Pattern Recognition 129 (2022) 108764

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Maximization and restoration: Action segmentation through dilation passing and temporal reconstruction

Junyong Park[a,1], Daekyum Kim[a,b,1], Sejoon Huh[a], Sungho Jo[a,*]

[a] School of Computing, KAIST, Daejeon 34141, Republic of Korea
[b] John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge MA 02138, USA

ARTICLE INFO

Article history:
Received 12 December 2021
Revised 11 April 2022
Accepted 29 April 2022
Available online 2 May 2022

Keywords:
Action segmentation
Temporal segmentation
Video understanding

ABSTRACT

Action segmentation aims to split videos into segments of different actions. Recent work focuses on dealing with long-range dependencies of long, untrimmed videos, but still suffers from over-segmentation and performance saturation due to increased model complexity. This paper addresses the aforementioned issues through a divide-and-conquer strategy that first maximizes the frame-wise classification accuracy of the model and then reduces the over-segmentation errors. This strategy is implemented with the Dilation Passing and Reconstruction Network, composed of the Dilation Passing Network, which primarily aims to increase accuracy by propagating information of different dilations, and the Temporal Reconstruction Network, which reduces over-segmentation errors by temporally encoding and decoding the output features from the Dilation Passing Network. We also propose a weighted temporal mean squared error loss that further reduces over-segmentation. Through evaluations on the 50Salads, GTEA, and Breakfast datasets, we show that our model achieves significant results compared to existing state-of-the-art models.

1. Introduction

From robotics [1] to behavior analysis [2] and surveillance [3,4], recent progress in a variety of fields has increased the need for accurate algorithms that analyze human actions. Action segmentation, or the partitioning of a video in which humans engage in a sequence of actions into segments corresponding to each action, is undeniably an essential tool for such analyses. Frames are extracted from the video and then sequentially labeled with the algorithm's best estimate of what the action within the frame may be.

Despite the extensive interest in action segmentation, segmenting long and untrimmed videos still proves to be a challenging task. Simply increasing modeling complexity, i.e. stacking multiple layers, does not necessarily translate to a corresponding increase in performance [5,6]. Furthermore, models tend to over-segment long videos as they obtain greater accuracy due to the classification of ambiguous classes. Frames in wrongly classified action chunks will occasionally be correctly classified and vice versa, resulting in inconsistency between frames, causing over-segmentation.

Recent studies have therefore focused on dealing with the long-range dependencies inherent in long, untrimmed videos. MS-TCN [7], for example, proposes a model inspired by Wavenet [8] that is composed of stacked dilated convolutions and a smoothing loss designed to alleviate over-segmentation issues. Other works succeeding MS-TCN have also proposed models and smoothing methods built upon MS-TCN to improve on its accuracy. These works can be categorized into two groups: one where changes are mostly focused on modifying MS-TCN's network architecture and design [9–11] and one where over-segmentation is explicitly handled through an additional module that computes segment boundaries [5,12,13]. While changes to network architecture are made to capture temporal information through a variety of receptive fields to improve overall performance, segment boundaries are used for postprocessing to smooth the original classification and reduce over-segmentation.

This paper proposes the Dilation Passing and Reconstruction Network (DPRN), furthering the state-of-the-art in action segmentation. Also built off of MS-TCN, DPRN addresses the action segmentation problem with a different approach from previous work: Maximization and Restoration. DPRN consists of two novel subnetworks that focus on their respective part within the Maximization

[*] Corresponding author.

E-mail addresses: jyp0802@kaist.ac.kr (J. Park), daekyum@seas.harvard.edu (D. Kim), sejoonhuh@kaist.ac.kr (S. Huh), shjo@kaist.ac.kr (S. Jo).

[1] These authors contributed equally to this work. Daekyum Kim was with KAIST [a] when this work was done.

https://doi.org/10.1016/j.patcog.2022.108764
0031-3203/© 2022 Elsevier Ltd. All rights reserved.

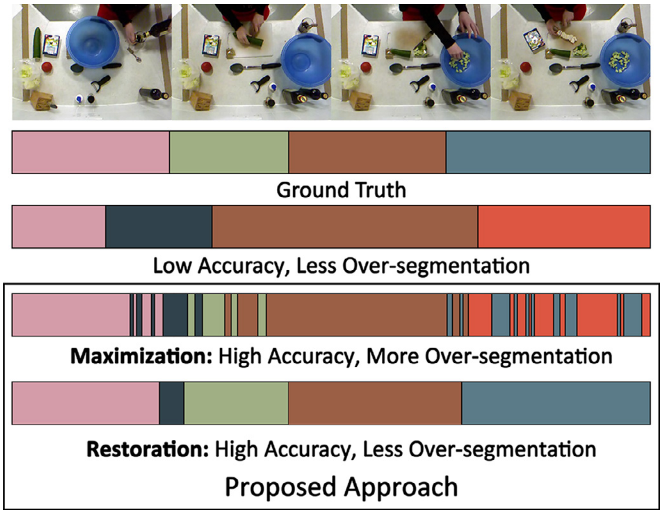
Ground Truth
Low Accuracy, Less Over-segmentation
Maximization: High Accuracy, More Over-segmentation
Restoration: High Accuracy, Less Over-segmentation
Proposed Approach

Fig. 1. Illustration of our proposed approach. We first increase frame-wise accuracy (Maximization), then reduce over-segmentation (Restoration).

and Restoration approach. Heeding the importance of utilizing different temporal resolutions, the Dilation Passing Network (DPN) aims to dominantly increase the frame-wise classification accuracy by utilizing High-to-Low and Low-to-High networks that pass information on different receptive fields between the dilated residual layers of MS-TCN. As postprocessing performance is necessarily constrained by the accuracy of the initial classification, the prioritization of accuracy maximization enables the classification of ambiguous frames that previous state-of-the-art models found difficult to identify. The Temporal Reconstruction Network (TRN) subsequently aims to restore over-segmented frames to enforce temporal consistency within each segment and better the final prediction. TRN temporally encodes the output of DPN to latent features and decodes them to refine the action outputs. To further reduce over-segmentation, we also propose a Weighted Temporal Mean Squared Error (WT-MSE) loss function. Fig. 1 illustrates the characteristics of the results from the proposed approach.

Instead of relying on postprocessing techniques to refine the original frame-wise classification, our Maximization and Restoration approach learns the action segmentation task in an end-to-end manner, avoiding several design issues: slight errors in the estimated boundaries can cause a more drastic decrease in performance; they introduce heuristics or extra hyperparameters that are not robust, which can introduce an additional source of error.

We evaluate our approach on three datasets that are frequently used in action segmentation: 50Salads, GTEA, and Breakfast. The main contributions are as follows:

- We propose a divide-and-conquer method to the action segmentation task. **Maximization** focuses on dominantly increasing the frame-wise accuracy. **Restoration** focuses on correcting inconsistent labels in order to reduce over-segmentation.
- We propose two models (Dilation Passing Network and Temporal Reconstruction Network) that are designed to be effective in their respective steps.
- Extensive evaluations show that our model achieves meaningful results on three challenging datasets: the 50Salads, GTEA, and Breakfast datasets, compared to existing state-of-the-art models.

2. Related work

Action segmentation aims to identify the actions that an actor performs in every frame in a video [14]. Recent research has looked for ways to handle the long-range dependencies of untrimmed videos. Lea et al. [14] proposed temporal convolutional networks (TCNs) that use an encoder-decoder framework. Lei and Todorovic [15] introduced temporal deformable residual networks that incorporate TCNs with deformable convolutions and residual streams to deal with the long-range issue. While these methods are limited to testing on downsampled videos, Farha and Gall [7] proposed the Multi-Stage Temporal Convolutional Network (MS-TCN) that analyzes videos with the full temporal resolution. It consists of multiple stages of dilated convolution sets, with each stage refining the predictions from the previous stage.

