# Dense Unsupervised Learning for Video Segmentation

**Anonymous Author(s)**
Affiliation
Address
email

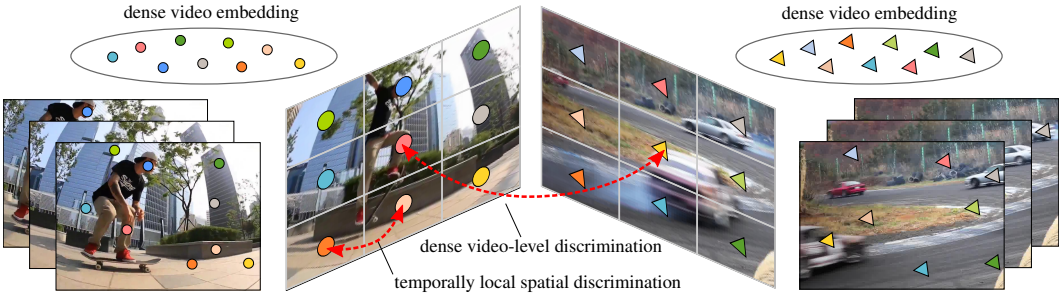



Figure 1. **Illustration of the main idea.** We learn dense feature representations in an unsupervised way by *(i)* discriminating the features both spatially and at the video-level, and *(ii)* embedding video representations in terms of a persistent set of video-specific clusters. We devise a simple and effective approach that sidesteps trivial solutions, yet efficiently learns features in a fully convolutional manner.

## Abstract

We present a novel approach to unsupervised learning for video object segmentation (VOS). In contrast to previous methods, our approach learns dense feature representations directly in a fully-convolutional regime. We rely on uniform grid sampling to extract a set of anchors and train our model to disambiguate between them on the inter- and intra-video levels. A naive scheme to train such a model results in a degenerate solution. However, we prevent it with simple regularisation accommodating equivariance property of the segmentation task to similarity transformations. Notably, our training objective with the regularisation can be jointly expressed by a single loss term. On established VOS benchmarks, our approach reaches segmentation accuracy on par or even higher than previous works despite using significantly less training data and compute power.[1]

## 1 Introduction

Unsupervised learning has recently made considerable and expeditious advances, in part already outperforming even supervised feature learning methods [5, 9, 11]. Most of these works, however, require substantial computational resources [5, 9], and only few accommodate one of the most ubiquitous types of visual data – videos [10, 26]. In contrast to image sets, temporal data embeds ample information about the transformations occurring in nature. Exploiting such cues may allow systems to learn useful invariances directly from data, instead of relying on the commonly used data augmentation [24]. In this work, we propose to learn such invariances with a novel framework for the task of video object segmentation (VOS). In contrast to previous works [13, 15], we learn dense

[1] Code and pre-trained models will be made publicly available (under Apache License).

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

representations in a *fully convolutional* manner, previously considered prone to degenerate solutions. Notably, we develop an approach that not only requires only single loss term for training, but additionally encodes the desired equivariance of the VOS task to similarity transformations into the learned representations. Our framework achieves remarkable computational and data efficiency: training on a single commodity GPU, we attain segmentation accuracy on par with or even outperforming alternatives [13, 31], which require orders of magnitude larger amounts of data.

## 2 Related work

Our work contributes to the research areas of unsupervised video object segmentation (VOS) and representation learning.

**VOS** is a dense tracking task, which requires to learn dense feature representations to establish temporal correspondences and to propagate the semantic labels. While most previous methods are supervised by pixel-wise mask annotation (*e. g.* [21]), an emerging direction is to leverage unsupervised learning. Our work contributes to these research efforts. Contrastive Random Walk (CRW) [13] and its predecessor [32] exploit a time-cycle consistency constraint, akin to the forward-backward consistency in unsupervised optical flow estimation [19]. Other methods [14, 15] learn the label propagation process at training time by substituting the mask (which is unavailable at training time) with the image, an idea inspired by video colorization [30]. The algorithm for label propagation itself significantly affects the VOS accuracy. Mask propagation with naive pixel-wise correspondences usually benefits greatly from the more sophisticated region-level propagation techniques [15, 16], which is not the focus of our work.

**Representation learning.** Although previous work on unsupervised learning pursued spatially view-invariant objectives [3, 5, 9, 11, 12, 20, 29], a concurrent study [8] finds that accommodating them to temporally-invariant, or *temporally-persistent* constraints, yields sizeable gains in terms of accuracy on action recognition benchmarks. These findings align well with previous and concurrent work tailored specifically to learning spatio-temporal representations [1, 7, 10, 25, 26] for video recognition and retrieval tasks. These works additionally distinguish the representations of frames from different videos (or, at different timesteps). The limitation of these works in the context of VOS is that they do not learn *dense*, but *global* feature representations on the image or video level, and typically require considerable computational budgets.

**Semi-supervised representation learning.** The concurrent work by Xiong et al. [33] and previous work [10] leverage optical flow [27] to ensure the representation similarity of temporally corresponding features. Such an objective can be alternatively supervised by motion segments [22]. In contrast to these works, we learn to establish temporal correspondences via a suitable feature representation learned in a completely unsupervised way.

