Abstract

We present a large-scale study on unsupervised spatiotemporal representation learning from videos. With a unified perspective on four recent image-based frameworks, we study a simple objective that can easily generalize all these methods to space-time. Our objective encourages temporally-persistent features in the same video, and in spite of its simplicity, it works surprisingly well across: (i) different unsupervised frameworks, (ii) pre-training datasets, (iii) downstream datasets, and (iv) backbone architectures. We draw a series of intriguing observations from this study, e.g., we discover that encouraging long-spanned persistency can be effective even if the timespan is 60 seconds. In addition to state-of-the-art results in multiple benchmarks, we report a few promising cases in which unsupervised pre-training can outperform its supervised counterpart. Code will be made available at https://github.com/facebookresearch/SlowFast.

Figure 1. Learning to maximize the similarity between different temporal clips of the same video encourages feature persistency over time. A query clip (q) is matched to multiple key clips (k_1, k_2, ...) that are temporally shifted. This method can be incorporated into several unsupervised learning frameworks (MoCo [36], SimCLR [12], BYOL [32], SwAV [9]). The figure on the top shows that increasing the number (ρ) of temporal clips improves representation quality for all these frameworks.

to be similar. We empirically find that this objective works well across different unsupervised frameworks (MoCo [36], SimCLR [12], BYOL [32], SwAV [9]), either with or without using dissimilar (negative) samples.

Our objective is a natural generalization of crops in images [18, 90] to clips in videos. This allows us to make use of the recent unsupervised learning frameworks with minimal modifications. We aim to learn a high-level representation of the categorical semantics present in a video by enforcing persistency of the representation over space-time. We investigate factors such as the effective timespan, t, between positives, and number of temporal clips, ρ, to find that longer timespans (up to a minute) and multiple samples are beneficial for downstream performance (Fig. 1).

Our unsupervised training is performed on large-scale data, including Kinetics [47] (240k videos) and three versions of million-scale Instagram sets. In addition to standard linear probing, we evaluate representation quality on multiple classification and detection downstream datasets, e.g., Charades [76], Something-Something [31], andAVA [33].
Our results suggest that unsupervised pre-training can achieve competitive performance in videos, and it can surpass the supervised pre-training counterparts in a few cases. Finally, our study also reveals room for improvement along multiple directions.

In summary, our large-scale study involves the following five aspects:

(i) Four unsupervised learning frameworks (MoCo [36], SimCLR [12], BYOL [32], SwAV [9]) viewed from a unified perspective, and incorporated with a simple temporal persistency objective;

(ii) Three pre-training datasets, including the relatively well-controlled Kinetics [47] and the relatively “in-the-wild” Instagram sets at million-scale;

(iii) Six downstream datasets/tasks for evaluating representation quality;

(iv) Ablation experiments on different factors, such as temporal samples, contrastive objective, momentum encoders, training duration, backbones, data augmentation, curated vs. uncurated, trimmed vs. untrimmed, etc.; and

(v) State-of-the-art results of unsupervised video representation learning on established benchmarks, UCF-101 [78], HMDB51 [50] and Kinetics-400 [47].

2. Related Work

Unsupervised learning in images has been actively researched recently with approaches focusing on various pretext tasks related to color- or patch-based processing [68, 96, 17, 65], instance discrimination with contrastive objectives [18, 90, 84, 40, 41, 46, 36, 97, 12, 82] and ones that focus on positive pairs [8, 9, 32].

Unsupervised learning in videos has followed a similar trajectory with earlier methods focusing on predictive tasks based on motion, color and spatiotemporal ordering [29, 43, 1, 44, 79, 86, 61, 85, 59, 58, 21, 51, 87, 67, 22, 48, 93, 16, 88, 71, 45], and contrastive objectives with visual [75, 80, 34, 53, 28, 94] and audio-visual input [66, 4, 5, 49, 3, 69, 70].

Several recent ones [28, 35, 3, 69, 2, 72, 94, 63] relate to image-based approaches [36, 8, 12, 90]. With some of them using additional modalities of optical-flow [82, 35], audio [3, 69, 2, 63] and text [80, 2] to transfer supervision from one modality to another.

In relation to these previous efforts, our work studies purely visual unsupervised learning from video and tries to compare the meta-methodologies on common ground.

Evaluation protocols and backbones in most image-based approaches have converged to ResNet-50 [39] encoders with ImageNet linear-classification protocol, and several smaller downstream tasks [36, 12, 32, 9] for evaluation. In video understanding research, the field has not yet converged and is using different backbones with focus on fine-tuning performance on two relatively small datasets [78, 50]. We investigate this aspect by looking at different encoders and 6 different downstream benchmarks for evaluation.

3. Approach

The objective of this work is to study several recent unsupervised representation learning methodologies to train a spatiotemporal encoder \( f_\theta \), exploring implementation details and comparing them on a common ground to measure their efficacy in video understanding. We focus on two contrastive approaches using positive and negative samples: SimCLR [12] and MoCo [36], as well as two approaches that solely rely on positives, BYOL [32] and SwAV [9] (Sec. 3.2).

