Long Short View Feature Decomposition via Contrastive Video Representation Learning

Nadine Behrmann[1]   Mohsen Fayyaz[2,†]   Juergen Gall[2]   Mehdi Noroozi[1]

[1]Bosch Center for Artificial Intelligence   [2]University of Bonn

{nadine.behrmann, mehdi.noroozi}@de.bosch.com, {fayyaz, gall}@iai.uni-bonn.de

Abstract

Self-supervised video representation methods typically focus on the representation of temporal attributes in videos. However, the role of stationary versus non-stationary attributes is less explored: Stationary features, which remain similar throughout the video, enable the prediction of video-level action classes. Non-stationary features, which represent temporally varying attributes, are more beneficial for downstream tasks involving more fine-grained temporal understanding, such as action segmentation. We argue that a single representation to capture both types of features is sub-optimal, and propose to decompose the representation space into stationary and non-stationary features via contrastive learning from long and short views, i.e. long video sequences and their shorter sub-sequences. Stationary features are shared between the short and long views, while non-stationary features aggregate the short views to match the corresponding long view. To empirically verify our approach, we demonstrate that our stationary features work particularly well on an action recognition downstream task, while our non-stationary features perform better on action segmentation. Furthermore, we analyse the learned representations and find that stationary features capture more temporally stable, static attributes, while non-stationary features encompass more temporally varying ones.

1. Introduction

Learning rich video representations is a key challenge for general video understanding. An ideal representation extracts useful information that benefits numerous downstream tasks such as action recognition, retrieval and action segmentation. Learning such representations in a supervised setting is inherently biased towards static features [24]. However, in order to solve more complex down-

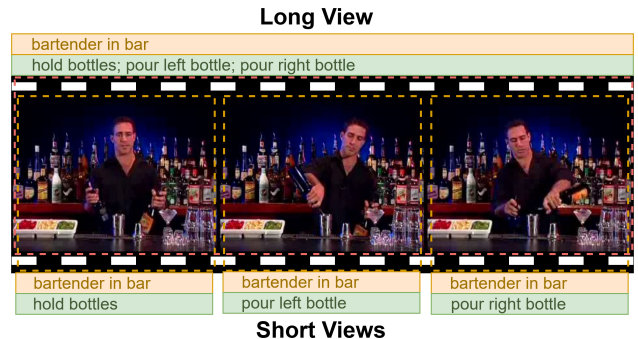† Work done during an internship at the Bosch Center for Artificial Intelligence.

Figure 1. Video attributes can be divided into two sections: Stationary features (shown in yellow), that are shared between long and short views, and non-stationary features (shown in green), that aggregate the short views to match the corresponding long view.

stream tasks, that require temporal attributes of videos such as temporal action segmentation, we need a more diverse set of features. As a remedy, we train our network to represent stationary and non-stationary features. To get an intuition, let's consider the following example: a video of a bartender mixing a cocktail in Figure 1 is stationary in one sense – we see a bartender in a bar with liquor bottles in the background – but non-stationary in another: different steps in preparing a cocktail are shown, such as holding the bottle, pouring the left bottle and pouring the right bottle. Here, the stationary attributes of the video enable us to predict the overall action class, *i.e.* mixing cocktails. On the other hand, the non-stationary attributes enable more fine-grained temporal distinctions, *e.g.* predicting when different steps occur in the video. Ideally, both types of attributes should be represented by video models.

Learning representations in a supervised setting, usually involves pretraining on large-scale labeled datasets such as Kinetics [17] with video-level annotations, inhibiting strong static biases [24]. As a result they capture mainly the stationary features, but largely ignore non-stationary features as the stationary features are sufficient for action recognition. Self-supervised leaning provides a promising direc-

tion to address this shortcoming. As the supervisory signals arise from the underlying structure of the data, it has the potential to extract more descriptive features. Several previous self-supervised methods aim to capture temporal features in videos by designing a temporal pretext task, *e.g.* predicting temporal transformations [15] or video speed [2]. These methods are not explicitly encouraged to capture stationary and non-stationary features. In contrast, we explicitly decompose the representation space into stationary and non-stationary features, enabling us to solve a more diverse set of downstream tasks, including action recognition and temporal action segmentation.

Following the recent trend in self-supervised learning, our proposed method fits in the contrastive learning framework. The supervisory signal that leads us to distinguish between the stationary and non-stationary features emerges from long and short views of a given video. Naively applying contrastive learning to long and short views results in a set of features that represent both long and short views similarly. We argue that this assumption is only valid for a subset of features, which we call stationary. The other subset, which we call non-stationary features, includes a set of features that aggregate from short to long views, *i.e.* combining attributes of several short views gives us the attributes of the long view, see Figure 1. Therefore, imposing a naive similarity between the long and short views is prone to ignoring the non-stationary features, which are crucial for different downstream tasks. Accordingly, we divide the final features into two disjoint subsets, which are later used in two separate contrastive losses. For the stationary features of a given long view, we provide a positive pair through the stationary features of a corresponding short view. Whereas an aggregated form of the non-stationary features of all corresponding short views forms a positive pair for the non-stationary features of the long view.

