Action Completion: A Temporal Model for Moment Detection

Farnoosh Heidarivincheh
Farnoosh.Heidarivincheh@bristol.ac.uk
Majid Mirmehdi
M.Mirmehdi@bristol.ac.uk
Dima Damen
Dima.Damen@bristol.ac.uk

Department of Computer Science
University of Bristol
Bristol, UK

Abstract

We introduce *completion moment* detection for actions - the problem of locating the *moment* of completion, when the action's goal is confidently considered achieved. The paper proposes a joint classification-regression recurrent model that predicts completion from a given frame, and then integrates frame-level contributions to detect sequence-level *completion moment*. We introduce a recurrent voting node that predicts the frame's relative position of the *completion moment* by either classification or regression. The method is also capable of detecting incompletion. For example, the method is capable of detecting a missed ball-catch, as well as the *moment* at which the ball is safely caught. We test the method on 16 actions from three public datasets, covering sports as well as daily actions. Results show that when combining contributions from frames prior to the *completion moment* as well as frames post completion, the *completion moment* is detected within one second in 89% of all tested sequences.

1 Introduction

An action, based on the Oxford Dictionary, is *the fact or process of doing something, typically to achieve an aim*. Previous works on action recognition from visual data, such as [3, 11, 17], have overlooked assessing whether the action's *aim* has actually been achieved, rather than merely attempted. The closely related action localisation problem, e.g. in [7, 9, 20, 25], predicts the temporal start and end of an action's attempt, without assessing whether the *aim* has been achieved either. The notion of assessing an action's completion was introduced in [8], with follow-up works [2, 4] that focus on measuring the action's progress under a linear assumption, or predicting the time till the next action. In this work, we attempt to detect (or locate) the *moment* in time when the action can indeed be considered completed.

We define the problem of *completion moment* detection as detecting the frame that separates pre-completion from post-completion per sequence, when present. Note that the *completion moment* is different from the typical 'start'/'end' frames in action localisation. The former focuses on the action's goal, while the latter separates the motion relevant to the action from other actions or background frames. For example, in action 'drink', the start of the action for localisation tends to be when a glass is lifted for drinking, and the end is when it is

placed down. Conversely, the *completion moment* we are after, is when the person consumes part of the beverage, marking their goal of drinking being achieved. The subtle nature of this *completion moment* thus requires a framework that is capable of robust moment detection.

Moment detection, including action *completion moment* detection, has potential applications in robot-human collaboration, health-care or assisted-living, where an agent can react to a human completing the goal or conversely, failing to complete the action. For example, switching the oven off, could trigger safety alarms.

In detecting the moment of completion, we take a supervised approach, where for training sequences, the *completion moment* is labeled when present (see Sec 3). Our proposed method uses a Convolutional-Recurrent Neural Network (C-RNN), and outputs per-frame votes for the presence and relative position of the *completion moment*. We then predict a sequence-level *completion moment* by accumulating these frame-level contributions. To showcase the generality of our method, we evaluate it on 16 actions from 3 public datasets [8, 12, 19]. These include sports-based (e.g. *basketball, pole vault*) as well as daily (e.g. *drink, pour*) actions. We show that both pre-completion and post-completion frames assist in *completion moment* detection for the variety of tested actions.

The remainder of this paper is organised as follows: related work in Sec. 2, problem definition in Sec. 3, proposed method in Sec. 4, experiments and results in Sec. 5 and conclusion and future work in Sec. 6.

2 Related Work

Current methods for action recognition focus on deploying convolutional neural networks (CNNs), either dual-stream convolutions [6, 17, 23] or 3D convolution filters from video snippets [11, 16, 21], as well as recurrent neural networks (RNNs) that accumulate evidence from frames over a sequence [3, 26, 28]. However, these approaches aim to label the sequence as a whole. One seminal work [22] deviates by encoding the action as precondition and effect, using a Siamese network that predicts the action as a transformation between the two states. In this section, we review related works that study partial observations within a video sequence for three problems of relevance to our proposed *moment detection* problem,

Action Proposal Generation: Action proposals and action-ness measures have become the platform for several action localisation approaches [7, 10, 24, 27, 29]. Among these, [29] and [24] focus on classifying these proposals into those that contain the 'completed' action, and incomplete proposals that should be rejected. While [24] applies an SVM to filter and reject spatio-temporal proposals containing incomplete or partial actions, [29] has embedded the rejection within an end-to-end CNN. These approaches classify each proposal, and do not attempt to locate or assess the *completion moment*.