These approaches were originally presented for learning image representations, and they all share the objective of learning invariant features across different views (crops/augmentations) of the spatial image input. In this paper, this idea is extended to the temporal domain. Our core idea is to learn an encoder \( f_\theta \) that produces embeddings which are persistent in space-time, over multiple \( (\rho) \) temporally distant clips of the same video. This is related to Slow Feature Analysis [89] where the objective is to minimize the representations’ temporal derivative over the input. The general idea of learning temporally persistent features is not new and has been proposed in the past with similar motivation e.g., [6, 62, 29].

3.1. Persistent temporal feature learning

Our framework takes different augmented clips \( x \) of an unlabeled video and passes them through an encoder \( f_\theta \) with weights \( \theta \) to obtain corresponding embeddings \( q = f_\theta(x) \). The encoder is spatiotemporal ConvNet, by default a ResNet-50 (R-50) [39], Slow-only pathway of SlowFast Networks [20], which is a 3D ResNet-50 [39] without temporal pooling in convolutional feature maps, followed by an MLP projection head, that produces and output of dimension \( d \).

The input clips are stacks of RGB frames of size \( 3 \times T \times S^2 \) for temporal \( \times \) spatial dimensions, which are sampled with temporal stride \( \tau \), i.e., the encoder processes only one out of \( T \) frames of the raw video. Therefore, \( T \times \tau \) define the timespan and resolution of the encoder.

Given a minibatch of \( B \) videos, our framework creates a set of \( \rho B \) positive examples by sampling \( \rho \) clips from the videos. The learning methodologies studied in this section maximize similarity of a “query” sample \( q \) with a set of positive “key” samples \( \{ k^+ \} \) that are encoded versions of different clips of the same video as \( q \) is computed from. Fig. 1 illustrates an example where \( \rho=3 \) clips are used.

The next section describes how the contrastive and non-contrastive unsupervised representation learning methodologies are exemplified.
3.2. Unsupervised learning frameworks

Contrastive learning maximizes the similarity of a sample \( q \) with positive ones \( \{k^+\} \) and minimizes similarity to negative ones \( \{k^-\} \). The contrastive approaches in this paper use the InfoNCE [84] objective,

\[
\mathcal{L}_q = - \log \frac{\sum_{k \in \{k^+\}} \exp \left( \frac{\text{sim}(q, k)}{\alpha} \right)}{\sum_{k \in \{k^+, k^-\}} \exp \left( \frac{\text{sim}(q, k)}{\alpha} \right)},
\]

with \( \alpha \) being a temperature hyper-parameter for scaling and \( \{k^+\} \) are embedded clips of the same video as \( q \). All the embeddings are \( \ell_2 \) normalized and dot product (cosine) similarity is used to compare them \( \text{sim}(q, k) = q^\top k / \|q\| \|k\| \).

SimCLR [12] (Fig. 2a) uses the embeddings of clips of other videos in the minibatch as negatives \( \{k^-\} \).

MoCo [36] (Fig. 2b) is a method that uses an explicit momentum encoder which parameters, \( \theta_m \), are moving average \( \theta_m \leftarrow m\theta_m + (1 - m)\theta \) with \( m \) a momentum parameter. In eq. (1) MoCo uses this encoder to compute the positive embeddings \( \{k^+\} \) from clips of the same video as \( q \), and negative embeddings \( \{k^-\} \) are taken from a queue that stores embeddings of clips from previous iterations. There is no backpropagation into the momentum-encoder weights \( \theta_m \).

BYOL [32] (Fig. 2c) can be viewed as a form of MoCo that does not use negative samples, but an extra predictor MLP with weights \( \theta_p \), which is stacked on top of \( f_0 \)’s MLP head. For a sample \( q = f_{\theta_p}(f_0(x)) \), BYOL minimizes negative cosine similarity,

\[
\mathcal{L}_q = - \sum_{k \in \{k^+\}} \text{sim}(q, k) = - \sum_{k \in \{k^+\}} q^{\top} k^+ / \|q\| \|k^+\|, \tag{2}
\]

with \( \{k^+ = f_{\theta_m}(x^+)\} \) being embedded clips \( x^+ \) from the same video as \( q \), encoded with momentum weights \( \theta_m \).

SwAV [9] (Fig. 2d) can be viewed as a form of SimCLR that does not use negative samples. SwAV first performs a linear mapping of the positive embeddings \( q, k^+ \) to learned prototypes \( \tilde{q}, \tilde{k}^+ \) and then transforms the targets with an extra Sinkhorn-Knopp (SK) step. Then the SwAV loss is

\[
\mathcal{L}_q = D_{\text{KL}}(\tilde{q} \| S\text{K}(\tilde{k}^+)),
\]

where \( D_{\text{KL}} \) is the The Kullback-Leibler divergence and gradients are not back-propagated through the SK operation.

Compared to SimCLR and MoCo, in BYOL and SwAV, \( q \) and \( k \) are not typical “query” and “key” samples (but rather “source” and “target” samples); however, for consistency we use \( q, k \) terminology in notation for all methods.

