1. 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 sinkhorn that describes the Sinkhorn-Knopp algorithm we refer to [6].

```python
# Pseudocode for ViCC-RGB-2 in PyTorch-like style
# rgb_model: encoder network for RGB
# flow_model: encoder network for flow, frozen
# temp: temperature
for rgb, flow in loader: # B samples
    z_rgb_i, flow_i = aug(rgb_i, flow_i)
    z_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_i, C_rgb), mm(z_rgb_j, C_rgb)
    sim_flow_i, sim_flow_j = mm(z_flow_i, C_rgb), mm(z_flow_j, 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))
    # 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_j * log(p_rgb_j) + q_rgb_i * log(p_rgb_i)
    l_flow_i = q_flow_i * log(p_flow_i) + q_flow_j * log(p_flow_j)
    l_flow_j = q_flow_j * log(p_flow_j) + q_flow_i * log(p_flow_i)
    # combine for total loss for rgb model
    loss = -1/4 * (1/3 * l_rgb_i + 1/3 * l_rgb_j +
                    1/3 * l_flow_i + 1/3 * 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)
```

2. Implementation Details

2.1. Implementation and Training

SGD with LARS [35] 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 $\tau$ 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 [17] to avoid the model expoting 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.

2.2. Queue

To store additional features in use for the assignment to prototypes, we employ a queue in line with [6]. 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.

3. Additional results

3.1. 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, 11, 5, 33]. 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.

3.1.1 Visualization of Prototypes

In Figure 1 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 1. **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).

<table>
<thead>
<tr>
<th>Method</th>
<th>Acc</th>
<th>NMI</th>
<th>ARI</th>
<th>Entropy</th>
<th>Max Purity</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViCC-RGB-1</td>
<td>32.3</td>
<td>62.5</td>
<td>16.4</td>
<td>1.6</td>
<td>36.8</td>
</tr>
<tr>
<td>ViCC-Flow-1</td>
<td>34.4</td>
<td>63.1</td>
<td>17.6</td>
<td>1.5</td>
<td>39.1</td>
</tr>
<tr>
<td>ViCC-RGB-2</td>
<td>40.8</td>
<td>67.8</td>
<td>24.5</td>
<td>1.4</td>
<td>45.1</td>
</tr>
<tr>
<td>ViCC-Flow-2</td>
<td>40.3</td>
<td>67.0</td>
<td>23.5</td>
<td>1.4</td>
<td>45.3</td>
</tr>
</tbody>
</table>

Table 1. **Cluster evaluation of ViCC prototypes** when rounding the assignments evaluated on the UCF101 test set.

### 3.1.2 Cluster evaluation

In this section, we evaluate the hard assignment of our prototype sets with standard cluster evaluation measures as done in [5, 3]. 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 [21] 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 1 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.

### 3.2. T-SNE Visualization

In this section, we visualize ViCC representations of the UCF101 test set using the t-SNE clustering algorithm [30] to project features to 2D. For clarity, only 10 random action classes are visualized with a limited amount of random features for each class. Figure 2 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.

3.3. 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 2. We report Top-1 accuracy on action recognition on UCF101 finetuning. 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.

3.4. More comparison with self-supervised works on action recognition

In Table 3 we list more results from self-supervised methods evaluated on action recognition. Results for the additional backbone R3D-18 \[16\] 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. \[23\].

<table>
<thead>
<tr>
<th>Method</th>
<th>Queue size</th>
<th>3840</th>
<th>1920</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViCC-RGB-2</td>
<td></td>
<td>84.5</td>
<td>84.3</td>
<td>84.7</td>
</tr>
<tr>
<td>ViCC-R+F-2</td>
<td></td>
<td>90.4</td>
<td>90.5</td>
<td>90.2</td>
</tr>
</tbody>
</table>

Table 2. Impact of queue size. We report Top-1 accuracy on action recognition finetuning on UCF101.

