Gated Forward Refinement Network for Action Segmentation

Dong Wang, Yuan Yuan*, Qi Wang

School of Computer Science and Center for OPTical IMagery Analysis and Learning (OPTIMAL),
Northwestern Polytechnical University,
Xi'an 710072*, P. R. China*

Abstract

Action segmentation aims at temporally locating and classifying video segments in long untrimmed videos, which is of particular interest to many applications like surveillance and robotics. While most existing methods tackle this task by predicting frame-wise probabilities and adjusting them via high-level temporal models, recent approaches classify every video frame directly with temporal convolutions. However, there are limits to generate high quality predictions due to ambiguous information in the video frames. In this paper, in order to address limitations of existing methods in temporal action segmentation task, we propose an end-to-end multi-stage architecture, Gated Forward Refinement Network (G-FRNet). In G-FRNet, each stage makes a prediction that is refined progressively by next stage. Specifically, we propose a new gated forward refinement network to adaptively correct the errors in the prediction from previous stage, where an effective gate unit is used to control the refinement process. Moreover, to efficiently optimize the proposed G-FRNet, we design an objective function that consists of a classification loss and a multi-stage sequence-level refinement loss that incorporates segmental edit score via policy gradient. Extensive evaluation on three challenging datasets (50Salads, Georgia Tech Egocentric Activities (GTEA), and the Breakfast dataset) shows our method achieves state-of-the-art results.

Keywords:

---

*Corresponding author
Email address: y.yuan1.ieee@gmail.com (Yuan Yuan*)

Action Segmentation, Video Analysis, Refinement Network, Policy Gradient

1. Introduction

Analyzing activities in videos has gained an increasing interest over recent years benefiting from the large amount of publicly available video data. While considerable advances have been made in human action recognition (classifying short trimmed videos) [1, 2], temporally locating and recognizing actions in long untrimmed videos are still challenging. Earlier approaches [3, 4, 5] for action segmentation use sliding temporal windows of different scales to detect action segments and predict its action label. Recently, inspired by success of neural network [6, 7, 8, 9], most methods directly predict frame-wise action labels with temporal deep models, e.g., temporal convolutional networks[10], or recurrent networks [11, 12]. Moreover, to capture long dependencies between video frames, the encoder-decoder architecture with stacked temporal convolutions is adapted for temporal action segmentation. Despite the advancements made by these methods, most proposed temporal deep models tend to generate non-smooth predictions as they ignore the temporal continuity of action labels.

In order to overcome these limitations, we present a new temporal convolutional model that stacks temporal convolutions in a multi-stage manner. Specifically, we follow the multi-stage structure in MS-TCN (multi-stage temporal convolutional network)[13] and take it as the baseline of our model. MS-TCN uses a sequence of dilated temporal convolutions as single-stage TCN and frame-wise features are fed into single-stage TCN to obtain an initial prediction. Moreover, to improve the accuracy of the prediction, MS-TCN sequentially stacks multiple single-stage TCNs that operate directly on the output of the previous one to get the final prediction. The effect of such composition is an implicit refinement of the predictions from the previous stages. In another words, MS-TCN simply feeds the output of the previous stage into next stage to refine the prediction. But for individual frame-wise prediction result, this implicit refinement does not tell whether it should be modified. As a result of this implicit refinement, the refined prediction from next stage may change the correct frame-wise prediction results in previous prediction and harm the subsequent refinement stage. In addition, the errors caused by implicit refinement are accumulated after several refinement stages and influence the final prediction significantly.

In order to alleviate above problems, we adopt the single-stage TCN from MS-TCN as the initial-stage TCN in our model and propose a new gated forward refinement network to adaptively correct the errors in previous prediction. Compared with implicit refinement in baseline method (MS-TCN), we introduce gate unit to control the refinement process over previous prediction. Specifically, the gated forward refinement network consists of one correct unit and one gate unit. Correct unit shares the same network architecture with initial-stage TCN, which takes the prediction from the previous stage as input and generates the corrected results. Gate unit outputs gated weight based on hidden representations and previous prediction, which finds the errors in previous prediction by exploiting the context in the neighboring labels and feature representations. And the gated forward refinement network refines the previous stage prediction in next stage according to the output of gate unit and correct unit, i.e., we refine the previous prediction with the corrected result from correct unit according to gated weight. In this way, the accumulated error from previous stages can be filtered out via small gated weight, which improves the accuracy of the refined prediction.

Moreover, MS-TCN applies the same loss function on every stage prediction, which does not explicitly require refined prediction better than previous prediction. This training objective function in MS-TCN can not force the refinement stage to correct the errors in the previous prediction. In order to tackle this problem, we introduce a multi-stage sequence-level refinement loss that forces refined prediction to achieve higher evaluation metrics than previous prediction. In detail, we integrate the traditional cross entropy loss with evaluation metric (segmental edit score) via policy gradient method, where segmental edit score difference between refined prediction and previous prediction are regarded as reward in policy gradient method. In training procedure, the policy gradient method maximizes this reward to force the proposed gated forward refinement network to only correct the errors in the previous prediction. In summary, we propose a new gated forward refinement network and introduce a multi-stage sequence-level refinement loss to address the problems in MS-TCN, which results in more robust action classification and accurate action segmentation in videos.

We evaluate the proposed model on the following benchmark datasets: University of Dundee 50 Salads (50Salads) [14], Georgia Tech Egocentric Activities (GTEA) [15], and the Breakfast dataset [16]. Our results demonstrate that G-FRNet is capable of capturing dependencies between distinct actions and producing smooth predictions. Also, G-FRNet outperforms the state-of-the-art on frame-wise accuracy, segmental edit score, and segmental overlap F1 score. Our key contributions include:

- We propose a new multi-stage temporal convolutional architecture with gated forward refinement network, which adaptively finds the errors in the previous prediction and corrects them based on the outputs of the proposed gate unit.

- To force the refinement network to only correct the errors in previous prediction, we introduce a multi-stage sequence-level refinement loss that directly optimizes the segmental edit score via policy gradient method and design a multi-stage refinement reward mechanism to suppress the errors in the predictions.

- We outperform the state of the art in action segmentation on the 50Salads, GTEA and Breakfast datasets.

