Tubelet-Contrastive Self-Supervision for Video-Efficient Generalization

Thoker, F.M.; Doughty, H.; Snoek, C.G.M.

DOI
10.48550/arXiv.2303.11003
10.1109/ICCV51070.2023.01270

Publication date
2023

Document Version
Final published version

Published in
2023 IEEE/CVF International Conference on Computer Vision

License
Article 25fa Dutch Copyright Act (https://www.openaccess.nl/en/in-the-netherlands/you-share-we-take-care)

Citation for published version (APA):
Tubelet-Contrastive Self-Supervision for Video-Efficient Generalization

Fida Mohammad Thoker, Hazel Doughty, Cees G. M. Snoek
University of Amsterdam

Abstract

We propose a self-supervised method for learning motion-focused video representations. Existing approaches minimize distances between temporally augmented videos, which maintain high spatial similarity. We instead propose to learn similarities between videos with identical local motion dynamics but an otherwise different appearance. We do so by adding synthetic motion trajectories to videos which we refer to as tubelets. By simulating different tubelet motions and applying transformations, such as scaling and rotation, we introduce motion patterns beyond what is present in the pretraining data. This allows us to learn a video representation that is remarkably data efficient: our approach maintains performance when using only 25% of the pretraining videos. Experiments on 10 diverse downstream settings demonstrate our competitive performance and generalizability to new domains and fine-grained actions. Code is available at https://github.com/fmthoker/tubelet-contrast.

1. Introduction

This paper aims to learn self-supervised video representations, useful for distinguishing actions. In a community effort to reduce the manual, expensive, and hard-to-scale annotations needed for many downstream deployment settings, the topic has witnessed tremendous progress in recent years [18, 31, 62, 79], particularly through contrastive learning [15, 56, 58, 60]. Contrastive approaches learn representations through instance discrimination [55], by increasing feature similarity between spatially and temporally augmented clips from the same video. Despite temporal differences, such positive video pairs often maintain high spatial similarity (see Figure 1), allowing the contrastive task to be solved by coarse-grained features without explicitly capturing local motion dynamics. This limits the generalizability of the learned video representations, as shown in our prior work [70]. Furthermore, prior approaches are constrained by the amount and types of motions present in the pretraining data. This makes them data-hungry, as video data has high redundancy with periods of little to no motion. In this work, we address the need for data-efficient and generalizable self-supervised video representations by proposing a contrastive method to learn local motion dynamics.

We take inspiration from action detection, where tubelets are used to represent the motions of people and objects in videos through bounding box sequences e.g., [29, 32, 42]. Typically, many tubelet proposals are generated for a video, which are processed to find the best prediction. Rather than finding tubelets in video data, we simulate them. In particular, we sample an image patch and 'paste' it with a randomized motion onto two different video clips as a shared tubelet (see Figure 1). These two clips form a positive pair for contrastive learning where the model has to rely on the spatiotemporal motion dynamics inside the simulated tubelets, while temporal contrastive pairs (top) suffer from a high spatial bias. Contrasting tubelets results in a data-efficient and generalizable video representation.

We make four contributions. First, we explicitly learn from local motion dynamics in the form of synthetic tubelets and design a simple but effective tubelet-contrastive
framework. Second, we propose different ways of simulating tubelet motion and transformations to generate a variety of motion patterns for learning. Third, we reveal the remarkable data efficiency of our proposal: on five action recognition datasets our approach maintains performance when using only 25% of the pretraining videos. What is more, with only 5-10% of the videos we still outperform the vanilla contrastive baseline with 100% pretraining data for several datasets. Fourth, our comparative experiments on 10 downstream settings, including UCF101 [67], HMDB51 [37], Something Something [20], and FineGym [63], further demonstrate our competitive performance, generalizability to new domains, and suitability of our learned representation for fine-grained actions.

2. Related Work

Self-Supervised Video Representation Learning. The success of contrastive learning in images [6, 21, 23, 52] inspired many video contrastive works [27, 45, 56, 58, 60, 69]. Alongside spatial invariances, these works learn invariances to temporal crops [56, 60, 61] and video speed [27, 45, 58]. Some diverge from temporal invariances and encourage equivariance [8, 57] to learn finer temporal representations. For instance, TCLR [8] enforces within-instance temporal feature variation, while TE [30] learns equivariance to temporal crops and speed with contrastive learning. Alternatively, many works learn to predict temporal transformations such as clip order [18, 39, 50, 79], speed [4, 7, 82] and their combinations [31, 47]. These self-supervised temporal representations are effective for classifying and retrieving coarse-grained actions but are challenged by downstream settings with subtle motions [62, 70]. Other works utilize the multimodal nature of videos [1, 2, 19, 22, 48, 51, 57] and learn similarity with audio [1, 2, 51] and optical flow [19, 22, 54, 77]. We contrast motions of synthetic tubelets to learn a video representation from only RGB data that can generalize to tasks requiring fine-grained motion understanding.

