7. Supplementary Material

7.1. Implementation

Our implementation as well as additional information is available under https://github.com/simplexsigil/p-hlvc.

7.2. Datasets

Kinetics-400 Kinetics-400 is provided as a list of downloadable online videos. For these experiments, we were able to download 208684 videos for the train set (94.1 % of a total of 221618 listed videos), 17168 videos for the val set (94.3 % of a total of 18205 listed videos) and 33621 videos for the test set (94.1 % of a total of 35727 listed videos). The total downloaded dataset size is 260013 videos.

Kinetics-Skeleton Kinetics-Skeleton does not provide body poses for all videos in Kinetics-400. We filter videos, which do not contain paired skeleton sequences or do not have a sufficient length. After filtering, we remain with 144085 videos (55.4%) which we use for training. The poses of Kinetics-Skeleton use the COCO-18-Keypoints pose format, extracted with OpenPose [9].

Sims4Action The Sims4Action dataset [58] was recently published and consists of roughly ten hours of scripted Sims4 gameplay. Ten action classes were specifically chosen to correspond to a subset of the 31 action classes in the Toyota Smarthome [17] dataset. While not every class does have a one-to-one correspondence, [58] do publish a mapping scheme, which for example maps different cooking action in Toyota Smarthome to the general action *Cook* in Sims4Action. In order to perform multi-modal training on Sims4Action, we extract body poses in COCO-18-Keypoint format with Alphapose [23, 42, 69], this works well despite the synthetic appearance of the action videos. Sims4Action is a balanced dataset and provides the same amount of training data for each class.

Other synthetic action recognition datasets While there are other synthetic datasets for action recognition, Sims4Action is the only public dataset providing a domain generalization benchmark, to the best of our knowledge. Other synthetic datasets are used by combining the synthetic data with real world data as a form of data augmentation (for example [29]). The only exception would be Kinetics-Gameplay [13], but this dataset is not publicly available. The authors offer extracted ResNet-101 features upon request, but not the source video material which would be necessary to apply our approach.

Toyota Smarthome Toyota Smarthome [17] is a dataset which provides unscripted recordings of 31 classes performed by 18 different subjects. Different camera views and sceneries within a single appartment are displayed. Since the dataset consists of unscripted recordings, a strong class imbalance exists whith some classes being present in significantly more training samples than others. Although Toyota Smarthome does provide body poses, we extract them a second time using Alphapose in order to have consistent pose representations on all three pose datasets; Kinetics-Skeleton, Sims4Action and Toyota Smarthome.

7.3. Model architecture

In Figure 11 and Listing 1, we provide a detailed overview of our pre-training network. The S3D contains a convolutional classification layer, we append two more fully connected layers which maintain an intermediate hidden width (512 for all our experiments) and a final representation dimensionality (256 for all our experiments). For our SkeleMotion calculations, the first two fully connected layers are both part of the backbone network, we append another fully-connected layer to map to the representation dimensionality. Note, that an additional set of parameters is maintained for the prototypes (not displayed in Listing 1). For downstream action recognition, we append classification layers after FC2, discarding the representation mapping and for action retrieval we use the features before FC2. Unless noted otherwise, the downstream classification is performed with a single linear layer, preceded by dropout and a ReLU activation function. For multi-modal training, we concatenate the feature vectors of the individual network branches before adding the classification layer. Downstream training is optimized with the cross-entropy loss.

7.4. Working with body poses

It is a major advantage of our approach that we are robust against noisy data as is the case with Kinetics-Skeleton. Since our contrastive learning technique operates on video clips of roughly one second of length, we only need consistent pose-track-IDs in that time frame. We do not rely on person IDs which are consistent for the full 10 second video. It is possible that a body pose disappears within a one second clip, in that case we simply fill up that part of the data with zero values. In training we calculate a loss on each combination of augmented video clip and body pose tracks per sample. This happens dynamically during training with each batch sample possibly having a different number of pose tracks.

