Temporal Segment Networks: Towards Good Practices for Deep Action Recognition Supplementary Materials

Limin Wang[1], Yuanjun Xiong[2], Zhe Wang[3], Yu Qiao[3], Dahua Lin[2], Xiaoou Tang[2], and Luc Van Gool[1]

[1]Computer Vision Lab, ETH Zurich, Switzerland
[2]Department of Information Engineering, The Chinese University of Hong Kong
[3]Shenzhen Institutes of Advanced Technology, CAS, China

1 Implementing the Temporal Segment Networks

Here we describe more details on implementing the temporal segment networks. In the data feeding pipeline, one input video will be first divided into K segments as described in the paper. Then we randomly pick one snippet from each segment to generate the input. In the minibatch SGD learning, for a minibatch of b videos, this means their inputs to the networks comprise $K \times b$ frames of RGB images or $K \times b$ stacked optical flow (grayscale) images.

After the inputs pass through the last layer of ConvNets, which is a fully-connected layer with 101 outputs, the activations will be in a $(K \times b, 101)$ tensor. We reshape the tensor to the shape of $(b, K, 101)$. Then the second dimension of tensor is reduced with the aggregation function g, producing a tensor with $(b, 101)$ dimensions. Loss values are then calculated on the probabilities produced by applying sample-wise Softmax to this tensor. The back-propagation is conducted accordingly.

2 Details on Model Visualization

The original visualization tool [1] only support visualization of models based on the GoogleNet architecture and with 3 channel inputs, i.e. RGB images. Our models use the BN-Inception [2] architecture, which has batch-normalization modules. To make the tool work with our models, we use a simple technique called *absorbing*, where we absorb the parameters in batch normalization modules into the weights and biases of their consecutive layers (usually convolutional or fully connected layers). The result is a normal ConvNet model without batch-normalization. This is also beneficial to the inference speed of the network. To make the tool work with models taking optical images a inputs, we first adapt the tool take inputs of arbitrary channels. After the gradient ascent process, we reconstruct the x and y-directional flow field images by averaging their corresponding channels in the results.

3 More Visualization Results

We present more visualization samples with higher resolution in following pages.

References

1. : Deep draw. https://github.com/auduno/deepdraw
2. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML. (2015) 448–456