Other self-supervised works learn from the spatiotemporal dynamics of video. Both BE [75] and FAME [9] remove background bias by adding static frames [75] or replacing the background [9] in positive pairs. Several works instead use masked autoencoding to learn video representations [13, 71]. However, these works are all limited to the motions present in the pretraining dataset. We prefer to be less dataset-dependent and generate synthetic motion tubelets for contrastive learning, which also offers a considerable data-efficiency benefit. CtP [74] and MoSI [28] both aim to predict motions in pretraining. CtP [74] learns to track image patches in video clips while MoSI [28] learns to predict the speed and direction of added pseudo-motions. We take inspiration from these works and contrast synthetic motions from tubelets which allows us to learn generalizable and data-efficient representations.

Supervised Fine-Grained Motion Learning. While self-supervised works have mainly focused on learning representations to distinguish coarse-grained actions, much progress has been made in supervised learning of motions. Approaches distinguish actions by motion-focused neural network blocks [36, 38, 43, 48], decoupling motion from appearance [40, 68], aggregating multiple temporal scales [14, 53, 80], and sparse coding to obtain a mid-level motion representation [49, 59, 64]. Other works exploit skeleton data [12, 24] or optical flow [16, 66]. Alternatively, several works identify motion differences within an action class, by repetition counting [26, 84, 85], recognizing adverbs [10, 11] or querying for action attributes [83]. Different from all these works, we learn a motion-sensitive video representation with self-supervision. We do so by relying on just coarse-grained video data in pretraining and demonstrate downstream generalization to fine-grained actions.

Tubelets. Jain et al. defined tubelets as class-agnostic sequences of bounding boxes over time [29]. Tubelets can represent the movement of people and objects and are commonly used for object detection in videos [17, 33, 34], spatiotemporal action localization [25, 29, 32, 42, 81, 86] and video relation detection [5]. Initially, tubelets were obtained by supervoxel groupings and dense trajectories [29, 73] and later from 2D CNNs [32, 42], 3D CNNs [25, 81] and transformers [86]. We introduce (synthetic) tubelets of pseudo-objects for contrastive video self-supervised learning.

3. Tubelet Contrast

We aim to learn motion-focused video representations from RGB video data with self-supervision. After revisiting temporal contrastive learning, we propose tubelet-contrastive learning to reduce the spatial focus of video representations and instead learn similarities between spatiotemporal tubelet dynamics (Section 3.1). We encourage our representation to be motion-focused by simulating a variety of tubelet motions (Section 3.2). To further improve data efficiency and generalizability, we add complexity and variety to the motions through tubelet transformations (Section 3.3). Figure 2 shows an overview of our approach.

Temporal Contrastive Learning. Temporal contrastive learning learns feature representations via instance discrimination [55]. This is achieved by maximizing the similarity between augmented clips from the same video (positive pairs) and minimizing the similarity between clips from different videos (negatives). Concretely given a set of videos \( V \), the positive pairs \( (v, v') \) are obtained by sampling different temporal crops of the same video [56, 58] and applying spatial augmentations such as cropping and color jittering. Clips sampled from other videos in the training set act as negatives. The extracted clips are passed through a video encoder and projected on a representation space by a non-linear projection head to obtain clip embeddings \( (Z_v, Z_{v'}) \).
We define a tubelet as a sequence of instance discrimination, allowing us to learn more general-

positive pairs are then employed to learn video representations via motion similarity and a low spatial similarity. Such positive pairs is the tubelets, the network must rely on temporal cues causing a motion-focused video representation.

**3.2. Tubelet Motion**

To learn motion-focused video representations, we need to give our tubelets motion variety. Here, we discuss how to simulate motions by generating different patch movements in the tubelets. Recall, Eq. (2) defines a tubelet by image patch p and its center coordinate in each video frame. We consider two types of tubelet motion: linear and non-linear.

