Learning to Discriminate Information for Online Action Detection

Hyunjun Eun[1,4*] Jinyoung Moon[2] Jongyoul Park[2] Chanho Jung[3] Changick Kim[1]

[1]Korea Advanced Institute of Science and Technology (KAIST)
[2]Electronics and Telecommunications Research Institute (ETRI)
[3]Hanbat National University [4]SK Telecom

Abstract

From a streaming video, online action detection aims to identify actions in the present. For this task, previous methods use recurrent networks to model the temporal sequence of current action frames. However, these methods overlook the fact that an input image sequence includes background and irrelevant actions as well as the action of interest. For online action detection, in this paper, we propose a novel recurrent unit to explicitly discriminate the information relevant to an ongoing action from others. Our unit, named Information Discrimination Unit (IDU), decides whether to accumulate input information based on its relevance to the current action. This enables our recurrent network with IDU to learn a more discriminative representation for identifying ongoing actions. In experiments on two benchmark datasets, TVSeries and THUMOS-14, the proposed method outperforms state-of-the-art methods by a significant margin. Moreover, we demonstrate the effectiveness of our recurrent unit by conducting comprehensive ablation studies.

1. Introduction

Temporal action detection [3, 20, 31, 34, 35] has been widely studied in an offline setting, which allows making a decision for the detection after fully observing a long untrimmed video. This is called offline action detection. In contrast, online action detection aims to identify ongoing actions from streaming videos, at every moment in time. This task is useful for many real-world applications (e.g., autonomous driving [18], robot assistants [19], and surveillance systems [16, 25]).

Recent methods [7, 32] for online action detection mostly employ recurrent neural networks (RNNs) with recurrent units (e.g., long short-term memory (LSTM) [14] and gated recurrent unit (GRU) [4]) for modeling the temporal sequence of an ongoing action. They introduce additional modules to learn a discriminative representation. However, these methods overlook the fact that the given input video contains not only the ongoing action but irrelevant actions and background. Specifically, the recurrent unit accumulates the input information without explicitly considering its relevance to the current action, and thus the learned representation would be less discriminative. Note that, in the task of detecting actions online, ignoring such a characteristic of streaming videos makes the problem more challenging [8].

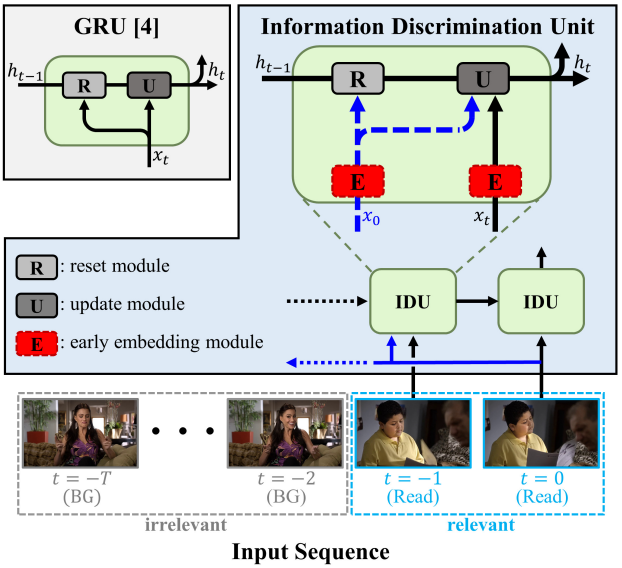


Figure 1. Comparison between GRU [4] and Information Discrimination Unit (IDU) which is proposed for our online action detection system. Our IDU extends GRU with two novel components, a mechanism utilizing current information (blue lines) and an early embedding module (red dash boxes). First, reset and update modules in our IDU additionally takes the current information (i.e., x_0), which enables to consider whether the past information (i.e, h_{t-1} and x_t) are relevant to an ongoing action such as x_0. Second, the early embedding module is introduced to consider the relation between high-level features for both information.

*Work done during a Ph.D. student at KAIST.

In this paper, we investigate on the question of *how RNNs can learn to explicitly discriminate relevant information from irrelevant information for detecting actions in the present*. To this end, we propose a novel recurrent unit that extends GRU [4] with a mechanism utilizing current information and an early embedding module (see Fig. 1). We name our recurrent unit Information Discrimination Unit (IDU). Specifically, our IDU models the relation between an ongoing action and past information (i.e., x_t and h_{t-1}) by additionally taking current information (i.e., x_0) at every time step. We further introduce the early embedding module to more effectively model the relation. By adopting action classes and feature distances as supervisions, our embedding module learns the features for the current and past information describing actions in a high level. Based on IDU, our Information Discrimination Network (IDN) effectively determines whether to use input information in terms of its relevance to the current action. This enables the network to learn a more discriminative representation for detecting ongoing actions. We perform extensive experiments on two benchmark datasets, where our IDN achieves state-of-the-art performances of 86.1% mcAP and 60.3% mAP on TVSeries [8] and THUMOS-14 [17], respectively. These performances significantly outperform TRN [32], the previous best performer, by 2.4% mcAP and 13.1% mAP on TVSeries and THUMOS-14, respectively.

Our contributions are summarized as follows:

- Different from previous methods, we investigate on how recurrent units can explicitly discriminate relevant information from irrelevant information for online action detection.
- We introduce a novel recurrent unit, IDU, with a mechanism using current information at every time step and an early embedding module to effectively model the relevance of input information to an ongoing action.
- We demonstrate that our IDN significantly outperforms state-of-the-arts in extensive experiments on two benchmark datasets.

