Abstract

Instance-level contrastive learning techniques, which rely on data augmentation and a contrastive loss function, have found great success in the domain of visual representation learning. They are not suitable for exploiting the rich dynamical structure of video however, as operations are done on many augmented instances. In this paper we propose ‘Video Cross-Stream Prototypical Contrasting’, a novel method which predicts consistent prototype assignments from both RGB and optical flow views, operating on sets of samples. Specifically, we alternate the optimization process; while optimizing one of the streams, all views are mapped to one set of stream prototype vectors. Each of the assignments is predicted with all views except the one matching the prediction, pushing representations closer to their assigned prototypes. As a result, more efficient video embeddings with ingrained motion information are learned, without the explicit need for optical flow computation during inference. We obtain state-of-the-art results on nearest-neighbour video retrieval and action recognition, outperforming previous best by +3.2% on UCF101 using the S3D backbone (90.5% Top-1 acc), and by +7.2% on UCF101 and +15.1% on HMDB51 using the R(2+1)D backbone.

1. Introduction

The goal of this paper is self-supervised representation learning for video. Visual representation learning methods based on instance-level contrasting have significantly reduced the gap with supervised learning in image-based tasks [12, 29, 65] and video [55, 27]. These contrastive learning frameworks require an augmentation module that obtains multiple views of one instance, and a loss function that contrasts between augmented views of instances. The objective can be viewed as instance discrimination: producing higher similarity scores between augmentations of the same instances, rather than with those that belong to different ones (negative examples). As a result, the methods rely heavily on data augmentation in order to learn powerful representations. Furthermore, a vast amount of negative examples has to be obtained which often relies on either memory banks [29] or large batch sizes [12].

To adopt these techniques into the video domain efficiently, we make the following observations. First, we notice that though video also provides natural augmentation with viewpoint changes, illumination (jittering) and deformation, still spatiotemporal coherence and motion are not explicitly used. We are inspired by the two-streams hypothesis for vision processing in the brain [57, 22], suggesting two pathways: the ventral stream involved in object recognition and the dorsal stream locating objects and recognizing motion. Motion without appearance information can be a rich source of information for humans [33], however more recent works propose that it is likely that the streams interact [45]. Second, we believe instance-level contrastive learning is inefficient and neglects the use of semantic similarity between instances. Low similarity scores are produced for a large pool of negative pairs regardless of their semantic similarity, resulting in undesirable distances between samples in the embedding. To resolve this, several works have explored alternatives to random sampling for negative examples [6, 13], such as hard negative mining [34, 56]. We are instead interested in leaving instance-level comparisons
and include mappings to prototypes (defined as representatives of semantically similar groups of features), providing a possible benefit on video representations without any potential costs from distance searches in the data.

In this work, we present a novel self-supervised method called Video Cross-Stream Prototypical Contrasting (ViCC) where we consider RGB and optical flow as distinct views for video contrastive learning, to influence appearance and motion learning respectively. The two input streams and spatiotemporal augmentations are united into one framework. In each iteration of the optimization of one stream, views are assigned to a set of prototypes and assignments are subsequently predicted from the features, see Figure 1.

Our contributions can be summarized as below.

- We introduce a novel visual-only self-supervised learning framework for video that contrasts using sets of views from two streams (RGB and flow). We demonstrate the benefits of operating on stream prototypes over contrastive instance learning, avoiding unnecessary comparisons and hence computations, while improving accuracy.
- We propose a new training mechanism for video, in which RGB and flow streams are interconnected in two ways: prototypes are predicted from both streams and the optimization process is alternated. As motion information is transferred to the RGB model, we can discard the optical flow network in deployment scenarios depending on speed and efficiency requirements.
- We perform extensive ablation studies to provide an in-depth analysis of our method. Our result reaches state-of-the-art on UCF101 [59] and HMDB51 [36] on the two backbones S3D [74] and R(2+1)D [64].

2. Related work

**Contrastive instance learning.** Instance discrimination considers each sample as its own class in the data. As such a classifier becomes computationally infeasible fast. [73] use noise-contrastive estimation [24] and a memory bank to store representations as their pool of negative samples. Other solutions include the work from Chen et al. [13], which retrieves more negative samples by using large batch sizes. He et al. [29] propose a momentum encoder with a dynamic dictionary look-up. Another line of work considering contrast between the global image and local patches [65, 30]. Our method instead uses complementary modalities as main views and intuitively learns its own positive and negative examples from both feature spaces through the prototypes. Contrasting is done between instances and prototypes, going beyond instance-level learning while avoiding the need for substantial batch sizes [12] or large memory banks [29].

**Clustering in latent space.** Combining clustering with representation learning to obtain pseudo-labels has been proposed in various self-supervised learning settings [76, 7, 2, 8, 41]. Asano et al. [3] propose a solution of degenerate solutions by casting clustering into an instance of the optimal transport problem. Caron et al. [8] use this clustering setup in a contrastive learning setting by enforcing consistency between different views, comparing cluster assignments instead of individual features. Furthermore, an online clustering and simultaneous feature learning mechanism was proposed in [60]. Our objective is most similar to [8] and [41], aligning cluster assignments for augmented instances in an online manner. However, we apply our method on video, use augmentation in the form of optical flow and alternate the training of models and prototypes to incorporate information in both streams.

**Self-supervised video and distillation.** Advances in 3DConvNets [63] [28] [64] have driven video research forwards. Self-supervised approaches exploring pretext tasks are often based on the temporal domain, such as the order of frames or clips [49] [19] [40] [75], learning the arrow of time [53] [72] or pace [7] [14] [69] [4]. Pretext tasks that were previously explored in the image domain have been proposed and extended [32] [35]. Other approaches include leveraging the consistency in frames by temporal correspondence [58] [39], tracking patches [70] [71], future frame prediction [23] [67] or future feature prediction [25] [26]. Multiple works explore optical flow for self-supervision [26] [27] [44]. A cross-stream approach was first proposed by [50]. As opposed to them, we use contrastive learning without dense trajectories. Mahendran et al. [44] use optical flow as supervision for RGB. Tian et al. [62] first explore the use of RGB and optical flow as views for contrastive learning. Most similar to our work are [69] and [27], which both employ RGB and flow in a two-stream manner for contrastive learning. Han et al. [27] use an alternated training process and samples hard positive examples from the other stream. Different from these works, we do not employ instance-level contrastive learning. As we use prototype mappings of our features and subsequently predict feature assignments, our streams leverage a stronger interplay. We also incorporate informed negative examples from both streams through our prototypes and we do not use a momentum encoder [29]. As optical flow computation can be computationally expensive, several works avoid flow computation during inference while utilizing it during training, e.g. through knowledge distillation [60] [16] [81] [20] which is related to our work. Our proposed method instead keeps two streams and leverages an alternated optimization process to perform a form of distillation through contrastive learning, avoiding the need for optical flow while still enabling its optional use.

**Multi-modal approaches.** Video allows for a multi-modal approach by using information such as audio [11] [2] and text [46] [61] to learn from correspondence between modalities. Alwassel et al. [1] use a cross-modal audio-video iterative clustering and relabeling algorithm. Asano et al. [2] employ both RGB and audio in a simultaneous
clustering and representation learning setting, following [3]. Our method strictly speaking does not leverage multiple modalities as we use an optical flow representation originally extracted from the RGB representation, without introducing any external information. However, our work similarly leverages the interplay of complementary information and could therefore be used alternatively as a multi-modal approach, e.g., leveraging audio in addition to optical flow in order to improve representations further.

3. Method

We first introduce preliminaries on instance-level contrastive learning in Section 3.1. We explain how we can use RGB and optical flow separately to predict and learn prototypes following [8] in Section 3.2. Finally, we introduce our contribution which consists of the cross-stream interplay and the steps of our algorithm in Section 3.3.