We validate our argument above empirically and train our method on the Kinetics dataset [17] without using any labels. We evaluate the model on several downstream tasks and datasets and analyse the learned representations; we highlight the main results here. We observe that stationary features are more beneficial for action classification, specifically on datasets with a high static bias [24] such as UCF101 [33]. On the other hand, non-stationary features achieve higher performance on downstream tasks involving more temporal aspects, such as temporal action segmentation on the Breakfast dataset [19]. We show that our non-stationary features perform substantially better than the stationary features on this dataset, supporting our hypothesis on the design choice. To the best of our knowledge, we are the first to conduct this evaluation for video representation learning. Our proposed method outperforms a strong contrastive learning baseline using data augmentations by significant margins on all three datasets and achieves state-of-

the-art retrieval results on HMDB51 [20]. Furthermore, we analyse the learned representations and find that stationary features capture attributes, that remain similar over time and can be detected in a few frames, while non-stationary features encompass more temporally varying attributes, which are revealed when observing more frames.

2. Related Work

Pretext tasks for videos. While image-level pretext tasks can be extended to videos [16, 18], the temporal structure in videos provides further opportunities for this purpose. For example, a supervision signal can be derived from the order of frames [27] or video clips [42]. Temporal context can be used to create a cloze procedure [25]. Other tasks are based on the frame rate, such as speed prediction [2], pace prediction [40], playback rate perception [43], or predicting temporal transformations [15].

Contrastive learning. Originally proposed in [6], instance recognition has become the underlying principle for many modern contrastive methods. The contrastive loss, which was first proposed in [32] and later popularized as *InfoNCE* loss by [37], involves positive and negative pairs of features. It aims to maximize the similarity of positive pairs – which are obtained by generating different views of the same data – and minimize the similarity of negative pairs. Views can be obtained from different channels [35], counting [28], permutation [26], or via augmentations.

The augmentations used to construct views, such as those explored in [4], have a substantial impact on the learned representation. For example, a heavy cropping leads to occlusion-invariant representations [31]. To prevent shortcuts via low-level image statistics (*e.g.* edges, corners, etc.), a challenging set of negatives is required. This is available via a memory bank [13], which enables the storage of a large set of of negative samples, or hard negative mining. For example, [9] and the follow-up work [10] obtain hard negatives from different spatio-temporal locations in the feature map, while [1] constructs negatives based on past and future clips. More recently, [5] and [8] have demonstrated that negatives can be omitted entirely. The role of hard positives is investigated in [11]. Our method can be interpreted similarly: By pairing long views with short or aggregated views, the difficulty of the task lies within the positive pairs, which require the model to make a connection in terms of stationary and non-stationary features.

3. Method

The key ideas of our **L**ong **S**hort View **F**eature **D**ecomposition (LSFD) method lie in decomposing the video representation into stationary and non-stationary features, and constructing short and long views of the data,
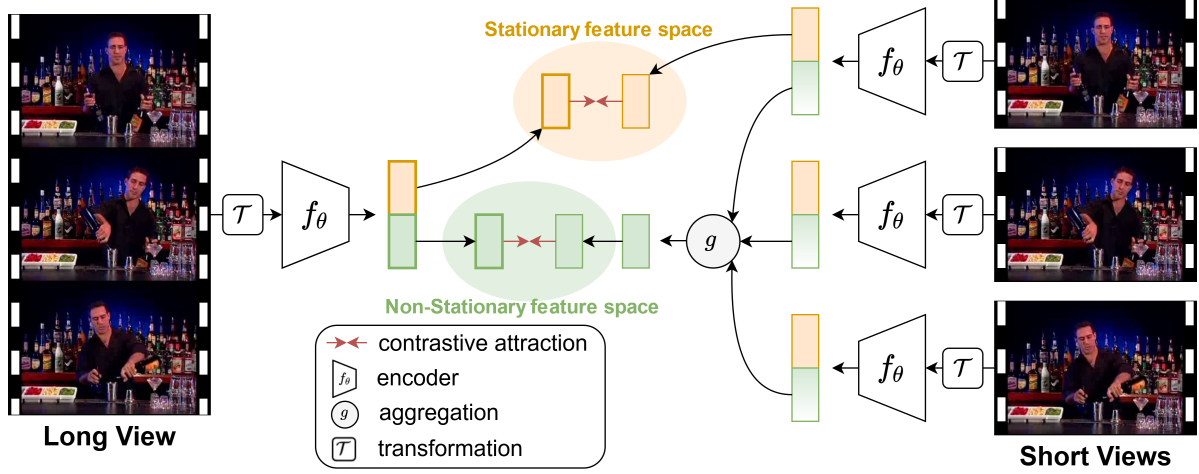
Figure 2. We extract features from long and short views and decompose them into stationary and non-stationary features, shown in orange and green, respectively. Stationary features remain similar over time and are shared by the long and short view, and serve as a positive pair (indicated by red arrows). Non-stationary features on the other hand capture temporal variances; we aggregate non-stationary features of short views to obtain the positive for the long view.

see Figure 2. Short views provide local attributes, as they span a limited temporal receptive field, while global temporal attributes are better perceived through the larger temporal receptive field of long views. Therefore, a naive solution of imposing similarity between short and long views is not necessarily optimal. We propose to establish a connection between long and short views by decomposing the representation space into two sections: One section represents stationary features that are shared between short and long views. The other section represents the attributes that should aggregate short views to the corresponding long view, which we call non-stationary features. As a result, the network is allowed to establish a proper connection between short and long views, without being forced to represent them similarly. We impose the concept of stationary and non-stationary features via separate contrastive losses on each section of the feature space. In the following, we discuss different components and design choices of our method in more detail.

3.1. Stationary and Non-Stationary Features

For a given sequence of video frames, we obtain a *long* view x_l that consists of all frames, and N non-overlapping sub-sequences, $x_s^{(1)}, \ldots, x_s^{(N)}$, which serve as *short* views. This allows us to construct positive pairs for stationary and non-stationary features, see Figure 2.