Several works have proposed modifications of MS-TCN to further improve performance. Li et al. [9] modified its dilated layers to dual dilated layers and introduced refinement stages, creating MS-TCN++. Chen et al. [10] trained a model that modifies the single stage TCN in MS-TCN to a domain adaptive TCN to segment actions by aligning different domain feature spaces. Gao et al. [11] proposed a global to local search scheme to find the most optimal receptive field combination for MS-TCN and other state-of-the-art models.

Despite improvements in performance, these state-of-the-art techniques still suffer from over-segmentation. MS-TCN somewhat addresses the issue with a truncated mean squared error loss between the log probability of the current scene and the previous scene. To further reduce over-segmentation, some recent studies have resorted to estimating the barriers of where actions start and end with which they refine the outputs from the segmentation models. Wang et al. [5] proposed local barrier pooling that smooths predictions by pooling neighboring frames after weighting them by how many barriers away they are. Ishikawa et al. [12] proposed a boundary regression branch that first regresses the class-agnostic action boundary probabilities, then classifies each segment with majority voting. Huang et al. [13] used graph convolutional networks (GCN) that refine classification as well as boundary regression outputs. However, these strategies face issues where frames that were originally correctly classified can be mislabeled during the refinement stage due to either errors in the boundary regression or noise in the backbone when classifying very ambiguous ground-truth segments.

Other approaches that focus on learning the relationships among actions have been studied as well. Ahn and Lee [16] proposed a Hierarchical Action Segmentation Refiner (HASR) that first extracts hierarchical video representations and then uses a GRU to refine the segmentation outputs from a backbone model. However, since they only propose a refiner network, their overall frame accuracy is generally bounded by the accuracy of the backbone. Such refinement networks are generally designed to reduce over-segmentation but not increase the frame accuracy; thus, it is essential that the accuracy of the backbone is maximized prior to being processed further. Yi et al. [17] proposed Transformer for Action Segmentation (ASFormer), which overcomes potential issues of utilizing the Transformer network [18] for the action segmentation datasets, such as the small size of the training datasets that makes it hard for the Transformer to learn from a large parametric space.

Unlike the above approaches that try to balance between increasing classification accuracy and reducing over-segmentation, our work is the first attempt of proposing a divide-and-conquer strategy that a backbone network exclusively focuses on accurately classifying hard-to-classify classes, and the refiner network focuses on minimizing over-segmentation errors. By prioritizing accuracy in the backbone, our model is able to correctly classify some ambiguous frames that are wrongly classified by other state-of-the-art models. Furthermore, as our refiner network does not use boundary regression to enforce temporal consistency, our model avoids the aforementioned risks of boundary regression-based refinement.

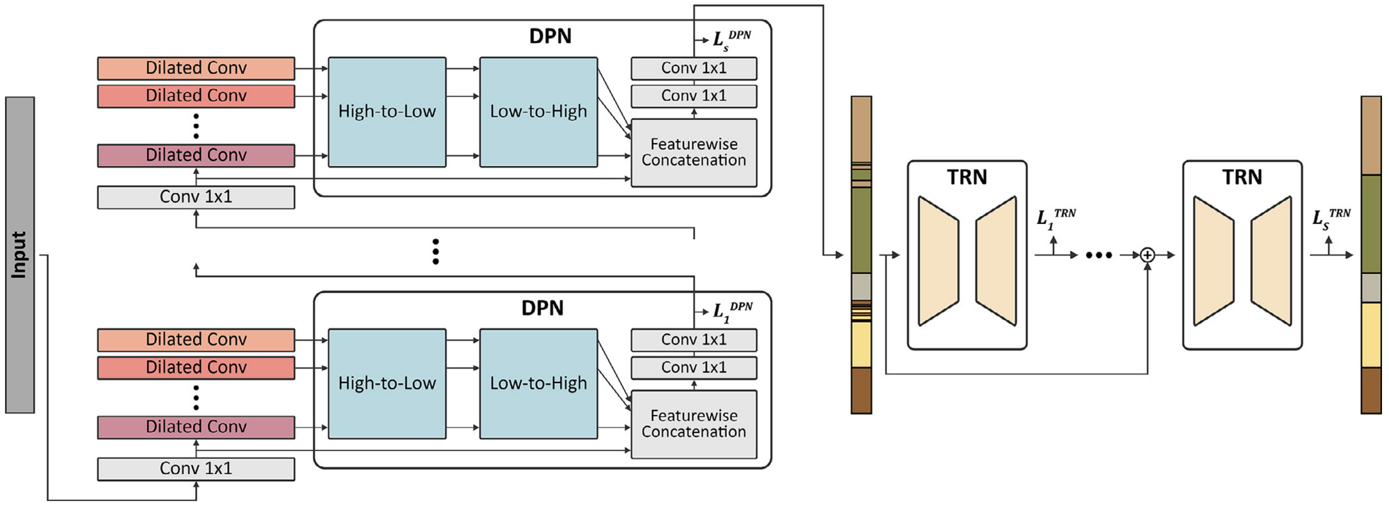

Fig. 2. Overview of the Dilation Passing and Reconstruction Network, composed of the Dilation Passing Network (DPN) and Temporal Reconstruction Network (TRN). DPN incorporates the Multi-stage Temporal Convolutional Network (MS-TCN) with High-to-Low and Low-to-High networks to pass information among dilated layers of different receptive field sizes. TRN restores frames over-segmented by DPN to enforce temporal consistency.

There have been other attempts to analyze human actions in videos. Richard et al. [19] supplemented a recurrent neural network (RNN) with a coarse probability model to propose a fine-to-coarse weak learning system. Ding and Xu [20] addressed the problems of using computationally expensive models like RNNs and proposed a temporal convolutional feature pyramid network. Xu et al. [21] proposed a self-supervised method to incorporate 3D convolutional neural networks to sort sequential unlabeled video samples. Wang et al. [22] proposed a self-supervised GCN module to handle temporal relations within a video. Gao et al. [23] addressed the temporal action detection task with a spatial segmentation approach. Gammuelle et al. [24] proposed a recurrent semi-supervised GAN model with a gated context extractor to capture complex temporal relationships. However, these studies are evaluated with different benchmarks and are therefore beyond our scope.

3. Proposed method

3.1. Overview

This section details the Dilation Passing and Reconstruction Network. Our model is composed of the Dilation Passing Network (DPN) and Temporal Reconstruction Network (TRN), as depicted in Fig. 2. We provide a summary of MS-TCN in Section 3.2, as we use features from each stage as inputs to DPN. We then introduce our DPN, TRN, and loss function in Sections 3.3–3.5, respectively.

3.2. MS-TCN

MS-TCN uses multiple stages of temporal convolutional networks, each composed of dilated residual layers with doubling dilation factors. Each dilated residual layer consists of a dilated convolution, the ReLU activation function, and a 1×1 convolution with a skip-connection.

I3D features [25] are passed through a 1×1 convolutional layer to adjust the input features to a fixed hidden feature dimension *dim_h* to create the hidden feature H_{in}, which is then passed to MS-TCN.

3.3. Dilation passing network

DPN consists of the High-to-Low and Low-to-High networks, as shown in Fig. 3. To dominantly increase the frame classification accuracy, DPN fully utilizes temporal information of different ranges to understand the context of each frame. The High-to-Low network passes longer range dependencies to lower dilation layers and the Low-to-High network passes shorter range dependencies to higher dilation layers. This way, every frame gains a greater understanding on its relations with frames both nearby and farther away.

