# When will you do what? - Anticipating Temporal Occurrences of Activities

Yazan Abu Farha, Alexander Richard, Juergen Gall
University of Bonn, Germany
{abufarha,richard,gall}@iai.uni-bonn.de

## Abstract

*Analyzing human actions in videos has gained increased attention recently. While most works focus on classifying and labeling observed video frames or anticipating the very recent future, making long-term predictions over more than just a few seconds is a task with many practical applications that has not yet been addressed. In this paper, we propose two methods to predict a considerably large amount of future actions and their durations. Both, a CNN and an RNN are trained to learn future video labels based on previously seen content. We show that our methods generate accurate predictions of the future even for long videos with a huge amount of different actions and can even deal with noisy or erroneous input information.*

## 1. Introduction

In the last years, we have seen a tremendous progress in the capabilities of computer systems to classify and segment activities in videos, *e.g.* [25, 22, 10]. These systems, however, analyze the past or in the case of real-time systems the present with a delay of a few milliseconds. For applications, where a moving system has to react or interact with humans, this is insufficient. For instance, collaborative robots that work closely with humans have to anticipate the activities of a human in the future. In contrast to humans that are very good in anticipating activities, developing methods that anticipate future activities from video data is very challenging and has just recently received an increase of interest.

Current works anticipate activities only for a very short time horizon of a few seconds. While early activity detection addresses the problem of inferring the class label of an action at the point when the activity starts or shortly thereafter [23, 5, 14, 24], other works predict the class label of the action that will happen next [19, 6, 9]. In the recent work [15], the starting time of the future activity is estimated as well.

In this work we go beyond the recognition of an ongoing activity or the anticipation of the next activity. We address the problem of anticipating all activities that will be happening within a time horizon of up to 5 minutes. This includes the classes and order of the activities that will occur as well as when each activity will start and end. Figure 5 shows a few example predictions.

To address this problem, we propose two novel approaches for this task. In both cases, we first infer the activities from the observed part of the video using an RNN-HMM [22]. The first approach builds on a recurrent neural network (RNN) that predicts for a given sequence of inferred activities the remaining duration of the ongoing activity as well as the duration and class of the next activity. The anticipated activities are then fed back to the RNN in order to anticipate activities for a longer time horizon. The second approach builds on a convolutional neural network (CNN). To this end, we convert the temporal sequence of inferred activities in a matrix that encodes both the length and the action label information. The CNN then predicts a matrix that encodes the length and the action labels of the anticipated activities. In contrast to the RNN approach, the CNN approach anticipates all activities in one pass.

We have evaluated the two approaches on two challenging datasets that contain long sequences and large variations. Both approaches outperform by a large margin a baseline that anticipates the activities based on a grammar and the average length of each activity. While the RNN and CNN perform similarly for a long time horizon of more than 40 seconds, the RNN performs better for shorter time horizons less than 20 seconds. Both approaches also outperform the method of [30] that does not anticipate activities directly but visual representations of future frames, which can then be used to classify the activities.

## 2. Related Work

Predicting future frames, poses or image segmentations in videos has been studied in several works [20, 16, 13, 31, 11, 27, 18, 29, 33, 32, 17, 12]. Approaches that predict future frames at a pixel-level, however, are limited to a few frames. Instead of predicting frames, a deep network is trained in [30] to predict visual representations of future frames. The predicted visual representations can then be used to classify actions or objects using standard classifiers. This approach, however, is also limited to a very short time horizon of only 5 seconds. This work has been extended in [3], where they use an encoder-decoder network to predict a sequence of future representations based on a history of previous frames. To get the predicted actions, the output of the encoder-decoder network is passed through another network that generates the action labels.

The task of early activity detection is also related, but it assumes that a partial observation of the ongoing activity is available, and the goal is to recognize this activity with the least possible amount of observations [23, 5]. Recent approaches for this task use Long Short-Term Memory (LSTM) networks with special loss functions that encourage early activity detection [14, 24].

A slightly longer time horizon is considered for approaches that predict the next action that will happen. [9] predict future actions from hierarchical representations of short clips or still images. They encode the observed frames in a multi-granular hierarchical representation of movements, and train different classifiers at each level in the hierarchy in a max-margin framework. Recently, [15] train a deep network to predict the future activity and its starting time from a sequence of preceding activities. Their model relies on appearance based and motion based features extracted from the observed activities to predict what will happen next in the future. [35] combine on-wrist motion sensing and visual observations to anticipate daily intentions. In [7], observed activities are modeled with spatio-temporal graphs which are used for anticipating object affordances, trajectories, and sub-activities. Besides of activities, anticipating goal states from first person videos has also been addressed. [21] model the human behavior with a Markov Decision Process (MDP) and use inverse reinforcement learning to infer all elements of the MDP and the reward function from first person vision videos. Then, they use the learned MDP to anticipate goal states and the length of trajectories towards them. Inverse reinforcement learning is also used in [36] for visual sequence forecasting.

Other works investigate future prediction in the sports domain. For example, [34] use augmented hidden conditional random field to predict the location of future events in sports, like predicting shot location in tennis. Recently, a framework has been introduced in [2] to predict the next move of players or the location of the ball in the immediate future from sports videos.

In contrast to previous work that anticipate the next activity only within a short time horizon of a couple of seconds, we address the problem of anticipating a sequence of activities including the start and end points within time horizons of up to 5 minutes.