2. Related Work

Temporally locating and recognizing action segments in long untrimmed videos have been studied by many researchers. Inspired by object detection, earlier approaches [3, 4] adopt sliding window with various temporal scales to detect action segments followed with non-maximum suppression. Then, traditional methods focus on designing sophisticated models to represent actions. Fathi and Rehg [17] model actions based on the change in the state of objects and materials. In [18], an action description is constructed in terms of the interactions between hands and objects and a hierarchical model of activities, actions and objects are proposed to provide context to recognize actions. Bhattacharya et al. [19] express each video as an ordered vector time series from linear dynamical systems theory, where each time step consists of the vector formed from the concatenated confidences of the pre-trained concept detectors. These concept-based temporal representations are obtained from overlapping temporal windows and suitable for the complex event recognition. Cheng et al. [5] model temporal dependencies in the data by a sequence of visual words learnt from the video, and model the long-range dependencies by employing a Bayesian non-parametric model of discrete sequences to jointly classify and segment video sequences.

Other approaches follow a two-step pipeline that applies high level temporal modeling over frame-wise classifier. Kuehne et al. [20] combine the hidden Markov model (HMM) with a context-free grammar to determine the most probable sequence of actions over frame-wise action representation. A variable-duration hidden Markov model is used in [21] to model durations of action states in addition to the transitions between action states. Vo and Bobick [22] address action segmentation by parsing a temporal sequence with a Bayes network, where the temporal structure of the high-level activity is represented by a stochastic context-free grammar. Moreover, Richard and Gall [23] use statistical length and language modeling to represent temporal and contextual structure and find the most likely action sequence and the corresponding segment boundaries. While these approaches achieve good results, they are very slow as these temporal models require solving a maximization problem over very long sequences.

Motivated by the success of temporal convolution in speech synthesis [24], many researchers have used and modified it for the temporal action segmentation task. Lea et al. [10] use a hierarchy of temporal convolutions to perform fine-grained action segmentation. Their approach uses pooling and upsampling in an encoder-decoder architecture to efficiently capture long-range temporal patterns. Ding et al. [25] replace the convolutional decoder in the approach of Lea et al. [10] with a bi-directional LSTM. In addition, Lei and Todorovic [26] use deformable temporal convolutions instead of the regular temporal convolution and add a temporal residual stream for resolving ambiguities about local, frame-to-frame segmentation. The approach of Farha and Gall [13] is the most related to ours, as they use a multi-stage architecture for the temporal action segmentation task. However, their model just simply stacks several identical sub-network to refine the initial prediction, whereas our model refines the previous predictions with a gated forward network.

In addition, there are a variety of different approches that address the action segmentation task with weakly supervised action labeling [27, 28, 12, 29, 30]. Kuehne et al. [29] use HMM to model the action and combine the corresponding transcripts to infer the scripted actions. They iteratively refine the segmentation by modifying the prediction to maximize the likelihood of all possiable sequences. Following a similar pipeline, Richard et al. [30] divide each acion into multiple sub-actions and propose an iterative fine-to-coarse action modeling mechanism with RNN and HMM. In the other hand, Bojanowski et al. [27] formulate the action segmentation task as a temporal assignment problem and propose ordering constrained discriminative clustering to tackle it. And Huang et al. [12] propose the extended connectionist temporal classification framework to efficiently evaluate all possible alignments via dynamic programming and explicitly enforce their consistency with frame-to-frame visual similarities. In contrast to these approaches, we address the temporal action segmentation task in a fully supervised setup and the weakly supervised case is beyond the scope of this paper.

3. Framework

The action segmentation is aimed at temporally locating and classifying action segments in long untrimmed videos, and we tackle it by predicting frame-wise action label for each frame. Given the frames of a video $x_{1:T} = (x_1, \cdots, x_T)$, our goal is to infer the class label for each frame $c_{1:T} = (c_1, \cdots, c_T)$, where T is the video length. The rest of this section is organized as follows. First, we give an overview of the proposed method in Section 3.1 and describe the basic-block model that stacks dilated temporal convolutions in Section 3.2, then we explain the gated forward refinement network across multiple stages in Section 3.3. Finally, we discuss the proposed multi-stage sequence-level refinement loss in Section 3.4.

3.1. Overview

As shown in object detection [31], semantic segmentation [32] and other tasks, the accuracy of the initial prediction can be improved significantly by iteratively refining the prediction results. Inspired by this observation, we attempt to reduce the errors in the initial prediction by correcting them with the proposed gated forward refinement network, which iteratively refines the initial prediction over multiple steps. An overview of the proposed model is shown in Figure 1. Our model consists of multiple stages where the initial prediction from the inital stage is progressively refined with the gated forward refinement network over several steps, and the refined prediction from last step are regarded as the final prediction. Meanwhile, the proposed multi-stage sequence-level refinement loss incorporates segmental edit score via policy gradient method and a multi-stage reward mechanism is designed to guide the refinement process across the stages. Specifically, for an input video sequence $x_{1:T}$, each of the inputs $x_1, \cdots, x_T$ is a D-dimension feature representations extracted with pre-trained I3D network [1]. First, the feature vectors $x_{1:T}$ are fed into the first stage network and the initial prediction $Y^0 = (y_1^0, \cdots, y_T^0)$ is obtained by the initial-stage temporal convolutional network (initial-stage TCN). Then, the initial prediction Y^0 is passed into the

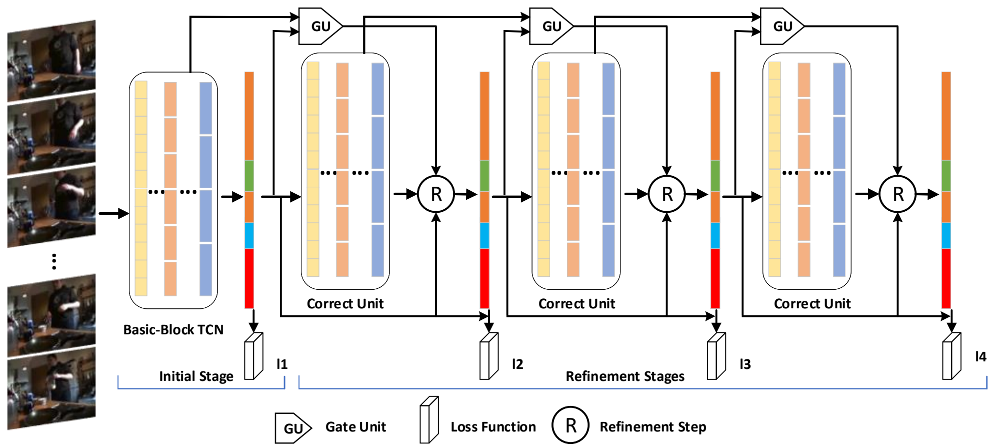

Figure 1: The overview of the proposed Gated Forward Refinement Network (G-FRNet). The frame-wise features are fed into the initial stage and obtain an initial prediction, which is progressively refined by applying several refinement stages. More details in section 3.

next refinement stage, which consists of basic-block temporal convolutional network and gated forward refinement network. The refined prediction Y^1 is then re-refined via the next refinement stage and all predictions after $S-1$ refinement stages, including the inital prediction, are denoted as $Y^s = (y_1^s, \cdots, y_T^s), s = 0,1,...,S-1$. Our method take the last stage prediction Y^{S-1} as the final result.