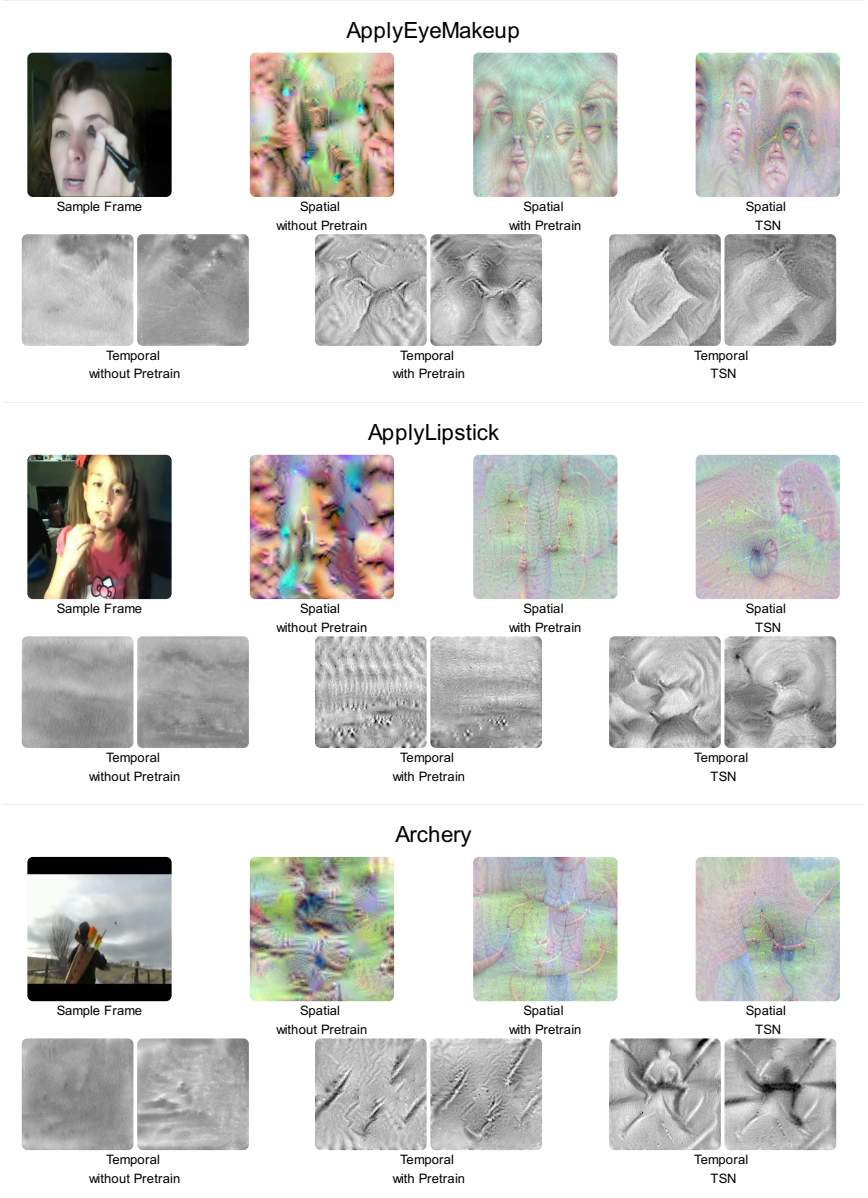

Fig. 1. Visualization.

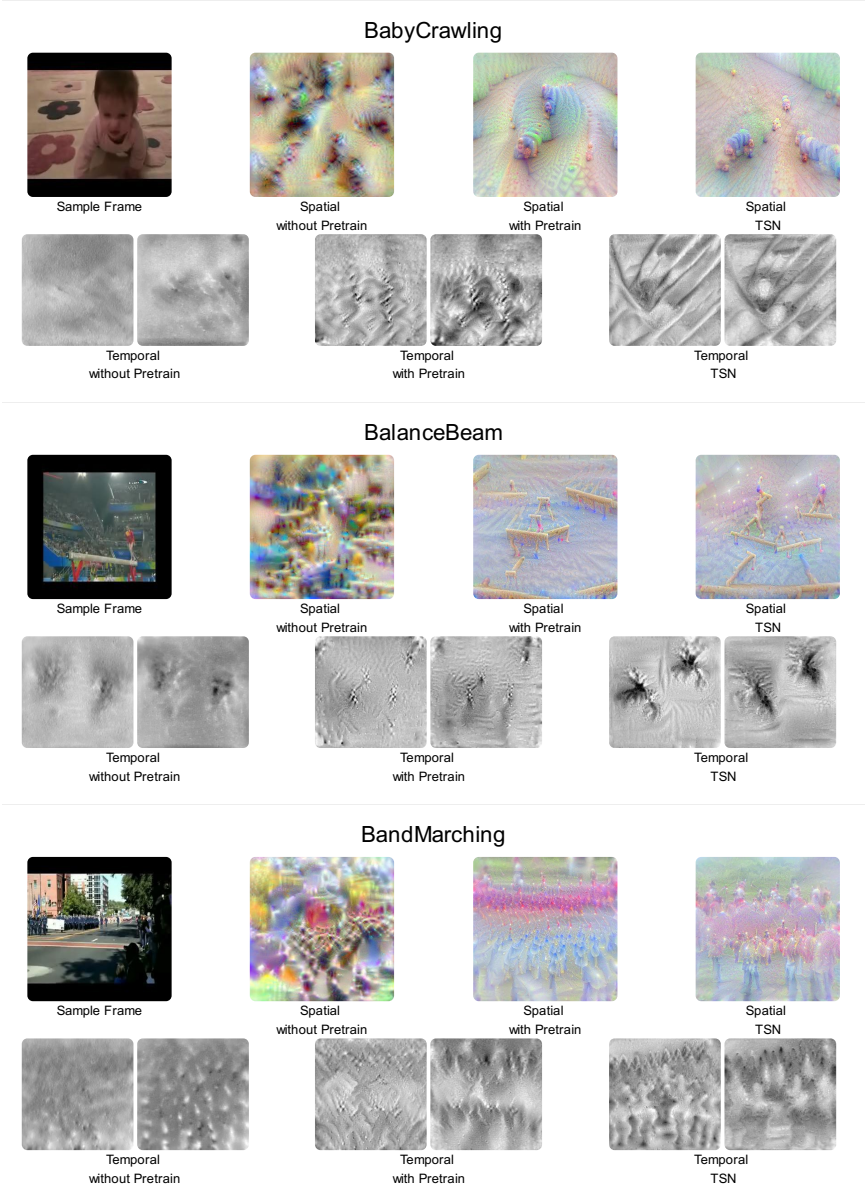

Fig. 2. Visualization.