2. Related Work

Offline Action Detection. The goal of offline action detection is to detect the start and end times of action instances from fully observed long untrimmed videos. Most methods [3, 24, 35] consist of two steps including action proposal generation and action classification. SSN [35] first evaluates actionness scores for temporal locations to generate temporal intervals. Then, these intervals are classified by modeling the temporal structures and completeness of action instances. TAL-Net [3] including the proposal generation and classification networks is the extended version of Faster R-CNN [22] for offline action detection. This method changes receptive field alignment, the range of receptive fields, and feature fusion to fit the action detection. Other methods [6, 33] with LSTM have been also studied for per-frame prediction.

Early Action Prediction. This task is similar to online action detection but focuses on recognizing actions from the partially observed videos. Hoai and la Torre [13] introduced a maximum margin framework with the extended structured SVM [29] to accommodate sequential data. Cai *et al.* [1] proposed to transfer the action knowledge learned from full actions for modeling partial actions.

Online Action Detection. Given a streaming video, online action detection aims to identify actions as soon as each video frame arrives, without observing future video frames. Geest *et al.* [8] introduced a new large dataset, TVSeries, for online action detection. They also analyzed and compared several baseline methods on the TVseries dataset. In [9], a two-stream feedback network with LSTM is proposed to individually perform the interpretation of the features and the modeling of the temporal dependencies. Gao, Yang, and Nevatia [7] proposed an encoder-decoder network with a reinforcement module, of which the reward function encourages the network to make correct decisions as early as possible. TRN [32] predicts future information and utilizes the predicted future as well as the past and current information together for detecting a current action.

Aforementioned methods [8, 7, 32] for online action detection adopt RNNs to model a current action sequence. However, the RNN units such as LSTM [14] and GRU [4] operate without explicitly considering whether input information is relevant to the ongoing action. Therefore, the current action sequence is modeled based on both relevant and irrelevant information, which results in a less discriminative representation.

3. Preliminary: Gated Recurrent Units

We first analyze GRU [4] to compare differences between the proposed IDU and GRU. GRU is one of the recurrent units, which is much simpler than LSTM. Two main components of GRU are reset and update gates.

The reset gate r_t is computed based on a previous hidden state h_{t-1} and an input x_t as follows:

$$r_t = \sigma(\mathbf{W}_{hr} h_{t-1} + \mathbf{W}_{xr} x_t), \tag{1}$$

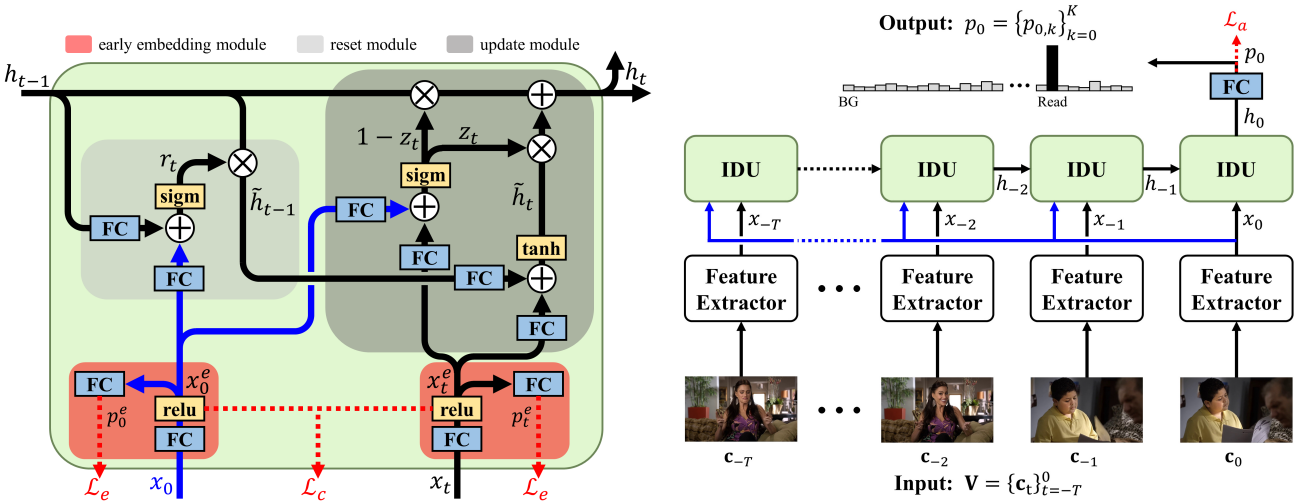
where $\mathbf{W}_{hr}$ and $\mathbf{W}_{xr}$ are parameters to be trained and σ is the logistic sigmoid function. Then, the reset gate determines whether a previous hidden state h_{t-1} is ignored as

$$\tilde{h}_{t-1} = r_t \odot h_{t-1}, \tag{2}$$

where $\tilde{h}_{t-1}$ is a new previous hidden state.