Implementation specifics. We implement the methods with a symmetric loss, as in original SimCLR, BYOL and SwAV, where every input clip is used to produce a loss (and gradient) signal. For each of the \( \rho \geq 2 \) clips, we compute \( q \), while all other \( \rho - 1 \) clips of the same video are used as \( \{k^+\} \) to evaluate sub-loss \( \mathcal{L}_q \) and the symmetric loss is the average over all \( \rho \) sub-losses. Thus, for MoCo and BYOL, every input clip is processed by both encoders.

For MoCo and BYOL, our symmetric loss is aggregated sequentially which implies that memory consumption for \( \rho > 2 \) equals to a single clips’ forward and backward pass, since these methods do not back propagate through the momentum encoder. For SimCLR and SwAV the overall loss is evaluated in parallel across all clips and therefore memory consumption grows linearly with the number of clips used.

All details on implementation and pre-training are in §B.1.

4. Experiments

Datasets. Unless otherwise noted, we perform unsupervised pre-training on Kinetics-400 [47] (K400) with ~240k training videos in 400 human action categories.
Table 1. Pre-training data statistics with timings in seconds.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#videos</th>
<th>$t_{\text{median}}$</th>
<th>$t_{\text{mean}}$</th>
<th>$t_{\text{std}}$</th>
<th>$t_{\text{min}}$</th>
<th>$t_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kinetics-400 (K400)</td>
<td>240K</td>
<td>10.0</td>
<td>9.3</td>
<td>1.7</td>
<td>1.0</td>
<td>10.0</td>
</tr>
<tr>
<td>IG-Curated [24]</td>
<td>1M</td>
<td>18.9</td>
<td>26.3</td>
<td>19.8</td>
<td>1.5</td>
<td>60.0</td>
</tr>
<tr>
<td>IG-Uncurated</td>
<td>1M</td>
<td>29.4</td>
<td>35.3</td>
<td>38.4</td>
<td>0.5</td>
<td>600.0</td>
</tr>
<tr>
<td>IG-Uncurated-Short</td>
<td>1M</td>
<td>13.0</td>
<td>13.1</td>
<td>1.6</td>
<td>10.0</td>
<td>15.9</td>
</tr>
</tbody>
</table>

Table 2. Number of temporal clips $\rho$. Data: K400, 200 epochs. Learning temporally persistent features ($\rho \geq 2$) is effective.

<table>
<thead>
<tr>
<th>ep</th>
<th>MoCo K400</th>
<th>BYOL K400</th>
<th>SimCLR K400</th>
<th>SwAV K400</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>61.0</td>
<td>90.8</td>
<td>60.6</td>
<td>91.2</td>
</tr>
<tr>
<td>2</td>
<td>65.8</td>
<td>91.0</td>
<td>65.8</td>
<td>92.7</td>
</tr>
<tr>
<td>3</td>
<td>67.3</td>
<td>92.8</td>
<td>68.3</td>
<td>93.8</td>
</tr>
<tr>
<td>4</td>
<td>67.8</td>
<td>93.5</td>
<td>68.9</td>
<td>93.8</td>
</tr>
</tbody>
</table>

Table 3. Training duration in epochs (ep): Dataset: K400, $\rho=2$. Training longer brings consistent gains for all methods up to 400 epochs and saturates for K400 but not for UCF101 at 800ep. SwAV is the strongest performer for short training (50ep).

Temporal augmentation. The first row in Table 2, $\rho=1$, uses two spatial crops at the same temporal instance, while the $\rho=2$ row uses clips at different temporal locations as positives; therefore, learns persistent features in time. This difference has a large impact on performance, especially for SimCLR ($60.5 \rightarrow 36.1$) and SwAV ($61.6 \rightarrow 38.6$) performance degrades significantly when sampling positives from the same temporal instance ($\rho=1$).

More clips are beneficial. The remaining rows in Table 2 show that accuracy is further increasing with the number of temporal samples per video, e.g. at $\rho=4$ the best accuracy is achieved with BYOL at 68.9% K400 and 93.8% UCF101.

Negatives do not help but momentum encoders do. When comparing the methods in Table 2, we see that:

(i) There is no clear performance difference between contrastive/non-contrastive methods. This indicates that learning space-time persistence within a video is key for the methods, but learning in-persistence across videos is not.

(ii) There is a clear difference of $\sim 4\%$ on K400 between methods that employ momentum encoders (MoCo, BYOL), vs. these that do not (SimCLR, SwAV).

Increasing the number of clips per training iteration increases training cost, so it is reasonable to compare it to training more epochs. Table 3 is studying the base case $\rho=2$ for various number of epochs (ep).

Overall, the results show that there is a clear gain for training longer which has been also observed in image-related tasks [12, 36, 32, 9]. BYOL performs the worst when training short durations. This might be related to hyper-parameter settings which we do not adjust for this experiment (the original implementation [32] uses different hyper-parameters for different number of training epochs).
4.2. Timespan between positives

All experiments with $\rho \geq 2$ so far were using global temporal sampling of positives, which means that the clips can be sampled at unconstrained temporal locations from the input video. This might be counter-productive because if there is a long duration that has passed between a pair of positive clips they might no longer share the same semantic context for learning high-level features corresponding in time.

