Supplementary Material: Self-supervised Learning for Unintentional Action Prediction

Olga Zatsarynna[1], Yazan Abu Farha[2], and Juergen Gall[1]

[1] University of Bonn
[2] Birzeit University
{zatsarynna, gall}@iai.uni-bonn.de
yabufarha@birzeit.edu

1 Implementation Details

We implemented our approach using the Pytorch [8] framework. As backbone network, we use a randomly initialized ResNet3D-18 [4]. For self-supervised training, two additional heads are added to the backbone network: projection head and order prediction head. We use a non-linear projection head for self-supervised learning following [1] and an order head for order prediction. Both projection head and order head operate on the output features of the backbone network and consist of two linear layers with a ReLU. They are only used for self-supervised learning and are discarded afterwards. For the loss computation, we sample 10 triplets from each video. For videos with less than 10 clips, we sample as many triplets as number of clips in the video. For the temporal contrastive loss, the temperature is set to $\tau = 0.07$. Training is done in mixed precision using Adam [7] optimizer with starting learning rate $lr = 0.001$ on 4 NVIDIA TITAN Xp GPUs with a per-GPU batch size of 28 examples.

For downstream tasks, the pre-trained backbone network is used as a feature extractor and a linear layer for the specific downstream task is added. For classification and localization, the network is trained on 4 GPUs with batch size 128 (linear) and 64 (finetuning). For anticipation, we use 2 GPUs with batch size 64. Other training parameters remain the same as during the pre-training stage.

2 Oops Dataset

2.1 Comparison to Fully-supervised Approaches

We compare our proposed self-supervised approach to the fully-supervised methods [3,11,12] for unintentional action prediction on the Oops dataset. While [3,11] tackle both classification and localization tasks, [12] only considered the problem of unintentional action localization. While our approach is trained self-supervised and uses the labels only for training the linear classifier, these methods train a full network on the labeled videos. Xu *et al.* [11] uses additional 3004 training videos that contain only intentional actions.

Method	Add. Data	Classifier	Finetune
Supervised			
LGF [3]	×	Transformer	77.9
UAL-CE [11]	✓	LSTM	82.6
Unsupervised			
Ours	×	Linear	61.9

Table 1. Classification accuracy on the Oops dataset.

We present the comparison of our approach to the fully-supervised methods for the tasks of classification and localization in Table 1 and Table 2, respectively. We can observe that for classification all considered approaches perform better than our method. However, it needs to be noted that the fully-supervised methods use non-linear classification layers. More specifically, Epstein *et al.* [3] use the Transformer encoder [9], while Xu *et al.* [11] make use of an LSTM network [5].

Although our approach uses only a linear classifier, our approach outperforms [3, 12] for the task of unintentional action localization with a threshold of 0.25 seconds. This shows that the video representations learned by our proposed self-supervised approach are sufficiently distinctive to detect the transition point. [11] shows the highest accuracy for both tasks, but it uses additional training data.

2.2 Cut Detection Network

To detect videos that contain two or more unrelated scenes due to failures of the automatic cut detection, we train a cut detection network. To this end, we harness a subset of 15000 videos from the Kinetics dataset [6] re-sampled at 16 fps, without using the video labels. Given these videos, we formulate the task of cut detection as a binary classification task. We train a cut detection network to predict whether a given 16-frame clip does or does not contain frames from two different scenes. For this, we create two types of clips during training. The clips of the first type consist of contiguous frames from a single video. Formally, from a video v_i, we randomly select a 16-frame clip $x_i = \{f_i^1, \ldots, f_i^{16}\}$. Such

Method	Add. Data	Classifier	Accuracy within	
			1 sec	0.25 sec
Supervised				
LGF [3]	×	Transformer	72.4	39.9
TLA [12]	×	LSTM	73.2	40.3
UAL-CE [11]	✓	LSTM	81.2	55.4
Unsupervised				
Ours	×	Linear	70.5	41.3

Table 2. Temporal localization accuracy on the Oops dataset.