7.5. SkeleMotion Data Conversion

SkeleMotion does not implement reading and processing OpenPose data in its original form. Joint movement directions and magnitudes are calculated between consecutive time steps. These value sequences per joint are then arranged next to each other in a semantically meaningful

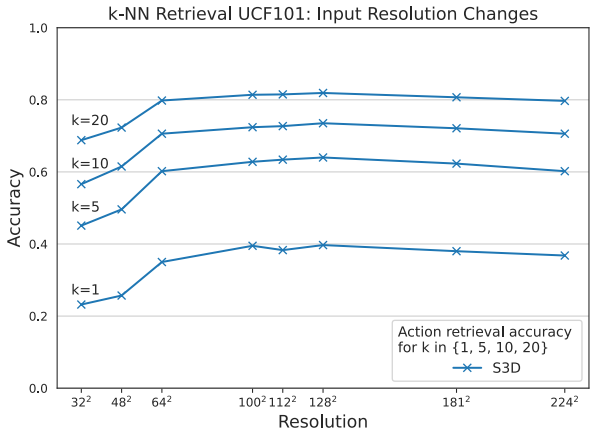

Figure 5. k-NN action retrieval accuracy with different spatial input resolutions.

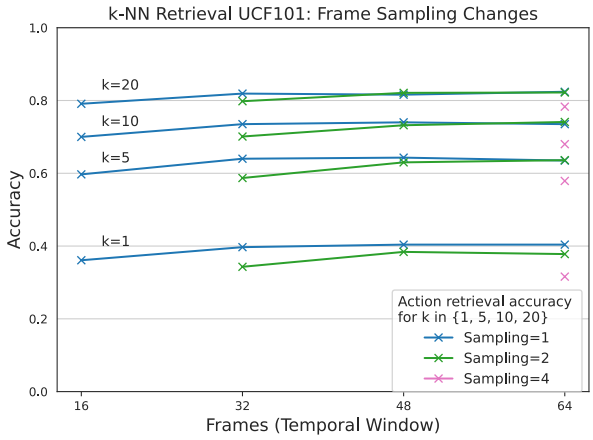

Figure 6. k-NN action retrieval accuracy under different clip lengths and frame sampling policies.

way, so a CNN can interpret them by detecting patterns between semantically connected joints. The order in which the joints are arranged is the chaining order and provided for the NTURGBD dataset joint format which has a different naming and a different total number of joints in comparison to the OpenPose COCO format[2]. We manually defined a chaining order for the OpenPose dataset which we deemed to be the best approximation of the NTURGBD chaining order.

Our OpenPose COCO Chaining order by joint id:
8, 1, 11, 1, 0, 1, 1, 5, 5, 6, 6, 7, 7, 5, 7, 1, 6, 1, 1, 2, 2, 3, 3, 4, 4, 2, 4, 1, 3, 1, 1, 11, 11, 12, 12, 13, 1, 8, 8, 9, 9, 10, 4, 0, 7, 4, 10, 13, 7

[2]OpenPose COCO format description

7.6. Training details

The R2D3D network was trained on 30-frame clips for 200 epochs with a batch size of 56 with $K_v = 2$ and $K_s \leq 2$. The S3D network was trained on 32 frame clips for 200 epochs with a batch size of 80, $K_v = 1$ and $K_s \leq 5$. The network was trained for another 30 epochs with additionally using a continuously updated queue of 1000 representation vectors in order to force the Sinkhorn-Knopp algorithm to distribute the representation vectors more broadly on the unit sphere, leading to a small but noticeable increase in performance. Both networks were trained on frames of size 128×128. For pre-training our networks on Kinetics-400 we used the Adam Optimizer with weight decay 1e-4. Learning rate at the beginning was 1e-6 which was increased to 1e-4 during training. The cluster prototypes were trained with 0.1 times the main learning rate. The prototypes were frozen for 10,000 iterations, also we limited the number of concurrent bodies per sample to one for the first 6,000 iterations (maximally five concurrent bodies afterwards). For R2D3D we made use of two concurrent video views, for S3D we only used one video view. Our best S3D network started with 3,000 prototypes and had 10,000 prototypes at the end of training. The Sinkhorn-Knopp epsilon for P-HLVC was 0.012 in pre-training, but for our domain calibration on Toyota Smarthome we used 0.05. The default augmentation settings are listed in table 6, for evaluation on the ROSE Challenge, we additionally made use of even stronger transformations such as motion blur, frame shuffling or noise.

