Temporal Recurrent Networks for Online Action Detection

Mingze Xu[1*] Mingfei Gao[2*] Yi-Ting Chen[3] Larry S. Davis[2] David J. Crandall[1]
[1]Indiana University [2]University of Maryland [3]Honda Research Institute, USA
{mx6,djcran}@indiana.edu, {mgao,lsd}@umiacs.umd.edu, ychen@honda-ri.com

Abstract

Most work on temporal action detection is formulated as an offline problem, in which the start and end times of actions are determined after the entire video is fully observed. However, important real-time applications including surveillance and driver assistance systems require identifying actions as soon as each video frame arrives, based only on current and historical observations. In this paper, we propose a novel framework, the Temporal Recurrent Network (TRN), to model greater temporal context of each frame by simultaneously performing online action detection and anticipation of the immediate future. At each moment in time, our approach makes use of both accumulated historical evidence and predicted future information to better recognize the action that is currently occurring, and integrates both of these into a unified end-to-end architecture. We evaluate our approach on two popular online action detection datasets, HDD and TVSeries, as well as another widely used dataset, THUMOS'14. The results show that TRN significantly outperforms the state-of-the-art.

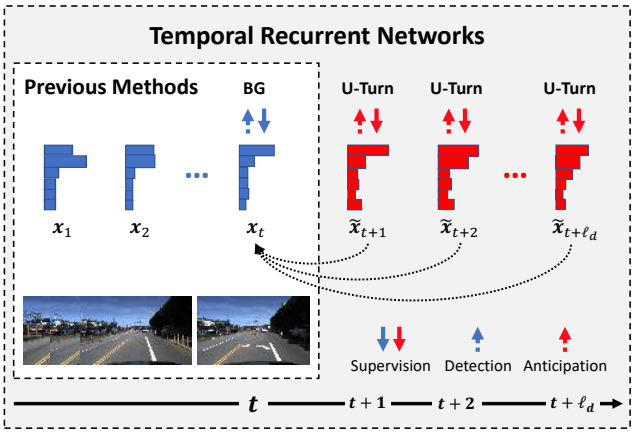


Figure 1: *Comparison between our proposed Temporal Recurrent Network (TRN) and previous methods.* Previous methods use only historical observations and learn representations for actions by optimizing current action estimation. Our approach learns a more discriminative representation by jointly optimizing current and future action recognition, and incorporates the *predicted* future information to improve the performance of action detection in the present.

1. Introduction

As we go about our lives, we continuously monitor the social environment around us, making inferences about the actions of others that might affect us. Is that child running into the road or just walking towards the sidewalk? Is that passerby outstretching his hand for a punch or a handshake? Is that oncoming car turning left or doing a U-turn? These and many other actions can occur at any time, without warning. In order to be able to react to the world around us, we must make and update our inferences in real-time, updating and refining our hypotheses moment-to-moment as we collect additional evidence over time.

In contrast, action recognition in computer vision is often studied as an offline classification problem, in which the goal is to identify a single action occurring in a short video clip given *all of its frames* [3, 5, 9, 18, 37, 53]. This offline formulation simplifies the problem considerably: a left turn can be trivially distinguished from a U-turn if the end of the action can be observed. But emerging real-world applications of computer vision like self-driving cars, interactive home virtual assistants, and collaborative robots require detecting actions online, in real-time. Several recent papers have considered this online action detection problem [11,12,17,19,36,50], but accuracies are generally lower than the offline case because using only current and past information makes the problem much more challenging.

Here we introduce the novel hypothesis that although future information is not available in an online setting, *explicitly predicting the future can help to better classify actions in the present.* We propose a new model to estimate and use this future information, and we present experimental results showing that predicted future information indeed improves the performance of online action recognition. This may seem like a surprising result because at test time, a model

*The first two authors contributed equally. Part of this work was done when MX and MG were interns at Honda Research Institute, USA.

that predicts the future to infer an action in the present observes exactly the same evidence as a model that simply infers the action directly. However, results in cognitive science and neuroscience suggest that the human brain uses prediction of the future as an important mechanism for learning to make estimates of the present [2, 8, 15, 16]. Our findings seem to confirm that the same applies to automatic online action recognition, suggesting that jointly modeling current action detection and future action anticipation during training forces the network to learn a more discriminative representation.

In more detail, in this paper we propose a general framework called Temporal Recurrent Network (TRN), in which future information is predicted as an anticipation task and used together with historical evidence to recognize action in the current frame (as shown in Fig. 1). To demonstrate the effectiveness of our method, we validate TRN on two recent online action detection datasets (Honda Research Institute Driving Dataset (HDD) [33] and TVSeries [11]) and a widely used action recognition dataset, THUMOS'14 [24]. Our model is general enough to use both visual and non-visual sensor data, as we demonstrate for the HDD driving dataset. Experimental results show that our approach significantly outperforms baseline methods, especially when only a fraction of an action is observed. We also evaluate action anticipation (predicting the next action), showing that our method performs better than state-of-the-art methods even though anticipation is not the focus of this work.

2. Related Work

Action and Activity Recognition. There is extensive work in the literature on action and activity recognition for videos of various types and applications, from consumer-style [52] and surveillance videos [40], to first-person videos from wearable cameras [27, 30, 31]. Early work used hand-crafted visual features, such as HOG [28], HOF [28], and MBH [44], and motion features, such as improved dense trajectories [43], while most recent methods use deep convolutional networks. Simonyan and Zisserman [38] propose a two-stream convolutional network that uses both RGB frames and optical flow as inputs [45], while others including Tran *et al*. [42] and Carreira *et al*. [4] avoid pre-computing optical flow by learning temporal information in an end-to-end manner using 3D convolution. Recurrent neural networks (RNNs), such as long short-term memory (LSTM) [23] and gated recurrent unit (GRU) [7] networks, have also been widely adopted to capture temporal dependencies [14] and motion information [49]. However, most of these methods focus on trimmed videos and cannot be directly applied to long video sequences that contain multiple actions and a wide diversity of backgrounds.