3.1. Preliminaries

Contrastive instance learning [29, 12] can be defined as a self-supervised learning method which contrasts in the latent space by maximizing agreement between different augmented views of the same data instances. Three key components in this framework are i) a data augmentation module that transforms a given sample into two views \( x_i \) and \( x_j \) by applying separate transformations \( t \) and \( t' \) sampled from the set of augmentations \( T \), ii) the embedding function \( f(\cdot) \) consisting of an encoder and a small MLP projection head that extracts feature vectors \( z_i \) and \( z_j \) from views, and iii) a contrastive loss function that contrasts between \( x_i \) and a set \( \{x_k\} \) of augmented pairs that includes our positive pair. Given a dataset \( X = \{x_1, x_2, ..., x_n\} \), we aim to learn a function \( f(\cdot) \) that maps \( X \) to \( Z = \{z_1, z_2, ..., z_n\} \). The contrastive loss objective for a positive pair \((i,j)\), referred to as the InfoNCE loss [58, 65, 12], is then given by

\[
\mathcal{L}_{\text{InfoNCE}}(z_i, z_j) = -\log \frac{\exp(z_i \cdot z_j / \tau)}{\sum_{k \neq i} \exp(z_i \cdot z_k / \tau)},
\]

where \( \tau \) is the temperature hyperparameter and \( z_i \cdot z_j \) refers to the dot product between normalized vectors, i.e., cosine similarity. The final loss is computed for all available positive pairs. Given a positive pair, a sufficiently large number of negative examples in \( \{x_k\} \) needs to be available for which storage of features besides the mini-batch is often needed. The contrastive learning mechanism also neglects to take into account the informativeness of samples.

3.2. Predicting stream prototype assignments

In our proposed method we avoid instance-level contrasting by using for each stream a set of prototypes in our contrasting. Furthermore, we extend the augmentation module by considering RGB frames and optical flow as views. Mathematically, given a video clip \( x \) we first consider the two streams as views, obtaining \( x = \{x^1, x^2\} \) which describe a RGB and a flow sample respectively. The objective is to learn the stream representations \( z^1 = f_1(x^1) \) and \( z^2 = f_2(x^2) \) through learning their encodings \( f_1(\cdot) \) and \( f_2(\cdot) \). Each of the encoders has a set of \( K \) trainable prototype vectors, \( \{c^1_1, ..., c^1_K\} \in C_1 \) and \( \{c^2_1, ..., c^2_K\} \in C_2 \), implemented as a linear layer in the networks.

Consider only the training of one encoder \( f_s \) on its own stream \( s \) where \( s \in \{1, 2\} \). We denote the corresponding prototype set as matrix \( C_s \) with columns \( c^s_1, ..., c^s_K \). Given input sample \( x^s \), we obtain two augmented versions \( \{x^s_i, x^s_j\} \). After applying the encoder \( f_s(\cdot) \) we obtain features \( \{z^s_i, z^s_j\} \). The features are mapped to the set of prototypes \( C_s \) to obtain cluster assignments \( \{q^s_i, q^s_j\} \), as detailed in the following section. The features and assignments are subsequently used in the following prediction loss:

\[
\mathcal{L}_{s}^{\text{Single-stream}}(z^s_i, z^s_j) = l_s(z^s_j, q^s_i) + l_s(z^s_i, q^s_j). \tag{2}
\]

Each of the terms represents the cross-entropy loss between the stream prototype assignment \( q \) and the probability obtained by a softmax on the similarity between \( z \) and \( C_s \):

\[
l_s(z^s_j, q^s_i) = -\sum_k q^s_i(k) \log \frac{\exp(z^s_i \cdot c^s_k / \tau)}{\sum_{k'} \exp(z^s_j \cdot c^s_{k'}/ \tau)}, \tag{3}
\]

where \( \tau \) is a temperature hyperparameter. The objective is to maximize the agreement of prototype assignments from multiple views of one sample (RGB or flow). Features are contrasted indirectly through comparing their prototype assignments. The total loss of training the encoder \( f_s \) on its own stream is taken over all videos and pairs of data augmentations, minimized with respect to both \( f_s \) and \( C_s \).

**Learning stream prototype assignments.** The assignments \( \{q^s_i, q^s_j\} \) are computed by matching features \( \{z^s_i, z^s_j\} \) to prototypes \( C_s \). In essence, we need to consider the cross-entropy for assigning each \( z \) to \( C_s \) and perform a mapping to assign labels automatically. Optimizing \( q \) directly leads to degeneracy. Following [3, 8] a uniform split of the features across prototypes is enforced, which avoids the collapse of assignments to one prototype. Given our feature vectors \( Z \) whose columns are \( z_1, ..., z_B \), we map them to \( C_s \) and optimize using an Optimal Transport [52] solver the mapping \( Q = q_1, ..., q_B \):

\[
\max_{Q \in \mathcal{Q}} \text{Tr} (Q^T C_s^T Z) + \epsilon H(Q), \tag{4}
\]

where \( H(Q) \) is the entropy of \( Q \) which acts as a regularizer. The \( \epsilon \) parameter controls the uniformity of the assignment where a low value helps to avoid collapse. Following [8], we restrict the transportation polytope to mini-batches:

\[
Q = \{Q \in \mathbb{R}^{K \times B} | Q_{1B} = \frac{1}{K} I_K, Q^T 1_K = \frac{1}{B} 1_B\}, \tag{5}
\]
where \( I_K \) denotes a vector of all ones with dimension \( K \). We preserve soft assignments \( Q^* \) and the solution of the transportation polytope, solved efficiently using the Sinkhorn-Knopp algorithm \([17]\) can be written as follows:

\[
Q^* = \text{Diag}(\alpha) \exp\left(\frac{1}{\epsilon} C^T_s Z\right) \text{Diag}(\beta),
\]

where \( \alpha \) and \( \beta \) denote renormalization vectors such that \( Q \) results in a probability matrix \([17]\). As the amount of batch terms is given by:

\[
l_s(z^s_i, q^t_l) = -\sum_k q^{t,(k)}_l \log \frac{\exp(z^s_i \cdot c^t_k/\tau)}{\sum_{k'} \exp(z^s_i \cdot c^t_{k'}/\tau)},
\]

where we predict the assignment \( q^t \) from stream \( t \) (obtained by matching corresponding feature \( z^s_i \) to the prototypes \( C_s \)) using one of the augmented features \( z^s_i \) from stream \( s \).

In summary, we predict assignments from each of the four views using features originating from three views, see Figure 2. In the prediction of each \( q \), we avoid the use of the feature \( z \) where \( s \) is equal to \( t \) (same stream) and \( i \) is equal to \( j \) (same augmentation). This setup forces the features to capture the same information by predicting consistent assignments from them. The total loss for cross-stream training on stream \( s \) is taken over all videos and pairs of augmentations, optimized with respect to \( f_s \) and \( C_s \).

**3.3. Learning cross-stream**

We are now interested in using information from both streams for each encoder. Consider again the encoder \( f_s \) and prototypes \( C_s \) from the stream \( s \) that is optimized in one alternation. We now add a second stream \( t \) where \( t \in \{1, 2\} \) and \( s \neq t \).