**Clustering.** Contrastive learning methods are inherently expensive due to the size of pairwise feature comparisons they consider in training [5, 11]. Clustering provides a more computationally efficient alternative [4], since pairwise distances need to be computed only *w. r. t.* a (moderately sized) set of pre-defined clusters. Caron et al. [3] propose instead to first cluster the embeddings and then enforce consistency between the cluster assignments to reduce the computations. The main challenge in this approach is to impose an expressive prior on the cluster assignments. For example, a uniform prior prevents trivial solutions where feature assignments collapse to a single cluster.

Here, we neither explicitly define the clusters nor the prior. Instead, we sample attractor points, called *anchors*, directly from the available feature representations using a special grid sampling strategy.

## 3 Method

Learning useful feature representations from unlabelled data is an inherently ill-posed problem and cannot be achieved without any assumptions. We begin with specifying the assumptions guiding our

unsupervised learning method before delving into the technical specifics of their implementation. We then look into the mechanisms our method exploits to implement those assumptions in more detail.

### 3.1 Assumptions

In the following, we articulate with the space of natural images and videos, denoted by $\mathcal{I}$ and represented by RGB values, and their corresponding dense feature representations, forming space $\mathcal{F}$.

**Assumption 1.** *Spatial feature diversity with spatial correlation.* We assume that natural scenes $\mathcal{I}$ exhibit a variety of distinguishable spatial artefacts that also correlate spatially: the more distant two image features are in $\mathcal{I}$, the more likely they belong to different entities, and vice versa. Dense representations $\mathcal{F}$ of the images should mirror this property. In support of this assumption, common scenes of interest often follow a layout structure (*e. g.*, ceiling on the top, floor on the bottom).

**Assumption 2.** *Dense temporal persistence of semantic content.* We further assume that the content of video clips remains unchanged, at least for some short time span. *I. e.*, if we represent a frame with a set of distinct features, the semantic content of all other temporally close frames can be faithfully represented with the same feature set. Time cycle consistency [13] relies on a somewhat stronger assumption that every frame in a sequence *must* contain the same feature set as the starting frame. In contrast, we impose a similar a constraint but on the video level, which can aid handling occlusions.

**Assumption 3.** *Temporal coherence.* Two video clips may contain the same semantic content (*e. g.*, cow on a farm), yet their feature representations should be distinguishable to ensure temporal coherence: a feature representation extracted from a frame $t$ in one video should be closer in the embedding space to the corresponding feature in another frame $t_i$ from the same sequence, rather than any frame in the other video. This is a desirable property for dense tracking applications focused on in this work.

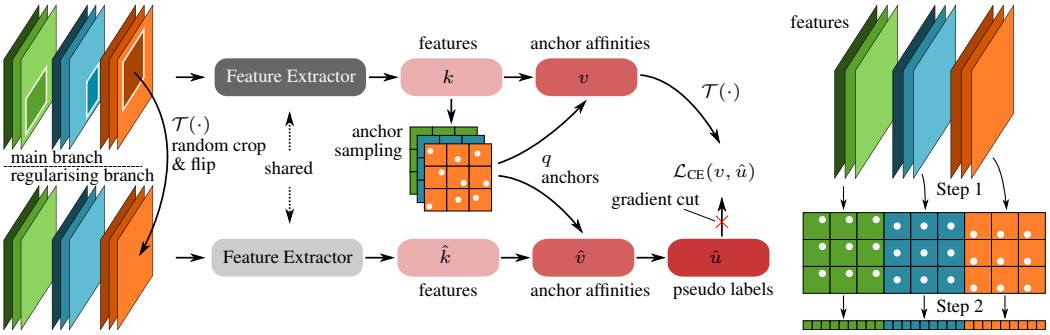
**Assumption 4.** *Model equivariance to similarity transformations.* Features of the original image should be equal to the extracted features of the scaled or flipped image *after* the same similarity transformation has been applied to the latter. This is in contrast to the common practice of aligning the feature representations of two views without applying the corresponding transformation for image recognition tasks [5], where the model only needs to be *invariant* to such transformations.

### 3.2 Overview

Our approach incurs a self-supervised loss satisfying three discriminative properties. First, we learn to densely represent every video frame in terms of spatially discriminative features. Second, our method learns to represent every frame in a video sequence as a composite of discriminative features extracted from a single reference frame. Lastly, the set of discriminative features representing a video is learned to be distinguishable from the set of another video sequence.

### 3.3 Framework

The main component of our framework, illustrated in Fig. 2a, is a feature extractor, implemented as a fully convolutional network [17]. The network processes two copies of the input image batch, comprising the same set of video sequences, in the *main* and *regularising* branches. The purpose of the regularising branch is to prevent degenerate solutions observed in previous attempts to learn feature representations in a fully convolutional manner [13]. While the main branch receives the original video frames, we feed a transformed version of these frames to the regularising branch. Specifically, we extract random multi-scale crops and apply horizontal flipping at random — we denote this random similarity transform as $\mathcal{T}(\cdot)$. Both, main and regularising branch produce dense $K$-dimensional feature tensors $k$ and $\hat{k}$, respectively. Formally, $k, \hat{k} \in \mathbb{R}^{B,T,K,h,w}$ and $\|k_{i,j,:,l,m}\| = \|\hat{k}_{i,j,:,l,m}\| = 1$, where $B$ is the batch size, $T$ is the number of selected frames from a video sequence, and $h, w$ are the spatial dimensions of these embeddings. We will distinguish the terms related to the regularising branch using the hat notation $(\hat{\ })$ also in the following presentation.




