Drop-DTW: Aligning Common Signal Between Sequences While Dropping Outliers (Supplementary Material)

Nikita Dvornik [1,2] Isma Hadji [1] Konstantinos G. Derpanis [1] Animesh Garg [2] Allan D. Jepson [1]
[1]Samsung AI Centre Toronto
[2]University of Toronto, Vector Institute
{isma.hadji, allan.jepson}@samsung.com
{n.dvornik, k.derpanis}@partner.samsung.com
garg@cs.toronto.edu

1 Summary

Our supplemental is organized as follows. Sec. 2.1 provides the details of the full version of our Drop-DTW algorithm that allows for dropping outliers from both sequences during alignment. Sec. 2.2 provides implementation details for the the asymmetric match cost. Sec. 2.3 describes the regularization loss used for the multi-step localization application (Sec. 4.2 in the main paper). Finally, Sec. 3 provides additional details for our experimental setups.

2 Technical approach details

In this section, we provide additional details of our Drop-DTW algorithm and its components.

2.1 Drop-DTW algorithm

Algorithm 1 presents the full version of Drop-DTW. This version allows to drop outlier elements from both sequences, X and Z. In Algorithm 1, the operation $\oplus$ between a set and a scalar increases every element of the set by the scalar, i.e., $\{s_i\}_i^N \oplus c = \{s_i + c\}_i^N$. The main idea of Algorithm 1 is to simultaneously solve four dynamic programs (and fill their corresponding tables), i.e., D^{zx}, D^{z-}, D^{-x}, and D^{--}:

- $D_{i,j}^{zx}$ corresponds to the optimal cost of a feasible prefix path that matches $(z_1, \dots, z_i)$ and $(x_1, \dots, x_j)$ with Drop-DTW, given that this feasible prefix path ends with z_i and x_j matched to each other.
- $D_{i,j}^{z-}$ represents the optimal matching cost for a prefix path ending with element x_j dropped from the matching, and z_i matched to some previous element $x_k, k < j$.
- $D_{i,j}^{-x}$ corresponds to dropping z_i while matching x_j.
- $D_{i,j}^{--}$ is for prefix paths ending by dropping both x_j and z_i.

The optimal alignment cost of matching $(z_1, \dots, z_i)$ and $(x_1, \dots, x_j)$ with Drop-DTW is then the minimum over the paths with the different types of endpoints, i.e., $D_{i,j} = \min\{D_{i,j}^{zx}, D_{i,j}^{-x}, D_{i,j}^{z-}, D_{i,j}^{--}\}$.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

Algorithm 1 Subsequence alignment with Drop-DTW.

Inputs: $C \in \mathbb{R}^{K \times N}$ (pairwise match cost matrix), d^z, d^x (drop costs for elements in X and Z respectively).
▷ *initializing DP tables for storing the optimal alignment path costs with different ends match conditions*
$D^{zx}_{0,0} = 0; D^{zx}_{i,0} = \infty; D^{zx}_{0,j} = \infty; \quad i \in [\![K]\!], j \in [\![N]\!]$ ▷ *matching ends in X and Z*
$D^{z-}_{0,0} = 0; D^{z-}_{i,0} = \infty; D^{z-}_{0,j} = \sum_{k=1}^{j} d^x_k; \quad i \in [\![K]\!], j \in [\![N]\!]$ ▷ *dropping X's end, but matching Z's end*
$D^{-x}_{0,0} = 0; D^{-x}_{i,0} = \sum_{k=1}^{i} d^z_k; D^{z-}_{0,j} = \infty; \quad i \in [\![K]\!], j \in [\![N]\!]$ ▷ *dropping Z's end, but matching X's end*
$D^{--}_{0,0} = 0; D^{--}_{i,0} = D^{x-}_{i,0}; D^{--}_{0,j} = D^{-z}_{0,j}; \quad i \in [\![K]\!], j \in [\![N]\!]$ ▷ *dropping ends in Z and X*

