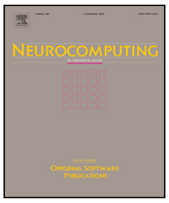
Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



STCLR: Sparse Temporal Contrastive Learning for Video Representation

Hussein Altabrawee [a,b], Mohd Halim Mohd Noor [a,*]

[a] Universiti Sains Malaysia, School of Computer Sciences, Main Campus, Gelugor, 11800, Penang, Malaysia
[b] Al-Muthanna University, Computer Center, Main Campus, Samawah, 66001, Al-Muthanna, Iraq

ARTICLE INFO

Communicated by Z. Wang

Dataset link: https://www.crcv.ucf.edu/data/UCF101/UCF101.rar, https://serre-lab.clps.brown.edu/resource/hmdb-a-large-human-motion-database, http://www.svcl.ucsd.edu/projects/resound/dataset.html

Keywords:
TCLR
Self-supervised learning
Visual representations learning
Action recognition

ABSTRACT

Temporal Contrastive Learning for Video Representation (TCLR) is the first contrastive framework that uses temporal losses to enforce the temporal distinctiveness of the features explicitly. However, its local–local temporal contrastive loss contrasts non-overlapping dense adjacent clips that cover small motion dynamics due to temporal coherence in videos. These adjacent local clips could represent the same action phase, and that makes the local–local loss enforce temporal distinctiveness of the features within the same action phase, whereas much better distinctiveness comes from contrasting different action phases. To overcome these limitations, Sparse TCLR framework (STCLR) is proposed, which uses random sparse sampling to sample local clips that cover different motion dynamics of the video. Using sparse local clips with the local–local loss overcomes the need for the second TCLR temporal loss, the global–local loss. In addition, a novel temporal pretext (Same Speed Localization) is proposed for intra/inter action temporal self-supervised learning and combined with the temporal loss. We prove that TCLR can be simplified to a framework that uses two losses only instead of three. STCLR achieves better feature transferability by a significant margin than TCLR on video retrieval. STCLR outperforms TCLR by 0.58%, 1.21%, 0.88%, and 0.98% and by 4.32%, 5.97%, 6.51%, and 2.78% on Top 1, Top 5, Top 10, and Top 20 retrieval measures on UCF-101 and HMDB-51, respectively. On action recognition, STCLR outperforms TCLR on Diving-48 by 9.68%, which is a significant improvement, and on HMDB-51 by 0.83%, while it achieves a lower accuracy by 0.72% on UCF-101.

1. Introduction

Creating and labeling extremely large video datasets is indispensable for video understanding progress and development [1]. However, the manual labeling process is extremely time-consuming, expensive, and tedious [2]. In addition, labeling videos is much more challenging than labeling images due to the temporal dimension of video data [1]. To overcome these limitations, self-supervised learning relies on using the structure and the internal properties of the data to learn useful representations from the data directly without any human annotations or supervision [3]. Fig. 1 shows the overall process of self-supervised learning, which has two consecutive phases: self-supervised pre-training and downstream task evaluation.

First, the intrinsic co-occurrence patterns, relationships, attributes, and structural elements of the unlabeled data are utilized to identify or create the self-supervision signal (e.g., temporal coherence). Then, the pretext task is defined and structured utilizing the self-supervision signal to automatically label the unlabeled data. An example of a pretext is temporal order verification [4], where the model is pre-trained to classify whether an input clip has the correct order of frames or not. Unlabeled video clips are sampled based on the pretext requirements and used along with the produced labels to train the model for solving the pretext task.

Next, the acquired representations are transferred to the target task using transfer learning, which includes fine-tuning, linear probing, or employing the self-supervised pre-trained model as a feature extractor. The labeled dataset is utilized in this phase for the supervised training/evaluation of the target task, and its performance is used to assess the efficacy of the pretext and the learned visual representations. Self-supervised techniques are categorized into four categories of pretext tasks: context-based, generative, contrastive, and contrastive generative [3,5]. Using self-supervised learning on the vast and massive volume of freely unlabeled videos from the internet has great potential to advance video understanding [1].

Various video properties and characteristics have been used in designing context-based pretexts. For example, video speed is employed in many pretexts, such as determining whether a video is being played at normal speed or at an accelerated speed [6], sorting shuffled clips based on their speed [7], and speed prediction and video generation [8]. Other pretexts rely on the temporal coherence of videos as a supervision signal; such pretexts include repeated scene localization [9],

* Corresponding author.
E-mail addresses: hussein@mu.edu.iq (H. Altabrawee), halimnoor@usm.my (M.H. Mohd Noor).

https://doi.org/10.1016/j.neucom.2025.129694
Received 23 September 2024; Received in revised form 11 January 2025; Accepted 8 February 2025
Available online 18 February 2025

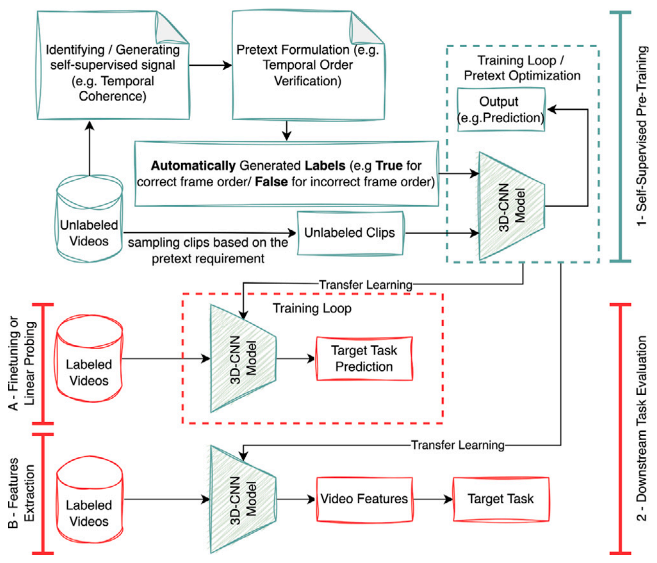

Identifying / Generating self-supervised signal (e.g. Temporal Coherence)
Pretext Formulation (e.g. Temporal Order Verification)
Training Loop / Pretext Optimization
Output (e.g.Prediction)
Automatically Generated Labels (e.g True for correct frame order/ False for incorrect frame order)
Unlabeled Videos
sampling clips based on the pretext requirement
Unlabeled Clips
3D-CNN Model
1- Self-Supervised Pre-Training
Transfer Learning
Training Loop
A - Finetuning or Linear Probing
Labeled Videos
Target Task Prediction
B - Features Extraction
Video Features
Target Task
2 - Downstream Task Evaluation

Fig. 1. The overall process of self-supervised learning.

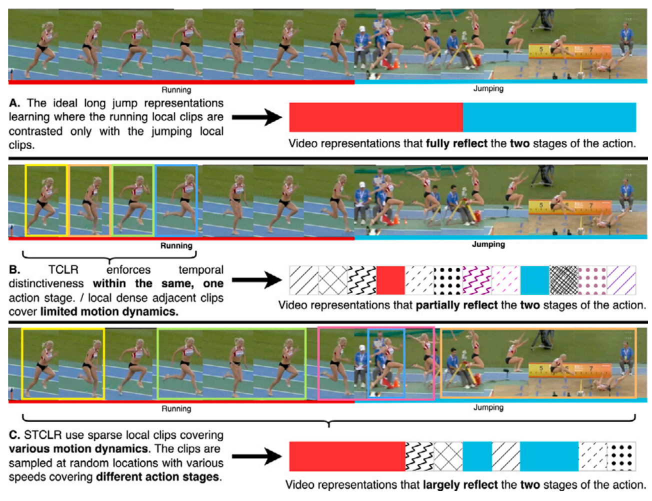

Running
Jumping
A. The ideal long jump representations learning where the running local clips are contrasted only with the jumping local clips.
Video representations that fully reflect the two stages of the action.
B. TCLR enforces temporal distinctiveness within the same, one action stage. / local dense adjacent clips cover limited motion dynamics.
Video representations that partially reflect the two stages of the action.
C. STCLR use sparse local clips covering various motion dynamics. The clips are sampled at random locations with various speeds covering different action stages.
Video representations that largely reflect the two stages of the action.

Fig. 2. Long jump has two distinct stages, running and jumping that are represented by red and cyan, respectively. Yellow, orange, green, blue, and pink boxes represent local clips. A. Contrasting running with jumping generates two-stage representations. B. TCLR generates suboptimal representations that partially model the two stages. C. STCLR contrasts various motion dynamics leading to much more distinct two-stage representations compared to TCLR.

classifying videos temporal transformations [10], tracking large video frame regions and forming fine-grained pixel-level correlations between successive video frames [11], and optimizing for temporal cycle consistency, in which video sequences representing the same action are temporally aligned [12].

Several pretexts rely on the pixels' values as a supervision signal; such pretexts include video frame reconstruction using a reference frame [13,14], using the preceding frame [15], or using a reference frame with keypoints [16]. In addition, future latent representations are used as a supervision signal in [17,18]. Numerous non-video based pretexts have been proposed, such as aligning the temporal information with the structural information of a temporal graph [19]. The temporal information is represented by the temporal conditional intensity, which is modeled using the historical neighbor sequences. The structural information is represented by the structural conditional intensity, which is modeled at two different scales, local and global. The features from the high-order neighbor sequences are aggregated to create the local structural intensity, while all the graph nodes are utilized to create a global representation. In addition, many self-supervised methods combine context-based pretexts with another self-supervised approach, contrastive learning [20–22].

Contrastive learning uses contrastive losses to learn useful data representations. The primary goal is learning a general feature function. The function's role is to map the input data to representations located in a hypersphere feature space [2]. Contrastive methods strive to learn features that are invariant to dissimilar data views of the same data sample. To achieve this, contrastive methods try to make the positive pairs close to each other while making the negative pairs far away from each other in the representation space. In addition, contrastive learning uses heavy data augmentations and the convolutional neural networks' abstraction capability to learn some aspects of the semantic structures of the input data [2].

Videos have rich temporal multimodal information used in a variety of tasks that are not limited to video understanding. For instance, [23] used the temporal information in videos for acoustic event classification, where a temporal multi-modal graph represents an acoustic event. The graph nodes represent audio segments or video segments related to the event. The temporal relationships that exist naturally among the nodes are employed as timestamps on their edges. The temporal dimension in videos plays an important role in video understanding tasks since it contains much richer information when compared with static images; however, the weight of that role varies. In action recognition, there are actions that require motion modeling to be classified; pushups and juggling balls are examples. In contrast, some actions are static and can be classified without using motion; walking with a dog and playing paintball are examples [24]. The motion in videos varies through time for actions that have multiple action stages, such as long jump as illustrated in Fig. 2, which consists of two distinct action phases, running and jumping [1]. In addition, the motion signal or the spatiotemporal context in videos changes very slowly and smoothly through time due to the high redundancy between consecutive video frames [13,25–28].

The Temporal Contrastive Learning for Video Representation Framework (TCLR) is a contrastive framework that uses two temporal contrastive losses, local–local loss and global–local loss, combined with the standard instance contrastive loss to explicitly influence the video representations to be temporally distinct [1]. However, the local–local loss contrasts dense, non-overlapping neighboring clips extracted from the video, as illustrated in part B of Fig. 2 by the yellow, orange, green, and blue boxes. This sampling process is not optimal for learning temporally distinct features because of two reasons: First, the clips are densely sampled, so they cover little motion due to the high redundancy in the consecutive video frames. Second, the clips are next to one another, neighboring clips, so there is a high possibility that they represent the same action stage (running in Fig. 2). In this case, the TCLR local–local loss considers clips from one action stage as negatives to one another. An optimal loss would choose clips from different action stages to be contrasted, as illustrated in part C of Fig. 2.

To overcome these limitations, we propose an improved TCLR framework called the Sparse TCLR (STCLR) framework. STCLR employs random sparse sampling to select local clips that represent a variety of motion dynamics, as illustrated in part C of Fig. 2 by the yellow, orange, green, blue, and pink boxes. STCLR implements two design choices to modify the TCLR local clips' sampling procedure. The first is sampling the local clips at random locations from the video instance to increase the probability of covering different action stages. The second is varying the sampling rate to increase the temporal extent of the covered motion and increase the temporal diversity of the local clips. Our method creates very different local clips, and they cannot be combined to create the global clip used by the TCLR global–local loss due to their random temporal locations and their different sampling rates. Even if there is a way to combine them to form the global clip, it would have temporally incoherent motion due to the abrupt spatiotemporal change between its frames. Modeling this global clip with the 3D-CNN would not be ideal because CNNs rely on the strong local correlations property of the input to work well and extract useful features. For this reason, we propose to neglect the use of the global–local temporal contrastive loss and only use the local–local temporal contrastive loss. In addition, we propose to combine the contrastive objective with a novel temporal pretext related to the speed of videos to enrich the video representations. The contributions of this research are as follows:

- We prove that the TCLR framework can be simplified to a framework that uses only two losses instead of three losses by changing its input clips and its sampling method.
- A novel framework, Sparse TCLR, that combines the Simplified TCLR with a novel temporal pretext is proposed. The STCLR relies on random sparse sampling and video speed modeling to achieve temporal distinctiveness of the features explicitly.
- Existing temporal pretexts only learn from a single video instance and ignore other videos of the same action or different actions. Therefore, a novel temporal pretext, Same Speed Localization (SSL), is proposed for intra/inter action temporal self-supervised learning.
- The Simplified TCLR, STCLR, and the SSL pretext are evaluated using three benchmark datasets: Diving-48, UCF-101, and HMDB-51. They achieve an outstanding performance in action recognition and video retrieval.