(a) Framework overview (b) Anchor sampling

Figure 2. **Overview.** Our framework *(a)* consists of single CNN-based feature extractor that processes video sequences in the main branch. The second regularising branch takes a transformed version (random cropping and flipping) of the videos as input and produces pseudo labels $\hat{u}$ *w. r. t.* the anchors extracted from the features of the main branch $q$. To sample the anchors *(b)*, we first randomly select one frame per video sequence in the current batch (Step 1), and then sample the features at random on a uniform grid in Step 2 to obtain $q$.

In the next step we compute pairwise feature distances. Recall from Sec. 3.2 that our goal is to discriminate features on three levels: *(i)* spatially for an individual frame, *(ii)* between video sequences, as well as *(iii)* to represent each frame in a video sequence in terms of the same feature set. A naive implementation would require computing three sets of pairwise distances and corresponding loss terms. The frame-specific distance tensor would be of size $B \times T \times (h \times w)^2$, computing the pairwise distances within a video clip would need $B \times (T \times h \times w)^2$, and computing the distances between the videos would have size $(B \times T \times h \times w)^2$. Considering the spatial dimensionality of the feature tensor (*e. g.* $h \times w = 1024$ for a modest $256 \times 256$ image resolution and output stride of 8), such naive approach would not scale well.

**Anchor Sampling.** In this work, we take a bi-level sampling approach based on clustering to improve the training efficiency, illustrated in Fig. 2b. First, for each video sequence in the batch we sample one frame at random. We define it as a *representative* frame for the video sequence from which it originates. Relying on Assumption 2 that the frames of the same sequence share semantic context, we leverage the randomly chosen frame for extracting a set of *video-level* features. Next, instead of computing the pairwise distances between every feature vector in the batch, we define a uniform grid of size $N \times N$ on the feature tensor of the representative frame, and draw one sample per grid cell. This is to select features that are spatially discriminant and cover the full image. As a result, we obtain $q_1 \in \mathbb{R}^{B \times N^2}$ $K$-dimensional feature embeddings, concatenated across the batch, which we will refer to as feature *anchors*. Finally, we compute the pairwise distances between the features *w. r. t.* the anchors, rather than features themselves. The outcome is a distance matrix of size $BThw \times BN^2$. This saves the memory and computation budget by a factor of $T \times hw \,/\, N^2$. For example, if $h \times w = 32 \times 32$ and we use a grid of size $8 \times 8$, this will be a saving factor of $T \times 16$.

**Computing affinity to anchors.** Following our cluster-based approach, we compute affinities of the features $k$ and $\hat{k}$ from the two branches *w. r. t.* the anchors $q$. Note that we extract the anchors only from the main branch and share them with the regularising branch. Since the anchors $q \in \mathbb{R}^{BN^2 \times K}$ are sampled from $k$, they also have unit L2-norm, *i. e.* $\|q_i\| = 1$ for all $i$. Let $v, \hat{v} \in \mathbb{R}^{BTHW \times BN}$ denote the affinities of $k$ and $\hat{k}$ *w. r. t.* $q$, respectively. To compute the affinities, we use the cosine similarity coupled with an attention mechanism, *i. e.*

$$v_{i,j} = \frac{\exp(k_i \cdot q_j / \tau)}{\sum_l \exp(k_i \cdot q_l / \tau)}, \qquad \hat{v}_{i,j} = \frac{\exp(\hat{k}_i \cdot q_j / \tau)}{\sum_l \exp(\hat{k}_i \cdot q_l / \tau)}, \tag{1}$$

where $\tau \in \mathbb{R}^+$ is a scalar temperature hyperparameter. Put simply, $v_{i,j} \in [0,1]^{BTHW \times BN}$ represents the similarity between feature $k_i$ and anchor $q_j$ (analogously for $\hat{v}_{i,j}$).

**Self-supervision.** We first generate pseudo-labels of anchor assignments by computing the dominant anchor index for each feature from the regularising branch,

$$\hat{u}_i = \arg\max_{j \in \mathcal{N}(i)} \hat{v}_{i,j}, \tag{2}$$

where $\mathcal{N}(i)$ denotes the indices of the frames from the video clip shared with frame $i$. Observe that $k$ and $\hat{k}$ are related via a similarity transformation $\mathcal{T}$. Our self-supervised loss consists of a *single* term and exploits the equivariance of the feature embedding under Assumption 4:

$$\mathcal{L}_{CE}(v^*, \hat{u}) = -\sum_i \log v^*_{i,\hat{u}_i} \ , \tag{3}$$

where $v^* = \mathcal{T}(v)$ are the affinies of the main branch aligned with the input of the regularising branch.

### 3.4 Implementation

```
def affinity(x,y,tau)
  z = einsum("bkn,bkhw->bnhw",x,y)
  return softmax(z,1)

for x in loader:
  # x_hat: cropped and flipped x
  # T: similarity transform
  x_hat, T = crop_and_flip(x)

  # computing feature embeddings
  k = net(x) # [BxKxHxW]
  # no backgropagation for x_hat
  with no_grad():
    k_hat = net(x_hat)

  # sampling anchors
  q = grid_sample(k) # [BxKxNN]

  # compute affinities
  # [BxNNxHxW]
  v = affinity(q, k, tau)
  v_hat = affinity(q, k_hat, tau)

  # transform affinity
  # of the main branch
  v_ast = transform(v, T) # [BxKxHxW]

  # compute pseudo labels
  u_hat = v_hat.argmax(1)

  # loss
  loss = cross_entropy(v_ast, u_hat)
  loss.backward()
  optimizer.step()
```

Listing 1. Pseudo code implementation of our method.