We propose a gated forward refinement network (G-FRNet) which forms the refinement stage of our model. Following previous work on multi-stage refinement [32, 33], G-FRNet leverages feature representations from the previous stage to obtain gated weight to correct the previous prediction. Meanwhile, the G-FRNet generates a corrected result based on the previous prediction and refines the previous prediction by adding the corrected result and previous prediction according to gated weight. In the other words, by using the gated weight, the G-FRNet finds errors in previous prediction and corrects it with the corrected result. In addition, to force the refinement network to correct the errors in previous prediction, we adopt a multi-stage sequence-level refinement loss to enforce the refined prediction from each stage to get higher evalution scores than the previous prediction. This multi-stage sequence-level refinement loss integrates the segmental edit score into the objective function and uses Reinforcement Learning (RL) method to optimize it, where rewards are introduced at each stage as intermediate supervision.

3.2. Initial-Stage TCN

Recently, temporal convolutional networks[34, 35] have proven their superior strength in modeling the time series data[24, 36, 37]. For example, van den Oord et al.[24] propose a fully convolutional model, called WaveNet, for audio signal generation, and Dauphin et al.[37] introduce a convolutional network for context dependencies modeling in language sequential analysis. Inspired by the success of convolutional approaches in the analysis of temporal sequential data, we leverage a stack of 1D convolutional layers to model the temporal dynamics and context dependencies over the video sequence frames. Specifically, as shown in Figure 1, the initial stage and correct unit in our model consist of an identical network architecture, referred as Basic-Block TCN (temporal convolutional network), which follows the design idea from single-stage TCN in MS-TCN.

In detail, the first layer of the initial-stage TCN is a 1D convolutional layer with kernel size 1, which takes pre-extracted features $x_{1:T}$ or predictions $y^s_{1:T}$ from previous stage as input. Then, the output is fed into the basic-block TCN, which stacks several the dilated residual layers and one classifier layer. Each dilated residual layer is composed of a dilated temporal convolution and a 1×1 convolutional layer. Specifically, the output of the previous layer is first passed into dilated convolutional layer and then its output is processed by 1×1 convolution. Moreover, residual connection is employed to facilitate gradients backpropagation. Formally, the operations at each dilated residual layer can be described as follows

$$
\begin{aligned}
\hat{H}_l &= ReLU(W_1 \circledast H_{l-1} + b_1) \\
H_l &= H_{l-1} + W_2 \circledast \hat{H}_l + b_2,
\end{aligned}
\tag{1}
$$

where $l \in [1, L]$ is the layer number, H_l is the output of l-th dilated residual layer, $\circledast$ denotes the convolution operators in dilated temporal convolution and a 1×1 convolutional layer. $W_1 \in R^{3\times D\times D}$ and $W_2 \in R^{1\times D\times D}$ are learnable weights and $b_1, b_2 \in R^D$ are bias vectors of the convolution layer. D is the number of convolutional filters.

In order to exploit the long-term temporal dependencies in the video, we stack several dilated residual layers to make the basic-block TCN cover a large receptive field. On the top of the last dilated residual layer, to get the prediction probabilities of action class, a 1×1 convolution followed by a SoftMax activation (classifier layer) is applied over the output of the last dilated residual layer, *i.e.*,

$$Y = softmax(W \circledast H_L + b)\,, \qquad (2)$$

where $Y = y_{1:T}$ are the class probabilities for the sequence, H_L is the output of the last dilated residual layer, $W \in R^{C \times D}$ and $b \in R^C$ are learnable weight and bias of the 1×1 convolutional layer, where C is the number of classes.

3.3. Gated Forward Refinement Network

Refining the initial predictions iteratively has shown significant improvements in many tasks like object detection [31] and semantic segmentation [32]. The idea of these multi-stage architectures is solving a complex problem with multiple steps such that each step only needs to make a refinement over the output of the previous one. Another benefit of the multi-stage method is it can effectively prevent the model from over-fitting the training data, because it contains fewer parameters than a big model.

Motivated by the success of these multi-stage architectures, we adopt the similar multi-stage structure in MS-TCN and take it as the baseline of our model. To improve the ability of the model, in this work, we introduce a new gate unit into refinement network to adaptively find the errors in the previous prediction, named gated forward refinement network. Meanwhile, a correct unit is employed to generate the corrected results from previous prediction. The refined prediction is obtained by weighted summation of the corrected result and previous prediction according to gate unit output. Through the cooperation between gate unit and correct unit, the proposed gated forward refinement network will correct the errors and keep the right results in previous prediction.

Specifically, As shown in Figure 1, the gated forward refinement network consists of two sub-networks, one (Correct Unit) takes the prediction from the previous stage as input and generates the corrected results, another one (Gate Unit) is fed with hidden representations and prediction from the previous stage and outputs gated weight to refine the previous prediction with the corrected result from correct unit. Specifically, correct unit and gate unit contain only temporal convolutional layers, and correct unit has the identical network architecture with Basic-Block TCN. Specifically, the correct unit is

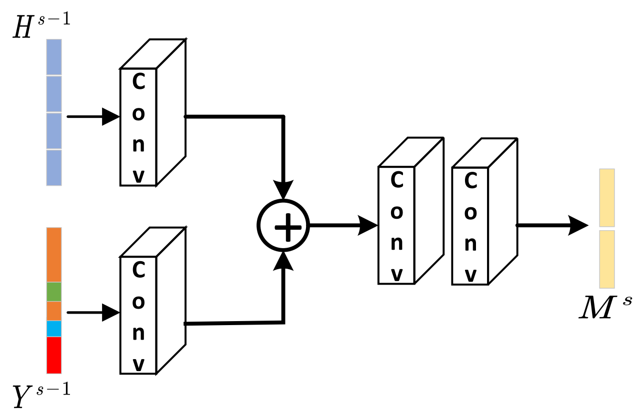

Figure 2: Overview of the gate unit.

composed of L dilated residual layers and its operations are denoted as follows

$$R^s = \mathcal{F}\left(Y^{s-1}\right), \tag{3}$$

where R^s is the corrected result at stage s over previous prediction, Y^{s-1} is the output at stage $s-1$ and $\mathcal{F}$ is the basic-block TCN discussed in Section 3.2. Operating on the previous prediction other than feature representation helps in capturing dependencies between action classes and generating plausible action sequence. Furthermore, since the dimension of prediction is far less than feature representation, there is a bottleneck layer between two successive stages, which helps in alleviating the over-fitting problem.