Similar to r_t, the update gate z_t is also computed based on h_{t-1} and x_t as

$$z_t = \sigma(\mathbf{W}_{xz} x_t + \mathbf{W}_{hz} h_{t-1}), \tag{3}$$



(a) Information Discrimination Unit (IDU) (b) Information Discrimination Network (IDN)

Figure 2. Illustration of our Information Discrimination Unit (IDU) and Information Discrimination Network (IDN). (a) Our IDU extends GRU with two new components, a mechanism using current information (i.e., x_0) (blue lines) and an early embedding module (red boxes). The first encourages reset and update modules to model the relation between past information (i.e., h_{t-1} and x_t) and an ongoing action. The second enables to effectively model the relation between high-level features for the input information. (b) Given an input streaming video $\mathbf{V} = \{\mathbf{c}_t\}_{t=-T}^{0}$ consisting of sequential chunks, IDN models a current action sequence and outputs the probability distribution p_0 of the current action over K action classes and background.

where $\mathbf{W}_{xz}$ and $\mathbf{W}_{hz}$ are learnable parameters. The update gate decides whether a hidden state h_t is updated with a new hidden state $\tilde{h}_t$ as follows:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t, \tag{4}$$

where

$$\tilde{h}_t = \eta(\mathbf{W}_{x\tilde{h}} x_t + \mathbf{W}_{\tilde{h}\tilde{h}} \tilde{h}_{t-1}). \tag{5}$$

Here $\mathbf{W}_{x\tilde{h}}$ and $\mathbf{W}_{\tilde{h}\tilde{h}}$ are trainable parameters and η is the tangent hyperbolic function.

Based on reset and update gates, GRU effectively drops and accumulates information to learn a compact representation. However, there are limitations when we applied GRU to online action detection as below:

First, the past information including x_t and h_{t-1} directly effects the decision of the reset and update gates. For online action detection, the relevant information to be accumulated is the information related to a current action. Thus, it is advantageous to make a decision based on the relation between the past information and the current action instead. To this end, we reformulate the computations of the reset and update gates by additionally taking the current information (i.e., x_0) as input.

This enables the reset and update gates to drop the irrelevant information and accumulate the relevant information regarding the ongoing action. Second, it is implicitly considered that the input features that the reset and update gates use represent valuable information. We augment GRU with an early embedding module with supervisions, action classes and feature distances, so that the input features explicitly describe actions. By optimizing features for the target task and dataset, our early embedding module also lets the reset and update gates focus on accumulating the relevant information along with the recurrent steps.

4. Approach

We present the schematic view of our IDU and the framework of IDN in Fig. 2. We first describe our IDU in details and then explain on IDN for online action detection.

4.1. Information Discrimination Units

Our IDU extends GRU with two new components, a mechanism utilizing current information (i.e., x_0) and an early embedding module. We explain IDU with early embedding, reset, and update modules, which takes a previous hidden state h_{t-1}, the features at each time x_t, and the features at current time x_0 as input and outputs a hidden state h_t (see Fig. 2.(a)).

Early Embedding Module. Our early embedding module individually processes the features at each time x_t and the features at current time x_0 and outputs embedded features x_t^e and x_0^e as follows:

$$x_t^e = \zeta(\mathbf{W}_{xe} x_t), \tag{6}$$

$$x_0^e = \zeta(\mathbf{W}_{xe} x_0), \tag{7}$$

where $\boldsymbol{W}_{xe}$ is a weight matrix and ζ is the ReLU [21] acti-

vation function. Note that we share $\boldsymbol{W}_{xe}$ for x_t and x_0. We omit a bias term for simplicity.

To encourage x_t^e and x_0^e to represent specific actions, we introduce two supervisions, action classes and feature distances. First, we process x_t^e and x_0^e to obtain probability distributions p_t^e and p_0^e over K action classes and background:

$$p_t^e = \xi(\mathbf{W}_{ep} x_t^e), \tag{8}$$

$$p_0^e = \xi(\mathbf{W}_{ep} x_0^e), \tag{9}$$

where $\boldsymbol{W}_{ep}$ is a shared weight matrix to be learned and ξ is the softmax function. We design a classification loss $\mathcal{L}_e$ by adopting the multi-class cross-entropy loss as

$$\mathcal{L}_e = -\sum_{k=0}^{K} \left(y_{t,k} \log(p_{t,k}^e) + y_{0,k} \log(p_{0,k}^e) \right), \tag{10}$$

where $y_{t,k}$ and $y_{0,k}$ are ground truth labels. Second, we use the contrastive loss [5, 10] proposed to learn an embedding representation by preserving the distance between similar data points close and dissimilar data points far on the embedding space in metric learning [28]. By using x_t^e and x_0^e as a pair, we design our contrastive loss $\mathcal{L}_c$ as

$$\begin{aligned} \mathcal{L}_c = & \mathbf{1}\{y_t = y_0\} D^2(x_t^e, x_0^e) \\ & + \mathbf{1}\{y_t \neq y_0\} \max(0, m - D^2(x_t^e, x_0^e)), \end{aligned} \tag{11}$$

where $D^2(a, b)$ is the squared Euclidean distance and m is a margin parameter.

We train our embedding module with $\mathcal{L}_e$ and $\mathcal{L}_c$, which provides more representative features for actions. More details on training will be provided in Section 4.2.

Reset Module. Our reset module takes the previous hidden state h_{t-1} and the embedded features x_0^e to compute a reset gate r_t as

$$r_t = \sigma(\mathbf{W}_{hr} h_{t-1} + \mathbf{W}_{x_0 r} x_0^e), \tag{12}$$

where $\mathbf{W}_{hr}$ and $\mathbf{W}_{x_0 r}$ are weight matrices which are learned. We define σ as the logistic sigmoid function same as GRU. We then obtain a new previous hidden state $\tilde{h}_{t-1}$ as follows:

$$\tilde{h}_{t-1} = r_t \odot h_{t-1}. \tag{13}$$

Different from GRU, we compute the reset gate r_t based on h_{t-1} and x_0^e. This enables our reset gate to effectively drop or take the past information according to its relevance to an ongoing action.