2. Related work

Sparse sampling has been used successfully to capture essential motion dynamics from videos. [26] demonstrated that sparse temporal sampling is an effective technique for modeling the long-range temporal structure of videos, which is crucial for motion dynamics understanding. In this approach, sparse temporal sampling was used to generate video-level representation by extracting a sequence of short snippets that were sampled sparsely and distributed uniformly over the temporal dimension of the long video. [28] confirmed that linking meaningful transformations of entities or objects in videos over time can be achieved using sparse temporal sampling. Using only sparsely and uniformly sampled frames, their approach can learn effectively from and reason about the long-term temporal dependencies and relations between the frames of the video.

In [29], various sized 3D space–time tubes were sparsely sampled from the full video to create learnable tokens used for jointly learning video and image representations by a vision transformer. [30] proposed a method where sparse frames were uniformly sampled from either one or a few sparsely sampled short video clips. These clips were extracted from the original long video during each training step to facilitate cost-effective video and language tasks end-to-end learning. [30] found that sparse sampling enhances the generalization of the model and is often more accurate when compared to dense sampling of offline features of long videos.

[31] proposed a video-language model where each raw video was depicted by only two comprehensive video clips, each consisting of a few frames that span the entire video. The video was divided into a few segments, and then a single video frame was randomly sampled from each segment to create a comprehensive clip. [32,33] have examined video masked autoencoders for pre-training and represented videos as tubes and investigated the sparseness in terms of masking. They have found similar results, indicating that sparseness is advantageous.

We utilize random sparse temporal sampling to extract various video segments that represent various temporal dynamics of the same video, which are then contrasted to one another to promote temporal distinctiveness learning. However, our sparse temporal sampling technique differs from [26,28,30] because our method samples sparse 3D clips at various sampling rates from various locations; as a result, these clips could overlap, whereas [26,28,30] sampled sparse frames uniformly. In addition, our method does not rely on using sparse tubes of different shapes as used in [29], or dividing the video into segments and sampling only two comprehensive clips as in [31], or representing videos as tubes as used in [32,33].

The most related pretexts in the literature are those that rely on the video speed, such as predicting if the given clip is sampled at normal speed or accelerated [6], speed classification and sorting the clips based on their speed [7], speed prediction and video generation [8], speed prediction and other temporal and spatial relations predictions [34], pace prediction and contrastive learning [21], recognizing temporal transformations [10], speed and direction and overlap/order classification in the auxiliary objective in [20]. These pretexts learn the temporal dynamics from video clips that represent one single action. This limits the learning ability of the network since it solves the pretexts by comparing clips sampled from a single action instance only without taking into consideration other instances from the same action or instances from different actions.

Video speed modeling is a very effective technique for self-supervised visual representation learning. [10] demonstrated that modeling the speed of videos yields visual representations that accurately capture long-range temporal motion statistics. These statistics require long-range reasoning and observation in the video temporal dimension. The model learns the important temporal aspects of video representation by discriminating and focusing on the temporal variation in videos while the spatial content has almost been preserved. Modeling the dynamics' subtleties of these motions entails more than motion estimation between two or three frames. Furthermore, these subtleties are exclusive to the moving entities and objects, requiring object detection and recognition. It is difficult to determine accurately the speed of an action, since different types of actions naturally have varying motion speeds. For example, the speed of applying make-up is very different from the speed of biking. Completing this task usually requires more profound comprehension of the physics and objects depicted in the video [10].

The most similar approach to our pretext is the contrastive objective of [20]. It applied the same relative transformations to two different examples and used the vectors that encode these relative transformations as positives. The approach in [20] differs from our pretext because it explicitly encouraged the feature vectors of the relative transformations of these different examples to be closely represented in the feature space by using contrastive loss. Our pretext relies on the network's ability to find spatially and temporally different clips that have the same speed using cross-entropy loss. The network has to learn the inter-class differences and dynamics for these different actions. In addition, the equivariance contrastive objective in [20] is practically difficult to optimize alone, and they needed three additional auxiliary pretexts to facilitate the learning process.

Contrastive video representation learning has been in the spotlight of the computer vision research community following the success of contrastive learning in the image domain [1]. In contrastive learning, choosing the positives and the negatives plays a vital role in controlling the representations' quality and making the representations temporally distinct or temporally invariant.

Positive clips could be sampled from the same video as the anchor, yet at different temporal locations, while negatives could be sampled from other different videos [35–37]. The speed in videos can also be used to create the positives and the negatives [38,39]. In [39], the positive pair was sampled from the same video, yet one clip was short and the other was long. The short clip covers a short temporal timestamp due to its small sampling frame rate, while the long clip covers a much larger temporal extent due to its large sampling frame rate. Short and long negatives were extracted from different videos. In [40], different views of a video frame were regarded as positives, and all the other views of a random frame sampled from random other videos were regarded as negatives. Unlike our proposed framework that enforces temporal distinctness using explicit temporal loss, these approaches focus on learning temporally invariant representations by using distant clips sampled from the same video as positives.

Another research has focused on mining inter-video positives, whereby the positives were sampled from other videos than the anchor to potentially group semantically similar videos together. RGB view and optical flow view were used to mine positive clips that are semantically similar to the anchor [41,42]. In [43], a contrastive probabilistic framework was proposed where the positives and the negatives were mined using the proposed probabilistic distance, which was used to group more semantically similar clips. In [44], a clustering method was used to create positive and negative clips. For a given video, the clips that were mined randomly from the same cluster as the video were considered positives, while negative clips were sampled from different clusters. Unlike these works, our instance contrastive loss uses two augmented clips sampled from the same video as positives, while the negatives are sampled from other videos. In addition, our temporal loss is used to achieve temporal distinctness of the video representation.

The temporal dimension of videos was employed to generate hard negatives that were used with a contrastive objective to learn useful representations. In [22,45], an intra-video negative clip was extracted from a different timestamp of the same video as the anchor. In [46], frame repeating and frame shuffling along the temporal dimension were used to generate intra-video negative clips from the anchor. Even though these approaches used hard negatives to ease the invariance constraint, they did not explicitly contrast the timestamps of the same video to represent the temporally distant clips of the video by distinct features. In addition, we do not rely on the triplet loss or use frame shuffling, frame repeating, or any other auxiliary modalities.

Iterative clustering was used in [47] to generate videos' pseudo-labels that were used to mine inter-video positives. Hard negatives were also mined using the pseudo-labels. Two triplet margin losses were used, instance-based triplet loss and temporal discrimination loss. The temporal discrimination triplet loss, similar to the TCLR local–local loss, uses a non-overlapping intra-video negative clip or a negative clip sampled from other videos in the same cluster as the anchor. Unlike [47], our approach contrasts five intra-video clips, sampled at random temporal locations with different speeds, using the InfoNCE contrastive loss. In addition, we do not use clustering or any auxiliary modality, such as optical flow, which was used in [47].

The video's multi-modal nature was used with contrastive objectives to learn video representations. For instance, text and RGB clips were used in [48,49]. Audio and the visual modality were used to drive the learning process in [50,51]. In addition, RGB videos, audio, and text were used in [52].

A recent research [53] highlighted that multimodal contrastive learning is limited by its reliance on the presumption that representation learning can only be achieved using the information shared between modalities and that this information is sufficient for the target tasks. [53] proved that this presumption is not accurate because information can exist solely within a single modality yet remain pertinent to the target task. [53] designed a new method capable of capturing both shared and distinct information. This method factorizes the information that is pertinent to the target task into shared representation and unique representation. Then, it acquires target task pertinent information using mutual information lower bounds maximization. Furthermore, it removes unrelated information using mutual information upper bounds minimization. In addition, it relies on multimodal data augmentations for achieving target task relevance.

Different from contrastive learning factorization, [54] proposed using multimodal redundancy reduction to discriminate between intermodal and intra-modal embeddings. This approach decouples the shared and distinct (modality-unique) representations across modalities for multimodal self-supervised learning. The dimensions of the feature embedding were divided into two parts, cross-modal shared and modality-distinct. The normalized cross-correlation matrix of the shared dimensions and the distinct dimensions between two modalities was computed during training. Next, the shared dimensions matrix was optimized toward the identity, while the distinct dimensions matrix was optimized toward zero. Consequently, shared embeddings aligned over modalities, whereas modality-specific embeddings were repelled. To prevent collapse, intra-modal learning was optimized where the full dimensions of the embedding were employed. The cross-correlation matrix, which was computed between two augmented views created from the same modality, was optimized toward the identity. Unlike these methods, we use the RGB visual modality only as the input to our framework without using any other auxiliary signal.

In [55], an unsupervised domain complementary adaptation approach was developed where domain mutual/domain structure-oriented contrastive learning was utilized. It leverages the diversity among sources and the discriminability of each one of them. Several domain branch networks were employed for learning various views of domain-invariant discriminative representations from each source. In addition, the domain-invariant representations coming from all domain branch networks were employed for training an ensemble classification network. This makes the domain branch networks provide diverse knowledge. [55] formulated a domain mutual contrastive loss that compels the domain branch networks to differ from each other while maintaining consistency with the ensemble classification network to acquire different domain-invariant features. Furthermore, a domain structure-oriented contrastive loss was presented to enhance the discriminability of domain branch networks by learning the intrinsic discriminative neighborhood structure across each target and source domain. Unlike this method, we use contrastive learning to explicitly contrast the timestamps of the same video to model various temporally distant clips of the video by distinct features.

The TCLR framework uses three contrastive losses [1], temporal local–local loss and global–local loss, to achieve the temporal distinctness of the representations and the standard instance contrastive loss. The local–local loss operates on non-overlapping neighboring clips, which are extracted from different timestamps from the same video instance. This loss guarantees that these clips are represented by distinct features. Positives are generated from each clip by applying random augmentations to the clip, while all the other non-overlapping clips and their augmentations are considered negatives. The global–local loss operates on the clip's feature map timesteps. This loss encourages the feature map's timesteps that are extracted from a global clip to be represented as close as possible to their temporally corresponding local clips' representations. The global clip is sampled from the video instance with a skip rate equal to 4 times the local clips' skip rate. The global clip covers the entire temporal segment of all the local clips. It is worth noting that these local clips are the same clips used in the local–local loss. In the TCLR framework, there is one global clip and there are four local clips.

However, the TCLR local clips' sampling technique is not optimal for achieving temporal distinctness because of the following reasons: First, the spatiotemporal change and motion that are covered by these local clips are very short since these clips are densely sampled. Second, one action stage could be represented by these clips, and that would make the objective of the local–local loss is creating different representations for the same action stage. In addition, dense consecutive local clips will be very similar to one another even when the video has one action stage. That is because videos are temporally coherent in nature, which means that the signal in videos changes smoothly and slowly in time.

3. Sparse TCLR

Our proposed framework has three losses, two contrastive losses, and one pretext cross-entropy loss, denoted by $L_{Pretext}$. The contrastive losses are based on the TCLR framework, where one loss is an instance contrastive loss, denoted by L_{IC}, that enforces the representations of the clips sampled from the same video instance to be close to one another in the feature space and far away from representations of other videos. The second contrastive loss is the temporal sparse local–local contrastive loss, denoted by L_{SLL}, that is used to learn distinct representations for clips sampled from the same video instance. This loss uses a clip and its augmented version as positives, while all other local clips and their augmented versions (sampled from the same video at different random timestamps) are regarded as negatives. Eq. (1) shows the STCLR loss.

$$STCLR\ Loss = L_{Pretext} + L_{IC} + L_{SLL} \tag{1}$$

3.1. Instance contrastive loss

Our instance contrastive loss is based on the InfoNCE contrastive loss. This loss has an objective similar to the TCLR instance contrastive loss [1]. Two clips are sampled randomly from random timestamps of the same video using different sampling rates. The starting frame of each clip is selected at random such that these clips could be overlapped yet their speeds are different, so they cover different action extents. These clips are considered positives and contrasted with all other clips sampled from other videos (negatives) in the mini-batch. In addition, a positive pair could be created using each one of the two clips paired with its own augmented version. The mini-batch is sampled at random and contains different videos. Let the number of videos in the mini-batch be N_B, then the total number of clips in the mini-batch will be $2N_B$. Typical random spatial and geometric transformations are applied to these clips. Then, 3D-CNN is used as a video encoder to process the augmented clips and produce the clip's features. After that, a non-linear multi-layer perceptron projection head is used to project the features into the feature space. For each video i in the mini-batch, there will be two clip features (G_i, G'_i). Eq. (2) represents the instance contrastive loss definition.

$$L^i_{IC} = -\log \frac{h(G_i, G'_i)}{\sum_{j=1}^{N_B} \left[\mathbb{1}_{[j\neq i]} h(G_i, G_j) + h(G_i, G'_j) \right]} \tag{2}$$

where $h(u, v) = \exp(u^T v/(\|u\|\|v\|\Theta))$ is the similarity function that calculates the similarity between the two feature vectors, u and v, while Θ is the temperature parameter. In addition, $\mathbb{1}_{[j\neq i]} \in \{0, 1\}$ represents the indicator function that equals 1 if $j \neq i$.

3.2. Temporal contrastive loss

To encourage the network to learn the temporal distinctness and variation in each video instance, a sparse local–local loss is used. This loss is the same as the TCLR local–local loss, except it operates on five local clips sampled at random temporal locations using different sampling rates from the same video. Each clip represents a snippet from the video instance, and it covers different action extent from the other clips. These five clips could overlap with each other. For each clip, a positive clip is created by applying a random transformation to the clip. All other local clips and their augmented versions are considered negatives. Formally, let N_{s_clips} represent five sparse local clips sampled at random timestamps using different speeds from the video instance i. A positive pair at timestamp p is created using the anchor representation $G_{i,p}$ and its augmented clip representation $G'_{i,p}$. All the other sparse local clips and their augmented versions that are sampled from different timestamps from the same video represent the negatives. There will be $(2N_{s\_clips} - 2)$ negatives for each positive pair. The sparse local–local loss is defined in Eq. (3):

$$L^i_{SLL} = -\sum_{p=1}^{N_{s\_clips}} \log \frac{h(G_{i,p}, G'_{i,p})}{\sum_{q=1}^{N_{s\_clips}} \left[\mathbb{1}_{[q\neq p]} h(G_{i,p}, G_{i,q}) + h(G_{i,p}, G'_{i,q}) \right]} \tag{3}$$

It is worth noting that we do not use other temporal loss, such as the TCLR global–local loss, since our local clips are sparse and cover non-adjacent temporal segments that could be hundreds of frames apart. Fig. 3 shows the sparse local–local loss, where the representations of the clips are generated from the last layer of a 3D-CNN backbone (Layer 4 of R3D-18). Next, a 3D adaptive average pooling layer is applied to reduce the size of the representations, which are then projected by the MLP head and used by the contrastive loss.