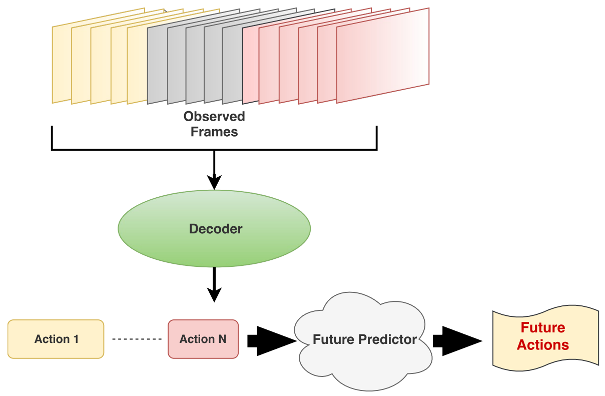


Figure 1. Proposed approach for future action prediction. From the observed frames, action labels are inferred by a decoder. The future predictor predicts from the inferred frame labels $\mathbf{c}_1^t$ the labels $\mathbf{c}_{t+1}^T$ that are yet to come.

## 3. Anticipating Activities

Our goal is to anticipate from an observed video what will happen next in the video for a given time horizon, which can take up to 5 minutes. As shown in Figure 5, we aim to predict for each frame in the future the label of the activity that will happen. More formally, let $\mathbf{x}_1^T = (x_1, \dots, x_T)$ be a video with $T$ frames. Given the first $t$ frames $\mathbf{x}_1^t$, the task is to predict the actions happening from frame $t+1$ to $T$. That is, we aim to assign action labels $\mathbf{c}_{t+1}^T = (c_{t+1}, \dots, c_T)$ to each of the unobserved frames.

### 3.1. Inferring Observed Activities

Instead of predicting the future actions $\mathbf{c}_{t+1}^T$ directly from the video frames $\mathbf{x}_1^t$, we first infer the actions $c_1, \dots, c_t$ for the given frames $x_1, \dots, x_t$ and then predict the future actions $c_{t+1}, \dots, c_T$ from the inferred actions $\mathbf{c}_1^t$ as it is illustrated in Figure 1. This has the advantage that we can separately study the impact of the network, which infers activities from observed video sequences, and the network that anticipates the future activities. In our experiments, we will also show that the predictor network performs worse if activities are directly anticipated from the observed video frames.

For inferring the activities $\mathbf{c}_1^t$ from $\mathbf{x}_1^t$, we use a hybrid RNN-HMM approach [22]. In contrast to [22], which train the method in a weakly supervised setting, we train the model fully supervised since in our training set each video frame $x_t$ is labeled with a class $c_t$.

For inferring the activities $\mathbf{c}_{t+1}^T$ from $\mathbf{c}_1^t$, we investigate two architectures. The first architecture is based on a recurrent neural network (RNN) and will be described in Section 3.2. The second architecture is based on a convolutional neural network (CNN) and will be described in Section 3.3.

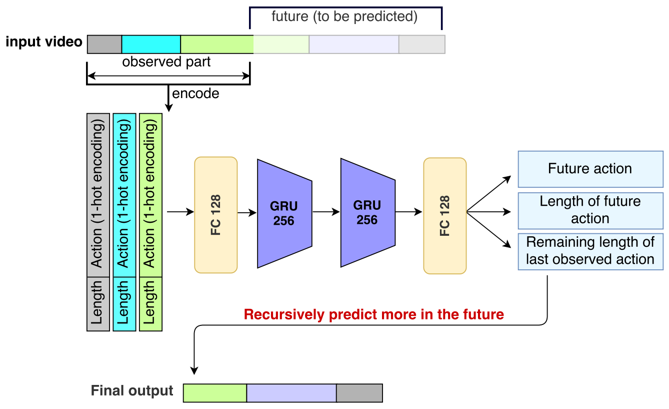


Figure 2. Architecture of the RNN system. The input is a sequence of *(length, 1-hot class encoding)*-tuples. The network predicts the remaining length of the last observed action and the label and length of the next action. Appending the predicted result to the original input, the next action segment can be predicted.

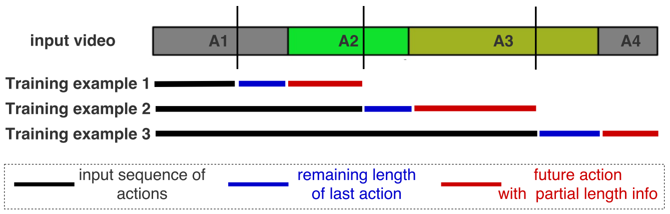

input video
A1
A2
A3
A4
Training example 1
Training example 2
Training example 3
input sequence of actions
remaining length of last action
future action with partial length info

Figure 3. Training data is generated by cutting the ground-truth segmentations at random points and using the left part as input and the next action segment to the right of the cut as ground-truth for the prediction.

## 3.2. RNN-based Anticipation

We can interpret future action prediction as a recursive sequence prediction: As input sequence, the RNN obtains all observed segments and predicts the remainder of the last segment as well as the next segment. This is repeated until the desired amount of future sequence of frames is reached.

More precisely, for each observed segment, the RNN gets its class label in form of a 1-hot encoding and the corresponding segment length, which is normalized by the video length, as input. Sequentially forwarding all those segments, three output predictions are made: the remaining length of the last observed segment as well as a label and a length for the next segment. This prediction is concatenated with the observed segments to form a new input for the network. The new input is again forwarded through the network to produce the next prediction. The final result is obtained by repeatedly forwarding the previously generated prediction until the desired amount of frames is predicted, see Figure 2.

As RNN architecture, we use two stacked layers of 256 gated recurrent units and fully connected layers at the input and output. As output layer for both length predictions, remaining length of current action and length of next action, we use a rectified linear unit to ensure positive length outputs. The label prediction is done via a softmax layer as usual for classification tasks.

