A. Appendix overview

The appendix is organized into the following sections:

- Section B: Hyperparameter and training details
- Section C: Architecture details
- Section D: Training Algorithm Overview
- Section E: Pretraining using the InfoNCE loss
- Section F: Retrieval Results without Projection Head
- Section G: Classification using RGB and optical flow
- Section H: Visualization of learned representations
- Section I: Confusion matrix for action classification

B. Hyperparameter and training details

In addition to the implementation settings described in Section 4.1, we describe here the hyperparameter and training settings used for the experiments shown in Section 4.

The triplet margins used are $m_1 = 0.2$ and $m_2 = 0.04$ for the instance-based and temporal discrimination triplet losses respectively, and the weighting on the temporal discrimination loss is $\lambda = 1$. The embeddings that are used for clustering are computed using the temporal center-crops (16 or 32 frames) of the videos in the training set. The cluster interval k is set to 5 epochs (i.e. clustering is performed for every 5 epochs during training), the probability of sampling positives from the same video (as opposed to the same cluster) is $p_\alpha = 0.2$ for UCF101 and $p_\alpha = 0.5$ for Kinetics400, and p_β is set to 0.75 (the probability of using optical flow as the positive is 0.25). We observe that a higher p_α works better for Kinetics400 since, relative to UCF101, the increased number of classes and videos make it harder to produce high quality clusters in the beginning of the pretraining.

For optimization, the SGD optimizer was used with a fixed learning rate of 0.1, a momentum of 0.5, and no weight decay. UCF101 training was done on 2 GPUs with a batch size of 16 samples per GPU, and lasted for 600 epochs. Kinetics400 training was done on 8 GPUs with a batch size of 13 samples per GPU, and lasted for 340 epochs.

C. Architecture details

We use the 3D ResNet-18 (R3D-18) architecture [29] for all experiments. During pretraining, R3D-18 is followed by a non-linear projection head to project the feature embeddings onto a 128-dimensional space, as done in SimCLR [13] and CoCLR [28]. The projection head consists of two fully-connected layers, with batch normalization and ReLU activation after the first fully-connected layer. The output from the projection head is directly used for the nearest neighbour retrieval task. When evaluating on action classification, the projection head is replaced with a single linear layer. The detailed architecture is shown in Table 6.

Table 6. **3D ResNet-18 Architecture [29].** The projection head is used during the pretraining stage and the nearest neighbour evaluation. During finetuning, the projection head is discarded, and a single linear layer is attached after the average pooling layer.

Layer Name	Architecture
conv1	$7 \times 7 \times 7$, 64, stride 1 (T), 2(XY)
conv2_x	$3 \times 3 \times 3$ max pool, stride 2 $\begin{bmatrix} 3 \times 3 \times 3, & 64 \\ 3 \times 3 \times 3, & 64 \end{bmatrix} \times 2$
conv3_x	$3 \times 3 \times 3$ max pool, stride 2 $\begin{bmatrix} 3 \times 3 \times 3, & 128 \\ 3 \times 3 \times 3, & 128 \end{bmatrix} \times 2$
conv4_x	$3 \times 3 \times 3$ max pool, stride 2 $\begin{bmatrix} 3 \times 3 \times 3, & 256 \\ 3 \times 3 \times 3, & 256 \end{bmatrix} \times 2$
conv3_x	$3 \times 3 \times 3$ max pool, stride 2 $\begin{bmatrix} 3 \times 3 \times 3, & 512 \\ 3 \times 3 \times 3, & 512 \end{bmatrix} \times 2$
	average pool
projection head	512×2048 FC BatchNorm1d ReLU 2048×128 FC
linear layer	$512 \times$ number of classes FC

D. Training Algorithm Overview

Algorithm 1: SLIC Training Algorithm

Input: encoder $f_\theta(\cdot)$, unlabeled data X
Procedure:
while *not max epoch* **do**
  if *epoch % cluster interval = 0* **then**
    $Z = f_\theta(X)$;
    $\{P_i\}$ = FINCH(Z)
    select cluster assignments C from partition P_1. $C = \{1, 2, ..., C_{P_1}\}$
  end
  for *x in DataLoader(X)* **do**
    $z = f_\theta(x)$
    z^+ = PositiveMining(z, C)
    z^- = NegativeMining(z, z^+, C)
    $\mathcal{L}_{triplet}$ = TripletMarginLoss($z, z^+, z^-; \theta, m_1$)
    $\mathcal{L}_{temporal}$ = TripletMarginLoss(z, $\texttt{aug}(z)$, $z^+; \theta, m_2$)
    $\mathcal{L}_{total} = \mathcal{L}_{triplet} + \lambda \mathcal{L}_{temporal}$
    θ = SGD($\mathcal{L}_{total}, \theta$)
  end
end