We use the encoder \( f_s \) with frozen weights and obtain samples \( \{x^s_1, x^s_2\} \) and features \( \{z^s_1, z^s_2\} \). By matching these features to prototypes \( C_s \), the assignments \( \{q^s_1, q^s_2\} \) are obtained. Given \( f_s \), \( C_s \), and \( f_t \), all initialized with prior representations learned on their own stream, the loss function for the prediction problem consist of four main parts:

\[
\mathcal{L}^\text{Cross-stream}_s(z^s_i, z^t_j, q^s_1, q^t_2) = L_s(z^s_i, z^t_j, q^s_1) + L_s(z^s_i, z^t_j, q^t_2) + L_s(z^s_i, z^s_i, q^s_1) + L_s(z^s_i, z^s_i, q^t_2),
\]

where the function \( L_s \) measures the fit between three features \( z \) and an assignment \( q \). For instance, the first of the \( L_s \) terms is given by:

\[
L_s(z^s_i, q^s_1) = l_s(z^s_i, q^s_1) = l_s(z^s_i, q^t_1) + l_s(z^s_i, q^t_2) + l_s(z^t_j, q^s_1).
\]

The total loss function therefore consist of 12 terms. Each of the terms \( l_s \) again represents the cross-entropy between one feature \( z \) and one assignment \( q \), e.g.:

\[
l_s(z^s_i, q^t_l) = -\sum_k q^{t,(k)}_l \log \frac{\exp(z^s_i \cdot c^t_k/\tau)}{\sum_{k'} \exp(z^s_i \cdot c^t_{k'}/\tau)},
\]

**Figure 2. Video Cross-Stream Prototypical Contrasting.** Two different augmented samples are obtained for both RGB and flow. The encoders \( f_1 \) and \( f_2 \) map samples from RGB and flow respectively to obtain features \( z^s_1, z^t_j, z^t_i, z^t_j \), which are in turn assigned to either RGB or flow prototype vectors, depending on which stream \( s \in \{1, 2\} \) is optimized. Next, the stream prototype assignments \( q^s_1, q^s_2, q^t_1, q^t_2 \) are predicted using features only from the three other views. The encoder and prototypes from the optimized stream are updated by backpropagation, while the other encoder remains fixed.

**3.3. Learning cross-stream**

We are now interested in using information from both streams for each encoder. Consider again the encoder \( f_s \) and prototypes \( C_s \) from the stream \( s \) that is optimized in one alternation. We now add a second stream \( t \) where \( t \in \{1, 2\} \) and \( s \neq t \).

We use the encoder \( f_s \) with frozen weights and obtain samples \( \{x^s_1, x^s_2\} \) and features \( \{z^s_1, z^s_2\} \). By matching these features to prototypes \( C_s \), the assignments \( \{q^s_1, q^s_2\} \) are obtained. Given \( f_s \), \( C_s \), and \( f_t \), all initialized with prior representations learned on their own stream, the loss function for the prediction problem consist of four main parts:

\[
\mathcal{L}^\text{Cross-stream}_s(z^s_i, z^t_j, q^s_1, q^t_2) = L_s(z^s_i, z^t_j, q^s_1) + L_s(z^s_i, z^t_j, q^t_2) + L_s(z^s_i, z^s_i, q^s_1) + L_s(z^s_i, z^s_i, q^t_2),
\]

where the function \( L_s \) measures the fit between three features \( z \) and an assignment \( q \). For instance, the first of the \( L_s \) terms is given by:

\[
L_s(z^s_i, q^s_1) = l_s(z^s_i, q^s_1) = l_s(z^s_i, q^t_1) + l_s(z^s_i, q^t_2) + l_s(z^t_j, q^s_1).
\]

The total loss function therefore consist of 12 terms. Each of the terms \( l_s \) again represents the cross-entropy between one feature \( z \) and one assignment \( q \), e.g.:

\[
l_s(z^s_i, q^t_l) = -\sum_k q^{t,(k)}_l \log \frac{\exp(z^s_i \cdot c^t_k/\tau)}{\sum_{k'} \exp(z^s_i \cdot c^t_{k'}/\tau)},
\]

where we predict the assignment \( q^t_i \) from stream \( t \) (obtained by matching corresponding feature \( z^s_i \) to the prototypes \( C_s \)) using one of the augmented features \( z^s_i \) from stream \( s \).

In summary, we predict assignments from each of the four views using features originating from three views, see Figure 2. In the prediction of each \( q \), we avoid the use of the feature \( z \) where \( s \) is equal to \( t \) (same stream) and \( i \) is equal to \( j \) (same augmentation). This setup forces the features to capture the same information by predicting consistent assignments from them. The total loss for cross-stream training on stream \( s \) is taken over all videos and pairs of augmentations, optimized with respect to \( f_s \) and \( C_s \).

**ViCC Algorithm.** Our complete algorithm is structured as follows. **Stage 1) Single-stream.** In the first stage, the two encoders \( f_1 \) and \( f_2 \) and their prototypes \( C_1 \) and \( C_2 \) are initialized from scratch and trained using their own input stream, following Equation 2. **Stage 2) Cross-stream.** In the second stage, cross-stream, the two models are trained in an alternating fashion using input from both streams. In one alternation, one of the streams \( s \) with encoder \( f_s \) and prototypes \( C_s \) is encouraged to predict mappings consistently following Equation 7 leveraging complementary information from the other stream through assigning views from \( f_t \) to \( C_s \). Both the prototype mappings and the alternation process in our cross-stream mechanism serve as means for transferring knowledge from motion (flow) to RGB.
Inference. At the inference stage, depending on speed vs. accuracy requirements, both the RGB model $f_1$ trained with ViCC self-supervision can be used for downstream tasks as well as both RGB $f_1$ and flow $f_2$ by averaging predictions from the models.

4. Experiments

4.1. Experimental setup

We use two datasets for our experiments: HMDB51 [36] and UCF101 [59]. UCF101 consists of 13K videos over 101 human action classes. HMDB51 is another widely used action recognition dataset and contains around 7K videos over 51 action classes. UCF101 and HMDB51 are both divided into three train/test splits. For self-supervised training we use UCF101 training split 1 without class labels. For downstream evaluation we use UCF101 and HMDB51 and evaluate on split 1 for both datasets, following prior work [27].

Data preprocessing. From the source videos at 25fps, input video clips are extracted at random time stamps. Our input video clips have a spatial resolution of $128 \times 128$ pixels. We use clips of 32 frames as input, without temporal downsampling for S3D. For R(2+1)D and R3D, we use input clips of 16 frames with temporal downsampling at rate 2. For optical flow, we use the widely used TV-L1 algorithm [79] and follow practice in [10] [27] for preprocessing. This means that we truncate large vectors with more than 20 in both channels, transform the values to range $[0, 255]$ and append a third channel of 0s. Random cropping, horizontal flipping, Gaussian blur and color jittering are used in a frame-consistent manner on RGB and flow clips following recent works [12] [29]. For temporal augmentation we take clips at different time stamps with 50% probability.

Implementation and training. As our base encoder architecture we use S3D [74]. We also test our method with the R(2+1)D-18 [64] architecture, following recent works [2] [16], and the R3D-18 [29] backbone. We use a 2-layer MLP projection head during self-supervised training that projects the backbone output to 128 dimensional space following SimCLR [12]. In line with SwaV [8], we employ a linear layer updated by backpropagation as the prototype implementation. The projection head and the prototype layer are removed for downstream evaluation. During self-supervised training, we use a queue that consists of 1920 features. We use $K=300$ as the number of prototypes. The single-stream stage consists of 300 epochs. Next, the cross-stream stage is initialized with models from the single-stream stage and is trained for two cycles. In one cross-stream cycle, we first train RGB for 100 epochs and then flow for 100 epochs, each time taking the newest models, following CoCLR [27]. We run all our experiments with 4 Titan RTX GPUs with a batch size of 48.

Evaluation methods. We evaluate the quality of our learned video representation using two downstream video understanding tasks: nearest neighbour video retrieval and action recognition. In the former, retrieval is performed without any supervised finetuning. We follow common protocol [48] [5] [75] by using videos form the test set as queries for k nearest-neighbour (kNN) retrieval in the training set. We report Recall at $k$ (R@K) where we mark the retrieval as correct if a video of the same class appears among the top kNN. In the latter downstream task, we initialize with our representation and evaluate two settings: linear probe and finetuning. For linear probe, we freeze the entire network and add a linear classifier. For finetuning, the entire network with linear layer is trained end-to-end. We report Top-1 accuracy for both settings. Data augmentation similar to the self-supervision stage is used except for Gaussian blur. At inference we follow the ten-crop procedure, where the center crop, four corners and the horizontal flipped version of these crops are obtained. The moving-window approach is used for taking clips followed by averaging the predictions.