Offline Action Detection. Offline methods observe an entire video and estimate the start and end moment of each action. Many of these methods are inspired by region-based deep networks from object detection [34] and segmentation [20]. Shou *et al*. [37] propose S-CNNs to localize actions in untrimmed videos by generating temporal action proposals, and then classifying them and regressing their temporal boundaries. TCN [9] performs proposal ranking but explicitly incorporates local context of each proposal. R-C3D [48] improves efficiency by sharing convolutional features across proposal generation and classification. SST [3] avoids dividing input videos into overlapping clips, introducing more efficient proposal generation in a single stream, and TURN TAP [18] builds on this architecture. TAL-Net [5] improves receptive field alignment using a multi-scale architecture that better exploits temporal context for both proposal generation and action classification. CDC [35] makes frame-level dense predictions by simultaneously performing spatial downsampling and temporal upsampling operations. But the above work assumes all video frames can be observed, which is not possible in the online task that we consider here.

Early Action Detection. Our work is also related to early action detection, which tries to recognize actions after observing a fraction of the event. Hoai *et al*. [22] propose a max-margin framework using structured SVMs for this problem. Ma *et al*. [32] design an improved technique based on LSTMs and modify the training loss to assume that the score margin between correct and incorrect classes should be non-decreasing as more observations are made.

Online Action Detection. Given a live video stream, online action detection tries to detect the actions performed in each frame as soon as it arrives, without considering future context. De Geest *et al*. [11] introduced a concrete definition and realistic dataset (TVSeries) for this problem. They later [12] proposed a two-stream feedback network, with one stream focusing on input feature interpretation and the other modeling temporal dependencies between actions. Gao *et al*. [17] propose a Reinforced Encoder-Decoder (RED) network and a reinforcement loss to encourage recognizing actions as early as possible. RED was designed for action anticipation – predicting actions a few seconds into the future – but can be applied to online detection by setting the anticipation time to 0. Shou *et al*. [36] identify the start time of each action using Generative Adversarial Networks and adaptive sampling to distinguish ambiguous backgrounds, and explicit temporal modeling around transitions between actions for temporal consistency.

In contrast to this existing online action detection work which only focuses on current and past observations, we introduce a model that learns to simultaneously perform online action detection and anticipation of the immediate future, and uses this estimated "future information" to improve the action detection performance of the present.

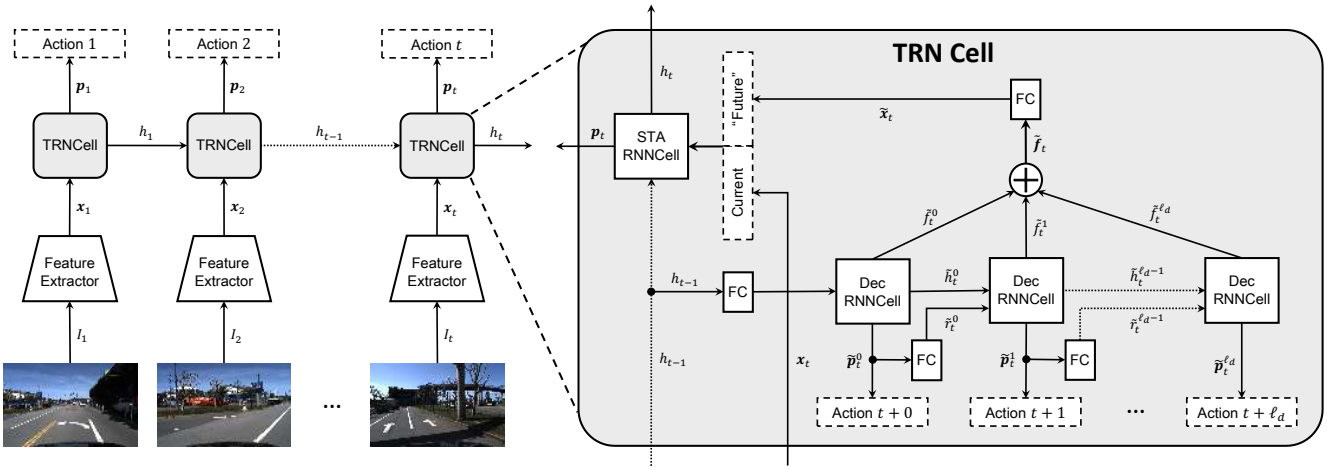


Figure 2: *Our proposed Temporal Recurrent Network (TRN)*, which sequentially processes input video frames and outputs frame-level action class probabilities, like any RNN. But while RNNs only model historical temporal dependencies, TRN anticipates the future via a temporal decoder, and incorporates that predicted information to improve online action detection.

3. Online Action Detection

Given a live video stream that contains one or more actions, our goal is to recognize actions of interest occurring in each video frame. Unlike most prior work that assumes the entire video is available at once, this *online action detection* problem requires us to process each frame as soon as it arrives, without accessing *any* future information. More formally, our goal is to estimate, for each frame I_t of an image sequence, a probability distribution $\mathbf{p}_t = [p_t^0, p_t^1, p_t^2, \cdots, p_t^K]$ over K possible actions, given only the past and current frames, $\{I_1, I_2, \cdots, I_t\}$ (where p_t^0 denotes the "background" probability that no action is occurring).