for $i = 1, \ldots, K$ **do** ▷ *iterating over elements in Z*
  for $j = 1, \ldots, N$ **do** ▷ *iterating over elements in X*
    ▷ *grouping costs into neighboring sets for convenience*
    diag_cells $= \{D^{zx}_{i-1,j-1}, D^{z-}_{i-1,j-1}, D^{-x}_{i-1,j-1}, D^{--}_{i-1,j-1}\}$
    left_cells_with_z $= \{D^{zx}_{i,j-1}, D^{z-}_{i,j-1}\}$
    top_cells_with_x $= \{D^{zx}_{i-1,j}, D^{-x}_{i-1,j}\}$
    left_cells_without_z $= \{D^{-x}_{i,j-1}, D^{--}_{i,j-1}\}$
    top_cells_without_x $= \{D^{z-}_{i-1,j}, D^{--}_{i-1,j}\}$
    ▷ *dynamic programming update on all tables*
    $D^{zx}_{i,j} = C_{i,j} + \min\{\text{diag\_cells} \cup \text{left\_cells\_with\_z} \cup \text{top\_cells\_with\_x}\}$ ▷ *consider matching z_i to x_j*
    $D^{z-}_{i,j} = d^x_j + \min\{\text{left\_cells\_with\_z}\}$ ▷ *consider dropping x_j*
    $D^{-x}_{i,j} = d^z_i + \min\{\text{top\_cells\_with\_x}\}$ ▷ *consider dropping z_i*
    $D^{--}_{i,j} = \min\{\text{top\_cells\_without\_x} \oplus d^z_i \cup \text{left\_cells\_without\_z} \oplus d^x_j\}$ ▷ *consider dropping x_j and z_i*
    $D_{i,j} = \min\{D^{zx}_{i,j}, D^{z-}_{i,j}, D^{-x}_{i,j}, D^{--}_{i,j}\}$ ▷ *select the optimal action*
  end for
end for
$M^* = \text{traceback}(D)$ ▷ *compute the optimal alignment by tracing back the minimum cost path*
Output: $D_{K,N}$, M^*

2.2 Asymmetric match costs

In cases where there is asymmetry between input sequences X and Z ingested by Drop-DTW, e.g., X is a video with outliers and Z is a sequence of (outlier-free) step labels contained in a video, we define an asymmetric match cost in Eq. 4 of the main paper. Expanding the softmax in Eq. 4, the asymmetric cost can be written as follows:

$$C^a_{i,j} = -\log\left(\frac{\exp(z_i^\top x_j)}{\sum_{z_k \in Z}\exp(z_k^\top x_j)}\right). \tag{1}$$

Note that our implementation slighly differs from the above formulation. Precisely, to avoid "over-smoothing" the softmax, we perform the summation in the denominator in Eq. 1 over the *unique* elements in Z only. This is implemented by performing the summation in the denominator over a different set of elements $z_k \in Z^+$, where Z^+ is obtained from Z by removing duplicate elements.

2.3 Training regularization for multi-step localization

In our experiments, we consider the Drop-DTW loss (i.e., Eq. 8 in the main paper) for video sequence labeling (i.e., Sec. 4.2 in the main paper), where a video sequence is aligned to a sequence of (discrete) labels from a finite set. An issue with this setting is that the alignments for minimizing the loss are prone to limiting correspondences to the most frequently occurring labels in the dataset.

To address this degeneracy, we augment our Drop-DTW loss with the following regularizer that promotes more uniform matching between the elements of the video sequence, $X \in \mathbb{R}^{N \times d}$, and the sequence of discrete labels, $Z \in \mathbb{R}^{K \times d}$:

$$\mathcal{L}_{\text{clust}} = ||I - \hat{X}Z^\top||_2, \tag{2}$$

where $I \in \mathbb{R}^{K \times K}$ is the identity matrix and $\hat{X} = (\hat{x}_1, \ldots, \hat{x}_K) \in \mathbb{R}^{K \times d}$. Each element $\hat{x}_i$ in $\hat{X}$ is defined according to

$$\hat{x}_i = \sum_{j=1}^{N} x_j \cdot \text{softmax}(X z_i / \gamma). \tag{3}$$

In other words, $\hat{x}_i$ in Eq. 3 defines attention-based pooling of sequence x, relative to an element z_i. Minimizing $\mathcal{L}_{\text{clust}}$ pushes every element in Z to have a unique match in X, which prevents overfitting to frequent labels in Z and encourages the clustering of the embeddings x_i around the appropriate label embeddings z_i.

3 Experiments details

3.1 Controlled synthetic experiments

In the main paper, we performed a controlled experiment using a synthetic dataset that we generated. Here, we provide a detailed description of this dataset and additional details of our retrieval experiments. In addition, we use this dataset here to further show the usage of Drop-DTW for subsequence localization in such controlled settings.