Gate Unit. The baseline method MS-TCN [13] stacks several identical predictors sequentially to correct errors in the inital prediction progressively. Instead of generating the refined prediction with other classfier directly, we introduce gate unit to control the refinement process over the previous prediction. The gate unit is designed to find the errors in previous prediction by exploiting the context in the neighboring labels and feature representations. Figure 2 illustrates the architecture detail of the proposed gate unit. Specifically, the gate unit at stage s takes prediction Y^{s-1} and feature representation H_L (denoted as H^{s-1} for simplicity) from previous stage $s-1$ as its input. The features in H^{s-1} express the similarity and dissimilarity between consecutive frames, whereas class probabilities in Y^{s-1} capture the rationality of predicted action sequences. The motivation for combining H^{s-1} and Y^{s-1} in gate unit is that two consecutive timesteps with similar feature representations should be labeled as the same action class and vice versa. A sequence of operations is carried out on H^{s-1} and Y^{s-1} followed by a softmax activation. Firstly, we apply a 1×1 convolution with D convolutional filters to both inputs respectively, note that H^{s-1} and Y^{s-1} have different dimensions. After these operations, the two outputs are concatenated and fed into a temporal convolution with kernel size 3. Finally, another convolution layer with softmax activation is used to obtain the gated refinement weight. The formulation of operations in gate unit can be written as follows

$$\begin{gathered} g_h = W_h \circledast H^{s-1} + b_h,\ g_y = W_y \circledast Y^{s-1} + b_y \\ g = W_g \circledast cat\left(g_h,\ g_y\right) + b_g \\ M^s = softmax\left(W \circledast g + b\right), \end{gathered} \tag{4}$$

where $W_h \in R^{1\times D\times D}$, $W_y \in R^{1\times C\times D}$ are the convolutional weights and b_h, b_y are the bias vectors. The outputs $g_h,\ g_y \in R^{N\times T\times D}$ are concatenated and a convolution layer with weight $W_g \in R^{3\times 2D\times D}$ is applied to fuse the information from H^{s-1} and Y^{s-1}. At last, the output $g \in R^{N\times T\times D}$ is fed into a temporal convolution followed by softmax activation, where the learnable weight is $W \in R^{3\times 2\times D}$ and output $M^s \in R^{N\times T\times 2}$ is gated refinement weight to control refinement process. N is the batch size.

Refinement Step. Given the previous prediction Y^{s-1}, corrected result R^s, and gated refinement weight M^s, the refinement over the previous prediction can be formulated as follows

$$Y^s = M\left[:,:,0\right] \odot R^s + M\left[:,:,1\right] \odot Y^{s-1}, \tag{5}$$

where $M\left[:,:,0\right], M\left[:,:,1\right] \in R^{N\times T\times 1}$ are splited from M and $\odot$ denotes element-wise product. Note that the sum of $M\left[n,t,0\right], M\left[n,t,1\right]$ equals 1 for every n and t.

3.4. Stage-wise Supervision

The multi-stage refinement approach described above results in a deep architecture and produces a sequence of prediction. Although we are principally interested in the prediction at the last stage, predictions produced at earlier stages allow incorporating the supervision into the intermediate layers, which is validated for the vanishing gradient problem. To be specific, a two-step training procedure is designed to optimize our network. First, for prediction at each stage s, we use a combination of typical classification loss and a regularization loss proposed in [13]. The cross entropy loss is adopted as classification loss

$$\mathcal{L}_{cls} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{c=1}^{C} v_{t,c}\log\left(y_{t,c}\right), \tag{6}$$

where $v_t \in R^C$ is the one-hot encoded vector of groundtruth action label and $y_t \in R^C$ is the predicted probabilities vector at time t. Moreover, an additional regularization loss proposed in [13] is employed to reduce over-segmentation errors. Formally, this regularization loss is a truncated mean squared error over the frame-wise log-probabilities

$$\begin{aligned} \mathcal{L}_{T-MSE} &= \frac{1}{TC}\sum_{t,c}\tilde{\Delta}_{t,c}^{2} \\ \tilde{\Delta}_{t,c} &= \begin{cases} \Delta_{t,c} & : \Delta_{t,c} < \tau \\ \tau & : otherwise \end{cases} \\ \Delta_{t,c} &= \left|\log y_{t,c} - \log y_{t-1,c}\right|, \end{aligned} \tag{7}$$

where T is the video length, C is the number of classes, and $y_{t,c}$ is the probability of class c at time t. Note that the gradients are only computed and backpropagated through $y_{t,c}$. By integrating these two losses at each stage s, we obtain the first-step training objective for the full architecture by sum of the losses over all stages:

$$\mathcal{L} = \sum_{s}\mathcal{L}_{cls}\left(Y^{s}\right) + \lambda\mathcal{L}_{T-MSE}\left(Y^{s}\right). \tag{8}$$

Multi-Stage Sequence-Level Refinement Loss. Training with the above loss function is prone to produce non-smooth predictions, which makes a significant impact on action segmentation performance. The reason behind this is that the log-likelihood score of the prediction ignores the relationship between temporal consecutive predicted scores and does not correlate well with the action segmentation evaluation metrics such as segmental edit score and temporal IoU ratio. In detail, firstly, the evaluation metrics need to identify the action segments, which is achieved by traversing the prediction in chronological order and finding the time interval with the same action label. Once one error appears, even surroundding with correct prediction, the obtained action segments show serious discrepancy between the groundtruth action segments. To address this problem, we incorporate the evaluation metrics into the training objective function that punishes the non-smooth prediction directly. Due to non-differentiability of the evaluation metrics, we consider the action segmentation process (frame-wise action label prediction) as a reinforcement learning problem, i.e., given an environment (the input videos), the proposed model are viewed as an agent that conduct the frame-wise action classification. After generating the complete prediction, the agent will observe a sequence-level reward (evaluation metrics).