3.1. Temporal Recurrent Network (TRN)

To solve this problem, we introduce a novel framework called a Temporal Recurrent Network (TRN). The main idea is to train a network that predicts actions several frames into the future, and then uses that prediction to classify an action in the present. Fig. 2 shows the architecture of TRN. The core of the network is a powerful recurrent unit, the *TRN cell*. Like a general RNN cell, at each time t a TRN cell receives a feature vector $\mathbf{x}_t$ corresponding to the observation at time t, which could include some combination of evidence from the appearance or motion in frame I_t or even other sensor modalities collected at time t, and the hidden state h_{t-1} from the previous time step. The cell then outputs $\mathbf{p}_t$, a probability distribution estimating which action is happening in I_t. The hidden state h_t is then updated and used for estimating the next time step. But while a traditional RNN cell only models *prior* temporal dependencies by accumulating historical evidence of the input sequence, a TRN cell also takes advantage of the temporal correlations between current and future actions by anticipating upcoming actions and explicitly using these estimates to help recognize the present action.

3.2. TRN Cell

The TRN cell controls the flow of internal information by using a temporal decoder, a future gate, and a spatiotemporal accumulator (STA). We use LSTMs [23] as the backbone for both the temporal decoder and the STA in our implementation, although other temporal models such as gated recurrent units (GRUs) [7] and temporal convolutional networks (TCNs) [29] could be used. The temporal decoder learns a feature representation and predicts actions for the future sequence. The future gate receives a vector of hidden states from the decoder and embeds these features as the future context. The STA captures the spatiotemporal features from historical, current, and predicted future information, and estimates the action occurring in the current frame.

Algorithm 1 Workflow of TRN Cell

Input: image feature $\mathbf{x}_t$ and previous hidden state h_{t-1}
Output: probabilities $\mathbf{p}_t$ and current hidden state h_t

Initialize $\widetilde{h}_t^{-1}$ with h_{t-1} embedded by an FC layer
Initialize $\widetilde{r}_t^{-1}$ with all zeros
for i = 0 : l_d **do**
    Update $\widetilde{h}_t^i$ using $\widetilde{r}_t^{i-1}$ and $\widetilde{h}_t^{i-1}$
    Compute f_t^i and $\widetilde{p}_t^i$ using $\widetilde{h}_t^i$
    Update $\widetilde{r}_t^i$ using $\widetilde{p}_t^i$
end for
Compute future context features $\widetilde{\mathbf{x}}_t$ as Eq. (1)
Update h_t with STA$(h_{t-1}, [\mathbf{x}_t, \widetilde{\mathbf{x}}_t])$
Compute $\mathbf{p}_t$ as Eq. (2)

We now describe each component of a TRN cell in detail, as summarized in Alg. 1.

The temporal decoder works sequentially to output the estimates of future actions and their corresponding hidden states $\{\widetilde{h}_t^0, \widetilde{h}_t^1, \cdots, \widetilde{h}_t^{\ell_d}\}$ for the next ℓ_d time steps, where h_t^i for $i \in [0, \ell_d]$ indicates the hidden state at the i-th time step after t. The input to the decoder at the first time step is all zeros. At other time steps t, we feed in the predicted action scores $\widetilde{r}_t^{i-1}$, embedded by a linear transformer.

The future gate takes hidden states from the decoder and models the feature representation of future context. For simplicity, our default future gate is an average pooling operator followed by an fully-connected (FC) layer, but other fusion operations such as non-local (NL) blocks [46] could be used. More formally, the future context feature $\widetilde{\mathbf{x}}_t$ is obtained by averaging and embedding the hidden state vector, $\widetilde{\mathbf{h}}_t$, gathered from all decoder steps,

$$\widetilde{\mathbf{x}}_t = \text{ReLU}(\mathbf{W}_f^T \text{AvgPool}(\widetilde{\mathbf{h}}_t) + \mathbf{b}_f). \tag{1}$$

The spatiotemporal accumulator (STA) takes the previous hidden state h_{t-1} as well as the concatenation of the image feature $\mathbf{x}_t$ extracted from I_t and the predicted future feature $\widetilde{\mathbf{x}}_t$ from the future gate, and updates its hidden states h_t. It then calculates a distribution over candidate actions,

$$\mathbf{p}_t = \text{softmax}(\mathbf{W}_c^T h_t + \mathbf{b}_c), \tag{2}$$

where $\mathbf{W}_c$ and $\mathbf{b}_c$ are the parameters of the FC layer used for action classification.

As we can see, in addition to the estimated action of the current frame t, TRN outputs predicted actions for the next ℓ_d time steps. In order to ensure a good future representation and jointly optimize online action detection and prediction, we combine the accumulator and decoder losses during training, *i.e.* the loss of one input sequence is

$$\sum_t \big(loss(\mathbf{p}_t, l_t) + \alpha \sum_{i=0}^{\ell_d} loss(\widetilde{\mathbf{p}}_t^i, l_{t+i}) \big), \tag{3}$$

where $\widetilde{\mathbf{p}}_t^i$ indicates the action probabilities predicted by the decoder for step i after time t, l_t represents the ground truth, $loss$ denotes cross-entropy loss, and α is a scale factor. We optimize the network using offline training in which labels of both current and future frames are used. At test time, our model uses the *predicted* future information *without accessing actual future frames*, and thus is an online model.

4. Experiments

We evaluated our online action detector against multiple state-of-the-art and baseline methods on three publicly-available datasets: HDD [33], TVSeries [11], and THUMOS'14 [24]. We chose these datasets because they include long, untrimmed videos from diverse perspectives and applications: HDD consists of on-road driving from a first-person (egocentric) view recorded by a front-facing dashboard camera, TVSeries was recorded from television and contains a variety of everyday activities, and THUMOS'14 is a popular dataset of sports-related actions.