Listing 1 demonstrates the pseudo code of our algorithm. We observe that our algorithm shares the simplicity of other unsupervised learning algorithms [5, 6, 35], yet targets temporally consistent embeddings without any iterative structures (*e. g.* as in time-cycle consistency [13]). All operations support efficient implementation in popular libraries, such as PyTorch.

Note that the regularising branch is always in "evaluation mode", which reduces the memory footprint of our approach compared to Siamese architectures [35].

### 3.5 Analysis

Let us look at how our framework implements the assumptions from Sec. 3.1.

**Spatial feature diversity (Assumption 1).** Our loss attracts feature representations to their closest anchors, while pushing it away from the others. By virtue of the sampling process, the anchors themselves participate in this loss, which implies that the loss encourages larger cosine distance between the anchors. Recall that the anchors are also sampled spatially from one of the video frames on a uniform grid, *i. e.* our loss learns to discriminate between spatially discrepant features. To avoid degenerate solutions, we also spatially jitter the grid. This way there is always a fixed distance between the anchors.

**Temporal persistence of semantic content (Assumption 2).** Under this assumption, a video clip consists of the same set of features that can be located in one of the frames. Again, this is ensured in part by our anchor sampling strategy, since all the anchors come from only one of the video frames. The design of the pseudo labels, on the other hand, ensures that all frames in the video should be representable in terms of the anchors extracted from one of the frames. This is facilitated by the $\arg\max$ operation in Eq. (2), which selects dominant anchors for a given frame only from the anchors extracted from a frame of the same video, while ignoring the others.

**Temporal coherence (Assumption 3).** We also want to discriminate features between different videos. Again, the $\arg\max$ operation in Eq. (2) complies with this assumption, since the feature representations should be closest to the ones stemming from the same video.

**Model equivariance (Assumption 4).** The regularising branch provides a transformed version of the feature embeddings. From our loss function Eq. (3), we observe that it will reach its minimum if the anchor assignments between the two feature representations agree. Importantly, we encourage this agreement only by the types of transformations pertinent to our task. Segmentation models are equivariant to the scaling and flipping that we use.

### 3.6 Discussion

**Preventing degenerate solutions.** In our preliminary experiments without the regularising branch, we observed that the model would instantly converge to a degenerate solution. Our investigation showed that the network was exploiting the spatial positioning of the features and was learning a sort of coordinate encoding. This is not surprising, as there is substantial evidence that CNNs are not truly spatially invariant. Part of the problem is the padding – no matter what padding strategy is used (zero, reflect, *etc*) it provides a stable and predictable pattern that the high-capacity networks are ready to exploit. We experimented with multiple alternatives to prevent degenerate solutions and found the presented solution to perform best both in terms of accuracy and efficiency.

**Scalability.** We use a grid structure instead of sampling every pixel individually. This allows for considerable memory savings. Suppose we have an output feature resolution of $32\times32$. Our loss computes the affinity to all frames in the batch. Suppose we have batch size B and video length T. We construct a tensor $\mathrm{B}\times\mathrm{N}\times32\times32$ times $\mathrm{B}\times\mathrm{T}\times32\times32$ (every pixel in). But if we have a grid of size $16\times16$ instead, we already have $\times2$ saving in terms of memory.

**Practical considerations.** The order of the frames from a sequence in the batch is irrelevant and can be randomly selected, since we make no assumptions about motion continuity. Further we do not use any momentum network or queue buffers, *e. g.* such as in [11]. Our training is surprisingly stable without any of them. Further, we only use similarity transformations to augment the training data, and no additional augmentations such as, *e. g.* photometric noise ubiquitous in previous work [5, 9, 11]. Since we use only a single loss term, it also spares our approach from additional hyperparameters.

**Comparison to Caron et al. [3].** Our method does not require an additional entropy term ensuring the features to be attracted to the clusters. This is implicit in our model by virtue of the pseudo label (*i. e.* we convert soft scores to hard labels). In contrast to Caron et al. [3], we also make no assumptions on the distribution of the cluster assignment. The clusters in [3] are equivalent to the anchors in our approach, hence are not global (*i. e.* not shared between the videos), but specific to the video sample.

**Comparison to Contrastive Random Walk (CRW) [13].** Our assumption on the semantic consistency within the video clip is the same as in CRW. However, our approach is simpler as we do not reason in terms of temporal causality, *i. e.* the frame ordering within the video clip is irrelevant for our approach. We also use a weaker assumption: we do not require every frame to contain the same set of features, which is implicitly enforced by the cycle consistency constraint in CRW. This can be advantageous to video sequences with occlusions. For example, if the object of interest disappears in the sequence, we can still handle this case (provided the reference frame contains this object). Cycle consistency, on the other hand, will encourage that even the frame without the object contains its feature representation. Additionally, our approach learns discriminative features on the inter-video level, which is not part of the contrastive random walk.