Action Anticipation: A few recent works [4, 15] focus on predicting the class label of the next unobserved action. Mahmud et al. [15] predict the next action as well as its starting time using a hybrid Siamese network in which an LSTM is used for temporal modelling. Farha et al. [4] estimate the time remaining until the next action, as well as the length and the label of the next action. The paper compares the usage of either an RNN or a CNN that takes as input concatenated frame-level features into a single tensor. These approaches do not discuss completion (or incompletion) of the observed action.

Early Detection: Several works [1, 2, 9, 13, 14, 25] address early detection of partially observed actions, from as few frames as possible. These mainly propose loss functions to encourage early detection [1, 14], but a few works attempt fine-grained understanding of the action's progression. In [9], an SVM classifier is trained to accumulate scores from partial

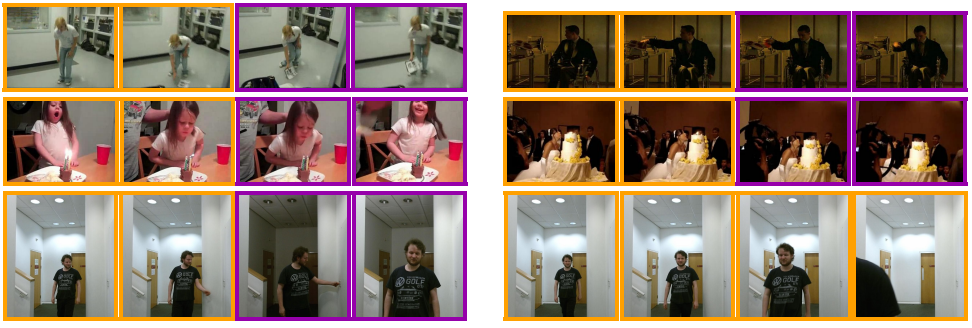

Figure 1: Annotation of *completion moment*: Two examples per action. *Pre-completion* frames are bordered in orange and *post-completion* in purple. **From Top:** HMDB *pick*, UCF101 *blowing candles*, RGBD-AC *switch* (one complete sequence and one incomplete).

observations of the action, where the score is highest when the action is fully observed. The approach has been tested on facial and gesture datasets. Similarly, in [2], an RNN is trained to predict the action label, as well as its linear progress towards its conclusion as a percentage (e.g. 50% of the action has taken place).

Two approaches [13, 25] which detect moments within the sequence have been proposed, albeit for early detection and localisation. In [25] individual frames predict the location of the next frame to be observed, using an RNN. The work aims for action detection with as few frames as possible, thus the trained model proposes transitions within the sequence, by predicting the relative position of the frame to be observed next. Our work is inspired by ideas in [13], where action detection uses a joint classification-regression RNN. The classification branch predicts the ongoing action label which is then used by a regression branch to predict the start and the end points of the action, relative to the current frame. A Gaussian scoring function is used to encode the prediction uncertainty. The approach was tested on 3D skeletal data for localisation, oblivious to the action's completion (or incompletion).

None of the works mentioned above consider whether the action actually achieves its aim. In this work, we build on our previous work that introduced the action completion problem [8] by classifying whole sequences into complete and incomplete, and take inspiration from [13] to propose a joint classification-regression architecture. As opposed to predicting the next or the ongoing action, we detect the *completion moment* by accumulating evidence from frame-level decisions. We further define the *completion moment* detection problem in the next section.

3 Action Completion - A Moment in Time

We first present our proposal for formulating the problem of localising an action's completion as detecting a moment in time, beyond which the action's goal is believed to be completed by a human observer. We make three reasonable assumptions:

- **Momentary Completion**: We aim to detect a single frame in the sequence - that is the first frame where a human observer would be sufficiently confident that the goal has been achieved. We refer to frames prior to the *completion moment* as *pre-completion* frames, and those from the *completion moment* onwards as *post-completion* frames.
- **Temporally Segmented Sequences:** We assume that the action is attempted during each

sequence, in train or test, at least once, but not necessarily completed. We aim to detect the first *completion moment* per sequence, if at all, or label the attempt as being incomplete.

- **Consistent Labeling**: For each action, we assume annotators are given a non-ambiguous definition of the *completion moment*, so all train and test sequences are labeled consistently. For example, in the action *'blowing candle'*, the consistent label for the *completion moment* should indicate the moment when the flames of all candles go out. Note that the proposed model is independent of the definition of the *completion moment* per action. It only assumes the moment is consistently labeled across sequences.