<table>
<thead>
<tr>
<th>$t_{\text{max}}$ in seconds</th>
<th>0</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>8</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>K400 acc in %</td>
<td>60.6</td>
<td>65.2</td>
<td>65.7</td>
<td>65.8</td>
<td>65.8</td>
<td>65.6</td>
<td>65.8</td>
</tr>
</tbody>
</table>

(a) Dataset: K400, 200 epochs training.

<table>
<thead>
<tr>
<th>$t_{\text{max}}$ in seconds</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>K400 acc in %</td>
<td>62.7</td>
<td>63.1</td>
<td>63.1</td>
<td>63.9</td>
<td>64.1</td>
</tr>
</tbody>
</table>

(b) Dataset: IG-Curated-1M, 50 epochs training.

<table>
<thead>
<tr>
<th>$t_{\text{max}}$ in seconds</th>
<th>12s</th>
<th>24</th>
<th>36</th>
<th>48</th>
<th>600</th>
</tr>
</thead>
<tbody>
<tr>
<td>K400 acc in %</td>
<td>59.3</td>
<td>59.2</td>
<td>59.9</td>
<td>59.6</td>
<td>58.9</td>
</tr>
</tbody>
</table>

(c) Dataset: IG-Uncurated-1M, 50 epochs training.

Table 4. Maximum frame distance for positives. Method: BYOL, $\rho = 2$. Training is surprisingly robust with increasing accuracy for increased distance between samples. Accuracy only (mildly) degrades when sampling positives that are more than 36 seconds apart when using uncurated (random) videos.

This experiment is concerned with the maximum distance between the positive training samples. We use BYOL pre-training on K400, IG-Curated-1M and IG-Uncurated-1M and report 400 linear readout accuracy in Table 4.

Table 4a shows performance for increasing the maximum temporal distance between positives in K400 pre-training. It can be seen that using positives from the same time ($t_{\text{max}} = 0$) degrades performance by 5% but other than that performance is relatively robust up to global sampling of positive clips from the whole video ($t_{\text{max}} = 10$s). This is interesting as it seems that a long-temporal correspondence objective does not hurt performance (but also does not boost it).

Table 4b shows performance for increasing the temporal distance between positive samples on IG-Curated-1M. This dataset has a maximum duration of 60 seconds; statistics are in Table 1. Table 4b shows that increasing the maximum duration between positive pairs is beneficial for performance and unrestricted sampling of positives is the best with 64.1% top-1 accuracy for evaluation on K400. This is especially interesting, as it shows that even longer videos benefit from global sampling. There is no benefit from restricting the time window of positives, which can be interpreted as the objective of learning extremely-slow features [89] that do not change over 60 seconds of video. Long-temporal-distance samples might also increase robustness of the model by providing “hard-positive” samples for learning. Note that here the videos are still sampled according to hashtags related to K400 classes [24]; therefore, the conjecture might be biased.

Finally, we are looking at the IG-Uncurated-1M dataset which consists of a random sampling of 1M videos from Instagram. These videos can be between 0.5s and 10 minutes of duration. Most of the videos however are much shorter than 10 minutes, with a mean duration of 35.3 seconds and a standard deviation of 38.4 seconds (Table 1). For this data, Table 4c shows the results of progressively increasing the maximum timespan between positive samples. It can be observed that increasing the maximum distance between positives up to 36 seconds is beneficial and beyond that performance decreases, but only slightly, even when performing global sampling of positives (the default).

4.3. Backbone architectures

So far all experiments were using a R-50, 8×8 Slow pathway [39, 20] as backbone. The next set of ablations studies different architectures for the spatiotemporal encoder.

<table>
<thead>
<tr>
<th>backbone</th>
<th>$T \times \tau$</th>
<th>FLOPs</th>
<th>Param</th>
<th>s/iter</th>
<th>sup. K400</th>
<th>MoCo ($\rho=2$) K400</th>
<th>UCF101</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-50</td>
<td>8×8</td>
<td>41.7G</td>
<td>31.8M</td>
<td>1.6s</td>
<td>74.7</td>
<td>65.8</td>
<td>91.0</td>
</tr>
<tr>
<td>R-18</td>
<td>8×8</td>
<td>20.0G</td>
<td>20.2M</td>
<td>1.2s</td>
<td>68.9</td>
<td>56.2</td>
<td>87.1</td>
</tr>
<tr>
<td>R-101</td>
<td>8×8</td>
<td>93.3G</td>
<td>51.4M</td>
<td>2.1s</td>
<td>75.8</td>
<td>67.7</td>
<td>92.4</td>
</tr>
<tr>
<td>R-50</td>
<td>16×4</td>
<td>83.5G</td>
<td>31.8M</td>
<td>2.5s</td>
<td>76.1</td>
<td>67.6</td>
<td>93.3</td>
</tr>
<tr>
<td>R-50</td>
<td>32×2</td>
<td>167.0G</td>
<td>31.8M</td>
<td>4.6s</td>
<td>76.3</td>
<td>67.3</td>
<td>94.2</td>
</tr>
<tr>
<td>R2+1D-18</td>
<td>32×2</td>
<td>48.5G</td>
<td>15.4M</td>
<td>4.0s</td>
<td>71.7</td>
<td>57.2</td>
<td>93.7</td>
</tr>
<tr>
<td>S3D-G</td>
<td>32×2</td>
<td>36.0G</td>
<td>9.1M</td>
<td>4.1s</td>
<td>74.7</td>
<td>63.2</td>
<td>94.5</td>
</tr>
</tbody>
</table>

Table 5. Backbone comparison. The ResNet [39] backbone (Slow pathway [20]) is used with different depth (R-18, R-50, R-101), input frames $T$ and stride $\tau$. R2+1D [83] and S3D-G [91] are commonly used backbones for unsupervised video representation learning with downstream evaluation on UCF101.