3.3. The same speed localization pretext

The same speed localization (SSL) pretext mimics the human's ability to recognize the speed of multiple actions simultaneously. In a complex scene where there are multiple actions, humans know the normal speed of each action instantly, and they can tell if two or more actions have the same speed. For example, given three videos for three actions, such as playing soccer, biking, and applying eye makeup, humans know if there are two videos played at a faster speed than their normal speed at the same time. The input to the network is a tuple of K clips sampled from different videos with a probability of C_1. Then, there will be a $1 - C_1$ chance that all clips come from the same video instance. This probability controls the difficulty level of the pretext, since finding the clips that have the same speed is more challenging when the clips are sampled from different videos. There are S clips that have the same speed with a probability of C_2, while there is a chance of $C_3 = 1 - C_2$ that all the K clips have different speeds. When $K = 3$ and $S = 2$, this is a four-way classification problem that can be optimized using cross-entropy loss. The four free labels are [12, 13, 23, None], which represent the location of the clips that have the same speed in the tuple. In the case of None, it represents the situation where the clips have different speeds and no same speed clips are found. Fig. 4 shows the SSL pretext.

Given a tuple of three clips, 3D-CNN is used as a feature encoder to extract the feature from each clip in the tuple. The result is a tuple of features $F = \langle f_1, f_2, f_3 \rangle$. Then, the features are pairwise concatenated, $\langle f_{12}, f_{13}, f_{23} \rangle$, and transformed using a linear layer, followed by applying a ReLU function to form a tuple of three vectors. These vectors are concatenated again and passed to a fully connected layer with softmax to predict the target class. A cross-entropy loss is used to measure the prediction quality.

4. Experimental results

4.1. Datasets

Three common benchmark datasets are used in our experiments: UCF-101 [56], HMDB-51 [57], and Diving-48 (version 1) [58]. Diving-48 is a challenging dataset because the main distinguishing factor between different action classes is their long-range motion patterns rather than their static frame appearance. Therefore, the model must capture and differentiate these long-term motion patterns to correctly identify the classes [20]. Actions in Diving-48 can be defined by a combination of the takeoff (dive groups), movements performed while in the air (somersaults and/or twists), and the way the diver enters the water (dive positions). Two categories that are otherwise identical might have only subtle differences that are limited to one of the three stages. Diving-48 is valuable for assessing the model's ability to represent fine-grained details [1]. We follow the protocol used in [1], where the training set is used for self-supervised pre-training, and then the model is supervised fine-tuned on the training set, and tested on the test set.

We use the training set of split 1 of UCF-101 in our self-supervised pre-training. We do not use any action labels or annotations during the self-supervised pre-training. In action recognition downstream task evaluation, all three splits of UCF-101 and HMDB-51 datasets are used. In video retrieval downstream task evaluation, only split 1 of each dataset is used following the common testing protocol that uses the test set clips to query the training set clips [1,41,59]. Table 1 shows the information of each dataset.

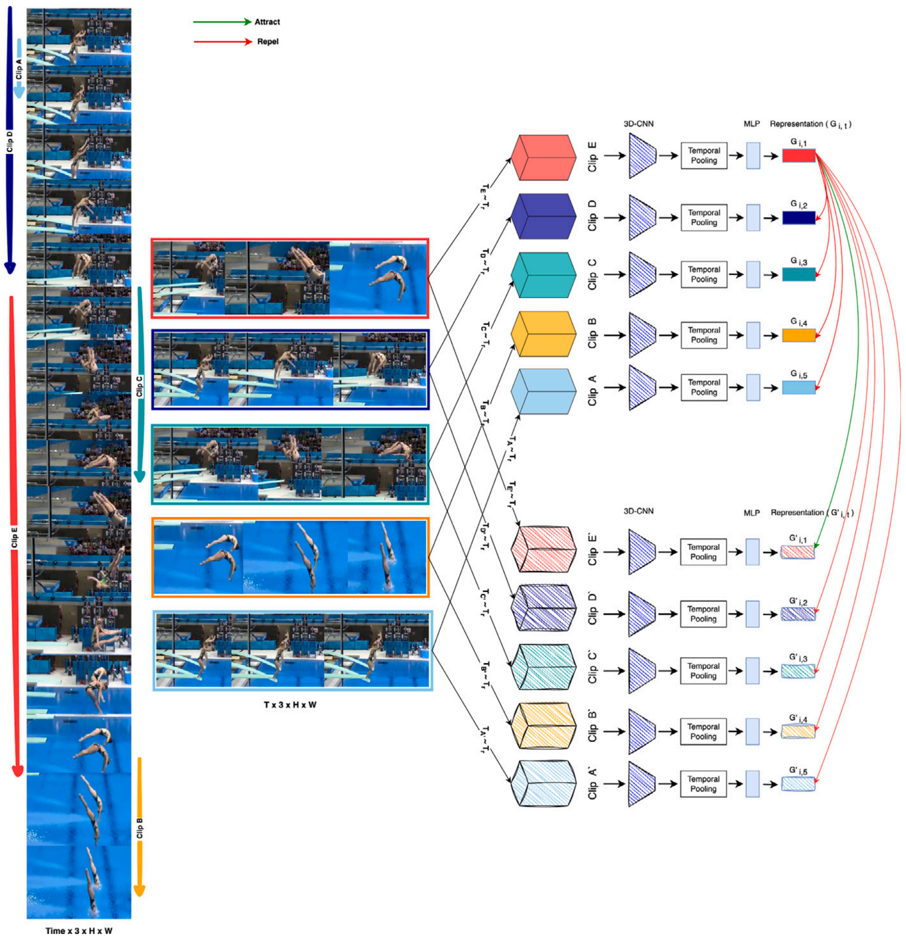


Fig. 3. The sparse local–local temporal contrastive loss, L_{SLL}. The video instance on the left represents diving which has three stages (takeoff, flight, entry). The red, dark blue, green, orange, and blue arrows represent five local clips sampled from the same video at random timestamps using different skip rates. Then for each clip, two versions are created using random transformations. The representations of those two clips form the positive pair for the loss. All other representations are considered negatives to the anchor clip. T_A... T_E are random transformations sampled from a universal transformation set T_r.

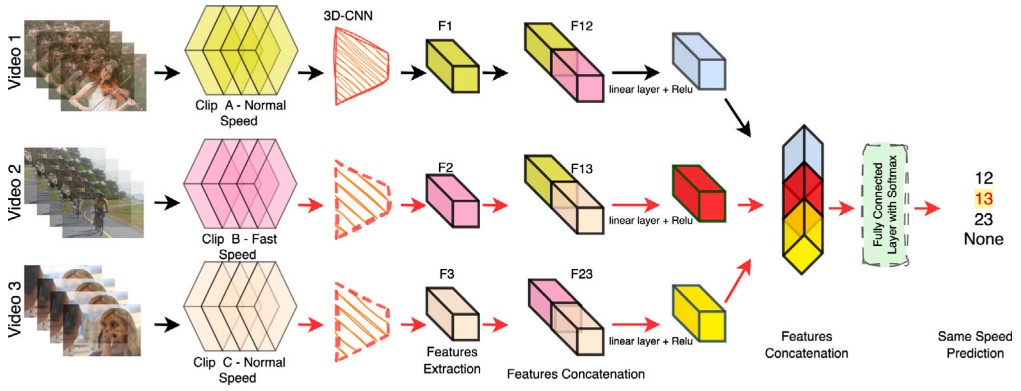


Fig. 4. The SSL pretext. An example of three clips sampled from three different videos. Clips A and C have the same speed. The correct label is "13" which is encoded as 1 in the implementation. The clip's features are extracted using a shared 3D-CNN backbone. The features are concatenated and passed to a linear layer with the ReLU function. Then, features are concatenated again and passed to a fully connected layer with softmax to produce the class probabilities. A cross-entropy loss is used in this SSL.

4.2. Ablation study

Three ablation experiments are conducted to validate our design choices. The first experiment tests the proposed pretext, SSL, only without using the contrastive objectives. The second experiment tests the Simplified TCLR framework in which only two losses are used, the instance contrastive loss and the proposed temporal sparse local–local contrastive loss. The third experiment evaluates the STCLR, which has

Table 1
Benchmark datasets used to evaluate our methods.

UCF-101 [56]	HMDB-51 [57]	Diving-48 (V1) [58]
13,320 videos	6849 videos	18K videos
101 classes	51 classes	48 fine-grained diving classes
YouTube videos	Mostly movies, YouTube, Prelinger, Google videos	Major diving competitions videos
3 splits, each has train and test set	3 splits, each has train and test set	Train set and test set
Human-object interaction, body-motion only, human-human interaction, playing musical instruments, sports	Facial actions w/o object manipulation, body movements w/o object interaction, body movements for human interaction	Various types of diving

three losses, using different levels of sparsity. Every experiment has two stages: the first is the self-supervised pre-training, while the second is the supervised downstream task evaluation.

4.2.1. *Same speed localization pretext*

This experiment confirms that our pretext is indeed capable of learning useful representations. The quality of the representations is evaluated on the action recognition downstream task by using the self-supervised, pre-trained model to initialize the weights of a new model backbone. A randomly initialized classification head is added to the backbone. Then, we fine-tuned all the layers of the new model to solve the supervised action recognition task.

Self-Supervised Pre-Training: In the SSL pretext self-supervised pre-training, a 3D-ResNet-18 model was used to solve the pretext. The training set of split 1 of UCF-101 was used to pre-train the network. The parameters that control this experiment were set to the following: $K = 3$, $S = 2$, $C_1 = 60\%$, $C_2 = 75\%$, and $C_3 = 25\%$, a max skip rate of 16 frames, and a dropout rate of 0.5. There are four possible class labels [12, 13, 23, None] that were encoded as [0, 1, 2, 3], respectively. Random augmentations were applied during the training that include random-sized crop, random horizontal flip, random Gaussian blur, random gray, and random color jitter. These are the default SSL pretext settings, any deviations from these settings will be specified throughout the paper.

Each clip has 16 frames with a resolution of 128×128. The model was trained for 499 epochs using the SGD optimizer, a learning rate of 1e–2, a weight decay of 1e–4, a momentum of 0.9, and a batch size of 24. One cycle scheduler was used during the training to change the learning rate.

Downstream Task Evaluation: We used action recognition to evaluate our proposed pretext by using the self-supervised, pre-trained model to initialize the weights of a new model backbone, 3D-ResNet-18. A randomly initialized two-layer classification head was added to the backbone. Then, we fine-tuned all the layers of the new model to solve the supervised action recognition task. We used all the UCF-101 three splits and reported the average Top1 video level classification accuracy. The video level prediction was calculated by averaging the predictions of ten uniformly spaced clips extracted from the video instance following prior works [1,59].

A clip with a resolution of $16{\times}128{\times}128$ was sampled with a skip rate equal to 4 and used as input to the model. The 3D-ResNet-18 model was trained for 519 epochs using the SGD optimizer, a learning rate of 3e–3, a weight decay of 2e–3, a momentum of 0.9, a batch size of 16, and a dropout rate of 0.9. One cycle scheduler was used during the training to change the learning rate. Standard data augmentations were used during training, such as color jittering, random horizontal flip, and random crop. In Table 2, we compare our action recognition accuracy with the speed-related pretexts.

Our SSL pretext outperforms all the other speed-related methods by a large margin except the pretext proposed by [20], which used a much larger batch size, 192 clips, during the self-supervised pre-training and used a much more computationally expensive procedure to compute the video level prediction. It is worth noting that our single objective pretext outperforms the other multi-objective pretexts that were pre-trained on much larger datasets, such as Kinetics [60].

Table 2
Same speed localization pretext performance on action recognition on UCF-101.

Method	Network	Input	UCF101	Pretraining
Multi-pretexts/Contrastive				
Time-Equivariant (Aux.) [20]	R3D-18	16×128	84.2	UCF-101
VTHCL/Contrastive [38]	R3D-50	8×224	82.1	Kinetics-400
Temp Trans [10]	R(2+1)D	16×112	81.6	UCF-101
Pace/Contrastive [21]	R(2+1)D	16×112	77.1	Kinetics-400
Pace/Contrastive [21]	R(2+1)D	16×112	75.9	UCF-101
Var. PSP [7]	R3D	16×112	69.0	UCF-101
ERUV [34]	R3D	16×112	68.8	Thumos14
PRP [8]	R3D	16×112	66.5	UCF-101
Single pretexts				
SpeedNet [6]	S3D-G	64×224	81.1	Kinetics
Our pretext	R3D-18	16×128	82.71	UCF-101

4.2.2. *The simplified TCLR*

The Simplified TCLR uses only two losses, the instance contrastive loss and the proposed sparse local–local temporal contrastive loss, without using the SSL pretext. This experiment proves that the Simplified TCLR achieves comparable performance or outperforms the original TCLR on action recognition and video retrieval using UCF-101 and HMDB-51 datasets. It is worth noting that our proposed approach has different input distribution since it uses different speeds and different sampling locations. In addition, our objectives are different from TCLR. We use two contrastive losses with/without pretext. Thus, the TCLR's settings and hyperparameters may not be the best fit for our models.

Self-Supervised Pre-Training: 3D-ResNet-18 model was used as the encoder to extract the features from the clips. The training set of split 1 of UCF-101 was used to pre-train the network. The input clips were 16 frames long, with a resolution of 112×112. Following [1], a multi-layer perceptron with one hidden layer was used as the projection head. Five clips were sampled from the video and used in the sparse local–local loss. These five clips were sampled from random locations using skip rates of 1, 4, 8, 16, and 24, respectively.

