Supplemental Material

Architecture. We use a 3D-Resnet18 backbone [2] unless otherwise noted and pool the feature map into a single 512-dimensional feature vector. The first 256 neurons serve as the static feature; the last 256 neurons serve as the dynamic feature. The MLP head h_i has 512 hidden units with ReLU activation, h_s and h_n have 256 hidden units. Note that the MLP heads are removed after self-supervised training and will not be transferred to downstream tasks.

In Section 4.1 we investigate the influence of different aggregation functions, namely `Sum`, `Linear`, `MLP`, and `GRU`. The `Sum` simply takes the sum over the non-stationary features of the short views, *i.e.* $\sum_{i=1}^{N} \phi_s^{(i)}$. For `Linear` and `MLP` we first concatenate the non-stationary features, *i.e.* $\left(\phi_s^{(1)}, \ldots, \phi_s^{(N)}\right) \in \mathbb{R}^{N\times 256}$, and then apply a linear layer or an MLP, respectively, mapping from $\mathbb{R}^{N\times 256}$ to $\mathbb{R}^{256}$. The `MLP` has $N \times 256$ hidden units with ReLU activation. The `GRU` aggregates the sequence of non-stationary features $\left(\phi_s^{(i)}\right)_{i=1}^{N}$ to produce a single aggregated feature $\phi_g \in \mathbb{R}^{256}$ of the same dimension as the non-stationary features of the long view. We use a one-layer ConvGRU with a kernel size of 1.

Datasets. We conduct experiments on four video datasets. For self-supervised learning, we use videos of **Kinetics-400** [4] and discard the labels. Our copy of the dataset consists of 234.584 training and 12.634 validation videos. We evaluate the learned representation on **UCF101** [8] and **HMDB51** [7] for action recognition and on the **Breakfast** dataset [6] for action segmentation.

Implementation Details. For self-supervised pretraining, we use the Adam optimizer [5] with weight decay $1e-5$, a batch size of 128 and an initial learning rate of $1e-3$, that is decreased by a factor of 10 when the validation loss plateaus. We set $\tau = 0.1$ and $m = 0.99$ for the momentum update of the key encoder. We use a memory bank size of 65.536 as in [3].

For finetuning we sample clips of 16 frames with a temporal stride of 3, and train the model end-to-end using the standard cross entropy loss. We train for 500 epochs using the Adam optimizer with an initial learning rate of $1e-4$, which we reduce at epoch 300 and 400 by a factor of 10. Weight decay is set to $1e-5$ and we use a dropout of 0.9. During inference we sample 10 clips from each test video, and use ten crop. The predictions are averaged to produce the final prediction of each test video.

Views and Augmentations. We construct long views by sampling $N \cdot L$ frames with a temporal stride of 3. We set $L = 8$ in all experiments, unless otherwise noted and provide experiments with different values of N. Given a long view of $N \cdot L$ frames, we divide them into N non-overlapping sub-sequences of L frames, which serve as short views. We apply spatial augmentations, such as crop and horizontal flip, and color augmentations to each view independently. More specifically, we use random resized crop with probability $p = 1.0$, where a spatial patch is selected covering 50% to 100% of the original frame with an aspect ratio between $3/4$ and $4/3$. Then, we resize the patch to the size 128×128. Horizontal flip is applied with probability $p = 0.5$. For color augmentations we use random color drop with probability $p = 0.1$, and apply color jitter with probability $p = 1.0$, where brightness, contrast, saturation and hue are shifted. We use a maximum brightness adjustment of 0.5, contrast of 0.5, saturation of 0.5, and hue of 0.25. The different views of a video sequence (long and short views) are augmented independently, but within a single view, the frames are augmented consistently, *i.e.* the same crop, color augmentation, etc. is selected for all frames of this view. During finetuning we keep the random crop and horizontal flip, but only apply color jitter as described above with a probability of $p = 0.3$.

Evaluation. To evaluate the learned video representations, we follow the most widely adopted approach of *finetuning*: We use the pretrained weights to initialize the 3D-Resnet18 backbone network, add a randomly initialized linear layer for classification and then train it end-to-end on split 1 of UCF101 and HMDB51.