Synthetic dataset. We start from the MNIST dataset [1] and use it to generate videos of digits moving around an empty canvas, cf. [2], with just one digit per canvas. In particular, we place 28×28 MNIST images on a 64×64 black canvas, where an image can move along one of eight pre-defined trajectories. The trajectories are: (a) figure "8", (b) figure "∞", (c) circle "$\bigcirc$", all clockwise; trajectories (d), (e), (f) are obtained from (a), (b), and (c) but moving counter-clockwise; and finally (i) and (j) are the square diagonals ("/" and "\").

To synthesize a moving digit video, s, we perform the following four steps:

1. Choose a digit, d, from $[0, \ldots, 9]$ and a trajectory, t, from $[a, \ldots, j]$.
2. Sample a random image I of digit d from MNIST.
3. Choose a random video length. $T \in [30, \ldots, 50]$, and sample T equally-spaced 2D points $[p_1, \ldots, p_T]$ along the trajectory t.
4. Synthesize each frame, s_i, in the output video, s, by placing the digit image I onto the canvas at location p_i.

Note that since each video length, T, is randomly selected, the moving speed of the digits vary across the generated videos.

For each digit-trajectory pair, we follow the steps described above and generate two videos: (i) the digit executes a full trajectory and (ii) the digit performs a portion of the trajectory. We term videos falling under these two categories, TMNIST-full and TMNIST-part, respectively, where TMNIST stands for Trajectory-MNIST. Each dataset contains 80 (10 digits $\times$ 8 trajectories) video clips in total. *Please refer to the supplemental video for sample videos from our dataset as well as illustrations of use cases in the retrieval and localization scenarios.*

Encoding TMNIST. We independently encode the frames of each video in the TMNIST datasets using a shallow 2D ConvNet. In particular, we use a four-layer ConvNet architecture. Each layer consists of the following building blocks: conv3 $\times$ 3 $\rightarrow$ bn $\rightarrow$ relu $\rightarrow$ maxpool2 $\rightarrow$ dropout. The last layer eschews the local max pooling block in favor of a global average pooling followed by a linear layer. Going through the various layers of the network an input image of size $28 \times 28 \times 3$ undergoes the following transformations: $(28 \times 28 \times 3) \rightarrow (14 \times 14 \times 64) \rightarrow (7 \times 7 \times 128) \rightarrow (3 \times 3 \times 64) \rightarrow (3 \times 3 \times 32) \rightarrow (32) \rightarrow (10)$. This network is trained on the MNIST dataset on the task of digit recognition, thereby yielding a 99.53% accuracy.

In our experiments, the frame encodings are based on the last convolutional layer of the network, such that the shape of each frame encoding is $3 \times 3 \times 32$.

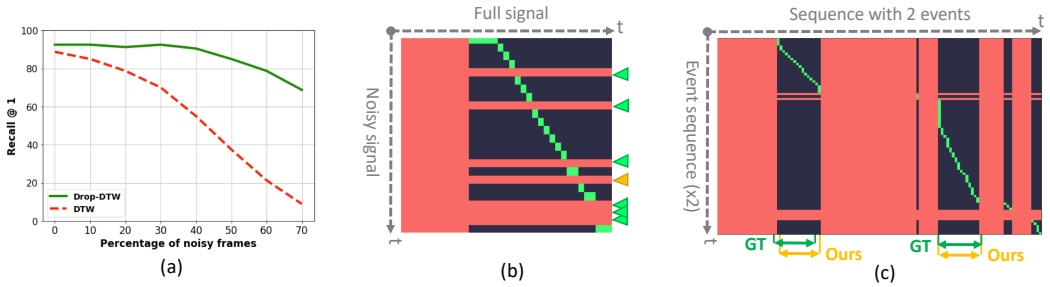

Figure 1: Drop-DTW for retrieval and localization on TMNIST. We consider queries, Z, from TMNIST-part with interspersed noise, for indexing into TMNIST-full. We show that Drop-DTW is more robust to interspersed noise. (a) We consider queries, Z, from TMNIST-part with interspersed noise, for indexing into TMNIST-full. Drop-DTW is more robust to interspersed noise than other alignment algorithms. (b) An example query-signal correspondence matrix illustrates outlier removal, where red rows (cols) depict dropped frames in Z (X, resp.), and green (yellow) arrows denote correct (incorrect) query outlier identification. (c) Replicating the query and dataset signal in time demonstrates temporal localization by Drop-DTW despite signal repetitions and interspersed outliers (see main text).