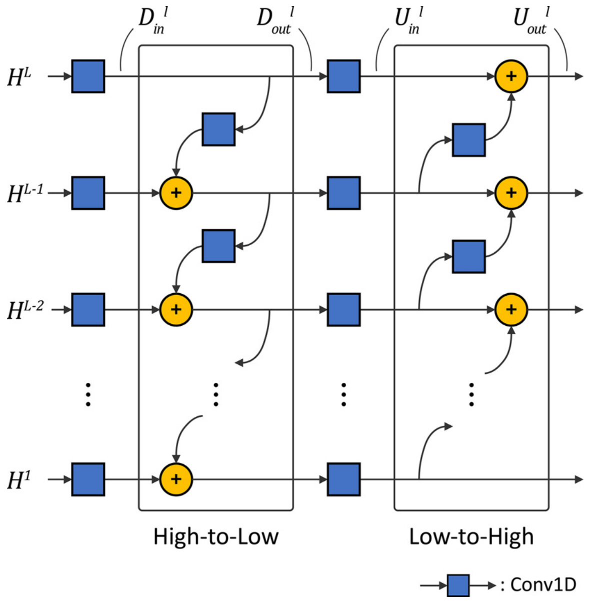

Fig. 3. Overview of the High-to-Low and Low-to-High networks in DPN. Dilated information of different receptive field sizes flows downward and upward.

DPN processes the outputs of the dilated residual layers within MS-TCN through the High-to-Low and Low-to-High networks to pass information among dilation features with different receptive fields. We indicate the output of each dilated residual layer $l \in \{1, 2, \ldots, L\}$ as H^l.

The High-to-Low network propagates the features from the top (highest dilation) to the bottom (lowest dilation). First, the output H^l of each dilated residual layer l from MS-TCN is encoded by a 1×1 convolution with the same filter size to create D_{in}^{l}. Then the High-to-Low features D_{out}^{l} are calculated for each layer from the highest to the lowest layer. We first let $D_{out}^{L} = D_{in}^{L}$. Then, D_{out}^{l+1} of

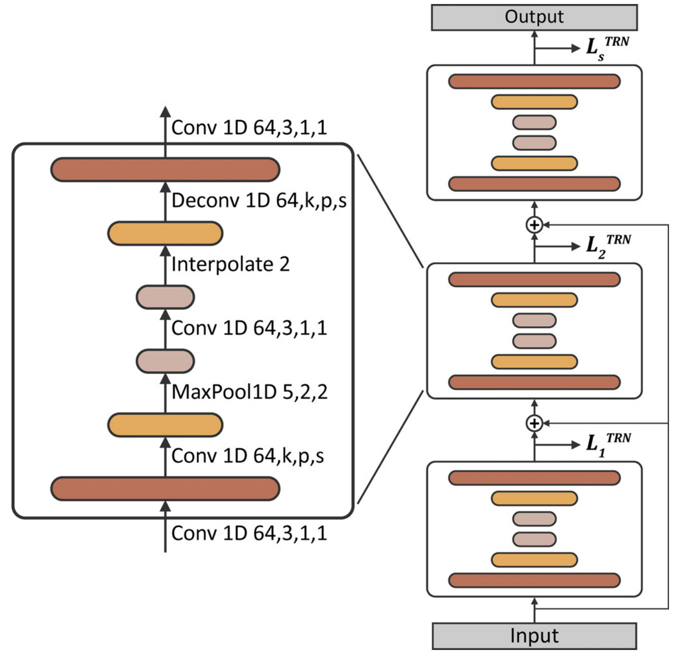

Fig. 4. Architecture of a 3-stage Temporal Reconstruction Network (TRN). Each TRN stage takes frame-wise class probabilities as input and temporally refines the features through a process of encoding and decoding.

the layer above is fed through a 1 × 1 convolutional layer, which is then added to the current layer's features D^l_{in} to create D^l_{out}. This operation is continued for the rest of the layers as follows:

$$D^l_{out} = Conv(D^{l+1}_{out}) + D^l_{in} \tag{1}$$

Then, in the Low-to-High network, we perform similar operations in the opposite direction. The features D^l_{out} are encoded by a 1 × 1 convolution to create U^l_{in}. For each layer, the features U^{l-1}_{in} of the layer below are encoded by a 1 × 1 convolution and added to the current layer's features U^l_{in} to create U^l_{out}, with $U^1_{out} = U^1_{in}$ for the bottom-most layer 1:

$$U^l_{out} = Conv(U^{l-1}_{in}) + U^l_{in} \tag{2}$$

The final hidden features U^l_{out} of each layer are concatenated with the first hidden feature H_{in} and are then reduced to the dimension *dim_h* using a 1 × 1 convolution layer. Finally, to get the frame-wise class probabilities, a final 1 × 1 convolution is applied, followed by a softmax function to produce the final output. For multi-stage DPN, this output is passed through MS-TCN again before it is passed onto the next DPN stage.

3.4. Temporal reconstruction network

DPN aims to correctly classify as many frames as possible. This approach can lead to over-segmentation in that consecutive frames sometimes end up being classified as different classes even though they actually belong to the same class. TRN, as shown in Fig. 4, is designed to overcome this. The encoder of TRN uses convolutional layers to better capture the context around each frame and create an encoded feature that focuses more towards the correctly classified frames while ignoring short, incorrectly classified frames. Then, as the decoder maps the encoded feature back to the original length, the deconvolutional layer passes on each encoded feature to multiple adjacent frames, to ultimately enforce temporal consistency within the frames.

For the training data, the output of DPN may not be significantly over-segmented. In order for the model to be robust against over-segmentation, we first intentionally increase over-segmentation errors for the training data with a segment replacement algorithm. The replacement algorithm first obtains the locations of each action boundary in the output of DPN. At each boundary, two numbers between 0 and $t/2$ are randomly generated, which indicate how many frames to the left and right of the boundary will be replaced. Once those are determined, a chunk of the appropriate length is then randomly chosen from the original output of DPN and replaces the frames near the boundary. For the N_{bd} extra replacements, replacement locations and replacing chunks were both chosen randomly. This way, frames are replaced by potentially incorrect labels, causing inconsistency between frames.

TRN takes as input the frame-wise class probability output of DPN that is modified through the segment replacement algorithm described above. Since TRN aims to reduce over-segmentation, it only uses frame-wise probabilities, not original video features, as input in order to focus solely on the temporal consistency between frame actions. It is composed of an encoder and decoder that reduces and expands this input, respectively. The encoder starts with a 1 × 1 convolution to expand the input to a hidden dimension *dim_h*, followed by the ReLU activation function. To reduce the temporal dimension of the hidden features, a 1D convolution of kernel size k, padding $k/2$, and stride 2 is first applied, producing features that are half the length of the input features. This is then followed by a MaxPool operation that further reduces the length by half, another 1D convolution and another ReLU. The length of the final features from the encoder becomes $\frac{1}{4}$ of the length of the input.

To expand the encoded features back to the video length, the decoder first upsamples the features by a scale factor of two, followed by a dropout layer. Then, a 1D deconvolution with the same parameters as the encoding convolution produces features with the same length as the original video input. Finally, a 1 × 1 convolution is applied to produce the class probabilities.

3.5. Loss function

Loss function for DPN To train DPN, we use the same loss function as in Farha and Gall [7], which combines the classification loss and the temporal mean squared error (T-MSE) loss as follows:

$$\mathcal{L}^{DPN}_s = \mathcal{L}_{cls} + \lambda_{DPN}\mathcal{L}_{T-MSE} \tag{3}$$

where λ_{DPN} is a hyperparameter.

Given the video length T and the number of action classes C, the classification loss $\mathcal{L}_{cls}$ and T-MSE loss $\mathcal{L}_{T-MSE}$ are as follows:

$$\mathcal{L}_{cls} = \frac{1}{T}\sum_t -log(y^{DPN}_{t,c}) \tag{4}$$

$$\mathcal{L}_{T-MSE} = \frac{1}{TC}\sum_{t,c}\hat{\Delta}^2_{t,c} \tag{5}$$

$$\hat{\Delta}_{t,c} = \begin{cases} \Delta_{t,c} & \text{if } \Delta_{t,c} \le \tau \\ \tau & \text{otherwise} \end{cases} \tag{6}$$

$$\Delta_{t,c} = (|logy^{DPN}_{t,c} - logy^{DPN}_{t-1,c}|) \tag{7}$$

where $y^{DPN}_{t,c}$ is the softmax probability of predicted class label c at t from the output of DPN, and τ is a hyperparamterer.

