

# Representation Report

## Temporal Cycle-Consistency Learning

Debidatta Dwibedi<sup>1</sup>, Yusuf Aytar<sup>2</sup>, Jonathan Tompson<sup>1</sup>, Pierre Sermanet<sup>1</sup>, and Andrew Zisserman<sup>2</sup>

<sup>1</sup> Google Brain <sup>2</sup> DeepMind

{debidattha, yusufaytar, tompson, sermanet, zisserman}@google.com



**Figure 1:** We present a self-supervised representation learning technique called temporal cycle consistency (TCC) learning. It is inspired by the temporal video alignment problem, which refers to the task of finding correspondences across multiple videos despite many factors of variation. The learned representations are useful for fine-grained temporal understanding in videos. Additionally, we can now align multiple videos by simply finding nearest-neighbor frames in the embedding space.

### Abstract

We introduce a self-supervised representation learning method based on the task of temporal alignment between videos. The method trains a network using temporal cycle-consistency (TCC), a differentiable cycle-consistency loss that can be used to find correspondences across time in multiple videos. The resulting per-frame embeddings can be used to align videos by simply matching frames using nearest-neighbors in the learned embedding space.

To evaluate the power of the embeddings, we densely label the Pouring and Penn Action video datasets for action phases. We show that (i) the learned embeddings enable few-shot classification of these action phases, significantly reducing the supervised training requirements; and (ii) TCC is complementary to other methods of self-supervised learning in videos, such as Shuffle and Learn and Time-Contrastive Networks. The embeddings are also used for a number of applications based on alignment (dense temporal correspondence) between video pairs, including transfer of metadata of synchronized modalities between videos (sounds, temporal semantic labels), synchronized playback of multiple videos, and anomaly detection. Project webpage: <https://sites.google.com/view/temporal-cycle-consistency>.

### 1. Introduction

The world presents us with abundant examples of sequential processes. A plant growing from a seedling to a tree, the daily routine of getting up, going to work and com-

ing back home, or a person pouring themselves a glass of water – are all examples of events that happen in a particular order. Videos capturing such processes not only contain information about the causal nature of these events, but also provide us with a valuable signal – the possibility of temporal correspondences lurking across multiple instances of the same process. For example, during pouring, one could be reaching for a teapot, a bottle of wine, or a glass of water to pour from. Key moments such as the first touch to the container or the container being lifted from the ground are common to all pouring sequences. These correspondences, which exist in spite of many varying factors like visual changes in viewpoint, scale, container style, the speed of the event, etc., could serve as the link between raw video sequences and high-level temporal abstractions (e.g. phases of actions). In this work we present evidence that suggests the very act of *looking for correspondences* in sequential data enables the learning of rich and useful representations, particularly suited for fine-grained temporal understanding of videos.

Temporal reasoning in videos, understanding multiple stages of a process and causal relations between them, is a relatively less studied problem compared to recognizing action categories [10, 42]. Learning representations that can differentiate between states of objects as an action proceeds is critical for perceiving and acting in the world. It would be desirable for a robot tasked with learning to pour drinks to understand each intermediate state of the world as it proceeds with performing the task. Although videos are a rich source of sequential data essential to understanding such state changes, their true potential remains largely un-

tapped. One hindrance in the fine-grained temporal understanding of videos can be an excessive dependence on pure supervised learning methods that require per-frame annotations. It is not only difficult to get every frame labeled in a video because of the manual effort involved, but also it is not entirely clear what are the exhaustive set of labels that need to be collected for fine-grained understanding of videos. Alternatively, we explore self-supervised learning of correspondences between videos across time. We show that the emerging features have strong temporal reasoning capacity, which is demonstrated through tasks such as action phase classification and tracking the progress of an action.

When frame-by-frame alignment (i.e. supervision) is available, learning correspondences reduces to learning a common embedding space from pairs of aligned frames (e.g. CCA [3, 4] and ranking loss [35]). However, for most of the real world sequences such frame-by-frame alignment does not exist naturally. One option would be to artificially obtain aligned sequences by recording the same event through multiple cameras [30, 35, 37]. Such data collection methods might find it difficult to capture all the variations present naturally in videos in the wild. On the other hand, our self-supervised objective does not need explicit correspondences to align different sequences. It can align significant variations within an action category (e.g. pouring liquids, or baseball pitch). Interestingly, the embeddings that emerge from learning the alignment prove to be useful for fine-grained temporal understanding of videos. More specifically, we learn an embedding space that maximizes one-to-one mappings (i.e. cycle-consistent points) across pairs of video sequences within an action category. In order to do that, we introduce two differentiable versions of cycle consistency computation which can be optimized by conventional gradient-based optimization methods. Further details of the method will be explained in section 3.

The main contribution of this paper is a new self-supervised training method, referred to as temporal cycle consistency (TCC) learning, that learns representations by aligning video sequences of the same action. We compare TCC representations against features from existing self-supervised video representation methods [27, 35] and supervised learning, for the tasks of action phase classification and continuous progress tracking of an action. Our approach provides significant performance boosts when there is a lack of labeled data. We also collect per-frame annotations of Penn Action [52] and Pouring [35] datasets that we will release publicly to facilitate evaluation of fine-grained video understanding tasks.

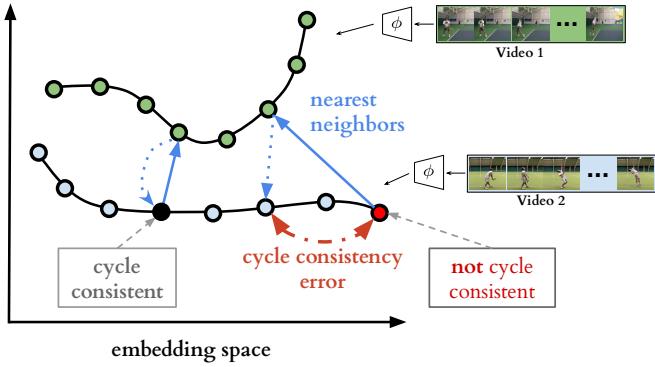
## 2. Related Work

**Cycle consistency.** Validating good matches by cycling between two or more samples is a commonly used technique in computer vision. It has been applied successfully

for tasks like co-segmentation [43, 44], structure from motion [49, 51], and image matching [54, 55, 56]. For instance, FlowWeb [54] optimizes globally-consistent dense correspondences using the cycle consistent flow fields between all pairs of images in a collection, whereas Zhou et al. [56] approaches a similar task by formulating it as a low-rank matrix recovery problem and solves it through fast alternating minimization. These methods learn robust dense correspondences on top of fixed feature representations (e.g. SIFT, deep features, etc.) by enforcing cycle consistency and/or spatial constraints between the images. Our method differs from these approaches in that TCC is a self-supervised representation learning method which learns embedding spaces that are optimized to give good correspondences. Furthermore we address a temporal correspondence problem rather than a spatial one. Zhou et al. [55] learn to align multiple images using the supervision from 3D guided cycle-consistency by leveraging the initial correspondences that are available between multiple renderings of a 3D model, whereas we don't assume any given correspondences. Another way of using cyclic relations is to directly learn bi-directional transformation functions between multiple spaces such as CycleGANs [57] for learning image transformations, and CyCADA [21] for domain adaptation. Unlike these approaches we don't have multiple domains, and we can't learn transformation functions between all pairs of sequences. Instead we learn a joint embedding space in which the Euclidean distance defines the mapping across the frames of multiple sequences. Similar to us, Aytar et al. [7] applies cycle-consistency between temporal sequences, however they use it as a validation tool for hyper-parameter optimization of learned representations for the end goal of imitation learning. Unlike our approach, their cycle-consistency measure is non-differentiable and hence can't be directly used for representation learning.