Figure 1 shows sample sequences, labeled with *completion moments*, for three actions from the various datasets we annotate and use: (i) *pick* from HMDB, where the *completion moment* is when the object is lifted off the surface (ii) *blowing candles* from UCF101, where the *completion moment* is when all the candles are blown out and (iii) *switch light* from RGBD-AC, where *completion moment* is when the room's illumination changes.

Labeled sequences for a given action are the input to our method, presented next. For each sequence i, one *completion moment* is labeled if present, which we refer to as τ_i, such that $1 \leq \tau_i \leq T_i$, and T_i is the sequence length, or the sequence is labeled as incomplete.

4 Temporal Model for Moment Detection

To detect the *completion moment* within a sequence, one could naively attempt to train a classifier that singularly separates the frame indicating the *completion moment* from the rest of the video. However, evidence for the *completion moment* can be collected from all (or any) frames in the sequence. Take for example the action '*pick*'; the pose of the person is likely to change and evolve as they approach the object to be picked, and similarly observing the object in hand as the hand retracts gives further support for completion. We propose a temporal model that learns local (i.e. frame-level) predictions, within a recurrent neural network, towards global (i.e. sequence-level) detection, trainable end-to-end.

Our proposed temporal model is a Convolutional-Recurrent Neural Network. We describe the frame-level voting nodes in Sec 4.1 and then show how the unfolded temporal model, over a sequence, can accumulate votes towards moment detection in Sec 4.2.

4.1 Frame-level Voting Recurrent Node

Each frame in the sequence, whether prior to the *completion moment*, or post completion, could contribute to the *completion moment* detection. We refer to this contribution as 'voting', i.e. a frame can vote for when the action will be (or has been) completed. Two ways are proposed in which such voting can take place:

1. Classification Voting: At each time step t, the sequence is split into two parts: $[1 \cdots t]$ and $[t+1 \cdots T]$, where T is the sequence length. The classification vote primarily distinguishes the split within which the *completion moment* resides.
2. Regression Voting: At each time step t, the relative position of the *completion moment* is predicted, normalised to allow for sequences of various lengths.

Figure 2 shows the architecture of our proposed frame-level voting recurrent node, which can be used to predict both the classification and regression votes defined above, trained using a joint classification-regression loss function. Each input frame is passed through convolutional, pooling and fully connected layers. Then, an LSTM layer combines past

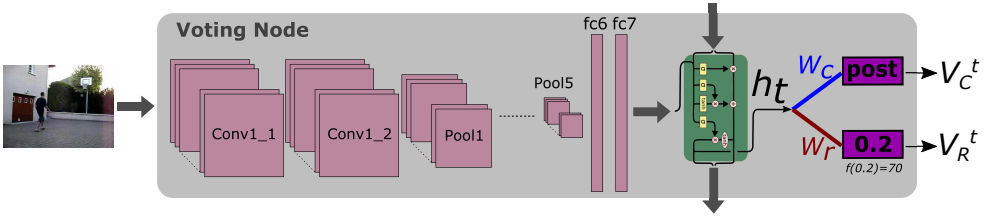

Figure 2: The input image passes through convolutional, pooling and fully-connected layers, and then an LSTM cell to capture temporal dependencies from the past. The node outputs classification V_C^t and regression V_R^t votes for the *completion moment*.

information with the current observation. The LSTM output h_t is trained to perform frame-level classification as well as frame-level regression as follows:

Frame-Level Classification Voting (V_C^t): To decide whether the *completion moment* is before or after the current time step t, we primarily need to predict whether the current observation is *pre-* or *post-* completion. We thus train for V_C^t by classifying the current observation, using a Sigmoid cross-entropy loss function on top of the LSTM hidden layer, such that

$$C_t = g(W_c h_t + B_c)\,, \tag{1}$$
$$L_{C_t} = -(y_t log(C_t) + (1-y_t) log(1-C_t))\,, \tag{2}$$

where W_c and B_c are the weights and biases for classification, respectively, g is the Sigmoid activation function and y_t is the supervised label. The *pre-* and *post-* class labels are assigned to all frames $t < \tau$ and $t \geq \tau$, respectively; sequence subscript is removed for simplicity.