Update Module. Our update module adopts the embedded features x_t^e and x_0^e to compute an update gate z_t as follows:

$$z_t = \sigma(\mathbf{W}_{x_t z} x_t^e + \mathbf{W}_{x_0 z} x_0^e), \tag{14}$$

where $\mathbf{W}_{x_t z}$ and $\mathbf{W}_{x_0 z}$ are trainable parameters. Then, a hidden state h_t is computed as follows:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t, \tag{15}$$

where

$$\tilde{h}_t = \eta(\mathbf{W}_{x_t \tilde{h}} x_t^e + \mathbf{W}_{\tilde{h}\tilde{h}} \tilde{h}_{t-1}). \tag{16}$$

Here $\tilde{h}_t$ is a new hidden state and η is the tangent hyperbolic function. $\mathbf{W}_{x_t \tilde{h}}$ and $\mathbf{W}_{\tilde{h}\tilde{h}}$ are trainable parameters.

There are two differences between the update modules of our IDU and GRU. The first difference is that our update gate is computed based on x_t^e and x_0^e. This allows the update gate to consider whether x_t^e is relevant to an ongoing action. Second, our update gate uses the embedded features which are more representative in terms of specific actions.

4.2. Information Discrimination Network

In this section, we explain our recurrent network, called IDN, for online action detection (see Fig. 2.(b)).

Problem Setting. To formulate the online action detection problem, we follow the same setting as in previous methods [7, 32]. Given a streaming video $\mathbf{V} = \{\mathbf{c}_t\}_{t=-T}^{0}$ including current and T past chunks as input, our IDN outputs a probability distribution $p_0 = \{p_{0,k}\}_{k=0}^{K}$ of a current action over K action classes and background. Here we define a chunk $c = \{I_n\}_{n=1}^{N}$ as the set of N consecutive frames. I_n indicates the nth frame.

Feature Extractor. We use TSN [30] as a feature extractor. TSN takes an individual chunk $\mathbf{c}_t$ as input and outputs an appearance feature vector x_t^a and a motion feature vector x_t^m. We concatenate $x_t^a \in \mathbb{R}^{d_a}$ and $x_t^m \in \mathbb{R}^{d_m}$ into a two-stream feature vector $x_t = [x_t^a, x_t^m] \in \mathbb{R}^{d_x}$. Here d_x equals to $d_a + d_m$. After that, we sequentially feed x_t and x_0 into our IDU.

Training. We feed the hidden state h_0 at current time into a fully connected layer to obtain the final probability distribution p_0 of an ongoing action as follows:

$$p_0^e = \xi(\mathbf{W}_{hp} h_0), \tag{17}$$

where $\mathbf{W}_{hp}$ is a trainable matrix and ξ is the softmax function.

We define a classification loss $\mathcal{L}_a$ for a current action by employing the standard cross-entropy loss as

$$\mathcal{L}_a = -\sum_{t=-T}^{0} \sum_{k=0}^{K} y_{t,k} \log(p_{t,k}), \tag{18}$$

where $y_{t,k}$ are the ground truth labels for the tth time step. We train our IDN by jointly optimizing $\mathcal{L}_a$, $\mathcal{L}_e$, and $\mathcal{L}_c$ by designing a multi-task loss $\mathcal{L}$ as follows:

$$\mathcal{L} = \mathcal{L}_a + \alpha(\mathcal{L}_e + \mathcal{L}_c), \tag{19}$$

where α is a balance parameter.

5. Experiments

In this section, we evaluate the proposed method on two benchmark datasets, TVSeries [8] and THUMOS-14 [17]. We first demonstrate the effectiveness of our IDU by conducting comprehensive ablation studies. We then report comparison results among our IDN and the state-of-the-art methods for online action detection.

5.1. Datasets

TVSeries [8]. This dataset includes 27 untrimmed videos on six popular TV series, divided into 13, 7, and 7 videos for training, validation, and test, respectively. Each video contains a single episode, approximately 20 minutes or 40 minutes long. The dataset is temporally annotated with 30 realistic actions (e.g., open door, read, eat, *etc*). The TVSeries dataset is challenging due to diverse undefined actions, multiple actors, heavy occlusions, and a large proportion of non-action frames.

THUMOS-14 [17]. The THUMOS-14 dataset consists of 200 and 213 untrimmed videos for validation and test sets, respectively. This dataset has temporal annotations with 20 sports actions (e.g., diving, shot put, billiards, *etc*). Each video includes 15.8 action instances and 71% background on average. As done in [7, 32], we used the validation set for training and the test set for evaluation.

