Supplementary Material: Learning to Predict Activity Progress by Self-Supervised Video Alignment

Gerard Donahue
Northeastern University
Boston, MA, USA
donahue.g@northeastern.edu

Ehsan Elhamifar
Northeastern University
Boston, MA, USA
e.elhamifar@northeastern.edu

In the supplementary material, we present details on the GTCC window, the GTCC drop curriculum, hyperparameters, and the MCN architecture. Additionally, we show two studies showing frame-retrieval results and online progress estimations on in-the-wild datasets.

1. Window Proportion

In *Sec. 4.1* of the main paper, we introduced the window (w) which is used when computing $\mu_{\boldsymbol{\beta},k}$ and $\sigma^2_{\boldsymbol{\beta},k}$ for the final loss, $\mathcal{L}_{\text{multi-cbr}}$. When developing GTCC, we considered three options for handling this window, shown in Fig. 1.

Option 1: No Window. Similar to TCC [1], we experimented with no window. In this case, all of $\boldsymbol{\beta}^{(k)}$ is used for calculating $\mu_{\boldsymbol{\beta},k}$ and $\sigma^2_{\boldsymbol{\beta},k}$. This approach is ill-suited for repeating actions, as two peaks will cause a central mean that is not representative of either peak. Fig. 1 shows low error when using no window, even though there are two peaks that both are distant from index i.

Option 2: Central Window. Next, we considered a central windowing approach, where i lies central in the window of size w. We observed that a central window still wrongly causes low error in many cases. If i is central to the window and the distribution observes high variance, then the mean will be central. This is an issue because in $\mathcal{L}_{\text{multi-cbr}}$, mean error is in the numerator and variance is in the denominator. So with high variance and low error we will observe extremely low error when there are no peaks in $\boldsymbol{\beta}$ around i.

Option 3: Stochastic Window. To overcome both aforementioned challenges with option 1 and 2, we employed a stochastic window in the GTCC formulation, where i exists in a random position in the bounds of the window. Shown in Fig. 1, we are able to capture the peaks of the distribution in proximity to i, exhibiting high error correctly. Intuitively, if the trained GTCC model can be robust to where the window is placed and observes low error for many random windows surrounding i, then there is a strong peak at i.

Setting w. We set w dynamically based on the length, N, of the given video, $\boldsymbol{U}$. Therefore, we set a proportion hyperparameter for each dataset, such that $w = \text{proportion} \cdot N$. For both in-the-wild datasets (*EgoProceL* and *COIN*) we set the proportion to 0.2, and for monotonic datasets (*Pouring* and *Penn Action*) we set the proportion to 0.5.

2. Drop Curriculum

To increase the stability of GTCC during training, we employed a curriculum for phasing drop capability into GTCC throughout training. The general philosophy of this curriculum is to begin training by aligning all frames, and then slowly allowing the model to begin identifying frames as droppable frames. To do this, we define a value $\eta \in [0, 1]$ that determines how slowly the curriculum progresses. If η is close to 1, then the drop capability is slow to arise. With η, we slightly modify the sigmoid function used in *Eq. 10* of the main paper to be

$$\sigma^{\text{drop}}(x) = \eta^E + (1 - \eta^E) \cdot \sigma(x),$$

which vertically shrinks the sigmoid function. Here, E is the currect epoch and σ is the traditional sigmoid function. This then leaves *Eq. 10* of the main paper to be

$$\text{P}_{\text{drop}}(\boldsymbol{u}_i | \boldsymbol{V}) = \sigma^{\text{drop}}\big(\begin{bmatrix} \boldsymbol{u}_i^\top & 1 \end{bmatrix} \boldsymbol{c}(\boldsymbol{V}) \big),$$

in practice. The value for η are shown in the next section.

3. Hyperparameters

Below is the table of relevant hyperparameters for our method. Please note that all hyperparameters for the baselines were taken from the original papers.

4. MCN Details

Multi-head Cross-task Network (MCN) begins by encoding the frames of the video with a single base network, which is the same ResNet architecture used in the single-task setting to obtain video $\boldsymbol{U} \in \mathbb{R}^{N \times D}$. This embedded video is

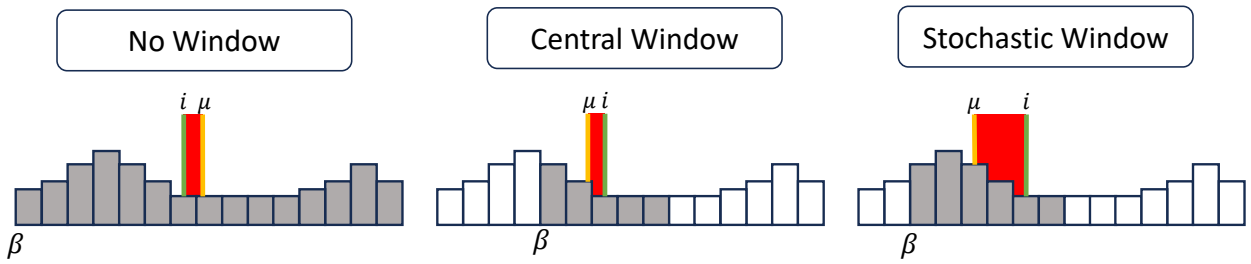

Figure 1. Here we show the affect of no window, central window, and our stochastic window on the loss calculation over discrete $\boldsymbol{\beta}$ distributions. The window is shown with darkened probability densities (white densities lie outside the window). The vertical green line represents frame $\boldsymbol{u}_i$, and the vertical orange line represents the mean index calculated from the darkened probabilities within the window. The red area between i and μ represents error.

Hyperparameter	**Value**
Batch Size	2
Number of Frames	20
Optimizer	ADAM
Learning Rate	1.0×10^{-4}
Softmax Temperature (τ)	0.1
Alignment Variance (λ)	1.0×10^{-3}
Drop-curriculum speed (η)	0.95 (E,C) 1.00 (PA,P)
GMM Components (K)	15 (E,C) 5 (PA,P)
Window Proportion	0.2 (E,C) 0.5 (PA,P)
Output Dimensionality (D)	128

Table 1. Hyperparameter Values

then given to L different head networks, which are fully-connected feed-forward networks. Each network outputs it's own video embedding, and all of the features are concatenated in the time dimension, where the attention network takes this concatenated matrix as input, and decides the active head networks at each time instant.

Because the MCN architecture is built for the multi-task setting, where one network is trained to align all tasks individually, we scale the number of attention heads, L, with the number of tasks being trained on for each dataset. As such, for Penn Action ($L = 13$), for CMU-MMAC ($L = 5$), and for EGTEA Gaze+ ($L = 7$). For all dropout networks, head networks, and attention networks in the MCN experiments, we used simple fully-connected feed-forward networks with $256, 1024, 512, 256$ neurons in each layer, respectively.

References

[1] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman. Temporal cycle-consistency learning. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.