need to be realized as new videos. These new videos are Frankenstein-like constructs from multiple real videos, and hence have highly unrealistic action transitions, and incoherent and sometimes impossible human motions. Therefore, the new videos require additional frame-feature augmentation to provide useful data augmentation.

Alternatives for generating transcripts. Tab. 2 com-

Method	Edit dist. (%) Top 10%	Edit dist. (%) Top 20%	Likelihood Top 20%
Auto-regression	19.5	22.0	0.59e-4
UCT	**14.0**	**15.2**	**1.41e-4**
Ground truth	-	-	1.45e-4

Table 2. Comparison of two alternative transcript generation methods – auto-regression by a Transformer Network [35], and our proposed UCT – in terms of an average edit distance between the generated and ground-truth transcripts for GTEA, and their average likelihood given by (1). The two metrics are averaged over the top 10% and 20% of generated transcripts closest to the ground-truth.

Frame Augmentation	F1@10,25,50			Edit	MoF
No augmentation [11]	85.8	83.4	69.8	79.0	76.3
Noise	85.7	83.2	71.9	79.6	77.5
Non-RL (Seq)	83.1	80.9	69.1	76.2	76.5
Non-RL (All)	83.9	81.5	70.4	77.6	77.7
Our RL-based	**90.9**	**88.2**	**79.2**	**88.8**	**79.6**

Table 3. MS-TCN performance on GTAE, when using the best data augmentation setting given in Tab. 1, and for alternative strategies of frame-feature augmentation. Supervised learning in "Non-RL" decreases the performance of MS-TCN.

Policies	F1@10,25,50			Edit	MoF
$\mu_1/10$	88.5	85.0	74.7	83.8	78.2
$\mu_1/20$	87.4	84.8	74.1	84.5	77.8
μ_1/μ_2	**90.9**	**88.2**	**79.2**	**88.8**	**79.6**

Table 4. MS-TCN performance on GTAE for the proposed multi-agent (μ_1/μ_2) and the alternative single-agent ($\mu_1/10$ or $\mu_1/20$).

pares two alternative methods for generating transcripts – namely, auto-regression and our UCT – in terms of: (i) edit distance to the ground-truth transcripts, and (ii) expected reward given by (1). The two metrics are averaged over the top 10% and 20% of generated transcripts that are the closest to the ground-truth transcripts. The auto-regressive model is a Transformer Network [35] trained with the cross-entropy loss to predict the next action class given an input sequence of action classes. From Tab. 2, our UCT generates transcripts which are more similar to the ground-truth ones than those produced by auto-regression. Also, the average likelihoods for UCT and ground truth are nearly the same.

RL vs Non-RL for frame-feature augmentation. Tab. 3 compares the proposed reinforcement learning and an alternative supervised learning for frame-feature augmentation, when using the best data augmentation setting given in Tab. 1. We consider several baselines. "Noise" refers to simply adding Gaussian noise with zero mean and 0.01 variance to the original deep features of frames, as the original I3D features have an average value of 0.2 and variance 0.06. From Tab. 3, such a frame-feature augmentation with Gaussian noise barely has any effect on MS-TCN's performance. "Non-RL" is another baseline that uses supervised learning for training the meta-model, as depicted in Fig. 1 (bottom left). In "Non-RL" the meta-model is a 1-stage TCN which outputs the offset feature for a given original feature at the input. As shown in Fig. 1 (bottom left), the resulting augmented feature is then passed to the pre-trained action segmenter. The meta-model is trained based on the cross-entropy loss of the pre-trained action segmenter, which serves as an oracle to ensure that the augmented feature remains in the action class of the original frame feature. For "Non-RL", we also regularize the resulting augmented feature to be different from the original feature with the standard contrastive loss. Moreover, we study two versions of "Non-RL" – one where all features of the entire video are modified at once, called "Non-RL (All)"; and the other with a sequential modification of frame features one action segment at a time, called "Non-RL (Seq)". From Tab. 3, both "Non-RL (All)" and "Non-RL (Seq)" decrease MS-TCN's performance. More details on "Non-RL" are presented in the supplement.