The same TCLR random augmentations were applied during the training. These augmentations include color jittering, random horizontal-flip, random cut-out, random grayscale, channel dropping, random cropping, and random scaling. The model was trained for 400 epochs using the Adam optimizer, a learning rate of 1e–3, a temperature of 0.3, a weight decay of 1e–9, and a batch size of 38. Reduce learning rate scheduler was used with patience of 12 epochs during the training to decrease the learning rate by a factor of 10 when the loss function stops decreasing. These are the default self-supervised pre-training settings, any deviations from these settings will be specified throughout the paper.

Downstream Task Evaluation: To evaluate the quality of the representations of the self-supervised Simplified TCLR model, action

Table 3
The video retrieval performance of the Simplified TCLR on **UCF-101** compared to the TCLR [1]. Numbers in red or green is $Our\ score - Three\ losses\ TCLR$.

Method	Losses	R@1	R@5	R@10	R@20
TCLR	$IC + LL$	51.1	67.83	74.57	80.89
TCLR	$IC + GL$	47.32	63.10	71.42	78.72
TCLR	$IC + LL + GL$	56.2	72.2	79.0	85.30
Simplified TCLR (10 clips)	$L_{SLL} + L_{IC}$	54.59 (−1.61)	73.43 (+1.23)	80.02 (+1.02)	85.62 (+0.32)
Simplified TCLR (9 clips)	$L_{SLL} + L_{IC}$	55.70 (−0.50)	73.49 (+1.29)	80.02 (+1.02)	85.96 (+0.66)

Table 4
The video retrieval performance of the Simplified TCLR on **HMDB-51** compared to the TCLR [1]. Numbers in red or green is $Our\ score - Three\ losses\ TCLR$.

Method	Losses	R@1	R@5	R@10	R@20
TCLR	$IC + LL$	19.07	42.42	54.97	69.35
TCLR	$IC + GL$	18.43	41.70	53.59	67.19
TCLR	$IC + LL + GL$	22.8	45.4	57.8	73.10
Simplified TCLR (10 clips)	$L_{SLL} + L_{IC}$	23.14 (+0.34)	46.27 (+0.87)	59.87 (+2.07)	73.99 (+0.89)
Simplified TCLR (9 clips)	$L_{SLL} + L_{IC}$	22.94 (+0.14)	47.12 (+1.72)	60.20 (+2.4)	74.58 (+1.48)

Table 5
The Simplified TCLR action recognition performance on UCF-101 and HMDB-51. **UCF-101 Split 1 training set** is used for the self-supervised pre-training.

Method	Architecture	Input	UCF	HMDB
TCLR [1]	R3D-18	16×112	83.90	53.50
Simplified TCLR (9 Clips)	R3D-18	16×112	83.32	55.04

recognition and video retrieval were used as the primary measures of the features' quality following [41]. Video retrieval measures the quality of the representations directly without any additional training using the nearest-neighbor retrieval task. The clips in the testing set were used to query the k-nearest neighbor clips from the training set. Correct video clip retrieval was scored when the top k nearest neighbors of a testing clip contained one video clip of the same class as the testing clip. Following the common practice, the video-level prediction was calculated by averaging the predictions of ten clips sampled uniformly from the video. In addition, we tested the model using nine uniformly sampled clips for calculating the video-level prediction. A full spatial crop was used. A skip rate of three frames was used to sample the clips. The clip features were obtained from the last layer of the model with spatial pooling applied following [1]. These are the default video retrieval downstream task evaluation settings, any deviations from these settings will be specified throughout the paper.

In action recognition, we followed a standard full fine-tuning procedure in which a randomly initialized classification head was connected to the self-supervised, pre-trained video encoder. Then, we trained all layers of the model using cross-entropy loss. We used a classification head that has two fully connected layers with a dropout layer in between. 3D-ResNet-18 network was used to solve the supervised action recognition task using both UCF-101 and HMDB-51 datasets. Specifically, we used the official three splits of each dataset and reported the average Top1 video-level classification accuracy. The video-level prediction was calculated by applying ten crops in the spatial dimension following [41]. The crops include one center crop and four corners with/without horizontal flipping for each video instance. For each crop, we sampled ten (nine in this experiment only) uniformly spaced clips extracted from the video instance and averaged their predictions. The video-level prediction was calculated by averaging the predicted probabilities for all the sampled crops. The input was a clip with a resolution of $16 \times 112 \times 112$ that was sampled at a speed equal to 4.

In the fine-tuning experiments, the model was trained for 460 epochs using the SGD optimizer, a learning rate equal to 3e–3, a weight decay of 6e–3, a momentum of 0.9, a batch size of 16, and a dropout rate of 0.9. One cycle scheduler was used during the training to change the learning rate. Standard data augmentations such as color jittering, random horizontal flip, and random crop were used during the training. The input clip was resized to a resolution of 128×171, and then a random crop of 112×112 was taken and used as input to the model. These are the default action recognition downstream task evaluation settings, any deviations from these settings will be specified throughout the paper. Tables 3 and 4 compare the video retrieval performance of the Simplified TCLR with the original TCLR, while Table 5 compares action recognition performance.

Tables 3 and 4 show that the Simplified TCLR achieves comparable or better performance than the TCLR while using only the instance contrastive loss and the sparse local–local temporal contrastive loss. The Simplified TCLR outperforms the TCLR in seven video retrieval measures out of eight when using ten clips to calculate the video-level prediction. It achieves lower accuracy only on the Top 1 retrieval measure on UCF-101. Interestingly, the Simplified TCLR achieves much better performance by outperforming the TCLR in seven video retrieval measures out of eight when using nine clips to calculate video-level prediction. Furthermore, the Top 1 retrieval performance gap on UCF-101 is decreased to only 0.5, which makes the Simplified TCLR score comparable to the TCLR score. In addition, our model with two losses outperforms the TCLR with two losses by a large margin, which proves that our sparse local–local loss with our sampling technique is more effective than the original TCLR when using two losses. Table 5 shows that the Simplified TCLR that uses nine clips to calculate video-level prediction achieves comparable action recognition accuracy to the TCLR on UCF-101, while it achieves better accuracy on HMDB-51. Our accuracy is lower than the TCLR by only 0.58% on UCF-101, while the Simplified TCLR outperforms the TCLR by 1.54% on HMDB-51. Our results on action recognition and video retrieval demonstrate that removing the TCLR's global–local loss and using only the local–local loss combined with random sparse sampling for temporal distinctiveness modeling performs better than the TCLR in most of the measures.

4.2.3. STCLR using various sparsity levels

This experiment shows the effect of using different sparsity levels to sample the local clips when pre-training STCLR model. Three different sparsity levels were used (low, medium, and high). We used skip rates of (1, 2, 3, 4, 5) for low sparsity, while we used (2, 4, 6, 8, 10) and (4, 8, 12, 16, 20) for medium and high sparsity, respectively.

We employed our default self-supervised pre-training settings mentioned in Section 4.2.2 to pre-train the STCLR models. We used an additional three clips with a resolution of $16 \times 112 \times 112$ to create the SSL input tuple. The same 3D-ResNet-18 encoder is used to extract the features from the tuple and to encode the local clips. We followed the default SSL pre-training settings mentioned in Section 4.2.1. The models were pre-trained for 300 epochs.

For action recognition, we followed the default fine-tuning settings mentioned in Section 4.2.2 with the following modifications. We report the classification accuracy for the first split only of UCF-101 and HMDB-51. In addition, the video-level prediction was calculated by applying a center crop and averaging the predictions of ten uniformly spaced clips extracted from the video instance, as in [59]. For video retrieval, we followed the same video retrieval evaluation procedure mentioned in Section 4.2.2. Figs. 5 and 6 present the STCLR video retrieval performance under different levels of sparsity on UCF-101 and HMDB-51, respectively. Fig. 7 shows the action recognition performance on UCF-101 and HMDB-51.

Fig. 5 shows that low sparsity STCLR achieved the best retrieval performance on UCF-101 compared to medium and high sparsity STCLR. The difference in performance is most obvious for the Top 1 measure, where increasing the sparsity reduces the Top 1 retrieval performance.

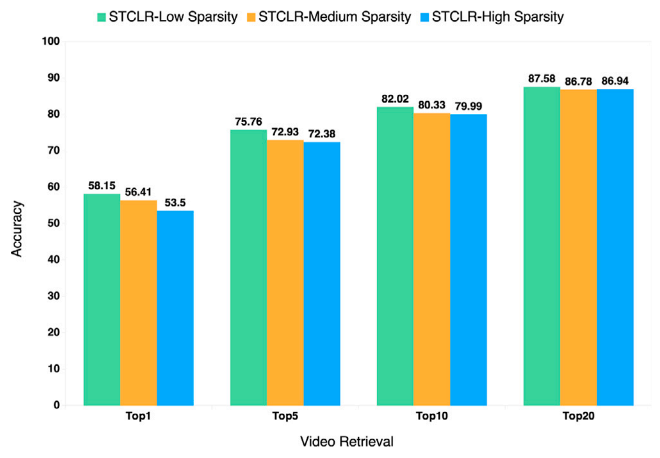


Fig. 5. Video retrieval on UCF-101 for STCLR with low, medium, and high sparsity.

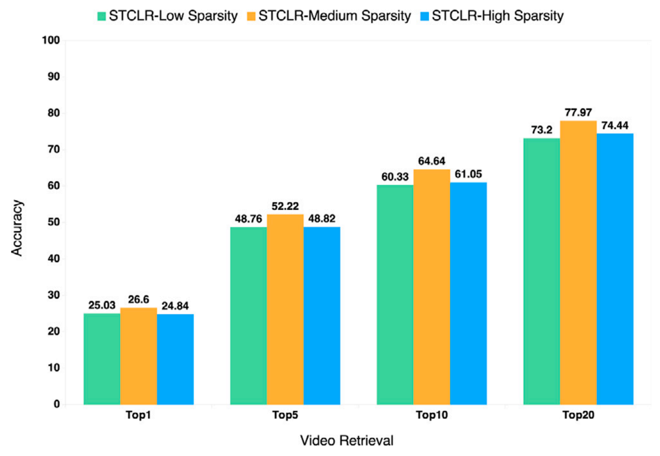


Fig. 6. Video retrieval on HMDB-51 for STCLR with low, medium, and high sparsity.

For Top 5, Top 10, and Top 20 measures, low sparsity continues to achieve the best performance, outperforming medium and high sparsity. In contrast, medium and high STCLR achieved comparable performance for these retrieval measures. This indicates that increasing the sparsity beyond low (medium and high) does not affect the Top 5, Top 10, and Top 20 retrieval measures.

Fig. 6 shows that medium sparsity STCLR achieved the best retrieval performance on all the measures on HMDB-51 compared to low and high sparsity STCLR. Using low and high sparsity reduces the performance for all the retrieval measures, as STCLR-Low achieved comparable performance to STCLR-High on Top 1, Top 5, Top 10, and Top 20.

Fig. 7 shows that changing the sparsity level of the STCLR does not affect the action recognition performance on UCF-101, since all three models achieved similar recognition accuracy. For HMDB-51, high sparsity STCLR achieved the best performance, yet the difference in performance is not very large; consequently, it can be concluded that changing the sparsity level does not affect the recognition accuracy.

Notably, although STCLR-Low and STCLR-Medium exhibit varying retrieval performance on UCF-101 and HMDB-51, both outperform TCLR in video retrieval across both datasets. Furthermore, our retrieval ablation experiments indicate that the optimal sparsity level for UCF-101 differs from that of HMDB-51, with low sparsity being most effective for UCF-101 and medium sparsity being optimal for HMDB-51. The disparate distributions of these datasets may account for the variation in optimal sparsity.

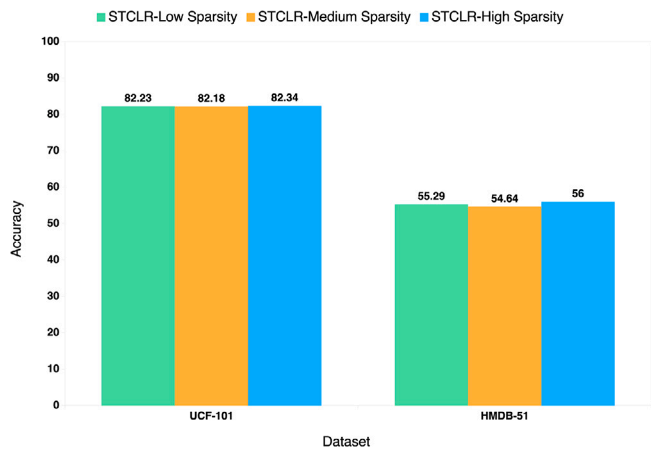


Fig. 7. Action recognition on UCF-101 and HMDB-51 for STCLR with low, medium, and high sparsity.

4.2.4. *The STCLR computational cost and efficiency*

Reading videos from the dataset is the most time-consuming operation during our self-supervised training. Hence, the quantity of videos that need to be processed in a single epoch is a key determinant of the training time for that epoch. Let N represent the total number of videos in the dataset. The TCLR requires reading N videos in each training epoch since it samples the global clip and the local clips from the same source video. In contrast, the STCLR approach samples the local clips from a source video and a tuple of K clips used in the SSL pretext from other source videos in each epoch. The K clips are sampled from different videos with a probability of C_1. Our STCLR processes total videos equal to $N + (1 - C_1)N + (K \times N \times C_1)$ at each epoch. The STCLR requires reading $2N$ videos in each training epoch when all the K clips in a tuple come from the same source video, C_1 equals zero. In this scenario, the STCLR performs two complete reading operations of the dataset within a single epoch. One reading operation is dedicated to local clips' sampling, while the other reading operation is for SSL tuple sampling. The worst case is when all the K clips in the tuple come from different source videos, C_1 equals one. In this scenario, the STCLR performs $N + (K \times N)$ complete reading operations of the dataset within a single epoch.