**Linear Motion.** We randomly sample the center locations of object appearances in each frame of a video clip. Let’s assume an object p of size $H' \times W'$ moving in a video clip v of length T. Then the tubelet is defined as follows:

$$
\text{Tubulet}_p = [(x^1, y^1), \ldots, (x^T, y^T)]
$$

where $(x^i, y^i)$ is the center coordinate of the object p in frame i of clip v. For this work, a random image patch of size $H' \times W'$ acts as a pseudo-object overlaid on a video clip to form a tubelet. To generate the tubelet we first make the object appear static, i.e., $x^1 \equiv x^2 = \ldots = x^T$ and $y^1 \equiv y^2 = \ldots = y^T$, and explain how we add motion in Section 3.2.

3.1. Tubelet-Contrastive Learning

Different from existing video contrastive self-supervised methods, we explicitly aim to learn motion-focused video representations while relying only on RGB data. To achieve this we propose to learn similarities between simulated tubelets. Concretely, we first generate tubelets in the form of moving patches which are then overlaid onto two different videos in order to simulate motions by generating different patch movements for a set of $M$ tubelets $\{\text{Tubulet}_{p_1}, \ldots, \text{Tubulet}_{p_M}\}$ from M patches randomly cropped from $v_1$ as:

$$
\hat{v}_1 = v_1 \odot \text{Tubulet}_{p_1} \quad \hat{v}_2 = v_2 \odot \text{Tubulet}_{p_2},
$$

where $\odot$ refers to pasting patch p in each video frame at locations determined by Tubelet$_p$. Eq. (3) can be extended for a set of $M$ tubelets $\{\text{Tubulet}_{p_1}, \ldots, \text{Tubulet}_{p_M}\}$ from M patches randomly cropped from $v_1$ as:

$$
\hat{v}_1 = v_1 \odot \{\text{Tubulet}_{p_1}, \ldots, \text{Tubulet}_{p_M}\} \quad \hat{v}_2 = v_2 \odot \{\text{Tubulet}_{p_1}, \ldots, \text{Tubulet}_{p_M}\}.
$$

As a result, $\hat{v}_1$ and $\hat{v}_2$ share the spatiotemporal dynamics of the moving patches in the form of tubelets and have low spatial bias since the two clips come from different videos. Finally, we adapt the contrastive loss from Eq. (1) and apply $L_{\text{contrast}}(\hat{v}_1, \hat{v}_2)$. Here the set of negatives $N$ contains videos with different tubelets. Since the only similarity in positive pairs is the tubelets, the network must rely on temporal cues causing a motion-focused video representation.

The noise contrastive estimation loss InfoNCE [55] is used for the optimization:

$$
L_{\text{contrast}}(v, v') = -\log \frac{h(Z_{v}, Z_{v'})}{h(Z_{v}, Z_{v'}) + \sum_{Z_n \sim N} h(Z_{v}, Z_n)}
$$

where $h(Z_{v}, Z_{v'}) = \exp(Z_v \cdot Z_{v'}/\tau)$, $\tau$ is the temperature parameter and $N$ is a set of negative clip embeddings.
obtain the following linear motion definition:

\[ \text{Tubelet}_{\text{Lin}} = [(x^1, y^1), (x^2, y^2), \ldots, (x^T, y^T)], \quad \text{s.t.} \quad (5) \]

\[ (x^i, y^i) = \begin{cases} (U(0, W), U(0, H)), & \text{if } i \in K \\ \text{Interp}((x^k, y^k), (x^{k+1}, y^{k+1})), & \text{otherwise} \end{cases} \]

where \( U \) is a function for uniform sampling, \( k \) and \( k+1 \) are the neighboring keyframes to frame \( i \) and \( \text{Interp} \) gives a linear interpolation between keyframes. To ensure smoothness, we constrain the difference between the center locations in neighboring keyframes to be less than \( \Delta \) pixels. This formulation results in tubelet motions where patches follow linear paths across the video frames. The left of Figure 3 shows examples of such linear tubelet motions.

**Non-Linear Motion.** Linear motions are simple and limit the variety of motion patterns that can be generated. Next, we simulate motions where patches move along more complex non-linear paths, to better emulate motions in real videos. We create non-linear motions by first sampling \( N \) 2D coordinates \( (N \gg T) \) uniformly from \( x \in [0, W] \) and \( y \in [0, H] \). Then, we apply a 1D Gaussian filter along \( x \) and \( y \) axes to generate a random smooth nonlinear path as:

\[ \text{Tubelet}_{\text{NonLin}} = [(g(x^1), g(y^1)), \ldots, (g(x^N), g(y^N))] \quad \text{s.t.} \quad g(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-z^2/2\sigma^2} \quad (6) \]

where \( \sigma \) is the smoothing factor for the gaussian kernels. Note the importance of sampling \( N \gg T \) points to ensure a non-linear path. If \( N \) is too small then the path becomes linear after gaussian smoothing. We downsample the resulting non-linear tubelet in Eq. (6) from \( N \) to \( T \) coordinates resulting in the locations for patch \( p \) in the \( T \) frames. The right of Figure 3 shows examples of non-linear tubelet motions.