**Video alignment.** When we have synchronization information (e.g. multiple cameras recording the same event) then learning a mapping between multiple video sequences can be accomplished by using existing methods such as Canonical Correlation Analysis (CCA) [3, 4], ranking [35] or match-classification [6] objectives. For instance TCN [35] and circulant temporal encoding [30] align multiple views of the same event, whereas Sigurdsson et al. [37] learns to align first and third person videos. Although we have a similar objective, these methods are not suitable for our task as we cannot assume any given correspondences between different videos.

**Action localization and parsing.** As action recognition is quite popular in the computer vision community, many studies [17, 38, 46, 50, 53] explore efficient deep architectures for action recognition and localization in videos. Past work has also explored parsing of fine-grained actions in videos [24, 25, 29] while some others [13, 33, 34, 36] discover sub-activities without explicit supervision of temporal boundaries. [20] learns a supervised regression model with



**Figure 2: Cycle-consistent representation learning.** We show two example video sequences encoded in an example embedding space. If we use nearest neighbors for matching, one point (shown in black) is cycling back to itself while another one (shown in red) is not. Our target is to learn an embedding space where maximum number of points can cycle back to themselves. We achieve it by minimizing the cycle consistency error (shown in red dotted line) for each point in every pair of sequences.

voting to predict the completion of an action, and [2] discovers key events in an unsupervised manner using a weak association between videos and text instructions. However all these methods heavily rely on existing deep image [19, 39] or spatio-temporal [45] features, whereas we learn our representation from scratch using raw video sequences. **Soft nearest neighbours.** The differentiable or soft formulation for nearest-neighbors is a commonly known method [18]. This formulation has recently found application in metric learning for few-shot learning [28, 31, 40]. We also make use of soft nearest neighbor formulation as a component in our differentiable cycle-consistency computation.

**Self-supervised representations.** There has been significant progress in learning from images and videos without requiring class or temporal segmentation labels. Instead of labels, self-supervised learning methods use signals such as temporal order [16, 27], consistency across viewpoints and/or temporal neighbors [35], classifying arbitrary temporal segments [22], temporal distance classification within or across modalities [7], spatial permutation of patches [5, 14], visual similarity [32] or a combination of such signals [15]. While most of these approaches optimize each sample independently, TCC jointly optimizes over two sequences at a time, potentially capturing more variations in the embedding space. Additionally, we show that TCC yields best results when combined with some of the unsupervised losses above.

### 3. Cycle Consistent Representation Learning

The core contribution of this work is a self-supervised approach to learn an embedding space where two similar video sequences can be aligned temporally. More specific-

ally, we intend to maximize the number of points that can be mapped one-to-one between two sequences by using the minimum distance in the learned embedding space. We can achieve such an objective by maximizing the number of cycle-consistent frames between two sequences (see Figure 2). However, cycle-consistency computation is typically not a differentiable procedure. In order to facilitate learning such an embedding space using back-propagation, we introduce two differentiable versions of the *cycle-consistency loss*, which we describe in detail below.

Given any frame  $s_i$  in a sequence  $S = \{s_1, s_2, \dots, s_N\}$ , the embedding is computed as  $u_i = \phi(s_i; \theta)$ , where  $\phi$  is the neural network encoder parameterized by  $\theta$ . For the following sections, assume we are given two video sequences  $S$  and  $T$ , with lengths  $N$  and  $M$ , respectively. Their embeddings are computed as  $U = \{u_1, u_2, \dots, u_N\}$  and  $V = \{v_1, v_2, \dots, v_M\}$  such that  $u_i = \phi(s_i; \theta)$  and  $v_i = \phi(t_i; \theta)$ .

#### 3.1. Cycle-consistency

In order to check if a point  $u_i \in U$  is cycle consistent, we first determine its nearest neighbor,  $v_j = \arg \min_{v \in V} \|u_i - v\|$ . We then repeat the process to find the nearest neighbor of  $v_j$  in  $U$ , i.e.  $u_k = \arg \min_{u \in U} \|v_j - u\|$ . The point  $u_i$  is *cycle-consistent* if and only if  $i = k$ , in other words if the point  $u_i$  cycles back to itself. Figure 2 provides positive and negative examples of cycle consistent points in an embedding space. We can learn a good embedding space by maximizing the number of cycle-consistent points for any pair of sequences. However that would require a differentiable version of cycle-consistency measure, two of which we introduce below.

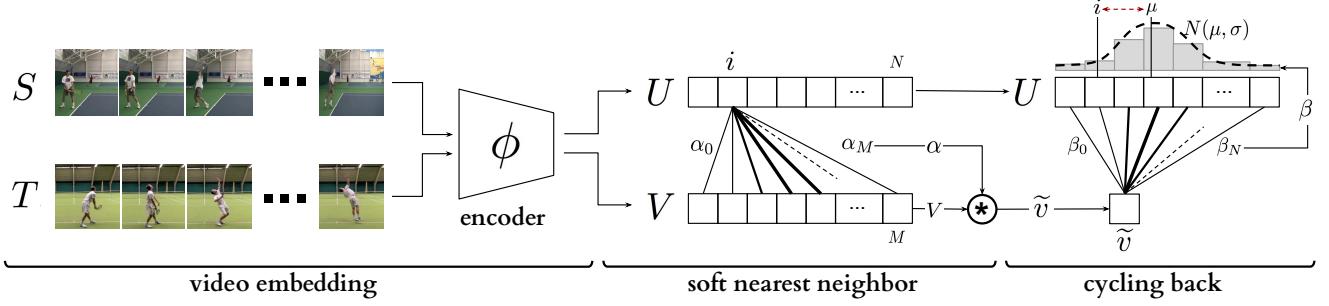
#### 3.2. Cycle-back Classification

We first compute the soft nearest neighbor  $\tilde{v}$  of  $u_i$  in  $V$ , then figure out the nearest neighbor of  $\tilde{v}$  back in  $U$ . We consider each frame in the first sequence  $U$  to be a separate class and our task of checking for cycle-consistency reduces to classification of the nearest neighbor correctly. The logits are calculated using the distances between  $\tilde{v}$  and any  $u_k \in U$ , and the ground truth label  $y$  are all zeros except for the  $i^{th}$  index which is set to 1.