We provide a training time and space complexity comparison between the original TCLR framework, the Simplified TCLR, and our proposed STCLR framework. All the experiments were executed on a server that has an AMD EPYC 7282 16-Core Processor and NVIDIA A40 GPU. The NVIDIA A40 has 48 GB of memory, while the server has 62 GB of main memory. We report the training time for one epoch on the full training set of UCF-101 split 1, which has 9538 videos.

Consistent training conditions were used when executing all the experiments. We followed the self-supervised training settings mentioned in Section 4.2.2, with the following exceptions. We used a temperature of 0.1 and patience of 9 epochs for TCLR training and followed the optimization objectives of the original TCLR [1], optimizing three contrastive losses. In the STCLR experiments, we used skip rates of 1, 4, 8, 16, and 32, respectively. We used the default SSL pre-training settings and random augmentations as mentioned in Section 4.2.1. Three values of C_1 were tested, which include 0, 1, and 0.6. Table 6 shows the training time and space complexity for each model.

The TCLR and the Simplified TCLR have almost the same epoch training time and GPU memory consumption. This is expected since both methods process the same number of clips (10 clips) for each one video instance. The STCLR training time is longer than the TCLR training time due to the additional overhead of the SSL pretext training, which is controlled by the value of C_1. The STCLR training time increases as the C_1 increases. In addition, the STCLR requires slightly

Table 6
The training time and space complexity for TCLR, the Simplified TCLR and the STCLR on the training set of UCF-101 split 1. Epoch time and GPU memory are reported using minutes and gigabytes, respectively.

Method	C_1	Epoch time (Min.)	GPU memory
TCLR [1]	–	11.33	46.69 GB
Simplified TCLR	–	11.06	46.46 GB
STCLR	0	22.03	47.22 GB
STCLR	0.6	34.36	47.22 GB
STCLR	1	43.32	47.22 GB

Table 7
The computational complexity (MegaFLOPs). TCLR uses $Bs = 40$, $d = 128$, $n = 4$, and ten clips per instance, one global and four local clips with their augmented versions. Simplified TCLR, and STCLR use $Bs = 38$, $d = 128$, $n = 5$. Simplified TCLR uses ten clips per instance, while STCLR uses thirteen. $X = 258.84$ GFLOPs.

Method	Input	IC	GL	LL	SSL	Total
TCLR [1]	$400X$	1.663	0.169	0.664	–	400 X + 2.498
Simplified TCLR	$380X$	1.501	–	0.986	–	380 X + 2.488
STCLR	$494X$	1.501	–	0.986	0.0006	494 X + 2.489

more GPU memory, approximately 0.53 GB extra memory, compared to the TCLR.

To compare the computational complexity of our approaches with the TCLR, we use the total FLOPs required to process one batch as a complexity measure. Our approaches and the TCLR use the same input dimensions (16 × 112 × 112) and the same network architecture for the backbone encoder (FLOPs of the SSL classification head are negligible). In addition, the losses do not introduce additional computations to the gradient computations in the shared backbone layers during the backward pass because the losses operate only at the final layer. The gradients of the losses are summed up and back-propagated to the shared layers. Because of these reasons, the total number of computations for the forward pass, backward pass, and applying the optimization step for one input clip is equal for all approaches. Let X represent the total FLOPs required for encoding one input clip in the forward pass, computing the gradients in the backward pass, and applying the optimization step. $X = 258.84$ GFLOPs, which is estimated using PyTorch profiler tools. Let Bs, d, and n represent the batch size, the dimension of the features, and the number of local clips, respectively.

The FLOPs required to calculate the instance contrastive loss (IC) for one batch equals to $(8Bs^2 \times d + 16Bs^2 - 6Bs)$. We consider one arithmetic operation (addition, subtraction, multiplication, division, logarithm, and exponentiation) to cost one FLOPs. The FLOPs required to calculate the local–local (LL) loss for one batch equals to $Bs \times (8n^2 \times d + 16n^2 - 6n)$, while the global–local (GL) loss requires $(4248Bs)$, and the SSL cross-entropy loss costs $(17Bs)$. The total FLOPs for one batch equals $((Bs \times \text{input clips per one instance}) \times X + FLOPs\,for\,Losses)$. The IC loss and the local–local loss have quadratic time complexity, $O((\text{Loss Input})^2)$. However, the most dominant computational expensive operations are represented by X, which is much larger than the losses cost. Table 7 shows the computational complexity (total FLOPs required to process one batch) of the TCLR, Simplified TCLR, and STCLR.

Table 7 shows that our methods require less FLOPs for IC loss calculation than TCLR, which is because we used smaller batch size than TCLR. In contrast, our local–local loss requires more FLOPs than TCLR, which is because we used five local clips while TCLR used four. The total costs for losses calculations are nearly identical for all methods. However, the most influential factor is the cost for forward and backward passes which is controlled by the total number of clips the model needs to process for each batch. The Simplified TCLR requires $20X$ less FLOPs than the TCLR, while the STCLR requires $94X$ more FLOPs. The Simplified TCLR achieved better accuracy while saving 5176.8 GFLOPs. It is worth noting that while both the Simplified TCLR and the TCLR have the same computational cost (FLOPs) when both use the same batch size, the Simplified TCLR reduces the optimization objectives, resulting in a streamlined, simpler optimization landscape. All the methods have a linear time complexity with respect to the batch size, $O((Bs \times \text{input clips per one instance}) \times X)$. When using three clips per tuple for the SSL pretext, the STCLR has $O(Bs \times 13 \times X)$ time complexity, while the TCLR has $O(Bs \times 10 \times X)$ time complexity.

4.3. Sparse TCLR

This section presents detailed information regarding the STCLR pre-training, action recognition, fine-grained action recognition, and video retrieval experiments.

4.3.1. Sparse TCLR self-supervised pre-training

The proposed Sparse TCLR framework uses three losses that include the instance contrastive loss, the proposed sparse local–local temporal contrastive loss, and the same speed localization pretext loss. We used our default self-supervised pre-training settings mentioned in Section 4.2.2 to pre-train the STCLR model. Skip rates of 1, 4, 8, 16, and 32 were used to sample the five sparse local clips. An additional three clips with a resolution of 16 × 112 × 112 were sampled to form a tuple, which was used as input to the SSL classification head. The same 3D-ResNet-18 backbone was used to extract the features from the tuple. We followed our default SSL settings mentioned in Section 4.2.1.

4.3.2. Action recognition

Action recognition is a common downstream task for assessing the quality of the self-supervised, pre-trained learned features. We followed our default fine-tuning settings mentioned in Section 4.2.2 with the following modifications. For split 3 of UCF-101, the model was trained for 490 epochs. For splits 1 and 3 of HMDB-51, we used a learning rate of 1e–2, a weight decay of 3e–3, and a dropout rate of 0.94. Reduce on plateau scheduler was used to change the learning rate during the training. The patience was set to six epochs. For split 2 of HMDB-51, we used a learning rate of 3e–3, an initial learning rate of 3e–3, a weight decay of 3e–3, and a dropout rate of 0.90. One cycle scheduler was used during the training to change the learning rate. The models were trained for 460 epochs for splits 1 and 3, while we used 390 epochs for split 2. In Table 8, we compare our scores with the original TCLR and with other self-supervised methods that use the same or comparable experimental settings and modality as ours.

Since our main objective is enhancing the TCLR, we compare our model with the TCLR. Our model outperforms the TCLR on HMDB-51 by 0.83%, while it achieves a lower accuracy by 0.72% on UCF-101. In addition, the lower part of Table 8 presents methods that cannot be compared fairly with our model. That is because these self-supervised methods [20,44], and [43] used much larger batch sizes, different resolutions, or different architectures during the self-supervised pre-training. It can be seen that the batch size during the self-supervised pre-training plays a vital role in controlling the quality of the representations. Such a role is obvious when we look at the model of ProViCo-2 [43], which is an R(2 +1)D model pre-trained with a batch size of 48 clips. This model achieves 82.1% accuracy on UCF-101. When the batch size increased from 48 to 96 in training the ProViCo-3 model, the accuracy increased by 4%. In addition, ProViCo-4 shows the effect of increasing the batch size, the depth of the model, and the input resolution. ProViCo-4, when compared to ProViCo-1, achieves a 10.9% increment on UCF-101 and an 11.1% on HMDB-51 by increasing the batch size, the model's depth, and the resolution of the input.

To understand the reason for the lower accuracy of the STCLR on UCF-101, we investigated the effect of regularization, which can lead to a performance gap between the two initialization methods. We conducted two sets of experiments that followed the same aforementioned UCF-101 action recognition fine-tuning procedure with 490 epochs. In these experiments, we used only the first split of UCF-101 for training and testing. Furthermore, the video-level prediction was

Table 8
STCLR action recognition performance on UCF-101 and HMDB-51. **UCF-101 Split 1 training set** is used for the self-supervised pre-training.

Method	Arch.	Input	UCF	HMDB
STS [61]	R3D-18	16×112	67.2	32.7
DPC [17]	R3D-18	40×128	60.6	–
VCOP [59]	R3D-18	16×112	64.9	29.5
Pace Pred [21]	R3D-18	16×112	65.0	–
VCP [62]	R3D-18	16×112	66.0	31.5
PRP [8]	R3D-18	16×112	66.5	29.7
Var. PSP [7]	R3D-18	16×112	69.0	33.7
MemDPC [18]	R3D-18	40×224	69.2	–
TCP [63]	R3D-18	$- \times 224$	64.8	34.7
CSJ [64]	R3D-18	16×224	70.4	36.0
BFP [65]	R3D-18	40×128	63.6	–
IIC (RGB) [46]	R3D-18	16×112	61.6	–
CVRL [35]	R3D-18	16×112	75.77	44.6
Temp Trans [10]	R3D-18	16×112	77.3	47.5
TCLR [1]	R3D-18	16×112	83.9	53.5
Methods use larger batch size, different resolution and/or architectures				
V3S [66]	R(2+1)D	16×112	79.1	38.7
Vi2CLR [44]	S3D	32×128	82.8	52.9
Time-Equivariant [20] (Aux/batch 192)	R3D-18	16×128	84.2	59.6
Time-Equivariant [20] (CL+Aux/batch 192)	R3D-18	16×128	83.7	60.8
ProViCo-1 (batch 96) [43]	R3D-18	16×112	83.7	57.1
ProViCo-2 (batch 48) [43]	R(2+1)D	16×112	82.1	–
ProViCo-3 (batch 96) [43]	R(2+1)D	16×112	86.1	58.0
ProViCo-4 (batch 512) [43]	R3D-50	16×224	94.6	68.2
Sparse TCLR (Ours)	**R3D-18**	$\mathbf{16 \times 112}$	**83.18**	**54.33**

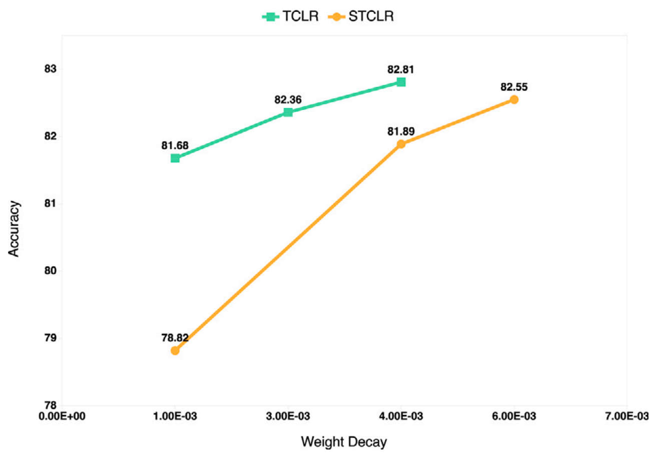


Fig. 8. Regularization effect on UCF-101 action recognition for the TCLR and STCLR.

calculated by applying a center crop and averaging the predictions of ten uniformly spaced clips extracted from the video instance, as in [59]. In addition, we used different values of weight decay in each set of experiments. One set of experiments used a model downloaded from the official TCLR GitHub repository as an initialization, while the other set of experiments used our STCLR model as an initialization. Both the TCLR model and the STCLR model were pre-trained on the training set of UCF-101 split 1. Fig. 8 shows the impact of regularization on fine-tuning action recognition models initialized by the TCLR or the STCLR.

Fig. 8 shows that the TCLR achieves comparable or better accuracy than the STCLR, while using less regularization. For instance, the TCLR model that used a weight decay of 1e–3 achieves comparable accuracy to the STCLR model that used a weight decay of 4e–3. This shows that the TCLR uses 0.25% of the regularization used by the STCLR. In addition, the TCLR model that used a weight decay of 3e–3 achieves comparable accuracy to the STCLR model that used a weight decay of 6e–3. This indicates that the TCLR used 0.5% of the regularization used by the STCLR. This shows that the STCLR initialization slightly makes the model overfit the dataset, since the STCLR needs more regularization than the TCLR on UCF-101. This behavior is understandable since different initialization methods can have different impacts on how quickly the model learns, how well it generalizes, and its ability to converge to a good solution [67].

In addition, our proposed method, Sparse TCLR, is a general technique that can be applied and used in combination with any instance contrastive model to enforce the representations to be distinct along the temporal dimension explicitly. For example, a potential approach involves replacing our instance contrastive loss, L_{IC}, with the stochastic contrastive loss proposed by [43], in which positive and negative pairs are formed according to the probabilistic distance. This method aims to mine more semantically related positive and negative pairs by representing the individual clips sampled from a video as normal distributions and modeling the whole video distribution by combining them into a Mixture of Gaussians for effective contrastive learning. In addition, our instance contrastive loss can be replaced by the framework proposed by [44], where clustering is used to create positive and negative samples. For each example in a specific cluster, positive pairs are mined randomly from the instances that belong to the same cluster. Negative samples are extracted from different clusters within the mini-batch during the training process, which alternates between learning representation and clustering. An alternative approach that could be used to replace our instance contrastive loss is a co-training technique used to enhance the infoNCE loss [41]. When using this technique, more semantically similar positive and negative pairs can be mined using the supplementary information obtained from various views of the data. These views could include RGB and optical flow views, where one view is used to mine positive class instances for the other view. Fig. 9 shows the attention maps of the top three predictions for two actions: haircut and ice dancing, using our action recognition model, which is fine-tuned on UCF-101.