Table 5 compares different backbones for usage with MoCo in our default setting ($\rho=2, 200$ epoch pre-training on K400). From left to right, the table shows the input duration $T$, sampling-rate $\tau$, FLOPs (at 224×224 spatial resolution) and parameters of these backbones, as well as the average duration for training one iteration of the MoCo algorithm (measured on a single machine with 8 V100 GPUs in PySlowFast [19] and torchvision decoder), the supervised performance on K400 and UCF101 (finetuned from K400), as well as the downstream performance for K400 linear evaluation and UCF101 finetuning.

The first observation in Table 5 is that for the Slow architecture [20], using shallower (R-18) or deeper (R-101) networks can influence supervised and downstream performance in a sizable manner, with MoCo, K400 evaluation benefiting from more parameters. Doubling the input frame-rate (8×8 → 16×4) boosts accuracy on UCF101.

The second observation is that R2+1D [83] has a large gap on Kinetics (71.7% supervised vs. 57.2% unsupervised), while being remarkably strong on UCF101 (93.7%). This gap is also observed for S3D-G [91]. The reason for this might be that UCF101 is a small dataset which is easy to overfit and can benefit from fewer parameters.
4.4. Uncurated data and video duration

In Table 6 we show the performance of all four methodologies on IG-Curated-1M (a), IG-Uncurated-1M (b) and IG-Uncurated-Short-1M (c) for pre-training with 50 and 200 epochs. We make the following observations:

(i) Among the methods MoCo performs the best with e.g. 69.0% vs. second-best 64.3% of SwAV on curated data (a).

(ii) MoCo and SwAV scale the best for training longer, gaining roughly 3-4% for 200ep vs. 50ep.

(iii) On uncurated data, MoCo and SwAV perform ~1% better on the unconstrained duration videos in Table 6b.

(iv) BYOL and SimCLR show better performance on IG-Uncurated-Short (10-16s videos) in Table 6c, seemingly benefiting from shorter videos, but there is no clear benefit from either longer or shorter duration among all methods.

(v) BYOL degrades performance for training longer which might be due to the requirement of different hyper-parameters for different schedules (as noted in Sec. 4.1).

We will return to this point in §A.1, where we show that increasing clips-size $\rho$ can overcome this issue in BYOL, along with further studies on the trade-off against training more epochs, and dataset scale.

4.5. Data augmentations

Importance of augmentations. Augmentations can have a major impact on visual unsupervised feature learning [12, 14]. In Fig. 3, we ablate spatial cropping (S), temporal clipping (T) and radiometric color (C) augmentations from the four unsupervised learning methods (e.g. “T S C” are the baselines using all augmentations and removing “S C” equals $\rho=1$ in Table 2). We make three main observations:

(i) Among the methods, MoCo and BYOL perform most robust for using fewer augmentations; their advantage over SimCLR and SwAV might be related to the momentum encoder which can provide extra augmentation in training.

(ii) When minimizing the augmentations by resizing the shorter size of the video to the input size of 224 and only cropping along the long side of the video (Base in Fig. 3), MoCo still provides 42.2% K400 linear accuracy, over BYOLs’ 32.4%, showing an advantage of the contrastive loss in a weak augmentation scenario.

(iii) Among the augmentations, learning temporal (T) persistence, has the largest impact on performance, except for MoCo which benefits more from color (C) (incl. grayscale) augmentations. Especially SimCLR and SwAV show significant drops in performance when removing T, i.e. when extracting positive clips from the same instance in time.

In the remainder of this section, we explore using stronger augmentations than the default ones in previous experiments. We perform the ablations with MoCo in the basic setting of $\rho=2$, 200 epochs K400 pre-training.

Table 7. Radiometric augmentation. Method: MoCo. 200 epochs, $\rho=2$. Dataset: K400. Stronger color augmentation in K400 pre-training can especially benefit UCF101 (+1.3%).

<table>
<thead>
<tr>
<th>color strength</th>
<th>grayscale probability</th>
<th>temporal difference</th>
<th>fps jitter</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.2</td>
<td>✓</td>
<td>66.2</td>
<td>91.3</td>
</tr>
<tr>
<td>0.5</td>
<td>0.2</td>
<td>✓</td>
<td>65.6</td>
<td>91.5</td>
</tr>
</tbody>
</table>

Stronger color augmentation. In Table 7 color strength of 0.5 indicates the default one for MoCo [14], 0.75 and 1.0 increase the strength of randomly jittering brightness, contrast, saturation and hue proportionally.