Loss function for TRN The loss function for TRN consists of the classification loss and a weighted reconstruction loss $\mathcal{L}_{WT-MSE}$. $\mathcal{L}_{WT-MSE}$ aims to further reduce over-segmentation errors, defined as follows:

$$\mathcal{L}_{WT-MSE} = \frac{1}{TC}\sum_{t,c}\Phi^2_{t,c} \tag{8}$$

$$\Phi_{t,c} = \frac{1}{2} W_t \left(|logy^{TRN}_{t,c} - logy^{TRN}_{t-1,c}| + |logy^{TRN}_{t,c} - logy^{TRN}_{t+1,c}| \right) \tag{9}$$

$$W_t = \frac{n(B_{t,w})}{\max(n(B_{1:T,w}))} \tag{10}$$

where $y^{TRN}_{t,c}$ is the softmax probability of the predicted class label c at t from the output of TRN, and $n(B_{t,w})$ is the number of boundaries within the segment of length w centered at time t of the input. The weight W_t is the normalized $n(B_{t,w})$ value and aims to emphasize regions in the video that are heavily over-segmented. As our $\mathcal{L}_{WT-MSE}$ loss uses input probabilities to calculate the weight of each frame, it is therefore designed only for models like TRN that take frame-wise class probabilities as inputs.

The overall loss function of TRN is then defined as:

$$\mathcal{L}^{TRN}_{s} = \mathcal{L}_{cls} + \lambda_{TRN} \mathcal{L}_{WT-MSE} \tag{11}$$

where, L_s is the loss function at stage s and λ_{TRN} is a hyperparameter.

While TRN uses our newly proposed $\mathcal{L}_{WT-MSE}$ loss which encourages frames to be classified as the same action as that of both the previous and the next frame, DPN uses the $\mathcal{L}_{T-MSE}$ loss from the works of MS-TCN [7] which encourages frames to be classified similarly to the next frame only. They have shown that $\mathcal{L}_{T-MSE}$ significantly reduces over-segmentation and also improves the frame accuracy by a fair margin. We will show from our experiments that TRN with $\mathcal{L}_{WT-MSE}$ can reduce over-segmentation better than TRN with $\mathcal{L}_{T-MSE}$. For DPN, however, because the main purpose is to maximize the frame accuracy even through inducing more over-segmentation, using $\mathcal{L}_{T-MSE}$ leads to an overall higher performance than using $\mathcal{L}_{WT-MSE}$.

3.6. Implementation details

We first train DPN with $\mathcal{L}^{DPN}_{s}$ for each stage. The underlying layers of MS-TCN are trained together with DPN. Once DPN is finished training, we train TRN using the output of the final stage of DPN as input. Fixing the weights of DPN, we calculate $\mathcal{L}^{TRN}_{s}$ on the output of each stage of TRN. TRN is trained after DPN as TRN requires a much shorter training time. To further improve performance, DPN and TRN may additionally be jointly fine-tuned with a lower learning rate.

DPN We use a four-stage architecture, where each stage consists of eleven dilated convolution layers. The dilation factor, starting from 1, is doubled at each layer up to 512, and the hidden dimension is set to 64 in all layers. The dilated layers all have a kernel size of 3, while the High-to-Low and Low-to-High network layers have a kernel size of 1. As in Farha and Gall [7], we set $\tau = 4$ and $\lambda_{DPN} = 0.15$ for the loss function. The Adam optimizer is used with a learning rate of 5e-4 for 50Salads and Breakfast and 1e-3 for GTEA.

TRN We use a three-stage architecture for TRN. All 1D convolutions have a hidden dimension of 64. The Adam optimizer is used with a learning rate of 1e-4 for all datasets. For segment replacement, $t_{\max}$ is set to 60 for 50Salads, 10 for GTEA, and 50 for Breakfast, and $t_{\min} = \frac{1}{5} t_{\max}$. For the encoding and decoding convolutions, we set the kernel size k to 101 for 50Salads, 71 for GTEA, and 161 for Breakfast. The segment length parameter w for $\mathcal{L}^{TRN}_{s}$ is set equal to the kernel size k for each dataset. Hyperparameters $t_{\max}$ and k are roughly correlated to the action lengths of each dataset; GTEA has the smallest parameters as the actions are short.

Joint fine-tuning The Adam optimizer is used with a learning rate of 1e − 5 for 50Salads and 1e-7 for GTEA and Breakfast.

Table 1
Comparison of baseline (MS-TCN), DPN, DPRN with $\mathcal{L}_{T-MSE}$, DPRN with $\mathcal{L}_{WT-MSE}$ and DPRN jointly fine-tuned on 50Salads.

50Salads	F1@{10,25,50}			Edit	Acc
MS-TCN [7]	76.3	74.0	64.5	67.9	80.7
DPN	31.5	30.4	26.9	25.6	86.9
DPRN ($\mathcal{L}_{T-MSE}$)	86.3	84.4	78.6	80.3	86.8
DPRN ($\mathcal{L}_{WT-MSE}$)	87.2	85.8	79.1	80.8	86.8
DPRN (jointly tuned)	**87.8**	**86.3**	**79.4**	**82.0**	**87.2**
MS-TCN + TRN	82.5	80.6	70.4	75.3	80.6
MS-TCN + layer concat.	42.8	41.4	36.0	37.1	82.6

The bold values indicate the highest value for each metric (column) in that table/comparison.

4. Experiment

Datasets The proposed model was evaluated using three datasets: 50Salads [26], GTEA [27], and Breakfast [28].

The **50Salads** dataset consists of 50 videos in which 25 people prepare two different salads. There are 17 action classes related to cutting, placing vegetables, adding ingredients, and serving. To evaluate the proposed model, five-fold cross validation was used as in Stein and McKenna [26].

The **GTEA** dataset consists of 28 videos of 4 people performing daily activities. The dataset has 11 action classes including the background class, each of which has 20 instances on average. 4-fold cross validation was used.

The **Breakfast** dataset contains 1712 videos of 10 activities related to breakfast preparation in 18 different kitchens. The dataset aims to classify the 48 action units within the videos, each of which has 6 action instances on average. The standard 4-fold cross validation was used as in Kuehne et al. [28].

Evaluation metrics To evaluate the proposed model, we use frame-wise accuracy, a commonly used evaluation metric that measures how many frames are correctly classified in a video. This metric cannot measure the degree of over-segmentation; we therefore also report the segmental edit score and the segmental F1 score. The edit score penalizes errors that come from over-segmentation. The F1 score aims to measure the prediction quality [14]; we use temporal intersection of union (tIoU) thresholds of 10%, 25%, and 50%. The action start and action end classes of 50Salads and the background class of GTEA were ignored when calculating the F1 and edit scores.

4.1. Study on DPRN

This section investigates the effect of the proposed method using 50Salads. We show MS-TCN as a baseline and compare it with incrementally integrated structures as follows: (1) DPN: incorporating the High-to-Low and Low-to-High networks with the dilated outputs from 4-stage MS-TCN; (2) DPRN: adding the 3-stage TRN with the classification and temporal mean squared losses as in Farha and Gall [7] to DPN; (3) DPRN w/ loss: using $\mathcal{L}_{WT-MSE}$ instead of $\mathcal{L}_{T-MSE}$ for TRN; and (4) DPRN (jointly tuned): jointly tuning DPRN with a lower learning rate after both models are trained. To verify the novelty of DPN and TRN, we also compare MS-TCN with (5) Appended TRN: attaching TRN with $\mathcal{L}_{WT-MSE}$ to the end of MS-TCN and (6) Layer concatenation: directly concatenating the features H^l from each dilation layer l of MS-TCN with the first hidden feature H_{in}, and then reducing the dimension to get the probabilities (essentially MS-TCN with DPN without the High-to-Low and Low-to-High networks). Table 1 shows the results.