4.2. Model ablations

Impact of training stages. In Table 1 results are shown for several stages of our method in order to evaluate the improvement that cross-stream (Stage 1) has over single-stream (Stage 2). We report action recognition and nearest-neighbour video retrieval on UCF101 split 1 and include [27] as our baseline model, as it uses the contrastive instance loss on RGB and flow with additional positive examples. Training settings are kept identical across self-supervised models. All methods, including single-stream, are trained on an equal amount of epochs (500 in total). Evaluated on nearest-neighbour retrieval, we observe that our RGB-1 network gains a significant performance benefit when learning and predicting from optical flow in stage 2, shown as RGB-2 (62.1% vs. 40.0%). Furthermore, when combining predictions from the RGB-2 model and the Flow-2 model, both trained with cross-stream, we obtain a further performance boost shown as ViCC-R+F-2 (65.1% vs. 62.1%). We outperform [27] on retrieval by +9.5%, demonstrating the benefit of cross-stream prototype contrasting in ViCC. In linear probe downstream classification, our RGB-2 model again outperforms the RGB-1 one by a significant margin (72.2% vs. 49.2%). When end-to-end finetuned our self-supervised RGB-2 outperforms RGB-1 (84.3% vs. 81.8%). Further improvement is found by combining the predictions of the two streams, obtaining the result for R+F (90.5% vs. 84.3%). Here, our performance for R+F is on par with the RGB model from [27]. As our cross-stream phase consists of cycles in which we alternate the training of streams, we further analyse the performance progress on video retrieval across training phases in Figure 3. We show the evolution from single-stream to cross-stream for both models, where cross-stream consists of two
We first consider the number of features for prediction, where the normal setting is to use all other available views from streams $s$ and $t$ for prediction of each assignment $q$. We now study the setting where we use two features for prediction originating only from the other stream $t$. Table 2 shows results for both settings, reporting Top-1 accuracy on UCF101 action recognition using the finetuning protocol. We find that using two features results in a slightly worse performance overall, suggesting that more views are beneficial for prediction of the assignments despite originating from the same stream as the assignment. The second setting that we evaluate is only using the other stream $t$ for assignment, where we map only the two features from stream $t$ to prototypes $C_s$. Note, the prediction is performed as normal, using all other available views. Both models are again slightly underperforming compared to using all views. The results for stream views in both settings suggest that the information used from the other stream in ViCC cross-stream training is of more significance than its own stream. Indeed, we find that ViCC is robust against changes in views from its own stream as it almost performs in line with results using all views for both prediction and assignment.

### Impact of number of prototypes

| Method      | Stage | Input | Classification | Retrieval |
|-------------|-------|-------|----------------|-----------|
|             |       |       | Linear Acc     | Finetune Acc | R@1       |
| ViCC-RGB-1  | 1     | RGB   | 49.2           | 81.8       | 40.0      |
| ViCC-Flow-1 | 1     | Flow  | 71.9           | 87.9       | 55.5      |
| ViCC-RGB-2  | 2     | RGB   | 72.2           | 84.3       | 62.1      |
| ViCC-Flow-2 | 2     | Flow  | 75.5           | 88.7       | 59.7      |
| CoCLR [27]  | 2     | R+F   | 72.1           | 87.3       | 55.6      |
| ViCC-R+F-2  | 2     | R+F   | 78.0           | 90.5       | 65.1      |

Table 1. Improvement of ViCC from single-stream (stage 1) to cross-stream (stage 2) evaluated on action recognition and nearest-neighbour retrieval on UCF101. [27] is included as a baseline cross-stream (stage 2) evaluated on action recognition and nearest-accuracy on action recognition finetuning on UCF101.

![Figure 3. Retrieval performance progress on our training phases. RGB and flow are subsequently optimized in one cross-stream cycle, where a dotted line indicates no optimization. We report Top-1 Recall (R@1) on UCF101.](image)

We now study the setting where we use two features for prediction originating only from the other stream $t$. Table 2 shows results for both settings, reporting Top-1 accuracy on UCF101 action recognition using the finetuning protocol. We find that using two features results in a slightly worse performance overall, suggesting that more views are beneficial for prediction of the assignments despite originating from the same stream as the assignment. The second setting that we evaluate is only using the other stream $t$ for assignment, where we map only the two features from stream $t$ to prototypes $C_s$. Note, the prediction is performed as normal, using all other available views. Both models are again slightly underperforming compared to using all views. The results for stream views in both settings suggest that the information used from the other stream in ViCC cross-stream training is of more significance than its own stream. Indeed, we find that ViCC is robust against changes in views from its own stream as it almost performs in line with results using all views for both prediction and assignment.

### Impact of number of prototypes

We evaluate the impact of the number of stream prototypes $K$. Explored previously by [8] on ImageNet [18], they found no significant impact on performance when varying the prototypes by several orders of magnitude using a sufficiently large amount of prototypes. In Table 3, we show results on varying the number of prototypes to $K = \{100, 1000\}$. We observe a slightly worse result for both settings for the RGB model and the R+F model. As we find no significant impact on the performance, our results are in line with previous work suggesting that the soft prototype mappings used for contrasting in ViCC are not necessarily a self-labeling approach similar to other pseudo-labeling approaches [8, 2, 20, 7, 76], despite the usefulness in contrasting for representation learning.

### 4.3. Comparison with state-of-the-art

In this section, we compare our method with self-supervised methods on action classification and video retrieval, reporting our models from the cross-stream stage.

**Action recognition.** We compare with several self-supervised methods on action recognition in Table 4 displaying our results for two backbone architectures. We organized the methods by backbone and include settings such as resolution (Res), number of frames and number of parameters (Param) for a fairer comparison. We include several methods pretrained on larger training datasets for both visual-only and multi-modal methods. In the following, we compare with visual-only modality on the same training set, with visual-only on larger datasets, and with cycles in which RGB and Flow are trained alternately. It can be seen that representations for both models continue to improve after one cycle, indicating that the alternating scheme is beneficial for ViCC representations.

### Ablations on stream views

We perform an ablation study on our model by investigating the importance of the streams used as views for both prediction and assignment. We first consider the number of features for prediction, where the normal setting is to use all other available views from streams $s$ and $t$ for prediction of each assignment $q$. We now study the setting where we use two features for prediction originating only from the other stream $t$. Table 2 shows results for both settings, reporting Top-1 accuracy on UCF101 action recognition using the finetuning protocol. We find that using two features results in a slightly worse performance overall, suggesting that more views are beneficial for prediction of the assignments despite originating from the same stream as the assignment. The second setting that we evaluate is only using the other stream $t$ for assignment, where we map only the two features from stream $t$ to prototypes $C_s$. Note, the prediction is performed as normal, using all other available views. Both models are again slightly underperforming compared to using all views. The results for stream views in both settings suggest that the information used from the other stream in ViCC cross-stream training is of more significance than its own stream. Indeed, we find that ViCC is robust against changes in views from its own stream as it almost performs in line with results using all views for both prediction and assignment.