4.1. Datasets

HDD [33] includes nearly 104 hours of 137 driving sessions in the San Francisco Bay Area. The dataset was collected from a vehicle with a front-facing camera, and includes frame-level annotations of 11 goal-oriented actions (*e.g.*, intersection passing, left turn, right turn, *etc.*). The dataset also includes readings from a variety of non-visual sensors collected by the instrumented vehicle's Controller Area Network (CAN) bus. We followed prior work [33] and used 100 sessions for training and 37 sessions for testing.

TVSeries [11] contains 27 episodes of 6 popular TV series, totaling 16 hours of video. The dataset is temporally annotated at the frame level with 30 realistic, everyday actions (*e.g.*, pick up, open door, drink, *etc.*). The dataset is challenging with diverse actions, multiple actors, unconstrained viewpoints, heavy occlusions, and a large proportion of non-action frames.

THUMOS'14 [24] includes over 20 hours of sports video annotated with 20 actions. The training set contains only trimmed videos that cannot be used to train temporal action detection models, so we followed prior work [17] and train on the validation set (200 untrimmed videos) and evaluate on the test set (213 untrimmed videos).

4.2. Implementation Details

We implemented our proposed Temporal Recurrent Network (TRN) in PyTorch [1], and performed all experiments on a system with Nvidia Quadro P6000 graphics cards. To learn the network weights, we used the Adam [26] optimizer with default parameters, learning rate 0.0005, and weight decay 0.0005. For data augmentation, we randomly chopped off $\Delta \in [1, \ell_e]$ frames from the beginning for *each* epoch, and discretized the video of length L into $(L-\Delta)/\ell_e$ non-overlapping training samples, each with ℓ_e consecutive frames. Our models were optimized in an end-to-end manner using a batch size of 32, each with ℓ_e input sequence length. The constant α in Eq. (3) was set to 1.0.

4.3. Settings

To permit fair comparisons with the state-of-the-art [11, 17, 33], we follow their experimental settings, including input features and hyperparameters.

HDD. We use the same setting as in [33]. Video frames and values from CAN bus sensors are first sampled at 3 frames per second (fps). The outputs of the `Conv2d_7b_1x1` layer

in InceptionResnet-V2 [41] pretrained on ImageNet [13] are extracted as the visual feature for each frame. To preserve spatial information, we apply an additional 1×1 convolution to reduce the extracted frame features from $8 \times 8 \times 1536$ to $8 \times 8 \times 20$, and flatten them into 1200-dimensional vectors. Raw sensor values are passed into a fully-connected layer with 20-dimensional outputs. These visual and sensor features are then concatenated as a *multimodal* representation for each video frame. We follow [33] and set the input sequence length ℓ_e to 90. The number of decoder steps ℓ_d is treated as a hyperparameter that we cross-validate in experiments. The hidden units of both the temporal decoder and the STA are set to 2000 dimensions.

TVSeries and THUMOS'14. We use the same setting as in [17]. We extract video frames at 24 fps and set the video chunk size to 6. Decisions are made at the chunk level, and thus performance is evaluated every 0.25 seconds. We use two different feature extractors, VGG-16 [39] and two-stream (TS) CNN [47]. VGG-16 features are extracted at the `fc6` layer from the central frame of each chunk. For the two-stream features, the appearance features are extracted at the `Flatten_673` layer of ResNet-200 [21] from the central frame of each chunk, and the motion features are extracted at the `global_pool` layer of BN-Inception [25] from precomputed optical flow fields between 6 consecutive frames. The appearance and motion features are then concatenated to construct the two-stream features. The input sequence length ℓ_e is set to 64 due to GPU memory limitations. Following the state-of-the-art [17], the number of decoder steps ℓ_d is set to 8, corresponding to 2 seconds. As with HDD, our experiments report results with different decoder steps. The hidden units of both the temporal decoder and the STA are set to 4096 dimensions.

4.4. Evaluation Protocols

We follow most existing work and use per-frame **mean average precision (mAP)** to evaluate the performance of online action detection. We also use per-frame **calibrated average precision (cAP)**, which was proposed in [11] to better evaluate online action detection on TVSeries,

$$cAP = \frac{\sum_k cPrec(k) * I(k)}{P}, \tag{4}$$

where calibrated precision $cPrec = \frac{TP}{TP+FP/w}$, $I(k)$ is 1 if frame k is a true positive, P denotes the total number of true positives, and w is the ratio between negative and positive frames. The advantage of cAP is that it corrects for class imbalance between positive and negative samples.

Another important goal of online action detection is to recognize actions as early as possible; *i.e.*, an approach should be rewarded if it produces high scores for target actions at their early stages (the earlier the better). To investigate our performance at different time stages, we follow [11] and compute mAP or cAP for each decile (ten-percent interval) of the video frames separately.

4.5. Baselines

CNN baseline models [38, 39] consider online action detection as a general image classification problem. These baselines identify the action in each individual video frame without modeling temporal information. For TVSeries and THUMOS'14, we reprint the results of CNN-based methods from De Geest *et al*. [11] and Shou *et al*. [35]. For HDD, we follow Ramanishka *et al*. [33] and use InceptionResnet-V2 [41] pretrained on ImageNet as the backbone and finetune the last fully-connected layer with softmax to estimate class probabilities.

LSTM and variants have been widely used in action detection [33, 51]. LSTM networks model the dependencies between consecutive frames and jointly capture spatial and temporal information of the video sequence. For each frame, the LSTM receives the image features and the previous hidden state as inputs, and outputs a probability distribution over candidate actions.

Encoder-Decoder (ED) architectures [6] also model temporal dependencies. The encoder is similar to a general LSTM and summarizes historical visual information into a feature vector. The decoder is also an LSTM that produces predicted representations for the future sequence based only on these encoded features. Since there are no published results of ED-based methods on HDD, we implemented a baseline with the same experimental settings as TRN, including input features, hyperparameters, loss function, *etc*.

