Am I Done? Predicting Action Progress in Videos

FEDERICO BECATTINI, TIBERIO URICCHIO, and LORENZO SEIDENARI,
University of Florence, Italy
LAMBERTO BALLAN, University of Padova, Italy
ALBERTO DEL BIMBO, University of Florence, Italy

In this article, we deal with the problem of predicting action progress in videos. We argue that this is an extremely important task, since it can be valuable for a wide range of interaction applications. To this end, we introduce a novel approach, named ProgressNet, capable of predicting *when* an action takes place in a video, *where* it is located within the frames, and *how far* it has progressed during its execution. To provide a general definition of action progress, we ground our work in the linguistics literature, borrowing terms and concepts to understand which actions can be the subject of progress estimation. As a result, we define a categorization of actions and their phases. Motivated by the recent success obtained from the interaction of Convolutional and Recurrent Neural Networks, our model is based on a combination of the Faster R-CNN framework, to make framewise predictions, and LSTM networks, to estimate action progress through time. After introducing two evaluation protocols for the task at hand, we demonstrate the capability of our model to effectively predict action progress on the UCF-101 and J-HMDB datasets.

CCS Concepts: • **Computing methodologies** → **Activity recognition and understanding**;

Additional Key Words and Phrases: Action progress, action recognition

ACM Reference format:
Federico Becattini, Tiberio Uricchio, Lorenzo Seidenari, Lamberto Ballan, and Alberto Del Bimbo. 2020. Am I Done? Predicting Action Progress in Videos. *ACM Trans. Multimedia Comput. Commun. Appl.* 16, 4, Article 119 (December 2020), 24 pages.
https://doi.org/10.1145/3402447

1 INTRODUCTION

Humans are not only able to recognize actions and activities, but also they can understand how far an action has progressed and make important decisions based on this information. From simple choices, like crossing the street when cars have passed, to more complex activities like intercepting the ball in a basketball game, an intelligent agent has to recognize and understand how far an

This work was partially done while L. Ballan was at the University of Florence, Italy.
The authors gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan XP GPU used for this research. Lamberto Ballan is partially supported by the PRIN 2017 project "PREVUE - PRediction of activities and Events by Vision in an Urban Environment."
Authors' addresses: F. Becattini, T. Uricchio, L. Seidenari, and A. Del Bimbo, University of Florence, Florence, Italy; emails: {federico.becattini, tiberio.uricchio, lorenzo.seidenari, alberto.delbimbo}@unifi.it; L. Ballan, University of Padova, Padova, Italy; email: lamberto.ballan@unipd.it.

1551-6857/2020/12-ART119 $15.00
https://doi.org/10.1145/3402447

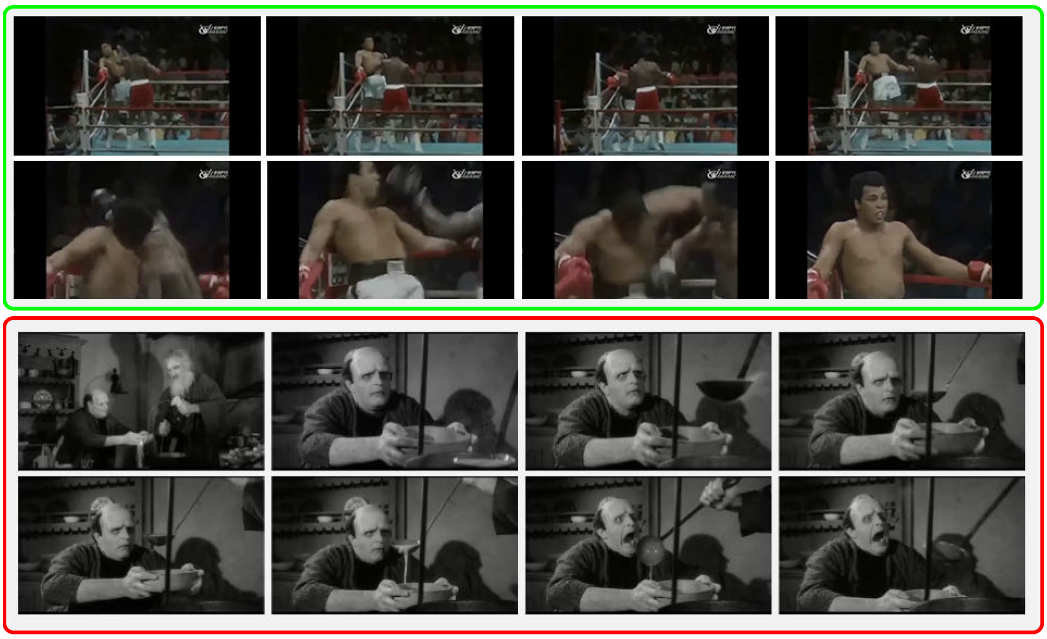

Fig. 1. The ability to estimate action progress is essential for interaction. In 1977 progress estimation allowed the boxer Muhammad Ali to dodge 21 punches in 10 s: https://youtu.be/nxZ-J7xit5Y. In the movie "Young Frankenstein" instead, the monster is not able to estimate progress and react accordingly: https://goo.gl/muYbCb.

action has advanced at an early stage based only on what it has seen so far. If an agent has to act to assist humans, then it cannot wait for the end of the action to perform the visual processing and act accordingly (Figure 1 shows an example sequence for this phenomena). Therefore, the ultimate goal of action understanding should be the development of an agent equipped with a fully functional perception action loop, from predicting an action before it happens to following its progress until it ends. This is supported also by experiments in psychology showing that humans continuously understand the actions of others to plan their goals [11]. Consequently, a model that is able to forecast action progress would enable new applications in robotics (e.g., human–robot interaction, realtime goal definition) and autonomous driving (e.g., avoid road accidents).

Broadly speaking, our work falls into the area of *predictive vision*, an emerging field that has been gaining much interest in recent years. Several approaches have been proposed to perform prediction of the near future, be that of a learned representation [29, 56], a video frame [38, 57], or directly the action that is going to happen [10, 31]. However, we believe that fully solving action understanding requires not only to predict the future outcome of an action but also to understand what has been observed so far in the progress of an action. As a result, in this article we introduce the novel task of predicting *action progress*, i.e., the prediction of how far an action has advanced during its execution. In other words, considering a partial observation of some human action, in addition to understanding *what* action and *where* it is happening, we want to infer *how long* this action has been executed for with respect to its duration. As a simple example of application, let us consider the use case of a social robot trained to interact with humans. The correct behaviour to respond to a handshake would be to anticipate, with the right timing, the arm motion, so as to avoid a socially awkward moment in which the person is left hanging. This kind of task cannot be solved unless the progress of the action is estimated accurately.

Some closely related problems have been recently addressed by the computer vision community. First, predicting action progress is conceptually different from action recognition and detection [14, 61, 62, 65], where the focus is on finding *where* the action occurred in time and space. Action completion [18–20, 60, 64, 65] is indeed a related task, where the goal is to predict when an action can be classified as complete to improve temporal boundaries and the classifier accuracy on incomplete sequences. However, this is easier than predicting action progress, because it does not require us to predict the partial progress of an action. Action progress prediction is an extremely challenging task, since, to be of maximum utility, the prediction should be made online while observing the video. While a thick crop of literature addresses action detection and spatio-temporal localization [12, 13, 23, 26, 35, 44, 59, 62, 66], predicting action progress is more closely related to online action detection [6, 22, 28, 50, 61]. Here the goal is to accurately detect, as soon as possible, when an action has started and when it has finished, but they do not have a model to estimate the progress. Some similarities are shared with the task of activity recognition [30, 54] where the goal is to detect which high-level phase is currently in progress in a long procedure composed by multiple actions. Differently, action progress focus on such actions that can be made of one or many movements but can be detected by visual means only. That is harder than activity recognition where the phases represent states of a process and can also be inferred by using a knowledge base or by the objects that are in use.

Not every action has a progress to be estimated. For instance, there are actions that are instantaneous (such as hitting a ball) or that do not have a clear goal given the available information (such as walking without a precisely defined destination). Thus, a classification of addressable actions is needed to successfully model action progress. Unfortunately, such a formal classification is a still an open research subject, but there is a building evidence in neuroscience that language is strongly linked to actions and is correlated to similar grammars [3, 42, 53]. As a result, in this work we take inspiration from the linguistics literature, which extensively discuss the topic of actions and the category of verbs used to refer to them. We propose a unified view of the problem discussing how progress can be estimated according to the corresponding verb classification.

In summary, in this article we propose ProgressNet, a model for action progress prediction of multiple actors using a supervised recurrent neural network fed with convolutional features. The main contributions of this work are the following:

- We define the new task of action progress prediction, which we believe is a fundamental problem in developing intelligent planning agents. We take inspiration from linguistics literature to classify which actions are suitable for the task. We also design and present an experimental protocol to assess performance.
- Given that actions are often composed of sub-action phases, we propose two formal models of action progress. A simple model that considers actions in their entirety and one that introduces a sequence of sub-phases.
- We present a holistic approach capable of predicting action progress while performing spatio-temporal action detection. ProgressNet can be naturally fitted into any online action detector model. To encourage precise prediction of progress, we also contribute a novel Boundary Observant loss that penalizes prediction errors on temporal boundaries.

2 RELATED WORK

Human action understanding has been traditionally framed as a classification task [1, 45]. However, in recent years, several works have emerged aiming at a more precise semantic annotation of videos, namely action localization, completion, and prediction.

Frame-level action localization has been tackled extending state-of-the art object detection approaches [46] to the spatio-temporal domain. A common strategy is to start from object proposals and then perform object detection over RGB and optical flow features using convolutional neural networks [14, 44, 47]. Gkioxari et al. generate action proposals by filtering Selective Search boxes with motion saliency and fuse motion and temporal decision using an Support Vector Machine (SVM) [14]. More recent approaches have devised end-to-end tunable architectures integrating region proposal networks in their model [4, 13, 23, 26, 35, 44, 47, 48, 50, 65]. As discussed in Reference [47], most action detection works do not deal with untrimmed sequences and do not generate action tubes. To overcome this limitation, Saha et al. [47] propose an energy maximization algorithm to link detections obtained with their framewise detection pipeline. Another way of exploiting the temporal constraint is to address action detection in videos as a tracking problem, learning action trackers from data [59]. To allow online action detection, Singh et al. [50] adapted the Single Shot Multibox Detector [33] to regress and classify action detection boxes in each frame. Then, tubes are constructed in real time via an incremental greedy matching algorithm. Considering that actions may have different temporal scales, in Reference [35], the authors go beyond predetermined temporal scales and propose to use Gaussian kernels to dynamically optimize them.