5.2. Evaluation Metrics

For evaluating performance in online action detection, existing methods [8, 7, 32] measure mean average precision (mAP) and mean calibrated average precision (mcAP) [8] in a frame-level. Both metrics are computed in two steps: 1) calculating the average precision over all frames for each action class and 2) averaging the average precision values over all action classes.

mean Average Precision (mAP). On each action class, all frames are first sorted in descending order of their probabilities. The average precision of the kth class over all frames is then calculated based on the precision at cut-off i (i.e., on the i sorted frames). The final mAP is defined as the mean of the AP values over all action classes.

mean calibrated Average Precision (mcAP). It is difficult to compare two different classes in terms of the AP values when the ratios of positive frames versus negative frames for these classes are different. To address this problem, Geest *et al.* [8] propose the calibrated precision as

$$\text{cPrec}(i) = \frac{w\text{TP}(i)}{w\text{TP}(i) + \text{FP}(i)}, \tag{20}$$

where w is a ratio between negative frames and positive frames. Similar to the AP, the calibrated average precision of the kth class over all frames is computed as

$$\text{cAP}_k = \frac{\sum_i \text{cPrec}(i)\mathbf{1}(i)}{N_P}. \tag{21}$$

Module	Type	Weight	Size
Early Embedding Module	FC	$\mathbf{W}_{xe}$	$d_x \times 512$
	FC	$\mathbf{W}_{ep}$	$512\times(K+1)$
Reset Module	FC	$\mathbf{W}_{hr}$	512×512
	FC	$\mathbf{W}_{x_a r}$	512×512
Update Module	FC	$\mathbf{W}_{x_t z}$	512×512
	FC	$\mathbf{W}_{x_0 z}$	512×512
	FC	$\mathbf{W}_{x_t \tilde{h}}$	512×512
	FC	$\mathbf{W}_{\tilde{h}\tilde{h}}$	512×512
Classification	FC	$\mathbf{W}_{hp}$	$512\times(K+1)$

Table 1. Specifications of our IDN. d_x is the dimension of the two-stream feature vector x_t and $K+1$ is the number of action and background classes.

Then, the mcAP is obtained by averaging the cAP values over all action classes.

5.3. Implementation Details

Problem Setting. We use the same setting as used in state-of-the-art methods [7, 32]. On both TVSeries [8] and THUMOS-14 [17] datasets, we extract video frames at 24 fps and set the number of frames in each chunk N to 6. We use 16 chunks (i.e., $T = 15$), which are 4 seconds long, for the input of IDN.

Feature Extractor. We use a two-stream network as a features extractor. In the two-stream network, one stream encodes appearance information by taking the center frame of a chunk as input, while another stream encodes motion information by processing an optical flow stack computed from an input chunk. Among several two-stream networks, we employ the TSN model [30] pretrained on the ActivityNet-v1.3 dataset [12]. Note that this TSN is the same feature extractor as used in state-of-the-art methods [7, 32]. The TSN model consists of ResNet-200 [11] for an appearance network and BN-Inception [15] for a motion network. We use the outputs of the `Flatten_673` layer in ResNet-200 and the `global_pool` layer in BN-Inception as the appearance features x_t^a and motion features x_t^m, respectively. The dimensions of x_t^a and x_t^m are $d_a = 2048$ and $d_m = 1024$, respectively, and d_x equals to 3072.

IDN Architecture. Table 1 provides the specifications of IDN considered in our experiments. In the early embedding module, we set the number of the hidden units for $\mathbf{W}_{xe}$ to 512. In the reset module, both weights $\mathbf{W}_{hr}$ and $\mathbf{W}_{x_a r}$ have 512 hidden units. In the update module, we use 512 hidden units for $\mathbf{W}_{x_t z}$, $\mathbf{W}_{x_0 z}$, $\mathbf{W}_{x_t \tilde{h}}$, and $\mathbf{W}_{\tilde{h}\tilde{h}}$. According to the number of action classes, we set $K+1$ to 31 for TVSeries and 21 for THUMOS-14.

IDN Training. To train our IDN, we use a stochastic gradient descent optimizer with the learning rate of 0.01 for both THUMOS-14 and TVSeries datasets. We set batch size to 128 and balance the numbers of action and back-

ground samples in terms of the class of $\mathbf{c}_0$. We empirically set the margin parameter m in Eq. (11) to 1.0 and the balance parameter α in Eq. (19) to 0.3.

5.4. Ablation Study

We evaluate RNNs with the simple unit, LSTM [14], and GRU [4]. We name these networks RNN-Simple, RNN-LSTM, and RNN-GRU, respectively. Although many methods [8, 7, 32] report the performances of these networks as baselines, we evaluate them in our setting to clearly confirm the effectiveness of our IDU.

In addition, we individually add IDU components to GRU as a baseline for analyzing their effectiveness:

Baseline+CI: We add a mechanism using current information to GRU in computing reset and update gates. Specifically, we replace Eq. (1) for r_t with

$$r_t = \sigma(\mathbf{W}_{hr} h_{t-1} + \mathbf{W}_{x_0 r} x_0) \tag{22}$$

and Eq. (3) for z_t with

$$z_t = \sigma(\mathbf{W}_{x_t z} x_t + \mathbf{W}_{x_0 z} x_0), \tag{23}$$

where $\mathbf{W}_{hr}$, $\mathbf{W}_{x_0 r}$, $\mathbf{W}_{x_t z}$, and $\mathbf{W}_{x_0 z}$ are trainable parameters. We construct a recurrent network with this modified unit.

Baseline+CI+EE (IDN): We incorporate our main components, a mechanism utilizing current information and an early embedding module, into GRU, which is our IDU. These components enable reset and update gates to effectively model the relation between an ongoing action and input information at every time step. Specifically, Eq. (12) and Eq. (14) are substituted for Eq. (1) and Eq. (3), respectively. We design a recurrent network with our IDU, which is the proposed IDN.