Dataset	H	S	B	MCA	HFP
HMDB51	180°	100%	80%	5%	50%
UCF101	180°	50%	50%	10%	50%

H=±Hue, S=±Saturation, B=±Brightness, MCA=Minimal Crop Area, HFP=Horizontal flip probability. Random cropping chooses an area between the minimal crop area and 1 uniformly at random and creates a square crop with that area.

Table 6. Augmentation settings for transfer learning on HMDB51 and UCF101.

For the final evaluation, we sample 10 clips at random temporal positions per video and scale them to 224 pixels on the shorter side. Per clip we either take center crops of size 130 x 130 with a probability of 40% or random sized crops with a minimal area of 60%. Afterwards, all crops are scaled to 128 x 128. Finally, the class of a video is determined by averaging the clip predictions. As described in 4.2.2, for action retrieval we average the clip representations in order to query the nearest neighbours, we use cosine similarity for this query.

7.7. Representation Stability on Input Changes

Figure 5 shows the k-Nearest Neighbour performance after generating representations with the S3D network for differently scaled input clips, each having a length of 32 frames. Despite pre-training on clips with a resolution of 128×128, the quality of representations is only slightly affected down to a input resolution of 64×64 which reduces the computational resource usage by 75%.

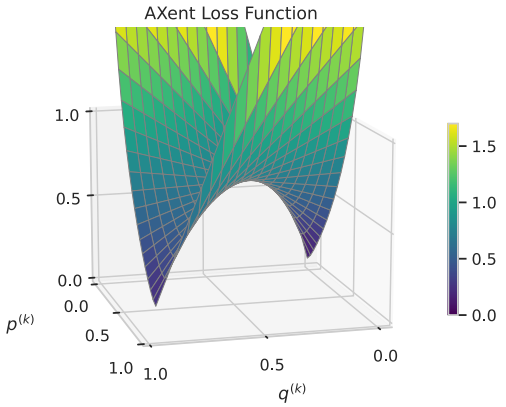

Figure 7. Element-wise AXent loss function on assignment prediction vectors p and target assignment vectors q.

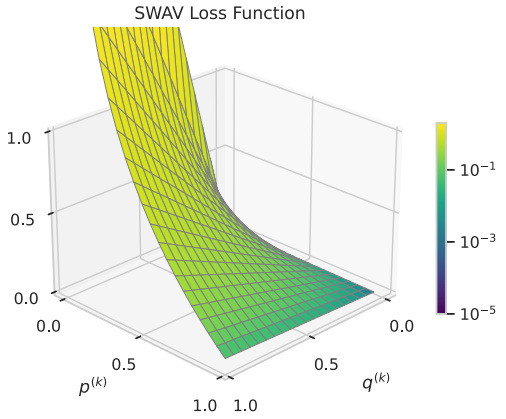

Figure 8. Element-wise SwAV loss function on assignment prediction vectors p and target assignment vectors q.

We analyse the stability of our produced action retrieval representations under changes of the input resolution as well as clip length and frame sampling strategy.

Similarly, Figure 6 presents the performance after changes to the sampling policy on frames of size 128×128. This figure varies the clip length between 16 and 64 frames and shows the k-NN results on clips where either every frame or only every second or every fourth frame is used. Note, that our S3D implementation expects at least 16 frames. Again, reducing the resource requirements only slightly affects the k-NN performance. [5] showed that the task of detecting changes to video speed (which is similar to downsampling) is also useful for self-supervised learning of representations. Our network is currently invariant to such changes and combining *P-HLVC* with video speed distinction is an interesting direction for further research.