Approaches concentrating in providing starting and ending timestamps of actions have been proposed [8, 12, 21, 32, 41, 48, 62]. Heilbron et al. [21] have recently proposed a very fast approach to generate temporal action proposals based on sparse dictionary learning. Yeung et al. [62] looked at the problem of temporal action detection as joint action prediction and iterative boundary refinement by training a RNN agent with reinforcement learning. In Shou et al. [48] a three-dimensional (3D) convolutional neural network is stacked with Convolutional-De-Convolutional filters to abstract semantics and predict actions at the frame-level granularity. They report an improved performance in action detection frame by frame, allowing a more precise localization of temporal boundaries. More recently, this line of research evolved into the stricter problem of locating an action having only a video-level label available [8, 32, 41, 65]. Zhao et al. [65] introduced an explicit modelization of starting, intermediate, and ending phases via structured temporal pyramid pooling for action localization. They show that this assumption helps to infer the completeness of the proposals. In [32, 41], the action-context separation is explicitly considered. The idea is that parts of the video where an action has not taken place can be used as context of the instance, modeled with an end-to-end [32] or probabilistic [41] approach, to better separate the temporal edges. Escorcia et al. [8] specialize on localizing human actions by considering actor proposals derived from a detector for human and non-human actors intended for images. They use an actor-based attention mechanism, which is end-to-end trainable. A distinct line of research considers explicitly the temporal dimension [23, 26] either using 3D ConvNets on tubes [23] or exploiting multiple frames to generate a tube [26]. However, all these methods do not understand and predict the progress of actions, but only the starting and ending points. Differently from them, we explicitly model and predict action progress during the entire action and not only the starting and ending boundaries.

Orthogonal approaches to ours that implicitly model action progress cues can be found in action anticipation [2, 6, 22, 36] and completion [18–20, 52, 60, 64, 65]. The former aims at predicting actions before they start or as soon as possible after their beginning, and the latter instead binarizes the problem by predicting if an ongoing action is finished or not. Early event detection was first proposed by Hoai et al. [22]. In Reference [22], a method based on Structured Output SVM is proposed to perform early detection of video events. To this end, they introduce a score function of class confidences that has a higher value on partially observed actions. In Reference [28], the same task is tackled using a very fast deep network. Interestingly, the authors noted that the predictability of an action can be very different, from instantly to lately predictable. In Reference [36], a Long Short Term Memory (LSTM) is used to obtain a temporally increasing confidence to discriminate between similar actions in early stages, while Aliakbarian et al. [2] combine action and context-aware features to make predictions when a very small percentage of the action is observed. In Reference [18], RGB-D data are used to discriminate between complete and incomplete actions and [52] try to detect missing sub-activities to timely remind them to the user. More recently, Heidarivincheh et al. [19] presented an LSTM-based architecture that is able to predict the frame's relative position of the completion moment by either classification or regression. This problem was also further addressed by the same authors in a weakly supervised setting [20].

The growing area of predictive vision [34, 38, 56–58] is also related to action progress prediction. Given an observed video, the goal is to obtain some kind of prediction of its near future. Vondrick et al. [56] predict a learned representation and a semantic interpretation, while subsequent works predict the entire video frame [38, 57]. All these tasks are complementary to predicting action progress, since, instead of analyzing the progress of an action, they focus on predicting the aftermath of an action based on some preliminary observations. A few recent works have addressed tasks that are very close to action progress prediction. Neumann and Zisserman [40] framed the problem of future event prediction by defining a number of representations and loss functions to detect if and when a specific event will occur. To this end, they attempt to predict and regress the time to event probability, which is highly related to predicting action progress. In Reference [16], a residual action recognition model is used to estimate the progress of human activities to implicitly learn a temporal task grammar with respect to which activities can be localized and predicted.

Finally, a few preliminary attempts to estimate activity progress have been presented in the medical domain [43, 54] aiming to predict remaining surgery durations or anatomical motions. Nonetheless, these approaches are limited to predicting which action is in progress while doing a durative activity made by a sequence of actions, without explicitly predicting how much each action is completed.

3 ACTION PROGRESS

In addition to categorizing actions (*action classification*), identifying their boundaries spanning through the video (*action detection*), and localizing the area where they are taking place within the frames (*action localization*), our goal is to learn to predict the progress of an ongoing action. We refer to this task as *action progress prediction*.

Providing a definition of action progress is not trivial. In the following, we analyze the problem thoroughly and propose two definitions to model progression: (i) a linear interpretation that is versatile and can be applied to any sequence annotated for action detection and (ii) a phase-based interpretation where actions are precisely split into sub-events and manually annotated to obtain a richer representation that captures non-linear dynamics.

To obtain a clear and well reasoned taxonomy of actions and a comprehensive representation of action progress, we borrow a few concepts from verb and action classifications in linguistics. These classifications are helpful to understand for which actions progress may be defined and thus to derive appropriate models. According to Reference [55], verbs can be classified into four categories: those that express *activity*, *accomplishment*, *achievement*, and *state*. For our goal, we are not interested in stative verbs, which are used to describe the truth of a certain property (e.g., "someone knows something" describes the state of knowledge and "to know" is considered a stative verb), and we are instead interested in the remaining categories, since they all describe actions. About actions, in Reference [5] a distinction is made between *punctual* and *durative* actions and whether actions have a defined goal (referred to as *telic* and *atelic*, respectively) (see Table 1). A logic test to identify telic from atelic actions is the following: *SUBJECT was VERB-ing until something happened and he/she stopped VERB-ing. Did he/she VERB?*

Table 1. Comrie's Categories of Verbs [5]

	Punctual	Durative
Telic	Achievement (*to release*)	Accomplishment (*to drown*)
Atelic	Semelfactive (*to knock*)	Activity (*to walk*)
Changeless	—	State (*to know*)

Example:

- Mark was *walking* until Bill made him fall. Did Mark *walk*? Yes → **Atelic**
- Susan was *shooting* a basketball until Anne made her fall. Did Susan *shoot* the ball? No → **Telic**

Progress can be defined clearly when the action implies a change of state on the actor. Therefore, durative telic actions are the most appropriate for our task, since they have a duration in time and a goal that clearly implies a change of state. Punctual actions, being telic or atelic, do not progress, since they do not have a clear time extent, i.e., they happen instantaneously.

Durative actions can possibly be decomposed in phases. Phases might have a different classification from the whole action. There are certain activities that are atelic if looked as a whole but can be decomposed in telic phases. Walking, for instance, can be reformulated as a sequence of movements of legs and feet: rise right foot, move leg, land right foot, rise left foot, and so on. All these sub-actions are telic-durative while the whole action of walking is atelic-durative. Similarly, telic actions might contain phases with an atelic nature. For instance, the execution of the telic action Pole Vault always has the same structure, made of telic and atelic phases: running (*atelic - durative*), sticking the pole in the ground (*telic - punctual*), jumping over the bar (*telic - durative*), touching ground (*telic - punctual*), and standing up (*telic - durative*).

For the scope of our research, there are two main problems that make it hard to develop a model of action progress. The first is the intrinsic difficulty of annotating temporal boundaries for any type of action [49]. The second is the difficulty of splitting durative actions into phases and collecting prior information regarding their structure or subparts. From a computer vision point of view, it is hard to correctly annotate every single step of a walking action, and it can be considered even harder to train a classifier to effectively recognize all single short spanned events of such kind.

Based on these considerations we have developed two distinct models of progression that are suited for distinct cases: (i) a linear interpretation that applies to actions regarded as a whole and (ii) a phase-based interpretation where actions are precisely split into sub-events and manually annotated to obtain a richer representation that captures non-linear dynamics. Annotating actions with linear progress is trivial yet might not yield semantically accurate labels, whereas phase-based progress will provide a fine modeling of the evolution of an action although requiring an extensive annotation cost. An in depth discussion is carried out in Sections 3.1 and 3.2.

Both interpretations are built upon the concept of *action tube* [14, 47], i.e., a sequence of bounding boxes spanning from a starting frame f_S to an ending frame f_E and enclosing the subject performing the action. It is important to tie the definition of progress with a specific action tube. Bounding boxes in fact are needed, because multiple actors may be present in the frame at the same time. Whereas they might be doing the same action, they are likely to exhibit a different action progress. Therefore, we use bounding boxes to be able to provide predictions for each actor independently, which will be impossible just focusing on the temporal aspect of the action. We predict the progress values online, frame by frame, by only observing the past frames of the tubes from t_S to the current frame t_i.

3.1 Linear Progress

We provide a first definition of action progress with a linear interpretation. Given a frame f_i belonging to an action tube in $[t_S, t_E]$, action progress can be interpreted as the fraction of the action that has already passed. Therefore, for each box in a tube at time t_i, we define the target action progress as:

$$p_i = \frac{t_i - t_S}{t_E - t_S} \in [0, 1]. \tag{1}$$

This definition is versatile and can be applied to any sequence annotated for action detection. A similar linear modeling of action evolution has been previously used in action anticipation [2] and in action synchronization [7] with encouraging results. Although simple, a key advantage of this definition is that it does not require us to define any prior information regarding the structure or subparts of an action, making it applicable to a large set of actions. As a result, we can learn predictive models from any dataset containing spatio-temporal boundary annotations. This means that the task of action progress in its linear interpretation does not require us to collect additional annotations for existing datasets, since the action progress values can be directly inferred from the temporal annotations. This is a major strength, also considering the intrinsic difficulty of annotating action temporal boundaries [49]. While this model is best suited for durative telic actions, in the experimental section we show that it is also reasonable in many cases, including durative atelic actions.