For the selected point  $u_i$ , we use the softmax function to define its soft nearest neighbor  $\tilde{v}$  as:

$$\tilde{v} = \sum_j^M \alpha_j v_j, \quad \text{where } \alpha_j = \frac{e^{-\|u_i - v_j\|^2}}{\sum_k^M e^{-\|u_i - v_k\|^2}} \quad (1)$$

and  $\alpha$  is the the similarity distribution which signifies the proximity between  $u_i$  and each  $v_j \in V$ . And then we solve the  $N$  class (i.e. number of frames in  $U$ ) classification problem where the logits are  $x_k = -\|\tilde{v} - u_k\|^2$  and the predicted labels are  $\hat{y} = \text{softmax}(x)$ . Finally we optimize the cross-



**Figure 3: Temporal cycle consistency.** The embedding sequences  $U$  and  $V$  are obtained by encoding video sequences  $S$  and  $T$  with the encoder network  $\phi$ , respectively. For the selected point  $u_i$  in  $U$ , soft nearest neighbor computation and cycling back to  $U$  again is demonstrated visually. Finally the normalized distance between the index  $i$  and cycling back distribution  $N(\mu, \sigma^2)$  (which is fitted to  $\beta$ ) is minimized.

entropy loss as follows:

$$L_{cbc} = - \sum_j^N y_j \log(\hat{y}_j) \quad (2)$$

### 3.3. Cycle-back Regression

Although cycle-back classification defines a differentiable cycle-consistency loss function, it has no notion of how close or far in time the point to which we cycled back is. We want to penalize the model less if we are able to cycle back to closer neighbors as opposed to the other frames that are farther away in time. In order to incorporate temporal proximity in our loss, we introduce cycle-back regression. A visual description of the entire process is shown in Figure 3. Similar to the previous method first we compute the soft nearest neighbor  $\tilde{v}$  of  $u_i$  in  $V$ . Then we compute the similarity vector  $\beta$  that defines the proximity between  $\tilde{v}$  and each  $u_k \in U$  as:

$$\beta_k = \frac{e^{-\|\tilde{v} - u_k\|^2}}{\sum_j^N e^{-\|\tilde{v} - u_j\|^2}} \quad (3)$$

Note that  $\beta$  is a discrete distribution of similarities over time and we expect it to show a peaky behavior around the  $i^{th}$  index in time. Therefore, we impose a Gaussian prior on  $\beta$  by minimizing the normalized squared distance  $\frac{|i - \mu|^2}{\sigma^2}$  as our objective. We enforce  $\beta$  to be more peaky around  $i$  by applying additional variance regularization. We define our final objective as:

$$L_{cbr} = \frac{|i - \mu|^2}{\sigma^2} + \lambda \log(\sigma) \quad (4)$$

where  $\mu = \sum_k^N \beta_k * k$  and  $\sigma^2 = \sum_k^N \beta_k * (k - \mu)^2$ , and  $\lambda$  is the regularization weight. Note that we minimize the log of variance as using just the variance is more prone to numerical instabilities. All these formulations are differentiable and can conveniently be optimized with conventional back-propagation.

Operations	Output Size	Parameters
Temporal Stacking	$k \times 14 \times 14 \times c$	Stack $k$ context frames
3D Convolutions	$k \times 14 \times 14 \times 512$	$[3 \times 3 \times 3, 512] \times 2$
Spatio-temporal Pooling	512	Global 3D Max-pooling
Fully-connected layers	512	$[512] \times 2$
Linear projection	128	128

**Table 1:** Architecture of the embedding network.

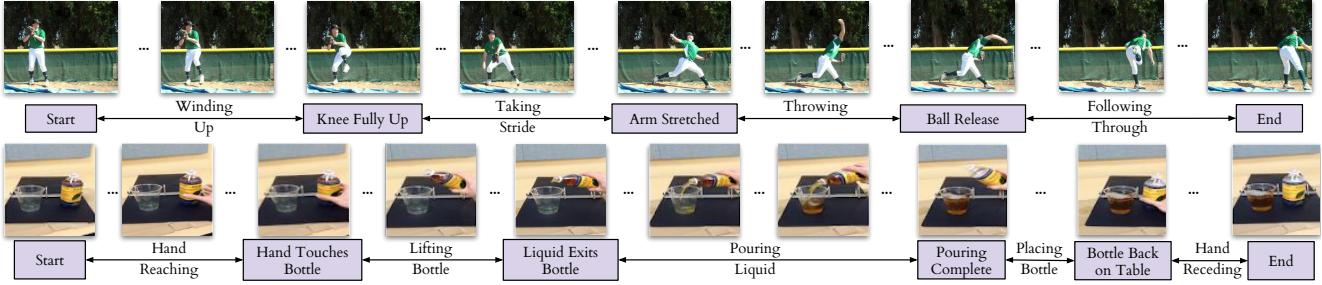
### 3.4. Implementation details

**Training Procedure.** Our self-supervised representation is learned by minimizing the cycle-consistency loss for all the pair of sequences in the training set. Given a sequence pair, their frames are embedded using the encoder network and we optimize cycle consistency losses for randomly selected frames within each sequence until convergence. We used Tensorflow [1] for all our experiments.

**Encoding Network.** All the frames in a given video sequence are resized to  $224 \times 224$ . When using ImageNet pretrained features, we use ResNet-50 [19] architecture to extract features from the output of *Conv4c* layer. The size of the extracted convolutional features are  $14 \times 14 \times 1024$ . Because of the size of the datasets, when training from scratch we use a smaller model along the lines of VGG-M [11]. This network takes input at the same resolution as ResNet-50 but is only 7 layers deep. The convolutional features produced by this base network are of the size  $14 \times 14 \times 512$ . These features are provided as input to our embedder network (presented in Table 1). We stack the features of any given frame and its  $k$  context frames along the dimension of time. This is followed by 3D convolutions for aggregating temporal information. We reduce the dimensionality by using 3D max-pooling followed by two fully connected layers. Finally, we use a linear projection to get a 128-dimensional embedding for each frame. More details of the architecture are presented in the supplementary material.

## 4. Datasets and Evaluation

We validate the usefulness of our representation learning technique on two datasets: (i) *Pouring* [35]; and (ii)



**Figure 4:** Example labels for the actions ‘Baseball Pitch’ (top row) and ‘Pouring’ (bottom row). The key events are shown in boxes below the frame (e.g. ‘Hand touches bottle’), and each frame in between two key events has a phase label (e.g. ‘Lifting bottle’).

*Penn Action* [52]. These datasets both contain videos of humans performing actions, and provide us with collections of videos where dense alignment can be performed. While *Pouring* focuses more on the objects being interacted with, *Penn Action* focuses on humans doing sports or exercise.

**Annotations.** For evaluation purposes, we add two types of labels to the video frames of these datasets: key events and phases. Densely labeling each frame in a video is a difficult and time-consuming task. Labelling only *key events* both reduces the number of frames that need to be annotated, and also reduces the ambiguity of the task (and thus the disagreement between annotators). For example, annotators agree more about the frame when the golf club hits the ball (a key event) than when the golf club is at a certain angle. The *phase* is the period between two key events, and all frames in the period have the same phase label. It is similar to tasks proposed in [9, 12, 23]. Examples of key events and phases are shown in Figure 4, and Table 2 gives the complete list for all the actions we consider.

We use all the real videos from the *Pouring* dataset, and all but two action categories in *Penn Action*. We do not use *Strumming guitar* and *Jumping rope* because it is difficult to define unambiguous key events for these. We use the train/val splits of the original datasets [35, 52]. We will publicly release these new annotations.