**3.3. Tubelet Transformation**

The tubelet motions are simulated by changing the position of the patch across the frames in a video clip, i.e., with translation. In reality, the motion of objects in space may appear as other transformations in videos, for instance, scale decreasing as the object moves away from the camera or motions due to planar rotations. Motivated by this, we propose to add more complexity and variety to the simulated motions by transforming the tubelets. In particular, we propose scale, rotation, and shear transformations. As before, we sample keyframes \( K \) with the first \( i=0 \) and last frames \( i=T \) always included. Transformations for remaining frames are linearly interpolated. Formally, we define a tubelet transformation as a sequence of spatial transformations applied to the patch \( p \) in each frame \( i \) as:

\[ \text{Trans}_F = [p, F(p, \theta^2), \ldots, F(p, \theta^T)], \quad \text{s.t.} \quad \theta^i = \begin{cases} U(\text{Min}, \text{Max}), & \text{if } i \in K \\ \text{Interp}(\theta^k, \theta^{k+1}), & \text{otherwise} \end{cases} \quad (7) \]

where \( F(p, \theta^i) \) applies the transformation to patch \( p \) according to parameters \( \theta^i \). \( U \) samples from a uniform distribution and \( \theta^k \) and \( \theta^{k+1} \) are the parameters for the keyframes neighboring frame \( i \). For the first keyframe, no transformation is applied thus representing the initial state of the patch \( p \). We instantiate three types of such tubelet transformations: scale, rotation, and shear. Examples are shown in Figure 4.

**Scale.** We scale the patch across time with \( F(p, \theta_i) \) and horizontal and vertical scaling factors \( \theta^i=(w^i, h^i) \). To sample \( w^i \) and \( h^i \), we use \( \text{Min}=0.5 \) and \( \text{Max}=1.5 \).

**Rotation.** In this transformation \( F(p, \theta_i) \) applies in-plane rotations to tubelet patches. Thus, \( \theta^i \) is a rotation angle sampled from \( \text{Min}=-90^\circ \) and \( \text{Max}=+90^\circ \).

**Shear.** We shear the patch as the tubelet progresses with \( F(p, \theta_i) \). The shearing parameters are \( \theta^i=(r^i, s^i) \) which are sampled using \( \text{Min}=-1.5 \) and \( \text{Max}=1.5 \).

With these tubelet transformations and the motions created in Section 3.2 we are able to simulate a variety of subtle motions in videos, making the model data-efficient. By learning the similarity between the same tubelet overlaid onto different videos, our model pays less attention to spatial features, instead learning to represent these subtle motions. This makes the learned representation generalizable to different domains and action granularities.

**4. Experiments**

**4.1. Datasets, Evaluation & Implementation**

**Pretraining Datasets.** Following prior work [8,27,56–58,74] we use **Kinetics-400** [35] for self-supervised pretraining. Kinetics-400 is a large-scale action recognition dataset containing 250K videos of 400 action classes. To show data
efficiency, we also pretrain with Mini-Kinetics [78], a subset containing 85K videos of 200 action classes.

### Downstream Evaluation.

To evaluate the video representations learned by our tubelet contrast, we finetune and evaluate our model on various downstream datasets summarized in Table 1. Following previous self-supervised work, we evaluate on standard benchmarks: UCF101 [67] and HMDB51 [37]. These action recognition datasets contain coarse-grained actions with domains similar to Kinetics-400. For both, we report top-1 accuracy on split 1 from previous work [70]. This consists of eight experiments over four downstream generalization factors: domain shift, sample efficiency, action granularity, and task shift. Domain shift is evaluated on Something-Something v2 [20] (SSv2) and FineGym [63] (Gym99) which vary in domain relative to Kinetics-400. Sample efficiency evaluates low-shot action recognition on UCF101 [67] and FineGym [63] with 1,000 training samples, referred to as UCF (1000) and Gym (1000). Action granularity evaluates semantically similar actions using FX-S1 and UB-S1 subsets from FineGym [63]. In both subsets, action classes belong to the same element of a gymnastic routine, e.g., FX-S1 is types of jump. Task shift evaluates tasks beyond single-label action recognition. Specifically, it uses temporal repetition counting on UCFRep [84], a subset of UCF-101 [84], and multi-label action recognition on Charades [65]. The experimental setups are detailed in Table 1 and all follow SEVERE [70].