4.3.3. *Fine-grained action recognition*

Diving-48 is used to evaluate the STCLR on modeling long-term temporal dynamics and fine-grained action classification. The STCLR model was self-supervised pre-trained for 100 epochs on the training set of Diving-48 using the self-supervised pre-training settings mentioned in Section 4.3.1. The model was evaluated on the test set of Diving-48

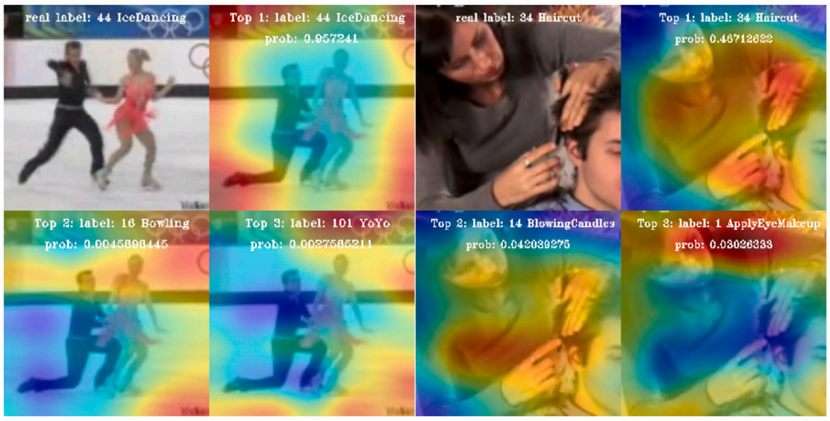


Fig. 9. The attention maps of the top three predictions for haircut and ice dancing using our action recognition classifier, fine-tuned on UCF-101. Our classifier focuses on the areas and the objects related to each one of the actions.

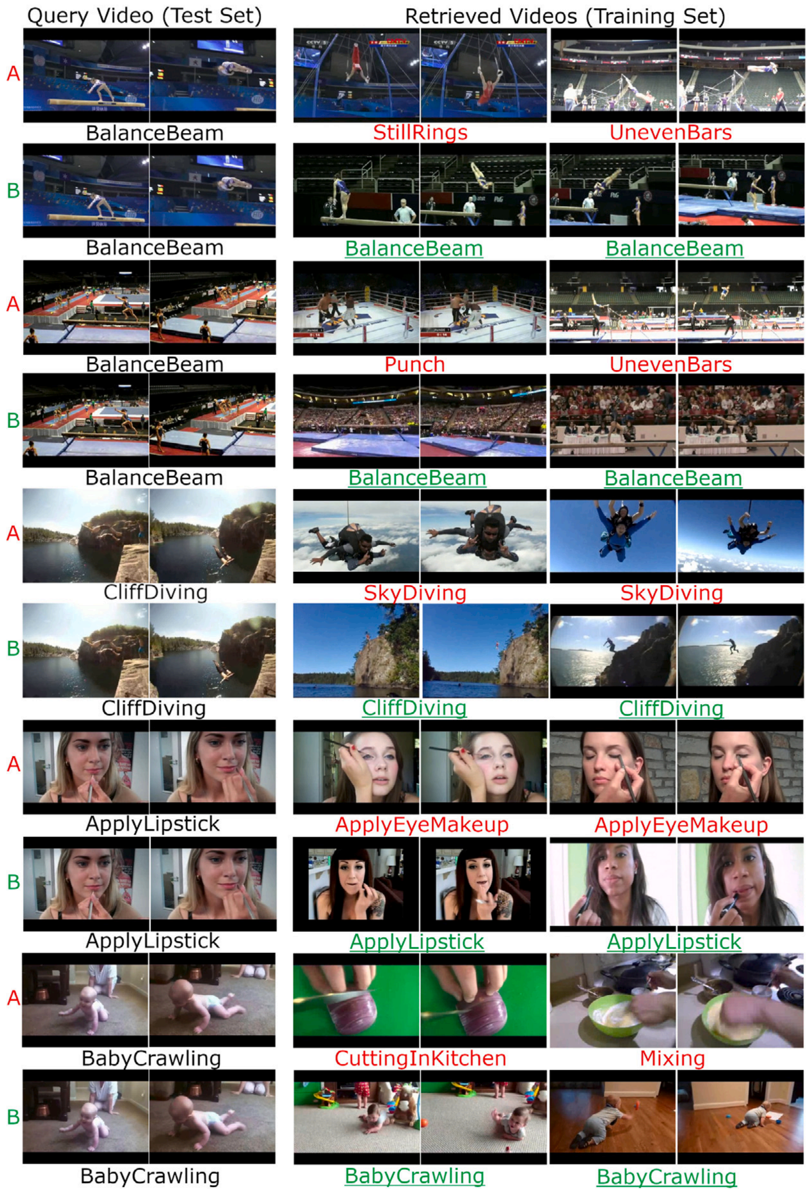


Fig. 10. Video retrieval qualitative analysis. Each clip is represented by two frames. For each testing video, the two nearest neighbors are retrieved from the training set of UCF-101. Row A displays the retrieved videos of the official TCLR model, while row B displays the retrieved videos of our model. Both rows, A and B, use the same query video.

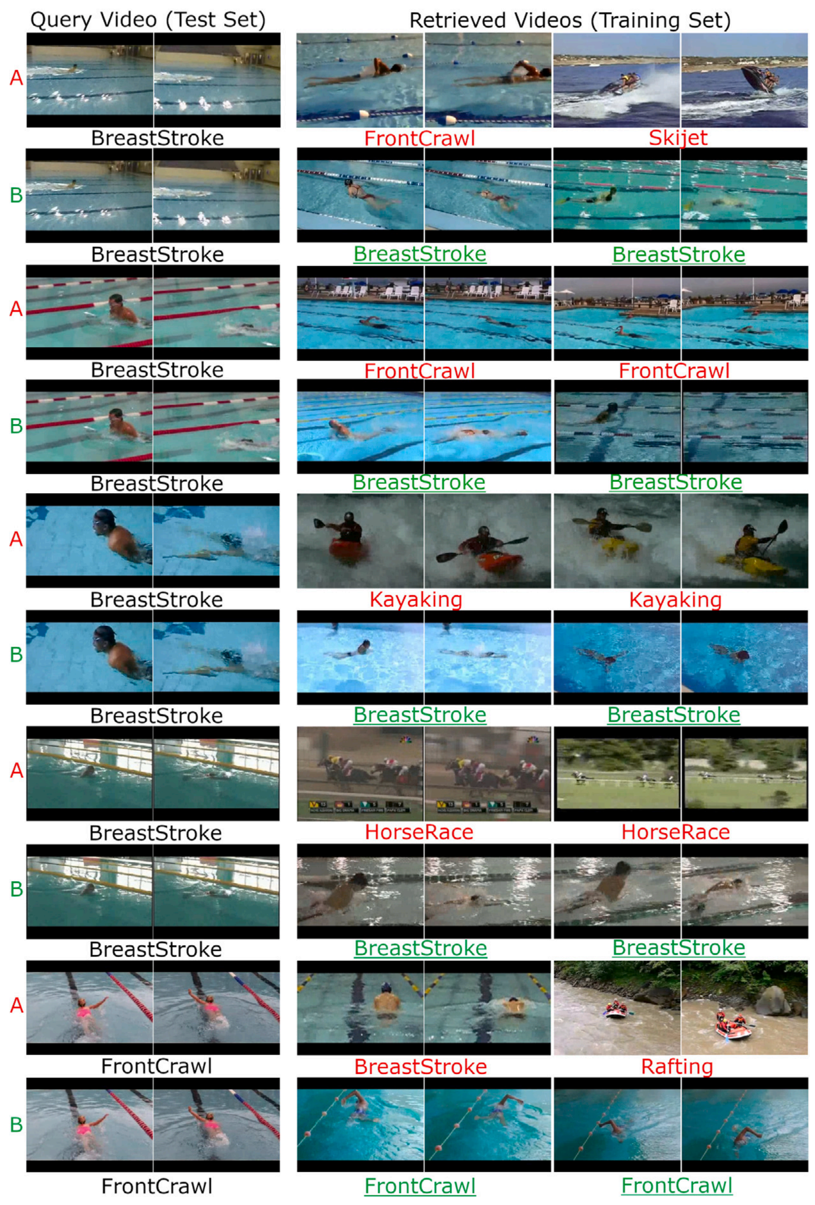


Fig. 11. Video retrieval of fine grained actions. Each clip is represented by two frames. For each testing video, the two nearest neighbors are retrieved from the training set of UCF-101. Row A displays the retrieved videos of the official TCLR model, while row B displays the retrieved videos of our model. Both rows, A and B, use the same query video.

Table 9
STCLR action recognition performance on Diving-48 (version 1). **Diving-48 training set** is used for the self-supervised pre-training.

Pre-training method	Arch.	Accuracy
None (Random initialization) [1]	R3D-18	13.4
VCOP [1]	–	14.7
Instance Contrastive [1]	R3D-18	15.8
RESOUND [58]	C3D	16.4
TSN [68]	BN-Inception	16.8
CVRL [1]	–	17.6
MiniKinetics Supervised [68]	R3D-18	18.0
MiniKinetics Supervised (Debiased) [68]	R3D-18	20.5
TCLR [1]	R3D-18	22.9
Time-Equivariant [20]	R(2+1)D	34.9
STCLR	**R3D-18**	**32.58**

using our default fine-tuning settings mentioned in Section 4.2.2. In Table 9, we compare the STCLR performance with the TCLR and other self-supervised methods.

The STCLR achieves an outstanding performance in modeling the long-term temporal dynamics of the fine-grained actions in Diving-48. It is evident that the STCLR can capture well the essential temporal aspects of video representation, such as gradual transitions and long-term dependencies. The STCLR outperforms the TCLR by 9.68% classification accuracy, which is an outstanding improvement. In addition, it achieves an accuracy gain of 19.18% over random initialization, demonstrating the effectiveness of using sparse local clips to achieve temporal distinctiveness of the features. The STCLR accuracy is lower than [20] by only 2.32%, even though [20] used a stronger architecture and a larger batch size for pre-training, which are known to affect contrastive learning performance.

4.3.4. *Video retrieval*

The video retrieval task is used as the main measure of the features' quality since it is calculated directly using the self-supervised, pre-trained backbone as a feature extractor without any further training. We followed the same video retrieval evaluation procedure mentioned in Section 4.2.2 to measure the Sparse TCLR framework performance.