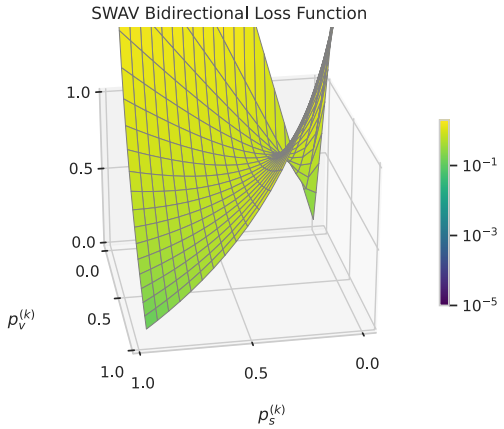

Figure 9. Simplified element-wise bidirectional SwAV loss under the assumption $q = p$.

Action classification. We fine-tune our pre-trained backbone networks on the target datasets UCF101 and HMDB51 and compare our results with existing methods which also have been pre-trained on Kinetics-400. During transfer, video clips were sampled the same way as in pre-training and the same set of augmentations was applied plus random horizontal flipping. For the final evaluation, we followed the common protocol and sampled 10 clips per video at random temporal positions, applied cropping and averaged their predictions.

We group all competing methods in Table 7, first according to the basic data modalities they use and then according to the network architectures. With the R2D3D architecture, our method is outperforming [25] and [26] on HMDB51 by a large margin, despite using a smaller resolution. On the S3D architecture, our network performs slightly worse than [27] on HMDB51. We want to note, that we pre-trained for 230 epochs while [27] pre-trained for a total of 400 epochs using frames of size 128×128. The authors of CoCRL [27] make a similar statement referring to CVRL [55], which was trained with even more epochs (800) even larger image sizes (224×224) and has almost four times more parameters. Interestingly, our performance gains are always larger on HMDB51 transfers than on UCF101 transfers. This indicates, that our person focused method is especially helpful for low quality videos and badly visible backgrounds, since HMDB51 videos are of lower quality. *P-HLVC* shows very good results while re-

Method	Architecture	Pretraining Modalities	Transfer Res.	Clip Length	Transfer Modality	UCF	HMDB
Self-Supervised Methods Requiring Video Only							
3D-RotNet [33]	R3D	V (RGB)	112	64	V (RGB)	**66.0**	**37.1**
ST-Puzzle [35]	R3D	V (RGB)	112	16	V (RGB)	63.9	33.7
DPC [25]	R-2D3D	V (RGB)	224	40	V (RGB)	75.7	35.7
MemDPC [26][1]	R-2D3D	V (RGB + F)	224	40	V (RGB)	**78.1**	41.2
P-HLVC	R-2D3D	V (RGB + B)	128	40	V (RGB)	74.5	**50.2**
SpeedNet [5]	S3D-G	V (RGB)	224	64	V (RGB)	81.1	48.8
ViCC [63]	S3D	V (RGB)	128	32	V (RGB)	82.8	52.4
CoCLR [27][1]	S3D	V (RGB + F)	128	32	V (RGB)	**87.9**	**54.6**
P-HLVC	S3D	V (RGB + B)	128	32	V (RGB)	83.2	54.4
P-HLVC	MVN A2	V (RGB + B)	128	32	V (RGB)	80.9	53.8
CoCon [56]	R3D-32	V (RGB + F)	224	NA	V (RGB)	79.1	48.5
CtP [66]	R(2+1)D	V (RGB)	112	16	V (RGB)	**88.4**	**61.7**
VCLR [39]	R2D-50	V (RGB)	224	30	V (RGB)	**85.6**	**54.1**
CVRL [55]	R3D-50	V (RGB)	224	16	V (RGB)	**92.1**	**65.4**
Self-Supervised Multi-Modal Methods							
AVTS [38]	I3D	V (RGB + A)	224	25	V (RGB)	83.7	53.0
CPD [43]	I3D	V (RGB + A/T)	224	16	V (RGB)	88.7	57.7
XDC [1]	R(2+1)D	V (RGB + A)	224	32	V (RGB)	84.2	47.1
SeLaVi [2]	R(2+1)D	V (RGB + A)	112	32	V (RGB)	83.1	47.1
GDT [52]	R(2+1)D	V (RGB + A)	112	32	V (RGB)	**89.3**	**60.0**

V Video, **F** Optical flow, **B** Body poses, **A** Audio, **T** Generated text

[1] [26, 27, 56] also publish results which make use of F on the transfer dataset, achieving higher performances.