### Impact of number of prototypes

We evaluate the impact of the number of stream prototypes $K$. Explored previously by [8] on ImageNet [18], they found no significant impact on performance when varying the prototypes by several orders of magnitude using a sufficiently large amount of prototypes. In Table 3, we show results on varying the number of prototypes to $K = \{100, 1000\}$. We observe a slightly worse result for both settings for the RGB model and the R+F model. As we find no significant impact on the performance, our results are in line with previous work suggesting that the soft prototype mappings used for contrasting in ViCC are not necessarily a self-labeling approach similar to other pseudo-labeling approaches [8, 2, 20, 7, 76], despite the usefulness in contrasting for representation learning.
| Method   | Year | Dataset | Backbone | Param | Res | Frames | Modality | UCF101 | HMDB51 |
|----------|------|---------|----------|-------|-----|--------|----------|--------|--------|
| OPN      | 2017 | UCF101  | VGG      | 8.6M  | 80  | 16     | V        | -      | 59.8   |
| VCOP     | 2019 | UCF101  | R(2+1)D | 14.4M | 112 | 16     | V        | -      | 74.2   |
| Var. PSP | 2020 | UCF101  | R(2+1)D | 14.4M | 112 | 16     | V        | -      | 74.8   |
| Pace Pred | 2020 | UCF101  | R(2+1)D | 14.4M | 112 | 16     | V        | -      | 75.9   |
| VCP      | 2020 | UCF101  | R(2+1)D | 14.4M | 112 | 16     | V        | -      | 66.3   |
| PRP      | 2020 | UCF101  | R(2+1)D | 14.4M | 112 | 16     | V        | -      | 77.1   |
| RTT      | 2020 | UCF101  | R(2+1)D | 14.4M | 112 | 16     | V        | -      | 81.6   |
| Pace Pred | 2020 | K-400   | R(2+1)D | 14.4M | 112 | 16     | V        | -      | 77.1   |
| XDC      | 2020 | K-400   | R(2+1)D | 14.4M | 112 | 16     | V        | -      | 86.8   |
| SeLaVi   | 2020 | VGG-sound | R(2+1)D | 14.4M | 112 | 16     | V+A      | -      | 87.7   |
| GDT      | 2020 | Audioset | R(2+1)D | 14.4M | 224 | 32     | V+A      | -      | 92.5   |
| ViCC-RGB | (ours) | UCF101 | R(2+1)D | 14.4M | 128 | 16     | V        | -      | 82.8   |
| ViCC-R+F | (ours) | UCF101 | R(2+1)D | 14.4M | 128 | 16     | V        | -      | 88.8   |
| Pace Pred | 2020 | UCF101  | S3D-G    | 9.6M  | 224 | 64     | V        | -      | 87.1   |
| CoCLR    | 2020 | UCF101  | S3D      | 8.8M  | 128 | 32     | V        | -      | 81.4   |
| CoCLR †  | 2020 | UCF101  | S3D      | 8.8M  | 128 | 32     | V        | -      | 87.3   |
| SpeedNet | 2020 | K-400   | S3D-G    | 8.8M  | 128 | 32     | V        | -      | 90.6   |
| MIL-NCE  | 2020 | HTM     | S3D      | 8.8M  | 128 | 32     | V+T      | -      | 91.3   |
| CBT      | 2019 | K-600   | S3D      | 8.8M  | 128 | 16     | V+T      | -      | 79.5   |
| ViCC-RGB | (ours) | UCF101 | S3D      | 8.8M  | 128 | 32     | V        | -      | 84.3   |
| ViCC-R+F | (ours) | UCF101 | S3D      | 8.8M  | 128 | 32     | V        | -      | 89.0   |

Table 4. Comparison with prior self-supervised works on video action recognition on UCF101 and HMDB51 for finetuning and linear probe. We report Top-1 accuracy and compare with self-supervision pretraining on UCF101. In grey color we show larger pretraining datasets such as K-400 and multi-modal datasets (where T is text, A is audio).

multi-modal approaches on end-to-end finetuning. First, we significantly outperform previous approaches pretrained on UCF101 when considering the visual modality (V). On the S3D backbone, our R+F model (obtained by averaging RGB and flow predictions) achieves a Top-1 accuracy of 90.5% on UCF101 and a Top-1 accuracy of 62.2% on HMDB51. Our approach outperforms the best model of Han et al. [27] by 3.2% on UCF101 and by 3.5% on HMDB51. We also achieve better performance than Pace Pred [68], which uses the S3D-G [74] backbone, on both UCF101 and HMDB51. Using the R(2+1)D backbone, we obtain a Top-1 accuracy of 82.8% on UCF101 and a Top-1 accuracy of 52.4% on HMDB51. When combining RGB and Flow predictions (R+F), we obtain 88.8% and 61.5% on the datasets respectively. We outperform VCOP [42], VCOP [75], PRP [77] by a wide margin for both our models. With the R+F model we obtain a 7.2% increase on UCF101 and a 15.1% increase over RTT [31], underlined in the table as the second-best result. ViCC models therefore consistently outperform previous works on both backbones and evaluation datasets, where optical flow provides only an optional performance boost. Comparing against visual-only information using larger training sets, we outperform methods that use Kinetics (K-400) pretraining on HMDB51, using UCF101 pretraining, such as Pace Pred [68] for R(2+1)D and SpeedNet [4] for S3D-G. We also perform better on HMDB51 than some multi-modal approaches that use text [46] and audio [3] for similar resolution, number of frames and backbone. Finally, comparing against methods on linear probe, we outperform CoCLR [27] on the same training dataset by a significant margin.

Nearest-neighbour retrieval. Next, we compare with self-supervised approaches on nearest-neighbour clip retrieval in Table 5. All methods are pretrained on UCF101. We also report results on R3D for a fairer comparison. Our ViCC approach outperforms all previous approaches by a significant margin on UCF101 and HMDB51 for both backbone networks R(2+1)D and S3D. Our R3D models outperforms previous methods with the same backbone significantly. We achieve a Top-1 Recall of 65.1% on UCF101 using the S3D backbone, outperforming the previous best by 9.2%. On HMDB51, we achieve a Top-1 Recall of 29.7%, which is a 8.8% increase on previous best. With the R(2+1)D backbone, we obtain a Top-1 Recall of 58.6% on UCF101 and 25.3% on HMDB51 for RGB, and 59.9% and 28.3% respectively for R+F. Compared to other self-supervised works apart from the second-best, the margins are significantly wider. We conclude that our cross-stream self-supervision model RGB learns useful motion features without needing optical flow during test time.

4.4. Nearest-neighbour retrieval

In Figure 4, we visualize query video clips from the UCF101 test set with its Top-3 nearest-neighbours from
### Table 5. Comparison with self-supervised methods on nearest-neighbour video retrieval.

All self-supervised methods are pretrained on UCF101 split 1. We show results on Top-k Recall (R@k) for $k = \{1, 5, 10, 20\}$ on UCF101 split 1 and HMDB51 split 1.

| Method     | Year | Backbone | Modality | UCF101 R@1 | UCF101 R@5 | UCF101 R@10 | UCF101 R@20 | HMDB51 R@1 | HMDB51 R@5 | HMDB51 R@10 | HMDB51 R@20 |
|------------|------|----------|----------|------------|------------|------------|------------|------------|------------|------------|------------|
| OPN 40     | 2017 | VGG      | V        | 19.9       | 28.7       | 34.0       | 40.6       | -          | -          | -          | -          |
| ST Order 5 | 2018 | CaffeNet | V        | 25.7       | 36.2       | 42.2       | 49.2       | -          | -          | -          | -          |
| ST-Puzzle 35 | 2019 | R3D      | V        | 19.7       | 28.5       | 33.5       | 40.0       | -          | -          | -          | -          |
| VCOP 75    | 2019 | R3D      | V        | 14.1       | 30.3       | 40.4       | 51.1       | 7.6        | 22.9       | 34.4       | 48.8       |
| Pace Pred | 69   | R3D      | V        | 23.8       | 36.2       | 42.2       | 49.2       | -          | -          | -          | -          |
| Var. PSP 14 | 2020 | R3D      | V        | 24.6       | 41.9       | 51.3       | 62.7       | -          | -          | -          | -          |
| RTT 31     | 2020 | R3D      | V        | 26.1       | 48.5       | 59.1       | 69.6       | -          | -          | -          | -          |
| ViCC-RGB (ours) | R3D | V | 50.3 | 70.9 | 78.7 | 85.6 | 22.7 | 46.2 | 60.9 | 74.1 |
| ViCC-R+F (ours) | R3D | V | 52.1 | 71.7 | 79.8 | 86.0 | 25.2 | 48.1 | 61.1 | 72.7 |
| MemDPC 26 | 2020 | R2D3D    | V        | 20.2       | 40.4       | 52.4       | 64.7       | 7.7        | 25.7       | 40.6       | 57.7       |
| VCP 43    | 2020 | R(2+1)D | V        | 19.9       | 33.7       | 42.0       | 50.5       | 6.7        | 21.3       | 32.7       | 49.2       |
| CoCLR 27  | 2020 | S3D      | V        | 55.9       | 70.8       | 76.9       | 82.5       | 26.1       | 45.8       | 57.9       | 69.7       |
| ViCC-RGB (ours) | R(2+1)D | V | 58.6 | 76.2 | 83.1 | 89.0 | 25.3 | 50.4 | 64.0 | 77.5 |
| ViCC-R+F (ours) | R(2+1)D | V | 59.9 | 77.6 | 84.6 | 90.6 | 28.3 | 52.7 | 65.3 | 77.0 |
| ViCC-RGB (ours) | S3D | V | 62.1 | 77.1 | 83.7 | 87.9 | 25.5 | 49.6 | 61.9 | 72.5 |
| ViCC-R+F (ours) | S3D | V | 65.1 | 80.2 | 85.4 | 89.8 | 29.7 | 54.6 | 66.0 | 76.2 |

**Figure 4. Nearest-neighbour retrieval results with our representations.** The query video from the UCF101 test set is shown on the left, the top-3 nearest neighbours from the UCF101 training set on the right. Each video is visualized with 2 frames and we show results for single-stream (RGB-1) and cross-stream (RGB-2). The action label is shown above the video (not used during training), where green denotes the correct label and red denotes an incorrect result. Best viewed in color.