More precisely, we train a parametric function f_θ that takes a sequence of videos frames and maps them to a representation space: $f_\theta(x) = \xi = (\psi, \phi)$, where ψ, ϕ denote the stationary and non-stationary features of x, respectively. We compute the features of the long and short views by

feeding them to a shared backbone:

$$f_\theta(x_l) = \xi_l = (\psi_l, \phi_l),$$
$$f_\theta(x_s^{(i)}) = \xi_s^{(i)} = (\psi_s^{(i)}, \phi_s^{(i)}) \quad \text{for } i \in \{1, \ldots, N\}.$$

This allows us to establish a connection between long and short views. Stationary features represent attributes that remain the same over time, while non-stationary features are aggregated over time. Overall, they satisfy the following two properties:

$$\psi_l \simeq \psi_s^{(i)} \quad \text{for } i \in \{1, \ldots, N\}, \tag{1}$$
$$\phi_l \simeq g(\phi_s^{(1)}, \ldots, \phi_s^{(N)}). \tag{2}$$

Here, the aggregation function g can be any function that takes the non-stationary features of the short views and maps them to non-stationary features of the long view. Choosing an appropriate aggregation function is non-trivial and deserves extensive investigation. Candidates range from simple functions, such as a sum, to more elaborate and learnable functions, *e.g.* a linear transformation, MLP, or a recurrent network. We provide an ablation in Section 4.1.

In order to enforce the similarities in Eqs. (1) and (2) during training, we construct the following positive pairs to be used in two separate contrastive losses. The first pair $(\psi_s^{(j)}, \psi_l)$ aims at features shared between the long and a short view according to Eq. (1). The second pair $(g(\phi_s^{(1)}, \ldots, \phi_s^{(N)}), \phi_l)$ corresponds to Eq. (2), which aggregates the non-stationary features of the short views to match the non-stationary features of the long view.

For any given pair of features (z_1, z_2) obtained via the described procedure above, we compute the similarity following the recent trend in contrastive learning [4]: We apply

a learnable transformation h, here an MLP, and a temperature parameter τ to scale the cosine similarity, denoted by

$$\text{sim}_h(z_1, z_2) = \frac{1}{\tau} \frac{h(z_1)^T h(z_2)}{\|h(z_1)\|\|h(z_2)\|}. \quad (3)$$

3.2. Training Objective

Our training objective consists of three separate InfoNCE losses applied on stationary, non-stationary, and full features:

$$\mathcal{L} = \mathcal{L}_{\text{stationary}} + \mathcal{L}_{\text{non-stationary}} + \mathcal{L}_{\text{instance}}, \quad (4)$$

which we will discuss below. We use three different MLP heas h_s, h_n, and h_i for the three separate losses. We use a set of negatives that consists of random videos:

$$\mathcal{N} = \{f_\theta(\bar{x}_l) = \xi_{\text{neg}} = (\psi_{\text{neg}}, \phi_{\text{neg}})|\bar{x}_l \text{ is a random video}\}.$$

To avoid shortcuts via low-level video statistics, we apply the same set of standard video augmentations, including random resized crop, horizontal flip, and color augmentation, to long and short views independently.

Stationary Loss. Attributes that remain the same over time – such as non-moving objects or the background – are shared between the long view and all short views; and thus should be represented similarly. Therefore, the stationary feature of a short view should capture the same attributes as the stationary feature of the long view. We encourage such a property by applying the following loss function:

$$\mathcal{L}_{\text{stationary}} = -\log \frac{\exp(\text{sim}_{h_s}(\psi_s^{(j)}, \psi_l))}{\sum\limits_{\bar{\psi}_l \in \mathcal{N}_\psi \cup \{\psi_l\}} \exp(\text{sim}_{h_s}(\psi_s^{(j)}, \bar{\psi}_l))}. \quad (5)$$

Here, $\psi_s^{(j)}$ is the stationary feature of a randomly selected short view, and $\mathcal{N}_\psi = \{\psi_{\text{neg}}|(\psi_{\text{neg}}, \phi_{\text{neg}}) \in \mathcal{N}\}$.

Non-Stationary Loss. Complementary to stationary features, non-stationary features represent video content that varies: moving objects and persons, motion, changes in the scene, etc. These temporal changes *aggregate* over time, *i.e.* the non-stationary feature of the long view should capture the temporal changes happening in all sub-sequences. We encourage such a property by applying the following loss:

$$\mathcal{L}_{\text{non-stationary}} = -\log \frac{\exp(\text{sim}_{h_n}(\phi_g, \phi_l))}{\sum\limits_{\bar{\phi}_l \in \mathcal{N}_\phi \cup \{\phi_l\}} \exp(\text{sim}_{h_n}(\phi_g, \bar{\phi}_l))} \quad (6)$$

where $\phi_g = g(\phi_s^{(1)}, \ldots, \phi_s^{(N)})$ is an aggregated version of the short view non-stationary features, and $\mathcal{N}_\phi = \{\phi_{\text{neg}}|(\psi_{\text{neg}}, \phi_{\text{neg}}) \in \mathcal{N}\}$.