## 4.1. Evaluation

We use three evaluation measures computed on the validation set. These metrics evaluate the model on fine-grained temporal understanding of a given action. Note, the networks are first trained on the training set and then frozen. SVM classifiers and linear regressors are trained on the features from the networks, with no additional fine-tuning of the networks. For all measures a higher score implies a better model.

**1. Phase classification accuracy:** is the per frame phase classification accuracy. This is implemented by training a SVM classifier on the phase labels for each frame of the training data.

**2. Phase progression:** measures how well the *progress*

of a process or action is captured by the embeddings. We first define an approximate measure of progress through a phase as the difference in time-stamps between any given frame and each key event. This is normalized by the number of frames present in that video. Similar definitions can be found in recent literature [8, 20, 26]. We use a linear regressor on the features to predict the phase progression values. It is computed as the average *R-squared* measure (coefficient of determination) [47], given by:

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

where  $y_i$  is the ground truth event progress value,  $\bar{y}$  is the mean of all  $y_i$  and  $\hat{y}_i$  is the prediction made by the linear regression model. The maximum value of this measure is 1.

**3. Kendall’s Tau** [48]: is a statistical measure that can determine how well-aligned two sequences are in time. Unlike the above two measures it does not require additional labels for evaluation. Kendall’s Tau is calculated over every pair of frames in a pair of videos by sampling a pair of frames  $(u_i, u_j)$  in the first video (which has  $n$  frames) and retrieving the corresponding nearest frames in the second video,  $(v_p, v_q)$ . This quadruplet of frame indices  $(i, j, p, q)$  is said to be *concordant* if  $i < j$  and  $p < q$  or  $i > j$  and  $p > q$ . Otherwise it is said to be *discordant*. Kendall’s Tau is defined over all pairs of frames in the first video as:

$$\tau = \frac{(\text{no. of concordant pairs} - \text{no. of discordant pairs})}{\frac{n(n-1)}{2}}$$

We refer the reader to [48] to check out the complete definition. The reported metric is the average Kendall’s Tau over all pairs of videos in the validation set. It is a measure of how well the learned representations generalize to aligning unseen sequences if we used nearest neighbour matching for aligning a pair of videos. A value of 1 implies the videos are perfectly aligned while a value of -1 implies the videos are aligned in the reverse order. One drawback of Kendall’s tau is that it assumes there are no repetitive frames in a video. This might not be the case if an action is being done slowly or if there is periodic motion. For the datasets we consider, this drawback is not a problem.

Action	Number of phases	List of Key Events	Train set size	Val set size
Baseball Pitch	4	Knee fully up, Arm fully stretched out, Ball release	103	63
Baseball Swing	3	Bat swung back fully, Bat hits ball	113	57
Bench-press	2	Bar fully down	69	71
Bowling	3	Ball swung fully back, Ball release	134	85
Clean and jerk	6	Bar at hip, Fully squatting, Standing, Begin Thrusting, Beginning Balance	40	42
Golf swing	3	Stick swung fully back, Stick hits ball	87	77
Jumping jacks	4	Hands at shoulder (going up), Hands above head, Hands at shoulders (going down)	56	56
Pullups	2	Chin above bar	98	101
Pushups	2	Head at floor	102	105
Situps	2	Abs fully crunched	50	50
Squats	4	Hips at knees (going down), Hips at floor, Hips at knee (going up)	114	116
Tennis forehand	3	Racket swung fully back, Racket touches ball	79	74
Tennis serve	4	Ball released from hand, Racket swung fully back, Ball touches racket	115	69
Pouring	5	Hand touches bottle, Liquid starts exiting, Pouring complete, Bottle back on table	70	14

**Table 2:** List of all key events in each dataset. Note that each action has a *Start* event and *End* event in addition to the key events above.

## 5. Experiments

### 5.1. Baselines

We compare our representations with existing self-supervised video representation learning methods. For completeness, we briefly describe the baselines below but recommend referring to the original papers for more details.

**Shuffle and Learn (SaL) [27].** We randomly sample triplets of frames in the manner suggested by [27]. We train a small classifier to predict if the frames are in order or shuffled. The labels for training this classifier are derived from the indices of the triplet we sampled. This loss encourages the representations to encode information about the order in which an action should be performed.

**Time-Contrastive Networks (TCN) [35].** We sample  $n$  frames from the sequence and use these as anchors (as defined in the metric learning literature). For each anchor, we sample positives within a fixed time window. This gives us  $n$ -pairs of anchors and positives. We use the  $n$ -pairs loss [41] to learn our embedding space. For any particular pair, the  $n$ -pairs loss considers all the other pairs as negatives. This loss encourages representations to be disentangled in time while still adhering to metric constraints.

**Combined Losses.** In addition to these baselines, we can combine our cycle consistency loss with both SaL and TCN to get two more training methods: TCC+SaL and TCC+TCN. We learn the embedding by computing both losses and adding them in a weighted manner to get the total loss, based on which the gradients are calculated. The weights are selected by performing a search over 3 values 0.25, 0.5, 0.75. All baselines share the same video encoder architecture, as described in section 3.4.

### 5.2. Ablation of Different Cycle Consistency Losses

We ran an experiment on the Pouring dataset to see how the different losses compare against each other. We also report metrics on the Mean Squared Error (MSE) version of the cycle-back regression loss (Equation 4) which is formulated by only minimizing  $|i - \mu|^2$ , ignoring the variance of

predictions altogether. We present the results in Table 3 and observe that the variance aware cycle-back regression loss outperforms both of the other losses in all metrics. We name this version of cycle-consistency as the final temporal cycle consistency (TCC) method, and use this version for the rest of the experiments.

Loss	Phase Classification(%)	Phase Progression	Kendall's Tau
Mean Squared Error	86.16	0.6532	0.6093
Cycle-back classification	88.06	0.6636	0.6707
Cycle-back regression	<b>91.82</b>	<b>0.8030</b>	<b>0.8516</b>

**Table 3:** Ablation of different cycle consistency losses.

### 5.3. Action Phase Classification

**Self-supervised Learning from Scratch.** We perform experiments to compare different self-supervised methods for learning visual representations from scratch. This is a challenging setting as we learn the entire encoder from scratch without labels. We use a smaller encoder model (i.e. VGG-M [11]) as the training samples are limited. We report the results on the *Pouring* and *Penn Action* datasets in Table 4. On both datasets, TCC features outperform the features learned by SaL and TCN. This might be attributed to the fact that TCC learns features across multiple videos during training itself. SaL and TCN losses operate on frames from a single video only but TCC considers frames from multiple videos while calculating the cycle-consistency loss. We can also compare these results with the supervised learning setting (first row in each section), in which we train the encoder using the labels of the phase classification task. For both datasets, TCC can be used for learning features from scratch and brings about significant performance boosts over plain supervised learning when there is limited labeled data.