DPN increases the frame-wise accuracy by a significant amount, as information on different dilations are passed from layer to layer. At the same time, the F1 and edit scores decrease significantly from the MS-TCN baseline, indicating over-segmentation. A frame

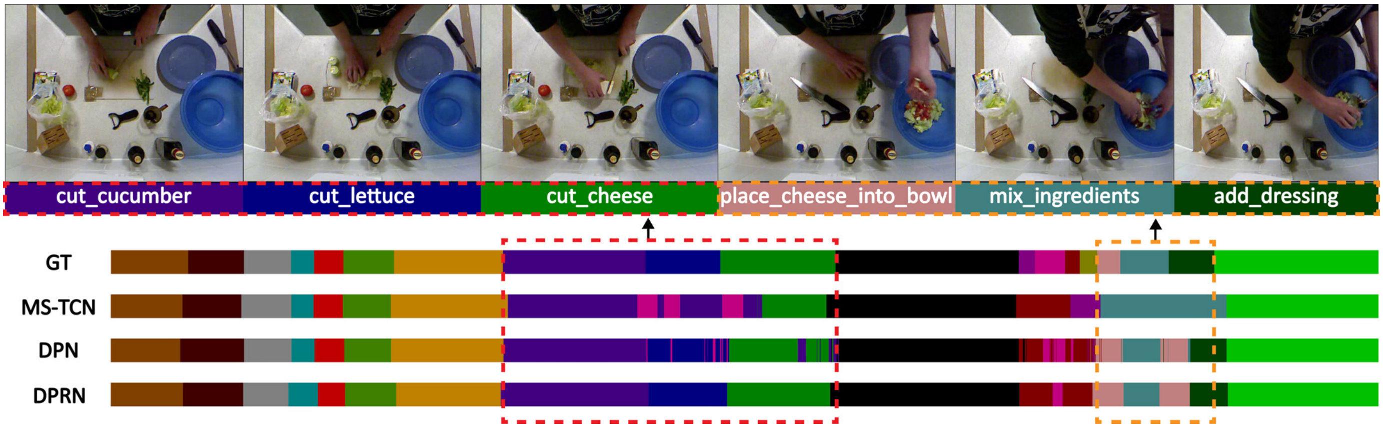

Fig. 5. Qualitative results on 50Salads. Each row represents the frame-wise results of the following, respectively: Ground-truth labels (GT), results from MS-TCN, results from DPN only, and results from DPRN. It is notable that DPN is capable of obtaining several correct frames in regions that MS-TCN labels incorrectly. However, this comes at the cost of increasing oversegmentation, which TRN corrects as shown by the results of DPRN.

correctly classified within an otherwise wrongly classified chunk only minutely increases the accuracy but greatly decreases the F1 and edit as the chunk is divided into multiple segments.

TRN fulfills its role well by correcting over-segmentation issues from DPN, as seen through the F1 and edit scores of (2). The frame-wise accuracy however is almost unchanged, showing that the accuracy of TRN is roughly bounded by the accuracy of its backbone, DPN. This phenomenon can also be seen in previous works [5,12,16], showing why our approach of Maximization and Reconstruction is essential for significantly improving the performance.

The weighted temporal mean squared error loss ($\mathcal{L}_{WT-MSE}$) further improves on all scores except for accuracy, which was maintained. As $\mathcal{L}_{WT-MSE}$ mainly penalizes predictions in over-segmented regions, it is successful in improving segmental scores while minimally affecting the classification score.

Fine-tuning DPRN on 50Salads with a lower learning rate ($lr = 1\mathrm{e}-5$) further improves all scores by at most 1.2 percent. However, this requires searching for the learning rate parameter that can produce optimal results.

Appending TRN to MS-TCN shows significant improvements over the original MS-TCN, improving the F1/edit scores by at least 5.9. This therefore verifies its effectiveness in reducing oversegmentation and potential for application to other action segmentation models. Similarly to DPN, layer concatenation of MS-TCN layers increases the frame-wise accuracy, but does so by trading off the F1 and edit scores. This is expected as concatenating different dilation information focuses more on each frame individually rather than the relationships among frames. However, simply concatenating layers results in an accuracy inferior by 4.3 percent to that of DPN, proving the effectiveness of our DPN architecture for the Maximization approach.

Fig. 5 visualizes the segmentation results for DPRN on 50Salads. As seen in the results, MS-TCN often misclassifies ambiguous action classes such as æcut_cucumberg, æcut_lettuceg, and æcut_cheeseg. Compared to MS-TCN, DPN performs more accurately in terms of frame-wise classifications; however, there still are some occasional misclassifications, causing over-segmentation. DPRN produces smoother results, removing action frames that are classified differently within an action chunk. Similar results can be seen in the results of GTEA and Breakfast in Fig. 6. As with 50Salads, DPN performs well in terms of the frame-wise accuracy while DPRN successfully reduces over-segmented regions. Based on Table 1 and Figs. 5 and 6, it can be concluded that TRN corrects sparse misclassifications leading to higher F1 and edit scores while maintaining frame-wise accuracy.

As TRN reduces then increases the temporal dimension, very short segments from DPN may occasionally be incorrectly removed. One example can be seen in the GTEA results (Fig. 6.a) where a short red segment by DPN in the middle of the video is removed by TRN. As the encoder of TRN encodes the input feature into a feature $\frac{1}{4}$ of the original length, extremely short actions may get ignored in the process. However, considering the frame rate of the input features (15 fps), short actions of even just 1 second would still take up around 4 frames in the encoded feature. Rather, if DPN correctly identifies an action segment, but inaccurately detects it to be much shorter than the true length, TRN may mistakenly remove such segments.

4.2. Ablation study

To analyze the influence of hyperparameters, we conducted ablation studies on the number of stages of DPN and TRN, and the kernel sizes of TRN.

Table 2 (top) shows the performance for different numbers of DPN stages with and without TRN. The 4-stage DPRN produces the best performance overall, showing that simply stacking more stages does not always increase performance. Nonetheless, other numbers of stages still outperform the baseline MS-TCN; the 1-stage DPRN shows a higher accuracy of 83.4 compared to MS-TCN with 80.7. For all models, TRN successfully refines the outputs of DPN to improve the F1 and edit scores.

Table 2 (bottom) tabulates the results of DPRN with different numbers of TRN stages on 50Salads. The results show robust performances regardless of the number of stages. The 3-stage TRN does not always provide the best results on all metrics; however, we chose the 3-stage TRN for our state-of-the-art comparisons, as it gives the highest average value on all metrics.

We also investigated whether changes in the kernel size of TRN affects over-segmentation. Fig. 7 shows the scores for different kernel sizes. We can see that the kernel size does not affect the frame-wise accuracy of TRN but does influence the extent of over-segmentation as can be seen from the changing F1 and edit scores. Even for the worst performing and smallest kernel size of $k = 21$, the F1@(10, 25, 50), edit, and accuracy scores are 82.1, 80.4, 73.8, 74.4, and 86.8, respectively, showing improvements from MS-TCN. Performance further increases and then stabilizes as the kernel size is increased. The best kernel size is $k = 101$ as it results in the best F1 and edit scores while conserving the accuracy. As the kernel size becomes greater than $k > 121$, the overall scores tend to slightly decrease. As the kernel gets larger, it loses the ability to focus on the most essential local context around each frame and in-

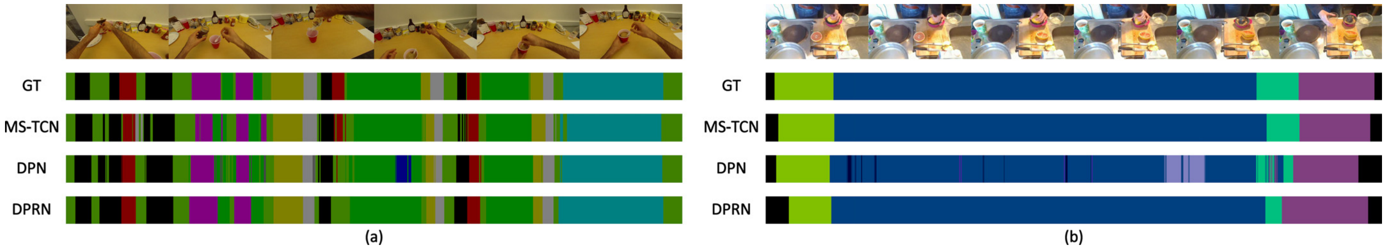

Fig. 6. Qualitative results on (a) GTEA and (b) Breakfast.