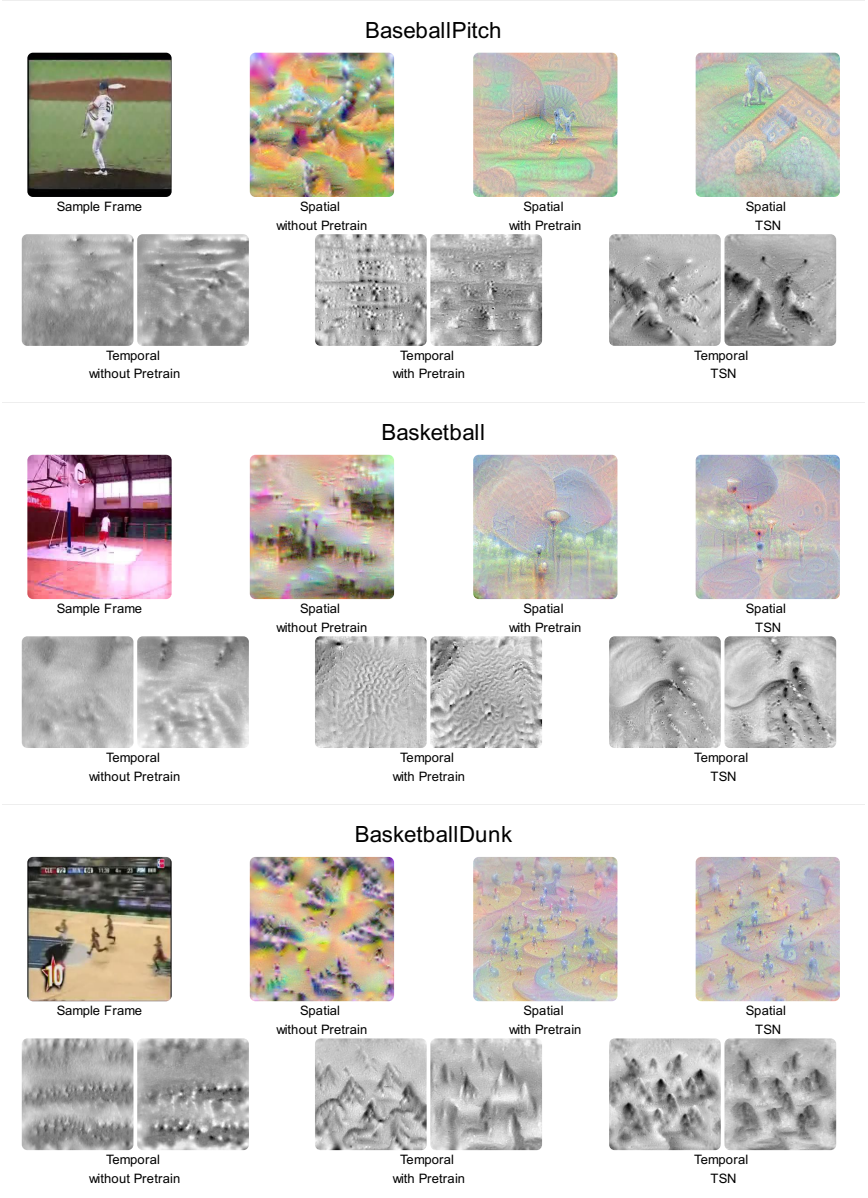

Fig. 3. Visualization.