3.2 Phase-based Progress

Despite the simplicity and effectiveness of adopting a linear interpretation of progress, certain actions may indeed exhibit dilations or contractions in the execution rate, yielding to a nonlinear progression. This behavior is observable when the action can be broken down into a sequence of sub-phases that defines its structure.

Additionally, in certain cases the structure is clear and sequential while in other cases it is not. Usually telic actions exhibit this structure. However, atelic actions, such as *Dancing*, may be defined by a random sequence of moves with no identifiable structure, which can even follow cyclic or erratic patterns.

The phase-based progress formulation provides a better definition of action progress. In this approach, we first split a durative action into phases, manually annotating their temporal boundaries. We note that each phase, let it be telic or atelic, is always delimited by a punctual action that denotes a transition from one phase to another. Therefore, we represent durative actions as a sequence of telic or atelic phases, separated by punctual actions. An important advantage of exploiting punctual actions is that they are unambiguous and easy to annotate, since they define a precise instant in time in the development of the action (a single frame) and there is no need to specify their telicity or atelicity. This yields also to a precise annotation of durative phases, since punctual ones act as temporal boundaries. Figure 4 helps to understand these concepts by showing frames of the action class PoleVault: punctual actions are clearly identifiable as single frames and therefore can be manually labeled without uncertainty by an annotator. It is interesting to notice that, despite intra-class variation, punctual phases all carry the same meaning and have common visual traits (e.g., the pole to the ground or the athlete over the bar).

After identifying phases, we need to assign a progress value to each frame in an action tube. We base our phase-based progress formulation on the fact that each instance of a punctual phase should have the same progress value. Since punctual phases act as boundaries for durative actions, which might exhibit high variability in duration and execution speed, this makes the overall resulting progress non linear.

To establish appropriate ground-truth progress values, we first build a prototype action for each class, where the length of each durative phase is averaged across all instances in the training set. A generic instance of an action can be interpreted as a deformation of the prototype, where durative phases are compressed or dilated (i.e., slowed down or accelerated) with respect to their expected duration. According to this, we label the action prototype with the linear interpretation of progress (Equation (1)), thus defining ground-truth progress for punctual phases, since these are shared across instances. Once punctual phases have been labeled, we can label durative phases knowing that their values must span over a well-defined interval $[p_S, p_E]$ given by the progress of the two punctual boundaries.

To assign a progress value p_i to frames belonging to durative phases, we adopt the following criteria, depending if the phase is telic or atelic:

- $p_i = \frac{i}{N_i} + p_S$ for telic phases
- $p_i = \frac{(p_S + p_E)}{2}$ for atelic phases

where i is the index of the frame within the current phase and N_i the prototype duration of the current phase. This corresponds to a linear progress for telic phases, which are usually executed uniformly, since they represent events with a simple structure (rising hands, a jump, or a kick of a ball). However, atelic phases can exhibit more complex behaviors with lack of structure. Since it is hard to establish a clear progression of states in atelic phases, we simply assign the expectation of progress inside the interval, based on the value of its boundaries.

It has to be noted that whereas the progress of prototype phases is defined linearly, the resulting progress values for real action tubes are in fact non linear (piecewise linear). Aligning the progress of each action tube to have the same value in correspondence of punctual phases is a form of action synchronization, where the execution rate of each phase is normalized to have the same speed. Executions of an action with a slow phase will therefore exhibit a slower growth of progress compared to other executions where the same phase happens quickly.

This definition of progress is more expressive than the linear interpretation, allowing us to model more complex dynamics. To better understand the advantage of adopting the phase-based progress, in Figure 3 a comparison is shown between the two types of annotations. When there is variability in the execution of a phase, progress values in the linear interpretation may become unaligned, yielding to an imprecise representation.

At the same time, the phase-based progress requires a demanding annotation process, since each action has to be divided into phases and all the boundaries need to be manually annotated. In Section 5.1 the annotation procedure for the UCF-101 dataset is detailed. The collected annotations will be released upon publication.

4 PROGRESSNET

The whole architecture of our method is shown in Figure 2, highlighting the first branch, which is dedicated to action classification and localization, and the second branch, which predicts action progress. We believe that sequence modelling can have a huge impact on solving the task at hand, since time is a signal that carries a highly informative content. Therefore, we treat videos as ordered sequences and propose a temporal model that encodes the action progress with a Recurrent Neural Network. In particular, we use a model with two stacked Long Short-Term Memory layers (LSTM), with 64 and 32 hidden units, respectively, plus a final fully connected layer with a sigmoid activation to predict action progress. Since actions can be also seen as transformations on the environment [58], we feed the LSTMs with a feature representing regions and their context. We

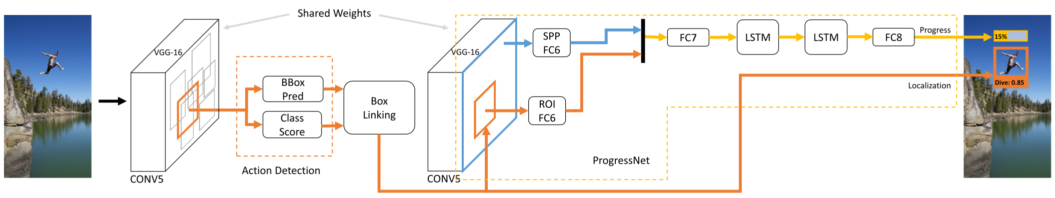

Fig. 2. Proposed Architecture. On the left (highlighted in orange), we show the classification and localization data flows for tube generation. On the right (highlighted in yellow), our ProgressNet. Region (ROI FC6) and Contextual features (SPP FC6) from the last convolutional map are concatenated and then fed to a Fully Connected layer (FC7). Two cascaded LSTMs perform action progress prediction.

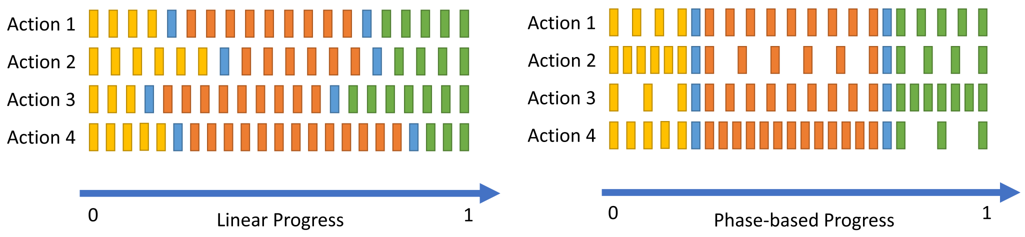

Fig. 3. Linear progress vs. Phase-based progress. Each rectangle represents a frame in an action tube. Different colors correspond to different phases. Blue rectangles represent punctual phases, which act as boundaries for durative phases. In the phase-based definition, progress is defined to align phase boundaries across different executions of the same action.

concatenate a contextual feature, computed by spatial pyramid pooling (SPP) of the whole frame [17], with a region feature extracted with ROI Pooling [46]. The two representations are blended with a fully connected layer (FC7). The usage of a SPP layer allows us to encode contextual information for arbitrarily-sized images. We named this model *ProgressNet*.

While the backbone of our model is inspired by Reference [47], our architecture has several novel design choices: the ability to process data including both the observed actor and context through RoI Pooling plus SPP; the combination of convolutional and recurrent layers to separate feature representation and temporal modeling; and a multi-task design capable of performing classification, localization, and progress estimation.

Our model emits a prediction $p_i \in [0, 1]$ at each time step i in an online fashion, with the LSTMs attention windows that keep track of the whole past history, i.e., from the beginning of the tube of interest until the current time step. We rely on an action detector [47] to obtain scored boxes at each frame. Such framewise action detectors are derived from object detector frameworks and fine-tuned on action bounding boxes. Features are extracted from the last convolutional feature map of such models by re-projecting each linked box onto it through ROI Pooling. Each tube is evaluated online independently in parallel.

We use ReLU non linearities after every fully connected layer and dropout to moderate overfitting. Our approach is *online* and does not require complete tubes to perform predictions at test time. The only requirement of ProgressNet is to obtain a sequence of linked bounding boxes forming the tube. Both online and offline solutions to this problem have been proposed [47, 50]. It has also to be noticed that ProgressNet adds a computational footprint of about 1 ms per frame on a TITAN XP Pascal GPU, making it feasible to work in real time.

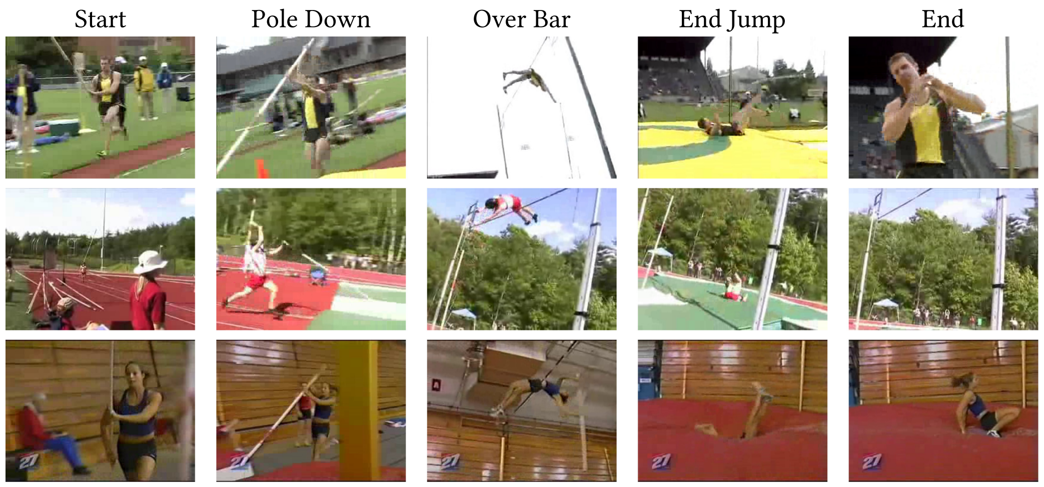

Fig. 4. Example of punctual phases for the PoleVault class. Five punctual phases can be found: the starting point of the action, the instant when the athlete puts the pole on the ground, the moment of maximum elevation during the jump, the end of the jump landing on the mattress, and the end point of the action. Start and ending points of the action correspond to the boundaries of the action tube. Intermediate punctual phases are clearly identifiable with a single video frame and therefore are easy to annotate. Punctual phases are useful, since they correspond to the boundaries of durative ones.