Table 2
Study on the number of DPN stages (top) and TRN stages(bottom) on 50Salads.

	DPN only						DPRN (using 3-stage TRN)				
DPN	F1@{10,25,50}			Edit	Acc		F1@{10,25,50}			Edit	Acc
1-stage	21.8	20.4	17.7	16.1	84.6		81.9	79.9	72.1	75.6	83.4
2-stages	29.3	27.7	24.4	22.2	85.9		84.5	82.7	75.6	79.2	85.3
3-stages	31.3	30.0	26.5	24.6	86.2		85.0	83.2	77.1	79.1	86.1
4-stages	31.5	30.4	26.9	25.6	86.9		**87.2**	**85.8**	**79.1**	**80.8**	**86.8**
5-stages	36.3	35.0	31.0	28.4	86.3		85.5	84.1	77.5	79.8	86.2
							DPRN (using 4-stage DPN)				
						TRN	F1@{10,25,50}			Edit	Acc
						1-stage	86.8	85.0	78.8	80.2	**87.0**
						2-stages	86.8	85.3	78.8	**81.3**	86.9
						3-stages	**87.2**	**85.8**	**79.1**	80.8	86.8
						4-stages	**87.2**	85.3	78.8	80.8	86.7

The bold values indicate the highest value for each metric (column) in that table/comparison.

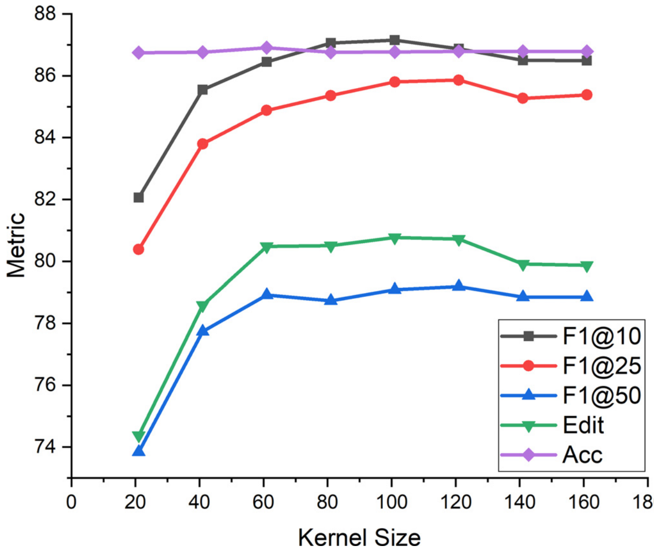

Fig. 7. Study on TRN's kernel size on 50Salads.

stead looks at each frame with a general context. This hinders the encoder and decoder from making relevant corrections for temporal consistency and results in a sub-optimal over-segmentation level. Nonetheless, TRN performs well within a reasonable range of the kernel size.

In general, these results show the robustness of our model to changes to its hyperparameters. Further ablation studies (e.g., comparison of λ values, effects of segment replacement and its $t_{\max}$) are provided in the Appendix.

4.3. Study on other backbones

DPN works with features extracted from varying temporal resolutions to better capture the temporal context of each frame. Consequently, it requires models such as MS-TCN where the input feature is processed through multiple layers of different dilations. To investigate whether DPRN works well across various backbone models, we experimented with two other backbone models: MS-TCN++ and ETSN. Other models such as BCN or ASRF that propose an extension or variation of MS-TCN were not considered, as they, much like our model, are models that entail extra computation on top of the dilated features. Furthermore, models such as Bi-LSTM or GTRM that build upon recurrent networks were also not considered since such backbones do not explicitly produce features of multiple temporal resolutions.

Table 3
Study on DPRN with different backbones on 50Salads.

50Salads	F1@{10,25,50}			Edit	Acc
MS-TCN [7]	76.3	74.0	64.5	67.9	80.7
MS-TCN & DPRN	87.2	85.8	79.1	80.8	86.8
ETSN [29]	85.2	83.9	75.4	78.8	82.0
ETSN & DPRN	85.9	84.5	77.4	79.7	85.9
MS-TCN+ [9]	80.7	78.5	70.1	74.3	83.7
MS-TCN+ & DPRN	85.5	84.0	76.4	78.2	85.0

As shown in Table 3, while DPRN performs best with MS-TCN as the backbone, it also shows meaningful improvements when used on other backbone models as well. Differences in the standalone performances of these models become less significant when they are used as DPRN's backbone. In fact, MS-TCN performs the worst out of the three models when used alone, but outperforms the others when used with DPRN. ETSN and MS-TCN++ modify the dilated layers of MS-TCN to capture multiple temporal dependencies within every layer. This overlap of functionality with DPN may have either induced over-fitting or caused the input features of DPN to be less comprehensive, leading to a worse performance than the MS-TCN backbone. Nonetheless, our results show that DPRN is applicable to any other backbone as long as the backbone generates features of multiple temporal resolutions for the High-to-Low and Low-to-High networks of DPN to work with.

Table 4
Study on the effect of model capacity on 50Salads. The number of parameters and FLOPs of our model are compared with other models. The FLOPs are calculated using an input feature from 50Salads with the length fixed at 1000 frames.

50Salads	#Params	FLOPs	F1@{10,25,50}			Edit	Acc
DPRN	4.1 M	7.0G	87.2	85.8	79.1	80.8	86.8
- DPN	1.6 M	3.2G	31.5	30.4	26.9	25.6	86.9
- TRN (3 stages)	2.5 M	3.8G	87.2	85.8	79.1	80.8	86.8
- TRN (2 stages)	1.7 M	2.6G	86.8	85.3	78.8	81.3	86.9
- TRN (1 stage)	0.8 M	1.2G	86.8	85.0	78.8	80.2	87.0
MS-TCN (Reported) [7]	0.8 M	1.6G	76.3	74.0	64.5	67.9	80.7
MS-TCN (Implemented)	0.8 M	1.6G	71.6	68.7	59.1	63.0	81.1
MS-TCN, *dim_h*=96	1.7 M	3.4G	61.8	59.2	52.5	55.2	82.7
MS-TCN, #*layers*=16	1.2 M	2.4G	57.3	54.3	46.1	48.5	82.0
MS-TCN, #*stages*=8	1.5 M	3.0G	71.0	68.1	59.9	63.3	82.1
ASFormer [17]	1.1 M	2.2G	85.1	85.4	79.3	81.9	85.9
ASRF [12]	1.3 M	2.6G	84.9	83.5	77.3	79.3	84.5
BCN [5]	12.8 M	25.6G	82.3	81.3	74.0	74.3	84.4
ASRF + HASR [16]	19.2 M	25.8G	82.3	81.3	74.0	74.3	84.4
MS-TCN+ [9]	1.0 M	2.0G	80.7	78.5	70.1	74.3	83.7
ETSN [29]	0.8 M	1.6G	85.2	83.9	75.4	78.8	82.0

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

	50Salads				
	F1@{10,25,50}			Edit	Acc
Bi-LSTM [30]	62.6	58.3	47.0	55.6	55.7
MS-TCN [7]	76.3	74.0	64.5	67.9	80.7
ETSN [29]	85.2	83.9	75.4	78.8	82.0
G2L [11]	80.3	78.0	69.8	73.4	82.2
GTRM [13]	75.4	72.8	63.9	67.5	82.6
DA [10]	82.0	80.1	72.5	75.2	83.2
MS-TCN+ [9]	80.7	78.5	70.1	74.3	83.7
ASRF + HASR [16]	86.6	85.7	78.5	81.0	83.9
BCN [5]	82.3	81.3	74.0	74.3	84.4
ASRF [12]	84.9	83.5	77.3	79.3	84.5
ASFormer [17]	85.1	85.4	79.3	81.9	85.9
DPRN	87.2	85.8	79.1	80.8	86.8
DPRN (jointly tuned)	**87.8**	**86.3**	**79.4**	**82.0**	**87.2**