Instance Recognition Loss. Finally, we add an instance recognition loss on the long views by applying InfoNCE on the full feature. To that end we obtain a second view of the long video sequence $\hat{\xi}_l$ via standard video augmentations. This corresponds to standard contrastive learning.

$$\mathcal{L}_{\text{instance}} = -\log \frac{\exp(\text{sim}_{h_i}(\xi_l, \hat{\xi}_l))}{\sum\limits_{\bar{\xi}_l \in \mathcal{N} \cup \{\hat{\xi}_l\}} \exp(\text{sim}_{h_i}(\xi_l, \bar{\xi}_l))}. \quad (7)$$

4. Experiments

We now evaluate our LSFD method on different downstream tasks. Previous methods evaluate learned representations based on their performance on action recognition tasks; most commonly, models are finetuned on UCF101 and HMDB51. Despite the practical value of finetuning, it is an uncontrolled evaluation [14] and prone to overfitting. Moreover, action classification provides a rather incomplete assessment of the learned representations due to static biases in these datasets [24]. A sparse, global frame sampling strategy [41] works well for action recognition; even in the extreme case where only a single frame is used, accuracy remains high on UCF101. This suggests that temporal information is less important for these tasks. To get a better understanding of our representations, we extend the current evaluation protocol by adding another transfer learning task: action segmentation.

In contrast to action recognition, in which a single action label per video is given, action segmentation uses fine-grained temporal annotations. As the scene and background often stay the same throughout the video, a better temporal understanding is needed in order to temporally segment a video into the different actions that occur. Additionally, our pretrained model is frozen in this evaluation and serves as a feature extractor – no finetuning is involved.

To extend the evaluation via downstream tasks, we analyse the properties of the learned representations. Here, we are aiming to get a better understanding of which types of attributes are represented by stationary and non-stationary features, and investigate how and why they are different.

For self-supervised learning, we use **Kinetics-400** [17] and discard the labels. We use a 3D-Resnet18 backbone [12] in all experiments unless otherwise noted and pool the feature map into a single 512-dimensional feature vector. We decompose this feature vector into two equal chunks (of size 256) of stationary and non-stationary features. We use three separate MLP heads h_s, h_n, and h_i, which are removed after self-supervised training and will not be transferred to downstream tasks. We use a memorybank to store 65.536 negatives. We construct long views by sampling $N \cdot L$ frames, which we divide into N non-overlapping subsequences of L frames (short views). We set $L = 8$ in

| Self-Supervised Methods | | top1 Accuracy | |
Method	Arch	UCF	HMDB
Shuffle&Learn[†] [27]	CaffeNet	50.2	18.1
OPN[†] [22]	VGG	59.8	23.8
VCOP[†] [42]	R3D	64.9	29.5
PRP[†] [43]	R3D	66.5	29.7
BFP [1]	2D3D-R18	66.4	45.3
DPC [9]	2D3D-R34	75.7	35.7
MemDPC [10]	2D3D-R34	78.1	41.2
CBT[‡] [34]	S3D	79.5	44.6
SpeedNet [2]	S3D-G	81.1	48.8
CoCLR [11]	S3D-G	87.9	54.6
3D-ST-Puzzle [18]	C3D	61.2	28.3
MA Stats [39]	C3D	61.2	33.4
Temp Trans[‡] [15]	C3D	69.9	39.6
LSFD (Ours)	C3D	**79.8**	**52.1**
3DRot[‡] [16]	3D-R18	62.9	33.7
3D-ST-Puzzle [18]	3D-R18	65.8	33.7
VIE [44]	3D-R18	72.3	44.8
LA-IDT [36]	3D-R18	72.8	44.0
Temp Trans[‡] [15]	3D-R18	**79.3**	49.8
$\mathcal{L}_{instance}$	3D-R18	72.7	46.3
LSFD (Ours)	3D-R18	77.2	**53.7**

Table 1. Comparison to previous methods via finetuning on UCF101 and HMDB51 split 1. The first block shows methods with different architectures (Arch) and pretraining datasets, while the last two blocks encompass methods with the same architecture and pretraining dataset as our method. [†] denotes methods pretrained on UCF101, while [‡] denotes methods pretrained on Kinetics-600. The remaining methods are pretrained on Kinetics-400.

| Loss | Agg | N | top1 Accuracy | |
			UCF101	HMDB51
$\mathcal{L}_{instance}$	Sum	2	72.7	46.3
$+\mathcal{L}_{stationary}$	Sum	2	74.4	48.7
$+\mathcal{L}_{non\text{-}stationary}$	Sum	2	74.8	51.6
all	Sum	2	77.2	**53.7**
all	Linear	2	77.1	51.3
all	MLP	2	75.7	49.6
all	GRU	2	75.5	51.0
all	Sum	3	77.8	52.1
all	Sum	4	**78.0**	52.3

Table 2. Ablations via finetuning on UCF101 and HMDB51 for our different loss terms and several design choices, *i.e.* aggregation function (Agg) and number of sub-sequences N. We see an inverse relation between the aggregation function complexity and performance of the learned representation. Performance of our method is marginally effected for larger values of N on HMDB51.

all experiments unless otherwise noted, and provide experiments with different values of N. More details are provided in the Sup.Mat.

Our most important baseline is a **contrastive learning baseline**, trained only with $\mathcal{L}_{instance}$ in Eq. (7). This corresponds to traditional contrastive learning with data augmentations [13] applied to videos without an explicit focus on temporal variations. Here, we use $L = 16$ frames.

4.1. Action Recognition

The most widely used framework for evaluating self-supervised representations utilizes the self-supervised pre-trained weights to initialize a network, and then finetune it on a smaller annotated dataset. We consider the standard benchmarks UCF101 and HMDB51 and compare our method to previous self-supervised video representation methods in Table 1. Hyperparameters used for finetuning are provided in the Sup.Mat.

Since our method is based on RGB-video input only, we exclude multi-modal approaches such as [30, 29]. The first

block in Table 1 includes methods with shallower networks such as CaffeNet and significantly deeper architectures such as Resnet34 and S3D, and can therefore not be directly compared to our method. The second and third block includes methods using the same network architecture. Here, LSFD is trained with $N = 2$ and Sum aggregation.