4.1 Learning

We initialize the spatio-temporal localization branch of our network, highlighted in orange in Figure 2, using a pre-trained action detector such as in References [47, 50] while the remaining layers are learned from scratch. To train our ProgressNet we use ground-truth action tubes as training samples.

To avoid overfitting the network, we apply the two following augmentation strategies. First, for every tube we randomly pick a starting point and a duration, so as to generate a shorter or equal tube (keeping the same ground-truth progress values in the chosen interval). For instance, if the picked starting point is at the middle of the video and the duration is half the video length, then the corresponding ground-truth progress targets would be the values in the interval $[0.5, 1.0]$. This also forces the model to not assume 0 as starting progress value. Second, for every tube we generate a subsampled version, reducing the frame rate by a random factor uniformly distributed in $[1, 10]$. This second strategy helps in generalizing with respect to the speed of execution of different instances of the same action class.

To encourage the network to be more precise on temporal boundaries of durative phases, we introduce a Boundary Observant (BO) loss. The idea is that phase boundaries (i.e., the punctual actions) do not have ambiguity and less error should be allowed on them. On the contrary, intermediate parts are less certain and a higher error can be tolerated. As an example, the frames shown in Figure 4 correspond to punctual phases and it is observable that their progress points can be roughly estimated even with single frames. What we want to enforce is a sufficiently low error on these instants that should help in modeling the evolution of what happens in between (i.e., the durative phases). We present our BO loss in a general form, considering actions composed by multiple phases. The linear interpretation is a simpler case of phase-based progress with a single durative phase, delimited by the start and the end of the whole action.

Given a prediction $\hat{p_i}$ and a target value p_i, for each phase k of an action spanning across an interval $[l_k, u_k]$, we define a potential $e_k(p_i, \hat{p_i})$ with unitary cost for predictions that exceed the boundaries, decreasing it toward zero as the target gets closer to the center of the interval $m_k = (l_k + u_k)/2$:

$$e_k(p_i, \hat{p_i}) = \min \left[1, \left(\frac{p_i - m_k}{r_k \sqrt{2}} \right)^2 + \left(\frac{\widehat{p_i} - m_k}{r_k \sqrt{2}} \right)^2 \right]. \tag{2}$$

The potential is derived from the circle equation $(x - x_c)^2 + (y - y_c)^2 = r^2$ of center (x_c, y_c) and radius r. Since we want $e_k = 1$ on the boundaries, we scale the circle to be circumscribed to the square of side $u_k - l_k$. In other words, representing these potentials on a 2D coordinate system with true and predicted progress values on the axes, we are identifying a circular region centered in (m_k, m_k) that spans over an entire phase and passes through the points l_k, l_k and (u_k, u_k). Since the phase is delimited by the interval $[l_k, u_k]$, the radius of the circle will be $r_k\sqrt{2}$, where $r_k = (u_k - l_k)/2$.

The potentials are computed for each phase and pooled together by a minimum operator. We want each phase to be defined only by its correspondent potential. This is possible with the minimum pooling, since $e_k < 1$ inside its circular region (the phase) and $e_k = 1$ outside and since phases cover non overlapping areas. The Boundary Observant loss is then obtained by weighing the error with an L1 loss and averaging across samples:

$$\mathcal{L}_{BO} = \frac{1}{N} \sum_{i}^{N} \min_{k} \{e_k\} \left| p_i - \widehat{p_i} \right|. \tag{3}$$

Compared to the $L2$ loss for regression, the BO loss penalizes errors on the boundaries more than in intermediate parts, since we want to precisely identify when the phase starts and ends. At the same time, it avoids the trivial solution of always predicting the intermediate value $\widehat{p} = 0.5$. Figure 5 shows the difference between the two loss functions, where predicted values are on the x-axis and targets on the y-axis. Whereas the L2 loss function is always the same, BO adapts to the structure of the action and its phases. Note from the definition of progress that only values in $[0, 1]$ can be expected.

We initialize all layers of ProgressNet with the Xavier [15] method and employ the Adam [27] optimizer with a learning rate of 10^{-4}. We use dropout with a probability of 0.5 on the fully connected layers.

5 EXPERIMENTAL SETTING

In this section, we discuss the experimental setting to evaluate action progress prediction for spatio-temporal tubes and propose two evaluation protocols. We introduce some simple baselines and show the benefits of our approach.

We experiment on the J-HMDB [25] and UCF-101 [51] datasets. J-HMDB consists of 21 action classes and 928 videos, annotated with body joints from which spatial boxes can be inferred. All the videos are temporally trimmed and contain only one action. Since clips are all very short, all actions can be considered to be telic. We use this dataset to benchmark action progress prediction with a linear interpretation. UCF-101 contains 24 classes annotated for spatio-temporal action localization. It is a more challenging dataset, because actions are temporally untrimmed and there can be more than one action of the same class per video. Moreover, it contains video sequences with large variation in appearance, scale, and illumination. We use this dataset to predict action progress both with the linear and the phase-based interpretations of progress, which required to manually identify and annotate all sub-phases of the actions in the dataset. Section 5.1 details the

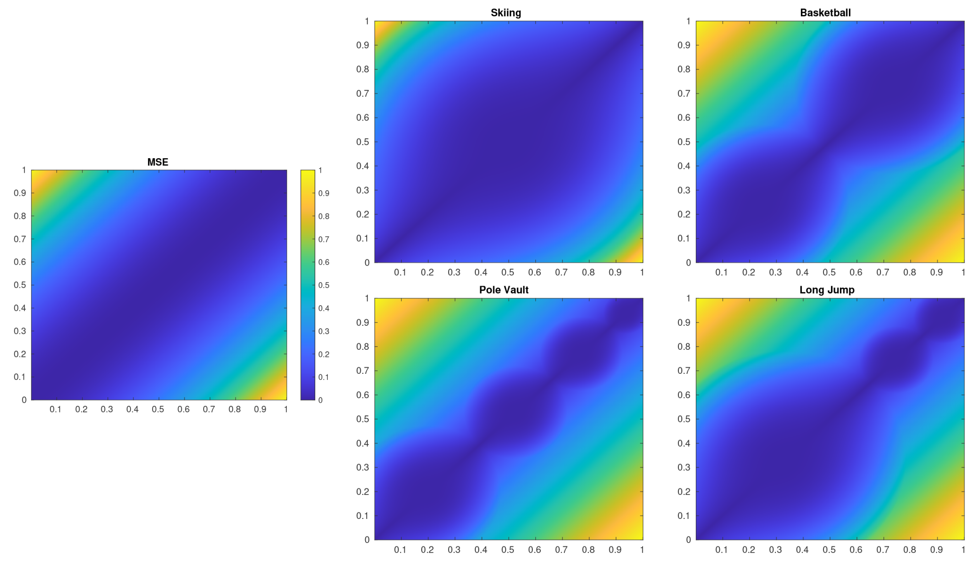

Fig. 5. Comparison between the $L2$ (left) and the Boundary Observant (right) loss functions. Predicted values and ground-truth targets are on the two axes. It can be seen that the Boundary Observant loss is stricter against errors on the action boundaries. The Boundary Observant loss depends on the structure of each action, penalizing errors close phase boundaries. Examples for a few action classes are shown.

annotation process and which phases have been used. We adopt the same split of UCF-101 used in References [44, 47, 59, 63].

Note that larger datasets such as THUMOS [24] and ActivityNet [9] do not provide bounding box annotations and therefore cannot be used in our setting.

5.1 Phase Annotation

We annotated phases for all actions in the UCF-101 dataset [51]. Punctual events are manually identified and used as boundaries for durative phases. We specify whether a durative phase is telic or atelic to assign the proper progress values as defined in Section 3.2.

We observe that the original annotations have imprecise temporal boundaries for our task. For instance, there is a different amount of waiting before a golf swing or a dive. We believe that such durative atelic phases are tied to the known difficulties [39] in clearly identifying the temporal boundaries of an action where *Pre-actional* phases may be present. Hence, by annotating punctual events, the annotator has unambiguous instructions about which frame to annotate and the uncertainty is reduced. We split every action in the dataset as in Table 2. To easily identify phases and facilitate the annotation process, we use verbs in their *-ing* form to denote durative phases and in the infinitive form for punctual ones.

The annotators were asked to identify punctual actions (i.e., select the corresponding frame) for each action tube, starting from the original spatio-temporal annotations.[1] Durative phases are automatically labeled, since they are defined by two punctual boundaries. Among all annotated actions, we have annotated a total of 27,120 phases, 15,789 of which are punctual and 11,331 durative. Punctual phases by definition have all length equal to 1. Instead, we found durative phases to have an average duration of 22 frames.

[1]We used the revised annotations available at https://github.com/gurkirt/corrected-UCF101-Annots.

Table 2. Action Phases Annotated for the UCF-101 Dataset

Action	Punctual	Durative	Punctual	Durative	Punctual	Durative	Punctual	Durative	Punctual	Durative	Punctual
Basketball	Start	Rising ball	Shoot ball	Lowering hands	End						
BasketballDunk	Start	Jumping	Dunk	Landing	End						
Biking	Start	Cycling	End								
CliffDiving	Start	Waiting	Start jump	Diving	Land in water	Swimming	End				
CricketBowling	Start	Running	Start charge	Charging shot	Shoot ball	Stop running	End				
Diving	Start	Approaching	Start jump	Jumping	Dive	Somersaulting	Stretch body	Falling	Land in water	Swimming	End
Fencing	Start	Fencing	End								
FloorGymnastics	Start	Standing	Start running	Running	Jump	Somersaulting	Land	Rising arms	Arms up	Standing	End
GolfSwing	Start	Standing	Start charge	Charging	Max charge	Shooting	Hit ball	Rising club	Top elevation	Waiting	End
HorseRiding	Start	Riding	End								
IceDancing	Start	Dancing	End								
LongJump	Start	Running	Jump	Jumping	Land	Standing up	End				
PoleVault	Start	Running	Pole down	Jumping	Over bar	Falling	End jump	Standing up	End		
RopeClimbing	Start	Climbing	End								
SalsaSpin	Start	Dancing	End								
SkateBoarding	Start	Skateboarding	End								
Skiing	Start	Skiing	End								
Skijet	Start	Skijet	End								
SoccerJuggling	Start	Juggling	End								
Surfing	Start	Surfing	End								
TennisSwing	Start	Opening	Open	Hitting ball	Hit ball	Closing	Closed	Moving	End		
TrampolineJumping	Start	Jumping	End								
VolleyballSpiking	Start	Running	Start jump	Jumping	Hit ball	Landing	Land	Moving	End		
WalkingWithDog	Start	Walking	End								

Punctual phases are denoted in blue, telic in green, and atelic in orange.