Stronger Baselines. In addition to the above basic baselines, we tested three types of stronger baselines that were designed for online action detection on TVSeries and THUMOS'14. **Convolutional-De-Convolutional (CDC)** [35] places CDC filters on top of a 3D CNN and integrates two reverse operations, spatial downsampling and temporal upsampling, to precisely predict actions at a frame-level. Note that CDC is an offline method, and comparing with CDC confirms the effectiveness of our model. **Two-Stream Feedback Network (2S-FN)** [12] is built on an LSTM with two recurrent units, where one stream focuses on the input interpretation and the other models temporal dependencies between actions. **Reinforced Encoder-Decoder (RED)** [17] with a dedicated reinforcement loss is an advanced version of ED, and currently performs the best among all the baselines for online action detection.

4.6. Results

4.6.1 Evaluation of Online Action Detection

Table 1 presents evaluation results on HDD. TRN significantly outperforms the state-of-the-art, Ramanishka *et al*. [33], by 5.4%, 2.8%, and 8.1% in terms of mAP with

Method	Inputs	Individual actions											Overall mAP
		intersection passing	L turn	R turn	L lane change	R lane change	L lane branch	R lane branch	crosswalk passing	railroad passing	merge	u-turn	
CNN	Sensors	34.2	72.0	74.9	16.0	8.5	7.6	1.2	0.4	0.1	2.5	32.5	22.7
LSTM [33]		36.4	66.2	74.2	26.1	13.3	8.0	0.2	0.3	0.0	3.5	33.5	23.8
ED		43.9	73.9	75.7	31.8	15.2	15.1	2.1	0.5	0.1	4.1	39.1	27.4
TRN		46.5	75.2	77.7	35.9	19.7	18.5	3.8	0.7	0.1	2.5	40.3	**29.2**
CNN	InceptionResNet-V2	53.4	47.3	39.4	23.8	17.9	25.2	2.9	4.8	1.6	4.3	7.2	20.7
LSTM [33]		65.7	57.7	54.4	27.8	26.1	25.7	1.7	16.0	2.5	4.8	13.6	26.9
ED		63.1	54.2	55.1	28.3	35.9	27.6	8.5	7.1	0.3	4.2	14.6	27.2
TRN		63.5	57.0	57.3	28.4	37.8	31.8	10.5	11.0	0.5	3.5	25.4	**29.7**
CNN	Multimodal	73.7	73.2	73.3	25.7	24.0	27.6	4.2	4.0	2.8	4.7	30.6	31.3
LSTM [33]		76.6	76.1	77.4	41.9	23.0	25.4	1.0	11.8	3.3	4.9	17.6	32.7
ED		77.2	74.0	77.1	44.6	41.4	36.6	4.1	11.4	2.2	5.1	43.1	37.8
TRN		79.0	77.0	76.6	45.9	43.6	46.9	7.5	13.4	4.5	5.8	49.6	**40.8**

Table 1: *Results of online action detection on HDD,* comparing TRN and baselines using mAP (%).

Method	Inputs	mcAP
CNN [11]	VGG	60.8
LSTM [11]		64.1
RED [17]		71.2
Stacked LSTM [12]		71.4
2S-FN [12]		72.4
TRN		**75.4**
SVM [11]	FV	74.3
RED [17]	TS	79.2
TRN		**83.7**

Table 2: *Results of online action detection on TVSeries,* comparing TRN and the state-of-the-art using cAP (%).

Method	mAP
Single-frame CNN [39]	34.7
Two-stream CNN [38]	36.2
C3D + LinearInterp [35]	37.0
Predictive-corrective [10]	38.9
LSTM [14]	39.3
MultiLSTM [51]	41.3
Conv & De-conv [35]	41.7
CDC [35]	44.4
RED [17]	45.3
TRN	**47.2**

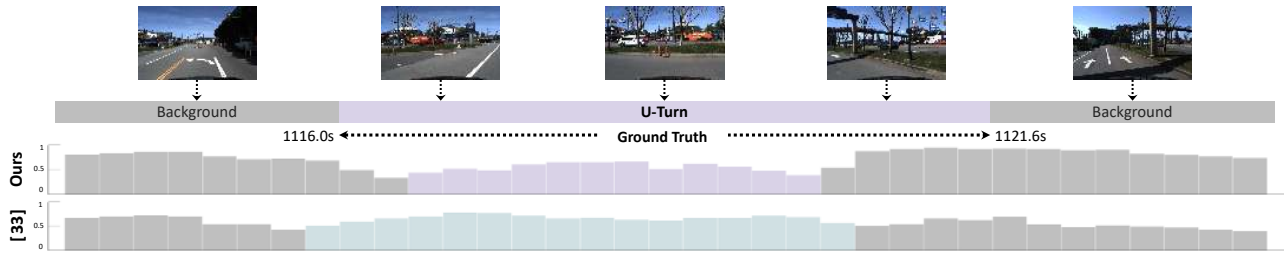
Table 3: *Results of online action detection on THUMOS'14,* comparing TRN and the state-of-the-art using mAP (%).

sensor data, InceptionResnet-v2, and multimodal features as inputs, respectively. Interestingly, the performance gaps between TRN and [33] are much larger when the input contains sensor data. Driving behaviors are highly related to CAN bus signals, such as steering angle, yaw rate, velocity, *etc*., and this result suggests that TRN can better take advantage of these useful input cues. Table 2 presents comparisons between TRN and baselines on TVSeries. TRN significantly outperforms the state-of-the-art using VGG (mcAP of 3.0% over 2S-FN [12]) and two-stream input features (mcAP of 4.5% over RED [17]). We also evaluated TRN on THUMOS'14 in Table 3. The results show that TRN outperforms all the baseline models (mAP of 1.9% over RED [17] and 2.8% over CDC [35]).