TMNIST for retrieval. In this experiment (i.e., Sec 4.1 in the main paper), we use videos from TMNIST-part as queries and look for the most similar videos in TMNIST-full. To demonstrate Drop-DTW's robustness to noise we introduce noise to the query videos. In particular, we randomly *blur* $p\%$ of the frames in the query. This blur is achieved by convolving the randomly selected frames with a Gaussian kernel of radius two.

TMNIST for localization. To demonstrate Drop-DTW's ability to identify start and end times we consider test sequences, X, formed by concatenating M clips from TMNIST-part, where exactly $N < M$ of these clips are from a selected digit-trajectory class. An example of a test sequence X is obtained by concatenating the sequences: [0, b] $\rightarrow$ [3, c] $\rightarrow$ [8, a] $\rightarrow$ [3, c] $\rightarrow$ [6, j], where each subsequence represents a [digit, trajectory] pair. Given a target sequence Z (e.g., [3, c]), the task is to find all time intervals corresponding to the event [3, c] in sequence X. For this purpose, we consider queries of the form $\hat{Z} = [Z^{(1)}, \ldots, Z^{(N)}]$, where each of the $Z^{(n)}$'s are instances of this same digit-trajectory class (i.e., [3, c]) in TMNIST-full.

Ideally, aligning $\hat{Z}$ and X should put a subsequence of each of the $Z^{(n)}$'s in correspondence with a matching subsequence in X (see Fig. 1 (c)). By construction, there should be N such matches, interspersed with outliers. Here, we use $N = 2$, $M \in \{5, \ldots, 7\}$, and randomly generate 1000 such sequences, X.

We use Drop-DTW to align the generated pairs, $\hat{Z}$ and X, and select the longest N contiguously matched subsequences. We recover start and end times as the endpoints of these N longest matched intervals. We find the localization accuracy on this data is $97.1 \pm 0.28\%$ and IoU is $91.2 \pm 0.71\%$ (see metrics definition in Sec. 4.2 of the main paper). These results demonstrate the efficacy of Drop-DTW for localizing events in long untrimmed videos. Note that we do not compare our method to any baseline as Drop-DTW is the only alignment-based method capable of solving this task.

3.2 Complete ablation study in instructional video step localization

Role of the regularization loss. Here, we investigate the role of each component of the proposed loss for step localization on CrossTask [3], COIN [4] and YouCook2 [5] dataset. We train the video embeddings with various alignment losses, with or without regularization, and compare with the pre-trained embeddings. At inference time, we use the learned embeddings to calculate pairwise similarities between all video clips and steps. From these similarities we obtain match and drop costs that are then used to assign a step label to each video clip.

As seen in Table 1, using the clustering loss alone on top of the pre-trained embeddings provides a strong baseline, as it effectively uses an unordered set of steps to learn the video representations. Interestingly, combining SmoothDTW with $\mathcal{L}_{\text{clust}}$ to introduce order information in training does not improve (and can decrease) performance. We conjecture that the outliers present in the sequences can negatively impact the training signal. In contrast, training with Drop-DTW yields a significant boost

Table 1: Ablation study of training and inference methods for step localization on CrossTask [3], COIN [4] and (right) YouCook2 [5]. The first column describes the loss function(s) used for training, while the first row gives the dataset. The second row specifies the measured metric (higher is better), and the rest of the table reports the results. Here, CT (in row 1), corresponds to the evaluation algorithm provided by CrossTask. In column 1, * indicates our re-implementation.

Method	CrossTask			COIN			YouCook2		
	Recall	Acc.	IoU	Recall	Acc.	IoU	Recall	Acc.	IoU
MIL-NCE* [6]	39.1	66.9	20.9	33.0	50.2	23.3	70.7	63.7	43.7
SmoothDTW [7]	29.6	48.5	9.0	29.1	38.8	17.8	70.2	61.1	39.7
Drop-DTW	33.8	56.6	14.2	30.6	43.7	20.1	71.5	63.4	41.6
$\mathcal{L}_{\text{clust}}$	43.4	70.2	29.8	37.7	53.6	27.6	75.8	66.3	47.3
SmoothDTW + $\mathcal{L}_{\text{clust}}$	43.1	70.2	30.5	37.7	52.7	27.7	75.3	66.0	47.5
Drop-DTW + $\mathcal{L}_{\text{clust}}$	**48.2**	**73.5**	**34.4**	**40.8**	**54.8**	**29.5**	**77.4**	**68.4**	**49.4**

Table 2: Ablation study on the role of the min operator used in our Drop-DTW algorithm.