clips are assigned the label $l = 0$. The clips of the second type are created by concatenating contiguous frames from two separate videos into a single clip. They get assigned the label $l = 1$. More specifically, given two separate videos v_i and v_j, we randomly select 16-frame clips $x_i = \{f_i^1, \ldots, f_i^{16}\}$ and $x_j = \{f_j^1, \ldots, f_j^{16}\}$ from each video. Then, we randomly select a time point t from the interval $\{4, \ldots, 12\}$ and create a concatenated clip $x = \{f_i^1, \ldots, f_i^t, f_j^{t+1}, \ldots, f_j^{16}\}$. As network, we use the same ResNet3D-18 [4] as for the self-supervised pre-training, which receives input clips of 16 frames. On top of the backbone network, we define a non-linear head for cut prediction that consists of two linear layers with a ReLU. The training is performed for 20 epochs with a batch size of 32, from which half of the clips have label $l = 0$ and the other half has label $l = 1$. Training is done in mixed precision using Adam [7] optimizer with learning rate $lr = 0.001$. After training the network for cut prediction, we use it in a sliding window fashion to detect the missed cuts in the training videos of the Oops dataset [2]. If at a certain time point of a video the score of label $l = 1$ exceeds the threshold $tr = 0.85$, the video is filtered out and is not used for the self-supervised pre-training. According to our estimates, approximately 20% of all videos from the dataset contain a missed scene change.

To understand the effect of the video filtering, we present the results for linear classification and linear localization if the representation has been learned on the filtered videos *(Filtered)* or all videos *(All)* in Table 3. While for linear classification, the network trained with all videos slightly outperforms the model trained only with the filtered videos, the localization performance suffers from using all videos for self-supervised learning. However, our approach outperforms in both cases the previous state-of-the-art method Video Speed [2].

3 LAD2000 Dataset

3.1 Dataset

LAD2000 [10] is a dataset of anomalous video sequences. It contains 2000 video sequences that include both normal and abnormal videos belonging to 14 anomalous categories. All videos have both a video-level label and frame-wise annotations specifying abnormal frames. As our focus is to learn representations from unintentional actions, we only consider videos with anomalous actions. So, in total we make use of 762 videos, 482 for training and 280 videos for testing.

Linear classifier			
	Localization		**Classification**
Method	1 sec	0.25 sec	All labels
Video Speed [2]	65.3	36.6	53.4
Filtered (Ours)	**70.5**	**41.3**	60.9
All	69.9	39.3	**61.6**

Table 3. Effect of video filtering on the Oops dataset.

Method	Classification	
	Linear	Non-Linear
Kinetics	63.2	65.8
Video Speed [2]	45.0	49.6
Ours	**51.3**	**57.3**

Table 4. Classification accuracy on the LAD2000 dataset.

Additionally, since our backbone network operates on clips rather than frames, we transform the original frame labels into clip labels. Specifically, we resample the original frames and labels from 25 fps to 16 fps. As previously, the input to our model are clips of 16 frames extracted from videos in 0.25 second intervals. Clip labels are defined as follows: if all clip frames are labeled as normal, then the clip is labeled as intentional; else if all clip frames are labeled as abnormal, the clip is labeled as unintentional. Otherwise the clip is labeled as transitional.

3.2 Classification

For the LAD2000 dataset, we evaluate our model only on the donwstream classification task. Apart from the linear classification, we also present results of training a two-layer classifier with ReLU. In both cases, we use the fixed backbone. Since the average number of clips per video in LAD2000 (192 clips/video) is higher than in the Oops dataset (34 clips/video) while the total number of videos is smaller, we sampled 100 triplets per video for training instead of sampling only 10 as for the Oops dataset. The training was performed on 2 GPUs with batch size 256.

We present the results in Table 4. Our approach substantially outperforms the previously proposed Video Speed [2] model by 6.3% and 7.7% in the linear and non-linear setting, respectively. However, we can also observe that the model pretrained with full supervision on the Kinetics dataset significantly outperforms our approach. Thus, there is still room for improvement for pre-training on unintentional videos.

References

1. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.E.: A simple framework for contrastive learning of visual representations. ArXiv **abs/2002.05709** (2020)
2. Epstein, D., Chen, B., Vondrick, C.: Oops! predicting unintentional action in video. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
3. Epstein, D., Vondrick, C.: Learning goals from failure. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021)
4. Hara, K., Kataoka, H., Satoh, Y.: Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2018)

5. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
6. Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, A., Suleyman, M., Zisserman, A.: The kinetics human action video dataset. ArXiv **abs/1705.06950** (2017)
7. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2015)
8. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32 (2019)
9. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems (2017)
10. Wan, B., Jiang, W., Fang, Y., Luo, Z., Ding, G.: Anomaly detection in video sequences: A benchmark and computational model. IET Image Processing (2021)
11. Xu, J., Chen, G., Lu, J., Zhou, J.: Unintentional action localization via counterfactual examples. IEEE Transactions on Image Processing **31**, 3281–3294 (2022)
12. Zhou, N., Chen, G., Xu, J., Zheng, W.S., Lu, J.: Temporal label aggregation for unintentional action localization. In: IEEE International Conference on Multimedia and Expo (ICME) (2021)