**Self-supervised Fine-tuning.** Features from networks trained for the task of image classification on the ImageNet dataset have been used for many other vision tasks. They are also useful because initializing from weights of

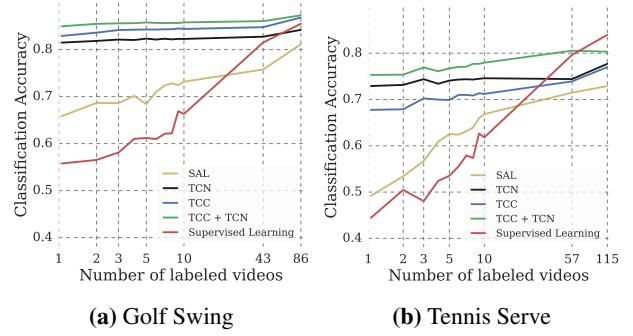
Datasets	% of Labels →	0.1	0.5	1.0
Penn Action	Supervised Learning	50.71	72.86	79.98
	SaL [27]	66.15	71.10	72.53
	TCN [35]	69.65	71.41	72.15
	TCC (ours)	<b>74.68</b>	<b>76.39</b>	<b>77.30</b>
Pouring	Supervised Learning	62.01	77.67	88.41
	SaL [27]	74.50	80.96	83.19
	TCN [35]	76.03	83.27	84.57
	TCC (ours)	<b>86.82</b>	<b>89.43</b>	<b>90.21</b>

**Table 4:** Phase classification results when training VGG-M from scratch.

Datasets	% of Labels →	0.1	0.5	1.0
Penn Action	Supervised Learning	67.10	82.78	86.05
	Random Features	44.18	46.19	46.81
	ImageNet Features	44.96	50.91	52.86
	SaL [27]	74.87	78.26	79.96
	TCN [35]	81.99	83.67	84.04
	TCC (ours)	81.26	83.35	84.45
	TCC + SAL (ours)	81.93	83.46	84.29
	TCC + TCN (ours)	<b>84.27</b>	<b>84.79</b>	<b>85.22</b>
	Supervised Learning	75.43	86.14	91.55
	Random Features	42.73	45.94	46.08
Pouring	ImageNet Features	43.85	46.06	51.13
	SaL [27]	85.68	87.84	88.02
	TCN [35]	89.19	90.39	90.35
	TCC (ours)	<b>89.23</b>	<b>91.43</b>	<b>91.82</b>
	TCC + SaL (ours)	89.21	90.69	90.75
	TCC + TCN (ours)	89.17	91.23	91.51

**Table 5:** Phase classification results when fine-tuning ImageNet pre-trained ResNet-50.

pre-trained networks leads to faster convergence. We train all the representation learning methods mentioned in Section 5.1 and report the results on the *Pouring* and *Penn Action* datasets in Table 5. Here the encoder model is a ResNet-50 [19] pre-trained on ImageNet dataset. We observe that existing self-supervised approaches like SaL and TCN learn features useful for fine-grained video tasks. TCC features achieve competitive performance with the other methods on the *Penn Action* dataset while outperforming them on the *Pouring* dataset. Interestingly, the best performance is achieved by combining the cycle-consistency loss with TCN (row 8 in each section). The boost in performance when combining losses might be because training with multiples losses reduces over-fitting to cues using which the model can minimize a particular loss. We can also look at the first row of their respective sections to compare with supervised learning features obtained by training on the downstream task itself. We observe that the self-supervised fine-tuning gives significant performance boosts in the low-labeled data regime (columns 1 and 2).



**Figure 5: Few shot action phase classification.** TCC features provide significant performance boosts when there is a dearth of labeled videos.

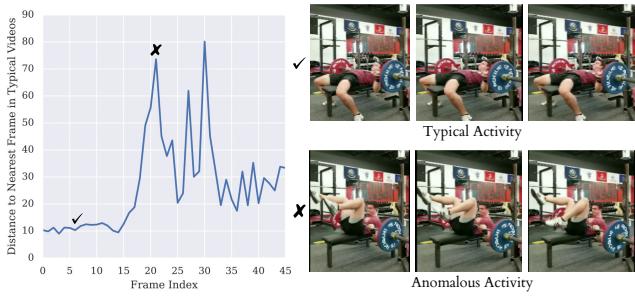
**Self-supervised Few Shot Learning.** We also test the usefulness of our learned representations in the few-shot scenario: we have many training videos but *per-frame labels* are only available for a few of them. In this experiment, we use the same set-up as the fine-tuning experiment described above. The embeddings are learned using either a self-supervised loss or vanilla supervised learning. To learn the self-supervised features, we use the entire training set of videos. We compare these features against the supervised learning baseline where we train the model on the videos for which labels are available. Note that one labeled video means hundreds of labeled frames. In particular, we want to see how the performance on the phase classification task is affected by increasing the number of labeled videos. We present the results in Figure 5. We observe significant performance boost using self-supervised methods as opposed to just using supervised learning on the labeled videos. We present results from *Golf Swing* and *Tennis Serve* classes above. With only one labeled video, TCC and TCC+TCN achieve the performance that supervised learning achieves with about 50 densely labeled videos. This suggests that there is a lot of untapped signal present in the raw videos which can be harvested using self-supervision.

Dataset → Tasks →	Penn Action		Pouring	
	Progress	$\tau$	Progress	$\tau$
SL from Scratch	0.5332	0.4997	0.5529	0.5282
SL Fine-tuning	0.6267	0.5582	0.6986	0.6195
SaL [27]	0.4107	0.4940	0.6652	0.6528
TCN [35]	0.4319	0.4998	0.6141	0.6647
TCC (ours)	<b>0.5383</b>	<b>0.6024</b>	<b>0.7750</b>	<b>0.7504</b>
Finetuning	0.5943	0.6336	0.7451	0.7331
	0.6762	0.7328	0.8057	0.8669
	0.6726	0.7353	0.8030	0.8516
	<b>0.6839</b>	0.7286	0.8204	0.8241
	0.6793	<b>0.7672</b>	<b>0.8307</b>	<b>0.8779</b>

**Table 6:** Phase Progression and Kendall’s Tau results. SL: Supervised Learning.



**Figure 6:** Nearest neighbors in the embedding space can be used for fine-grained retrieval.



**Figure 7:** Example of anomaly detection in a video. Distance from typical action trajectories spikes up during anomalous activity.

#### 5.4. Phase Progression and Kendall’s Tau

We evaluate the encodings for the remaining tasks described in Section 4.1. These tasks measure the effectiveness of representations at a more fine-grained level than phase classification. We report the results of these experiments in Table 6. We observe that when training from scratch TCC features perform better on both phase progression and Kendall’s Tau for both the datasets. Additionally, we note that Kendall’s Tau (which measures alignment between sequences using nearest neighbors matching) is significantly higher when we learn features using the combined losses. TCC + TCN outperforms both supervised learning and self-supervised learning methods significantly for both the datasets for fine-grained tasks.

## 6. Applications

**Cross-modal transfer in Videos.** We are able to align a dataset of related videos without supervision. The alignment across videos enables transfer of annotations or other modalities from one video to another. For example, we can use this technique to transfer text annotations to an en-

tire dataset of related videos by only labeling one video. One can also transfer other modalities associated with time like sound. We can *hallucinate* the sound of pouring liquids from one video to another purely on the basis of visual representations. We copy over the sound from the retrieved nearest neighbors and stitch the sounds together by simply concatenating the retrieved sounds. No other post-processing step is used. The results are in the supplementary material.