Table 7 shows that increasing it to 0.75 can improve K400/UCF101 accuracy. Increasing the random grayscale probability from 0.2 to 0.4 does not provide an improvement on either of the datasets. However, using a temporal difference augmentation which randomly (with probability 0.2) first converts the frames to grayscale and then subtracts difference augmentation which randomly (with probability 0.5) indicates the default one for MoCo [14], 0.75 and 1.0 increase the strength of randomly jittering brightness, contrast, saturation and hue proportionally.
Spatial cropping. Our default implementation uses VGG-style [77, 39] cropping that randomly resizes the shorter spatial side of a video between [256, 320] pixels and takes a random 224² crop extended over time to extract a clip [20].

Since unsupervised learning might benefit from more aggressive cropping, we explore Inception-style [81] cropping with aspect ratio augmentation that is commonly used in unsupervised learning from images [36, 12, 32, 9]. This cropping procedure randomly resizes the input area between a minimum scale and a maximum scale and jitters aspect ratio between 3/4 to 4/3, before taking a 224² crop.

We do not change the cropping for downstream training, as this can drop accuracy significantly (by ~2% on K400).

In Table 8 we ablate this approach for MoCo (the augmentation in the downstream evaluators are unchanged).

The first ablation shows the comparison of default cropping [77, 39] with a similar version that randomly crops a fraction between [0.49, 0.76] = [224²/320², 224²/256²] of the original area, instead of the short-side. The performance degrades by 1% on K400 linear evaluation. Randomly cropping based on area favors larger crops over the short-side resizing and we observe lower training error for this variant.

Next, adding aspect ratio augmentation can recover some of this performance (65.4%), and using a smaller minimum area of 0.2, with the maximum area of 0.76 leads to best performance of 66.8%. Using the default values for Inception [81] training, [0.08, 1.00], appears to be too aggressive.

<table>
<thead>
<tr>
<th>area</th>
<th>aspect ratio</th>
<th>K400</th>
<th>UCF101</th>
</tr>
</thead>
<tbody>
<tr>
<td>min, max</td>
<td>(x/y)</td>
<td>accuracy</td>
<td></td>
</tr>
<tr>
<td>default [77, 39, 20]</td>
<td>65.8</td>
<td>91.9</td>
<td></td>
</tr>
<tr>
<td>0.49, 0.76</td>
<td>✓</td>
<td>64.8</td>
<td>91.7</td>
</tr>
<tr>
<td>0.49, 0.76</td>
<td>✓</td>
<td>65.4</td>
<td>91.7</td>
</tr>
<tr>
<td>0.20, 0.76</td>
<td>✓</td>
<td><strong>66.8</strong></td>
<td><strong>91.8</strong></td>
</tr>
<tr>
<td>0.20, 0.50</td>
<td>✓</td>
<td>66.3</td>
<td>91.8</td>
</tr>
<tr>
<td>0.20, 1.00</td>
<td>✓</td>
<td>66.6</td>
<td>91.7</td>
</tr>
<tr>
<td>0.08, 0.50</td>
<td>✓</td>
<td>64.3</td>
<td>91.6</td>
</tr>
<tr>
<td>0.08, 1.00</td>
<td>✓</td>
<td>65.3</td>
<td>91.2</td>
</tr>
</tbody>
</table>

Table 8. Cropping augmentation. Method: MoCo, 200 epochs, ρ = 2. Dataset: K400. Stronger cropping and aspect ratio augmentation can be beneficial by +1.0% (K400) and 0.7% UCF101.

4.6. Alternative downstream tasks

The gap between K400 and UCF101 accuracy in Sec. 4.3 question if solely looking at typical evaluation of UCF101 (or the smaller HMDB51) is enough to identify and rank approaches for unsupervised learning in video.

Table 10 studies several new downstream tasks for unsupervised representation learning in video. We use our MoCo, SimCLR, BYOL and SwAV models trained with ρ=3 for 200 epochs on K400 and evaluate their performance by finetuning on Charades [76], AVA [33], or Something-Something [31] (in addition to the K400 linear readout performance and UCF101 performance reported in Table 2). Details on implementation are given in §8B.

The first two rows in Table 10 show the two main competitors for this evaluation: (i) training from scratch on the datasets and (ii) K400 pre-training. First, we observe that the supervised pre-trained backbones outperform the train-from-scratch counterpart significantly, as expected.

Downstream datasets. For K400 pre-training and linear evaluation, its supervised counterpart has an advantage between 12.7% and 6.4% top-1 accuracy among the methods.

On UCF101 unsupervised pre-training is only 1% lower than the supervised counterpart for BYOL (the strongest).

On AVA short-term action detection we observe that the BYOL pre-trained model is able to outperform the supervised counterpart by +1.2% mAP, when using the same, fixed region proposals [20]. This result is significant, as e.g. switching from K400 to K600 (nearly double the size of K400) pre-training on AVA leads to a smaller gains in performance [20]. Overall this is a surprising result as the tasks in K400 and AVA are similar [52], only that the temporal granularity of the actions in AVA is finer while their semantic granularity is coarser; e.g. “shoot” in AVA vs. “playing paintball” in Kinetics, which might be better captured by the BYOL objective which solely works on positive temporal samples of a video, without contrasting them to other videos (“shoot” might be a positive appearing in many different videos and contrasting them could be harmful to downstream performance). This line of thinking is supported with MoCo’s (contrastive objective) performance that is 3.1% worse than BYOL on AVA. Similarly, SimCLR (contrastive) is worse than SwAV (non-contrastive) when benchmarked on AVA.