In the UCF101 training set, retrieved using the ViCC representation without labels. The ground truth action labels are included above the video clips. We visualize results for single-stream (RGB-1) and cross-stream (RGB-2). Our qualitative results further support the benefit of cross-stream training, showing that it helps to retrieve videos from the same semantic categories compared to single-stream, despite significant changes in appearance and background (e.g., Swing and WalkingWithDog). More difficult is the retrieval for the query video from class BlowDryHair, but we again observe that cross-stream training improves retrieval.

**5. Conclusion**

In this paper, we present the Video Cross-Stream Proto-typical Contrasting (ViCC) framework for self-supervised representation learning. We demonstrate the advantages of using similar semantic groupings of RGB and flow views over methods that use instance-level contrastive learning, avoiding redundant comparisons and improving performance. By learning through predicting consistent prototype assignments from views originating from both streams, ViCC effectively transfers knowledge from the motion representation to appearance and vice versa. We demonstrate state-of-the-art performance on downstream video recognition tasks using visual-only self-supervision.

**Acknowledgments.** We would like to thank Prof. dr. Cees G.M. Snoek for the helpful comments and feedback. The presentation of this paper at the conference was financially supported by the Amsterdam ELLIS Unit, Qualcomm and the Master AI program of the University of Amsterdam.
References

[1] Humam Alwassel, Dhruv Mahajan, Bruno Korbar, Lorenzo Torresani, Bernard Ghanem, and Du Tran. Self-Supervised Learning by Cross-Modal Audio-Video Clustering. In NeurIPS, 2020.

[2] Yuki M. Asano, Mandela Patrick, Christian Rupprecht, and Andrea Vedaldi. Labelling unlabelled videos from scratch with multi-modal self-supervision. In NeurIPS, 2020.

[3] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. In ICLR, 2020.

[4] Sagie Benaim, Ariel Ephrat, Oran Lang, Inbar Mosseri, William T. Freeman, Michael Rubinstein, Michal Irani, and Tali Dekel. SpeedNet: Learning the Speediness in Videos. In CVPR, 2020.

[5] Uta Büchler, Biagio Brattoli, and Björn Ommer. Improving Spatiotemporal Self-Supervision by Deep Reinforcement Learning. In ECCV, 2018.

[6] Yue Cao, Zhenda Xie, Bin Liu, Yutong Lin, Zheng Zhang, and Han Hu. Parametric Instance Classification for Unsupervised Visual Feature Learning. In NeurIPS, 2020.

[7] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep Clustering for Unsupervised Learning of Visual Features. In ECCV, 2018.

[8] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised Learning of Visual Features by Contrastive Cluster Assignments. In NeurIPS, 2020.

[9] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A Short Note about Kinetics-600. ArXiv preprint arXiv:1808.01340, 2018.

[10] Joao Carreira and Andrew Zisserman. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In CVPR, 2017.

[11] Hongjie Chen, Weidi Xie, Andrea Vedaldi, and Andrew Zisserman. Vggsound: A Large-Scale Audio-Visual Dataset. In ICASSP, 2020.

[12] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In ICML, 2020.

[13] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved Baselines with Momentum Contrastive Learning. ArXiv preprint arXiv:2003.04297, 2020.

[14] H. Cho, Tae-Hoon Kim, H. J. Chang, and Wonjun Hwang. Self-Supervised Spatiotemporal Representation Learning Using Variable Playback Speed Prediction. ArXiv preprint arXiv:2003.02692, 2020.

[15] Ching-Yao Chuang, Joshua Robinson, Lin Yen-Chen, Antonio Torralba, and Stefanie Jegelka. Debiasied Contrastive Learning. In NeurIPS, 2020.

[16] Nieves Crasto, Philippe Weinzaepfel, Kartek Alahari, and Cordelia Schmid. MARS: Motion-Augmented RGB Stream for Action Recognition. In CVPR, 2019.

[17] Marco Cuturi. Sinkhorn Distances: Lightspeed Computation of Optimal Transportation Distances. In NeurIPS, 2013.

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In CVPR, 2009.

[19] Basura Fernando, Hakan Bilen, Efstratios Gavves, and Stephen Gould. Self-Supervised Video Representation Learning With Odd-One-Out Networks. In CVPR, 2017.

[20] Kirill Gavrilyuk, Mihir Jain, Ilia Karmanov, and Cees G. M. Snoek. Motion-Augmented Self-Training for Video Recognition at Smaller Scale. ArXiv preprint arXiv:2105.01646, 2021.

[21] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio Set: An ontology and human-labeled dataset for audio events. In ICASSP, 2017.

[22] Melvyn A Goodale and A. David Milner. Separate visual pathways for perception and action. Trends in Neurosciences, 15:20–25, 1992.

[23] Ross Goroshin, Joan Bruna, Jonathan Tompson, David Eigen, and Yann LeCun. Unsupervised Learning of Spatiotemporally Coherent Metrics. In ICCV, 2015.

[24] Michael U Gutmann and Aapo Hyvarinen. Noise-Contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics. Journal of Machine Learning Research, 13(11):307–361, 2012.

[25] Tengda Han, Weidi Xie, and Andrew Zisserman. Video Representation Learning by Dense Predictive Coding. In ICCV Workshop, 2019.

[26] Tengda Han, Weidi Xie, and Andrew Zisserman. Memory-augmented Dense Predictive Coding for Video Representation Learning. In ECCV, 2020.

[27] Tengda Han, Weidi Xie, and Andrew Zisserman. Self-supervised Co-training for Video Representation Learning. In NeurIPS, 2020.

[28] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can Spatiotemporal 3D CNNs Retrace the History of 2D CNNs and ImageNet? In CVPR, 2018.

[29] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. In CVPR, 2020.

[30] R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karon Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In NeurIPS, 2019.

[31] S. Jenni, Givi Meishvili, and P. Favaro. Video Representation Learning by Recognizing Temporal Transformations. In ECCV, 2020.

[32] Longlong Jing, Xiaodong Yang, Jingen Liu, and Yingli Tian. Video Representations That Work for Action Recognition? A New Model and the Kinetics Dataset. In NeurIPS, 2017.

[33] Gunnar Johansson. Visual perception of biological motion and a model for its analysis. Perception & psychophysics, 14(2):201–211, 1973.

[34] Yannis Kalantidis, Mert Bulent Sariyildiz, and Noe Pion. Hard Negative Mixing for Contrastive Learning. In NeurIPS, 2020.
[72] Donglai Wei, Joseph J Lim, and Andrew Zisserman. Learning and Using the Arrow of Time. In CVPR, 2018.

[73] Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. Unsupervised Feature Learning via Non-Parametric Instance-level Discrimination. In CVPR, 2018.

[74] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking Spatiotemporal Feature Learning: Speed-Accuracy Trade-offs in Video Classification. In ECCV, 2018.