The classifier then allows to vote for the presence of the *completion moment* in one of two splits of the sequence, namely $[1 \cdots t]$ or $[t+1 \cdots T]$. Specifically, if the observation at time t is classified as being *pre*-completion, then the *completion moment* is believed to be within $[t+1 \cdots T]$, or could be incomplete. To account for incompletion, we extend the end of the second split to $T+1$, to allow votes to be cast for an incomplete sequence, so the second split becomes $[t+1 \cdots T+1]$. Otherwise, the *completion moment* is believed to be within $[1 \cdots t]$. The classification vote contributes equally to voting within the split. We define V_C^t as a one-dimensional vector of length $T+1$, representing the vote assigned to all frames in the sequence. For each frame j, the vote cast by the current frame t, $V_C^t(j)$, is

$$V_C^t(j) = \begin{cases} \frac{1}{T-t+1} & j > t \ \wedge \ C_t = \text{pre} \\ \frac{1}{t} & j \leq t \ \wedge \ C_t = \text{post} \\ 0 & \text{otherwise} \end{cases} \qquad \forall j = 1 \cdots T+1. \tag{3}$$

The frame-level classification votes V_C^t are then accumulated (see Sec. 4.2).

Frame-Level Regression Voting (V_R^t): While V_C^t assigns an equal vote to all frames within each of the splits in the sequence, defined by t, regression voting V_R^t provides stronger evidence that can localise the *completion moment*, by predicting its relative position to t. This relative position encapsulates the *remaining* time to or *elapsed* time from the *completion moment*. We compute the relative time as that between t and the *completion moment* τ, normalised by τ, i.e. $\frac{t-\tau}{\tau}$. This provides a more robust relative temporal position than the alternative $\frac{t-\tau}{T}$ which would differ with the length of the sequence. Note that this value is negative during pre-completion, that is $t < \tau$.

To train for frame-level regression, the hidden output h_t in the voting recurrent node learns to predict the relative time, using a Euclidean loss function, to obtain

$$R_t = W_r h_t + B_r \,, \tag{4}$$

$$L_{R_t} = (R_t - \frac{(t-\tau)}{\tau})^2 \,, \tag{5}$$

where W_r and B_r are the weights and biases for regression, respectively. R_t can then be used to predict the *completion moment* at the corresponding time t as $f(t,R_t) = \frac{t}{R_t+1}$.

Similar to classification voting, we define V_R^t as a one-dimensional voting vector, and use a Gaussian with uncertainty σ around the predicted *completion moment*, $f(t,R_t)$, such that

$$V_R^t(j) = \beta e^{\frac{-(j-f(t,R_t))^2}{2\sigma^2}} \qquad \forall j = 1 \cdots T+1, \tag{6}$$

where β represents the inverse of the selected area under curve of the Gaussian. Experimentally, we only compute the regression vote within a window of size αT, in order to reduce the complexity of calculating the vote for all time steps in the sequence.

Training Loss: As a forward recurrent neural network, we can then train all parameters using a combined loss on all sequences and their frames, specified as,

$$L = \sum_i \Big(\frac{1}{T_i} \sum_{t_i=1}^{T_i} (L_{C_{t_i}} + L_{R_{t_i}}) \Big), \tag{7}$$

allowing all sequences to contribute equally to the loss function regardless of the sequence length. The loss is propagated back through the recurrent voting nodes.

4.2 Sequence-level prediction of *completion moment*

The votes by individual frames are accumulated to make sequence-level predictions of the *completion moment*. Note that we do not propagate ambiguity in the decisions of the individual frames, and assume each frame is equally certain about its votes. Other approaches that could integrate frame voting uncertainty, or learn temporal attention, are left for future investigation. We focus on assessing the robustness of using the classification vs the regression votes as follows: **(i) Classification$_{pre}$-Classification$_{post}$ (C-C)**: all frames use classification-based voting, **(ii) Regression$_{pre}$-Regression$_{post}$ (R-R):** all frames vote using their regression-based voting, **(iii) Regression$_{pre}$-Classification$_{post}$ (R-C):** frames classified as *pre-completion* use their regression-based voting, while *post-* frames use classification-based voting, and correspondingly **(iv) Classification$_{pre}$-Regression$_{post}$ (C-R)**. Symbolically,

$$V_{C-C} = \sum_t V_C^t \quad (8) \qquad\qquad V_{R-C} = \sum_{t:C_t=\text{pre}} V_R^t + \sum_{t:C_t=\text{post}} V_C^t \quad (10)$$

$$V_{R-R} = \sum_t V_R^t \quad (9) \qquad\qquad V_{C-R} = \sum_{t:C_t=\text{pre}} V_C^t + \sum_{t:C_t=\text{post}} V_R^t \quad (11)$$