On Charades, long-term action classification, we observe the opposite. Here, the contrastive MoCo is clearly the best performer with 33.5% mAP (close to the supervised pre-training performance of 34.7% mAP), while the non-contrastive BYOL is 12.5% lower. Similarly, now SimCLR (contrastive) is better than SwAV (non-contrastive). Compared to AVA, Charades is a temporally less localized dataset containing activities that need to be recognized from a longer temporal range video, for which contrastive pre-training appears to be outperforming the non-contrastive variants.
On Something-Something v2 (SSv2 in Table 10), all the methods perform strong, with BYOL pre-training showing the largest gain of +3% over supervised pre-training on Kinetics (55.8% vs. 52.8% top-1 accuracy).

**Pre-training sets: Kinetics vs. IG.** Next, we experiment with pre-training on videos from the web. We first investigate IG-Curated-1M [24], which is a dataset that has been collected with hashtags that are similar to Kinetics labels. This data is a 1M subset of the original 65M introduced in [24]. Using this data (penultimate row in Table 10) can excels the performance of MoCo with K400 pre-training, which has a training set of 240K samples (roughly $4.2 \times$ smaller), and surprisingly even outperforms pre-training on K400 linear readout itself (69.9% vs. 67.3% accuracy).

Second, we ablare the effect of using uncured videos, with IG-Uncurated-1M which are purely random videos taken from the web. On most downstream tasks performance shown in the last row of Table 10 is equal or only slightly lower than pre-training on K400. Specifically, MoCo changes by -1.3% on K400 (as expected), +0.1% on UCF, +0.2% on AVA, -2.2% on Charades and -1.2% on Something-Something v2. This is an encouraging result for unsupervised learning, as only ~4.2$\times$ the number of videos but random ones are required to match the performance of supervised K400 pre-training on the UCF101 and AVA.

Overall, our results indicate that unsupervised pre-training can be a new paradigm for all of these downstream tasks, for which supervised pre-training is the de-facto standard to achieve best performance. Further, the large difference in performance for pre-training methodologies and objectives (e.g. contrastive/non-contrastive) revealed in the light of these benchmarks signals large room for future work.

### 4.7. Comparison to previous work

In a final experiment we take the best model from Table 9 and compare it with the state-of-the-art using the commonly used protocols on UCF101 and HMDB51 (across all 3 train/val splits) and K400. In Table 11 we show the results.

The strongest previous approaches are using multi-modal input, Vision “V”, Audio “A”, Text “T”, to train a contrastive objective across modalities; XDC [3] performs DeepCluster [8] on (V+A), CVRL [72], GDT [69] and MMV [2] use an objective similar to SimCLR on (V), (V-A), and (V+A+T), with the latter training on a Audioset (AS) [23] and HowTo100M (HT) [60], and CoCLR [35] can be seen as a variant of MoCo on rgb and optical-flow input.

In comparisons, our best performing model pBYOL, which is BYOL trained with temporal persistency over $\rho=4$ clips, (cf. Tables 2 & 9), provides a substantial performance gain over the best published method [35]: +5.7% and +12.1% top-1 accuracy on UCF101 and HMDB51 (using identical backbone and pre-training data).

On K400 linear evaluation with the same data and R-50, Slow pathway [20] as backbone, our approach outperforms the previous best CVRL [72] by +5.4% accuracy.

### 5. Conclusion

This paper has studied four meta-methodologies for unsupervised learning from video. Our findings include that it is beneficial to sample positives with longer timespans between them, contrastive objectives are less influential than momentum encoders, and training duration, backbones, video augmentation and curation are all critical for good performance. Our resulting models which learn persistent features across augmented spacetime clips set a new state-of-the-art.

We observed that linear readout on Kinetics is a good indicator of the performance on other datasets and that unsupervised pre-training can compete with the supervised counterpart on several datasets, but there is room for improvement. We hope that our baselines will foster research and provide common ground for future comparisons.