### Tubelet Generation and Transformation.

Our clips are 16112×112 frames with standard spatial augmentations: random crops, horizontal flip, and color jitter. We randomly crop 2 patches to generate M=2 tubelets (Eq. 4). The patch size H×W is uniformly sampled from [16×16, 64×64]. We also randomly sample a patch shape from a set of predefined shapes. For linear motions, we use ∆=[40−80] displacement difference. For non-linear motion, we use N=48 and a smoothing factor of σ=8 (Eq. 6). For linear motion and all tubelet transformations, we use K=3 keyframes.

### Networks, Pretraining and Finetuning.

We use R(2+1)D-18 [72] as the video encoder, following previous self-supervision works [8, 9, 56–58, 76]. The projection head is a 2-layer MLP with 128D output. We use momentum contrast [23] to increase the number of negatives |N| (Eq. 1) to 16,384 for Mini-Kinetics and 65,536 for Kinetics. We use temperature τ=0.2 (Eq. 1). The model is optimized using SGD with momentum 0.9, learning rate 0.01, and weight decay 0.0001. We use a batch size of 32 for Mini-Kinetics and 128 for Kinetics, a cosine scheduler [46], and pretrain for 100 epochs. After pretraining, we replace the projection head with a task-dependent head as in SEVERE [70] and finetune the whole network with labels for the downstream task. We provide finetuning details in the supplementary.

### 4.2. Ablation Studies & Analysis.

To ablate the effectiveness of individual components we pretrain on Mini-Kinetics and evaluate on UCF (1000), Gym (1000), Something-Something v2 and UB-S1. To decrease the finetuning time we use a subset of Something Something (SSv2-Sub) with 25% of the training data (details in supplementary). Unless specified otherwise, we use non-linear motion and rotation to generate tubelets.

### Tubelet-Contrastive Learning.

Table 2 shows the benefits brought by our tubelet-contrastive learning. We first observe that our full tubelet-contrastive model improves considerably over the temporal contrastive baseline, which uses MoCo [23] with a temporal crop augmentation. This

---

**Table 1: Benchmark Details**

<table>
<thead>
<tr>
<th>Evaluation Factor</th>
<th>Experiment</th>
<th>Dataset</th>
<th>Task</th>
<th>#Classes</th>
<th>#Finetuning</th>
<th>#Testing</th>
<th>Eval Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Standard</strong></td>
<td>UCF101</td>
<td>UCF 101 [67]</td>
<td>Action Recognition</td>
<td>101</td>
<td>9,537</td>
<td>3,783</td>
<td>Top-1 Accuracy</td>
</tr>
<tr>
<td></td>
<td>HMDB51</td>
<td>HMDB 51 [37]</td>
<td>Action Recognition</td>
<td>101</td>
<td>3,570</td>
<td>1,530</td>
<td>Top-1 Accuracy</td>
</tr>
<tr>
<td></td>
<td>Gym99</td>
<td>FineGym [63]</td>
<td>Action Recognition</td>
<td>99</td>
<td>20,484</td>
<td>8,521</td>
<td>Top-1 Accuracy</td>
</tr>
<tr>
<td><strong>Sample Efficiency</strong></td>
<td>UCF (1000)</td>
<td>UCF 101 [67]</td>
<td>Action Recognition</td>
<td>101</td>
<td>1,000</td>
<td>3,783</td>
<td>Top-1 Accuracy</td>
</tr>
<tr>
<td></td>
<td>Gym (1000)</td>
<td>FineGym [63]</td>
<td>Action Recognition</td>
<td>101</td>
<td>1,000</td>
<td>3,783</td>
<td>Top-1 Accuracy</td>
</tr>
<tr>
<td><strong>Action Granularity</strong></td>
<td>FX-S1</td>
<td>FineGym [63]</td>
<td>Action Recognition</td>
<td>11</td>
<td>1,882</td>
<td>777</td>
<td>Mean Class Acc</td>
</tr>
<tr>
<td></td>
<td>UB-S1</td>
<td>FineGym [63]</td>
<td>Action Recognition</td>
<td>15</td>
<td>3,511</td>
<td>1,471</td>
<td>Mean Class Acc</td>
</tr>
<tr>
<td><strong>Task Shift</strong></td>
<td>UCF-RC</td>
<td>UCFRep [84]</td>
<td>Repetition Counting</td>
<td>-</td>
<td>421</td>
<td>105</td>
<td>Mean Error mAP</td>
</tr>
<tr>
<td></td>
<td>Charades</td>
<td>Charades [65]</td>
<td>Multi-label Recognition</td>
<td>157</td>
<td>7,985</td>
<td>1,863</td>
<td></td>
</tr>
</tbody>
</table>