With the 3D-Resnet18 backbone our method improves over our contrastive baseline ($\mathcal{L}_{instance}$) on both datasets by a fair margin. While we outperform previous methods with a similar architecture on HMDB51, our method is inferior to the method of [15] on UCF101. We attribute this smaller relative gain on UCF101 to its inherently static bias [24, 41], which is less pronounced on HMDB51. Moreover, note that since self-supervised learning is more relevant for smaller datasets like HMDB51, the results are more important compared with those on UCF101, which is a mid-sized dataset. For better comparisons, we additionally provide results with a C3D backbone (second block). Here, we outperform all previous methods. Namely, we outperform [15] on UCF101 and HMDB51 by 10% and 12%, respectively.

How much influence does each loss term have? Our LSFD method consists of three separate InfoNCE losses. We investigate the impact that each of them has on the resulting representation by progressively adding them to $\mathcal{L}_{instance}$; the results are provided in the first block of Table 2. Adding either $\mathcal{L}_{stationary}$ or $\mathcal{L}_{non\text{-}stationary}$ improves the performance on both UCF101 and HMDB51, suggesting that both stationary and non-stationary features are useful for action recognition. Note that the relative gain of adding $\mathcal{L}_{non\text{-}stationary}$ is higher on HMDB51 compared with $\mathcal{L}_{stationary}$. Adding both loss terms gives the highest performance, as is expected.
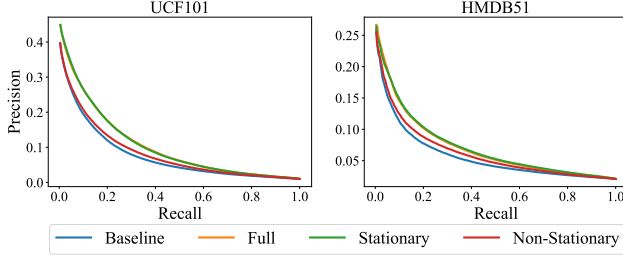
Figure 3. Precision-Recall-Curves for UCF101 and HMDB51. Stationary features are superior to non-stationary features on action recognition; both improve over our contrastive baseline.

What is the impact of aggregation? The function used to aggregate non-stationary features in Eq. (2) plays a critical role in our proposed method. The simplest, non-parametric function we consider takes the `Sum` over non-stationary features. We also test parametric and increasingly more complex aggregation functions: a `Linear` mapping, an `MLP`, and a `GRU` (for details we refer to the Sup.Mat.). The results are provided in the second block of Table 2. Overall, we find that the simplest aggregation in form of a `Sum` yields the highest performance. Using a non-parametric aggregation function puts more load on the backbone, obligating it to do the heavy lifting, whereas parametric aggregation functions relax the task, allowing a potential shortcut via changing the impact of each ϕ_i. We keep the `Sum` aggregation for the remaining experiments.

How many sub-sequences should we use? We ablate the effect of different numbers of sub-sequences N have on the representations in the third block of Table 2. We observed that training $N = 3$ and $N = 4$ from scratch was suboptimal; the task becomes increasingly difficult for larger N (details are provided in the Sup.Mat.). Therefore, we follow a curriculum learning strategy, that uses the pretrained model trained with $N - 1$ sub-sequences to initialize training for N sub-sequences. While increasing N improves the representation on UCF101, we observe a drop on HMDB51.

4.2. Video Retrieval

Next, we evaluate our method on video retrieval. We follow the protocol of [42]: We use the pretrained network to extract convolutional features at the last layer for all videos in the dataset. For each test video we retrieve the top k nearest neighbors from the training videos. For R@k results in Table 3, we count a correct retrieval if the k nearest neighbors contain at least one video of the same class.

While we improve over previous methods on HMDB51, our retrieval results on UCF101 are inferior to CoCLR [11]. Note that CoCLR uses a significantly deeper architecture. This is in line with our observation from the finetuning evaluation, where the relative improvement is higher on

HMDB51 compared with UCF101. Note that this evaluation doesn't measure *precision* of the retrieved samples properly for $k > 1$. For this reason, we also present precision-recall curves in Figure 3 (for details see Sup.Mat.). Our stationary features perform on-par with full features; non-stationary features slightly worse. This is even more pronounced on UCF101, where the static bias is higher [24]. Both stationary and non-stationary features improve over our contrastive baseline on both datasets.

4.3. Temporal Action Segmentation

We evaluate the universal representation learning capability of our method via a temporal action segmentation downstream task. Given an untrimmed video, the goal of temporal action segmentation is to simultaneously segment every action in time and classify each obtained segment. Recent state-of-the-art action segmentation methods such as [7, 23] train a temporal action segmentation model on top of pre-extracted features of the video frames. Usually, video frames are represented using deep 3D CNNs such as I3D [3] pretrained on the Kinetics dataset, or hand-crafted IDT features [38]. In this experiment, we use the Breakfast dataset. This dataset consists of untrimmed videos containing fine-grained actions, which are distinguishable mostly via temporal variations in the video, since the scene, actor, and objects remain similar throughout the video.

Using the Breakfast dataset allows us to better evaluate our methods ability to represent temporal variations. To that end, we use our frozen pretrained model as a feature extractor and compute features for each video frame of the datatset following [7]. Then, we add an action segmentation model on top, namely MS-TCN [7], and train for action segmentation in a fully supervised fashion. We use the official publicly available code of MS-TCN for training and evaluation; more training details are provided in the Sup.Mat. As the segmentation model relies on the pre-extracted features, this evaluation reveals more reliably than finetuning how well our learned representations are suited for this downstream task that involves a better temporal understanding. All models in this section are based on a 3D-Resnet18 backbone that operates on RGB input only.