[75] Dejing Xu, Jun Xiao, Zhou Zhao, Jian Shao, Di Xie, and Yueting Zhuang. Self-Supervised Spatiotemporal Learning via Video Clip Order Prediction. In CVPR, 2019.

[76] Xueting Yan, Ishan Misra, Abhinav Gupta, Deepti Ghadiyaram, and Dhruv Mahajan. ClusterFit: Improving Generalization of Visual Representations. In CVPR, 2020.

[77] Yuan Yao, Chang Liu, Dezhang Luo, Yu Zhou, and Qixiang Ye. Video Playback Rate Perception for Self-supervised Spatio-Temporal Representation Learning. In CVPR, 2020.

[78] Yang You, Igor Gitman, and Boris Ginsburg. Large Batch Training of Convolutional Networks. ArXiv preprint arXiv:1708.03888, 2017.

[79] C. Zach, T. Pock, and H. Bischof. A Duality Based Approach for Realtime TV-L1 Optical Flow. In DAGM-Symposium, 2007.

[80] Xiaohang Zhan, Jiahao Xie, Ziwei Liu, Yew Soon Ong, and Chen Change Loy. Online Deep Clustering for Unsupervised Representation Learning. In CVPR, 2020.

[81] Jiaojiao Zhao and Cees G. M. Snoek. Dance with Flow: Two-in-One Stream Action Detection. In CVPR, 2019.

[82] Chengxu Zhuang, Tianwei She, Alex Andonian, Max Sobol Mark, and Daniel Yamins. Unsupervised Learning from Video with Deep Neural Embeddings. In ArXiv Preprint arXiv:1905.11954, 2020.
Supplementary Material for
Self-supervised Video Representation Learning with Cross-Stream Prototypical Contrasting

A. Example code for ViCC

Here, we provide pseudocode in PyTorch-like style for the implementation of the cross-stream stage of ViCC-RGB. For the definition of the function \texttt{sinkhorn} that describes the Sinkhorn-Knopp algorithm we refer to [8].

Pseudocode for ViCC-RGB-2 in PyTorch-like style

```python
# rgb_model: encoder network for RGB
# flow_model: encoder network for flow, frozen
# temp: temperature
for rgb, flow in loader: # B samples
    rgb_i, flow_i = aug(rgb_i, flow_i)
    rgb_j, flow_j = aug(rgb_j, flow_j)
    # get RGB and flow embeddings: 2B x D
    z_rgb = cat(rgb_model(rgb_i), rgb_model(rgb_j))
    z_flow = cat(flow_model(flow_i), flow_model(flow_j))
    # get similarity with prototypes C_rgb, C_rgb in D x K
    sim_rgb_i, sim_rgb_j = mm(z_rgb, C_rgb)
    sim_flow_i, sim_flow_j = mm(z_flow, C_rgb)
    # compute assignments
    with torch.no_grad():
        q_rgb_i, q_rgb_j, q_flow_i, q_flow_j =
            sinkhorn(sim_rgb_i), sinkhorn(sim_rgb_j),
            sinkhorn(sim_flow_i), sinkhorn(sim_flow_j)
    # convert similarity scores to probabilities
    p_rgb_i, p_rgb_j, p_flow_i, p_flow_j =
        softmax(sim_rgb_i / temp), softmax(sim_rgb_j / temp),
        softmax(sim_flow_i / temp), softmax(sim_flow_j / temp)
    # predict cluster assignments using three other views
    l_rgb_i = q_rgb_i * log(p_rgb_i) + q_rgb_j * log(p_rgb_j)
    l_rgb_j = q_rgb_i * log(p_rgb_i) + q_rgb_j * log(p_rgb_j)
    l_flow_i = q_flow_i * log(p_flow_i) + q_flow_j * log(p_flow_j)
    l_flow_j = q_flow_i * log(p_flow_i) + q_flow_j * log(p_flow_j)
    # combine for total loss for rgb model
    loss = -\frac{1}{4} \times (l_{rgb_i} + l_{rgb_j} + l_{flow_i} + l_{flow_j})
    # optimizer update and normalize prototypes
    optimizer.update(rgb_model.params), update(C_rgb)
    with torch.no_grad():
        C_rgb = normalize(C_rgb, dim=0, p=2)
```

B. Implementation Details

B.1. Implementation and Training

SGD with LARS [25] is used as the optimizer. A learning rate of 0.6, a weight decay of $10^{-6}$ and a cosine learning rate schedule with a final learning rate of $6 \times 10^{-4}$ are chosen. The temperature $T$ is set to 0.1, the Sinkhorn regularization parameter $\epsilon$ is set to 0.05 and we perform 3 iterations of the Sinkhorn-Knopp algorithm. We use batch shuffle [29] to avoid the model exploiting local intra-batch information leakage for trivial solutions. For single-stream, the prototypes are frozen during the first 100 epochs of training. For cross-stream, the prototypes are directly updated from the start of the training.

B.2. Queue

To store additional features for use in the assignment to prototypes, we employ a queue in line with [8]. With 4 GPUs and a total batch size of $48 \times 4 = 192$, we adopt a queue of size 1920 to store features from the last 10 batches. The queue is introduced when the evolution of features is slowing down, i.e. when the decrease of the loss function is moderate. For single-stream RGB (RGB-1) we introduce the queue at 150 epochs and for Flow-1 we introduce the queue at 200 epochs. For the cross-stream stage, we introduce the queue at 25 epochs in each alternation.

C. Additional results

C.1. Analysis of Prototypes

This section focuses on further analysis of the prototypes. The main purpose of the prototype sets in ViCC is to guide the contrasting of groups of views from streams in each iteration. In combination with the relatively stable performance observed when varying the number of prototypes, we conjecture that the prototypes are not a pseudo-labeling approach similar to other methods [3, 2, 20, 7, 76]. Despite this intuition and our use of soft assignments, we investigate the prototypes by visualizing video samples assigned to the same prototypes when rounding the assignments. We also evaluate the rounded prototype assignments from several of our self-supervised stages on standard cluster evaluation metrics.

C.1.1 Visualization of Prototypes

In Figure 5 we show the hard assignment of video samples to random prototypes. Video samples with the highest similarity scores to the prototype clusters are visualized. Prototype scores are indicated on the samples and the ground truth class labels of the samples are indicated below the groups. We can observe that video samples assigned to the same prototypes share semantic similarity and even belong to the same action class, despite the fact that class labels are not used during ViCC training. The prototypes seem effective at grouping together views from the same semantic class label, as the samples visualized are all from the same class. These semantically similar sets in ViCC thereby provide an advantage for video representation learning over methods that use contrastive instance learning.
Figure 5. **Visualization of rounded assignments to random ViCC prototypes** using videos from UCF101. Samples with high similarity scores (visualized on the samples) to the prototypes are shown. The ground truth labels of all the video samples are included below (not used during training).

| Method       | Acc  | NMI  | ARI  | Entropy | Max Purity |
|--------------|------|------|------|---------|------------|
| ViCC-RGB-1   | 32.3 | 62.5 | 16.4 | 1.6     | 36.8       |
| ViCC-Flow-1  | 34.4 | 63.1 | 17.6 | 1.5     | 39.1       |
| ViCC-RGB-2   | 40.8 | 67.8 | 24.5 | 1.4     | 45.1       |
| ViCC-Flow-2  | 40.3 | 67.0 | 23.5 | 1.4     | 45.3       |

Table 6. **Cluster evaluation of ViCC prototypes** when rounding the assignments evaluated on the UCF101 test set.

C.1.2 Cluster evaluation

In this section, we evaluate the hard assignment of our prototype sets with standard cluster evaluation measures as done in [7, 8]. Although the ground truth number of clusters is not known in advance for self-supervised training, we set the number of prototypes to $K=101$ for evaluation purposes only to match the number of class labels for UCF101. The Hungarian algorithm [37] is then used to match self-supervised labels to the ground truth labels to obtain accuracy (Acc). We also report the Normalized Mutual Information (NMI), Adjusted Rand Index (ARI), mean entropy per cluster (where the optimal number is 0) and mean maximal purity per cluster as defined in [3]. For example, the NMI ranges from 0 (no mutual information) to 100% (implying perfect correlation between self-supervised labels and the ground truth labels). Table 6 shows that our prototypes from the cross-stream stage (RGB-2 and Flow-2) obtain better performance on all measures compared to prototypes learned only on their own stream (RGB-1 and Flow-1), achieving e.g. a higher NMI, lower mean entropy per cluster and higher mean maximal purity.