In Table 2, we report the performances of five networks on the TVSeries dataset [8]. Among RNN-Simple, RNN-LSTM, and RNN-GRU, RNN-GRU results in the highest mcAP of 81.3%. By comparing RNN-GRU (Baseline) with Baseline+CI, we first analyze the effect of using x_0 in calculating reset and update gates. This component enables the gates to decide whether input information at each time is relevant to a current action. As a result, Baseline-CI achieves the performance gain of 2.1% mcAP, which demonstrates the effectiveness of using x_0. Next, we observe that adding the early embedding module improves the performance by 1.3% mcAP from the comparison between Baseline+CI and Baseline+CI+EE (IDN). Note that our IDN achieves mcAP of 84.7% with a performance gain of 3.4% mcAP compared with Baseline.

We conduct the same experiment on the THUMOS-14 dataset [17] to confirm the generality of the proposed components. We obtain performance gains as individually incorporating the proposed components into GRU (see Table 3), where our IDN achieves an improvement of 3.3% mAP compared to Baseline. These results successfully demonstrate the effectiveness and generality of our components.

Method	mcAP (%)
RNN-Simple	79.9
RNN-LSTM	80.9
RNN-GRU (Baseline)	81.3
Baseline+CI	83.4
Baseline+CI+EE (IDN)	**84.7**

Table 2. Ablation study of the effectiveness of our proposed components on TVSeries [8]. CI and EE indicate additionally using the current information and early embedding input information, respectively.

Method	mAP (%)
RNN-Simple	45.5
RNN-LSTM	46.3
RNN-GRU (Baseline)	46.7
Baseline+CI	48.6
Baseline+CI+EE (IDN)	**50.0**

Table 3. Ablation study of the effectiveness of our proposed components on THUMOS-14 [17]. CI and EE indicate additionally using the current information and early embedding input information, respectively.

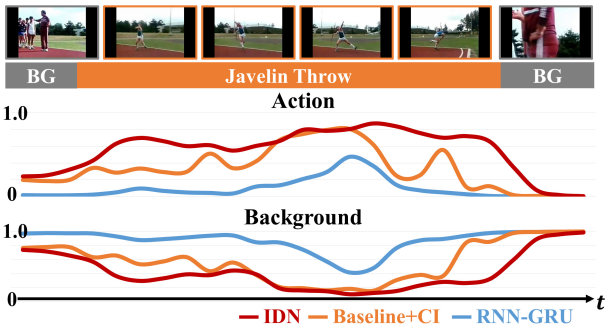


Figure 3. Qualitative comparisons on predicted and GT probabilities for action (top) and background (bottom).

Figure 3 shows qualitative comparisons on predicted and GT probabilities, where our IDN achieves the best results on both action and background frames.

To confirm the effect of our components, we compare the values of the update gates z_t between our IDU and GRU. For a reference, we introduce the relevance score R_t of each chunk regarding a current action. Specifically, we set the scores of input chunks representing the current action as 1, otherwise 0 (see Fig. 4). Note that the update gate controls how much information from the input will carry over to the hidden state. Therefore, the update gate should drop the irrelevant information and pass over the relevant information related to the current action. In Fig. 5, we plot the z_t values of IDU and GRU and relevance scores against each time step. On the input sequences containing from one to

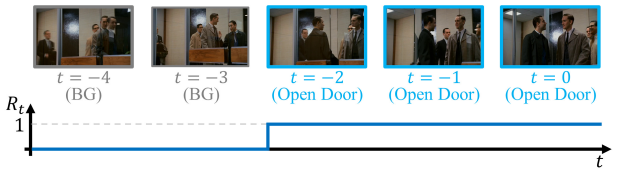
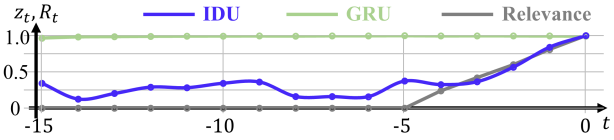
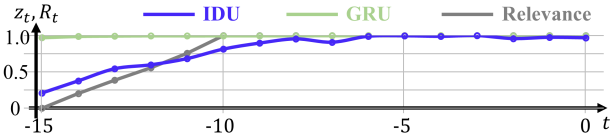


Figure 4. Example of relevance scores R_t of input chunks.



(a) On the input sequences containing from one to five relevant chunks (i.e., from $t = 0$ to $t = -4$).



(b) On the input sequences containing from 11 to 15 relevant chunks (i.e., from $t = -10$ to $t = -14$).

Figure 5. Comparison between the update gate z_t values of our IDU and GRU [4]. Update gate values are measured on the input sequences containing (a) from one to five relevant chunks and (b) from 11 to 15 relevant chunks.

five relevant chunks, the z_t values of GRU are very high at all time steps. In contrast, our IDU successfully learns the z_t values following the relevance scores (see Fig. 5(a)). We also plot the average z_t values on the input sequences including from 11 to 15 relevant chunks in Fig. 5(b), where our IDU yields the z_t values similar to the relevance scores. These results demonstrate that our IDU effectively models the relevance of input information to the ongoing action.

Compared to GRU, IDU has additional weights $\mathbf{W}_{xe} \in \mathbb{R}^{d_x \times 512}$ and $\mathbf{W}_{ep} \in \mathbb{R}^{512 \times (K+1)}$ in the early embedding module. Our early embedding module reduces the dimensions of x_t, $x_0 \in \mathbb{R}^{d_x \times 512}$, which makes the parameters (i.e., $\mathbf{W}_{x_0 r}$, $\mathbf{W}_{x_t z} \in \mathbb{R}^{512 \times 512}$) in IDU less than the parameters (i.e., $\mathbf{W}_{xr}$, $\mathbf{W}_{xz} \in \mathbb{R}^{d_x \times 512}$) in GRU. The other weights have the same number of parameters in IDU and GRU. As a result, the number of parameters for IDU is 75.3% of that for GRU with $d_x = 3072$ and $K = 20$.