Evaluation metrics. For evaluation of the segmentation models, we report the frame-wise accuracy (Acc), segmental edit distance, and the segmental F1 score at overlapping thresholds 10%, 25% and 50% as proposed by [21] (for details see Sup.Mat.). While the frame-wise accuracy provides a basic rating, it is rather insensitive to oversegmentation errors and short action classes; longer action classes have a higher impact than short action classes. The segmental edit distance measures how well the model predicts the ordering of action classes, and is not impacted by the duration of action classes. The segmental F1 score mea-

Method	Architecture	UCF101				HMDB51			
		R@1	R@5	R@10	R@20	R@1	R@5	R@10	R@20
VCOP [42]	R3D	14.1	30.3	40.4	51.1	7.6	22.9	34.4	48.8
VCP [25]	R3D	18.6	33.6	42.5	53.5	7.6	24.4	36.3	53.6
MemDPC [10]	2D3D-Resnet18	20.2	40.4	52.4	64.7	7.7	25.7	40.6	57.7
SpeedNet [2]	S3D-G	13.0	28.1	37.5	49.5	-	-	-	-
PRP [43]	R3D	22.8	38.5	46.7	55.2	8.2	25.8	38.5	53.3
Temp Trans [15]	3D-Resnet18	26.1	48.5	59.1	69.6	-	-	-	
CoCLR [11]	S3D-G	**53.3**	**69.4**	**76.6**	**82.0**	23.2	43.2	53.5	65.5
LFSD (Ours)	3D-Resnet18	44.9	64.0	73.2	81.4	**26.7**	**54.7**	**66.4**	**76.0**

Table 3. Comparison to other methods via nearest neighbor video retrieval on UCF101 and HMDB51.
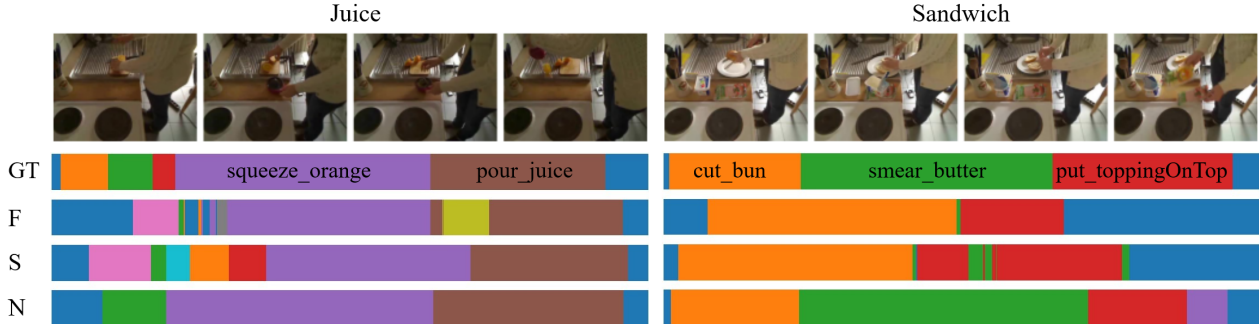

Figure 4. Qualitative results of LSFD for two videos from the Breakfast dataset, showing the quality of stationary (S), non-stationary (N), and full (F) features for action segmentation. Non-stationary features provide higher quality representations than stationary features, validating their ability to capture fine-grained temporal variations.

Method	Acc	Edit	F1@{10, 25, 50}		
Random init.	32.6	41.2	39.3	32.4	21.8
Kinetics sup.	45.1	54.5	47.1	41.7	31.0
$\mathcal{L}_{instance}$	57.9	50.3	44.6	39.9	31.4
LSFD, F	60.1	56.3	46.1	41.7	32.6
LSFD, S	58.9	54.7	40.6	35.7	28.8
LSFD, N	**60.6**	**60.0**	**52.0**	**42.8**	**35.3**

Table 4. Action Segmentation on the Breakfast dataset split 1. We report results for several baselines and our LSFD method. We further split up the full (F) feature in stationary (S) and non-stationary (N) features to investigate their effect. The backbone of all of the feature extraction models is 3D-ResNet18.

sures the general quality of the segmentation model, as it penalizes over-segmentation and is also insensitive to the duration of the action classes.

Results. In Table 4 we provide the results for our unsupervised LSFD method as well as several baselines, including random initialization, a model pretrained with Kinetics supervision, and our contrastive baseline ($\mathcal{L}_{instance}$). Interestingly, we observe that all features obtained via unsupervised pretraining, *i.e.* the second block of the table, improve over

a model trained with Kinetics supervision. This validates our argument that unsupervised learning may provide richer representations, capable of transferring better to different, unseen tasks. Furthermore, our method improves over the contrastive baseline as well as the supervised baseline by a significant margin, demonstrating that our long and short views enable a better temporal understanding. While the difference in accuracy is noticeable, it is even more prominent in the segmental edit distance and F1 scores, which better measure the overall quality of the segmentations.