5.2 Metrics

To evaluate the task of action progress prediction, we introduce two evaluation metrics.

Framewise Mean Squared Error. This metric tells how well the model behaves at predicting action progress when the spatio-temporal coordinates of the actions are known. Test data are evaluated frame by frame by taking the predictions $\widehat{p}_i$ on the ground-truth boxes B_i and comparing them with action progress targets p_i. We compute mean squared error $MSE = ||\widehat{p}_i - p_i||^2$ across each class. Being computed on ground-truth boxes, this metric assumes perfect detections and thus disregards the action detection task, only evaluating how well progress prediction works.

Average Progress Precision. Average Progress Precision (APP) is identical to framewise Average Precision (Frame-AP) [14] with the difference that true positives must have a progress that lays within a margin from the ground-truth target. Frame-AP measures the area under the precision-recall curve for the detections in each frame. A detection is considered a hit if its Intersection over Union (IoU) with the ground truth is bigger than a threshold τ and the class label is correct. In our case, we fix $\tau = 0.5$ and evaluate the results at different progress margins m in $[0, 1]$. A predicted bounding box $\widehat{B}_i$ is matched with a ground-truth box B_i and considered a true positive when B_i has not been already matched and the following conditions are met:

$$IoU(\widehat{B}_i, B_i) \geq \tau, \quad |\widehat{p}_i - p_i| \leq m, \tag{4}$$

where $\widehat{p_i}$ is the predicted progress, p_i is the ground-truth progress, and m is the progress margin. We compute Average Progress Precision for each class and report a mean (mAPP) value for a set of m values.

5.3 Implementation Details

In practice, we observed that on some classes of the UCF-101 dataset it is hard to learn accurate progress prediction models. These are action classes like *Biking* or *WalkingWithDog* that are completely atelic and even for a human observer is hard to guess how far the action has progressed. Therefore, we extended our framework by adopting a curriculum learning strategy. First, the model is trained as described in Section 4.1 on classes that have at least a telic phase.[2] Then, we fix all convolutional, FC and LSTM layers and fine-tune the FC8 layer that is used to perform progress prediction from the last LSTM output on the whole UCF-101 dataset. This strategy improves the convergence of the model.

6 EXPERIMENTS

In this section, we report the experimental results on the task of predicting action progress. We first present an analysis of ProgressNet using a linear progress formulation and then using phase-based progress. Since ProgressNet is a multitask approach, we start by measuring progress estimation on perfectly localized actions, i.e., discarding possible action localization errors. In addition, we perform several ablation studies, underlining the importance of the Boundary Observant loss and the behavior of our method on partially observed action tubes. We then test the method on real detected tubes with the full model and, finally, report a qualitative analysis that shows some success and failure cases.

6.1 Linear Progress

Action progress on correctly localized actions. In this first experiment, we evaluate the ability of our method to predict action progress on correctly localized actions in both time and space. We take the ground-truth tubes of actions on the test set and compare the MSE of three variants of our method: the full architecture trained with our Boundary Observant loss (*ProgressNet*), the same model trained with L2 loss (*ProgressNet L2*), and a reduced memoryless variant (*ProgressNet Static*).

The comparison of our full model against ProgressNet L2 is useful to understand the contribution of the Boundary Observant loss with respect to a simpler L2 loss. To underline the importance of using recurrent networks in action progress prediction, in the variant ProgressNet Static we substitute the two LSTMs with two fully connected layers that predict progress framewise.

In Table 3, we report the MSE results for a linear progress on both the J-HMDB and UCF-101 datasets. In addition to the variants of our models, we provide two baselines: random prediction and constant prediction. The random prediction provides us with a higher bound on the MSE values. For the constant prediction, we always predict the progress expectation $\widehat{p} = 0.5$ for every frame, which is a trivial solution that obtains good MSE results. Both are clearly far from being informative for the task.

We first observe that the MSE values are consistent among the two datasets, with ProgressNet models ahead of the other methods. ProgressNet and ProgressNet L2 obtain a much lower error than ProgressNet Static and the baselines. This confirms the ability of our model to understand action progress. In particular, the best result is obtained with ProgressNet, proving that our

[2]This subset consists of the following classes: *Basketball*, *BasketballDunk*, *CliffDiving*, *CricketBowling*, *Diving*, *FloorGymnastics*, *GolfSwing*, *LongJump*, *PoleVault*, *TennisSwing*, *VolleyballSpiking*.

Table 3. Mean Square Error Values for Action Progress Prediction on the UCF-101 and J-HMDB Datasets

	J-HMDB	UCF-101
Random	0.166	0.166
0.5	0.084	0.083
ProgressNet Static	0.079	0.104
ProgressNet L2	0.032	0.052
ProgressNet	**0.026**	**0.049**

Results are averaged among all classes.

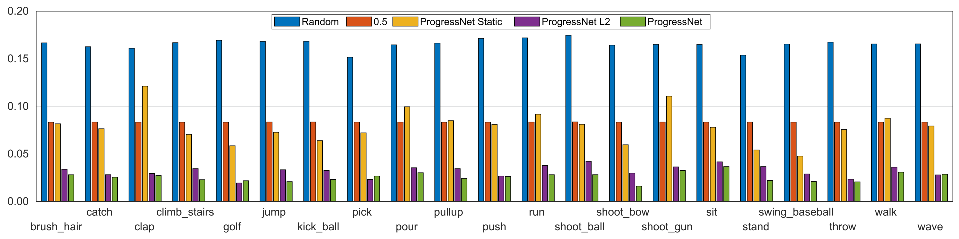

Fig. 6. Mean Squared Error obtained by three formulations of our model (ProgressNet Static, ProgressNet L2, and ProgressNet) on the J-HMDB dataset. Random and constant 0.5 predictions are reported as a reference.

Boundary Observant loss plays an important role in training the network effectively. ProgressNet Static has an inferior MSE than the variants with memory, suggesting that single frames for some classes can be ambiguous and a temporal context can help to accurately predict action progress. In particular, observing the class breakdown for J-HMDB in Figure 6, we note that the static model gives better MSE values for some actions such as *Swing Baseball* and *Stand*. This is due to the fact that such actions have clearly identifiable states, which help to recognize the development of the action. However, classes such as *Clap* and *Shoot Gun* are hardly addressed with models without memory, because they exhibit only few key poses that can reliably establish the progress.

In Figure 7, we show t-SNE [37] embeddings of the hidden states of the second LSTM layer. Each point represents a frame of the J-HMDB test set, and its projected hidden state has accumulated knowledge over all previous frames in the action tube. Color coding in Figure 7 reflects the ground-truth progress of the samples; therefore a desirable t-SNE projection should highlight the presence of a structure where data are distributed according to color (i.e., progress). We report both the embeddings for the whole test set and for four classes projected independently. In all figures, we note that points are organized in the t-SNE projection space such that progress increases radially from points labeled with 0.1 toward points labeled with 1.0. This hints at the fact that ProgressNet has learned directions in the LSTM hidden state space that follow action progress, which allows the final regressor to estimate an accurate value. Looking at the classwise projections, it can be observed that several directions are discovered, taking into account intra-class variations.

The t-SNE projections suggest that the architecture is modeling action progress as a distance from a starting point in the latent space along these directions. Since the observed directions in certain cases vary a lot, this might look difficult to achieve with a single linear layer. Despite this, it must be noted that t-SNEs are a tool for qualitative inspection used to highlight patterns. Samples are in fact projected from a much higher dimensional space and therefore the plot might not show the full non-linearity of the hidden space, which may sometimes hide linear structures. Furthermore, the whole network is trained as a multitask framework, predicting class and progress

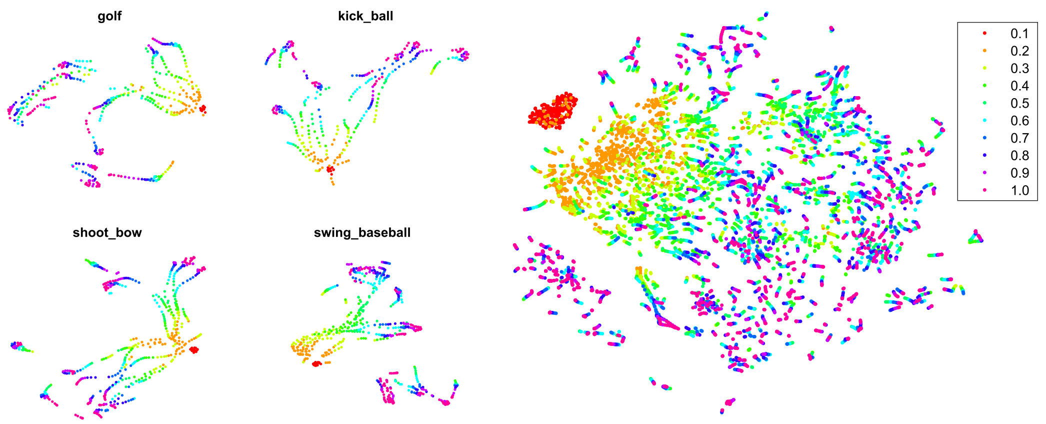

Fig. 7. t-SNE visualizations of ProgressNet's hidden states of the second LSTM layer on the J-HMDB test set. Each point corresponds to a frame colored with its ground-truth progress, quantized with a 0.1 granularity. On the left, the states of four classes are shown separately, while in the right all the test frames are shown together.