**RNN Training.** The training data generation for the RNN is illustrated in Figure 3. Given a ground-truth labeling of a training sequence with $n$ action segments, $n-1$ training examples are generated out of it. For a segment $i < n$, a random split point is defined. Everything before this point is encoded as a sequence of $i$ tuples containing the length of the observed segment and its label as 1-hot encoding. Each such sequence is an input training example for the network. For segment $i+1$, another random split point is defined. The values between the first and second split point define the target the network should predict: A triplet consisting of the remaining length of segment $i$ ($l_r$), the length of the next action $i+1$ from its start up to the split point ($l_n$), and the label of the next action ($c$). Processing each training sequence like this, a large amount of input tuple sequences and target triplets is generated.

As loss for a single training example, we use

$$\mathcal{L} = -\log \hat{p}_c + (l_r - \hat{l}_r)^2 + (l_n - \hat{l}_n)^2, \qquad (1)$$

where $\hat{l}_r$ denotes the predicted remaining length of the current action, $\hat{l}_n$ denotes the predicted length of the next action, and $\hat{p}_c$ the predicted class probability of the next action. For training, we minimize the loss, which is summed over all training examples, by backpropagation through time.

## 3.3. CNN-based Anticipation

The CNN-based anticipation approach aims at predicting all actions directly in one single step, rather than relying on a recursive strategy such as the RNN. The given frame-wise labels $\mathbf{c}_1^t$ are encoded in a matrix $X$ with $C$ columns and $S$ rows. While the columns correspond to the $C$ action classes, the rows correspond to action segments. The number of rows for an action segment of length $l$ is given by $\lfloor \frac{l}{t} S \rfloor$ and, for each row $s$, $X_{sc} = 1$ for the label $c$ of the corresponding action segment and zero otherwise. The matrix is filled in the order the actions occur as illustrated in Figure 4. We set $S$ large enough such that each action segment covers at least one row.

The matrix $X$ that encodes the observed labels $\mathbf{c}_1^t$ is forwarded through a CNN which consists of two convolutional layers and two fully connected layers. The convolutional layers have 8 and 16 feature maps respectively and perform a $5 \times 1$ convolution followed by a rectified linear unit and max pooling. After the fully connected layers, the output layer is reshaped to a matrix $Y$, which has the same size as matrix $X$, and each row of $Y$ is $\ell_2$ normalized. We further

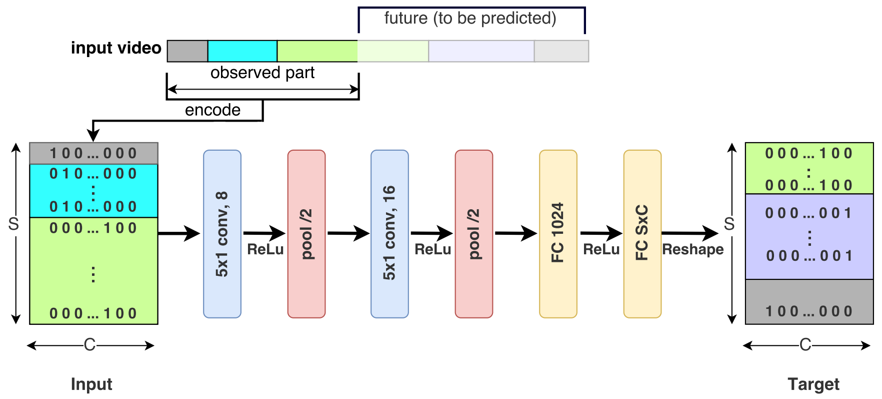


Figure 4. Architecture of the CNN-based anticipation approach. Both the input sequence and output sequence are converted into a matrix form where $C$ denotes the number of classes and $S$ corresponds to the number of video segments of a certain length. The binary values of the matrix indicate the label of each video segment.

apply a 1D Gaussian filter along each column for temporal smoothing. We will show in the experiments that the additional smoothing reduces the effect of spurious predictions of very short segments, see Figure 10. To convert $Y$ back into a sequence of labels $\mathbf{c}_{t+1}^{T}$, we compute for each row

$$\hat{c}_s = \arg\max_c Y_{sc} \tag{2}$$

and concatenate $\hat{c}_s$ where each row corresponds to $\lfloor \frac{T-t-1}{S} \rfloor$ frames.

**CNN Training.** Since the CNN approach predicts all actions directly while the RNN uses a recursive strategy, we also have to prepare the training data slightly differently. For each video in the training set, we generate 4 training examples by using the first 10%, 20%, 30%, and 50% of the video, respectively, as observation and the following 50% of the video as ground-truth for the prediction. For each training example, we convert the sequence of action labels $\mathbf{c}_1^t$ into the matrix $X$ and the labels of the following 50% of the video frames into the ground-truth matrix $Y$. To train the network, we use the squared error criterion over all output elements

$$\mathcal{L} = \frac{1}{SC} \sum_{s,c} (Y_{sc} - \hat{Y}_{sc})^2, \tag{3}$$

where $\hat{Y}$ is the prediction of the network. We found this loss in combination with the row-wise $\ell_2$-normalization of the output more robust than a standard softmax output with cross-entropy loss which we attribute to the smoothing properties of the softmax function, see Section 4 for an evaluation.

## 4. Experiments

### 4.1. Setup

We conduct our experiments on two benchmark datasets for action recognition. The **Breakfast** dataset [8] contains $1,712$ videos of 52 different actors making breakfast. Overall, there are 48 fine-grained action classes and about 6 action instances for each video. The average duration of the videos is 2.3 minutes. We use the four splits as proposed in [8].