Qualitative Results are shown in Fig. 3. In Fig. 3a, we visualize and compare the results of TRN, and compare with Ramanishka *et al*. [33] on HDD. As shown, *u-turn* is difficult to classify from a first-person perspective because the early stage is nearly indistinguishable from *left turn*. With the help of the learned better representation and predicted future information, TRN differentiates between subtle differences and "looks ahead" to reduce this ambiguity. As shown in Table 1, TRN beats the baseline models on most of the actions using *multimodal* inputs, especially on "difficult" classes such as *lane branch* and *u-turn*. Qualitative results also clearly demonstrate that TRN produces not only the correct action label, but also better boundaries. Fig. 3b and 3c show promising results on TVSeries and THUMOS'14. Note that TVSeries is very challenging; for example, the drinking action in Fig. 3b of the person in the upper left of the background is barely visible.

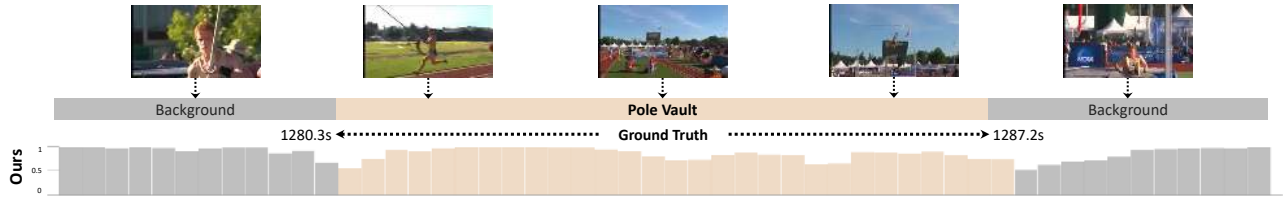
4.6.2 Ablation Studies

Importance of Temporal Context. By directly comparing evaluation results of TRN with CNN and LSTM baselines, we demonstrate the importance of explicitly modeling temporal context for online action detection. LSTMs capture long- and short-term temporal patterns in the video by receiving accumulated historical observations as input. Comparing TRN and LSTM measures the benefit of incorporating predicted action features as future context. CNN-based methods conduct online action detection by only con-



(a) Qualitative comparison between our approach and [33] on HDD dataset. *U-Turn* is shown in **purple**, *Left Turn* is shown in **green**, and *Background* is shown in **gray**.

(b) Qualitative result of our approach on TVSeries dataset. *Drink* is shown in **pink** and *Background* is shown in **gray**.



(c) Qualitative result of our approach on THUMOS'14 dataset. *Pole Vault* is shown in **yellow** and *Background* is shown in **gray**.

Figure 3: *Qualitative results of our approach and baselines on HDD, TVSeries, and THUMOS'14 datasets.* The vertical bars indicate the scores of the predicted class. (Best viewed in color.)

sidering the image features at each time step. Simonyan *et al*. [38] build a two-stream network and incorporate motion features between adjacent video frames by using optical flow as input. Table 3 shows that this motion information yields a 1.5% improvement. *TRN-TS* also takes optical flow as input and we can clearly see a significant improvement (83.7% vs. 75.4%) on TVSeries.

Future Context: An "Oracle" Study. To demonstrate the importance of using predictions of future context, we implemented an oracle baseline, *RNN-offline*. RNN-offline shares the same architecture as RNN but uses the features extracted from both the current and future frames as inputs. Note that RNN-offline uses future information and thus is not an online model; our goal is to quantify (1) the effectiveness of incorporating future information in action detection, given access to actual (instead of predicted) future information, and (2) the performance gap between estimated future information of TRN and "real" future information of RNN-offline. To permit fair comparison, the input to RNN-offline is a concatenation of the feature from the current frame and the average-pooled features of the next ℓ_d frames (where ℓ_d is the same as the number of decoder steps of TRN).

The results of RNN-offline are 41.6%, 85.3%, and 47.3% on HDD, TVSeries, and THUMOS'14 datasets, respectively. Comparing RNN-offline with the RNN baseline, we see that the "ground-truth" future information significantly improves detection performance. We also observe that the performance of TRN and RNN-offline are comparable, *even though TRN uses predicted rather than actual future information*. This may be because TRN improves its representation during learning by jointly optimizing current and future action recognition, while RNN-offline does not. We also evaluated TRN against ED-based networks, by observing that ED can also improve its representation by jointly conducting action detection and anticipation. Thus, comparisons between TRN with ED and its advanced version [17] measure how much benefit comes purely from explicitly incorporating anticipated future information.

Effect of Decoder Step Count. Finally, we evaluated the effectiveness of different decoder step counts, $\ell_d = \{4, 6, 8, 10\}$. Table 6 shows the results, with the performance of action anticipation averaged over the decoder steps. The results show that a larger number of decoder steps does not guarantee better performance. This is because anticipation accuracy usually decreases for longer future sequences, and thus creates more noise in the input features of STA. To be clear, we follow the state-of-the-art [17] and set ℓ_d to 2 video seconds (6 frames in HDD, 8 frames in TVSeries and THUMOS'14) when comparing with baseline methods of online action detection in Tables 1, 2, and 3.