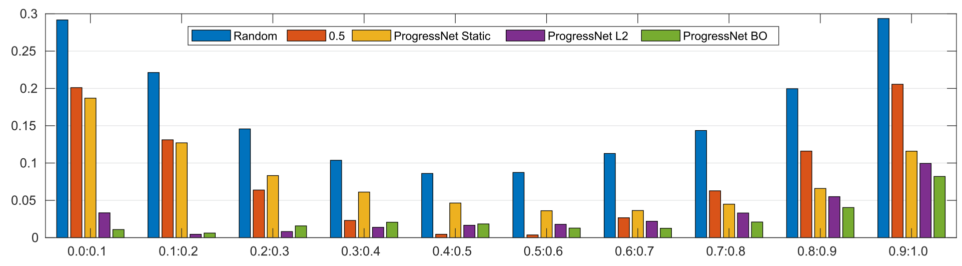

Fig. 8. MSE breakdown grouped by progress intervals for the J-HMDB dataset. Similar results are obtained with UCF-101.

at the same time, so the features before the LSTMs encode the class of the action. Since data are projected from a 64-dimensional to a 2-dimensional space, all this information is necessarily mixed and what could be represented linearly in the original feature space might not appear so in a projection trained to reflect geometric properties such as distance between points.

Error at different progress points. To understand when our model is more prone to errors, with respect to the action progression, in Figure 8 we show a breakdown of the error by dividing the MSE obtained in the previous experiment according to the ground-truth progress. It can be seen how action progress is harder at the boundaries and how our Boundary Observant loss helps in mitigating this difficulty.

Expected length and partially observed tubes. Despite the simple progress model, our approach is not just predicting incrementing linear values over an expected tube length. To show that we are able to understand the action progress even on truncated or generally incomplete tubes, we test an additional baseline where progress is estimated as the ratio between the frame ID in the tube and the expected length of the predicted class (estimated on the training videos).

This baseline obtains on UCF-101 an MSE of 0.112, which is only lower than Random (0.166) and is largely outperformed by our method. Moreover, we show how it suffers on partially observed tubes. In Figure 9, we report MSE values for ProgressNet and the expected length baseline varying

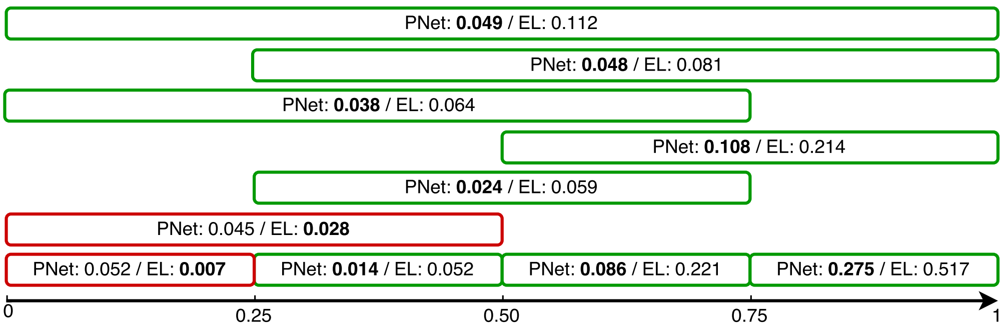

Fig. 9. MSE on partially observed tubes of UCF-101. Each block depicts an experimental setting with a different tube fraction, picking the beginning and the end in [0, 0.25, 0.5, 0.75, 1.0]. The first value is the MSE of ProgressNet (PNet) while the second one of the expected length baseline (EL). ProgressNet wins in 8 of 10 cases.

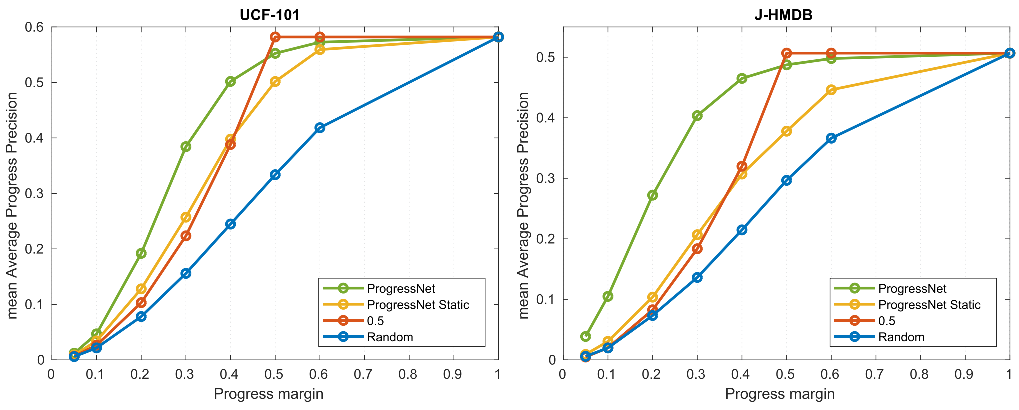

Fig. 10. mAPP on the UCF-101 and J-HMDB datasets.

the observation window of the tubes. ProgressNet largely outperforms the baseline except when observing a small portion of the tube at the beginning of the action.

Action progress with the full pipeline. In this experiment, we evaluate the performance of action progress while also performing the spatio-temporal action detection with the entire pipeline. Differently from the previous experiment, we test the full approach where action tubes are generated by the detector. In Figure 10, we report the mAPP of ProgressNet (trained with BO loss), ProgressNet Static, and the two baselines Random and 0.5 on both the UCF-101 and J-HMDB benchmarks. Note that the mAPP upper bound is given by standard mean Frame-AP [14], which is equal to mAPP with margin $m = 1$. In the $\widehat{p} = 0.5$ baseline, this upper bound is reached with $m = 0.5$.

It can be seen that ProgressNet has mAPP higher than the baselines for stricter progress margin. This confirms that our approach is able to predict action progress correctly even when tubes are noisy such as those generated by a detector. ProgressNet Static exhibits a lower performance than ProgressNet, confirming again that the memory is helpful to model action progress.

Architecture of the Network. We trained ProgressNet with just the first LSTM layer and report an increase of MSE to 0.081 on the UCF-101 dataset. This performs on par with the 0.5 baseline (0.083) and providing good results only on actions with simple dynamics (i.e., *Diving*, *GolfSwing*).

Table 4. Mean Squared Error Values on UCF-101 with Phase-based Progress

	All	Telic	Atelic
ProgressNet Static	0.142	0.197	0.132
ProgressNet L2	0.024	0.045	**0.008**
ProgressNet	**0.021**	**0.037**	0.010

Results are shown averaged over all classes as well as considering only telic actions or atelic actions.

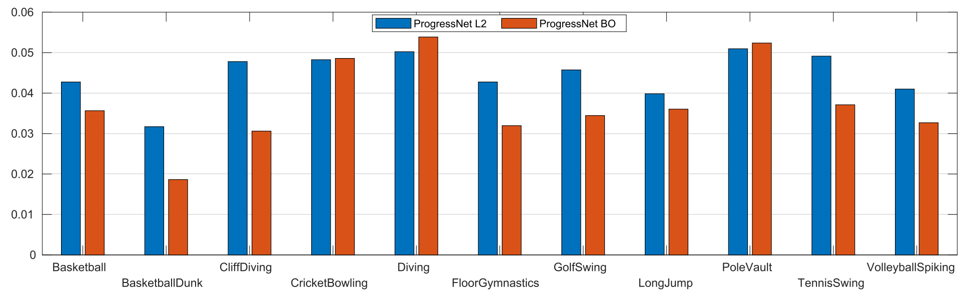

Fig. 11. MSE values with phase-based progress for telic actions, comparing ProgressNet trained with the L2 and BO losses.

6.2 Phase-based Progress

We trained ProgressNet also using phase-based annotations on UCF-101. Labeling actions with a phase based progress allows us to provide a better characterization of the evolution of the action.

Action progress on correctly localized actions. In Table 4, we report the MSE obtained by the three variants of our method. Again, ProgressNet Static obtains a much higher error than the versions equipped with LSTMs, and using the Boundary Observant loss slightly improves the capability of the network over an L2 loss. Interestingly, the biggest gain is observed on telic actions (i.e., actions with at least a telic phase), whereas atelic action does not seem to be affected by the BO loss. This is due to the fact that purely atelic actions do not have well-defined boundaries. In fact in the case of UCF-101 the boundaries of atelic actions often correspond with starting and ending frames of the videos. To better underline the importance of the Boundary Observant loss for telic actions, a classwise comparison of ProgressNet with L2 and BO losses is shown in Figure 11. It can be seen that most actions improve considerably when the model is forced to perform well on boundaries.

Since the Boundary Observant loss forces training to penalize errors more on phase boundaries, we investigate the behavior of the model on punctual phases. Since such phases are always defined within a single frame, we measure the average number of frames Δ_f between the frame f_k representing the punctual phase and the frame with the closest predicted progress $\hat{p}$:

$$\Delta_f = \frac{1}{K}\sum_{k=1}^{K} |f_k - \arg\min_i (p_k - \hat{p}_i)|. \tag{5}$$

This metric highlights the temporal offset between the punctual phase and when its progress is estimated by the model. Also in this case, as can be observed in Figure 12, a comparison with the L2 loss highlights the importance of the Boundary Observant loss. The model largely benefits from

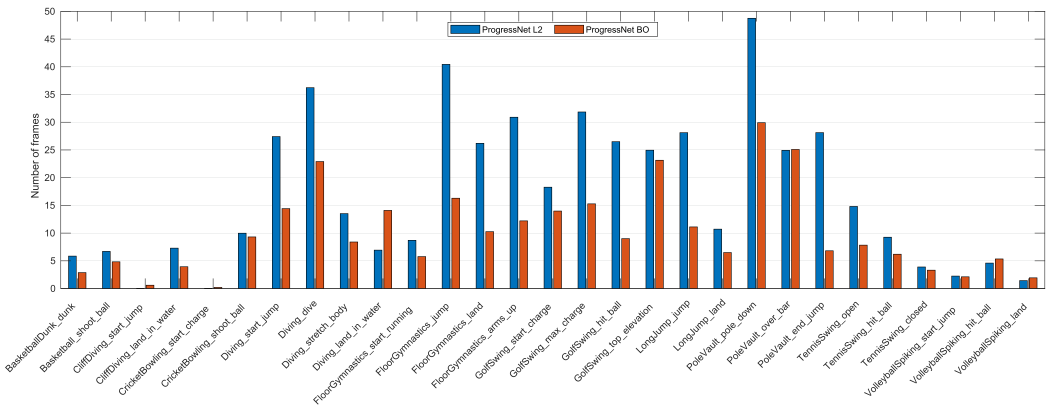

Fig. 12. Temporal offset between real and predicted progress for punctual phases, measured in number of frames.