Fig. 3 illustrates the various approaches to frame-based votes. The predicted sequence-level *completion moment* τ^p is then the frame with the maximum accumulative vote:

$$\tau^p = \arg\max_j V(j). \tag{12}$$

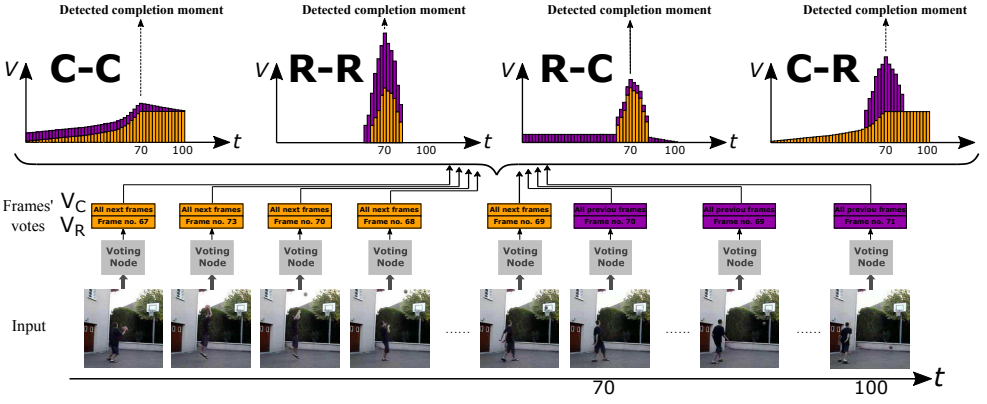

Figure 3: Sequence-level completion detection by accumulating frames' votes. The schemes use classification and/or regression voting. Sample sequence from action *basketball*.

5 Experiments and Results

Dataset and Completion Annotation: To show the generality of our work, we select 16 actions from 3 public datasets, and annotate them for their *completion moments*. We avoid actions for which completion would be difficult to define or it just marks the end of the action, *e.g. run, play piano, laugh*. However, we select actions that cover both sports-based and daily actions. For each sequence, we provide an annotation of the first *completion moment*, by a single annotator[1].

HMDB [12]: We annotate all sequences of 5 actions: *catch, drink, pick, pour* and *throw*. In total, these are 494 sequences, of which 93.5% are complete, i.e. the action's goal is successfully achieved. While HMDB does not aim for completion detection, a few sequences include attempts that are unsuccessful.

UCF101 [19]: We annotate all sequences of 5 actions: *basketball, blowing candles, frisbee catch, pole vault* and *soccer penalty*. These are 650 sequences, of which 80.5% are complete.

RGBD-AC [8]: We use the RGB input our previously introduced dataset [8], and annotate all 414 sequences which include 6 actions: *switch, plug, open, pull, pick* and *drink*, of which 50.5% are complete. In this dataset, subjects are disrupted from completing the action, e.g. a drawer they attempt to open is locked.

We apply 'leave-one-person-out' to evaluate the RGBD-AC dataset, while for HMDB and UCF101, the provided train and test splits are used.

Implementation Details: For the convolution and pooling layers, we use the spatial stream CNN from [17] which uses the VGG-16 architecture [18], pre-trained on UCF101. This CNN is then fine-tuned per action, using the two classes of *pre-* and *post-completion* frames. For fine-tuning, 20 epochs are performed, and the learning rate is started at 10^{-3}, divided by 10 at epochs 3 and 5. All the other hyper-parameters are set as in [5].

The 4096-dimension vector of $fc7$ forms the input to the single LSTM layer with 128 hidden units. Initialisation is random, trained for 10 epochs. The learning rate is 10^{-2} for the first 5 epochs and is fixed at 10^{-3} for the remaining epochs. We use a mini-batch size of one sequence and parameters α, β and σ are set 0.1, 0.5 and 30, respectively. While

[1] Annotations available at: https://github.com/FarnooshHeidari/CompletionDetection.