To investigate the feature decomposition we use only the stationary or only the non-stationary feature of our pretrained model as input for the segmentation model. The quantitative results in Table 4 show that our non-stationary features outperform stationary features across all metrics, providing higher quality representations for the temporal segmentation model. As can be seen in the raw frames in Figure 4, the static information across the video remains similar for most of the temporal segments. Hence, temporally segmenting such videos to fine-grained actions is a challenging problem requiring high quality features in terms of temporal variation representation. Additionally, we provide some qualitative results in Figure 4, where we observe a higher quality of non-stationary features compared with stationary features.
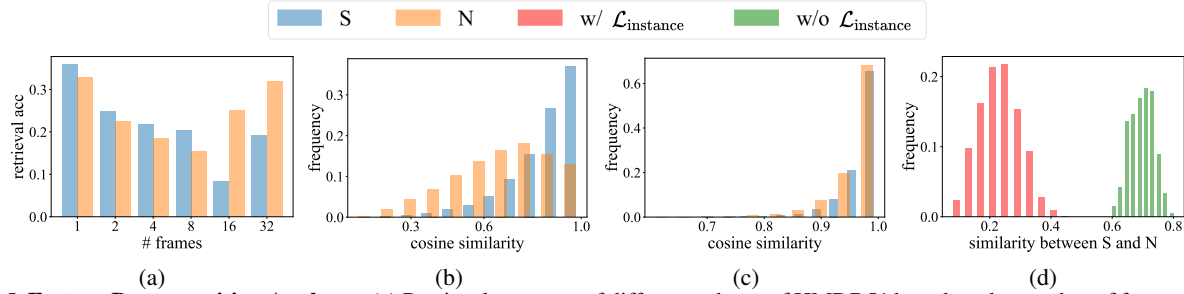
Figure 5. **Feature Decomposition Analyses.** (a) Retrieval accuracy of different subsets of HMDB51 based on the number of frames needed for classification. We observe that non-stationary features (N) outperform stationary features (S) on retrieving videos that require more frames (long views). The opposite happens for videos that require less frames (short views). Cosine similarities over time (b) with and (c) without $\mathcal{L}_{\text{instance}}$. We compute similarities of S and N features over time and show the histogram of the computed similarities on HMDB51. (b) S features are centered around high values when training with $\mathcal{L}_{\text{instance}}$; N features are distributed more uniformly. (c) When training without $\mathcal{L}_{\text{instance}}$ both S and N features are stable over time. (d) Similarities between S and N features when trained with and without $\mathcal{L}_{\text{instance}}$. We observe that removing $\mathcal{L}_{\text{instance}}$ results in a degenerate solution where S and N features are similar.

4.4. Feature Decomposition Analyses

The experiments above validate our hypothesis on feature decomposition by evaluating the performance of stationary and non-stationary features on dedicated downstream tasks. In the following, we conduct more specific analyses to obtain a better insight into our decomposed features. For experimental details we refer to the Sup.Mat.

Do we capture short- and long-term attributes? Figure 5(a) shows the retrieval accuracy of stationary and non-stationary features among the videos that can be classified with different numbers of frames. In each case, we exclude videos that are correctly classified with fewer frames, *i.e.* video that can be classified with a single frame are excluded from the set of videos that can be classified with two frames, etc. While stationary features achieve stronger retrieval accuracy on videos that can be classified with less frames (short views), *i.e.* less temporal context, non-stationary features are more beneficial for videos that require more frames (long views) and a longer temporal context.

Are stationary features more stable over time? We divide a given video into clips of 16 frames, compute the stationary and non-stationary features of all clips and similarities of the features over time. We show the histogram of the computed similarities on HMDB51 in Figure 5(b). Stationary features are centered around high values, while non-stationary features are distributed more uniformly. This suggests that stationary features remain more stable over time, whereas non-stationary feature vary.

What is the impact of $\mathcal{L}_{\text{instance}}$? Figure 5(c) shows the same histogram as Figure 5(b) for a network trained without $\mathcal{L}_{\text{instance}}$. We observe that in contrast to Figure 5(b), stationary and non-stationary features behave similarly – both are

stable over time. Moreover, we compare the histogram of similarities between stationary and non-stationary features of long views with and without $\mathcal{L}_{\text{instance}}$ in Figure 5(d). Stationary and non-stationary features are very similar to each other when trained without $\mathcal{L}_{\text{instance}}$, suggesting a degenerate solution that copies stationary features as non-stationary ones. One reason why $\mathcal{L}_{\text{instance}}$ avoids this could be that it pushes the network towards exploiting the full capacity of the feature space, preventing redundant information in the full feature space.

5. Conclusion

In this paper, we have introduced a novel method to decompose video representations into stationary and non-stationary features via contrastive learning from long and short views. We evaluated the learned representations extensively on multiple downstream tasks and datasets, and investigated various design choices and the role of stationary and non-stationary features. Overall, we find an interesting correlation between the type of feature and the nature of the downstream task: Stationary features perform better on tasks and datasets with static biases, such as action recognition on UCF101, while non-stationary features are more beneficial for action segmentation, which requires better temporal understanding. We demonstrated a substantial gain in performance on the HMDB51 dataset for action recognition, and outperform a supervised baseline on the Breakfast dataset for action segmentation. Furthermore, we analysed our feature decomposition and found that stationary features are more stable over time, while non-stationary features vary.