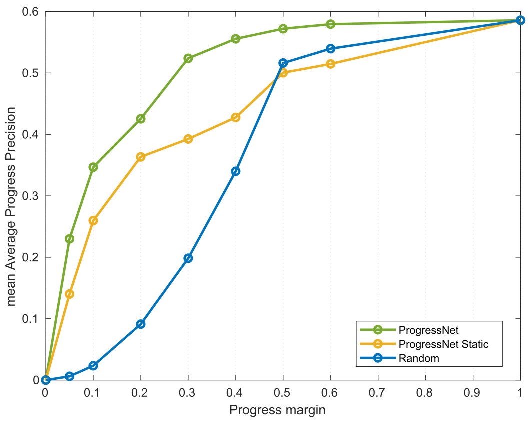

Fig. 13. mAPP on the UCF-101 using phase-based annotations.

the usage of the loss, with the temporal lag of the predictions Δ_f being reduced to under a second (25 frames) in almost all cases.

Action progress with the full pipeline. When testing ProgressNet with the full pipeline, i.e., evaluating also how well the localization branch performs in conjunction with progress estimation, we can see that predictions are closer to the ground truth compared to the same model trained with the linear progress formulation. This hints to the fact that phase-based progress reduces annotation ambiguity and therefore aids the optimization of the model. This trend can be observed in Figure 13, where the mAPP for ProgressNet and ProgressNet Static, along with the Random baseline, is reported varying the margin threshold. Interestingly, even the static version of the model is able to perform better than its counterpart with linear progress labels (see Figure 10), suggesting once again that the model is able to better understand the visual cues that are part of the development of the action.

Embedding comparison. We compare the results obtained by ProgressNet adopting feature embeddings from a state-of-the-art method that performs a related task, namely future frame synthesis. The rationale of this experiment is to validate the architectural choice of using a feature extractor specifically targeted for action progress estimation rather than taking generic features

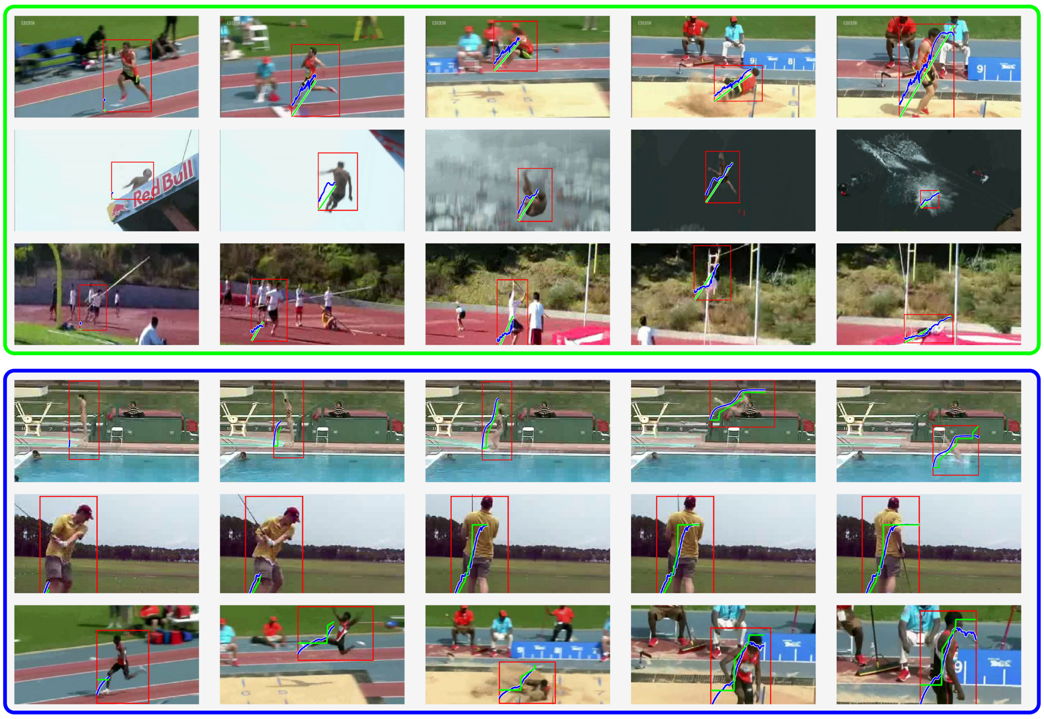

Fig. 14. Qualitative results with the linear based model (green frame) and phase-based model (blue frame). Each row represents the progression of an action. Progress values are plotted inside the detection box with time on the x-axis and progress values on the y-axis. Progress targets are shown in green and predicted progresses in blue.

from a pretrained model. As a feature extractor, we adopt Deep Voxel Flow (DWF) [34]. The model is composed of a fully convolutional encoder and decoder with a hourglass structure that takes a pair of temporally adjacent frames and generates the future one. We extract a convolutional feature map with 256 channels from the output of the encoder and apply ROI Pooling to project the bounding box enclosing the actor onto it. The resulting features are then fed to the temporal branch of ProgressNet that processes them with the LSTMs and the final fully connected layer to emit predictions. Since Deep Voxel Flow uses pairs of frames as input, we feed also the previous frame along with the current one to the model. We utilize the model with pretrained weights on UCF-101, which has been released online by the authors. As for ProgressNet we keep the training settings unchanged and we adopt phase-based annotations.

ProgressNet with DWF embeddings is still able to achieve appreciable performance on the task of action progress prediction, yet demonstrating a lower precision than the regular ProgressNet. In fact, the model obtains an MSE of 0.032, which is 52% higher than the original error (Table 4). Whereas the model is still able to correctly model atelic actions, achieving an error of 0.012, with a negligible increment of 0.002, DWF features lead to a worse modeling of telic actions: Whereas ProgressNet obtains an MSE of 0.037, the error of ProgressNet with Deep Voxel Flow rises to 0.055.

This highlights that, despite the tasks of estimating action progress and predicting future frames are indeed relevant to each other (since they both need to understand how an action evolves through time), action progress prediction is better dealt with a specifically designed model.

6.3 Qualitative Analysis

We inspect the results of ProgressNet trained with linear annotations and with phase-based annotations. Figure 14 shows some qualitative results with the two models on the UCF-101 dataset. It is interesting to notice how in some of the examples with the linear model, the predicted progress does not have a decise linear trend. Instead, it appears to follow the visual appearance of the action, changing its trend when there is a change in the semantics of the action. It appears that the model trained with the linear model has discovered to some extent the states that are made explicit with the phase-based annotations. For instance, the running phases in the first (*LongJump*) and third row (*PoleVaults*) clearly exhibit a different trend compared to the final parts of the actions, where the predictions increase more steadily toward completion. This behavior becomes evident when action phases are taken into account by the model. It can be seen that in the three examples the predicted progress curves change their slope when entering into a different phase.

7 CONCLUSION

In this article, we introduced ProgressNet, a model that can predict spatio-temporal localization of actions and at the same time understand their evolution by predicting progress online. We proposed two interpretations on progress: first, a linear one that has the advantage of being simple and applicable to any action detection dataset without any manual annotation, and, second, a phase-based interpretation that is more complex and requires a detailed manual annotation but at the same time provides a much richer and precise description of the ongoing action. To offer an appropriate study of action progress, we grounded our findings in the linguistics literature and terminology to characterize actions. In addition to our model, we proposed a Boundary Observant loss that helps to avoid trivial solutions by taking into account punctual phases that are present in the execution of the action. Experiments on two datasets showed that the proposed model is able to obtain promising performance.

REFERENCES

[1] J. K. Aggarwal and M. S. Ryoo. 2011. Human activity analysis: A review. *Comput. Surveys* 43, 3 (2011), 16:1–16:43.

[2] Mohammad Sadegh Aliakbarian, Fatemeh Sadat Saleh, Mathieu Salzmann, Basura Fernando, Lars Petersson, and Lars Andersson. 2017. Encouraging lstms to anticipate actions very early. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 280–289.

[3] Michael A. Arbib. 2006. A sentence is to speech as what is to action? *Cortex* 42, 4 (2006), 507–514.

[4] Shyamal Buch, Victor Escorcia, Chuanqi Shen, Bernard Ghanem, and Juan Carlos Niebles. 2017. SST: Single-stream temporal action proposals. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[5] Bernard Comrie. 1976. *Aspect: An Introduction to the Study of Verbal Aspect and Related Problems*. Vol. 2. Cambridge university press.

[6] Roeland De Geest, Efstratios Gavves, Amir Ghodrati, Zhenyang Li, Cees GM Snoek, and Tinne Tuytelaars. 2016. Online action detection. In *European Conference on Computer Vision*. 269–284.

[7] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. 2019. Temporal cycle-consistency learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1801–1810.

[8] Victor Escorcia, Cuong D Dao, Mihir Jain, Bernard Ghanem, and Cees Snoek. 2020. Guess where? Actor-supervision for spatiotemporal action localization. *Computer Vision and Image Understanding* 192 (2020), 102886.

[9] Bernard Ghanem Fabian Caba Heilbron, Victor Escorcia and Juan Carlos Niebles. 2015. ActivityNet: A large-scale video benchmark for human activity understanding. In *IEEE Conference on Computer Vision and Pattern Recognition*. 961–970.