Method	Inputs	Portion of video									
		0%-10%	10%-20%	20%-30%	30%-40%	40%-50%	50%-60%	60%-70%	70%-80%	80%-90%	90%-100%
CNN [11]	VGG	61.0	61.0	61.2	61.1	61.2	61.2	61.3	61.5	61.4	61.5
LSTM [11]	VGG	63.3	64.5	64.5	64.3	65.0	64.7	64.4	64.3	64.4	64.3
TRN	VGG	73.9	74.3	74.7	74.7	75.1	75.1	75.3	75.2	75.2	75.3
SVM [11]	FV	67.0	68.4	69.9	71.3	73.0	74.0	75.0	76.4	76.5	76.8
TRN	TS	78.8	79.6	80.4	81.0	81.6	81.9	82.3	82.7	82.9	83.3

Table 4: *Online action detection results when only portions of videos are considered* in terms of cAP (%) on TVSeries.

Method	Time predicted into the future (seconds)								Avg
	0.25s	0.5s	0.75s	1.0s	1.25s	1.5s	1.75s	2.0s	
ED [17]	78.5	78.0	76.3	74.6	73.7	72.7	71.7	71.0	74.5
RED [17]	79.2	78.7	77.1	75.5	74.2	73.0	72.0	71.2	75.1
TRN	79.9	78.4	77.1	75.9	74.9	73.9	73.0	72.3	**75.7**

(a) Results on TVSeries dataset in terms of *cAP* (%).

Method	Time predicted into the future (seconds)								Avg
	0.25s	0.5s	0.75s	1.0s	1.25s	1.5s	1.75s	2.0s	
ED [17]	43.8	40.9	38.7	36.8	34.6	33.9	32.5	31.6	36.6
RED [17]	45.3	42.1	39.6	37.5	35.8	34.4	33.2	32.1	37.5
TRN	45.1	42.4	40.7	39.1	37.7	36.4	35.3	34.3	**38.9**

(b) Results on THUMOS'14 dataset in terms of *mAP* (%).

Table 5: *Action anticipation results* of TRN compared to state-of-the-art methods using two-stream features.

Dataset	Task	Decoder steps (ℓ_d)			
		4	6	8	10
HDD	Online Action Detection	39.9	40.8	40.1	39.6
	Action Anticipation	34.3	32.2	28.8	25.4
TVSeries	Online Action Detection	83.5	83.4	83.7	83.5
	Action Anticipation	77.7	76.4	75.7	74.1
THUMOS'14	Online Action Detection	46.0	45.4	47.2	46.4
	Action Anticipation	42.6	39.4	38.9	35.0

Table 6: *Online action detection and action anticipation results* of TRN with decoder steps $\ell_d = 4, 6, 8, 10$.

4.6.3 Evaluation of Different Stages of Action

We evaluated TRN when only a fraction of each action is considered, and compared with published results [11] on TVSeries. For example, 20%-30% means only frames in the 20%-30% time range of action sequences were evaluated. Table 4 shows that TRN significantly outperforms existing methods at every time stage. Specifically, when we compare *TRN-TS* with the best baseline *SVM-FV*, the performance gaps between these two methods are roughly in ascending order as less and less of the actions are observed (the gaps are 6.5%, 6.4%, 6.3%, 7.3%, 7.9%, 8.6%, 9.7%, 10.5%, 11.2% and 11.8% from actions at 100% observed to those are 10% observed). This indicates the advantage of our approach at earlier stages of actions.

4.6.4 Evaluation of Action Anticipation

We also evaluated TRN on predicting actions for up to 2 seconds into the future, and compare our approach with the state-of-the-art [17] in Table 5. The results show that TRN performs better than RED and ED baselines (mcAP of 75.7% vs. 75.1% vs. 74.5% on TVSeries and mAP of 38.9% vs. 37.5% vs. 36.6% on THUMOS'14). The average of anticipation results over the next 2 seconds on HDD is 32.2% in terms of per-frame mAP.

5. Conclusion

In this paper, we propose Temporal Recurrent Networks (TRNs) to model greater temporal context, and we evaluate them on the online action detection problem. Unlike previous methods that consider only historical temporal consistencies, TRN jointly models the historical and future temporal context under the constraint of an online setting. Experimental results on three popular datasets demonstrate that incorporating predicted future information improves learned representation of actions and significantly outperforms the state-of-the-art. Moreover, TRN shows greater advantage at earlier stages of actions and in predicting future actions. More generally, we believe that our approach of incorporating estimated future information could benefit many other online tasks, such as video object localization and tracking, and plan to pursue this in future work.

Acknowledgments. This work was supported in part by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/Interior Business Center (DOI/IBC) contract number D17PC00345, the National Science Foundation (CAREER IIS-1253549), Honda Research Institute USA, and the Indiana University Office of the Vice Provost for Research, the College of Arts and Sciences, and the School of Informatics, Computing, and Engineering through the Emerging Areas of Research Project *Learning: Brains, Machines, and Children.* The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of the U.S. Government, or any sponsor. We also thank the anonymous reviewers for their very helpful suggestions.

References

[1] http://pytorch.org/. 4

[2] Andreja Bubic, D Yves Von Cramon, and Ricarda I Schubotz. Prediction, cognition and the brain. *Frontiers in Human Neuroscience*, 2010. 2

[3] Shyamal Buch, Victor Escorcia, Chuanqi Shen, Bernard Ghanem, and Juan Carlos Niebles. SST: Single-stream temporal action proposals. In *CVPR*, 2017. 1, 2

[4] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the Kinetics dataset. In *CVPR*, 2017. 2

[5] Yu-Wei Chao, Sudheendra Vijayanarasimhan, Bryan Seybold, David A Ross, Jia Deng, and Rahul Sukthankar. Rethinking the Faster R-CNN architecture for temporal action localization. In *CVPR*, 2018. 1, 2

[6] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv:1406.1078*, 2014. 5

[7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv:1412.3555*, 2014. 2, 3

[8] Andy Clark. Whatever next? Predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Sciences*, 2013. 2