We cast our action segmenmtation model in the terminology as in [38]. Our proposed network at each stage can be viewed as an agent that interacts with external envioronment (input features). The policy at stage s, denoted as p_{θ_s}, is defined by the parameters of the network θ_s. In our model, the policy p_{θ_s} outputs the action label sequence A^s that is obtained from the predicted class probabilities Y^s via multinomial sampling. Then, the reward is computed by an multi-stage reward mechanism that utilizes the evaluation metric (segmental edit score in this work). The goal of training is to minimize the negative expected reward over all stages:

$$\mathcal{L}_{rl}(\theta) = -\sum_{s} \mathcal{L}_{rl}(\theta_s) = -\sum_{s} \mathbb{E}_{A^s \sim p_{\theta_s}} \left[R\left(A^s\right) \right], \tag{9}$$

where $A^s = \{a_0^s, ..., a_T^s\}$ and a_t^s is sampled from the stage s at time step t. $R\left(A^s\right)$ is the proposed multi-stage reward mechanism.

Compared with first-step training objective in Equation 8, The principal idea of our multi-stage reward mechanism is that the refined prediction at stage s should achieve higher evalution score than the previous stage, which explicitly forces the refinement stage to correct the errors in previous prediction. Moreover, we require the result obtained at test-mode (max sampling) to be no worse than results sampled from multinomial distribution Y^s. Particularly, $R\left(A^s\right)$ is defined as:

$$R(A^s) = \begin{cases} [r(\hat{A}^s) - r(A^s)], & : s = 0 \\ [r(\hat{A}^s) - r(A^s)] + \gamma[r(A^s) - r(A^{s-1})], & : otherwise \end{cases}, \tag{10}$$

where $\hat{A}^s$ is the action label sequsence via max sampling. $r\left(\cdot\right)$ is the segmental edit score computed by comparing the predicted action sequence to corresponding ground-truth action sequence. In the initial stage ($s = 0$), the reward tends to supress those samples that have higher scores than max-sampled result, because we hope the max-sampled results (same as testing) achieved higher score. For the refinement stages ($s = 1, ..., S-1$), an additional item is used to increases the probability of the samples from stage s that outperform the samples from stage $s-1$ and punishes the inferior samples.

After obtaining the reward, we use the REINFORCE algorithm [39] to compute the gradient $\nabla_{\theta_s} L_{rl}(\theta_s)$. REINFORCE is based on the observation that the expected gradient of a non-nondifferentiable reward function can be computed as follows:

$$\nabla_{\theta_s} L_{rl}(\theta_s) = -\sum_{s} \mathbb{E}_{A^s \sim p_{\theta_s}} \left[R(A^s) \nabla_{\theta_s} \log_{p_{\theta_s}}(A^s) \right]. \tag{11}$$

In practice the expected gradient can be approximated using a single Monte-Carlo sample $A^s = \{a_0^s, ..., a_T^s\}$ from p_{θ_s}, for each training example in the minibatch, the gradients are:

$$\nabla_{\theta_s} L_{rl}(\theta_s) \approx -R(A^s) \nabla_{\theta_s} \log_{p_{\theta_s}}(A^s). \tag{12}$$

In practice, in the second training step, we update the network by summing the gradients from Equation 8 and Equation 12.

4. Experiments

4.1. Datasets and Experiment Setup

We evaluate our approach on three datasets: University of Dundee 50 Salads (50Salads) [14], Georgia Tech Egocentric Activities (GTEA) [15], and the Breakfast dataset [16], which are widely used in action segmentation task. The 50salads dataset contains 50 videos with 17 action classes. On average, each video contains 20 action instances and is 6.4 minutes long. The videos depict the salad preparation activities, which are performed by 25 actors where each actor prepares two different salads. For evaluation on this dataset, we perform the same 5-fold cross-validation as the state-of-the-art methods, and report the average results. The GTEA dataset contains 28 videos of seven finegrained types of daily activities in a kitchen, like preparing coffee or cheese sandwich, performed by 4 subjects. Each video are annotated with 11 action classes including background, showing a sequence of 20 action instances including the background action. For this dataset, we perform the same 4-fold cross-validation as prior work, and report the average results. The Breakfast dataset is the largest among the three datasets with 1712 videos. As the name of the dataset indicates, the videos are recorded in 18 different kitchens with breakfast related activities. Overall, there are 48 different actions where each video contains 6 action instances on average. For evaluation, we use the standard 4 splits as proposed in [16] and report the average.

Evaluation Metrics. For all the three datasets, we use the following evaluation metrics as in [10]: frame-wise accuracy, segmental edit score and segmental overlap F1 score with threshold 10%, 25% and 50%, denoted by F_1@{10,25,50}. The overlapping threshold is computed based on the intersection over union (IoU) ratio. While the frame-wise accuracy is the most commonly used metric for action segmentation, it does not take into account the temporal structure of the prediction. For example, results with the same frame-wise accuracy may show large qualitative differences. Also, this metric does not take the temporal continuity of human actions into consideration. To address these limitations, a segmental edit score is proposed [40, 41] to penalizes this over-segmentation error. Similarly, segmental overlap F1 score also penalizes oversegmentation errors, but ignores minor temporal shifts between the predictions and ground truth (which might arise from annotation noise).

4.2. Implementation Details and Baselines

Our approch consists of one initial stage and three refinement stages that are implemented with PyTorch, i.e., $S = 4$. Each basic-block TCN is composed of 10 (L=10) dilated residual layers and the dilation factor is doubled at each layer followed by an dropout layer. We set the number of convolutional filters D to 256 in all the layer and filter size is 3. As input, following the related work [13], we first downsample the video frames and then extract I3D [1] features with 2048 dimension. For GTEA and Breakfast datasets we use the videos temporal resolution at 15 fps, while for 50Salads we downsampled the features from 30 fps to 15 fps to be consistent with the other datasets. Parameters of our network are learned using the Adam optimizer with a fixed learning rate of 5e-5. The batch size N and the number of epoches are 1 and 100, respectively. We first pre-train the network with Equation 8 from scratch for 50 epoches, and the obtained network is fine-tuned by combining Equation 8 and Equation 12 for another 50 epoches.

Baselines. We take the approach of Farha and Gall [13] MS-TCN as the baseline method, which also use a multi-stage architecture for action segmentation, is the most related to ours. Compared with simply stacking several identical network in [13], our approch proposes an gated forward refinement network to refine the previous prediction and an novel sequence-level refinement loss to guide the refinement stages. Extensive evaluation on three challenging datasets demonstrates the superiority of our approch over [13]. For ablation studies, we specify the following G-FRNet variants: (1) FRNet: G-FRNet without gate unit (i.e., summing R^s and Y^{s-1} directly in Equation 5); (2) G-FRNet$_{pre}$: G-FRNet trained only with Equation 8; and (3) G-FRNet$_S k$ ($k \in \{0, ..., S-2\}$): predicted results from the stage k. Note that the results of our approch is the prediction from the last stage $S-1$. We also compare with the following closely related work: ST-CNN [40]: Temporal convolutional filter that builds on top of spatial CNN to capture scene changes over the course of action; ED-TCN [10]: encoder-decoder temporal convolution neural network; and TDRN [26]: two-stream network that consists of temporal residual modules with deformable convolutions.