	CrossTask		
	Recall	Acc.	IoU
SoftMin [8]	45.9	71.8	32.9
Hard min	47.8	72.2	34.3
SmoothMin [7]	**48.2**	**73.5**	**34.4**

in performance on all the metrics. This suggests that filtering out the background is key to training with step order supervision on instructional videos.

Role of the min operator. In the main paper, we use the SmoothMin approximation of the min operator proposed in [7] to enable differentiability. This is not a key component of our contribution and other differentiable min operators such as SoftMin [8] and the (hard) min operator can be used as well. Corresponding results, shown in Table 2, highlight the stability of Drop-DTW across min operators.

Role of the percentile choice in the drop cost. When training representations with Drop-DTW, the matching costs obtained by the model evolve during the training. To have the drop cost change accordingly, we define it as a percentile of the matching costs, in Eq. 5 of the main paper. Here, we provide results of setting the drop cost to various percentile values. We also include comparison to setting the drop cost as learned component as defined in Eq. 6 of the main paper.

The results provided in Table 3, demonstrate the robustness of the proposed Drop-DTW algorithm for various choices of the percentile. Interestingly, the learned drop cost yields the best overall performance, which demonstrates the adaptability of the Drop-DTW algorithm.

3.3 Drop-DTW for representation learning

As mentioned in the main paper, we use the PennAction dataset [9] to evaluate Drop-DTW for representation learning. To evaluate the robustness to outliers, we contaminate PennAction sequences with $p\%$ interspersed outlier frames. Here, we give more details on the contamination and alignment procedures.

Table 3: Ablation study on the role of the percentile used in the drop cost definition of Drop-DTW.

	CrossTask		
	Recall	Acc	IoU
p = 0.1	47.1	72.5	34.9
p = 0.3	48.2	**73.5**	34.4
p = 0.5	46.2	71.5	32.0
p = 0.7	45.3	71.4	32.0
p = 0.9	44.6	71.7	31.0
learned drop-cost	**49.1**	71.5	**34.5**

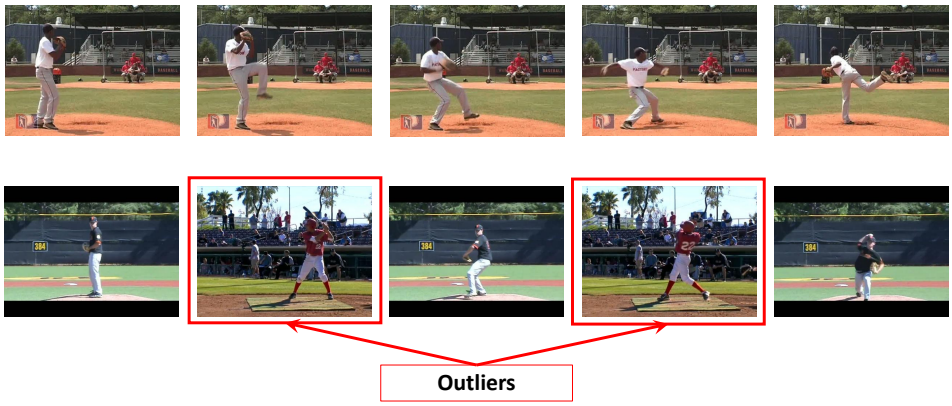

Figure 2: Illustration of the PennAction outlier contamination process. Sequence X (top) depicts a clean baseball pitch sequence, while sequence $\hat{Y}$ (bottom) has outlier frames from a baseball swing interspersed within the second baseball pitch sequence Y.

PennAction contamination. At training time, a batch contains a set of paired sequences. Given two such sequences, X and Y, of the same action (e.g., baseball pitch), we extract, $N = 20$, frames from both sequences. To ensure that the extracted frames cover the entire duration of both sequences, we rely on strided sampling. We then select $p\%$ outlier frames from another action (e.g., baseball swing) and we randomly intersperse them within sequence Y, thereby yielding a sequence, $\hat{Y}$, containing outlier elements. We otherwise leave sequence X untouched. This process is illustrated in Fig. 2. Under these settings, Drop-DTW is expected to learn strong representations that rely on the common signal between X and $\hat{Y}$, while ignoring the interspersed outliers. The results reported in Fig. 5 in the main paper support this intuition and speak decisively in favor of Drop-DTW.