**Fine-grained retrieval in Videos.** We can use the nearest neighbours for fine-grained retrieval in a set of videos. In Figure 6, we show that we can retrieve frames when the glass is half full (Row 1) or when the hand has just placed the container back after pouring (Row 2). Note that in all retrieved examples, the liquid has already been transferred to the target container. For the *Baseball Pitch* class, the learned representations can even differentiate between the frames when the leg was up before the ball was pitched (Row 3) and after the ball was pitched (Row 4).

**Anomaly detection.** Since we have well-behaved nearest neighbors in the TCC embedding space, we can use the distance from an *ideal* trajectory in this space to detect anomalous activities in videos. If a video’s trajectory in the embedding space deviates too much from the ideal trajectory, we can mark those frames as anomalous. We present an example of a video of a person attempting to bench-press in Figure 7. In the beginning the distance of the nearest neighbor is quite low. But as the video progresses, we observe a sudden spike in this distance (around the 20<sup>th</sup> frame) where the person’s activity is very different from the ideal bench-press trajectory.

**Synchronous Playback.** Using the learned alignments, we can transfer the pace of a video to other videos of the same action. We include examples of different videos playing synchronously in the supplementary material.

## 7. Conclusion

In this paper, we present a self-supervised learning approach that is able to learn features useful for temporally fine-grained tasks. In multiple experiments, we find self-supervised features lead to significant performance boosts when there is a lack of labeled data. With only one labeled video, TCC achieves similar performance to supervised learning models trained with about 50 videos. Additionally, TCC is more than a proxy task for representation learning. It serves as a general-purpose temporal alignment method that works without labels and benefits any task (like annotation transfer) which relies on the alignment itself.

**Acknowledgements:** We would like to thank Anelia Angelova, Relja Arandjelović, Sergio Guadarrama, Shefali Umrania, and Vincent Vanhoucke for their feedback on the manuscript. We are also grateful to Sourish Chaudhuri for his help with the data collection and Alexandre Passos, Allen Lavoie, Bryan Seybold, and Priya Gupta for their help with the infrastructure.

## References

- [1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.
- [2] Jean-Baptiste Alayrac, Piotr Bojanowski, Nishant Agrawal, Ivan Laptev, Josef Sivic, and Simon Lacoste-Julien. Unsupervised learning from narrated instruction videos. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [3] Theodore Wilbur Anderson. *An introduction to multivariate statistical analysis*, volume 2. Wiley New York, 1958.
- [4] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep canonical correlation analysis. In *International Conference on Machine Learning*, pages 1247–1255, 2013.
- [5] Rodrigo Santa Cruz Basura Fernando Anoop and Cherian Stephen Gould. Deeppermnet: Visual permutation learning. *learning*, 33:25.
- [6] Relja Arandjelovic and Andrew Zisserman. Look, listen and learn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 609–617. IEEE, 2017.
- [7] Yusuf Aytar, Tobias Pfaff, David Budden, Tom Le Paine, Ziyu Wang, and Nando de Freitas. Playing hard exploration games by watching youtube. *arXiv preprint arXiv:1805.11592*, 2018.
- [8] Federico Becattini, Tiberio Uricchio, Lorenzo Seidenari, Alberto Del Bimbo, and Lamberto Ballan. Am i done? predicting action progress in videos. *arXiv preprint arXiv:1705.01781*, 2017.
- [9] Piotr Bojanowski, Rémi Lajugie, Francis Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, and Josef Sivic. Weakly supervised action labeling in videos under ordering constraints. In *European Conference on Computer Vision*, pages 628–643. Springer, 2014.
- [10] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 4724–4733. IEEE, 2017.
- [11] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.
- [12] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 720–736, 2018.
- [13] Luca Del Pero, Susanna Ricco, Rahul Sukthankar, and Vittorio Ferrari. Articulated motion discovery using pairs of trajectories. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2151–2160, 2015.
- [14] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015.
- [15] Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2051–2060, 2017.
- [16] Basura Fernando, Hakan Bilen, Efstratios Gavves, and Stephen Gould. Self-supervised video representation learning with odd-one-out networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5729–5738. IEEE, 2017.
- [17] Rohit Girdhar, Deva Ramanan, Abhinav Gupta, Josef Sivic, and Bryan Russell. Actionvlad: Learning spatio-temporal aggregation for action classification. In *CVPR*, volume 2, page 3, 2017.
- [18] Jacob Goldberger, Geoffrey E Hinton, Sam T Roweis, and Ruslan R Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems*, pages 513–520, 2005.
- [19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [20] Farnoosh Heidarivincheh, Majid Mirmehdi, and Dima Damen. Action completion: A temporal model for moment detection. *arXiv preprint arXiv:1805.06749*, 2018.
- [21] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. *arXiv preprint arXiv:1711.03213*, 2017.
- [22] Aapo Hyvärinen and Hiroshi Morioka. Unsupervised feature extraction by time-contrastive learning and nonlinear ica. In *Advances in Neural Information Processing Systems*, pages 3765–3773, 2016.
- [23] Hilde Kuehne, Ali Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 780–787, 2014.
- [24] Tian Lan, Yuke Zhu, Amir Roshan Zamir, and Silvio Savarese. Action recognition by hierarchical mid-level action elements. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4552–4560, 2015.
- [25] Colin Lea, Austin Reiter, René Vidal, and Gregory D Hager. Segmental spatiotemporal cnns for fine-grained action segmentation. In *European Conference on Computer Vision*, pages 36–52. Springer, 2016.
- [26] Shugao Ma, Leonid Sigal, and Stan Sclaroff. Learning activity progression in lstms for activity detection and early detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1942–1950, 2016.
- [27] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, pages 527–544. Springer, 2016.
- [28] Yair Movshovitz-Attias, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 360–368. IEEE, 2017.
- [29] Hamed Pirsiavash and Deva Ramanan. Parsing videos of actions with segmental grammars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 612–619, 2014.