	GTEA					Breakfast				
	F1{10,25,50}			Edit	Acc	F1{10,25,50}			Edit	Acc
GRU [19]	–	–	–	–	–	–	–	–	60.6	–
Bi-LSTM [30]	66.5	59.0	43.6	–	55.5	–	–	–	–	–
MS-TCN [7]	85.8	83.4	69.8	79.0	76.3	52.6	48.1	37.9	61.7	66.3
ETSN [29]	91.1	90.0	77.9	86.2	78.2	74.0	69.0	56.2	70.3	67.8
G2L [11]	–	–	–	–	–	**76.3**	69.9	54.6	74.5	70.8
GTRM [13]	–	–	–	–	–	57.5	54.0	43.3	58.7	65.0
DA [10]	90.5	88.4	76.2	85.8	80.0	74.2	68.6	56.5	73.6	71.0
MS-TCN + [9]	88.8	85.7	76.0	83.5	80.1	64.1	58.6	45.9	65.6	67.6
ASRF + HASR [16]	89.2	87.2	74.8	84.5	76.9	74.7	69.5	57.0	71.9	69.4
SSTDA + HASR [16]	90.9	88.6	76.4	87.5	78.7	–	–	–	60.6	–
BCN [5]	88.5	87.1	77.3	84.4	79.8	68.7	65.5	55.0	66.2	70.4
ASRF [12]	89.4	87.8	79.8	83.7	77.3	74.3	68.9	56.1	72.4	67.6
ASFormer [17]	90.1	88.8	79.2	84.6	79.7	76.0	**70.6**	57.4	75.0	**73.5**
DPRN	92.6	91.3	81.0	89.9	81.7	75.2	70.2	57.4	74.7	71.6
DPRN (jointly tuned)	**92.9**	**92.0**	**82.9**	**90.9**	**82.0**	75.6	70.5	**57.6**	**75.1**	71.7

The bold values indicate the highest value for each metric (column) in that table/comparison.

4.4. Study on model capacity

The High-to-Low and Low-to-High networks of DPN introduce roughly $4SL$ 1×1 convolutional networks, where S is the number of stages and L is the number of layers in each stage. This results in DPN having roughly 1.6 M parameters when using 4 stages and 10 dilation layers, while MS-TCN has around 0.8 M parameters. To show that the increase in performance of our model is due to our design choices rather than an increase in model capacity, we compared our model with modifications of MS-TCN where we increased the number of parameters by increasing three different hyperparameters: hidden dimension, number of stages, and number of layers.

The results, provided in Table 4, show that the number of parameters of the three modifications of MS-TCN are roughly the same as that of DPN. However, although the modifications show a slight increase in frame-wise accuracy compared to the original MS-TCN, the improvements are much less significant than the improvement shown by DPN. This shows that the High-to-Low and Low-to-High networks are particularly effective in capturing the relevant temporal features for more accurate frame classification.

Furthermore, we have tabulated the number of parameters and FLOPs (floating point operations) of state-of-the-art models for reference. FLOPs is a widely used metric in computer vision and action segmentation [17] to compare computational time while factoring out differences in hardware resources. Compared to models such as ASFormer, ASRF, and MS-TCN that perform fairly well with around 1M parameters and 2G FLOPs, DPRN requires a greater number of parameters and FLOPs, being composed of two explicit models. However, the increased model capacity is within an acceptable range, remaining much smaller than larger models such as BCN and ASRF + HASR. Furthermore, TRN takes up a large portion of the model capacity. However, considering that TRN performs as well even with fewer stages, as shown in Table 2, the number of stages in TRN can be reduced in order to minimize model capacity without much influence on performance.

4.5. Comparison with the state-of-the-art

This section compares DPRN with state-of-the-art models on three datasets: 50Salads, GTEA, and Breakfast. Table 5 tabulates the results.

For baseline comparisons, we referred to Bi-LSTM [30] for 50Salads and GTEA and GRU [19] for Breakfast. For recent state-of-the-art models, MS-TCN [7], GTRM [13], MS-TCN++ [9], BCN [5], ASRF [12], DA [10], G2L [11], ETSN [29], HASR [16] and ASFormer [17] were used. We report the results of jointly fine-tuned DPRN separately for thorough comparison.

In terms of the frame-wise accuracy, DPRN outperforms all other models by at least 0.9 and 1.6 percent on 50Salads and GTEA, respectively, and achieves the second highest accuracy for Breakfast, trailing behind ASFormer by 1.9 percent. On GTEA, DPRN outperforms all the aforementioned models on all metrics, while on 50Salads, it outperforms the models on the accuracy and the segmental F1@10 and F1@25 scores, but is second to ASFormer for the F1@50 and edit scores. On Breakfast, DPRN performs sub-optimally to ASFormer, but remains the second best performing model for most metrics.

Joint fine-tuning improves the performance of all metrics by at most 1.2, 1.9 and 0.4 percent for each dataset, respectively. Consequently, jointly tuned DPRN outperforms all models on both 50Salads and GTEA, and performs similarly to ASFormer on all metrics except the frame accuracy on Breakfast.

5. Conclusion

In this paper, we presented a model called Dilation Passing and Reconstruction Network (DPRN) for temporal action segmentation. We proposed a two-stage strategy to perform action segmentation: **Maximization** of frame-wise classification accuracy, and **Restoration** of over-segmentation. This strategy is well-suited for improving both frame classification and segmentation performance as post-processing networks that refine over-segmentation exhibit frame accuracies that are bounded by their backbones. The Dilation Passing Network (DPN) first maximizes the frame-wise classification accuracy by passing information of different dilations to model long-range dependencies within the video. Then, the Temporal Reconstruction Network (TRN) greatly reduces over-segmentation errors by encoding and decoding the segmentation outputs in the temporal dimension. We also introduced a weighted temporal reconstruction loss that further reduces over-segmentation errors. Through the evaluations of each sub-component of our model, we have shown that DPN is capable of significantly increasing the accuracy, and TRN is able to minimize over-segmentation while maintaining the accuracy. Furthermore, through empirical evaluations on three datasets, we have shown that DPRN outperforms state-of-the-art methods.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by the Technology Innovation Program funded By the Ministry of Trade, Industry & Energy (MOTIE, Korea) under Grant 20007058; the National Research Foundation of Korea Grant funded by the Korean Government (MSIT) under Grant NRF-2016R1A5A1938472; and the “Robot Social Interaction Technology Development” project funded by KT (KT award B210000715).

Appendix A. Impact of λ_{DPN} and λ_{TRN}

We studied how λ_{DPN} and λ_{TRN} affect the performance of DPRN (Table A.6). For DPN, although $\lambda_{DPN} = 0.15$ obtains the highest accuracy, the F1 and edit scores are lower compared to those for $\lambda_{DPN} = 0.05$ and $\lambda_{DPN} = 0.25$. As previously mentioned, these results show that the model becomes overly segmented as it starts to obtain correct predictions for sparse frames within action chunks that are incorrectly predicted. As over-segmentation is handled by TRN, $\lambda_{DPN} = 0.15$ results in the best overall performance for DPRN.

For λ_{TRN}, DPRN with $\lambda_{TRN} = 1.0$ obtains the highest score on all metrics except the F1@50. These scores are closely followed by those for $\lambda_{TRN} = 0.6$. For the F1@50 score, however, $\lambda_{TRN} = 1.8$ results in the highest performance. Overall, changes in λ_{TRN} barely affect the high performance of DPRN, with differences of at most 0.6 for the scores from the best overall $\lambda_{TRN} = 1.0$. This shows that DPRN is robust to the λ_{TRN} parameter.

Table A.6
Effect of λ_{DPN} (top) and λ_{TRN} (bottom) on 50Salads.