References


<table>
<thead>
<tr>
<th>Method</th>
<th>Year</th>
<th>Dataset</th>
<th>Backbone</th>
<th>Param</th>
<th>Res</th>
<th>Frames</th>
<th>Modality</th>
<th>UCF101</th>
<th>HMDB51</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPN [22]</td>
<td>2017</td>
<td>UCF101</td>
<td>VGG</td>
<td>8.6M</td>
<td>80</td>
<td>16</td>
<td>V</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VCOP [22]</td>
<td>2019</td>
<td>UCF101</td>
<td>R(2+1)D</td>
<td>14.4M</td>
<td>112</td>
<td>16</td>
<td>V</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Var. PSP [10]</td>
<td>2020</td>
<td>UCF101</td>
<td>R(2+1)D</td>
<td>14.4M</td>
<td>112</td>
<td>16</td>
<td>V</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Pace Pred [51]</td>
<td>2020</td>
<td>UCF101</td>
<td>R(2+1)D</td>
<td>14.4M</td>
<td>112</td>
<td>16</td>
<td>V</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VCP [23]</td>
<td>2020</td>
<td>UCF101</td>
<td>R(2+1)D</td>
<td>14.4M</td>
<td>112</td>
<td>16</td>
<td>V</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PRP [44]</td>
<td>2020</td>
<td>UCF101</td>
<td>R(2+1)D</td>
<td>14.4M</td>
<td>112</td>
<td>16</td>
<td>V</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RTT [18]</td>
<td>2020</td>
<td>UCF101</td>
<td>R(2+1)D</td>
<td>14.4M</td>
<td>112</td>
<td>16</td>
<td>V</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Pace Pred [33]</td>
<td>2020</td>
<td>K-400</td>
<td>R(2+1)D</td>
<td>14.4M</td>
<td>112</td>
<td>16</td>
<td>V</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MotionFit [14]</td>
<td>2021</td>
<td>K-400</td>
<td>R(2+1)D</td>
<td>14.4M</td>
<td>112</td>
<td>32</td>
<td>V</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SeLaVi [2]</td>
<td>2020</td>
<td>VGG-sound</td>
<td>R(2+1)D</td>
<td>14.4M</td>
<td>112</td>
<td>30</td>
<td>V+A</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GDT [2]</td>
<td>2020</td>
<td>Audioset</td>
<td>R(2+1)D</td>
<td>14.4M</td>
<td>112</td>
<td>32</td>
<td>V+A</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[ViCC-RGB (ours)]</td>
<td>UCF101</td>
<td>R(2+1)D</td>
<td>14.4M</td>
<td>128</td>
<td>16</td>
<td>16</td>
<td>V</td>
<td>74.4</td>
<td>30.8</td>
</tr>
<tr>
<td>[ViCC-R+F (ours)]</td>
<td>UCF101</td>
<td>R(2+1)D</td>
<td>14.4M</td>
<td>128</td>
<td>16</td>
<td>16</td>
<td>V</td>
<td>78.3</td>
<td>45.2</td>
</tr>
<tr>
<td>[DPC [13]]</td>
<td>2019</td>
<td>UCF101</td>
<td>R2D3D</td>
<td>14.2M</td>
<td>128</td>
<td>40</td>
<td>V</td>
<td>60.6</td>
<td>-</td>
</tr>
<tr>
<td>MemDPC [14]</td>
<td>2020</td>
<td>UCF101</td>
<td>R2D3D</td>
<td>14.2M</td>
<td>224</td>
<td>40</td>
<td>V</td>
<td>84.3</td>
<td>-</td>
</tr>
<tr>
<td>VCOP [32]</td>
<td>2019</td>
<td>UCF101</td>
<td>R3D</td>
<td>14.2M</td>
<td>112</td>
<td>16</td>
<td>V</td>
<td>64.9</td>
<td>29.5</td>
</tr>
<tr>
<td>Var. PSP [10]</td>
<td>2020</td>
<td>UCF101</td>
<td>R3D</td>
<td>14.2M</td>
<td>112</td>
<td>16</td>
<td>V</td>
<td>69.0</td>
<td>33.7</td>
</tr>
<tr>
<td>VCP [23]</td>
<td>2020</td>
<td>UCF101</td>
<td>R3D</td>
<td>14.2M</td>
<td>112</td>
<td>16</td>
<td>V</td>
<td>66.0</td>
<td>31.5</td>
</tr>
<tr>
<td>PRP [44]</td>
<td>2020</td>
<td>UCF101</td>
<td>R3D</td>
<td>14.2M</td>
<td>112</td>
<td>16</td>
<td>V</td>
<td>66.5</td>
<td>29.7</td>
</tr>
<tr>
<td>RTT [18]</td>
<td>2020</td>
<td>UCF101</td>
<td>R3D</td>
<td>14.2M</td>
<td>112</td>
<td>16</td>
<td>V</td>
<td>77.3</td>
<td>47.5</td>
</tr>
<tr>
<td>ST-Puzzle [20]</td>
<td>2019</td>
<td>K-400</td>
<td>R3D</td>
<td>14.2M</td>
<td>128</td>
<td>40</td>
<td>V</td>
<td>68.2</td>
<td>34.5</td>
</tr>
<tr>
<td>DPC [13]</td>
<td>2019</td>
<td>K-400</td>
<td>R3D</td>
<td>14.2M</td>
<td>128</td>
<td>40</td>
<td>V</td>
<td>72.3</td>
<td>44.8</td>
</tr>
<tr>
<td>VIE [36]</td>
<td>2020</td>
<td>K-400</td>
<td>R3D</td>
<td>14.2M</td>
<td>128</td>
<td>40</td>
<td>V</td>
<td>72.8</td>
<td>44.7</td>
</tr>
<tr>
<td>CVRL [28]</td>
<td>2021</td>
<td>K-400</td>
<td>R3D</td>
<td>14.2M</td>
<td>128</td>
<td>40</td>
<td>V</td>
<td>84.3</td>
<td>47.9</td>
</tr>
<tr>
<td>[ViCC-RGB (ours)]</td>
<td>UCF101</td>
<td>R3D</td>
<td>14.2M</td>
<td>128</td>
<td>16</td>
<td>32</td>
<td>V</td>
<td>69.0</td>
<td>44.2</td>
</tr>
<tr>
<td>[ViCC-R+F (ours)]</td>
<td>UCF101</td>
<td>R3D</td>
<td>14.2M</td>
<td>128</td>
<td>16</td>
<td>32</td>
<td>V</td>
<td>73.3</td>
<td>46.7</td>
</tr>
</tbody>
</table>

Table 3. 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 [16]. In grey color we show larger pretraining datasets such as K-400 [8] and multi-modal datasets (where T is text, A is audio).