<table>
<thead>
<tr>
<th>method</th>
<th>pre-train</th>
<th>backbone</th>
<th>$T$</th>
<th>mod</th>
<th>UCF</th>
<th>HMDB</th>
<th>K400</th>
</tr>
</thead>
<tbody>
<tr>
<td>XDC [3]</td>
<td>K400</td>
<td>R(2+1)D-18</td>
<td>15.4M</td>
<td>32</td>
<td>V+A</td>
<td>84.2</td>
<td>47.1</td>
</tr>
<tr>
<td>GDT [69]</td>
<td>K400</td>
<td>R(2+1)D-18</td>
<td>15.4M</td>
<td>32</td>
<td>V+A</td>
<td>89.3</td>
<td>60.0</td>
</tr>
<tr>
<td>MMV [2]</td>
<td>AS+HT</td>
<td>S3D-G</td>
<td>9.1M</td>
<td>32</td>
<td>V+A</td>
<td>92.5</td>
<td>69.6</td>
</tr>
<tr>
<td>SpeedNet [7]</td>
<td>K400</td>
<td>S3D-G</td>
<td>9.1M</td>
<td>64</td>
<td>81.1</td>
<td>48.8</td>
<td></td>
</tr>
<tr>
<td>CoCLR [35]</td>
<td>K400</td>
<td>S3D-G</td>
<td>9.1M</td>
<td>32</td>
<td>87.9</td>
<td>54.6</td>
<td></td>
</tr>
<tr>
<td>CoCLR [35]</td>
<td>K400</td>
<td>2×S3D-G</td>
<td>9.1M</td>
<td>32</td>
<td>90.6</td>
<td>62.9</td>
<td></td>
</tr>
<tr>
<td>VTHCL [94]</td>
<td>K400</td>
<td>R-50</td>
<td>31.8M</td>
<td>8</td>
<td>82.1</td>
<td>49.2</td>
<td>37.8</td>
</tr>
<tr>
<td>CVRL [72]</td>
<td>K400</td>
<td>R-50</td>
<td>31.8M</td>
<td>32</td>
<td>92.2</td>
<td>66.7</td>
<td>66.1</td>
</tr>
<tr>
<td>pBYOL</td>
<td>K400</td>
<td>R-50</td>
<td>31.8M</td>
<td>8</td>
<td>94.2</td>
<td>72.1</td>
<td>70.0</td>
</tr>
<tr>
<td>pBYOL</td>
<td>K400</td>
<td>R-50</td>
<td>31.8M</td>
<td>16</td>
<td>95.5</td>
<td>73.6</td>
<td>71.5</td>
</tr>
<tr>
<td>pBYOL</td>
<td>K400</td>
<td>R(2+1)D-18</td>
<td>15.4M</td>
<td>32</td>
<td>94.4</td>
<td>72.2</td>
<td></td>
</tr>
<tr>
<td>pBYOL</td>
<td>K400</td>
<td>S3D-G</td>
<td>9.1M</td>
<td>32</td>
<td>96.3</td>
<td>75.0</td>
<td></td>
</tr>
</tbody>
</table>

Table 11. Comparison with state-of-the-art. “param” indicates the number of parameters, $T$ inference frames, in the backbone. “V” is Vision, “A” is Audio, “T” Text modality. pBYOL is our best model trained with temporal persistency of $\rho=4$. We report fine-tuning accuracy on UCF/HMDB and linear accuracy on K400.

### Table 10. Downstream benchmarks: We use linear evaluation on K400 and finetuning accuracy on the other datasets. 200 epochs, $\rho=3$.

<table>
<thead>
<tr>
<th>method</th>
<th>protocol</th>
<th>finetuning accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>supervised</td>
<td>scratch</td>
<td>74.7</td>
</tr>
<tr>
<td>supervised</td>
<td>K400-240k</td>
<td>68.8</td>
</tr>
<tr>
<td>SimCLR</td>
<td>K400-240k</td>
<td>62.0</td>
</tr>
<tr>
<td>SwAV</td>
<td>K400-240k</td>
<td>62.7</td>
</tr>
<tr>
<td>BYOL</td>
<td>K400-240k</td>
<td>68.3</td>
</tr>
<tr>
<td>MoCo</td>
<td>K400-240k</td>
<td>67.3</td>
</tr>
<tr>
<td>MoCo</td>
<td>IG-Curated-1M</td>
<td>69.9</td>
</tr>
<tr>
<td>MoCo</td>
<td>IG-Uncurated-1M</td>
<td>66.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>method</th>
<th>protocol</th>
<th>finetuning accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>XDC [3]</td>
<td>K400</td>
<td>94.8</td>
</tr>
<tr>
<td>GDT [69]</td>
<td>K400</td>
<td>94.8</td>
</tr>
<tr>
<td>MMV [2]</td>
<td>AS+HT</td>
<td>94.8</td>
</tr>
<tr>
<td>SpeedNet [7]</td>
<td>K400</td>
<td>94.8</td>
</tr>
<tr>
<td>CoCLR [35]</td>
<td>K400</td>
<td>94.8</td>
</tr>
<tr>
<td>CoCLR [35]</td>
<td>K400</td>
<td>94.8</td>
</tr>
<tr>
<td>VTHCL [94]</td>
<td>K400</td>
<td>94.8</td>
</tr>
<tr>
<td>CVRL [72]</td>
<td>K400</td>
<td>94.8</td>
</tr>
<tr>
<td>pBYOL</td>
<td>K400</td>
<td>94.8</td>
</tr>
<tr>
<td>pBYOL</td>
<td>K400</td>
<td>94.8</td>
</tr>
<tr>
<td>pBYOL</td>
<td>K400</td>
<td>94.8</td>
</tr>
<tr>
<td>pBYOL</td>
<td>K400</td>
<td>94.8</td>
</tr>
<tr>
<td>pBYOL</td>
<td>K400</td>
<td>94.8</td>
</tr>
</tbody>
</table>
References


[34] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. In Workshop on Large Scale Holistic Video Understanding, ICCV, 2019. 2