Class transition	$2,5$	$2,7$	$5,9$	$8,10$	$10,4$
# Original dataset	1	1	2	3	3
# Augmented dataset	2	2	3	5	5

Table 5. Count of the least represented action transitions in the original and augmented training sets of GTEA.

Policy. Tab. 4 compares MS-TCN performance on GTAE for our multi-agent formulation and an alternative single-agent feature augmentation. The latter does not learn frame masking, and hence does not have our agent 2. The single-agent, denoted as $\mu_1/10$ or $\mu_1/20$, uses only policy μ_1 for augmenting frame features. The values 10 and 20 indicate the percent of video randomly selected for augmentation at each policy iteration. Our multi-agent, denoted as μ_1/μ_2, uses both μ_1 and μ_2 policies. Tab. 4 shows that adding the second agent significantly improves the results.

Improving transcripts variability. Generating new transcripts modify the total count of action transition initially present in the original set of transcripts. The advantage of our MCTS based generation approach is the emphasis on exploration. The number of poorly represented action transitions are increased with the new transcripts in training. Tab. 5 shows some example action transitions that are the least represented in the original training set get at least one new instance when adding the generated transcripts.

6.2. Impact of MVGA on SOTA

The top of Tab. 6 reports results of state-of-the-art (SOTA) fully supervised methods that do not use video augmentation in training (some of them use sophisticated post-processing, which we do not have). Tab. 6 also shows how MVGA affects performance of SOTA approaches – including: MS-TCN[11], BCN [40], ASFormer [43], and BridgePrompt [26] – on Breakfast, GTAE, and 50Salads. As can be seen, for all of the four SOTA approaches, data augmen-

Dataset	Breakfast					GTEA					50Salads				
Method	F1@10,25,50			Edit	MoF	F1@10,25,50			Edit	MoF	F1@10,25,50			Edit	MoF
MSTCN++ [27]	64.1	58.6	45.9	65.6	67.6	87.8	86.2	74.4	82.6	78.9	80.7	78.5	70.1	74.3	83.7
ASRF [18]	74.3	68.9	56.1	72.4	67.6	89.4	87.8	79.8	83.7	77.3	84.9	83.5	77.3	79.3	84.5
HASR [1]	74.7	69.5	57.0	71.9	69.4	89.2	87.2	74.8	84.5	76.9	86.6	85.7	78.5	81.0	83.9
SSTDA [7]	75.0	69.1	55.2	73.7	70.2	90.0	89.1	78.0	86.2	79.8	83.0	81.5	73.8	75.8	83.2
G2L [13]	74.9	69.0	55.2	73.3	70.7	89.9	87.3	75.8	84.6	78.5	80.3	78.0	69.8	73.4	82.2
UVAST [2]	76.7	70.0	56.6	68.2	86.2	77.1	69.7	54.2	90.5	62.2	81.2	70.4	83.9	77.1	69.7
MSTCN [11]	52.6	48.1	37.9	61.7	66.3	85.8	83.4	69.8	79.0	76.3	76.3	74.0	64.5	67.9	80.7
MSTCN (Ours)	58.1	54.6	44.4	63.4	66.8	87.3	84.9	72.4	82.0	78.0	75.8	73.8	66.8	68.2	82.7
MSTCN + MVGA	68.9	66.1	55.8	68.1	71.1	90.9	88.8	79.2	88.2	79.6	80.1	77.9	71.2	74.1	83.2
BCN [40]	68.7	65.5	55.0	66.2	70.4	88.5	87.1	77.3	84.4	79.8	82.3	81.3	74.0	74.3	84.4
BCN + MVGA	70.8	67.7	57.3	68.3	71.8	91.3	90.0	80.6	86.7	80.5	83.7	82.4	75.8	77.2	85.6
ASFormer [43]	76.0	70.6	57.4	75.0	73.5	90.1	88.8	79.2	84.6	79.7	85.1	83.4	76.0	79.6	85.6
ASFormer + MVGA	**75.6**	**72.1**	**59.7**	**76.8**	**74.2**	91.3	90.5	79.3	86.4	80.3	86.3	84.9	78.7	81.3	86.0
Bridge-Prompt [26]						94.1	92.0	83.0	91.6	81.2	89.2	87.8	81.3	83.8	88.1
Bridge-Prompt + MVGA						**95.2**	**93.4**	**84.8**	**92.1**	**82.6**	**90.8**	**89.4**	**83.2**	**86.1**	**88.9**