The **50Salads** dataset [28] contains 50 videos with 17 fine-grained action classes. With an average length of 6.4 minutes, the videos are comparably long and contain 20 action instances per video on average. Following [28], we use five-fold cross-validation.

As evaluation metric for both datasets, we report the accuracy of the predicted frames as mean over classes (MoC, Mean over Classes).

**Video Representation.** We evaluate our systems on two settings. The first is with ground-truth observations, *i.e.* the observed labels are the ground truth annotation. This setting allows for a clean analysis of the prediction capabilities of our systems. As a second setting, we consider the labels of the observed part of the videos to be obtained using the decoder from [22]. This way, already the observed labels can contain errors and prediction is much harder as previous errors are propagated into the future. In order to obtain the decoded labels from [22], we compute improved dense trajectories over the observed frames $\mathbf{x}_1^t$ and then reduce the dimensionality to 64 using PCA. After that, Fisher vectors are computed for each frame using a temporal window of size 20. For Fisher vectors computation, a Gaussian mixture model (GMM) with 64 Gaussians is built using $150,000$ random samples of the dense trajectory features. Power- and $\ell_2$-normalization are applied, and at the end, the dimensionality is again reduced to 64 using PCA.

**Parameters.** For the CNN approach, we set the number of rows $S$ of the matrix $X$ to 128 for Breakfast. This ensures that each action segment covers at least one row. Since the average video length for 50Salads is about four times larger than for Breakfast, we use 512 for 50Salads. Since increasing $S$ and therefore sampling at a finer temporal resolution did not significantly change the results, we stick to these values for the remainder of the paper. For the post-processing, we use Gaussian smoothing with $\sigma = 3$ for Breakfast and $\sigma = 13$ for 50Salads. For the RNN approach, the normalized length input is scaled by the average number of actions in videos to ensure numerical stability. In all experiments, the Adam optimizer is used with a learning rate of 0.001.

**Baselines.** We define a naive baseline for future action prediction to compare against our two proposed systems. Based on the training data, the mean length of each action class is estimated. Further, a finite grammar generating all action sequences that have been observed during training is created. This is the same grammar as proposed in [22]. Given the observed actions $\mathbf{c}_1^t$, either based on the ground truth or on the action decoder [22], we randomly select an action sequence from the grammar that has $\mathbf{c}_1^t$ as prefix. We then predict action labels $\mathbf{c}_{t+1}^T$ such that the labels are consistent with the chosen action sequence of the grammar. Each action class from the chosen grammar path that has not been observed, *i.e*. that is to appear in the future, is added with its mean class length to the prediction until all required frames from $t+1$ to $T$ are predicted.

As a second baseline, we use a nearest neighbor approach. We search the nearest neighbor in the training set based on the observed part and use the remaining part as future prediction.

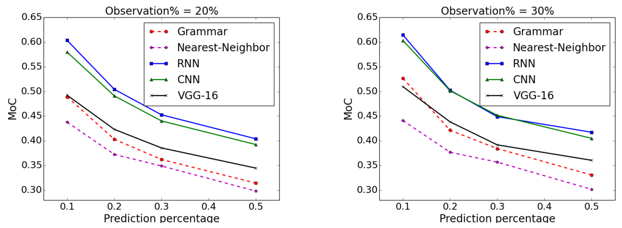
## 4.2. Prediction with ground-truth Observations

In order to provide a fair evaluation, we first assume that the observed segmentation $\mathbf{c}_1^t$ of the video is perfect, *i.e*. that the observed labels do not contain any errors. While this is not the case in most realistic settings, it allows to get a clean evaluation how well the systems can predict the future. With noisy observations, the results are more delusive as errors in the observed part are propagated to the future.

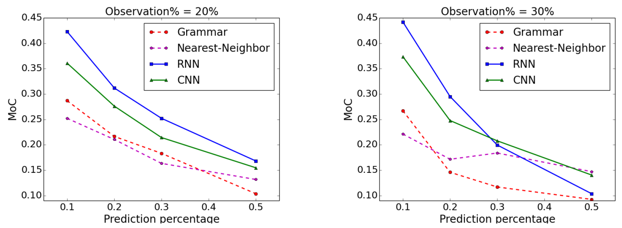
In Table 1 and Figure 6, the results on Breakfast and 50Salads are shown. Both the RNN model and the CNN model show good performance compared to the baseline. Independent of the fraction of the video that has been observed, *i.e*. 20%, or 30%, however, the RNN outperforms the CNN in most cases. The cases where the CNN is stronger are usually those where a large fraction of the video is predicted. The reason lies in the recursive structure of

Table 1. Results for future action prediction with ground-truth observations. Numbers represent accuracy as mean over classes.

| Observation % | **20%** | | | | **30%** | | | |
|---|---|---|---|---|---|---|---|---|
| Prediction % | **10%** | **20%** | **30%** | **50%** | **10%** | **20%** | **30%** | **50%** |
| ***Breakfast*** | | | | | | | | |
| Grammar | 0.4892 | 0.4033 | 0.3624 | 0.3146 | 0.5266 | 0.4215 | 0.3844 | 0.3309 |
| Nearest-Neighbor | 0.4378 | 0.3726 | 0.3492 | 0.2984 | 0.4412 | 0.3769 | 0.3570 | 0.3019 |
| RNN model | **0.6035** | **0.5044** | **0.4528** | **0.4042** | **0.6145** | **0.5025** | 0.4490 | **0.4175** |
| CNN model | 0.5797 | 0.4912 | 0.4403 | 0.3926 | 0.6032 | 0.5014 | **0.4518** | 0.4051 |
| ***50Salads*** | | | | | | | | |
| Grammar | 0.2869 | 0.2165 | 0.1832 | 0.1037 | 0.2671 | 0.1459 | 0.1169 | 0.0925 |
| Nearest-Neighbor | 0.2521 | 0.2105 | 0.1634 | 0.1317 | 0.2212 | 0.1715 | 0.1838 | **0.1471** |
| RNN model | **0.4230** | **0.3119** | **0.2522** | **0.1682** | **0.4419** | **0.2951** | 0.1996 | 0.1038 |
| CNN model | 0.3608 | 0.2762 | 0.2143 | 0.1548 | 0.3736 | 0.2478 | **0.2078** | 0.1405 |