[9] Xiyang Dai, Bharat Singh, Guyue Zhang, Larry S Davis, and Yan Qiu Chen. Temporal context network for activity localization in videos. In *ICCV*, 2017. 1, 2

[10] Achal Dave, Olga Russakovsky, and Deva Ramanan. Predictive-corrective networks for action detection. In *CVPR*, 2017. 6

[11] Roeland De Geest, Efstratios Gavves, Amir Ghodrati, Zhenyang Li, Cees Snoek, and Tinne Tuytelaars. Online action detection. In *ECCV*, 2016. 1, 2, 4, 5, 6, 8

[12] Roeland De Geest and Tinne Tuytelaars. Modeling temporal structure with lstm for online action detection. In *WACV*, 2018. 1, 2, 5, 6

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 5

[14] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 2, 6

[15] Grace Edwards, Petra Vetter, Fiona McGruer, Lucy S. Petro, and Lars Muckli. Predictive feedback to V1 dynamically updates with sensory input. *Scientific Reports*, 2017. 2

[16] Joseph Fruchter, Tal Linzen, Masha Westerlund, and Alec Marantz. Lexical preactivation in basic linguistic phrases. *Journal of Cognitive Neuroscience*, 2015. 2

[17] Jiyang Gao, Zhenheng Yang, and Ram Nevatia. RED: Reinforced encoder-decoder networks for action anticipation. In *BMVC*, 2017. 1, 2, 4, 5, 6, 7, 8

[18] Jiyang Gao, Zhenheng Yang, Chen Sun, Kan Chen, and Ram Nevatia. TURN TAP: Temporal unit regression network for temporal action proposals. *ICCV*, 2017. 1, 2

[19] Mingfei Gao, Mingze Xu, Larry S Davis, Richard Socher, and Caiming Xiong. Startnet: Online detection of action start in untrimmed videos. *ICCV*, 2019. 1

[20] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 2

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5

[22] Minh Hoai and Fernando De la Torre. Max-margin early event detectors. In *IJCV*, 2014. 2

[23] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997. 2, 3

[24] Haroon Idrees, Amir R Zamir, Yu-Gang Jiang, Alex Gorban, Ivan Laptev, Rahul Sukthankar, and Mubarak Shah. The THUMOS challenge on action recognition for videos in the wild. *CVIU*, 2017. 2, 4

[25] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*, 2015. 5

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014. 4

[27] Kris M Kitani, Takahiro Okabe, Yoichi Sato, and Akihiro Sugimoto. Fast unsupervised ego-action learning for first-person sports videos. In *CVPR*, 2011. 2

[28] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008. 2

[29] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks for action segmentation and detection. In *CVPR*, 2017. 3

[30] Yin Li, Zhefan Ye, and James M Rehg. Delving into egocentric actions. In *CVPR*, 2015. 2

[31] Minghuang Ma, Haoqi Fan, and Kris M Kitani. Going deeper into first-person activity recognition. In *CVPR*, 2016. 2

[32] Shugao Ma, Leonid Sigal, and Stan Sclaroff. Learning activity progression in lstms for activity detection and early detection. In *CVPR*, 2016. 2

[33] Vasili Ramanishka, Yi-Ting Chen, Teruhisa Misu, and Kate Saenko. Toward driving scene understanding: A dataset for learning driver behavior and causal reasoning. In *CVPR*, 2018. 2, 4, 5, 6, 7

[34] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 2

[35] Zheng Shou, Jonathan Chan, Alireza Zareian, Kazuyuki Miyazawa, and Shih-Fu Chang. CDC: Convolutional-deconvolutional networks for precise temporal action localization in untrimmed videos. In *CVPR*, 2017. 2, 5, 6

[36] Zheng Shou, Junting Pan, Jonathan Chan, Kazuyuki Miyazawa, Hassan Mansour, Anthony Vetro, Xavier Giro-i Nieto, and Shih-Fu Chang. Online action detection in untrimmed, streaming videos-modeling and evaluation. In *ECCV*, 2018. 1, 2

[37] Zheng Shou, Dongang Wang, and Shih-Fu Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *CVPR*, 2016. 1, 2

[38] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 2, 5, 6, 7

[39] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014. 5, 6

[40] Waqas Sultani, Chen Chen, and Mubarak Shah. Real-world anomaly detection in surveillance videos. In *CVPR*, 2018. 2

[41] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv:1602.07261*, 2016. 5

[42] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 2

[43] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *CVPR*, 2011. 2

[44] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Dense trajectories and motion boundary descriptors for action recognition. *IJCV*, 2013. 2

[45] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016. 2

[46] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. 4

[47] Yuanjun Xiong, Limin Wang, Zhe Wang, Bowen Zhang, Hang Song, Wei Li, Dahua Lin, Yu Qiao, Luc Van Gool, and Xiaoou Tang. CUHK & ETHZ & SIAT submission to activitynet challenge 2016. *arXiv:1608.00797*, 2016. 5

[48] Huijuan Xu, Abir Das, and Kate Saenko. R-C3D: Region convolutional 3d network for temporal activity detection. In *ICCV*, 2017. 2

[49] Mingze Xu, Aidean Sharghi, Xin Chen, and David Crandall. Fully-coupled two-stream spatiotemporal networks for extremely low resolution action recognition. In *WACV*, 2018. 2

[50] Yu Yao, Mingze Xu, Yuchen Wang, David J Crandall, and Ella M Atkins. Unsupervised traffic accident detection in first-person videos. *IROS*, 2019. 1

[51] Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. In *IJCV*, 2018. 5, 6

[52] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015. 2

[53] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. Temporal action detection with structured segment networks. In *ICCV*, 2017. 1