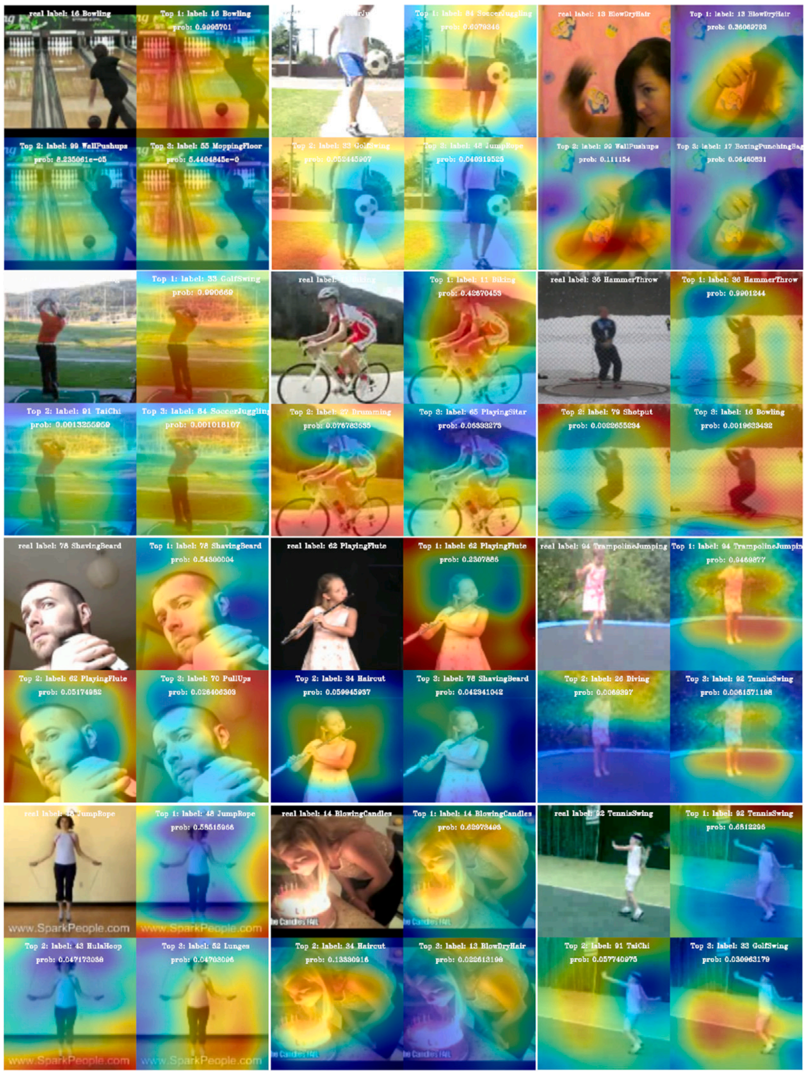

real label: 16 Bowling
Top 1: label: 16 Bowling prob: 0.9995701
Top 1: label: 84 SoccerJuggling prob: 0.6079346
real label: 13 BlowDryHair
Top 1: label: 13 BlowDryHair prob: 0.36069793
Top 2: label: 99 WallPushups prob: 8.235061e-05
Top 3: label: 55 MoppingFloor prob: 5.4404845e-0
Top 2: label: 33 GolfSwing prob: 0.052445907
Top 3: label: 48 JumpRope prob: 0.040319525
Top 2: label: 99 WallPushups prob: 0.111154
Top 3: label: 17 BoxingPunchingBag prob: 0.06460631
Top 1: label: 33 GolfSwing prob: 0.990669
real label: 11 Biking
Top 1: label: 11 Biking prob: 0.42670453
real label: 36 HammerThrow
Top 1: label: 36 HammerThrow prob: 0.9901244
Top 2: label: 91 TaiChi prob: 0.0013255959
Top 3: label: 84 SoccerJuggling prob: 0.001018107
Top 2: label: 27 Drumming prob: 0.076783635
Top 3: label: 65 PlayingSitar prob: 0.06333273
Top 2: label: 79 Shotput prob: 0.0022655234
Top 3: label: 16 Bowling prob: 0.0019633492
real label: 78 ShavingBeard
Top 1: label: 78 ShavingBeard prob: 0.54300004
real label: 62 PlayingFlute
Top 1: label: 62 PlayingFlute prob: 0.2307886
real label: 94 TrampolineJumping
Top 1: label: 94 TrampolineJumpin prob: 0.9469877
Top 2: label: 62 PlayingFlute prob: 0.05174952
Top 3: label: 70 PullUps prob: 0.026406303
Top 2: label: 34 Haircut prob: 0.059945937
Top 3: label: 78 ShavingBeard prob: 0.042341042
Top 2: label: 26 Diving prob: 0.0069397
Top 3: label: 92 TennisSwing prob: 0.0061571198
real label: 48 JumpRope
Top 1: label: 48 JumpRope prob: 0.58515966
real label: 14 BlowingCandles
Top 1: label: 14 BlowingCandles prob: 0.62973493
real label: 92 TennisSwing
Top 1: label: 92 TennisSwing prob: 0.6812296
Top 2: label: 43 HulaHoop prob: 0.047173038
Top 3: label: 52 Lunges prob: 0.04703096
Top 2: label: 34 Haircut prob: 0.13330916
Top 3: label: 13 BlowDryHair prob: 0.022613198
Top 2: label: 91 TaiChi prob: 0.057740975
Top 3: label: 33 GolfSwing prob: 0.030963179
www.SparkPeople.com

Fig. 12. The attention maps of the top three predictions for twelve actions using our action recognition classifier, fine-tuned on UCF-101.

In Tables 10 and 11, we compare our retrieval scores with the original TCLR and other self-supervised methods.

The STCLR outperforms the TCLR on both UCF-101 and HMDB-51. Specifically, our method outperforms the TCLR on UCF-101 by 0.58%, 1.21%, 0.88%, and 0.98% on Top 1, Top 5, Top 10, and Top 20 retrieval measures, respectively. Furthermore, the STCLR achieves outstanding performance by a large margin on HMDB-51 when it outperforms the TCLR by 4.32%, 5.97%, 6.51%, and 2.78% on Top 1, Top 5, Top 10, and Top 20 retrieval measures, respectively. Our Sparse TCLR produces much better transferable features than the TCLR. In contrast, other self-supervised methods that use larger batch sizes can achieve better performance than our method, in the lower part of Tables 10 and 11. However, our STCLR framework is general and can be combined with these methods, as explained in Section 4.3.2.

5. Qualitative analysis

Figs. 10 and 11 show a qualitative analysis of our proposed approach. We compare the video retrieval performance of the STCLR model with the performance of the official TCLR model, which is downloaded from the official TCLR GitHub repository. Both models are pre-trained on UCF-101 dataset. In the figures, each clip is represented by two video frames. For each video in the testing set, the two nearest neighbors videos are retrieved from the training set of UCF-101. Row

Table 10
STCLR video retrieval performance on **UCF-101**. Numbers in red or green is *Our score−Three losses TCLR*.

Method	R@1	R@5	R@10	R@20
R3D-18 backbone				
VCOP [59]	14.1	30.3	40.4	51.1
VCP [62]	18.6	33.6	42.5	53.5
Pace Pred [21]	23.8	38.1	46.4	56.6
Var. PSP [7]	24.6	41.9	51.3	62.7
Temp Trans [10]	26.1	48.5	59.1	69.6
MemDPC [18]	20.2	40.4	52.4	64.7
V3S [66]	28.3	43.7	51.3	60.1
RSPNet [69]	41.1	59.4	68.4	77.8
MFO [70]	39.6	57.6	69.2	78.0
TCLR [1]	56.2	72.2	79.0	85.3
Methods use larger batch size, resolution, or architectures				
Time-Equivariant [20] (Aux - batch 192)	63.6	79.0	84.8	89.9
ProViCo (batch 96) [43]	63.8	75.1	84.8	89.2
Sparse TCLR	56.78 (+0.58)	73.41 (+1.21)	79.88 (+0.88)	86.28 (+0.98)

Table 11
STCLR video retrieval performance on **HMDB-51**. Numbers in red or green is *Our score−Three losses TCLR*.

Method	R@1	R@5	R@10	R@20
R3D-18 backbone				
VCOP [59]	7.6	22.9	34.4	48.8
VCP [62]	7.6	24.4	36.6	53.6
Pace Pred [21]	9.6	26.9	41.1	56.1
Var. PSP [7]	10.3	26.6	38.8	51.6
Temp Trans [10]	–	–	–	–
MemDPC [18]	7.7	25.7	40.6	57.7
V3S [66]	10.8	30.6	42.3	56.2
RSPNet [69]	–	–	–	–
MFO [70]	18.8	39.2	51.0	63.7
TCLR [1]	22.8	45.4	57.8	73.1
Methods use larger batch size, resolution, or architectures				
Time-Equivariant [20] (Aux - batch 192)	32.2	60.3	71.6	81.5
ProViCo (batch 96) [43]	35.9	55.2	74.3	81.8
Sparse TCLR	27.12 (+4.32)	51.37 (+5.97)	64.31 (+6.51)	75.88 (+2.78)

A shows the retrieved videos of the official TCLR model, while row B shows the retrieved videos of the STCLR model. Both rows, A and B, use the same query video, as depicted in the selected representative frames. Fig. 10 shows that our model outperforms the TCLR model on actions that have multiple phases, such as BalanceBeam. Our model retrieved two training videos that have the correct label, BalanceBeam, while the TCLR model incorrectly retrieved videos from other similar actions, such as StillRings and UnevenBars. In addition, the TCLR model incorrectly retrieved SkyDiving, ApplyEyeMakeup, and CuttingInKitchen videos for CliffDiving, ApplyLipstick, and BabyCrawling query videos, respectively. Fig. 11 shows the retrieval of challenging actions (spatially similar actions), where distinct classes are mainly characterized and defined by fine-grained, varying motion patterns. Our model retrieved correct videos, while the TCLR model incorrectly retrieved BreastStroke instead of FrontCrawl and vice versa. This proves that our model is better at encoding the subtle motion dynamics, which are used to differentiate between spatially similar actions. In addition, Fig. 12 shows the attention maps of the top three predictions for twelve actions using our action recognition classifier, which is fine-tuned on UCF-101. Clearly, our model focuses on areas and objects related to each action.

6. Limitations

Designing a multitask contrastive framework by combining diverse pretexts with a contrastive objective is a powerful self-supervised technique. Examples of such frameworks include [20,71]. This design often produces high-quality learned representations. However, only one novel pretext, same speed localization, is combined and optimized with our contrastive objectives. Adding more diverse pretext tasks could enhance the performance further. In addition, we used Diving-48, UCF-101 and HMDB-51 datasets that contain trimmed videos, where each video focuses on an action. Exploring untrimmed videos could be a thought-provoking endeavor. Similarly, exploring significantly larger datasets like Kinetics, Moments in Time, or Something-Something is intriguing, but it incurs substantial computational costs. In addition, we used the RGB modality of videos only for self-supervised representation learning. It is worth trying to explore using other complementary signals, such as audio or optical flow, to discover the impact of these modalities on the temporal distinctiveness of the features. Lastly, because our local clips are chosen at random, there is no assurance that the sampled local clips contain high-motion information, which is essential for learning actions. The quality of the representations could be improved by sampling clips with a strong motion signal.

7. Conclusions

We proposed the STCLR framework, which is a self-supervised contrastive framework that uses a sparse local–local temporal contrastive loss to enforce the distinctiveness along the temporal dimension of the representations explicitly. Unlike the TCLR local–local loss, our temporal loss relies on contrasting different views of the video. These views are represented by five local clips sampled from the video at random locations and different speeds. These clips cover different motion dynamics of the video. We showed that the temporal distinctiveness of the representations can be achieved using only two losses instead of the three TCLR losses. In addition, a novel temporal pretext (SSL) is proposed that encourages the model to learn inter-class speed dynamics by comparing clips that represent different actions. Our SSL pretext outperforms the other speed-related pretexts on action recognition on UCF-101. By combining the SSL pretext with the sparse local–local temporal loss, a much better transferability of features is achieved on the HMDB-51 video retrieval task. The STCLR outperforms the TCLR on both UCF-101 and HMDB-51 datasets. More precisely, the STCLR outperforms the TCLR by 0.58%, 1.21%, 0.88%, and 0.98% and by 4.32%, 5.97%, 6.51%, and 2.78% on Top 1, Top 5, Top 10, and Top 20 retrieval measures on UCF-101 and HMDB-51, respectively. The STCLR demonstrated exceptional performance on HMDB-51 dataset, surpassing the TCLR by a significant margin. On action recognition, the STCLR outperforms the TCLR on Diving-48 by 9.68%, which is a significant accuracy gain, and on HMDB-51 by 0.83%, while it achieves a lower accuracy by 0.72% on UCF-101.

CRediT authorship contribution statement

Hussein Altabrawee: Writing – original draft, Methodology, Conceptualization. **Mohd Halim Mohd Noor:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been supported in part by the Ministry of Higher Education Malaysia for Fundamental Research Grant Scheme with Project Code: FRGS/1/2023/ICT02/USM/02/2.

Data availability

The video datasets that support the findings of this study are publicly available from https://www.crcv.ucf.edu/data/UCF101/UCF101.rar, https://serre-lab.clps.brown.edu/resource/hmdb-a-large-human-motion-database, and http://www.svcl.ucsd.edu/projects/resound/dataset.html.

References

[1] I. Dave, R. Gupta, M.N. Rizve, M. Shah, Tclr: Temporal contrastive learning for video representation, Comput. Vis. Image Underst. 219 (2022) 103406, http://dx.doi.org/10.1016/j.cviu.2022.103406.
[2] F. Wang, H. Liu, Understanding the behaviour of contrastive loss, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 2495–2504, http://dx.doi.org/10.48550/arXiv.2012.09740.
[3] L. Jing, Y. Tian, Self-supervised visual feature learning with deep neural networks: A survey, IEEE Trans. Pattern Anal. Mach. Intell. 43 (11) (2020) 4037–4058, http://dx.doi.org/10.1109/TPAMI.2020.2992393.

[4] I. Misra, C.L. Zitnick, M. Hebert, Shuffle and learn: Unsupervised learning using temporal order verification, in: B. Leibe, J. Matas, N. Sebe, M. Welling (Eds.), Computer Vision – ECCV 2016, Springer International Publishing, 2016, pp. 527–544, http://dx.doi.org/10.1007/978-3-319-46448-0_32.

[5] J. Gui, T. Chen, J. Zhang, Q. Cao, Z. Sun, H. Luo, D. Tao, A survey on self-supervised learning: Algorithms, applications, and future trends, IEEE Trans. Pattern Anal. Mach. Intell. 46 (12) (2024) 9052–9071, http://dx.doi.org/10.1109/TPAMI.2024.3415112.

[6] S. Benaim, A. Ephrat, O. Lang, I. Mosseri, W.T. Freeman, M. Rubinstein, M. Irani, T. Dekel, Speednet: Learning the speediness in videos, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 9922–9931, http://dx.doi.org/10.48550/arXiv.2004.06130.

[7] H. Cho, T. Kim, H.J. Chang, W. Hwang, Self-supervised visual learning by variable playback speeds prediction of a video, IEEE Access 9 (2021) 79562–79571, http://dx.doi.org/10.1109/ACCESS.2021.3084840.

[8] Y. Yao, C. Liu, D. Luo, Y. Zhou, Q. Ye, Video playback rate perception for self-supervised spatio-temporal representation learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 6548–6557.

[9] H. Altabrawee, M.H. Mohd Noor, Repeat and learn: Self-supervised visual representations learning by repeated scene localization, Pattern Recognit. 156 (2024) 110804, http://dx.doi.org/10.1016/j.patcog.2024.110804, https://www.sciencedirect.com/science/article/pii/S0031320324005557.

[10] S. Jenni, G. Meishvili, P. Favaro, Video representation learning by recognizing temporal transformations, in: European Conference on Computer Vision, Springer, 2020, pp. 425–442.

[11] X. Li, S. Liu, S. De Mello, X. Wang, J. Kautz, M.-H. Yang, Joint-task self-supervised learning for temporal correspondence, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 32, Curran Associates, Inc., 2019.

[12] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, A. Zisserman, Temporal cycle-consistency learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 1801–1810.

[13] Z. Lai, W. Xie, Self-supervised learning for video correspondence flow, 2019, http://dx.doi.org/10.48550/arXiv.1905.00875, arXiv preprint arXiv:1905.00875.

[14] Z. Lai, E. Lu, W. Xie, Mast: A memory-augmented self-supervised tracker, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 6479–6488, http://dx.doi.org/10.48550/arXiv.2002.07793.

[15] S. Kong, C. Fowlkes, Multigrid predictive filter flow for unsupervised learning on videos, 2019, http://dx.doi.org/10.48550/arXiv.1904.01693, arXiv preprint arXiv:1904.01693.