4.3. Experimental results

4.3.1. Comparison with Temporal Convolution Models

Quantitative results on 50Salads and GTEA dataset are depicted in Table 1. We compare our G-FRNet with the most related temporal convolution models, including ED-TCN [10], TResNet [42], TDRN [26] and MS-TCN [13], which also is our baseline method. As shown in the table, the performance of ED-TCN is worse than that of TResNet, and that TDRN is able to improve over TResNet by computing deformable temporal convolutions in two temporal streams. These methods are all single stage model that predicts the action labels without refinement. As for the multi-stage architecture, MS-TCN outperforms the other single stage methods on three evaluation metrics with a large margin (up to 12.6% for frame-wise accuracy (Acc) on the 50Salads dataset). In contrast with this baseline, our G-FRNet achieves the significant better results in terms of segmental overlap F1 score and segmental edit score. Specifically, our approach makes a moderate improvement, i.e., 2-3% in segmental overlap F1 score with different thresholds and segmental edit score on the both datasets, demonstrating its superiority in addressing the oversegmentation errors and non-smooth prediction results. Moreover, our model achieve these improvements without increasing the frame-wise accuracy, suggesting the advantage of correcting errors in the refinement stages. The effect of the refinement stage can also be seen in the qualitative results shown in Fig 4.

Table 1: Performance comparison with respect to the most related temporal convolution models.

Method	50Salads			GTEA		
	F_1@{10,25,50}	Edit	Acc	F_1@{10,25,50}	Edit	Acc
ED-TCN [10]	68.0,63.9,52.6	59.8	64.7	72.2,69.3,56.0	-	64.0
TResNet [42]	69.2,65.0,54.4	60.5	66.0	74.1,69.9,57.6	64.4	65.8
TDRN [26]	72.9,68.5,57.2	66.0	68.1	79.2,74.4,62.7	74.1	70.1
MS-TCN [13]	76.3,74.0,64.5	67.9	80.7	85.8,83.4,69.8	79.0	76.3
G-FRNet	**78.0,76.2,67.0**	**71.4**	**80.7**	**89.1,87.5,72.8**	**83.5**	**76.7**

4.3.2. Effect of the Refinement Stages

To demonstrate the effectiveness of stacking several refinement stages over the initial stage, we compare the results from the initial stage and the results from different refinement stages, which are summarized in Table 2. As shown in the table, the initial stage TCN already achieves a comparable frame-wise accuracy, and there is no obvious imporvement on frame-wise accuracy with refiement stages. Nevertheless, as we can see, the segmental edit score and segmental overlap F1 score of these predictions are of great difference, indicating the quality of these predictions are very different. To be specific, the initial prediction only gets 17.7% segmental edit score, and it increases to 48.7% after one refinement stage. On the other hand, Adding more refinement stages can progressively improves the scores and reduces the over-segmentation errors. There is a huge to the segmental edit score when two or three refinement stages are used, and the fourth refinement stage still improves the results but not as significant as the previous stages. The similar improvements are shown on segmental overlap F1 score with different thresholds, indicating the refinement stage's capacity for correcting the errors in previous prediction. Moreover, in Figure 3, we qualitatively compare the prediction from different stages on a sample test video from the 50 Salads dataset. As we can see, the errors in previous prediction are gradually corrected with the refinement stages. Also, the figure shows that the action boundaries are refined to more precise and the non-smooth results in groundtruth action segments are removed after applying more refinement stages. In another word, the sequential continuity of the action labels in one action segment and temporal dependencies between different action segments are modeled well with refinement stages. This suggests that using refinement

Figure 3: Qualitative result from the 50Salads dataset for comparing different predictions from differnt stages.

stage is critical for improving quality of prediction.

Table 2: Effect of the refinement stages on the 50Salads dataset.

	F_1@{10,25,50}			Edit	Acc
Inital-Stage TCN	24.1	22.2	18.6	17.9	79.3
G-FRNet (2 stages)	55.9	53.8	46.9	48.7	80.2
G-FRNet (3 stages)	70.9	69.2	60.6	63.7	80.6
G-FRNet (4 stages)	78.0	76.2	67.0	71.4	80.7

4.3.3. Ablation Studies

To verify the superiority of the proposed gated forward refinement network and multi-stage sequence-level refinement loss in temporal action segmentation, we compare our method with several variants baseline, including (1) FRNet: G-FRNet without gate unit (i.e., summing R^s and Y^{s-1} directly in Equation 5), (2) G-FRNet$_{pre}$: G-FRNet trained only with Equation 8, and MS-TCN-256 (each layer with 256 convolutional filters). All these variants are implemented with four stages and tested on the 50Salads dataset. As shown in Table. 4, MS-TCN-256 is worse than MS-TCN-64 in [13] on segmental edit score and segmental overlap F1 score, indicating an over-fitting problem occurs in MS-TCN method as a result of increasing the number of parameters. Note that MS-TCN-256 achieves the best frame-wise accuracy, showing the benefit from increasing the parameters. In contrast, our G-FRNet and its variants achieve comparative performance with 256 convolutional filters, which demonstrates the preferable generalization capacity of our model. FRNet$_{pre}$ denotes the model FRNet trained without the proposed multi-stage sequence-level refinement loss. G-FRNet$_{pre}$ outperforms this baseline in terms of three evaluation metrics, which validates that the proposed gate unit can adaptively control the refinement process over the previous prediction. Additionally, FRNet and G-FRNet represent the models fine-tuned by the proposed multi-stage sequence-level refinement loss, and the comparison FRNet *vs* FRNet$_{pre}$ and G-FRNet *vs* G-FRNet$_{pre}$ show the effectiveness of the proposed refinement loss. Particularly, the segmental edit score and segmental overlap F1 score of these models are significantly improved, such as 67.0% vs 57.4% on $F_1@\{50\}$, 71.4% vs 63.1% on segmental edit score between G-FRNet and G-FRNet$_{pre}$. Note that G-FRNet outperforms these variants baseline on all evaluation metrics, demonstrating the superiority of our method in temporal action segmentation.

Table 3: Comparisons of performance with several variants of the proposed method on 50Salads dataset.