		Accuracy						RD					
		Pre-V	V_R^I	C-C	R-R	R-C	C-R	Pre-V	V_R^I	C-C	R-R	R-C	C-R
HMDB	*catch*	77.3	79.1	75.9	80.5	76.7	**82.3**	0.23	0.21	0.24	0.20	0.23	**0.18**
	drink	77.3	69.3	73.2	78.0	75.9	**80.5**	0.21	0.31	0.27	0.22	0.24	**0.19**
	pick	80.6	79.5	79.7	79.9	74.7	**84.2**	0.22	0.20	0.20	0.20	0.25	**0.16**
	pour	76.5	68.3	71.1	80.0	78.7	**81.2**	0.23	0.32	0.29	0.20	0.21	**0.19**
	throw	68.7	74.3	63.4	74.6	65.8	**80.4**	0.32	0.26	0.37	0.25	0.34	**0.20**
UCF101	*basketball*	**86.5**	78.0	84.5	79.5	79.1	85.1	0.21	0.22	0.16	0.20	0.21	**0.15**
	blowing candles	86.8	88.3	86.4	84.2	78.2	**90.9**	0.16	0.12	0.14	0.16	0.22	**0.09**
	frisbee catch	81.7	84.1	80.3	78.3	74.6	**85.9**	0.24	0.16	0.20	0.22	0.25	**0.14**
	pole vault	85.0	83.3	82.6	88.4	80.1	**90.6**	0.19	0.17	0.17	0.12	0.20	**0.09**
	soccer penalty	85.5	86.6	85.8	87.1	85.6	**88.5**	0.15	0.13	0.14	0.13	0.14	**0.11**
RGBD-AC	*switch*	**99.9**	93.9	**99.9**	98.1	92.7	98.9	**0.00**	0.06	**0.00**	0.02	0.07	0.01
	plug	98.3	93.2	**98.5**	96.1	93.0	97.2	0.02	0.07	**0.01**	0.04	0.07	0.03
	open	**91.1**	86.1	**91.1**	86.7	80.4	89.9	0.12	0.14	**0.09**	0.13	0.20	0.10
	pull	97.7	89.1	**97.8**	94.1	91.5	97.0	0.10	0.11	**0.02**	0.06	0.08	0.03
	pick	91.5	89.1	89.9	93.2	83.6	**95.0**	0.11	0.11	0.10	0.07	0.16	**0.05**
	drink	88.6	79.0	85.3	90.9	85.8	**92.1**	0.11	0.21	0.15	0.09	0.14	**0.08**
complete		82.3	78.1	79.6	83.1	77.7	**85.6**	0.19	0.22	0.20	0.17	0.22	**0.14**
incomplete		93.4	94.8	94.3	90.4	88.8	**96.1**	0.13	0.05	0.06	0.10	0.11	**0.04**
total		85.0	82.2	83.2	84.9	80.4	**88.1**	0.17	0.18	0.17	0.15	0.20	**0.12**

Table 1: Results on all 16 actions, comparing frame-level classification, last-frame regression and the four sequence-level voting schemes.

the proposed method represents an end-to-end trainable model, in the presented results, we train a CNN and feed the $fc7$ features into the LSTM. Efficient end-to-end training of the proposed temporal model is challenging using available hardware, and is left for future work.

Evaluation Metrics: We assess the proposed model using two evaluation metrics, (i) Accuracy: for every sequence, we compute the ratio of frames that are consistently labeled as *pre-* or *post-* the *completion moment*, given the predicted τ_i^p and labeled τ_i^g moments,

$$\text{Accuracy} = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{T_i}\sum_{t_i}\left(t_i < \tau_i^p \ \wedge\ t_i < \tau_i^g\right) \vee \left(t_i \geq \tau_i^p \ \wedge\ t_i \geq \tau_i^g\right), \tag{13}$$

where M is the number of sequences, and (ii) the average relative distance error (RD) in predicting the *completion moment*,

$$RD = \frac{1}{M}\sum_{i=1}^{M}\frac{||\tau_i^p - \tau_i^g||}{T_i}. \tag{14}$$

Results: In Table 1, C-R voting outperforms R-R and R-C in all actions of the three datasets. It also outperforms C-C for most actions. This outcome shows that *pre-completion* frames, while confident about the *completion moment* being later, are unable to robustly predict the remaining time to completion. In contrast, a *post-completion* frame which has indeed observed the completion moment is able to have a more reliable prediction of its relative position via regression. This also explains the poor results of method R-C in which only *pre-completion* frames use regression-based voting.