5.5. Performance Comparison

In this section, we compare our IDN with state-of-the-art methods on TVSeries [8] and THUMOS-14 [17] datasets. We use three types of input including RGB, Flow, and Two-Stream. As the input of our IDU, we take only appearance features for the RGB input and motion features for the Flow input. IDN, TRN [32], RED [7], and ED [7] use same two-stream features for the Two-Stream input, which allows a fair comparison. We also employ another feature extractor,

Input	Method	mcAP (%)
RGB	LRCN [6]	64.1
	RED [7]	71.2
	2S-FN [9]	72.4
	TRN [32]	75.4
	IDN	**76.6**
Flow	FV-SVM [8]	74.3
	IDN	**80.3**
Two-Stream	RED [7]	79.2
	TRN [32]	83.7
	IDN	**84.7**
	IDN-Kinetics	**86.1**

Table 4. Performance comparison on TVSeries [8]. IDN, TRN [32], RED [7], and ED [7] use same two-stream features for the Two-Stream input.

Setting	Method	mAP (%)
Offline	CNN [27]	34.7
	CNN [26]	36.2
	LRCN [6]	39.3
	MultiLSTM [33]	41.3
	CDC [23]	44.4
Online	RED [7]	45.3
	TRN [32]	47.2
	IDN	**50.0**
	IDN-Kinetics	**60.3**

Table 5. Performance comparison on THUMOS-14 [17]. IDN, TRN [32], RED [7], and ED [7] use same two-stream features.

the TSN model [30] pretrained on the Kinetics dataset [2]. We name our IDN with this feature extractor IDN-Kinetics.

We report the results on TVSeries in Table 4. Our IDN significantly outperforms state-of-the-art methods on all types of input, where IDN achieves 76.6% mcAP on the RGB input, 80.3% mcAP on the Flow input, and 84.1% mcAP on the Two-Stream input. Furthermore, IDN-Kinetics achieves the best performance of 86.1% mcAP. Note that IDN effectively reduces wrong detection results occurred from the irrelevant information by discriminating the relevant information. However, 2S-FN, RED, and TRN accumulates the input information without considering its relevance to an ongoing action. In addition, our IDN yields better performance than TRN [32] although IDN takes shorter temporal information than IDN (i.e., 16 chunks vs. 64 chunks).

In Table 5, we compare performances between our IDN and state-of-the-art approaches for online and offline action detection. The compared offline action detection methods perform frame-level prediction. As a result, both IDN and IDN-Kinetics outperforms all methods by a large margin.

In online action detection, it is important to identify actions as early as possible. To compare this ability, we mea-

Method	Portion of action									
	0%-10%	10%-20%	20%-30%	30%-40%	40%-50%	50%-60%	60%-70%	70%-80%	80%-90%	90%-100%
CNN [8]	61.0	61.0	61.2	61.1	61.2	61.2	61.3	61.5	61.4	61.5
LSTM [8]	63.3	64.5	64.5	64.3	65.0	64.7	64.4	64.4	64.4	64.3
FV-SVM [8]	67.0	68.4	69.9	71.3	73.0	74.0	75.0	75.4	76.5	76.8
TRN [32]	78.8	79.6	80.4	81.0	81.6	81.9	82.3	82.7	82.9	83.3
IDN	80.6	81.1	81.9	82.3	82.6	82.8	82.6	82.9	83.0	83.9
IDN-Kinetics	**81.7**	**81.9**	**83.1**	**82.9**	**83.2**	**83.2**	**83.2**	**83.0**	**83.3**	**86.6**

Table 6. Performance comparison for different portions of actions on TVSeries [8] in terms of mcAP (%). The corresponding portions of actions are only used to compute mcAP after detecting current actions on all frames in an online manner.

Figure 6. Qualitative evaluation of IDN on TVSeries [8] (top) and THUMOS-14 [17] (bottom). Each result shows frames, ground truth, and estimated probabilities.

sure the mcAP values for every 10% portion of actions on TVSeries. Table 6 shows the comparison results among IDN, IDN-Kinetics, and previous methods, where our methods achieve state-of-the-art performance at every time interval. This demonstrates the superiority of our IDU in identifying actions at early stages as well as all stages.

5.6. Qualitative Evaluation

For qualitative evaluation, we visualize our results on TVSeries [8] and THUMOS-14 [17] in Fig. 6. The results on the TVSeries dataset show high probabilities on the true action label and reliable start and end time points. Note that identifying actions at the early stage is very challenging in this scene because the only subtle change (i.e., opening a book) happens. On THUMOS-14, our IDN successfully identifies ongoing actions by yielding the contrasting probabilities between true action and background labels.

6. Conclusion

In this paper, we proposed IDU that extends GRU [4] with two novel components: 1) a mechanism using current information and 2) an early embedding module. These components enable IDU to effectively decide whether input information is relevant to a current action at every time step. Based on IDU, our IDN effectively learns to discriminate relevant information from irrelevant information for identifying ongoing actions. In comprehensive ablation studies, we demonstrated the generality and effectiveness of our proposed components. Moreover, we confirmed that our IDN significantly outperforms state-of-the-art methods on TVSeries [8] and THUMOS-14 [17] datasets for online action detection.