C.2. T-SNE Visualization

In this section, we visualize ViCC representations of the UCF101 test set using the t-SNE clustering algorithm [66] to project features to 2D. For clarity, only 10 random classes are visualized with a limited amount of random features for each class. Figure 6 shows the t-SNE visualization of features extracted from single-stream (RGB-1)
and cross-stream (RGB-2) trained using the same number of epochs (500). It can be observed that the inter-class distance between certain classes such as CricketBowling and GolfSwing is increased from RGB-1 to RGB-2. Moreover, the intra-class distance is reduced for classes FrisbeeCatch, BasketballDunk and ApplyEyeMakeup, which can be attributed to the benefit of motion learning from the flow encoder in cross-stream.

### C.3. Impact of queue size

We investigate the effect of the queue size on performance. The queue is used in the assignment of features to $K$ prototypes. In theory, using more features in each iteration on top of the current batch should result in a more accurate assignment for the Sinkhorn-Knopp algorithm. Results for queue sizes $\{3840, 1920, 0\}$ are shown in Table 7. We report Top-1 accuracy on action recognition finetuning on UCF101. For queue size 3840, we observe that the larger queue size is not necessary or beneficial for UCF101 self-supervised pretraining, as the differences in performance are minimal. We also find that using no queue almost performs on par with our default queue size of 1920. We conjecture that our mini-batches may already provide enough features for ViCC self-supervision on UCF101.

### C.4. More comparison with self-supervised works on action recognition

In Table 8 we list more results from self-supervised methods evaluated on action recognition. Results for the additional backbone R3D-18 [28] are included. We achieve better performance than several methods that use the R3D backbone. Our overall best result on the S3D backbone still outperforms almost all methods pretrained on UCF101. We also outperform several methods pretrained on the larger dataset K-400, and achieve competitive performance compared to CVRL [55].

| Method     | Queue size | Top-1 Accuracy |
|------------|------------|----------------|
|            | 3840       | 84.5           |
|            | 1920       | 84.3           |
|            | 0          | 84.7           |
| ViCC-RGB-2 |            |                |
|            | 3840       | 90.4           |
|            | 1920       | 90.5           |
|            | 0          | 90.2           |

Table 7. **Impact of queue size.** We report Top-1 accuracy on action recognition finetuning on UCF101.
| Method     | Year | Dataset   | Backbone  | Param | Res | Frames | Modality | Finetune | Linear | Pretrain stage |
|------------|------|-----------|-----------|-------|-----|--------|----------|----------|--------|----------------|
| OPN        | 2017 | UCF101    | VGG       | 8.6M  | 80  | 16     | V         | -        | -      | 59.8           |
| VCOP       | 2019 | UCF101    | R(2+1)D  | 14.4M | 112 | 16     | V         | -        | -      | 72.4           |
| Var. PSP   | 2020 | UCF101    | R(2+1)D  | 14.4M | 112 | 16     | V         | -        | -      | 74.8           |
| Pace Pred  |       |           |           |       |     |        |           |          |        | 75.9           |
| VCP        | 2020 | UCF101    | R(2+1)D  | 14.4M | 112 | 16     | V         | -        | -      | 66.3           |
| PRP        | 2020 | UCF101    | R(2+1)D  | 14.4M | 112 | 16     | V         | -        | -      | 72.1           |
| RTT        | 2020 | UCF101    | R(2+1)D  | 14.4M | 112 | 16     | V         | -        | -      | 81.6           |
| Pace Pred  | 2020 | K-400     | R(2+1)D  | 14.4M | 112 | 16     | V         | -        | -      | 74.8           |
| MotionFit  | 2021 | K-400     | R(2+1)D  | 14.4M | 112 | 32     | V         | -        | -      | 88.9           |
| XDC        | 2020 | K-400     | R(2+1)D  | 14.4M | 224 | 32     | V+A       | -        | -      | 90.7           |
| ScLaVi     | 2020 | VGG-sound | R(2+1)D  | 14.4M | 112 | 30     | V+A       | -        | -      | 91.3           |
| GDT        | 2020 | AudioSet  | R(2+1)D  | 14.4M | 224 | 32     | V+A       | -        | -      | 92.5           |
| ViCC-RGB   | 2019 | UCF101    | R(2+1)D  | 14.4M | 128 | 16     | V         | -        | -      | 74.4           |
| ViCC-R+F   | 2019 | UCF101    | R(2+1)D  | 14.4M | 128 | 16     | V         | -        | -      | 78.3           |
| MemDPC     | 2020 | UCF101    | R2D3D    | 14.2M | 224 | 40     | V         | -        | -      | 60.6           |
| VCOP       | 2019 | UCF101    | R3D      | 14.2M | 112 | 16     | V         | -        | -      | 64.9           |
| Var. PSP   | 2020 | UCF101    | R3D      | 14.2M | 112 | 16     | V         | -        | -      | 69.0           |
| VCP        | 2020 | UCF101    | R3D      | 14.2M | 112 | 16     | V         | -        | -      | 66.0           |
| PRP        | 2020 | UCF101    | R3D      | 14.2M | 112 | 16     | V         | -        | -      | 66.5           |
| RTT        | 2020 | UCF101    | R3D      | 14.2M | 112 | 16     | V         | -        | -      | 77.3           |
| RotNet3D   | 2019 | K-400     | R3D      | 33.6M | 224 | 16     | V         | -        | -      | 62.9           |
| ST-Puzzle  | 2019 | K-400     | R3D      | 33.6M | 224 | 16     | V         | -        | -      | 65.8           |
| DPC        | 2019 | K-400     | R3D      | 14.2M | 128 | 40     | V         | -        | -      | 68.2           |
| VIE        | 2020 | K-400     | R3D      | 14.2M | 112 | 40     | V         | -        | -      | 72.3           |
| CVRL        | 2021 | K-400     | R3D-50   | 36.1M | 224 | 16     | V         | -        | -      | 92.1           |
| Pace Pred  | 2020 | UCF101    | R3D      | 14.2M | 128 | 16     | V         | -        | -      | 69.0           |
| CoCLR†      | 2020 | UCF101    | S3D-G    | 9.6M  | 224 | 64     | V         | -        | -      | 87.1           |
| CoCLR       | 2020 | UCF101    | S3D      | 8.8M  | 128 | 32     | V         | 70.2    | 39.1  | 81.4           |
| SpeedNet   | 2020 | K-400     | S3D-G    | 8.8M  | 128 | 32     | V         | 72.1    | 40.2  | 87.3           |
| MIL-NCE    | 2020 | HTM       | S3D      | 8.8M  | 224 | 32     | V+T       | 82.7    | 53.1  | 91.3           |
| CBT         | 2019 | K-600     | S3D      | 8.8M  | 112 | 16     | V+T       | 54.0    | 29.5  | 79.5           |
| ELo         | 2020 | K-400     | S3D      | 8.8M  | 224 | 32     | V+T       | -       | -     | 93.8           |
| ViCC-RGB   | 2019 | UCF101    | S3D      | 8.8M  | 128 | 32     | V         | -       | -     | 72.2           |
| ViCC-R+F   | 2019 | UCF101    | S3D      | 8.8M  | 128 | 32     | V         | -       | -     | 78.0           |

Table 8. **Comparison with prior self-supervised works on video action recognition** on UCF101 and HMDB51 for finetuning and linear probe. We report Top-1 accuracy, compare with self-supervision pretraining on UCF101 and additionally report results on backbone R3D [28]. In grey color we show larger pretraining datasets such as K-400 [10] and multi-modal datasets (where T is text, A is audio).