---

**Table 2: Tubelet-Contrastive Learning**

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Dataset</th>
<th>Task</th>
<th>#Classes</th>
<th>#Finetuning</th>
<th>#Testing</th>
<th>Eval Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCF (1000)</td>
<td>Gym (1000)</td>
<td>SSv2-Sub</td>
<td>UB-S1</td>
<td>Temporal Contrast</td>
<td>57.5</td>
<td>29.5</td>
</tr>
<tr>
<td><strong>Tublet Contrast</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tubelet Generation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>48.2</td>
</tr>
<tr>
<td>Tubelet Motion</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>63.0</td>
</tr>
<tr>
<td>Tubelet Transformation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>65.5</td>
</tr>
</tbody>
</table>
Temporal Contrastive Learning proves considerably, tubelet motion is added (Section 3.2), performance improves from the temporal contrast baseline. When from our tubelet generation (Section 3.1) actually decreases presentations required for finer temporal understanding. These results demonstrate that learning similarities between synthetic tubelets produces generalizable, but motion-focused, video representations required for finer temporal understanding.

It is clear that the motion within tubelets is critical to our model’s success as contrasting static tubelets obtained from our tubelet generation (Section 3.1) actually decreases the performance from the temporal contrast baseline. When tubelet motion is added (Section 3.2), performance improves considerably, e.g., Gym (10^3) +17.4% and SSv2-Sub +7.4%. Finally, adding more motion types via tubelet transformations (Section 3.3) further improves the video representation quality, e.g., UCF (10^3) +2.5% and Gym (10^3) +2.4%. This highlights the importance of including a variety of motions beyond what is present in the pretraining data to learn generalizable video representations.

**Tubelet Motions.** Next, we ablate the impact of the tubelet motion type (Section 3.2) without transformations. We compare the performance of static tubelets with no motion, linear motion, and non-linear motion in Table 3. Tubelets with simple linear motion already improve performance for all four datasets, e.g., +6.4% on Gym (10^3). Using non-linear motion further improves results, for instance with an additional +11.0% improvement on Gym (10^3). We conclude that learning from non-linear motions provides more generalizable video representations.

**Tubelet Motions.** Table 3: Tubelet Motions. Learning from tubelets with non-linear motion benefits multiple downstream settings.

<table>
<thead>
<tr>
<th>Tubelet Motion</th>
<th>UCF (10^3)</th>
<th>Gym (10^3)</th>
<th>SSv2-Sub</th>
<th>UB-S1</th>
</tr>
</thead>
<tbody>
<tr>
<td>No motion</td>
<td>48.2</td>
<td>28.2</td>
<td>40.1</td>
<td>84.1</td>
</tr>
<tr>
<td>Linear</td>
<td>55.5</td>
<td>34.6</td>
<td>45.3</td>
<td>88.5</td>
</tr>
<tr>
<td>Non-Linear</td>
<td>63.0</td>
<td>45.6</td>
<td>47.5</td>
<td>90.3</td>
</tr>
</tbody>
</table>

Table 3: Tubelet Motions. Learning from tubelets with non-linear motion benefits multiple downstream settings.

**Tubelet Transformation.** Table 4: Tubelet Transformation. Adding motion patterns to tubelet-contrastive learning through transformations improves downstream performance. Best results for rotation.

<table>
<thead>
<tr>
<th>Transformation</th>
<th>UCF (10^3)</th>
<th>Gym (10^3)</th>
<th>SSv2-Sub</th>
<th>UB-S1</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>63.0</td>
<td>45.6</td>
<td>47.5</td>
<td>90.5</td>
</tr>
<tr>
<td>Scale</td>
<td>65.1</td>
<td>46.5</td>
<td>47.0</td>
<td>90.5</td>
</tr>
<tr>
<td>Shear</td>
<td>65.2</td>
<td>47.5</td>
<td>47.3</td>
<td>90.9</td>
</tr>
<tr>
<td>Rotation</td>
<td>65.5</td>
<td>48.0</td>
<td>47.9</td>
<td>90.9</td>
</tr>
</tbody>
</table>

Table 4: Tubelet Transformation. Adding motion patterns to tubelet-contrastive learning through transformations improves downstream performance. Best results for rotation.