- [30] Jérôme Revaud, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. Event retrieval in large video collections with circulant temporal encoding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2459–2466, 2013.
- [31] Ignacio Rocco, Mircea Cimpoi, Relja Arandjelović, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Neighbourhood consensus networks. In *Advances in Neural Information Processing Systems*, pages 1658–1669, 2018.
- [32] Artsiom Sanakoyeu, Miguel A Bautista, and Björn Ommer. Deep unsupervised learning of visual similarities. *Pattern Recognition*, 78:331–343, 2018.
- [33] Fadime Sener and Angela Yao. Unsupervised learning and segmentation of complex activities from video. *arXiv preprint arXiv:1803.09490*, 2018.
- [34] Ozan Sener, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Unsupervised semantic parsing of video collections. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4480–4488, 2015.
- [35] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. *Proceedings of International Conference in Robotics and Automation (ICRA)*.
- [36] Eli Shechtman and Michal Irani. Matching local self-similarities across images and videos. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.
- [37] Gunnar Sigurdsson, Abhinav Gupta, Cordelia Schmid, Ali Farhadi, and Karteek Alahari. Actor and observer: Joint modeling of first and third-person videos. In *CVPR-IEEE Conference on Computer Vision & Pattern Recognition*, 2018.
- [38] Gunnar A Sigurdsson, Santosh Kumar Divvala, Ali Farhadi, and Abhinav Gupta. Asynchronous temporal fields for action recognition. In *CVPR*, volume 5, page 7, 2017.
- [39] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [40] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.
- [41] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, pages 1857–1865, 2016.
- [42] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.
- [43] Fan Wang, Qixing Huang, and Leonidas J Guibas. Image co-segmentation via consistent functional maps. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 849–856, 2013.
- [44] Fan Wang, Qixing Huang, Maks Ovsjanikov, and Leonidas J Guibas. Unsupervised multi-class joint image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3142–3149, 2014.
- [45] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3551–3558, 2013.
- [46] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European Conference on Computer Vision*, pages 20–36. Springer, 2016.
- [47] Wikipedia contributors. Coefficient of determination — Wikipedia, the free encyclopedia, 2018.
- [48] Wikipedia contributors. Kendall rank correlation coefficient — Wikipedia, the free encyclopedia, 2018.
- [49] Kyle Wilson and Noah Snavely. Network principles for sfm: Disambiguating repeated structures with local context. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 513–520, 2013.
- [50] Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *International Journal of Computer Vision*, 126(2-4):375–389, 2018.
- [51] Christopher Zach, Manfred Klopschitz, and Marc Pollefeys. Disambiguating visual relations using loop constraints. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1426–1433. IEEE, 2010.
- [52] Weiyu Zhang, Menglong Zhu, and Konstantinos G Derpanis. From actemes to action: A strongly-supervised representation for detailed action understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2248–2255, 2013.
- [53] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. Temporal action detection with structured segment networks. *ICCV, Oct. 2*, 2017.
- [54] Tinghui Zhou, Yong Jae Lee, Stella X Yu, and Alyosha A Efros. Flowweb: Joint image set alignment by weaving consistent, pixel-wise correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1191–1200, 2015.
- [55] Tinghui Zhou, Philipp Krahenbuhl, Mathieu Aubry, Qixing Huang, and Alexei A Efros. Learning dense correspondence via 3d-guided cycle consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 117–126, 2016.
- [56] Xiaowei Zhou, Menglong Zhu, and Kostas Daniilidis. Multi-image matching via fast alternating minimization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4032–4040, 2015.
- [57] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.

## Appendix

### A. Synchronous Playback

One direct application of being able to align videos is that we can play multiple videos with the pace of a reference video. The task of synchronizing videos manually can be very time-consuming, often requiring multiple cuts and frame rate changes. We show how we can use self-supervised learning to reduce the effort required to synchronize videos. We present these results here: <https://sites.google.com/corp/view/temporal-cycle-consistency/home/visualizations-results>. We produce these videos by first embedding all frames in all the videos using our trained encoder. We choose a reference video with whose pace we want to play all the other videos. For every other video, we choose the matching frame in the whole video using dynamic time warping. This is done to enforce temporal constraints on a per-frame basis. No other post processing steps are used.

### B. Sound Transfer

We can also transfer other meta-data or modalities (that are synchronized with the frames in a video) only on the basis of the visual similarity. We showcase an example of such a transfer by using sound, which is arguably the most commonly available synchronized modality. Please find examples of sound transfer in the teaser video. In order to transfer the sound, we look up the nearest neighbor frames in a video that has sound. For each frame in the target video, we copy over the block of sound associated with the nearest neighbor frame. We concatenate these blocks of sound. Note, how the sound changes as the liquid flows into the container. This presents further evidence the embeddings are able to capture progress in a particular task. We use multiple frames in the sound synthesis. We average the embeddings for the multiple frames and concatenate the corresponding sounds to produce the sound blocks. We do this so that edge artifacts are reduced when we synthesize the sound for the whole video. No other post processing steps are used.

### C. t-SNE Visualization

We also present examples of t-SNE visualization of the embeddings in the teaser video and Figure 8. For each action, we show trajectories of 4 videos in the embedding space. The borders are color-coded differently for each video. We sample two random time-steps for each video and show the corresponding frame and embedding location. Frames with the same border color are sampled from different time-steps in the same video. The visualization indicates how the embeddings change as an action is carried out. Additionally, they also highlight how corresponding frames from different videos in the validation set are closer to each other in the learned embedding space as compared to non-corresponding frames. This structure in the embedding space, induced by the self-supervised objectives during training, is why we are able to align different videos and perform fine-grained retrieval by simply using nearest neighbors.

### D. Fine-grained Retrieval

We provide additional results for fine-grained retrieval in Figure 9.

### E. Data Augmentation

We use data augmentation during training. We randomly flip an entire video horizontally. We perturb brightness by adding a random number between -32 and 32 to the raw pixels. We change contrast by a random factor sampled uniformly between 0.5 and 1.5. All training algorithms have the same data augmentation pipeline.

### F. Alignments under Different Losses

We show how the alignment between two videos evolves as training proceeds in Figure 10. The similarity matrices are calculated on the basis of the distance in the embedding space. The intensity at  $(i, j)$  coordinates of the matrices encodes the similarity between the  $i^{th}$  frame of video 1 and  $j^{th}$  frame of video 2. The more bright a cell is, the more similar those frames are. In the beginning, the nearest neighbor matches (encoded as the brightest cells for each row/column) don't provide good alignment. As we train for more iterations, alignment between the two videos emerges as more *similar* (brighter) frames exist along the diagonal. The alignment that emerges by using the cycle-back regression loss is more ordered than the cycle-back classification loss which does not take time into account while applying the cycle-consistency loss.

### G. Hyperparameters

In Table 7, we tabulate the list of values of the hyperparameters.

Hyperparameter	Value
Batch Size	4
Number of frames	20
Optimizer	ADAM
Learning Rate	$1.0 \times 10^{-4}$
Weight Decay	$1.0 \times 10^{-5}$
Alignment Variance $\lambda$	0.001
TCN Positive Window Size	5
Frames per second	20 (Penn Action), 30 (Pouring)
SaL Classifier FC Sizes	128, 64
SaL Fraction Shuffled	0.75