Table 6. Impact of MVGA on SOTA methods on Breakfast, GTEA and 50Salads, and comparison with the other SOTA approaches that do not use data augmentation.

Dataset	Breakfast					GTEA					50Salads				
Method	F1@10,25,50			Edit	MoF	F1@10,25,50			Edit	MoF	F1@10,25,50			Edit	MoF
MSTCN + C2F[34]	58.2	55.0	47.8	62.9	66.5	86.2	83.5	78.1	78.8	76.5	77.3	74.2	65.5	67.3	80.8
MSTCN + Speed	55.0	52.7	41.9	60.1	65.9	84.8	80.8	66.1	78.8	75.7	77.3	75.9	67.8	68.8	81.2
MSTCN + MVGA	**68.9**	**66.1**	**55.8**	**68.1**	**71.1**	**90.9**	**88.8**	**79.2**	**88.2**	**79.6**	**80.1**	**77.9**	**71.2**	**74.1**	**83.2**
ASFormer + C2F [34]	75.8	72.3	57.5	75.2	72.9	90.3	88.0	78.8	84.2	78.9	84.5	82.6	76.1	78.7	84.0
ASFormer + Speed	75.2	69.1	56.9	74.8	71.9	89.3	87.2	77.7	85.2	78.5	84.4	82.9	75.1	77.2	85.1
ASFormer + MVGA	**75.6**	**72.1**	**59.7**	**76.8**	**74.2**	**91.3**	**90.5**	**79.3**	**86.4**	**80.3**	**86.3**	**84.9**	**78.7**	**81.3**	**86.0**

Table 7. Comparison of different data augmentation methods on Breakfast, GTEA and 50Salads.

tation by MVGA consistently produces performance gains relative to their respective performance without data augmentation. For instance, MSTCN+MVGA outpeforms both MSTCN [11] and its more advanced relative MSTCN++ [27] without data augmentation. MVGA also improves the original ASFormer [43] producing the new state of the art performance on Breakfast. Applying MVGA on the features provided by Bridge-Prompt [26] and following the same training procedure as for the ASFormer gives the method Bridge-Prompt+MVGA with the best results on GTAE and 50Salads. In Tab. 6, we could not report Bridge-Prompt+MVGA performance on Breakfast, since the Bridge-Prompt model for action segmentation on Breakfast is not publicly available, and our own training of Bridge-Prompt on Breakfast gave poor results.

Video augmentation has been considered in the context of self-supervised learning (typically as a part of pretraining). However, a comparison with such methods is beyond our scope, because MVGA directly allows for the unified fully supervised training on a larger, augmented training dataset. In Tab. 7, we compare MVGA with some baseline video augmentation methods which also directly enable fully supervised learning, including Speed and C2F [34]. The former randomly selects certain action instances in each original training video, and performs “speeding-up” (i.e., frame subsampling) or “slowing-down” (i.e., frame upsampling with interpolation) of their temporal intervals, and thus doubles the original training set. C2F randomly selects certain time intervals in every original training video. Then, for each selected interval, C2F assigns the dominant ground-truth action class with the largest temporal support in that interval, and also replaces all frame features within the interval with their max-pooled feature. As can be seen in Tab. 7, MVGA provides for better data augmentation for MS-TCN and ASFormer than Speed and C2F.