## 4 Experiments

To evaluate the learned feature representations, we conduct experiments in the setting of semi-supervised video object segmentation (VOS). The task provides a set of segmentation masks for the first frame in a video sequence and requires the evaluated algorithm to produce the corresponding masks in the remaining frames. We largely follow the VOS setup of Jabri et al. [13] and evaluate our method on DAVIS-2017 [28]. In addition, we test our approach on the YouTube-VOS benchmark [34].

**Implementation details.** Similarly to Jabri *et al.* [13], we use ResNet-18 as the backbone network for our feature extractor and evaluate the correspondences using the output of the third (penultimate) residual block. Since we work directly on image frames rather than patches, we replace the strides of the last two residual blocks with dilated convolutions of rates $2$ and $4$, respectively, as is common in semantic segmentation models. The output stride of our feature extractor is $8$, as in some previous work [13, 31]. At training time, we first scale the video frames such that the lowest side is between $256$ and $320$ pixels, and extract random crops of size $256 \times 256$. We train our network with Adam

Table 1. **Video object segmentation quality** on the DAVIS-2017 validation set in terms of the mask ($\mathcal{J}$) and boundary ($\mathcal{F}$) accuracy. Subscript $[\cdot]_r$ denotes recall of the metric, while $[\cdot]_m$ signifies the mean. Column $\times 2$ specifies that the output stride is 4 instead of 8, *i. e.* the spatial resolution of the features is twice as large. $\#$ / $t$ details dataset characteristics: the number of videos / the total dataset duration in hours.

| Method | $\times 2$ | Train Data | $\#$ / $t$ | $\mathcal{J}_m$ | $\mathcal{J}_r$ | $\mathcal{F}_m$ | $\mathcal{F}_r$ | $\mathcal{J}\&\mathcal{F}_m$ |
|---|---|---|---|---|---|---|---|---|
| Baseline (random initialisation) | — | — | — | — | — | — | — | 43.1 |
| ContrastCorr [31] | — | TrackingNet | 30K / 140 | 60.5 | — | 65.5 | — | 63.0 |
| CRW [13] | — | Kinetics-400 | 300K / 833 | 64.8 | 76.1 | 70.2 | 82.1 | 67.6 |
| CorrFlow [14] | — | OxUvA | 366 / 14 | 48.4 | 53.2 | 52.2 | 56.0 | 50.3 |
| MAST [15] | ✓ | | | 61.2 | 73.2 | 66.3 | 78.3 | 63.7 |
| Ours | — | | | **62.9** | **77.1** | **67.3** | **80.7** | **65.1** |
| MAST [15] | ✓ | YT-VOS | 4.5K / 5 | 63.3 | 73.2 | 67.6 | 77.7 | 65.5 |
| Ours | — | | | **65.3** | **80.0** | **69.4** | **82.1** | **67.3** |

and learning rate of $10^{-4}$. The only hyperparameter of our approach, the temperature $\tau$, is set to $0.05$ throughout our experiments. We train our models on a *single* Titan X GPU with 12GB of memory, except for our ablation study, where some configurations exceed this modest computation budget and we use one $A100$ GPU instead.

**Label propagation.** We use our own implementation of the label propagation approach closely following that of [13]. In particular, for every feature embedding in the current frame, we compute their cosine distance *w. r. t.* the features in previous $N_T$ context frames and the first reference frame. We select $N_K$ feature embeddings with the lowest distance, divide the distance by $\tau$ and compute the softmax to obtain the weights. The mask prediction for this feature embedding is a convex combination of the mask predictions of its neighbours *w. r. t.* to the weights. In contrast to Jabri *et al.* [13], however, we use correlation operation and compute the distances only in the region of interest defined by a patch size $P$ for efficiency. The supplementary material provides further details.

### 4.1 Evaluation on DAVIS-2017

To evaluate on DAVIS-2017 [23] *val*, we train two separate feature extractors using our method. For the first, we use `OxUvA` dataset [28] for training, which spans 366 video sequences with a total duration of 14 hours. The second dataset used for training is the YouTube-VOS [34] *train* split. In comparison to `OxUvA`, it contains more videos, around 4000 sequences, but has a shorter overall duration of only 5.6 hours. For our label propagation algorithm, we use $N_T = 20$ context frames, and consider $N_K = 5$ closest neighbours in the embeddings space defined by the spatial neighbourhood size of $P = 25$. We process the frames at $480p$ resolution for fair comparison.

Table 1 reports the segmentation accuracy of our method and compares it to previous work. In a comparable setting where the methods use the same dataset for training, our approach improves over the state-of-the-art accuracy. When trained on `OxUvA`, our approach outperforms MAST [15] by $1.4$ in $\mathcal{J}\&\mathcal{F}_m$ score. This improvement is consistent when trained on `YouTube-VOS`, and we achieve higher segmentation accuracy than MAST [15] by $1.8$ in $\mathcal{J}\&\mathcal{F}_m$ score. This improvement is especially notable, since MAST [15] produces its feature embeddings at twice the resolution of our method, hence has also a larger memory footprint. Remarkably, our method improves over CRW [13] on some of the metrics ($\mathcal{J}_m$ and $\mathcal{J}_r$) while staying on par for the other despite using only a fractional amount of data for training. We also observe that the diversity of the dataset, *i. e.* the number of videos rather than the total duration, tends to improve the quality of our learned features in terms of the segmentation accuracy. Our approach achieves tangibly higher accuracy using the shorter but more diverse `YouTube-VOS` dataset compared to using `OxUvA`.