In Table 1, we also show two baselines: **(i) Pre-Voting** (Pre-V): the classification output of the LSTM hidden layer is used solely without voting. This frame-level result can have fluctuations as shown in Fig. 4. In this case, we use the first predicted *post-completion* frame as τ_i^p. For HMDB and UCF101 datasets, the proposed method outperforms the frame-level classification, while for RGBD-AC, they perform comparably. This is because the RGBD-AC dataset is captured in one environment with a single viewpoint and thus the frame-level classifications tend to generalise easily to new sequences. Note that for action *basketball* from UCF101, while Pre-V performs highly on the accuracy evaluation metric, the *RD* error is higher than that of our proposed method. **(ii) Last-frame regression (V_R^T):** We only use the regression vote of the last frame. As a forward RNN is used one might question whether

Figure 4: Sample results for four sequences: *soccer penalty, pick, pole vault* and *pour*.

the accumulated result at the end of the sequence is sufficient. We show that this result is less robust than accumulating votes from all frames. Table 1 also summarises the results of complete and incomplete test sequences separately. Further action-specific results for complete and incomplete sequences are included in supplementary material.

Four qualitative examples are presented in Fig. 4. (1) For *soccer penalty*, only C-R matches exactly the ground-truth with Pre-V and C-C giving comparable results. Using regression-voting for pre-completion frames negatively affects the *completion moment* detection. (2) An incomplete *pick* is correctly recognized by both C-R and C-C voting methods. (3) For *pole vault*, fluctuating frame-level classifications are shown. C-R provides the closest estimation for the *completion moment*. (4) For *pour*, the *completion moment* when the liquid is poured is predicted 5 frames earlier when using C-R voting, compared to 10 frames when R-R is used [2].

We also present the accumulative percentage of sequences which detect the *completion moment* within a certain threshold in Fig. 5. We define that threshold as the absolute difference, in frames, between the predicted and ground-truth *completion moments*. Results are shown for the C-R method. We correctly detect the completion moment within 1 second (30 frames) in 89% of all test sequences, and within 0.5 second (15 frames) in 74% of sequences. Also *completion moment* is detected for 30.4% of sequences at the very same

[2] Video of results is available at: https://youtu.be/Hrxehk3Sutc.

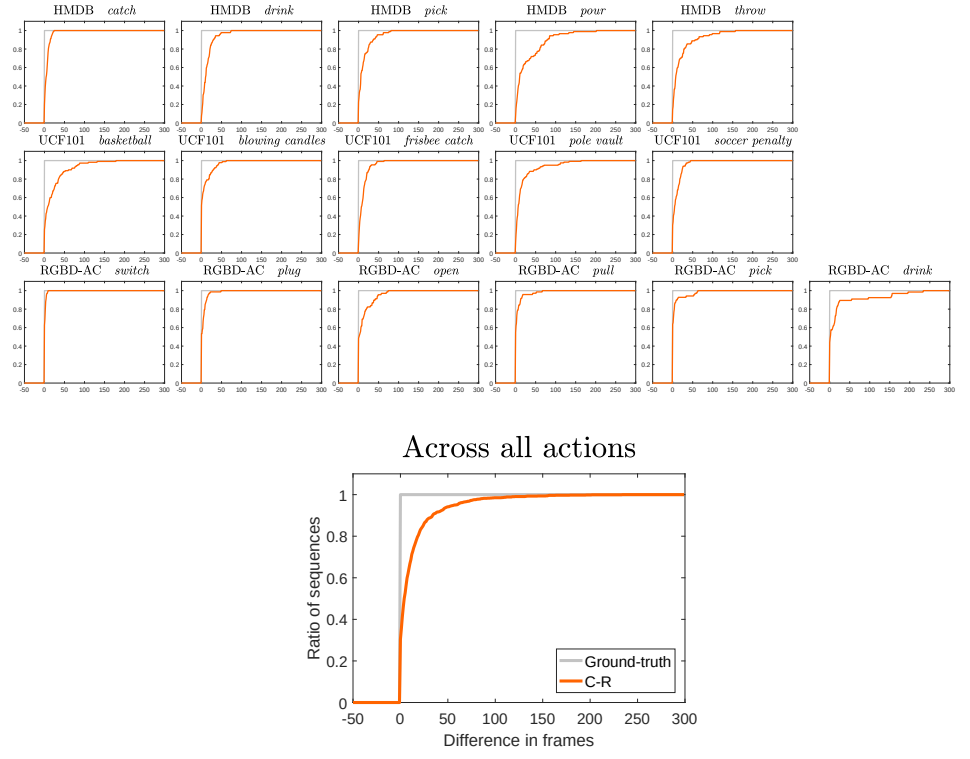

Figure 5: Cumulative percentage of sequences where the completion moment is detected within x frames. Acceptance threshold x is shown on the x-axis. Results are shown for each of the 16 actions as well as across all actions.

frame as ground-truth (i.e. 0 frame difference). Graphs are plotted for each of the 16 actions as well as the total of all actions.

