# Long-Term Anticipation of Activities with Cycle Consistency (Supplementary Material)

Yazan Abu Farha[1], Qiuhong Ke[2], Bernt Schiele[3], and Juergen Gall[1]

[1]University of Bonn, Germany
[2]The University of Melbourne, Australia
[3]MPI Informatics, Germany

We provide more details about the attention module that we use for the sequence-to-sequence model. We further provide more ablation experiments to study the impact of the input to the recurrent encoder and decoder.

## 1 The Attention Module

For the sequence-to-sequence model, we use a recurrent encoder-decoder architecture based on gated recurrent units (GRUs). The encoder encodes the observed frames in a single vector which will be used to decode the future activities. Since the input to the encoder are frame-wise features which might be very long, the output of the encoder $h^e_{t_o}$ might not be able to capture all the relevant information. To alleviate this problem we combine the decoder with an attention mechanism using a multi-head attention module.

Given the current hidden state of the decoder $h^d_m$ and the output of the encoder at each step $h^e_{1:t_o} = (h^e_1, \dots, h^e_{t_o})$, the attention output for a single head is computed by first generating the query (q), the keys (K) and values (V) as described in the following

$$q = W_q h^d_m, \tag{1}$$
$$K = W_k h^e_{1:t_o}, \tag{2}$$
$$V = W_v h^e_{1:t_o}, \tag{3}$$

where $q \in \mathbb{R}^{d_A}$ is a column vector, $K,\ V \in \mathbb{R}^{d_A \times t_o}$ and $d_A$ is set to be one-eighth of the hidden state size of the GRU. We normalize these tensors by the $\|.\|_2$ norm and then compute the output using a weighted sum of the values

$$O_h = Softmax(q^T K)V^T, \tag{4}$$

where $O_h \in \mathbb{R}^{d_A}$ is the output of a single head. We use 8 heads and concatenate the output of these heads, apply a linear transformation and then concatenate the output with the input of the decoder at each step.

**Table 1.** Comparison between passing features vs. passing class probabilities of the observed frames from the recognition module to the subsequent modules on the Breakfast dataset. Numbers represent mean over classes (MoC) accuracy.

| Observation % | 20% | | | | 30% | | | |
|---|---|---|---|---|---|---|---|---|
| Prediction % | 10% | 20% | 30% | 50% | 10% | 20% | 30% | 50% |
| Ours with class probabilities | 25.53 | 23.39 | 22.50 | **21.70** | 28.95 | 27.36 | 25.06 | 24.32 |
| Ours with features | **25.88** | **23.42** | 22.42 | 21.54 | **29.66** | **27.37** | **25.58** | **25.20** |
| Ours with features and prob. | 25.15 | 23.41 | **22.53** | 21.64 | 29.21 | 26.62 | 24.65 | 24.69 |

**Table 2.** Impact of passing the predicted duration at each step to the recurrent decoder on the Breakfast dataset. Numbers represent mean over classes (MoC) accuracy.

| Observation % | 20% | | | | 30% | | | |
|---|---|---|---|---|---|---|---|---|
| Prediction % | 10% | 20% | 30% | 50% | 10% | 20% | 30% | 50% |
| Ours with duration input | 25.21 | 20.94 | 19.61 | 20.38 | 28.20 | 24.71 | 23.58 | 23.15 |
| Ours without duration input | **25.88** | **23.42** | **22.42** | **21.54** | **29.66** | **27.37** | **25.58** | **25.20** |

## 2 Frame-wise Features vs. Frame-wise Labels

In our model, we use the output features of the recognition module as input to the subsequent modules. Since the recognition module also predicts frame-wise class probabilities of the observed frames, another option would be to pass these probabilities instead of the features. In this section, we provide a comparison between these two variants. As shown in Table 1, passing frame-wise class probabilities gives a comparable performance to frame-wise features. Nevertheless, using the features is slightly better. This is expected as the features contain much richer information that can be utilized by the subsequent modules. We also tried concatenating the frame-wise features and the frame-wise class probabilities and pass the concatenated tensor to the sequence-to-sequence module. However, it did not improve the results.

## 3 Impact of Passing the Duration to the Decoder

The GRU in the recurrent decoder takes as input at each step the predicted activity label from the previous step as described in (2) in the main paper. In this section, we study the impact of passing the predicted duration to the decoder as well, *i.e.*

$$h_m^d = GRU([A_{m-1}, \hat{\ell}_{m-1}],\ h_{m-1}^d). \tag{5}$$

As shown in Table 2, passing the duration to the decoder results in a degradation in performance. This is due to passing the duration before applying the softmax, which means that there might be huge variations in the duration value between different iterations. While passing the duration after applying the softmax is not feasible, passing only the activity labels provides enough information for the decoder. We also tried to predict the absolute duration. In this case, no

EOS symbol is predicted and the decoder keeps generating future activities and their duration until the desired time horizon is predicted. However, the model performed very poorly in this setup. This is expected since predicting a stopping symbol is much easier than predicting the absolute duration of all activities.