(a)Results on Breakfast



(b)Results on 50Salads

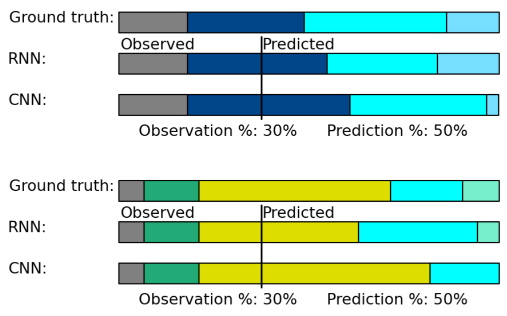
Figure 6. Results for future action prediction with ground-truth observations.

the RNN predictions: once a segment is predicted, it is appended to the observed part and used as input to predict the next sequence. Consequently, if the RNN output an erroneous prediction at some point, this error is likely to propagate through time. The CNN, on the contrary, uses the observed part of the video to predict all future actions directly, so errors are less likely to propagate from one segment to another. The behaviour can also be observed in the qualitative results in Figure 5 (a) and (c). For example in the case of (c), the CNN tends to miss small action segments, whereas the RNN seems to be more reliable.

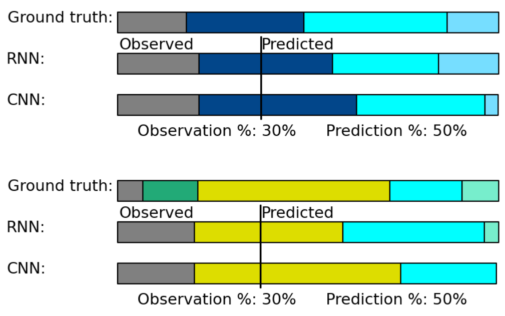
## 4.3. Prediction without ground-truth Observations

In this section, we evaluate the performance of our proposed systems given noisy annotations. In contrast to the clean ground-truth observations from the previous section, we now assume that the observed part of the video has been decoded using the system of [22] and, thus $\mathbf{c}_1^t$ is not perfect anymore but is likely to contain errors. The mean over frames accuracy of [22] when observing 30% of the video, for instance, is 43% on Breakfast and 68% on 50Salads. While the accuracy when observing 20% of the video is only 37% on Breakfast and 67% on 50Salads

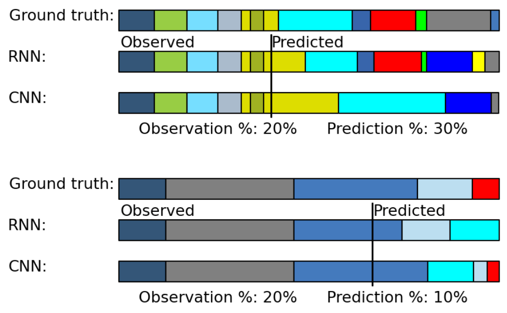
Compared to the amount of noise in the observed video



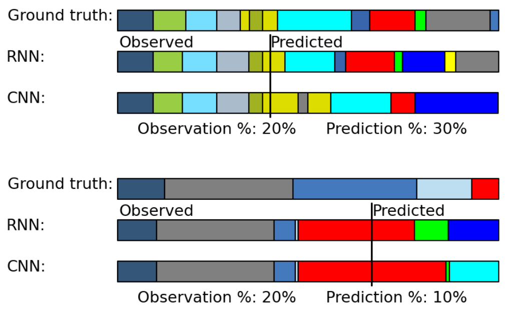
(a) Results on Breakfast with ground-truth observation



(b) Results on Breakfast without ground-truth observation



(c) Results on 50Salads with ground-truth observation



(d) Results on 50Salads without ground-truth observation

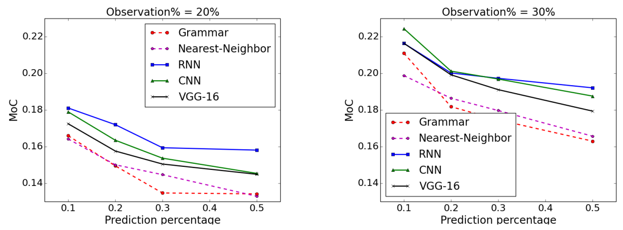
Figure 5. Qualitative results for the future action prediction task for both, RNN and CNN with and without ground-truth observations.

labeling, the prediction results are still surprisingly stable, see Table 2 and Figure 7. While the drop in performance on the Breakfast dataset (Table 1 vs. Table 2) is comparably large, both systems still achieve a good performance compared to the baseline. On 50Salads, the overall loss of accuracy compared to the system with perfect observations is surprisingly small. This can on the one hand be attributed to the better performance of the decoder [22]. On the other hand, the inter-class dependencies on 50Salads are very strong, making it easier for both RNN and CNN to learn valid action sequences.