6 Conclusion and Future Work

This paper presents action *completion moment* detection as the task of localising the moment in time when a human observer believes an action's goal has been achieved. The approach goes beyond recognition of completion towards a fine-grained perception of completion. We use a supervised approach for detecting completion per action, and propose an end-to-end trainable recurrent model. We show that individual frames can contribute to predicting a sequence-level *completion moment* via voting, and propose four methods to accumulate frame-level votes. Results show that using classification voting for pre-completion frames, and regression-voting for post-completion frames achieves the overall best result.

We foresee the proposed temporal model as a powerful learning method for moment detection in actions, for and beyond action completion. We aim to pursue two directions for future work. First, we plan to extend our work to untrimmed videos and propose temporal models able to detect multiple *completion moments*. Second, we shall explore weakly-supervised approaches to *completion moment* detection.

References

[1] M. S. Aliakbarian, F. S. Saleh, M. Salzmann, B. Fernando, L. Petersson, and L. Andersson. Encouraging LSTMs to anticipate actions very early. In *CVPR*, 2017.

[2] F. Becattini, T. Uricchio, L. Ballan, L. Seidenari, and A. Del Bimbo. Am I done? predicting action progress in videos. *arXiv preprint arXiv:1705.01781*, 2018.

[3] 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 *CVPR*, 2015.

[4] Y. A. Farha, A. Richard, and J. Gall. When will you do what? anticipating temporal occurrences of activities. In *CVPR*, 2018.

[5] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016.

[6] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Spatiotemporal multiplier networks for video action recognition. In *CVPR*, 2017.

[7] G. Gkioxari and J. Malik. Finding action tubes. In *CVPR*, 2015.

[8] F. Heidarivincheh, M. Mirmehdi, and D. Damen. Beyond action recognition: Action completion in RGB-D data. In *BMVC*, 2016.

[9] M. Hoai and F. De la Torre. Max-margin early event detectors. *IJCV*, 2014.

[10] M. Jain, J. Van Gemert, H. Jégou, P. Bouthemy, and C. Snoek. Action localization with tubelets from motion. In *CVPR*, 2014.

[11] S. Ji, W. Xu, M. Yang, and K. Yu. 3D convolutional neural networks for human action recognition. *PAMI*, 2013.

[12] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011.

[13] Y. Li, C. Lan, J. Xing, W. Zeng, C. Yuan, and J. Liu. Online human action detection using joint classification-regression recurrent neural networks. In *ECCV*, 2016.

[14] S. Ma, L. Sigal, and S. Sclaroff. Learning activity progression in LSTMs for activity detection and early detection. In *CVPR*, 2016.

[15] T. Mahmud, M. Hasan, and A. K. Roy-Chowdhury. Joint prediction of activity labels and starting times in untrimmed videos. In *ICCV*, 2017.

[16] Z. Shou, D. Wang, and S. Chang. Temporal action localization in untrimmed videos via multi-stage CNNs. In *CVPR*, 2016.

[17] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*. 2014.

[18] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

[19] K. Soomro, A. Roshan Zamir, and M. Shah. A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[20] Y. Tian, R. Sukthankar, and M. Shah. Spatiotemporal deformable part models for action detection. In *CVPR*, 2013.

[21] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. In *ICCV*, 2015.

[22] X. Wang, A. Farhadi, and A. Gupta. Actions ~ transformations. In *CVPR*, 2016.

[23] Y. Wang, M. Long, J. Wang, and P. S. Yu. Spatiotemporal pyramid network for video action recognition. In *CVPR*, 2017.

[24] Y. Xiong, Y. Zhao, L. Wang, D. Lin, and X. Tang. A pursuit of temporal accuracy in general activity detection. *arXiv preprint arXiv:1703.02716*, 2017.

[25] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *CVPR*, 2016.

[26] S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and F.-F. Li. Every moment counts: Dense detailed labeling of actions in complex videos. *IJCV*, 2018.

[27] G. Yu and J. Yuan. Fast action proposals for human action detection and search. In *CVPR*, 2015.

[28] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015.

[29] Y. Zhao, Y. Xiong, L. Wang, Z. Wu, X. Tang, and D. Lin. Temporal action detection with structured segment networks. In *ICCV*, 2017.