Table 2. **Additional VOS experiments.** (a) We test our model on YouTube-VOS 2018 *valid*. ($^\dagger$) denotes the use of our label propagation; ($^\ddagger$) training on Kinetics-400. (b) In our ablation study we investigate *(i)* the trade-off between the number of videos and their length in a batch, *(ii)* the grid resolution, and *(iii)* the video length.

(a) Results on YouTube-VOS 2018 *valid*

| Method | Seen | | Unseen | | $\mathcal{J}\&\mathcal{F}_m$ |
|---|---|---|---|---|---|
| | $\mathcal{J}$ | $\mathcal{F}$ | $\mathcal{J}$ | $\mathcal{F}$ | |
| *Unsupervised* | | | | | |
| Colorize [30] | 43.1 | 38.6 | 36.6 | 37.4 | 38.9 |
| CorrFlow [14] | 50.6 | 46.6 | 43.8 | 45.6 | 46.6 |
| MAST [15] | **63.9** | **64.9** | **60.3** | **67.7** | **64.2** |
| CRW$^{\dagger\ddagger}$ [13] | 62.0 | 61.9 | 55.6 | 61.7 | 60.3 |
| Ours$^\dagger$ | 61.7 | 61.5 | 53.9 | 60.0 | 59.3 |
| *Supervised* | | | | | |
| OSVOS [2] | 59.8 | 60.5 | 54.2 | 60.7 | 58.8 |
| S2S [34] | 71.0 | 70.0 | 55.5 | 61.2 | 64.4 |
| PreMVOS [18] | 71.4 | 75.9 | 56.5 | 63.7 | 66.9 |
| STM [21] | **79.7** | **84.2** | **72.8** | **80.9** | **79.4** |

(b) Ablation Study

| Configuration | $\mathcal{J}\&\mathcal{F}_m$ | $\mathcal{J}_r$ | $\mathcal{F}_r$ |
|---|---|---|---|
| *(i)* $B \times T = 80$ | | | |
| $B = 80, T = 1$ | 61.4 | 70.9 | 72.1 |
| $B = 40, T = 2$ | 64.2 | 73.8 | 77.3 |
| $B = 16, T = 5$ | **67.3** | **80.0** | **82.1** |
| $B = 8, T = 10$ | 64.9 | 74.8 | 79.6 |
| *(ii)* Grid size $N \times N$ | | | |
| $4 \times 4$ | 64.4 | 76.1 | 78.9 |
| $8 \times 8$ | **67.3** | **80.0** | **82.1** |
| $16 \times 16$ | 65.5 | 77.1 | 80.0 |
| *(iii)* Video set size $T$; fixed $B = 16$ | | | |
| $T = 5$ | **67.3** | **80.0** | **82.1** |
| $T = 10$ | 65.5 | 76.9 | 79.5 |
| $T = 15$ | 66.0 | 76.7 | 81.9 |

### 4.2 Evaluation on YouTube-VOS

Following Lai et al. [15], we additionally evaluate our features on YouTube-VOS 2018 *valid* split. This set comprises 474 video sequences with 91 unique object classes. We train our model on the *train* split for a fair comparison with previous methods. The *train* set contains only some of the object classes present in *valid*; the benchmark evaluates these "seen" and "unseen" categories separately. We submit our results to the official evaluation server to obtain the quantitative metrics. `YouTube-VOS` *valid* has additional specifics compared to the DAVIS benchmark and previous work tend to use a different label propagation strategy to address them. For example, objects may appear at intermediate frames, rather than only in the first frame, as in DAVIS. This poses the question of, for example, how to handle these cases *w. r. t.* the context frames. Alternative solutions include, *e. g.* creating a separate set of context frames for each tracked object or using a shared set.

We did our best to adhere to the strategy used by [15]. However, the label propagation algorithm is part of MAST [15], and still differs from ours. For example, they use a sophisticated technique to localise ROIs in the frames, *etc*. Since we are interested in only evaluating the representational quality of our features, we used the publicly available CRW [13] with our label propagation method.

We report the results in Table 2a. Similarly to our result on DAVIS, we observe that our method achieves on par segmentation accuracy with CRW despite using significantly less data for training. Although we do not outperform MAST [15], which we credit to using a more primitive label propagation strategy, our attained segmentation accuracy is still competitive and outperforms a number of previous unsupervised methods, and even a supervised approach [2].

### 4.3 Further analysis

We study three aspects of our model to evaluate the interplay between number of frames $T$ per video, number of video sequences $B$ per batch and the grid size $N \times N$ for sampling of the anchors. All experiments are performed with the corresponding model trained on the *train* split of `YouTube-VOS`.