**Number of Tubelets.** Table 5: Number of Tubelets. Overlaying two tubelets in positive pairs improves downstream performance.

<table>
<thead>
<tr>
<th>#Tubelets</th>
<th>UCF (10^3)</th>
<th>Gym (10^3)</th>
<th>SSv2-Sub</th>
<th>UB-S1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>62.0</td>
<td>39.5</td>
<td>47.1</td>
<td>89.5</td>
</tr>
<tr>
<td>2</td>
<td>65.5</td>
<td>48.0</td>
<td>47.9</td>
<td>90.9</td>
</tr>
<tr>
<td>3</td>
<td>66.5</td>
<td>46.0</td>
<td>47.5</td>
<td>90.9</td>
</tr>
</tbody>
</table>

Table 5: Number of Tubelets. Overlaying two tubelets in positive pairs improves downstream performance.

**Transformation.** Table 4 compares the proposed tubelet transformations (Section 3.3). All four datasets benefit from transformations, with rotation being the most effective. The differences in improvement for each transformation are likely due to the types of motion present in the downstream datasets. For instance, Gym (10^3) and UB-S1 contain gymnastic videos where actors are often spinning and turning but do not change in scale due to the fixed camera, therefore rotation is more helpful than scaling. We also experiment with combinations of transformations in supplementary but observe no further improvement.

**Number of Tubelets.** We investigate the number of tubelets used in each video in Table 5. One tubelet is already more effective than temporal contrastive learning, e.g., 29.5% vs. 39.5% for Gym (10^3). Adding two tubelets improves accuracy on all datasets, e.g., +8.5% for Gym (10^3).

**Analysis of Motion-Focus.** To further understand what our model learns, Figure 5 visualizes the class agnostic activation maps [3] without finetuning for the baseline and our approach. We observe that even without previously seeing any FineGym data, our approach attends better to the motions than the temporal contrastive baseline, which attends to the background regions. This observation is supported by the linear classification and finetuning results on UCF101 and Gym99 (appearance-focused) and Gym99 (motion-focused) in Table 6. When directly predicting from the learned features with linear classification, our model is less effective than temporal contrast for appearance-based actions in UCF101, but positively affects actions requiring fine-grained motion understanding in Gym99. With finetuning, our tubelet-contrastive representation is able to add spatial appearance understanding and maintain its ability to capture temporal motion dynamics, thus it benefits both UCF101 and Gym99.

**Appearance vs Motion.** Our method learns to capture motion dynamics with pretraining and can easily learn appearance features with finetuning.

<table>
<thead>
<tr>
<th>Linear Classification</th>
<th>Finetuning</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCF101</td>
<td>Gym99</td>
</tr>
<tr>
<td>Temporal Contrast</td>
<td>58.9</td>
</tr>
<tr>
<td>Tubelet Contrast</td>
<td>30.0</td>
</tr>
</tbody>
</table>

Table 6: Appearance vs Motion. Our method learns to capture motion dynamics with pretraining and can easily learn appearance features with finetuning.
4.3. Video-Data Efficiency

To demonstrate our method’s data efficiency, we pretrain using subsets of the Kinetics-400. In particular, we sample 5%, 10%, 25%, 33% and 50% of the Kinetics-400 training set with three random seeds and pretrain our model and the temporal contrastive baseline. We compare the effectiveness of these representations after finetuning on UCF (10^3), Gym (10^3), SSv2-Sub, UB-S1, and HMDB51 in Figure 6. On all downstream setups, our method maintains similar performance when reducing the pretraining data to just 25%, while the temporal contrastive baseline performance decreases significantly. Our method is less effective when using only 5% or 10% of the data, but remarkably still outperforms the baseline trained with 100% data for Gym (10^3), UB-S1, and HMDB51. We attribute our model’s data efficiency to the tubelets we add to the pretraining data. In particular, our non-linear motion and transformations generate a variety of synthetic tubelets that simulate a greater variety of fine-grained motions than are present in the original data.

4.4. Standard Evaluation: UCF101 and HMDB51

We first show the effectiveness of our proposed method on standard coarse-grained action recognition benchmarks UCF101 and HMDB51, where we compare with prior video self-supervised works. For a fair comparison, we only report methods in Table 7 that use the R(2+1)D-18 backbone and Kinetics-400 as the pretraining dataset.