E. Pretraining using the InfoNCE loss

To examine how performance would vary if a different loss function is used for instance-based discrimination, we pretrain the model using the InfoNCE loss [25] (replacing our instance-based triplet loss) with a batch size of 32 across 2 GPUs, and temperature set to 0.1 (these results are not meant to be an improved version of SLIC, they are a more detailed analysis on the loss function used). The results for action recognition and video retrieval are presented in Table 7 and 8 (the SLIC results are those presented in Section 4). When using the InfoNCE loss during pretraining, SLIC outperforms its triplet loss counterpart in top-1 video retrieval for UCF101, but underperforms in all other evaluation tasks.

Table 7. **Linear probing and end-to-end finetuning results for action classification using the InfoNCE loss** pretrained on UCF101, then fine-tuned on UCF101 and HMDB51, using only visual inputs. The right-most columns indicate the top-1 accuracy for each dataset. 'Frozen ✓' indicates classification with a linear layer on top of a frozen backbone; 'Frozen ✗' indicates end-to-end finetuning.

Method	Input Size	Arch.	Frozen	UCF101	HMDB51
SLIC	16×128^2	R3D	✓	72.3	41.8
SLIC	32×128^2	R3D	✓	**77.7**	**48.3**
SLIC-infoNCE	16×128^2	R3D	✓	70.2	39.8
SLIC-infoNCE	32×128^2	R3D	✓	75.4	44.7
SLIC	16×128^2	R3D	✗	77.4	46.2
SLIC	32×128^2	R3D	✗	**83.2**	**54.5**
SLIC-InfoNCE	16×128^2	R3D	✗	76.8	45.5
SLIC-InfoNCE	32×128^2	R3D	✗	81.9	53.6

Table 8. **Nearest neighbour video retrieval results using the InfoNCE loss** on UCF101 and HMDB51 (both split-1). Testing set clips are used as queries to retrieve the top-k nearest neighbors in the training set, where $k \in [1, 5, 10, 20]$. N_T indicates the temporal input size.

		UCF101 / HMDB51			
Method	N_T	R@1	R@5	R@10	R@20
SLIC	16	66.7 / 25.3	77.3 / 49.8	82.0 / 64.9	86.4 / 76.1
SLIC	32	71.6 / **28.9**	**82.4 / 52.8**	**86.6 / 65.4**	**90.3 / 77.8**
SLIC-InfoNCE	16	66.8 / 21.4	74.5 / 45.1	77.7 / 57.2	81.3 / 71.6
SLIC-InfoNCE	32	**74.4** / 26.2	81.0 / 49.7	84.1 / 62.5	87.1 / 75.9

F. Retrieval Results without Projection Head

In this section, we present the nearest neighbour retrieval results on UCF101 and HMDB51 without the projection head. After pretraining on the UCF101 dataset, we discard the non-linear projection head and use the intermediate embeddings for retrieval evaluation. The results are presented in Table 9. When evaluating without the non-linear projection head on retrieval, SLIC with 32-frame input size achieved similar results compared to Table 1.

Table 9. **Video retrieval results without projection head** on UCF101 and HMDB51. N_T indicates the temporal input size.

		UCF101 / HMDB51			
Method	N_T	R@1	R@5	R@10	R@20
CoCLR-RGB [28]	32	53.3 / 23.2	69.4 / 43.2	76.6 / 53.5	82.0 / 65.5
TCLR [15]	16	56.2 / 22.8	72.2 / 45.4	79.0 / 57.8	85.3 / 73.1
SLIC (S3D-23)	16	58.9 / 22.4	74.0 / 47.3	80.7 / 60.6	86.2 / 74.1
SLIC (S3D-23)	32	67.5 / 28.0	80.1 / 55.1	85.2 / 67.6	90.0 / 79.6
SLIC (R3D-18)	16	61.0 / 24.4	76.2 / 49.0	83.1 / 62.6	88.8 / 75.2
SLIC (R3D-18)	32	**70.8 / 28.9**	**84.6 / 55.6**	**89.0 / 68.8**	**92.7 / 81.3**

G. Classification using RGB and optical flow

We additionally conduct a study on the effect of using both the RGB and optical flow views during finetuning for action classification. The results from this study are not meant to be an improved version of SLIC, instead they are only meant to show that using both views can improve performance at the cost of additional memory (from a separate optical flow classification model) and processing time (from computing optical flow for test clips). After pretraining on Kinetics, we finetune the same pretrained model on RGB and optical flow inputs separately, then average the predictions. We compare the results to other methods that use 2-stream inputs, and present the results in Table 10. It is observed in all methods that using multi-view information during finetuning helps improve the action classification results on both UCF101 and HMDB51.