**Balance between the number of videos and their size: Table 2b-*(i)*.** The effective batch size of our method is $B \times T$, *i. e.* it depends on the number of videos and their duration. Here, we investigate the balance between the size of the video set and the number of distinct video sequences used in a single training batch. We fix $B \times T$ to 80 in order to remain within feasible computational capacity and evaluate for configuration of $B$ and $T$. Our first observation is that *using multiple frames is*

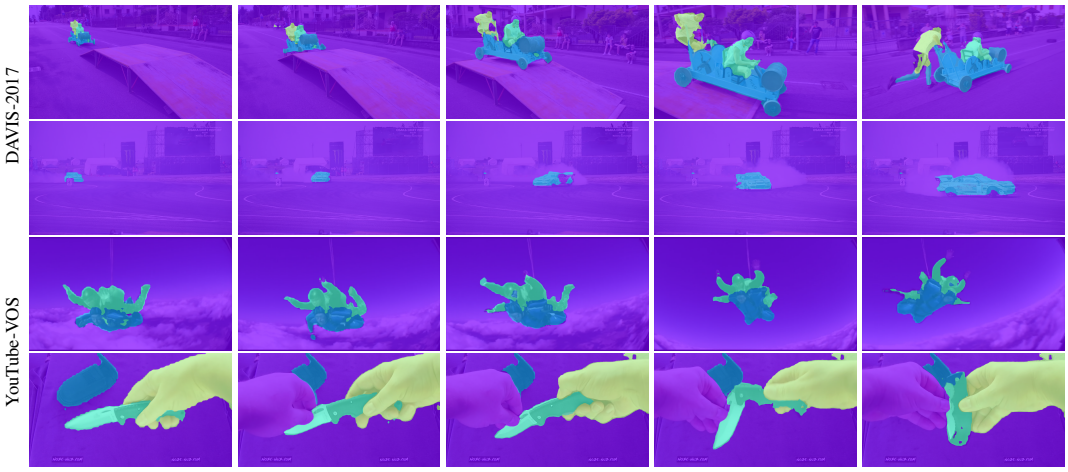



Figure 3. **Qualitative examples on DAVIS-2017 and YouTube-VOS.** The representation learned by our method is robust to occlusions, scaling as well as naturally occurring isometric and piecewise rigid transformations.

*crucial*: training with only a single frame per video ($T = 1$) significantly reduces the quality of learned feature representations in terms of VOS accuracy. In light of Assumption 2, this is expected, as only in a multi-frame setting our model can learn a shared representation of the video set.

**Grid resolution: Table 2b-*(ii)*.** We test our model with three grid sizes: $4 \times 4$, $8 \times 8$ and $16 \times 16$. We expect that low grid resolution, *i. e.* large $N$, may be too coarse in terms of discriminating a sufficient level of detail required in VOS. On the other hand, high grid resolution, *i. e.* small $N$, may violate Assumption 1, *i. e.* tend to distinguish spatially correlated phenomena of shared semantic meaning (*i. e.* discriminating two body parts annotated in VOS as a single segment). We find that both of these extremes lead to a tangible lower VOS accuracy compared to $8 \times 8$ grid. Curiously, this nearly coincides with the patch grid resolution of $7 \times 7$ found to be optimal in related work [13].

**Effect of the video set size: Table 2b-*(iii)*.** In this experiment, we fix the number of videos sequences we sample for each batch $B = 16$ and vary the size of the video set. We increase the size of the temporal window used for sampling the frames proportionally ($5 \times T$). Following our Assumption 2 of temporal persistence, longer video sequences tend to exhibit increased amount of occlusion, as well as disocclusion. Therefore, the requirement that all frames in a video set are representable in terms of a feature set extracted from a single frame becomes too strict for higher sizes of the video set. The results confirm this expectation: increasing the size of the video set somewhat decreases the accuracy *w. r. t.* a model with a moderate video size. Again, this aligns well with an equivalent setting in previous work [13] (the recommended video length was 4).

**Qualitative examples.** Fig. 3 illustrates examples of mask propagation leveraging our learned feature representation. We observe the temporal correspondence to remain stable despite a wide range of interfering transformations, scaling, rotations as well as natural transformation pertaining to human and rigid objects (*e. g.* the Swiss knife). See supplementary mat. for more qualitative results.

## 5 Conclusion

We presented a simple method for unsupervised learning for VOS, that is both data and computationally efficient. In contrast to previous research efforts, we achieve competitive segmentation accuracy by training our framework directly in a fully convolutional regime and without use of unnatural photometric augmentations. We recognise the possibility of our research results to be used with malicious intents, such as unauthorised surveillance. However, as a product of fundamental research with low technology readiness levels of 1 to 3, this is highly unlikely without expert advice that we commit not to provide. Our conclusions currently rely on experiments of moderate scale. Considering our encouraging results, we are excited to test our approach on a larger scale, both in terms of the used backbone architectures and the dataset size.

# References


[1] Y. Bai, H. Fan, I. Misra, G. Venkatesh, Y. Lu, Y. Zhou, Q. Yu, V. Chandra, and A. L. Yuille. Can temporal information help with contrastive self-supervised learning? *arXiv:2011.13046 [cs.CV]*, 2020.

[2] S. Caelles, K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. V. Gool. One-shot video object segmentation. In *CVPR*, pages 5320–5329, 2017.

[3] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*2020*.

[4] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, volume 14 of *Lecture Notes in Computer Science*, pages 139–156, 2018.

[5] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, pages 1597–1607, 2020.

[6] X. Chen and K. He. Exploring simple siamese representation learning. *CoRR*, abs/2011.10566, 2020.

[7] I. Dave, R. Gupta, M. N. Rizve, and M. Shah. TCLR: temporal contrastive learning for video representation. *arXiv:2101.07974 [cs.CV]*, 2021.

[8] C. Feichtenhofer, H. Fan, B. Xiong, R. B. Girshick, and K. He. A large-scale study on unsupervised spatiotemporal representation learning. *arXiv:2104.14558 [cs.CV]*, 2021.