Acknowledgments

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.B0101-15-0266, Development of High Performance Visual Big-Data Discovery Platform for Large-Scale Realtime Data Analysis).

References

[1] Y. Cai, H. Li, J.-F. Hu, and W.-S. Zheng. Action knowledge transfer for action prediction with partial videos. In *Proc. Association for the Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence*, pages 8118–8125, Jan. 2019.

[2] J. Carreira and A. Zisserman. Quo vaids, action recognition? a new model and the kinectics dataset. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733, Jul. 2017.

[3] Y.-W. Chao, S. Vijayanarasimhan, B. Seybold, D. A. Ross, J. Deng, and R. Sukthankar. Rethinking the faster r-cnn architecture for temporal action localization. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1130–1139, Jun. 2018.

[4] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Oct. 2014.

[5] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 539–546, Jun. 2005.

[6] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2625–2634, Jun. 2015.

[7] J. Gao, Z. Yang, and R. Nevatia. Red: Reinforced encoder-decoder networks for action anticipation. In *Proc. British Machine Vision Conference (BMVC)*, pages 92.1–92.11, Sep. 2017.

[8] R. De Geest, E. Gavves, A. Ghodrati, Z. Li, G. Snoek, and T. Tuytelaars. Online action detection. In *Proc. European Conference on Computer Vision (ECCV)*, pages 269–285, Oct. 2016.

[9] R. De Geest and T. Tuytelaars. Modeling temporal structure with lstm for online action detection. In *Proc. IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1549–1557, Mar. 2018.

[10] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1735–1742, Jun. 2006.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 771–778, Jul. 2016.

[12] F. C. Heilbron, B. Ghanem V. Escorcia, and J. C. Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–970, Jun. 2015.

[13] M. Hoai and F. De la Torre. Max-margin early event detector. *International Journal of Computer Vision (IJCV)*, 2:191–202, Apr. 2014.

[14] S. Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, Dec. 1997.

[15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*, 2015.

[16] Y. Iwashita, M. Ryoo, T. J. Fuchs, and C. Padgett. Recognizing humans in motion: Trajectory-based aerial video analysis. In *Proc. British Machine Vision Conference (BMVC)*, pages 127.1–127.11, Sep. 2013.

[17] Y. G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. Thumos challenge: Action recognition with a large number of classes, 2014. http://crcv.ucf.edu/THUMOS14/.

[18] J. Kim, T. Misu, Y.-T. Chen, A. Tawari, and J. Canny. Grounding human-to-vehicle advice for self-driving vehicles. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10591–10599, Jun. 2019.

[19] H. S. Koppula and A. Saxena. Anticipating human activities for reactive robotic response. In *Proc. International Conference on Intelligent Robots and Systems (IROS)*, pages 2071–2071, Nov. 2013.

[20] Y. Liu, L. Ma, Y. Zhang, W. Liu, and S.-F. Chang. Multi-granularity generator for temporal action proposal. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3604–3613, Jun. 2019.

[21] V. Nair and G. E. Hinton. Rectified linear units improve restricted obltzmann machines. In *Proc. International Conference on Machine Learning (ICML)*, Jun. 2010.

[22] S. Ren, K. He, R. Girchick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proc. Conference on Neural Information Processing Systems (NeurIPS)*, pages 91–99, Dec. 2015.

[23] Z. Shou, J. Chan, A. Zareian, K. Miyazawa, and S.-F. Chang. Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5734–5743, Jul. 2017.

[24] Z. Shou, D. Wang, and S.-F. Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1049–1058, Jun. 2016.

[25] T. Shu, D. Xie, B. Rothrock, S. Todorovic, and S. Chun Zhu. Joint inference of groups, events and human roles in aerial videos. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4756–4584, Jun. 2015.

[26] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Proc. Conference on Neural Information Processing Systems (NeurIPS)*, pages 568–576, Dec. 2014.

[27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. International Conference on Learning Representations (ICLR)*, May 2015.

[28] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Proc. Conference on Neural Information Processing Systems (NeurIPS)*, pages 1857–1865, Dec. 2016.

[29] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research (JMLR)*, 6:1453–1484, Sep. 2005.

[30] L. Wang, Y. Xiong, Z. Wang, Y. Qiao D. Lin, X. Tang, and L. van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *Proc. European Conference on Computer Vision (ECCV)*, pages 20–36, Oct. 2016.

[31] H. Xu, A. Das, and K. Saenko. R-c3d: Region convolutional 3d network for temporal activity detection. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 5783–5792, Oct. 2017.

[32] M. Xu, M. Gao, Y.-T. Chen, L. S. Davis, and D. J. Crandall. Temporal recurrent networks for online action detection. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 5532–5541, Oct. 2019.

[33] S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and L. Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *International Journal of Computer Vision (IJCV)*, 126:375–389, Apr. 2018.

[34] X.-Y. Zhang, H. Shi, C. Li, X. Zhu K. Zheng, and L. Duan. Learning transferable self-attentive representations for action recognition in untrimmed videos with weak supervision. In *Proc. Association for the Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence*, pages 9227–9242, Jan. 2019.

[35] Y. Zhao, Y. Xiong, L. Wang, Z. Wu, X. Tang, and D. Lin. Temporal action detection with structured segment networks. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2914–2923, Oct. 2017.