References

[1] Nadine Behrmann, Juergen Gall, and Mehdi Noroozi. Unsupervised video representation learning by bidirectional feature prediction. In *WACV*, 2021. 2, 5

[2] Sagie Benaim, Ariel Ephrat, Oran Lang, Inbar Mosseri, William T. Freeman, Michael Rubinstein, Michal Irani, and Tali Dekel. Speednet: Learning the speediness in videos. In *CVPR*, 2020. 2, 5, 7

[3] João Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In *CVPR*, 2017. 6

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 2, 3

[5] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *arXiv*, abs/2011.10566, 2020. 2

[6] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. In *TPAMI*, 2016. 2

[7] Yazan Abu Farha and Juergen Gall. MS-TCN: Multi-stage temporal convolutional network for action segmentation. In *CVPR*, 2019. 6

[8] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent - a new approach to self-supervised learning. In *NeurIPS*, 2020. 2

[9] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. In *ICCV Workshop*, 2019. 2, 5

[10] Tengda Han, Weidi Xie, and Andrew Zisserman. Memory-augmented dense predictive coding for video representation learning. In *ECCV*, 2020. 2, 5, 7

[11] Tengda Han, Weidi Xie, and Andrew Zisserman. Self-supervised co-training for video representation learning. In *NeurIPS*, 2020. 2, 5, 6, 7

[12] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and ImageNet? In *CVPR*, 2018. 4

[13] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 2, 5

[14] Kaiming He, Ross Girshick, and Piotr Dollar. Rethinking imagenet pre-training. In *ICCV*, 2019. 4

[15] Simon Jenni, Givi Meishvili, and Paolo Favaro. Video representation learning by recognizing temporal transformations. In *ECCV*, 2020. 2, 5, 7

[16] Longlong Jing and Yingli Tian. Self-supervised spatiotemporal feature learning via video rotation prediction. *arXiv*, abs/1811.11387, 2018. 2, 5

[17] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *arXiv*, abs/1705.06950, 2017. 1, 2, 4

[18] Dahun Kim, Donghyeon Cho, and In So Kweon. Self-supervised video representation learning with space-time cubic puzzles. In *AAAI*, 2019. 2, 5

[19] Hilde Kuehne, Ali B. Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *CVPR*, 2014. 2

[20] Hilde Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. HMDB: A large video database for human motion recognition. In *ICCV*, 2011. 2

[21] Colin Lea, Michael D. Flynn, René Vidal, Austin Reiter, and Gregory D. Hager. Temporal convolutional networks for action segmentation and detection. In *CVPR*, 2017. 6

[22] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequences. In *ICCV*, 2017. 5

[23] Shi-Jie Li, Yazan AbuFarha, Yun Liu, Ming-Ming Cheng, and Juergen Gall. MS-TCN++: Multi-stage temporal convolutional network for action segmentation. *TPAMI*, 2020. 6

[24] Yingwei Li, Yi Li, and Nuno Vasconcelos. Resound: Towards action recognition without representation bias. In *ECCV*, 2018. 1, 2, 4, 5, 6

[25] Dezhao Luo, Chang Liu, Yu Zhou, Dongbao Yang, Can Ma, Qixiang Ye, and Weiping Wang. Video cloze procedure for self-supervised spatio-temporal learning. In *AAAI*, 2020. 2, 7

[26] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *CVPR*, 2020. 2

[27] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. Shuffle and learn: Unsupervised learning using temporal order verification. In *ECCV*, 2016. 2, 5

[28] Mehdi Noroozi, Hamed Pirsiavash, and Paolo Favaro. Representation learning by learning to count. In *ICCV*, 2017. 2

[29] Mandela Patrick, Yuki M. Asano, Polina Kuznetsova, Ruth Fong, João F. Henriques, Geoffrey Zweig, and Andrea Vedaldi. Multi-modal self-supervision from generalized data transformations. *arXiv*, abs/2003.04298, 2020. 5

[30] A. J. Piergiovanni, Anelia Angelova, and Michael S. Ryoo. Evolving losses for unsupervised video representation learning. In *CVPR*, 2020. 5

[31] Senthil Purushwalkam and Abhinav Gupta. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. In *NeurIPS*, 2020. 2

[32] Kihyuk Sohn. Improved deep metric learning with multiclass n-pair loss objective. In *NIPS*, 2016. 2

[33] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv*, abs/1212.0402, 2012. 2

[34] Chen Sun, Fabien Baradel, Kevin Murphy, and Cordelia Schmid. Learning video representations using contrastive bidirectional transformer. *arXiv*, abs/1906.05743, 2019. 5

[35] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv*, abs/1906.05849, 2019. 2

[36] Pavel Tokmakov, Martial Hebert, and Cordelia Schmid. Unsupervised learning of video representations via dense trajectory clustering. In *ECCV Workshops*, 2020. 5

[37] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv*, abs/1807.03748, 2018. 2

[38] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 6

[39] Jiangliu Wang, Jianbo Jiao, Linchao Bao, Shengfeng He, Yunhui Liu, and Wei Liu. Self-supervised spatio-temporal representation learning for videos by predicting motion and appearance statistics. In *CVPR*, 2019. 5

[40] Jiangliu Wang, Jianbo Jiao, and Yun-Hui Liu. Self-supervised video representation learning by pace prediction. In *ECCV*, 2020. 2

[41] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks for action recognition in videos. *TPAMI*, 2019. 4, 5

[42] Dejing Xu, Jun Xiao, Zhou Zhao, Jian Shao, Di Xie, and Yueting Zhuang. Self-supervised spatiotemporal learning via video clip order prediction. In *CVPR*, 2019. 2, 5, 6, 7

[43] Yuan Yao, Chang Liu, Dezhao Luo, Yu Zhou, and Qixiang Ye. Video playback rate perception for self-supervised spatio-temporal representation learning. In *CVPR*, 2020. 2, 5, 7

[44] Chengxu Zhuang, Tianwei She, Alex Andonian, Max Sobol Mark, and Daniel Yamins. Unsupervised learning from video with deep neural embeddings. In *CVPR*, 2020. 5