Table 10. **End-to-end finetuning top-1 accuracy results for action classification** pretrained on Kinetics400, then fine-tuned on UCF101 and HMDB51, using only visual inputs. Methods with † use optical flow in addition to RGB as inputs to the action classification model. SLIC$\hat{}$ indicates that pretraining used 16 frame inputs. Dashes indicate unavailable information.

Pretrain. Method	Input Size	Arch.	UCF101	HMDB51
CoCon-RGB* [44]	$(-) \times 224^2$	R3D-18	71.6	46.0
CoCon-E*† [44]	$(-) \times 224^2$	R3D-18	78.1	52.0
CoCLR-RGB [28]	32×128^2	S3D-23	87.9	54.6
CoCLR† [28]	32×128^2	S3D-23	**90.6**	**62.9**
SLIC$\hat{}$	32×128^2	R3D-18	83.2	52.2
SLIC$\hat{}^\dagger$	32×128^2	R3D-18	86.3	57.3

H. Visualization of learned representations

We use t-SNE to visualize the learned representations for the UCF101 test set from self-supervised pretraining on the UCF101 training set (split-1). We compare our method (the version trained with 16 frame inputs) to CoCLR [28] for 20 randomly selected classes (for better visibility) from UCF101. For SLIC, the center 16 frames are used as inputs, and for CoCLR, the center 32 frames are used as inputs since CoCLR was trained using 32-frame inputs. In both cases, the embeddings visualized are those that were used for the

video retrieval task (the 128-d outputs of the non-linear projection head for SLIC, and the 1024-d backbone features for CoCLR), and they are reduced to 50 dimensions using PCA. The perplexity for t-SNE is set to 30. By comparing Figure 4 and Figure 5, we can see that SLIC representations result in well separated clusters compared to those of CoCLR.

I. Confusion matrix for action classification

We visualize the confusion matrix of the 19 most confusing classes from UCF101 after end-to-end finetuning in Figure 6. We can see that most action class pairs that are poorly distinguished are likely difficult due to their visually similar semantics. Examples of these confusing action class pairs are visualized in Figure 7.