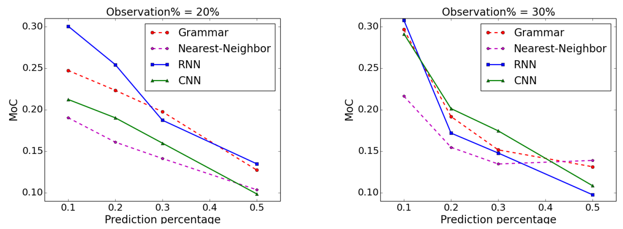
We would also like to put emphasis on the qualitative results for noisy observations, see Figure 5 (b) and (d). Particularly, the case of (d) is interesting: the decoder mistakenly predicted incorrect label for the last observed segment. For both models, RNN and CNN, this error propagates further to future segment predictions.

Table 2. Results for future action prediction without ground-truth observations. Numbers represent accuracy as mean over classes.

| Observation % | 20% | | | | 30% | | | |
|---|---|---|---|---|---|---|---|---|
| Prediction % | **10%** | **20%** | **30%** | **50%** | **10%** | **20%** | **30%** | **50%** |
| ***Breakfast*** | | | | | | | | |
| Grammar | 0.1660 | 0.1495 | 0.1347 | 0.1342 | 0.2110 | 0.1818 | 0.1746 | 0.1630 |
| Nearest-Neighbor | 0.1642 | 0.1501 | 0.1447 | 0.1329 | 0.1988 | 0.1864 | 0.1797 | 0.1657 |
| RNN model | **0.1811** | **0.1720** | **0.1594** | **0.1581** | 0.2164 | 0.2002 | **0.1973** | **0.1921** |
| CNN model | 0.1790 | 0.1635 | 0.1537 | 0.1454 | **0.2244** | **0.2012** | 0.1969 | 0.1876 |
| ***50Salads*** | | | | | | | | |
| Grammar | 0.2473 | 0.2234 | **0.1976** | 0.1274 | 0.2965 | 0.1918 | 0.1517 | 0.1314 |
| Nearest-Neighbor | 0.1904 | 0.1610 | 0.1413 | 0.1037 | 0.2163 | 0.1548 | 0.1347 | **0.1390** |
| RNN model | **0.3006** | **0.2543** | 0.1874 | **0.1349** | **0.3077** | 0.1719 | 0.1479 | 0.0977 |
| CNN model | 0.2124 | 0.1903 | 0.1598 | 0.0987 | 0.2914 | **0.2014** | **0.1746** | 0.1086 |



(a)Results on Breakfast



(b)Results on 50Salads

Figure 7. Results for future action prediction without ground-truth observations.

As the videos have different lengths, the proposed models might behave differently depending on the length of the videos. An evaluation on three categories of videos from Breakfast based on the prediction length is shown in Figure 8 for the case of observing 30% of the video and predicting the 50% that follows. While the models perform well on both short and long videos, they achieve a better performance for shorter videos. This is mainly because the

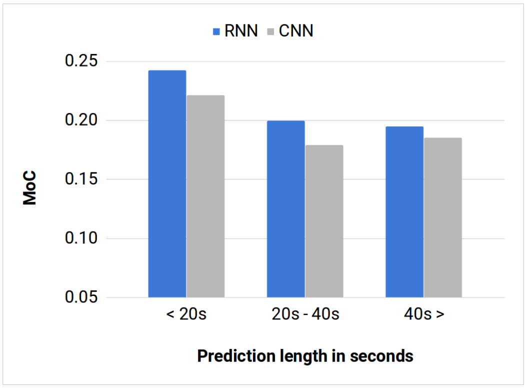


Figure 8. Performance of the models on videos with different lengths from the Breakfast dataset without ground-truth observations for the case of observing 30% of the videos and predicting the remaining 50%.

Table 3. Accuracy over next, 2nd action, and 3rd action for 30% observation and 50% prediction for Breakfast without ground-truth observations. The action is correctly detected if the IoU of the predicted action segment with the annotated action segment is $> 0.5$.

| | 1st action | 2nd action | 3rd action |
|---|---|---|---|
| Grammar | 0.2232 | 0.1325 | 0.1117 |
| Nearest-Neighbor | 0.2295 | 0.1323 | 0.1234 |
| RNN | **0.2643** | **0.1773** | **0.1792** |
| CNN | 0.2595 | 0.1603 | 0.1425 |

duration of the predicted future is less for shorter videos, which makes the prediction task much easier compared to long video sequences. A similar behaviour can also be observed when considering the number of actions that are predicted in the future. As shown in Table 3, the accuracy drops as we predict more in the future.

## 4.4. Analysis of the CNN Model

**Model Architecture** Compared to commonly used CNN architectures such as VGG-16 or ResNet, our model is comparably small. We stick to this simple architecture since we have very limited amount of training data, and complex models would easily overfit on the training set. A comparison of our architecture to a VGG-16 [26] on Breakfast with and without ground truth annotation is provided in Figure 6 and Figure 7 respectively. The performance of our architecture compared to VGG-16 is clearly better with an improvement up to 9% when using the ground truth annotation as observations. Note that also looking small, our architecture already has around 6m parameters due to the fully connected layers at the end.

**Loss Function** Our choice of the loss function for the CNN based model is a squared loss preceded by an $\ell_2$ normalization layer. However, for classification tasks, a softmax with cross-entropy loss is usually used. Since the future action prediction task can be viewed as special kind of classification, a softmax layer trained with the cross-entropy loss seems a more suitable choice on first glance. Yet, a comparison of both loss types shows a clear superiority of the squared loss combined with the $\ell_2$ normalization layer, see Table 4. We attribute this to the smoothing properties of the softmax layer. Even large differences of the input values to a softmax layer lead to comparably small differences in the output probability distribution. This is a desirable effect in classification tasks, for our task, however, it leads to frequent changes of the maximizing label index, which corresponds to over-segmentations. The effect can also be observed in Figure 9. While strong over-segmentation can be observed for the softmax output, this effect nearly completely vanishes using the $\ell_2$ normalization layer. Nevertheless, even with the $\ell_2$ normalization layer, a small over-segmentation effect might still be visible in some cases, which can be eliminated by the post-processing step as shown in Figure 10.