3.4 Drop-DTW for audio-visual localization

We report results on the task of audio-visual localization in the main paper. Here, we provide further details on the training procedure. Given an audio-visual pair, (X, Z), from the AVE dataset [10], we start by splitting each modality into consecutive one second long segments and encode each segment using the same backbones used in the original paper [10] for each modality. We then calculate a pairwise cost matrix between the sequences of the two modalities, using the symmetric match cost defined in Eq. 4 in the main paper. Next, we use a DTW-based approach to obtain the optimal match cost. In the case of Drop-DTW, the optimal match cost corresponds to $D_{K,N}$ in Algorithm 1 and we use 70%-percentile drop costs defined in Eq. 5 of the main paper. To train the networks for cross-modal localization we use a margin loss defined as:

$$\mathcal{L}_{\text{marg}}(X, Z, \hat{Z}) = \max(D(X, Z)_{K,N} - D(X, \hat{Z})_{K,N} + \beta, 0), \tag{4}$$

where (X, Z) represent an audio-visual (visual-audio) pair from the same sequence, whereas the visual (audio) signal $\hat{Z}$ is from a different sequence. β is set to 0.5 in all our experiments.

Once we have learned representations using Eq. 4, we strictly follow the experimental protocol from [10].

3.5 Additional qualitative results

In Sec. 4.2 of the main paper, we demonstrate the ability of Drop-DTW to tackle multi-step localization and compare it to alternative alignment-based approaches. In Fig. 4 of the main paper, we provide a qualitative comparison between various alignment-based approaches when each algorithm is used both at training and inference time. Here, we provide more such qualitative results in Fig. 3. Collectively, these results demonstrate Drop-DTW's unique capability to locate subsequences with interspersed outliers. Moreover, we show that Drop-DTW is versatile to also handle situations with no interspersed outliers (e.g., see the *Make French Toast* example in Fig. 3). In addition, in Fig. 4, we provide qualitative results showing the advantage of using Drop-DTW at inference time even on representations learned with other alignment-based approaches. From these last results we show

Figure 3: Step localization with DTW variants used for training and inference. In each panel, rows two to four show step assignment results when the same alignment method is used for training and inference. Drop-DTW allows to identify interspersed unlabelled clips and much more closely approximates the ground truth.

that Drop-DTW is a valuable inference time tool for subsequence localization. *A visual demo of the subsequence localization application is provided in the supplemental video.*

Figure 4: Step localization with DTW variants used for training *only*, while always using Drop-DTW for inference. In each panel, rows two to four show in each sub figure show step assignment results when different alignment methods are used for training but Drop-DTW is used for inference in all cases. In the top part of the figure (highlighted in green) we illustrate scenarios where using Drop-DTW both during training and inference is more beneficial, while the two bottom examples (highlighted in red) do not show clear advantage of Drop-DTW at training time but clearly show the benefit of using it at inference time in all cases. Qualitative results in this figure correspond to quantitative results in Table 2 of the main paper.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 1998.

[2] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using LSTMs," in *International Conference on Machine Learning (ICML)*, 2015.

[3] D. Zhukov, J.-B. Alayrac, R. G. Cinbis, D. Fouhey, I. Laptev, and J. Sivic, "Cross-task weakly supervised learning from instructional videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[4] Y. Tang, D. Ding, Y. Rao, Y. Zheng, D. Zhang, L. Zhao, J. Lu, and J. Zhou, "COIN: A large-scale dataset for comprehensive instructional video analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[5] L. Zhou, C. Xu, and J. J. Corso, "Towards automatic learning of procedures from web instructional videos," in *AAAI Conference on Artificial Intelligence*, 2018.

[6] A. Miech, J.-B. Alayrac, L. Smaira, I. Laptev, J. Sivic, and A. Zisserman, "End-to-End Learning of Visual Representations from Uncurated Instructional Videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[7] I. Hadji, K. G. Derpanis, and A. D. Jepson, "Representation learning via global temporal alignment and cycle-consistency," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[8] M. Cuturi and M. Blondel, "Soft-DTW: A differentiable loss function for time-series," in *International Conference on Machine Learning (ICML)*, 2017.

[9] W. Zhang, M. Zhu, and K. G. Derpanis, "From actemes to action: A strongly-supervised representation for detailed action understanding," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2013.

[10] Y. Tian, J. Shi, B. Li, Z. Duan, and C. Xu, "Audio-visual event localization in unconstrained videos," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.