Table 7. State of the art comparison for video-only transfer learning from Kinetics-400. For comparison we also list supervised methods. Higher scores on HMDB and UCF101 have been achieved using (significantly) larger pre-training datasets, for example [21] (video only) or [1, 38, 52, 53] (video and audio). For a detailed list covering multiple datasets see also table 2 in [27].

Number of Clusters	HMDB51 Acc.
3000	49.3
5000	50.4
10000	54.4

Table 8. Comparison with varying numbers of clusters.

quiring significantly less computing power than other methods and while performing slightly worse, our results indicate it to be more efficient than [27], at least for transfer learning on difficult datasets like HMDB51.

7.8. Impact of Varying Cluster Counts.

We present an ablation of our cluster growing strategy, which enables us to increase the number of clusters after establishing a strong initial training signal in Table 8. In order to show its impact at different number of clusters, we trained our S3D model for 14 epochs and then either continued with the initial number of 3000 clusters or iteratively increased this to 5000 or 10000 clusters. We then evaluated the transfer performance on HMDB51.

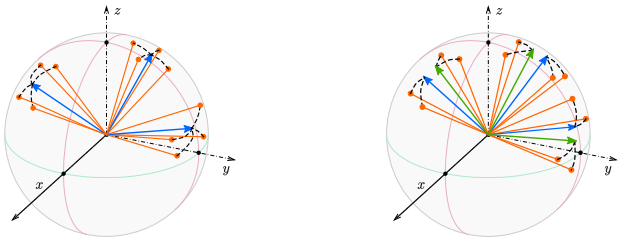

Figure 10. Representation spaces before (left) and after cluster splitting. Representations (orange) and cluster centers (blue/-green) are distributed on the unit sphere.

7.9. The Agreement Accentuating Cross-Entropy Loss

Figure 7 and 8 show the element-wise AXent loss and the original element-wise SwAV loss in comparison. As explained in Section 3.3, the AXent loss minimizes the loss by enforcing an agreement on p and q, this is easily visible in the figure. For the original SwAV loss, this is not so

obvious, since $(p, q) = (1, 0)$ also minimizes the element-wise loss. Yet, under the right conditions it performs similar to AXent. This is the case, because both losses are applied in both directions, from the video view to the body pose view and from the body pose view to the video view. For a single sample pair (video and body pose cluster assignment predictions p_v and p_s and cluster target assignments q_v and q_s), this is shown in Equations 4 and 5.

$$l_{\mathrm{S}}(p_v, p_s, q_v, q_s) = \sum_k \frac{q_s^{(k)} \log p_v^{(k)} + q_v^{(k)} \log p_s^{(k)}}{2} \quad (4)$$

$$l_{\mathrm{A}}(p_v, p_s, q_v, q_s) = \frac{l_{\mathrm{AXent}}(p_s, q_v) + l_{\mathrm{AXent}}(p_v, q_s)}{2} \quad (5)$$

Equation 4 can not be plotted element-wise directly, but for simplicity it can be approximated by assuming $q = p$. The resulting plot ignores the influence of the sinkhorn algorithm but provides an immediate intuition for the characteristics of the loss function. It is shown in Figure 9. Similar to AXent, the bidirectional SwAV loss also minimizes the loss by enforcing an agreement between p and q as long as $q \sim p$, but it is unbalanced.

```
==========================================================================
Layer                          Kernel Shape      Output Shape        Param #
==========================================================================
  S3D:                                           [16, 512]           8,434,848

  Vid-FC-2:                                      [16, 512]
      BatchNorm1d:             [512]             [16, 512]           1,024
      ReLU:                                      [16, 512]
      Linear:                  [512, 512]        [16, 512]           262,656

  Vid-Rep-FC:                                    [16, 256]
      ReLU:                                      [16, 512]
      Linear:                  [512, 256]        [16, 256]           131,328

---------

  SkeleMotionBackbone:                           [16, 512]           1,351,152
      Conv2d:                  [6, 16, 3, 3]     [16, 16, 30, 47]    880
      ReLU:                                      [16, 16, 30, 47]
      Conv2d:                  [16, 32, 3, 3]    [16, 32, 28, 45]    4,640
      MaxPool2d:                                 [16, 32, 26, 43]
      ReLU:                                      [16, 32, 26, 43]
      Conv2d:                  [32, 32, 3, 5]    [16, 32, 24, 39]    15,392
      MaxPool2d:                                 [16, 32, 11, 19]
      ReLU:                                      [16, 32, 11, 19]
      Conv2d:                  [32, 64, 3, 3]    [16, 64, 9, 17]     18,496
      MaxPool2d:                                 [16, 64, 4, 8]
      ReLU:                                      [16, 64, 4, 8]
      Flatten:                                   [16, 2048]
      Linear (FC1):            [2048, 512]       [16, 512]           1,049,088
      ReLU:                                      [16, 512]
      Linear (FC2):            [512, 512]        [16, 512]           262,656

  Sk-Rep-FC:                                     [16, 256]           131,328
      ReLU:                                      [16, 512]
      Linear:                  [512, 256]        [16, 256]           131,328
==========================================================================
```