[10] Cornelia Fermüller, Fang Wang, Yezhou Yang, Konstantinos Zampogiannis, Yi Zhang, Francisco Barranco, and Michael Pfeiffer. 2016. Prediction of manipulation actions. *International Journal of Computer Vision* (2016), 1–17.

[11] J. Randall Flanagan and Roland S. Johansson. 2003. Action plans used in action observation. *Nature* 424, 6950 (2003), 769.

[12] Adrien Gaidon, Zaid Harchaoui, and Cordelia Schmid. 2013. Temporal localization of actions with actoms. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 11 (2013), 2782–2795.

[13] Jiyang Gao, Zhenheng Yang, Chen Sun, Kan Chen, and Ram Nevatia. 2017. TURN TAP: Temporal unit regression network for temporal action proposals. In *IEEE International Conference on Computer Vision*.

[14] Georgia Gkioxari and Jitendra Malik. 2015. Finding action tubes. In *IEEE International Conference on Computer Vision*. 759–768.

[15] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks.. In *Artificial Intelligence and Statistics Conference*. 249–256.

[16] Tengda Han, Jue Wang, Anoop Cherian, and Stephen Gould. 2017. Human action forecasting by learning task grammarss. *arXiv preprint arXiv:1412.6980* (2017).

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2014. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision*. 346–361.

[18] Farnoosh Heidarivincheh, Majid Mirmehdi, and Dima Damen. 2016. Beyond action recognition: Action completion in RGB-D data. In *British Machine Vision Conference*.

[19] Farnoosh Heidarivincheh, Majid Mirmehdi, and Dima Damen. 2018. Action completion: A temporal model for moment detection. In *Proceedings of the British Machine Vision Conference (BMVC)*.

[20] Farnoosh Heidarivincheh, Majid Mirmehdi, and Dima Damen. 2019. Weakly-supervised completion moment detection using temporal attention. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*.

[21] Fabian Caba Heilbron, Juan Carlos Niebles, and Bernard Ghanem. 2016. Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In *IEEE Conference on Computer Vision and Pattern Recognition*. 1914–1923.

[22] Minh Hoai and Fernando De la Torre. 2014. Max-margin early event detectors. *International Journal of Computer Vision* 107, 2 (2014), 191–202.

[23] Rui Hou, Chen Chen, and Mubarak Shah. 2017. Tube convolutional neural network (T-CNN) for action detection in videos. In *IEEE International Conference on Computer Vision*.

[24] Haroon Idrees, Amir R Zamir, Yu-Gang Jiang, Alex Gorban, Ivan Laptev, Rahul Sukthankar, and Mubarak Shah. 2017. The THUMOS challenge on action recognition for videos "in the wild". *Computer Vision and Image Understanding* 155 (2017), 1–23.

[25] Hueihan Jhuang, Juergen Gall, Silvia Zuffi, Cordelia Schmid, and Michael J. Black. 2013. Towards understanding action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*. 3192–3199.

[26] Vicky Kalogeiton, Philippe Weinzaepfel, Vittorio Ferrari, and Cordelia Schmid. 2017. Action tubelet detector for spatio-temporal action localization. In *IEEE International Conference on Computer Vision*.

[27] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[28] Yu Kong, Zhiqiang Tao, and Yun Fu. 2017. Deep sequential context networks for action prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1473–1481.

[29] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. 2017. Unsupervised representation learning by sorting sequences. In *IEEE International Conference on Computer Vision*.

[30] Xinyu Li, Yanyi Zhang, Jianyu Zhang, Moliang Zhou, Shuhong Chen, Yue Gu, Yueyang Chen, Ivan Marsic, Richard A. Farneth, and Randall S. Burd. 2017. Progress estimation and phase detection for sequential processes. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 3, Article Article 73 (Sept. 2017), 20 pages. DOI: https://doi.org/10.1145/3130936

[31] Junwei Liang, Lu Jiang, Juan Carlos Niebles, Alexander G. Hauptmann, and Li Fei-Fei. 2019. Peeking into the future: Predicting future person activities and locations in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5725–5734.

[32] Daochang Liu, Tingting Jiang, and Yizhou Wang. 2019. Completeness modeling and context separation for weakly supervised temporal action localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1298–1307.

[33] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. 2016. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*. Springer, 21–37.

[34] Ziwei Liu, Raymond A. Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. 2017. Video frame synthesis using deep voxel flow. In *Proceedings of the IEEE International Conference on Computer Vision*. 4463–4471.

[35] Fuchen Long, Ting Yao, Zhaofan Qiu, Xinmei Tian, Jiebo Luo, and Tao Mei. 2019. Gaussian temporal awareness networks for action localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 344–353.

[36] Shugao Ma, Leonid Sigal, and Stan Sclaroff. 2016. Learning activity progression in LSTMs for activity detection and early detection. In *IEEE Conference on Computer Vision and Pattern Recognition*. 1942–1950.

[37] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.

[38] Michael Mathieu, Camille Couprie, and Yann LeCun. 2015. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440* (2015).

[39] Davide Moltisanti, Michael Wray, Walterio Mayol-Cuevas, and Dima Damen. 2017. Trespassing the boundaries: Labeling temporal bounds for object interactions in egocentric video. In *Proceedings of the IEEE International Conference on Computer Vision*. 2886–2894.

[40] A. Neumann and Andrew Zisserman. 2019. Future event prediction: If and when. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*.

[41] Phuc Xuan Nguyen, Deva Ramanan, and Charless C. Fowlkes. 2019. Weakly-supervised action localization with background modeling. In *Proceedings of the IEEE International Conference on Computer Vision*. 5502–5511.

[42] Katerina Pastra and Yiannis Aloimonos. 2012. The minimalist grammar of action. *Philosophical Transactions of the Royal Society B: Biological Sciences* 367, 1585 (2012), 103–117.

[43] A. Patra and J. A. Noble. 2018. Sequential anatomy localization in fetal echocardiography videos. *arXiv preprint arXiv:1810.11868* (2018).

[44] Xiaojiang Peng and Cordelia Schmid. 2016. Multi-region two-stream R-CNN for action detection. In *European Conference on Computer Vision*. 744–759.

[45] Ronald Poppe. 2010. A survey on vision-based human action recognition. *Image and Vision Computing* 28, 6 (2010), 976–990.

[46] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*. 91–99.

[47] Suman Saha, Gurkirt Singh, Michael Sapienza, Philip HS Torr, and Fabio Cuzzolin. 2016. Deep learning for detecting multiple space-time action tubes in videos. In *British Machine Vision Conference*.

[48] Zheng Shou, Jonathan Chan, Alireza Zareian, Kazuyuki Miyazawa, and Shih-Fu Chang. 2017. CDC: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[49] G. A. Sigurdsson, O. Russakovsky, and A. Gupta. 2017. What actions are needed for understanding human actions in videos? In *IEEE International Conference on Computer Vision*. 2137–2146.

[50] Gurkirt Singh, Suman Saha, Michael Sapienza, Philip Torr, and Fabio Cuzzolin. 2017. Online real-time multiple spatiotemporal action localisation and prediction. In *IEEE International Conference on Computer Vision*.

[51] Khurram Soomro, Amir R. Zamir, and Mubarak Shah. 2012. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402* (2012).

[52] Bilge Soran, Ali Farhadi, and Linda Shapiro. 2015. Generating notifications for missing actions: Don't forget to turn the lights off!. In *IEEE International Conference on Computer Vision*. 4669–4677.

[53] James Steele, Pier Francesco Ferrari, and Leonardo Fogassi. 2012. From action to language: Comparative perspectives on primate tool use, gesture and the evolution of human language. *Philosophical Transactions of the Royal Society B: Biological Sciences* 367, 1585 (2012), 4.

[54] Andru Putra Twinanda, Gaurav Yengera, Didier Mutter, Jacques Marescaux, and Nicolas Padoy. 2018. RSDNet: Learning to predict remaining surgery duration from laparoscopic videos without manual annotations. *IEEE Transactions on Medical Imaging* 38, 4 (2018), 1069–1078.

[55] Zeno Vendler. 1957. Verbs and times. *The Philosophical Review* 66, 2 (1957), 143–160.

[56] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. 2016. Anticipating visual representations with unlabeled video. In *IEEE Conference on Computer Vision and Pattern Recognition*. 98–106.

[57] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. 2016. Generating videos with scene dynamics. In *Advances In Neural Information Processing Systems*. 613–621.

[58] Xiaolong Wang, Ali Farhadi, and Abhinav Gupta. 2016. Actions ~ transformations. In *IEEE Conference on Computer Vision and Pattern Recognition*. 2658–2667.

[59] Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. 2015. Learning to track for spatio-temporal action localization. In *IEEE International Conference on Computer Vision*. 3164–3172.

[60] Yuanjun Xiong, Yue Zhao, Limin Wang, Dahua Lin, and Xiaoou Tang. 2017. A pursuit of temporal accuracy in general activity detection. *arXiv preprint arXiv:1703.02716* (2017).

[61] Mingze Xu, Mingfei Gao, Yi-Ting Chen, Larry S. Davis, and David J. Crandall. 2019. Temporal recurrent networks for online action detection. In *Proceedings of the IEEE International Conference on Computer Vision*. 5532–5541.

[62] Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei. 2016. End-to-end learning of action detection from frame glimpses in videos. In *IEEE Conference on Computer Vision and Pattern Recognition*. 2678–2687.

[63] Gang Yu and Junsong Yuan. 2015. Fast action proposals for human action detection and search. In *IEEE Conference on Computer Vision and Pattern Recognition*. 1302–1311.

[64] Zehuan Yuan, Jonathan C. Stroud, Tong Lu, and Jia Deng. 2017. Temporal action localization by structured maximal sums. In *IEEE Conference on Computer Vision and Pattern Recognition*. 3684–3692.

[65] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. 2017. Temporal action detection with structured segment networks. In *IEEE International Conference on Computer Vision*.

[66] Hongyuan Zhu, Romain Vial, and Shijian Lu. 2017. TORNADO: A spatio-temporal convolutional regression network for video action proposal. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5813–5821.

Received March 2020; revised April 2020; accepted May 2020