7. Conclusion

We have specified video data augmentation for action segmentation within the reinforcement learning framework. Our approach generates new action transcripts and their corresponding new videos, as well as modifies the feature embedding of video frames. The transcript generation uses the efficient Monte-Carlo Tree Search to produce new, legal, high-likelihood action sequences. Optimal amount and temporal locations of feature changes in the video are learned with a two-agent actor-critic network. Our video augmentation in training of representative, state-of-the-art, convolutional and transformer-based action segmenters leads to their significant performance gains on the benchmark Breakfast, GTEA, and 50salads datasets, in comparison to their original training without video augmentation.

Acknowledgments This work was supported in part by DARPA MCS Award N66001-19-2-4035.

References

[1] Hyemin Ahn and Dongheui Lee. Refining action segmentation with hierarchical video representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16302–16310, 2021.

[2] 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. *arXiv preprint arXiv:2209.00638*, 2022.

[3] Míriam Bellver Bueno, Xavier Giró-i Nieto, Ferran Marqués, and Jordi Torres. Hierarchical object detection with deep reinforcement learning. *Deep Learning for Image Processing Applications*, 31(164):3, 2017.

[4] Juan C Caicedo and Svetlana Lazebnik. Active object localization with deep reinforcement learning. In *Proceedings of the IEEE international conference on computer vision*, pages 2488–2496, 2015.

[5] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

[6] Boyu Chen, Dong Wang, Peixia Li, Shuang Wang, and Huchuan Lu. Real-time'actor-critic'tracking. In *Proceedings of the European conference on computer vision (ECCV)*, pages 318–334, 2018.

[7] Min-Hung Chen, Baopu Li, Yingze Bao, Ghassan AlRegib, and Zsolt Kira. Action segmentation with joint self-supervised temporal domain adaptation. In *CVPR*, 2020.

[8] César Roberto de Souza, Adrien Gaidon, Yohann Cabon, Naila Murray, and Antonio Manuel López. Generating human action videos by coupling 3d game engines and probabilistic graphical models. *International Journal of Computer Vision*, 128(5):1505–1536, 2020.

[9] César Roberto de Souza12, Adrien Gaidon, Yohann Cabon, and Antonio Manuel López. Procedural generation of videos to train deep action recognition networks. 2017.

[10] Li Ding and Chenliang Xu. Weakly-supervised action segmentation with iterative soft boundary assignment. In *CVPR*, June 2018.

[11] Yazan Abu Farha and Jurgen Gall. MS-TCN: Multi-stage temporal convolutional network for action segmentation. In *CVPR*, 2019.

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

[13] Shang-Hua Gao, Qi Han, Zhong-Yu Li, Pai Peng, Liang Wang, and Ming-Ming Cheng. Global2local: Efficient structure search for video action segmentation. In *CVPR*, 2021.

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[15] Junling Hu and Michael P Wellman. Nash q-learning for general-sum stochastic games. *Journal of machine learning research*, 4(Nov):1039–1069, 2003.

[16] Chen Huang, Simon Lucey, and Deva Ramanan. Learning policies for adaptive tracking with deep feature cascades. In *Proceedings of the IEEE international conference on computer vision*, pages 105–114, 2017.

[17] Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In *International conference on machine learning*, pages 2961–2970. PMLR, 2019.

[18] Yuchi Ishikawa, Seito Kasai, Yoshimitsu Aoki, and Hirokatsu Kataoka. Alleviating over-segmentation errors by detecting action boundaries. In *WACV*, 2021.

[19] Takashi Isobe, Jian Han, Fang Zhuz, Yali Liy, and Shengjin Wang. Intra-clip aggregation for video person re-identification. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 2336–2340. IEEE, 2020.

[20] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[21] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[22] Hilde Kuehne, Ali Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *CVPR*, 2014.