Table 4. Comparing different loss functions for the CNN model on Breakfast with ground-truth observations. Numbers represent accuracy as mean over classes.

| Observation % | 20% | | | | 30% | | | |
|---|---|---|---|---|---|---|---|---|
| Prediction % | **10%** | **20%** | **30%** | **50%** | **10%** | **20%** | **30%** | **50%** |
| CNN (cross-entropy) | 0.5177 | 0.4280 | 0.3940 | 0.3625 | 0.5499 | 0.4838 | 0.4324 | 0.3858 |
| CNN (squared loss) | **0.5797** | **0.4912** | **0.4403** | **0.3926** | **0.6032** | **0.5014** | **0.4518** | **0.4051** |

## 4.5. Future Prediction Directly from Features

So far, we considered observations that are either frame-wise ground truth action labels, or those labels that are obtained by decoding the observed frame-wise features. In this section, we evaluate the performance of our models when applied directly on the observed frame-wise features and compare it to the two-step approach. We only use the CNN model for this evaluation. Originally, the input of the model is a matrix with $C$ columns that correspond to the number of classes, and $S$ rows that represent the temporal resolution. Each row is a 1-hot encoding that represents the action label of the corresponding frames in the observed sequence. However, when applying this model to features directly, $C$ is equal to the dimensionality of the features, which is 64 in our case for the Fisher vectors features. $S$ is kept at 128 as before. The observed video features are down- or upsampled to have exactly 128 frames. We use the same training protocol that is used for the previous experiments, by considering the set $\{10\%, 20\%, 30\%, 50\%\}$ as observation percentages, and the target is always the 50% that follows immediately after the observations. Table 5 shows the results of the CNN model when applied on features directly compared to the two-step approach. As shown in the table, using the two-step approach outperforms the direct prediction from features by a large margin, *i.e.* up to 5%. Predicting the future directly from features is a harder

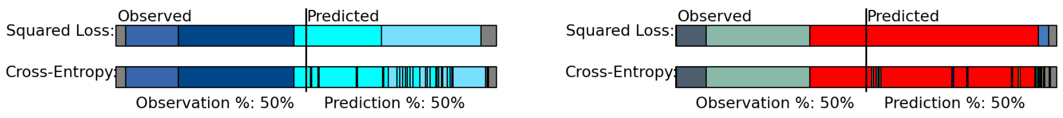


Figure 9. Results of the CNN model using the cross-entropy loss vs. the squared-error loss.

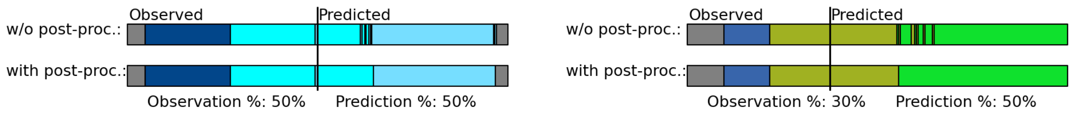


Figure 10. Results of the CNN model with and without post-processing.

problem, since the model has to recognize the observed actions first, and capture the relevant information to anticipate the future, while for the two-step approach, these two tasks are decoupled. This allows to use a strong decoding model to recognize the observed actions, and restricting the future predictor to capture the context over action classes only instead of frame-wise features. A similar conclusion was reached by [4] where semantic labels of the unseen parts are predicted in the spatial domain of an image. It has been shown that segmenting the observed part of the image first and then using the segmented image for prediction achieves better results than using the RGB image directly.

Table 5. Results for future action prediction directly from features on the Breakfast dataset. Numbers represent accuracy as mean over classes.

| Observation % | 20% | | | | 30% | | | |
|---|---|---|---|---|---|---|---|---|
| Prediction % | 10% | 20% | 30% | 50% | 10% | 20% | 30% | 50% |
| CNN (features) | 0.1278 | 0.1162 | 0.1121 | 0.1027 | 0.1772 | 0.1687 | 0.1548 | 0.1409 |
| CNN (w/o GT obs.) | **0.1790** | **0.1635** | **0.1537** | **0.1454** | **0.2244** | **0.2012** | **0.1969** | **0.1876** |

## 4.6. Comparison with the State-of-the-Art

To the best of our knowledge, long term future action prediction has not been addressed before. Most works focus on predicting the immediate future. Vondrick *et al.* [30], for instance, train a model to predict AlexNet features of a frame one second in the future. Based on these predictions, an SVM is trained to determine the action label of the future frame.

Since [30] train their future prediction model on $600h$ of videos from the web, which are not made publicly available, we use the recent Kinetics network to generate deep CNN features that have shown to generalize extremely well on several action recognition datasets and are the current state-of-the-art [1]. We run the approach of [30] on both, Breakfast and 50Salads, and compare their results to our model. To provide a fair comparison, our model is trained in a way that the input sequences always end one second before the next action segment starts. The results are shown in Table 6. Our approach outperforms the system of [30] by a large margin. Note that for both datasets, predicting an action label only based on a single frame is particularly hard and framewise action classifiers achieve less than 10% accuracy. In contrast to [30], our approaches make use of temporal context of the previously observed video content, which is crucial for reliable predictions.