Listing 1. Structure of our pre-training network. The output shapes are calculated on the assumption of video input of the shape $(16, 3, 32, 128, 128)$ with 16 being the batch-size. Likewise, the SkeleMotion input is assumed to be of shape $(16, 6, 32, 49)$. The Structure of S3D itself is ommited.

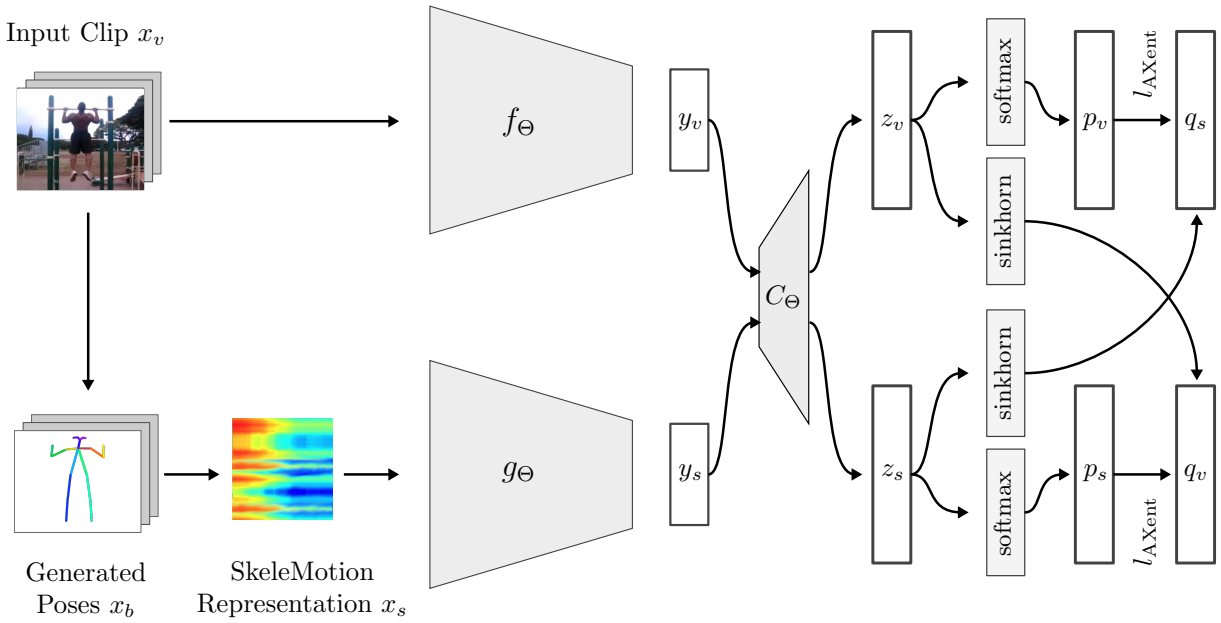

Figure 11. P-HLVC makes use of paired body pose sequences x_b encoded in a SkeleMotion representation x_s. Using the input video clip x_v and x_s, two different CNNs then generate representations y_v and y_s which are forwarded to our contrastive clustering network head.