	$F_1@\{10,25,50\}$			Edit	Acc
MS-TCN(64)	76.3	74.0	64.5	67.9	80.7
MS-TCN(256)	55.8	63.3	48.1	51.1	81.4
FRNet$_{pre}$	68.1	66.3	57.5	60.7	79.1
FRNet	71.7	69.5	61.3	63.5	78.4
G-FRNet$_{pre}$	70.3	67.1	57.4	63.1	74.2
G-FRNet	78.0	76.2	67.0	71.4	80.7

4.3.4. Comparing Different Reward Functions

As the core of multi-stage sequence-level refinement loss function, we design a multi-stage reward mechanism to force the refinement stage to correct the errors in previous prediction. As shown in previous section, while this refinement loss does not improve the frame-wise accuracy, we find this loss produces a huge boost to the segmental edit score and segmental overlap F1 score, indicating its superiority in reducing the errors in previous prediction. The reason behind this improvement is that the evaluation metric segmental edit score are used to optimize the model via policy gradient, where desiging reward function $R(\cdot)$ is the most important factor. To demonstrate the superiority of the proposed reward function in Equation 10, we compare our method with several different reward functions that also incorporate segmental edit score into training. In detail, first, we directly takes the segmental edit score of the prediction as the reward for that stage, i.e., $R(A^s) = r(A^s)$, denoted as Reward$_r$. Second, we remove the first term in Equation 10 for the stage $s > 0$ and use the segmental edit score as the reward for initial stage, $R(A^0) = r(A^0), R(A^s) = r(A^s) - r(A^{s-1}), s > 0$, denoted as Reward$_{r-r}$. And the proposed reward function 10 is denoted as Reward$_{all}$. As shown in Table 4, Reward$_{all}$ outperforms other methods on multiple metrics, especially segmental edit score. And the Reward$_{r-r}$ achieve better results on all metrics than Reward$_r$, which indicates that reward function comparing the predictions from consecutive two stage can guide the model to correct the errors. Futhermore, compared with the G-FRNet$_{pre}$ results from Table 3, we can find that the models trained with these three reward funtions all outperforms the pre-traind model, demonstrating the superiority of incorporating the segmental edit score into training.

Table 4: Comparisons of performance with different reward function on 50Salads dataset.

	F_1@{10,25,50}			Edit	Acc
Reward$_r$	77.0	74.6	65.7	69.7	80.0
Reward$_{r-r}$	**78.6**	76.0	66.7	70.4	80.6
Reward$_{all}$	78.0	**76.2**	**67.0**	**71.4**	**80.7**

4.3.5. Comparison with the State-of-the-Art

In this section, we compare the proposed model to the state-of-the-art methods on three datasets: 50Salads, Georgia Tech Egocentric Activities (GTEA) and Breakfast datasets. The results are presented in Table 5. As shown in the table, the proposed G-FRNet outputforms the state-of-the-art methods on the three datasets in terms of all metrics. Specifically, our approach makes a moderate improvement on 50Salads and GTEA dataset, i.e., 2%-4% in segmental edit score and segmental overlap F1 score, demonstratin its superiority in reducing the non-smooth errors. As for Breakfast datasets, our approch outperforms MS-TCN with a huge boost in items of all evaluation metrics, i.e., over 10% improvement on segmental overlap F1 score with different threshold. In addition, MS-TCN(IDT) replace the I3D features with the improved dense trajectories (IDT) features, which are the standard used features for the Breakfast dataset. And the improvement is not shown in evaluation metrics, indicating the impact of the features is unstable. This

Table 5: Comparison with the state-of-the-art on 50Salads, GTEA, and the Breakfast dataset.

50Salads	F_1@{10,25,50}			Edit	Acc
IDT+LM [23]	44.4	38.9	27.8	45.8	48.7
Bi-LSTM [11]	62.6	58.3	47.0	55.6	55.7
ED-TCN [10]	68.0	63.9	52.6	59.8	64.7
TDRN [26]	72.9	68.5	57.2	66.0	68.1
MS-TCN [13]	76.3	74.0	64.5	67.9	80.7
G-FRNet	**78.0**	**76.2**	**67.0**	**71.4**	**80.7**
GTEA	F_1@{10,25,50}			Edit	Acc
Bi-LSTM [11]	66.5	59.0	43.6	-	55.5
ED-TCN [10]	72.2	69.3	56.0	-	64.0
TDRN [26]	79.2	74.4	62.7	74.1	70.1
MS-TCN [13]	85.8	83.4	69.8	79.0	76.3
G-FRNet	**89.1**	**87.5**	**72.8**	**83.5**	**76.7**
Breakfast	F_1@{10,25,50}			Edit	Acc
ED-TCN [10]	-	-	-	-	43.3
TCFPN [28]	-	-	-	-	52.0
GRU [30]	-	-	-	-	60.6
MS-TCN (I3D) [13]	52.6	48.1	37.9	61.7	66.3
MS-TCN (IDT) [13]	58.2	52.9	40.8	61.4	65.1
G-FRNet	**71.1**	**65.7**	**53.6**	**70.6**	**67.7**

suggests that designing suitable model is critical for improving accuracy of action segmentation. Futhermore, since our model does not use any recurrent layers, it is very fast both during training and testing.

5. Conclusions

We present a gated forward refinement network for the temporal action segmentation task. The model consists of two components: an initial-stage TCN for obtaining the initial prediction and several refinement stages for refining previous prediction. The refinement stage learns to correct the errors in the previous prediction with the proposed gated forward refinement network, which adaptively control the refinement process with gate unit. The experimental evaluations demonstrate the capability of this multi-stage architecture in capturing temporal dependencies among action classes. Moreover, in order to force the refinement network to correct the errors in previous prediction, we introduce a multi-stage sequence-level refinement loss that incorporates the non differentiable segmental edit score into training via policy gradient method, and the non-smooth and over-segmentation errors in the prediction are significantly reduced. By combining these improvements, our model outperforms the state-of-the-art methods on three challenging datasets with a large margin and extensive evaluations have demonstrated the superiority of the proposed methods in addressing non-smooth and over-segmentation errors.

Acknowledgements

This work was supported by the National Key R&D Program of China under Grant 2017YFB1002202, National Natural Science Foundation of China under Grant 61632018, 61825603, U1864204 and 61773316.

References

[1] J. Carreira, A. Zisserman, Quo vadis, action recognition? a new model and the kinetics dataset, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 6299–6308.

[2] C. Feichtenhofer, A. Pinz, R. Wildes, Spatiotemporal residual networks for video action recognition, in: Advances in Neural Information Processing Systems, 2016, pp. 3468–3476.