[16] M. Minderer, C. Sun, R. Villegas, F. Cole, K.P. Murphy, H. Lee, Unsupervised learning of object structure and dynamics from videos, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 32, Curran Associates, Inc., 2019.

[17] T. Han, W. Xie, A. Zisserman, Video representation learning by dense predictive coding, in: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, 2019.

[18] T. Han, W. Xie, A. Zisserman, Memory-augmented dense predictive coding for video representation learning, in: European Conference on Computer Vision, Springer, 2020, pp. 312–329.

[19] M. Liu, K. Liang, Y. Zhao, W. Tu, S. Zhou, X. Gan, X. Liu, K. He, Self-supervised temporal graph learning with temporal and structural intensity alignment, IEEE Trans. Neural Netw. Learn. Syst. (2024) 1–13, http://dx.doi.org/10.1109/TNNLS.2024.3386168.

[20] S. Jenni, H. Jin, Time-equivariant contrastive video representation learning, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 9970–9980.

[21] J. Wang, J. Jiao, Y.-H. Liu, Self-supervised video representation learning by pace prediction, in: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVII 16, Springer, 2020, pp. 504–521.

[22] X. Yang, M. Mirmehdi, T. Burghardt, Back to the future: Cycle encoding prediction for self-supervised contrastive video representation learning, 2020, http://dx.doi.org/10.48550/arXiv.2010.07217, arXiv preprint arXiv:2010.07217.

[23] M. Liu, K. Liang, D. Hu, H. Yu, Y. Liu, L. Meng, W. Tu, S. Zhou, X. Liu, TMac: Temporal multi-modal graph learning for acoustic event classification, in: Proceedings of the 31st ACM International Conference on Multimedia, MM '23, Association for Computing Machinery, New York, NY, USA, 2023, pp. 3365–3374, http://dx.doi.org/10.1145/3581783.3611853.

[24] D.-A. Huang, V. Ramanathan, D. Mahajan, L. Torresani, M. Paluri, L. Fei-Fei, J.C. Niebles, What makes a video a video: Analyzing temporal information in video understanding models and datasets, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7366–7375.

[25] L. Fan, S. Buch, G. Wang, R. Cao, Y. Zhu, J.C. Niebles, L. Fei-Fei, Rubiksnet: Learnable 3d-shift for efficient video action recognition, in: European Conference on Computer Vision, Springer, 2020, pp. 505–521.

[26] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, L. Van Gool, Temporal segment networks: Towards good practices for deep action recognition, in: European Conference on Computer Vision, Springer, 2016, pp. 20–36.

[27] Z. Lan, Y. Zhu, A.G. Hauptmann, S. Newsam, Deep local video feature for action recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017, pp. 1–7, http://dx.doi.org/10.48550/arXiv.1701.07368.

[28] B. Zhou, A. Andonian, A. Oliva, A. Torralba, Temporal relational reasoning in videos, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 803–818, http://dx.doi.org/10.48550/arXiv.1711.08496.

[29] A. Piergiovanni, W. Kuo, A. Angelova, Rethinking video ViTs: Sparse video tubes for joint image and video learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2023, pp. 2214–2224.

[30] J. Lei, L. Li, L. Zhou, Z. Gan, T.L. Berg, M. Bansal, J. Liu, Less is more: ClipBERT for video-and-language learning via sparse sampling, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2021, pp. 7331–7341.

[31] Y. Gao, Z. Lu, SST-VLM: Sparse sampling-twice inspired video-language model, in: Proceedings of the Asian Conference on Computer Vision, ACCV, 2022, pp. 1194–1210.

[32] C. Feichtenhofer, h. fan, Y. Li, K. He, Masked autoencoders as spatiotemporal learners, in: S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, A. Oh (Eds.), Advances in Neural Information Processing Systems, Vol. 35, Curran Associates, Inc., 2022, pp. 35946–35958.

[33] Z. Tong, Y. Song, J. Wang, L. Wang, Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training, in: S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, A. Oh (Eds.), Advances in Neural Information Processing Systems, Vol. 35, Curran Associates, Inc., 2022, pp. 10078–10093.

[34] D. Luo, Y. Zhou, B. Fang, Y. Zhou, D. Wu, W. Wang, Exploring relations in untrimmed videos for self-supervised learning, ACM Trans. Multimed. Comput. Commun. Appl. (TOMM) 18 (1s) (2022) 1–21, http://dx.doi.org/10.1145/3473342.

[35] R. Qian, T. Meng, B. Gong, M.-H. Yang, H. Wang, S. Belongie, Y. Cui, Spatiotemporal contrastive video representation learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 6964–6974, http://dx.doi.org/10.48550/arXiv.2008.03800.

[36] S. Guo, Z. Xiong, Y. Zhong, L. Wang, X. Guo, B. Han, W. Huang, Cross-architecture self-supervised video representation learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 19270–19279, http://dx.doi.org/10.48550/arXiv.2205.13313.

[37] Z. Qing, S. Zhang, Z. Huang, Y. Xu, X. Wang, M. Tang, C. Gao, R. Jin, N. Sang, Learning from untrimmed videos: Self-supervised video representation learning with hierarchical consistency, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2022, pp. 13821–13831, http://dx.doi.org/10.48550/arXiv.2204.03017.

[38] C. Yang, Y. Xu, B. Dai, B. Zhou, Video representation learning with visual tempo consistency, 2020, http://dx.doi.org/10.48550/arXiv.2006.15489, arXiv preprint arXiv:2006.15489.

[39] J. Wang, G. Bertasius, D. Tran, L. Torresani, Long-short temporal contrastive learning of video transformers, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 14010–14020, http://dx.doi.org/10.48550/arXiv.2106.09212.

[40] Y. Tian, D. Krishnan, P. Isola, Contrastive multiview coding, in: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16, Springer, 2020, pp. 776–794.

[41] T. Han, W. Xie, A. Zisserman, Self-supervised co-training for video representation learning, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, Vol. 33, Curran Associates, Inc., 2020, pp. 5679–5690.

[42] C.-E. Wu, F. Lai, Y.H. Hu, A. Kadav, Self-supervised video representation learning with cascade positive retrieval, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 4070–4079, http://dx.doi.org/10.48550/arXiv.2201.07989.

[43] J. Park, J. Lee, I.-J. Kim, K. Sohn, Probabilistic representations for video contrastive learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 14711–14721, http://dx.doi.org/10.48550/arXiv.2204.03946.

[44] A. Diba, V. Sharma, R. Safdari, D. Lotfi, S. Sarfraz, R. Stiefelhagen, L. Van Gool, Vi2clr: Video and image for visual contrastive learning of representation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 1502–1512.

[45] J. Wang, Y. Lin, A.J. Ma, P.C. Yuen, Self-supervised temporal discriminative learning for video representation learning, 2020, http://dx.doi.org/10.48550/arXiv.2008.02129, arXiv preprint arXiv:2008.02129.

[46] L. Tao, X. Wang, T. Yamasaki, Self-supervised video representation learning using inter-intra contrastive framework, in: Proceedings of the 28th ACM International Conference on Multimedia, 2020, pp. 2193–2201, http://dx.doi.org/10.1145/3394171.3413694.

[47] S.H. Khorasgani, Y. Chen, F. Shkurti, Slic: Self-supervised learning with iterative clustering for human action videos, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 16091–16101.

[48] A. Miech, J.-B. Alayrac, L. Smaira, I. Laptev, J. Sivic, A. Zisserman, End-to-end learning of visual representations from uncurated instructional videos, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 9879–9889, http://dx.doi.org/10.48550/arXiv.1912.06430.

[49] C. Sun, F. Baradel, K. Murphy, C. Schmid, Learning video representations using contrastive bidirectional transformer, 2019, http://dx.doi.org/10.48550/arXiv.1906.05743, arXiv preprint arXiv:1906.05743.

[50] B. Korbar, Co-training of audio and video representations from self-supervised temporal synchronization, 2018.

[51] P. Morgado, N. Vasconcelos, I. Misra, Audio-visual instance discrimination with cross-modal agreement, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 12475–12486, http://dx.doi.org/10.48550/arXiv.2004.12943.

[52] A. Rouditchenko, A. Boggust, D. Harwath, B. Chen, D. Joshi, S. Thomas, K. Audhkhasi, H. Kuehne, R. Panda, R. Feris, et al., Avlnet: Learning audio-visual language representations from instructional videos, 2020, http://dx.doi.org/10.48550/arXiv.2006.09199, arXiv preprint arXiv:2006.09199.

[53] P.P. Liang, Z. Deng, M.Q. Ma, J.Y. Zou, L.-P. Morency, R. Salakhutdinov, Factorized contrastive learning: Going beyond multi-view redundancy, in: A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, S. Levine (Eds.), Advances in Neural Information Processing Systems, Vol. 36, Curran Associates, Inc., 2023, pp. 32971–32998.

[54] Y. Wang, C.M. Albrecht, N.A.A. Braham, C. Liu, Z. Xiong, X.X. Zhu, Decoupling common and unique representations for multimodal self-supervised learning, in: A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, G. Varol (Eds.), Computer Vision – ECCV 2024, Springer Nature Switzerland, Cham, 2025, pp. 286–303.

[55] C. Zhou, Z. Wang, X. Zhang, B. Du, Domain complementary adaptation by leveraging diversity and discriminability from multiple sources, IEEE Trans. Multimed. 26 (2024) 4490–4501, http://dx.doi.org/10.1109/TMM.2023.3323868.

[56] K. Soomro, A.R. Zamir, M. Shah, UCF101: A dataset of 101 human actions classes from videos in the wild, 2012, http://dx.doi.org/10.48550/arXiv.1212.0402, arXiv preprint arXiv:1212.0402.

[57] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, T. Serre, HMDB: a large video database for human motion recognition, in: 2011 International Conference on Computer Vision, IEEE, 2011, pp. 2556–2563, http://dx.doi.org/10.1109/ICCV.2011.6126543.

[58] Y. Li, Y. Li, N. Vasconcelos, RESOUND: Towards action recognition without representation bias, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018.

[59] D. Xu, J. Xiao, Z. Zhao, J. Shao, D. Xie, Y. Zhuang, Self-supervised spatiotemporal learning via video clip order prediction, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 10334–10343.

[60] J. Carreira, A. Zisserman, Quo vadis, action recognition? a new model and the kinetics dataset, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 6299–6308, http://dx.doi.org/10.48550/arXiv.1705.07750.

[61] J. Wang, J. Jiao, L. Bao, S. He, W. Liu, Y.-H. Liu, Self-supervised video representation learning by uncovering spatio-temporal statistics, IEEE Trans. Pattern Anal. Mach. Intell. 44 (7) (2021) 3791–3806, http://dx.doi.org/10.1109/TPAMI.2021.3057833.

[62] D. Luo, C. Liu, Y. Zhou, D. Yang, C. Ma, Q. Ye, W. Wang, Video cloze procedure for self-supervised spatio-temporal learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 11701–11708, http://dx.doi.org/10.1609/aaai.v34i07.6840.

[63] G. Lorre, J. Rabarisoa, A. Orcesi, S. Ainouz, S. Canu, Temporal contrastive pretraining for video action recognition, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2020, pp. 662–670.

[64] Y. Huo, M. Ding, H. Lu, Z. Huang, M. Tang, Z. Lu, T. Xiang, Self-supervised video representation learning with constrained spatiotemporal jigsaw, in: Z.-H. Zhou (Ed.), Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21, International Joint Conferences on Artificial Intelligence Organization, 2021, pp. 751–757, http://dx.doi.org/10.24963/ijcai.2021/104.

[65] N. Behrmann, J. Gall, M. Noroozi, Unsupervised video representation learning by bidirectional feature prediction, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2021, pp. 1670–1679, http://dx.doi.org/10.48550/arXiv.2011.06037.

[66] W. Li, D. Luo, B. Fang, Y. Zhou, W. Wang, Video 3d sampling for self-supervised representation learning, 2021, http://dx.doi.org/10.48550/arXiv.2107.03578, arXiv preprint arXiv:2107.03578.

[67] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification, in: Proceedings of the IEEE International Conference on Computer Vision, ICCV, 2015.

[68] J. Choi, C. Gao, J.C.E. Messou, J.-B. Huang, Why can't I dance in the mall? Learning to mitigate scene bias in action recognition, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 32, Curran Associates, Inc., 2019.

[69] P. Chen, D. Huang, D. He, X. Long, R. Zeng, S. Wen, M. Tan, C. Gan, Rspnet: Relative speed perception for unsupervised video representation learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, 2021, pp. 1045–1053, http://dx.doi.org/10.1609/aaai.v35i2.16189.

[70] R. Qian, Y. Li, H. Liu, J. See, S. Ding, X. Liu, D. Li, W. Lin, Enhancing self-supervised video representation learning via multi-level feature optimization, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 7990–8001, http://dx.doi.org/10.48550/arXiv.2108.02183.

[71] Y. Bai, H. Fan, I. Misra, G. Venkatesh, Y. Lu, Y. Zhou, Q. Yu, V. Chandra, A. Yuille, Can temporal information help with contrastive self-supervised learning?, 2020, http://dx.doi.org/10.48550/arXiv.2011.13046, arXiv preprint arXiv:2011.13046.

Hussein Altabrawee is currently a Ph.D. candidate in computer science at the Universiti Sains Malaysia. He has worked as a university lecturer in computer science at Al-Muthanna University from 2014 to 2020. Hussein received a Bachelor degree in computer science from the University of Babylon in 2007. He studied computer science at California State University Fullerton, and received a Master of Science degree in 2013. His research interest focuses on machine learning, deep learning, and computer vision.

Mohd Halim Mohd Noor received the B.Eng. degree (Hons.) in 2004, the M.Sc. degree in 2009, and the Ph.D. degree in computer systems engineering from the University of Auckland, New Zealand, in 2017. He is currently an Associate Professor with the School of Computer Sciences at Universiti Sains Malaysia. His research interests include machine learning, deep learning, computer vision, and pervasive computing.