	DPN only					DPRN				
λ_{DPN}	F1@{10,25,50}			Edit	Acc	F1@{10,25,50}			Edit	Acc
0.05	48.0	45.7	40.3	38.1	84.5	83.8	82.0	75.3	76.6	84.8
0.15	31.5	30.4	26.9	25.6	**86.9**	**87.2**	**85.8**	**79.1**	**80.8**	86.8
0.25	52.9	50.0	44.1	43.4	84.4	84.2	82.5	74.8	77.7	84.6
						DPRN				
					λ_{TRN}	F1@{10,25,50}			Edit	Acc
					0.2	87.0	85.5	79.1	80.2	86.7
					0.6	87.1	**85.8**	79.2	80.6	**86.8**
					1.0	**87.2**	**85.8**	79.1	**80.8**	**86.8**
					1.4	86.9	85.6	79.0	80.3	86.7
					1.8	87.1	85.3	**79.3**	80.3	86.5

The bold values indicate the highest value for each metric (column) in that table/comparison.

Appendix B. Impact of segment replacement and $t_{\max}$

To investigate the influence of segment replacement, we compared the performance of DPRN with and without segment replacement, and with different $t_{\max}$ values (Table B.7). Using segment replacement improves the F1 and edit scores, with increases ranging from 0.9 to 2.2. The accuracy, however, is slightly greater without segment replacement. Nonetheless, DPRN without segment replacement still shows meaningful results in comparison to state-of-the-art models.

While the overall metrics exhibit the robustness of the proposed model in regard to $t_{\max}$, the frame-wise accuracy does slightly decrease as $t_{\max}$ increases. This is because as the replacement size increases, the output of DPN, which primarily aims to

Table B.7
Effect of segment replacement (top) and parameter $t_{\max}$ (bottom) on 50Salads.

SR	F1@{10,25,50}			Edit	Acc
with	**87.2**	**85.8**	**79.1**	**80.8**	86.8
without	85.9	84.9	77.6	78.6	**87.3**
$t_{\max}$	**F1**@{10,25,50}			**Edit**	**Acc**
30	86.7	85.5	78.5	80.4	**87.0**
60	**87.2**	**85.8**	79.1	**80.8**	86.8
90	**87.2**	85.7	**79.3**	80.5	86.5
120	86.5	85.5	77.4	78.7	86.1

The bold values indicate the highest value for each metric (column) in that table/comparison.

increase the accuracy, becomes more corrupted. $t_{\max} = 60$ seems to give the best overall F1 and edit scores.

References

[1] D. Kim, B.B. Kang, K.B. Kim, H. Choi, J. Ha, K.-J. Cho, S. Jo, Eyes are faster than hands: a soft wearable robot learns user intention from the egocentric view, Sci. Robot. 4 (26) (2019) eaav2949.

[2] M. Devanne, S. Berretti, P. Pala, H. Wannous, M. Daoudi, A. Del Bimbo, Motion segment decomposition of RGB-D sequences for human behavior understanding, Pattern Recognit. 61 (2017) 222–233.

[3] K. Doshi, Y. Yilmaz, Online anomaly detection in surveillance videos with asymptotic bound on false alarm rate, Pattern Recognit. 114 (2021) 107865.

[4] O.P. Popoola, K. Wang, Video-based abnormal human behavior recognition—Areview, IEEE Trans. Syst., Man, Cybern., Part C (Applications and Reviews) 42 (6) (2012) 865–878.

[5] Z. Wang, Z. Gao, L. Wang, Z. Li, G. Wu, Boundary-aware cascade networks for temporal action segmentation, in: European Conference on Computer Vision, Springer, 2020, pp. 34–51.

[6] K. He, X. Zhang, S. Ren, J. Sun, Deep residuallearning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[7] Y.A. Farha, J. Gall, MS-TCN: multi-stage temporal convolutional network for action segmentation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 3575–3584.

[8] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu, WaveNet: a generative model for raw audio, arXiv preprint arXiv:1609.03499(2016).

[9] S.-J. Li, Y. AbuFarha, Y. Liu, M.-M. Cheng, J. Gall, MS-TCN++: multi-stage temporal convolutional network for action segmentation, in: IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020, doi:10.1109/TPAMI.2020.3021756.

[10] M.-H. Chen, B. Li, Y. Bao, G. AlRegib, Action segmentation with mixed temporal domain adaptation, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2020, pp. 605–614.

[11] S.-H. Gao, Q. Han, Z.-Y. Li, P. Peng, L. Wang, M.-M. Cheng, Global2local: efficient structure search for video action segmentation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 16805–16814.

[12] Y. Ishikawa, S. Kasai, Y. Aoki, H. Kataoka, Alleviating over-segmentation errors by detecting action boundaries, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2021, pp. 2322–2331.

[13] Y. Huang, Y. Sugano, Y. Sato, Improving action segmentation via graph-based temporal reasoning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 14024–14034.

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

[15] P. Lei, S. Todorovic, Temporal deformable residual networks for action segmentation in videos, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 6742–6751.

[16] H. Ahn, D. Lee, Refining action segmentation with hierarchical video representations, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 16302–16310.

[17] F. Yi, H. Wen, T. Jiang, Asformer: transformer for action segmentation(2021).

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Proceedings of the International Conference on Neural Information Processing Systems, 2017, pp. 6000–6010.

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

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

[21] D. Xu, J. Xiao, Z. Zhao, J. Shao, D. Xie, Y. Zhuang, Self-supervised spatiotemporal learning via video clip order prediction, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 10334–10343.

[22] D. Wang, D. Hu, X. Li, D. Dou, Temporal relational modeling with self-supervision for action segmentation, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 2729–2737.

[23] Z. Gao, L. Wang, Q. Zhang, Z. Niu, N. Zheng, G. Hua, Video imprint segmentation for temporal action detection in untrimmed videos, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 8328–8335.

[24] H. Gammulle, S. Denman, S. Sridharan, C. Fookes, Fine-grained action segmentation using the semi-supervised action GAN, Pattern Recognit. 98 (2020) 107039.

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

[26] S. Stein, S.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, 2013, pp. 729–738.

[27] A. Fathi, X. Ren, J.M. Rehg, Learning to recognize objects in egocentric activities, in: CVPR 2011, IEEE, 2011, pp. 3281–3288.

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

[29] Y. Li, Z. Dong, K. Liu, L. Feng, L. Hu, J. Zhu, L. Xu, S. Liu, et al., Efficient two-step networks for temporal action segmentation, Neurocomputing 454 (2021) 373–381.

[30] 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: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1961–1970.

Junyong Park received the B.S. and the M.S. degrees in computer science at KAIST, Daejeon, Republic of Korea, in 2018 and 2020, respectively, where he is currently pursuing the Ph.D. degree with the School of Computing. His research interests include human-robot interactions, computer vision, robotics, and machine learning.

Daekyum Kim received the B.S. degree in mechanical engineering from the University of California, Los Angeles, Los Angeles, CA, USA, in 2015, and received the Ph.D. degree in Computer Science at KAIST, Daejeon, Republic of Korea, in 2021. He is currently a Postdoctoral Research Fellow at the John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA. His research interests are in the areas of machine learning, computer vision, and robotics.

Sejoon Huh received the B.S. and the M.S. degrees in computer science at KAIST, Daejeon, Republic of Korea, in 2019 and 2021, respectively, where he is currently pursuing the Ph.D. degree with the School of Computing. His research interests are brain-computer interface applications and applied machine learning.

Sungho Jo received the B.S. degree in the school of mechanical & aerospace engineering from the Seoul National University, Seoul, Republic of Korea, the S.M. degree in mechanical engineering, and the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, MA, in 1999, 2001, and 2006 respectively. While pursuing the Ph.D., he was associated with the Computer Science and Artificial Intelligence Laboratory (CSAIL), Laboratory for Information Decision and Systems (LIDS), and Harvard-MIT HST NeuroEngineering Collaborative. Before joining the faculty with KAIST, he worked as a postdoctoral researcher with MIT media laboratory. Since December in 2007, he has been with the School of Computing, KAIST, where he is currently professor. His research interests include intelligent robots, neural interfacing computing, and wearable computing.