[3] M. Rohrbach, S. Amin, M. Andriluka, B. Schiele, A database for fine grained activity detection of cooking activities, in: IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 1194–1201.

[4] S. Karaman, L. Seidenari, A. Del Bimbo, Fast saliency based pooling of fisher encoded dense trajectories, in: European Conference on Computer Vision, THUMOS Workshop, Vol. 1, 2014, p. 5.

[5] Y. Cheng, Q. Fan, S. Pankanti, A. Choudhary, Temporal sequence modeling for video event detection, in: IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 2227–2234.

[6] Q. Wang, J. Gao, Y. Yuan, Embedding structured contour and location prior in siamesed fully convolutional networks for road detection, IEEE Transactions on Intelligent Transportation Systems 19 (1) (2018) 230–241.

[7] B. Zhao, X. Li, X. Lu, Hierarchical recurrent neural network for video summarization, in: ACM international conference on Multimedia, 2017, pp. 863–871.

[8] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, L. Van Gool, Temporal segment networks: Towards good practices for deep action recognition, in: European conference on computer vision, 2016, pp. 20–36.

[9] Y. Zhao, Y. Xiong, L. Wang, Z. Wu, X. Tang, D. Lin, Temporal action detection with structured segment networks, in: IEEE International Conference on Computer Vision, 2017, pp. 2914–2923.

[10] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, G. D. Hager, Temporal convolutional networks for action segmentation and detection, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 156–165.

[11] B. Singh, T. K. Marks, M. Jones, O. Tuzel, M. Shao, A multi-stream bi-directional recurrent neural network for fine-grained action detection, in: IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1961–1970.

[12] D.-A. Huang, L. Fei-Fei, J. C. Niebles, Connectionist temporal modeling for weakly supervised action labeling, in: European Conference on Computer Vision, 2016, pp. 137–153.

[13] Y. A. Farha, J. Gall, Ms-tcn: Multi-stage temporal convolutional network for action segmentation, arXiv preprint arXiv:1903.01945.

[14] S. Stein, S. J. McKenna, Combining embedded accelerometers with computer vision for recognizing food preparation activities, in: ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2013, pp. 729–738.

[15] A. Fathi, X. Ren, J. M. Rehg, Learning to recognize objects in egocentric activities, in: IEEE Conference on Computer Vision and Pattern Recognition, 2011, pp. 3281–3288.

[16] H. Kuehne, A. Arslan, T. Serre, The language of actions: Recovering the syntax and semantics of goal-directed human activities, in: IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 780–787.

[17] A. Fathi, J. M. Rehg, Modeling actions through state changes, in: IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 2579–2586.

[18] A. Fathi, A. Farhadi, J. M. Rehg, Understanding egocentric activities, in: IEEE International Conference on Computer Vision, 2011, pp. 407–414.

[19] S. Bhattacharya, M. M. Kalayeh, R. Sukthankar, M. Shah, Recognition of complex events: Exploiting temporal dynamics between underlying concepts, in: IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 2235–2242.

[20] H. Kuehne, J. Gall, T. Serre, An end-to-end generative framework for video segmentation and recognition, in: IEEE Winter Conference on Applications of Computer Vision (WACV), 2016, pp. 1–8.

[21] K. Tang, L. Fei-Fei, D. Koller, Learning latent temporal structure for complex event detection, in: IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 1250–1257.

[22] N. N. Vo, A. F. Bobick, From stochastic grammar to bayes network: Probabilistic parsing of complex activity, in: IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 2641–2648.

[23] A. Richard, J. Gall, Temporal action detection using a statistical language model, in: IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 3131–3140.

[24] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, K. Kavukcuoglu, Wavenet: A generative model for raw audio., ISCA Speech Synthesis Workshop (SSW) 125.

[25] L. Ding, C. Xu, Tricornet: A hybrid temporal convolutional and recurrent network for video action segmentation, arXiv preprint arXiv:1705.07818.

[26] P. Lei, S. Todorovic, Temporal deformable residual networks for action segmentation in videos, in: IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 6742–6751.

[27] P. Bojanowski, R. Lajugie, F. Bach, I. Laptev, J. Ponce, C. Schmid, J. Sivic, Weakly supervised action labeling in videos under ordering constraints, in: European Conference on Computer Vision, 2014, pp. 628–643.

[28] L. Ding, C. Xu, Weakly-supervised action segmentation with iterative soft boundary assignment, in: IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 6508–6516.

[29] H. Kuehne, A. Richard, J. Gall, Weakly supervised learning of actions from transcripts, Computer Vision and Image Understanding 163 (2017) 78–89.

[30] A. Richard, H. Kuehne, J. Gall, Weakly supervised action learning with rnn based fine-to-coarse modeling, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 754–763.

[31] Z. Cai, N. Vasconcelos, Cascade r-cnn: Delving into high quality object detection, in: IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 6154–6162.

[32] M. Amirul Islam, M. Rochan, N. D. Bruce, Y. Wang, Gated feedback refinement network for dense image labeling, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 3751–3759.

[33] G. Lin, A. Milan, C. Shen, I. Reid, Refinenet: Multi-path refinement networks for high-resolution semantic segmentation, in: IEEE conference on computer vision and pattern recognition, 2017, pp. 1925–1934.

[34] H. Ding, X. Jiang, B. Shuai, A. Qun Liu, G. Wang, Context contrasted feature and gated multi-scale aggregation for scene segmentation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2393–2402.

[35] J. Gu, G. Wang, J. Cai, T. Chen, An empirical study of language cnn for image captioning, in: IEEE International Conference on Computer Vision, 2017, pp. 1222–1231.

[36] G. Varol, I. Laptev, C. Schmid, Long-term temporal convolutions for action recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 40 (6) (2017) 1510–1517.

[37] Y. N. Dauphin, A. Fan, M. Auli, D. Grangier, Language modeling with gated convolutional networks, in: International Conference on Machine Learning, 2017, pp. 933–941.

[38] M. Ranzato, S. Chopra, M. Auli, W. Zaremba, Sequence level training with recurrent neural networks, in: International Conference on Learning Representations, 2016.

[39] R. J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Machine learning 8 (3-4) (1992) 229–256.

[40] C. Lea, A. Reiter, R. Vidal, G. D. Hager, Segmental spatiotemporal cnns for fine-grained action segmentation, in: European Conference on Computer Vision, 2016, pp. 36–52.

[41] C. Lea, R. Vidal, G. D. Hager, Learning convolutional action primitives for fine-grained action recognition, in: IEEE International Conference on Robotics and Automation, 2016, pp. 1642–1649.

[42] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.