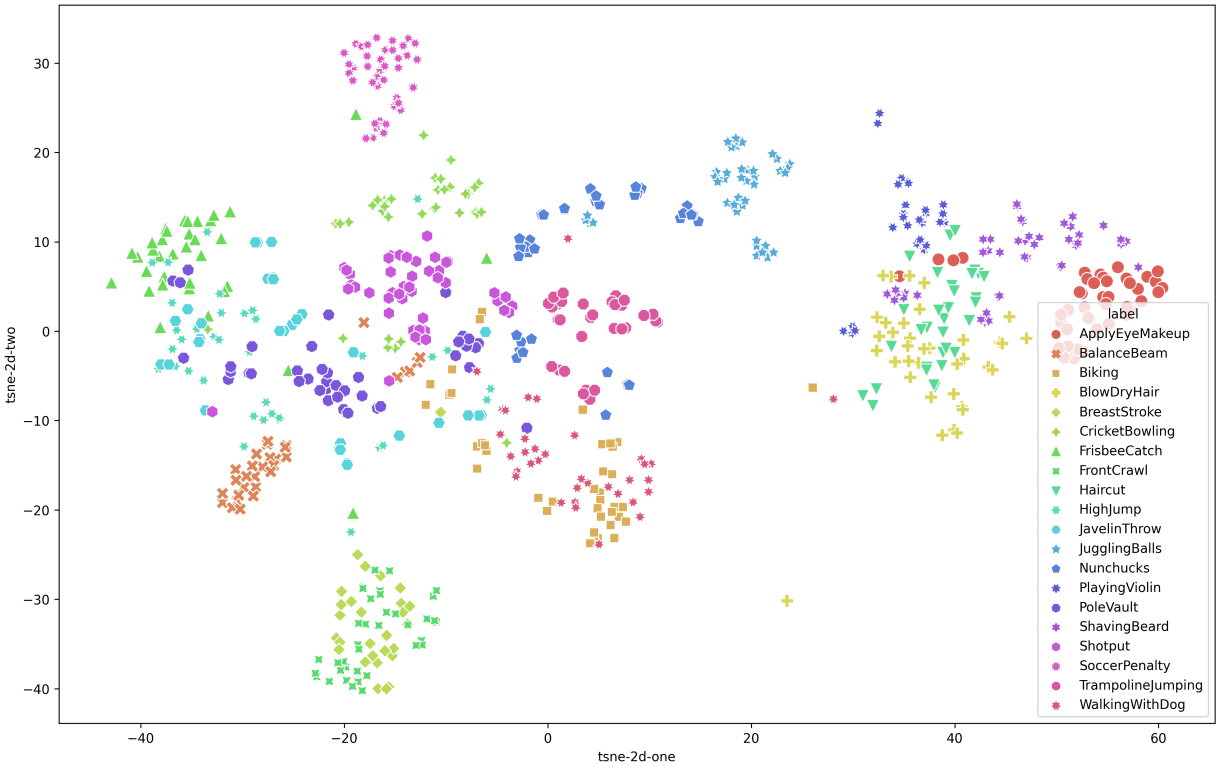

Figure 4. t-SNE visualization of SLIC embeddings for 20 randomly chosen action classes from the UCF101 test set (split-1).

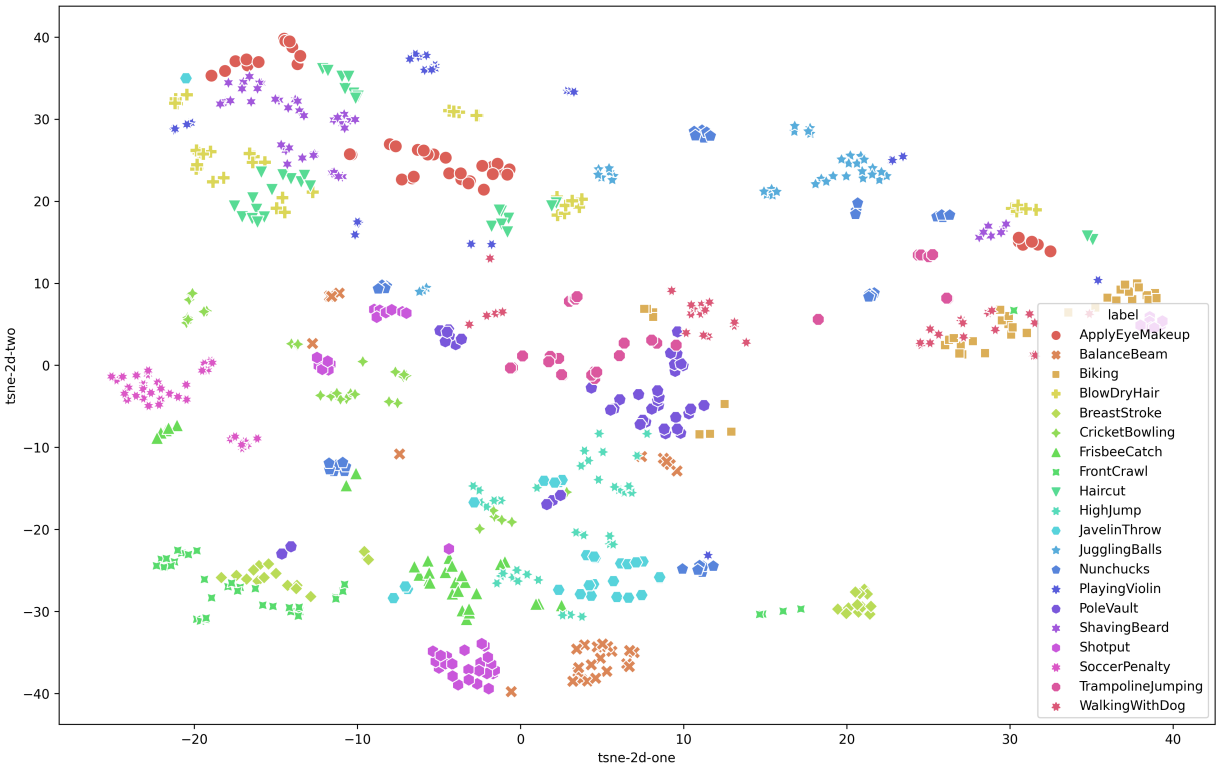

Figure 5. t-SNE visualization of CoCLR embeddings for 20 randomly chosen action classes from the UCF101 test set (split-1).