[9] J. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. Á. Pires, Z. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko. Bootstrap your own latent - A new approach to self-supervised learning. In *NeurIPS*2020*, pages 21271–21284.

[10] T. Han, W. Xie, and A. Zisserman. Self-supervised co-training for video representation learning. In *NeurIPS*2020*.

[11] K. He, H. Fan, Y. Wu, S. Xie, and R. B. Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, pages 9726–9735, 2020.

[12] O. J. Hénaff. Data-efficient image recognition with contrastive predictive coding. In *ICML*, pages 4182–4192, 2020.

[13] A. Jabri, A. Owens, and A. A. Efros. Space-time correspondence as a contrastive random walk. In *NeurIPS*2020*, pages 19545–19560.

[14] Z. Lai and W. Xie. Self-supervised video representation learning for correspondence flow. In *BMVC*, 2019.

[15] Z. Lai, E. Lu, and W. Xie. MAST: A memory-augmented self-supervised tracker. In *CVPR*, pages 6478–6487, 2020.

[16] X. Li, S. Liu, S. D. Mello, X. Wang, J. Kautz, and M. Yang. Joint-task self-supervised learning for temporal correspondence. In *NeurIPS*2019*, pages 317–327.

[17] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015.

[18] J. Luiten, P. Voigtlaender, and B. Leibe. Premvos: Proposal-generation, refinement and merging for video object segmentation. In *ACCV*, volume 11364, pages 565–580, 2018.

[19] S. Meister, J. Hur, and S. Roth. UnFlow: Unsupervised learning of optical flow with a bidirectional census loss. In *AAAI*, pages 7251–7259, 2018.

[20] I. Misra and L. van der Maaten. Self-supervised learning of pretext-invariant representations. In *CVPR*, pages 6706–6716, 2020.

[21] S. W. Oh, J. Lee, N. Xu, and S. J. Kim. Video object segmentation using space-time memory networks. In *ICCV*, pages 9225–9234, 2019.

[22] D. Pathak, R. B. Girshick, P. Dollár, T. Darrell, and B. Hariharan. Learning features by watching objects move. In *CVPR*, pages 6024–6033, 2017.

[23] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. V. Gool, M. H. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, pages 724–732, 2016.

[24] L. Perez and J. Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv:1712.04621 [cs.CV]*, 2017.

[25] R. Qian, T. Meng, B. Gong, M.-H. Yang, H. Wang, S. Belongie, and Y. Cui. Spatiotemporal Contrastive Video Representation Learning. *arXiv:2008.03800 [cs.CV]*, 2020.

[26] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, and S. Levine. Time-contrastive networks: Self-supervised learning from video. In *ICRA*, pages 1134–1141, 2018.

[27] Z. Teed and J. Deng. RAFT: recurrent all-pairs field transforms for optical flow. In *ECCV*, volume 2, pages 402–419, 2020.

[28] J. Valmadre, L. Bertinetto, J. F. Henriques, R. Tao, A. Vedaldi, A. W. M. Smeulders, P. H. S. Torr, and E. Gavves. Long-term tracking in the wild: A benchmark. In *ECCV*, volume 3, pages 692–707, 2018.

[29] A. van den Oord, Y. Li, and O. Vinyals. Representation Learning with Contrastive Predictive Coding. *arXiv:1807.03748 [cs.LG]*, 2018.

[30] C. Vondrick, A. Shrivastava, A. Fathi, S. Guadarrama, and K. Murphy. Tracking emerges by colorizing videos. In *ECCV*, volume 13, pages 402–419, 2018.

[31] N. Wang, W. Zhou, and H. Li. Contrastive transformation for self-supervised correspondence learning. In *AAAI*, 2021.

[32] X. Wang, A. Jabri, and A. A. Efros. Learning correspondence from the cycle-consistency of time. In *CVPR*, pages 2566–2576, 2019.

[33] Y. Xiong, M. Ren, W. Zeng, and R. Urtasun. Self-supervised representation learning from flow equivariance. *arXiv:2101.06553 [cs.CV]*, 2021.

[34] N. Xu, L. Yang, Y. Fan, J. Yang, D. Yue, Y. Liang, B. L. Price, S. Cohen, and T. S. Huang. YouTube-VOS: Sequence-to-sequence video object segmentation. In *ECCV*, volume 5, pages 603–619, 2018.

[35] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. Barlow twins: Self-supervised learning via redundancy reduction. *CoRR*, abs/2103.03230, 2021.

# Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We discuss the limitations in Sec. 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We discuss it in Sec. 5.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We provide the pseudo code mirroring our implementation in the main text and commit to release our code to the community for reproducibility upon acceptance.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We provide the training details in Sec. 4 and elaborate on further details (*e. g.* the label propagation algorithm) in the supplementary material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Training with fully convolutional architectures is a computationally demanding process. We believe that estimation of the error bars will shed little scientific insight at a significant environmental impact. Nevertheless, we will be willing to add a sensitivity analysis to our results, if explicitely requested by the reviewers.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We provide these details in Sec. 4.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] We specify the license of the code we will release upon acceptance.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include a video demonstrating the qualitative results produced by our method in the supplementary material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] We use publicly available datasets in full compliance with their terms of use.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] We ensure that the qualitative examples displayed in our work does not exhibit identifiable artefacts.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]