[23] Colin Lea, Michael D. Flynn, Rene Vidal, Austin Reiter, and Gregory D. Hager. Temporal convolutional networks for action segmentation and detection. In *CVPR*, 2017.

[24] Peng Lei and Sinisa Todorovic. Temporal deformable residual networks for action segmentation in videos. In *CVPR*, 2018.

[25] Jie Li, Mingqiang Yang, Yupeng Liu, Yanyan Wang, Qinghe Zheng, and Deqiang Wang. Dynamic hand gesture recognition using multi-direction 3d convolutional neural networks. *Engineering Letters*, 27(3), 2019.

[26] Muheng Li, Lei Chen, Yueqi Duan, Zhilan Hu, Jianjiang Feng, Jie Zhou, and Jiwen Lu. Bridge-prompt: Towards ordinal action understanding in instructional videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19880–19889, 2022.

[27] Shi-Jie Li, Yazan AbuFarha, Yun Liu, Ming-Ming Cheng, and Juergen Gall. MS-TCN++: Multi-stage temporal convolutional network for action segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[28] Xuan Liao, Wenhao Li, Qisen Xu, Xiangfeng Wang, Bo Jin, Xiaoyun Zhang, Yanfeng Wang, and Ya Zhang. Iteratively-refined interactive 3d medical image segmentation with multi-agent reinforcement learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9394–9402, 2020.

[29] Shiqi Lin, Tao Yu, Ruoyu Feng, Xin Li, Xin Jin, and Zhibo Chen. Local patch autoaugment with multi-agent collaboration. *arXiv preprint arXiv:2103.11099*, 2021.

[30] Michael L Littman et al. Friend-or-foe q-learning in general-sum games. In *ICML*, volume 1, pages 322–328, 2001.

[31] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. *arXiv preprint arXiv:2106.13230*, 2021.

[32] Chaofan Ma, Qisen Xu, Xiangfeng Wang, Bo Jin, Xiaoyun Zhang, Yanfeng Wang, and Ya Zhang. Boundary-aware supervoxel-level iteratively refined interactive 3d image segmentation with multi-agent reinforcement learning. *IEEE Transactions on Medical Imaging*, 40(10):2563–2574, 2020.

[33] Naima Otberdout, Mohammed Daoudi, Anis Kacem, Lahoucine Ballihi, and Stefano Berretti. Dynamic facial expression generation on hilbert hypersphere with conditional wasserstein generative adversarial nets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[34] Dipika Singhania, Rahul Rahaman, and Angela Yao. Coarse to fine multi-resolution temporal convolutional network. In *CVPR*, 2021.

[35] Li Siyao, Weijiang Yu, Tianpei Gu, Chunze Lin, Quan Wang, Chen Qian, Chen Change Loy, and Ziwei Liu. Bailando: 3d dance generation by actor-critic gpt with choreographic memory. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11050–11059, 2022.

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

[37] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[38] Liangliang Wang, Lianzheng Ge, Ruifeng Li, and Yajun Fang. Three-stream cnns for action recognition. *Pattern Recognition Letters*, 92:33–40, 2017.

[39] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. In *CVPR*, 2021.

[40] Zhenzhi Wang, Ziteng Gao, Limin Wang, Zhifeng Li, and Gangshan Wu. Boundary-aware cascade networks for temporal action segmentation. In *ECCV*, 2020.

[41] Dongxu Wei, Xiaowei Xu, Haibin Shen, and Kejie Huang. Gac-gan: A general method for appearance-controllable human video motion transfer. *IEEE Transactions on Multimedia*, 23:2457–2470, 2020.

[42] Di Wu, Junjun Chen, Nabin Sharma, Shirui Pan, Guodong Long, and Michael Blumenstein. Adversarial action data augmentation for similar gesture action recognition. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[43] Fangqiu Yi, Hongyu Wen, and Tingting Jiang. ASFormer: Transformer for action segmentation. *CoRR*, arXiv 2110.08568, 2021.