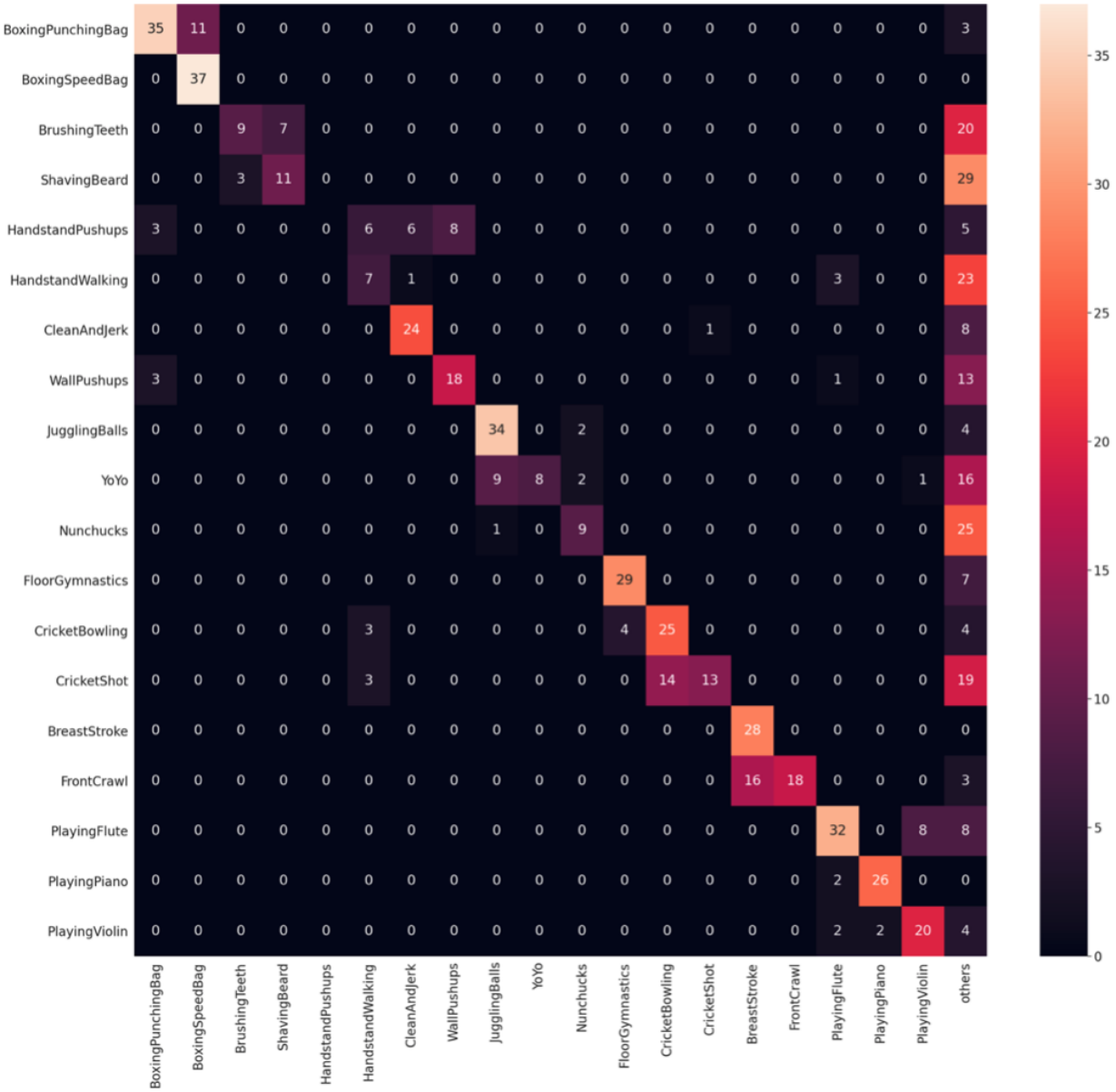

Figure 6. Confusion matrix of UCF101 generated by SLIC after pretraining on UCF101.

Figure 7. Visualization of 3 pairs of action classes that are easily confused.