The exact choice of the framework used for finetuning influences the final accuracies substantially; an apples-to-apples comparison between different methods is impossible. For this reason, we additionally provide retrieval results. Here, the pretrained network serves as a feature extractor and is kept fixed. We extract features for all videos in the dataset and compute R@k: For each video in the test set we retrieve the top k nearest neighbor and count a correct retrieval if at least one of the videos is of the same class as the test video. Note that R@k does not measure the precision of the retrieved results. Therefore, we also present precision-recall-curves. Here, we compute precision and recall for all values of k and plot the resulting curves. Precision and recall are calculated as it is the standard approach in retrieval. Precision is the fraction of relevant instances among the retrieved instances, while recall is the fraction of relevant instances that were retrieved.

Finally, we evaluate our models on another transfer learning task: action segmentation. We use the pretrained model to extract features from the video frames of the Breakfast dataset and subsequently train a temporal action segmentation model on top of them. Again, this evaluation does not involve any finetuning on the target dataset. This provides a more elaborate assessment of the learned

N	Training	top1 Accuracy	
		UCF101	HMDB51
2	scratch	77.2	53.7
3	scratch	75.5	49.6
4	scratch	76.5	50.9
3	curriculum	77.8	52.1
4	curriculum	78.0	52.3

Table 1. Finetuning results on UCF101 and HMDB51. We train LSFD using different numbers of sub-sequences N, either from scratch (scratch) or in a curriculum learning regime (curriculum). We notice that training from scratch is sub-optimal, compared to curriculum learning, which we attribute to the increased difficulty of the task for larger N.

representations. We evaluate the segmentation model via frame-wise accuracy, segmental edit distance and F1 scores at overlapping thresholds 10%, 25% and 50%. More specifically, for the F1 scores we determine for each predicted action segment whether it is a true or false positive by taking a threshold on the IoU with the ground truth. Then we compute precision and recall summed over all classes and compute F1 $= 2\frac{prec \cdot recall}{prec + recall}$.

Curriculum learning for larger N. We investigate the effect different numbers of sub-sequences have on the learned representations. We keep $L = 8$ fixed in this experiment and only vary N. We notice that training LSFD with $N > 2$ from scratch is sub-optimal, decreasing the performance on both UCF101 and HMDB51, see Table 1. We attribute this to the increased difficulty of the task for larger N, and propose to follow a curriculum learning strategy instead. Here, we use the pretrained weights obtained from training LSFD with $N - 1$ to initialize the training for N. We train $N = 2$ from scratch for 100 epochs, and subsequently train $N = 3$ and $N = 4$ for 40 epochs with a reduced learning rate and weight decay of $1e-4$ and $1e-6$, respectively. As evident in Table 1, this approach improves the downstream performance compared with training from scratch.

Implementation details for MS-TCN. We use the official publicly available code of MS-TCN [1] for training and evaluation. The MS-TCN model consists of four stages, each containing ten dilated convolutional layers. We train the model for 295 epochs using the Adam optimizer with an initial learning rate of 0.0005 and the `ReduceLROnPlateau` learning rate scheduler on the average loss per epoch. The first layer of MS-TCN adjusts the dimension of the input features (*i.e.* 512 for full features and 256 for stationary and non-stationary features in our experiments) using a 1×1 convolution; the remaining layers have 64 channels.

Feature Decomposition Analyses. In this section, we are aiming to get a better understanding of our feature decomposition; specifically, we are interested in the difference between our stationary and non-stationary features. To that end, we use the pretrained model as a feature extractor (without the MLP heads h_s and h_n). To compute similarities between two feature vectors x and y, we use the cosine similarity:

$$\frac{x^T y}{\|x\|\|y\|}.$$

Furthermore, we compute retrieval accuracies among videos that can be classified with different numbers of frames. First, we train separate model receiving $N = 1, 2, 4, 8, 16, 32$ frames as input. Then, we group the videos into disjoint subsets based on the numbers of frames that are needed for classification. For $N = 1$ this subset consists of all videos that are correctly classified by the model trained with $N = 1$ frames. For $N = 2$ the subset consists of those video that are correctly classified by the model with $N = 2$ frames as input, but that were misclassified by the $N = 1$ model, for $N = 4$ we exclude the videos from $N = 1$ and $N = 2$, etc.

References

[1] Yazan Abu Farha and Juergen Gall. MS-TCN: Multi-stage temporal convolutional network for action segmentation. In *CVPR*, 2019. 2

[2] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and ImageNet? In *CVPR*, 2018. 1

[3] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 1

[4] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *arXiv*, abs/1705.06950, 2017. 1

[5] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 1

[6] Hilde Kuehne, Ali B. Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *CVPR*, 2014. 1

[7] Hilde Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. HMDB: A large video database for human motion recognition. In *ICCV*, 2011. 1

[8] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv*, abs/1212.0402, 2012. 1