First, we observe that our method obtains the best results for UCF101 and HMDB51. The supplementary material shows we also achieve similar improvement with the R3D and I3D backbones. In particular, with R(2+1)D our method beats CtP [74] by 2.6% and 2.4%, TCLR [8] by 2.8% and 4.1%, and TE [30] by 2.8% and 1.9% all of which aim to learn finer temporal representations. This confirms that explicitly contrasting tubelet-based motion patterns results in a better video representation than learning temporal distinctiveness or prediction. We also outperform prior multi-modal works which incorporate audio or explicitly learn motion from optical flow. Since our model is data-efficient, we can pretrain on Mini-Kinetics and still outperform all baselines which are trained on the 3x larger Kinetics-400.

4.5. SEVERE Generalization Benchmark

Next, we compare to prior works on the challenging SEVERE benchmark [70], which evaluates video representations for generalizability in domain shift, sample efficiency, action granularity, and task shift. We follow the same setup as in the original SEVERE benchmark and use an R(2+1)D-18 backbone pretrained on Kinetics-400 with our tubelet-contrast before finetuning on the different downstream settings. Results are shown in Table 8.

### Domain Shift

Among the evaluated methods our proposal achieves the best results on SSv2 and Gym99. These datasets differ considerably from Kinetics-400, particularly...
in regard to the actions, environment and viewpoint. Our improvement demonstrates that the representation learned by our tubelet-contrast is robust to various domain shifts.

**Sample Efficiency.** For sample efficiency, we achieve a good gain over all prior works on Gym (10^3), e.g., +20.7% over TCLR [8] and +14.1% over CtP [74]. Notably, the gap between the second best method GDT [57] and all others is large, demonstrating the challenge. For UCF (10^3), our method is on par with VideoMoCo [56] and CtP but is outperformed by GDT and RSPNet [58]. This is likely due to most actions in UCF101 requiring more spatial than temporal understanding, thus it benefits from the augmentations used by GDT and RSPNet. Our motion-focused representation requires more finetuning samples on such datasets.

**Action Granularity.** For fine-grained actions in FX-S1 and UB-S1, our method achieves the best performance, even outperforming supervised Kinetics-400 pretraining. We achieve a considerable improvement over other RGB-only models, e.g., +19.6% and +6.3% over TCLR, as well as audio-visual models, e.g., +14.1% and +7.6% over GDT. These results demonstrate that the video representation learned by our method are better suited to fine-grained actions than existing self-supervised methods. We additionally report results on Diving48 [41] in the supplementary.

**Task Shift.** For the task shift to repetition counting, our method is on par with AVID-CMA [51] and RSPNet, but worse than GDT. For multi-label action recognition on Charades, our approach is 3rd, comparable to VideoMoCo but worse than TCLR. This suggests the representations learned by our method are somewhat transferable to tasks beyond single-label action recognition. However, the remaining gap between supervised and self-supervised highlights the need for future work to explore task generalizability further.

**Comparison with Transformers.** Table 8 also contains recent transformer-based self-supervised works SVT [61] and VideoMAE [71]. We observe that both SVT and VideoMAE have good performance with large amounts of finetuning data (SSv2), in-domain fine-tuning (UCF(10^3)), and multi-label action recognition (Charades). However, they considerably lag in performance for motion-focused setups Gym99, FX-S1, UB-S1, and repetition counting compared to our tubelet contrast with a small CNN backbone.

## Overall SEVERE Performance
Finally, we compare the mean and the average rank across all generalizability factors. Our method has the best mean performance (66.5) and achieves the best average rank (4.1). When pretraining with the 3x smaller Mini-Kinetics our approach still achieves impressive results. We conclude our method improves the generalizability of video self-supervised representations across these four downstream factors while being data-efficient.

## 5. Conclusion
This paper presents a contrastive learning method to learn motion-focused video representations in a self-supervised manner. Our model adds synthetic tubelets to videos so that the only similarities between positive pairs are the spatiotemporal dynamics of the tubelets. By altering the motions of these tubelets and applying transformations we can simulate motions not present in the pretraining data. Experiments show that our proposed method is data-efficient and more generalizable to new domains and fine-grained actions than prior self-supervised methods.

### Acknowledgements
This work is part of the research programme Perspectief EDL with project number P16-25 project 3, which is financed by the Dutch Research Council (NWO) domain Applied and Engineering/Sciences (TTW). We thank Piyush Bagad for help with experiments and Artem Moskalev for useful discussions.
References