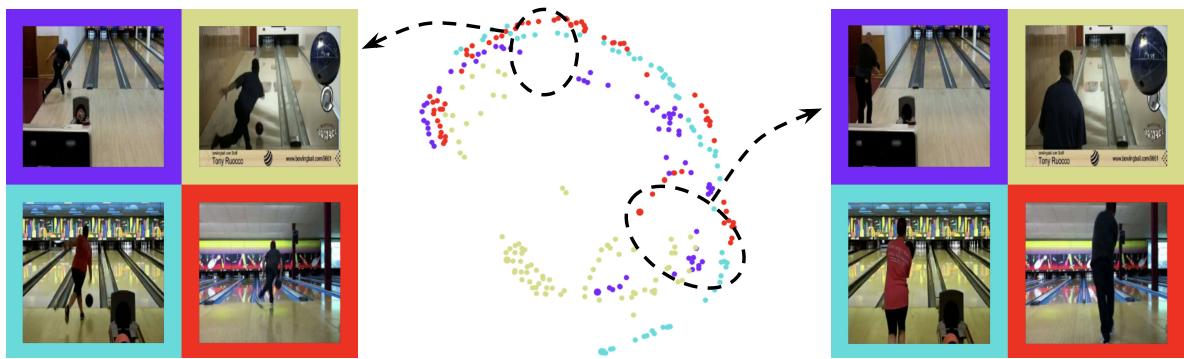
Table 7: List of hyperparameters used

### H. Architecture Details

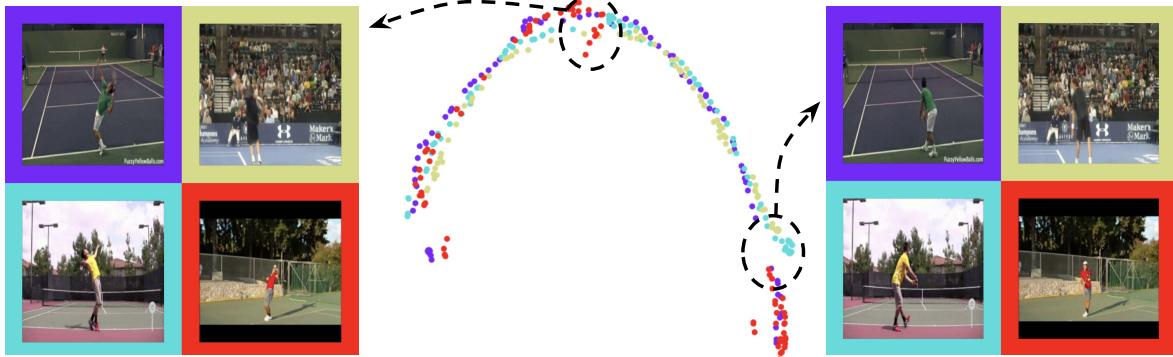
We describe the complete architecture of our encoder  $\phi$  in Table 8. It is composed of 2 parts: *Base Network* and *Embedder Network*. The *Base Network* acts on individual frames to extract convolutional features from them. Depending on the chosen base



(a) Pouring



(b) Bowling



(c) Tennis serve

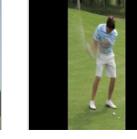
**Figure 8: t-SNE Visualization of Embeddings.**

network,  $c_4$  ( $c$  in Table 1 of the main paper) is either 1024 or 512. The *Embedder Network* collects convolutional features of each frame and its context window and embeds them into a single 128 dimensional vector. All the different training algorithms in our experiments are applied on top of these 128 dimensional vectors. While initially we were experimenting with larger values of  $k$ , we found even with  $k = 2$  we can get good performance on both datasets. The gap between the two frames is approximately 0.75 seconds (15 frames at 20 fps) for the Penn Action dataset and 0.3 seconds (9 frames at 30 fps) for the Pouring dataset.

Query



Golf-stick swung up before hitting the ball



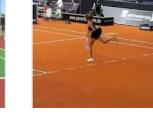
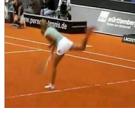
Golf-stick hitting the ball



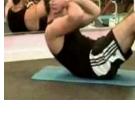
Golf-stick swung up after hitting the ball



Hand swung fully back before serving in tennis



Leg up after serving in tennis



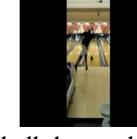
Bending during sit-up



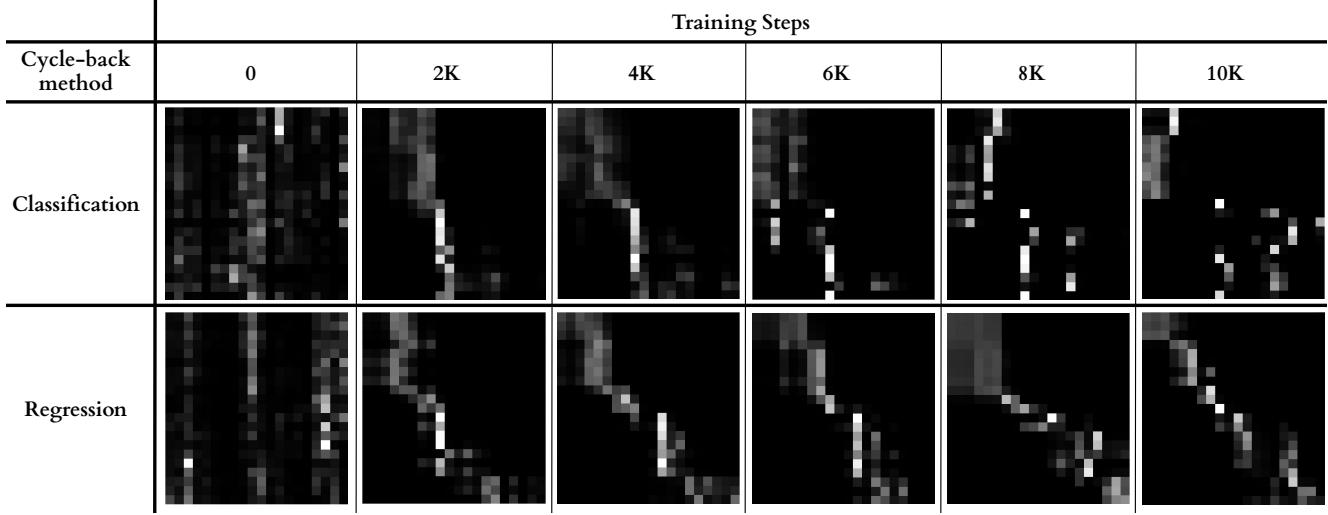
Lying down during sit-up



Just before releasing ball during bowling



Just after releasing ball during bowling



**Figure 10: Evolution of similarity matrices under different losses.** The matrices above encode the similarity between frames of two *Baseball Pitch* videos. As training proceeds, the videos get aligned more along the diagonal of the matrix, but the cycle-back regression loss is more effective at aligning the videos as compared to the cycle-back classification loss. More details in Section F.

Model	Layer	Output Size	Pre-trained ResNet-50	VGG-M Like (Scratch)
Base Network	conv1	$112 \times 112 \times c_1$	$7 \times 7, 64, \text{stride } 2$	
			$3 \times 3 \text{ max pool, stride } 2$	
	conv2_x	$56 \times 56 \times c_2$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 1$
	conv3_x	$28 \times 28 \times c_3$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 1$
	conv4_x	$14 \times 14 \times c_4$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 1$
	Temporal Stacking	$k \times 14 \times 14 \times c_4$	Stack $k$ context frame features in time axis	
	conv5_x	$k \times 14 \times 14 \times 512$	$3 \times 3 \times 3, 512$	$\times 1$
	$3 \times 3 \times 3, 512$		$3 \times 3 \times 3, 512$	
Embedder Network	Spatio-temporal Pooling	512	Global 3D Max-Pool	
	fc6_x	512	$512$	$\times 1$
	Embedding	128	128	

**Table 8:** Architectures used in our experiments. The network produces an embedding for each frame (and its context window).  $c_i$  depends on the choice of the base network. Inside the square brackets, the parameters in the form of: (1)  $[n \times n, c]$  refers to 2D Convolution filter size and number of channels respectively (2)  $[n \times n \times n, c]$  refers to 3D Convolution filter size and number of channels respectively (3)  $[c]$  refers to channels in a fully-connected layer. Downsampling in ResNet-50 is done using convolutions with stride 2, while in VGG-M models we use MaxPool with stride 2 for downsampling.