Table 6. Comparison with the state-of-the-art prediction approaches: accuracy of predicting future actions one second before they start.

| | Breakfast | 50Salads |
|---|---|---|
| Vondrick *et al.* [30] | 8.1 | 6.2 |
| RNN model | **30.1** | **30.1** |
| CNN model | 27 | 29.8 |

## 5. Conclusion

We have introduced two efficient methods to predict future actions in videos, which is a task that has not been addressed before. While most existing prediction approaches focus on early anticipation of an already ongoing action or predict at most one action in the future, our methods are the first to predict video content of up to several minutes length. Proposing two models to address the task, an RNN and a CNN, we obtain accurate predictions that scale well along different datasets and videos with varying lengths, varying quality of observed data, and huge variations in the possible future actions.

## References

[1] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 8

[2] P. Felsen, P. Agrawal, and J. Malik. What will happen next? forecasting player moves in sports videos. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[3] J. Gao, Z. Yang, and R. Nevatia. RED: reinforced encoder-decoder networks for action anticipation. In *British Machine Vision Conference (BMVC)*, 2017. 2

[4] M. Garbade and J. Gall. Thinking outside the box: Spatial anticipation of semantic categories. In *British Machine Vision Conference (BMVC)*, 2017. 8

[5] M. Hoai and F. De la Torre. Max-margin early event detectors. *International Journal of Computer Vision*, 107(2):191–202, 2014. 1, 2

[6] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[7] H. S. Koppula and A. Saxena. Anticipating human activities using object affordances for reactive robotic response. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):14–29, 2016. 2

[8] H. Kuehne, A. Arslan, and T. Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 780–787, 2014. 4

[9] T. Lan, T.-C. Chen, and S. Savarese. A hierarchical representation for future action prediction. In *European Conference on Computer Vision (ECCV)*, pages 689–704. Springer, 2014. 1, 2

[10] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager. Temporal convolutional networks for action segmentation and detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1

[11] X. Liang, L. Lee, W. Dai, and E. P. Xing. Dual motion gan for future-flow embedded video prediction. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 1

[12] W. Lotter, G. Kreiman, and D. Cox. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*, 2016. 1

[13] P. Luc, N. Neverova, C. Couprie, J. Verbeek, and Y. LeCun. Predicting deeper into the future of semantic segmentation. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 1

[14] S. Ma, L. Sigal, and S. Sclaroff. Learning activity progression in lstms for activity detection and early detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1942–1950, 2016. 1, 2

[15] T. Mahmud, M. Hasan, and A. K. Roy-Chowdhury. Joint prediction of activity labels and starting times in untrimmed videos. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 2

[16] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015. 1

[17] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2863–2871, 2015. 1

[18] V. Patraucean, A. Handa, and R. Cipolla. Spatio-temporal video autoencoder with differentiable memory. *arXiv preprint arXiv:1511.06309*, 2015. 1

[19] M. Pei, Z. Si, B. Z. Yao, and S.-C. Zhu. Learning and parsing video events with goal and intent prediction. *Computer Vision and Image Understanding*, 117(10):1369–1383, 2013. 1

[20] M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra. Video (language) modeling: a baseline for generative models of natural videos. *arXiv preprint arXiv:1412.6604*, 2014. 1

[21] N. Rhinehart and K. M. Kitani. First-person activity forecasting with online inverse reinforcement learning. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[22] A. Richard, H. Kuehne, and J. Gall. Weakly supervised action learning with rnn based fine-to-coarse modeling. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2, 4, 5, 6

[23] M. S. Ryoo. Human activity prediction: Early recognition of ongoing activities from streaming videos. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 1036–1043, 2011. 1, 2

[24] M. Sadegh Aliakbarian, F. Sadat Saleh, M. Salzmann, B. Fernando, L. Petersson, and L. Andersson. Encouraging lstms to anticipate actions very early. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 2

[25] Z. Shou, D. Wang, and S.-F. Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1049–1058, 2016. 1

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 7

[27] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using lstms. In *International Conference on Machine Learning (ICML)*, pages 843–852, 2015. 1

[28] S. Stein and S. J. McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 2013), Zurich, Switzerland*, 2013. 4

[29] R. Villegas, J. Yang, Y. Zou, S. Sohn, X. Lin, and H. Lee. Learning to generate long-term future via hierarchical prediction. *arXiv preprint arXiv:1704.05831*, 2017. 1

[30] C. Vondrick, H. Pirsiavash, and A. Torralba. Anticipating visual representations from unlabeled video. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 98–106, 2016. 1, 8

[31] C. Vondrick and A. Torralba. Generating the future with adversarial transformers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[32] V. Vukotić, S.-L. Pintea, C. Raymond, G. Gravier, and J. Van Gemert. One-step time-dependent future video frame prediction with a convolutional encoder-decoder neural network. *arXiv preprint arXiv:1702.04125*, 2017. 1

[33] J. Walker, K. Marino, A. Gupta, and M. Hebert. The pose knows: Video forecasting by generating pose futures. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 1

[34] X. Wei, P. Lucey, S. Vidas, S. Morgan, and S. Sridharan. Forecasting events using an augmented hidden conditional random field. In *Asian Conference on Computer Vision (ACCV)*, pages 569–582, 2014. 2

[35] T.-Y. Wu, T.-A. Chien, C.-S. Chan, C.-W. Hu, and M. Sun. Anticipating daily intention using on-wrist motion triggered sensing. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[36] K.-H. Zeng, W. B. Shen, D.-A. Huang, M. Sun, and J. Carlos Niebles. Visual forecasting by imitating dynamics in natural sequences. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 2