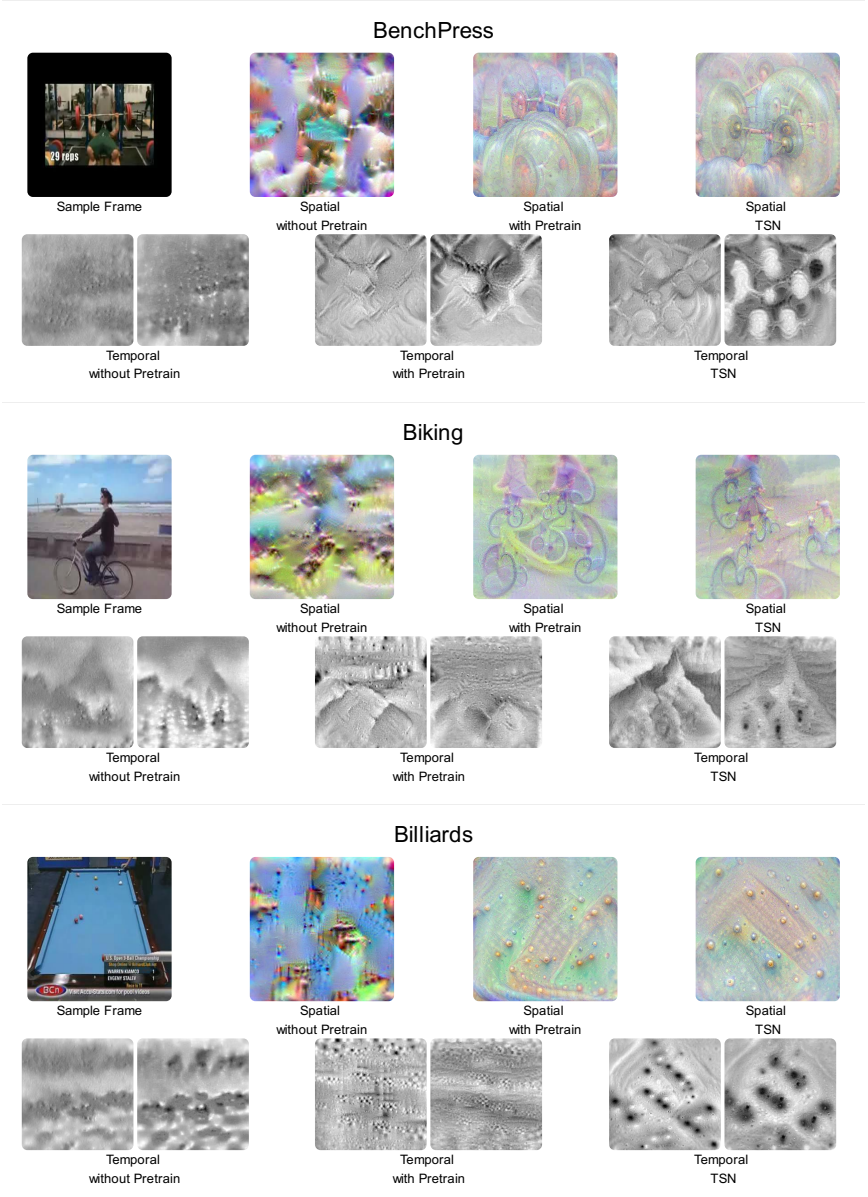

Fig. 4. Visualization.

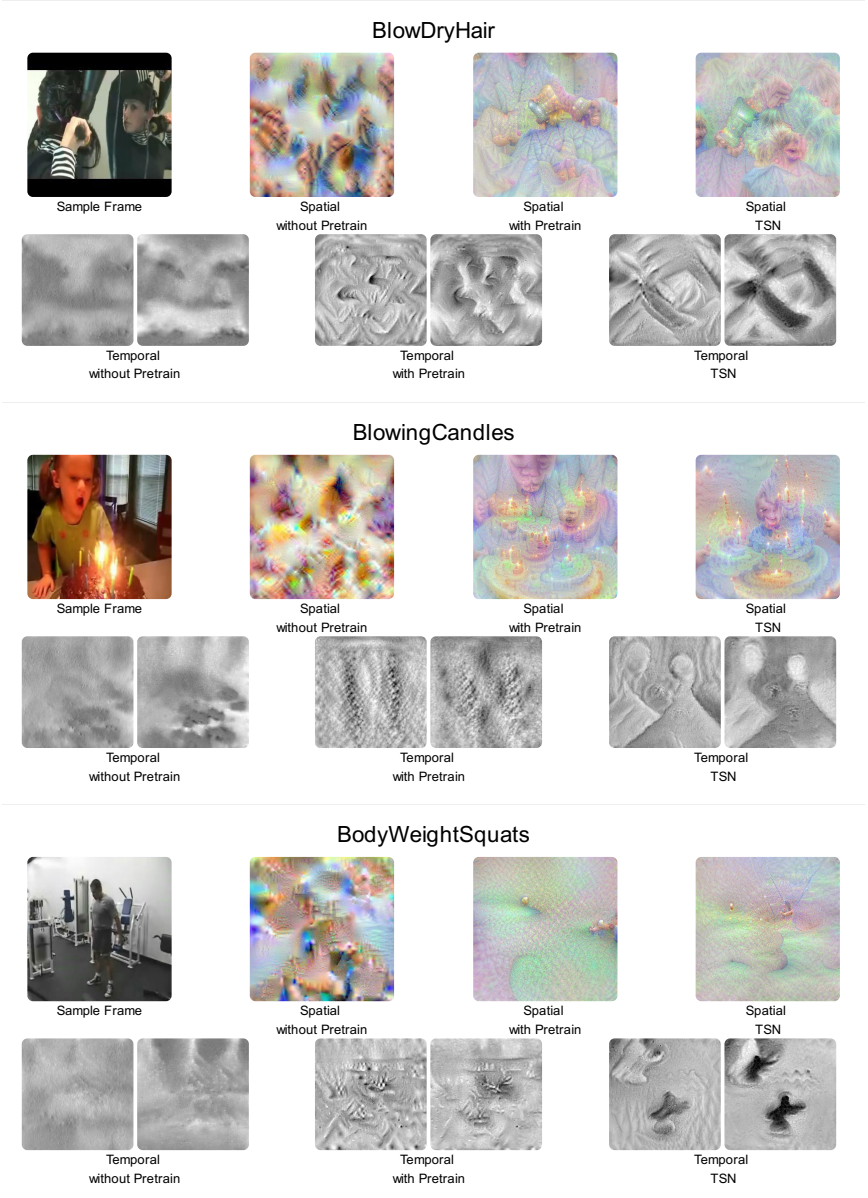

Fig. 5. Visualization.

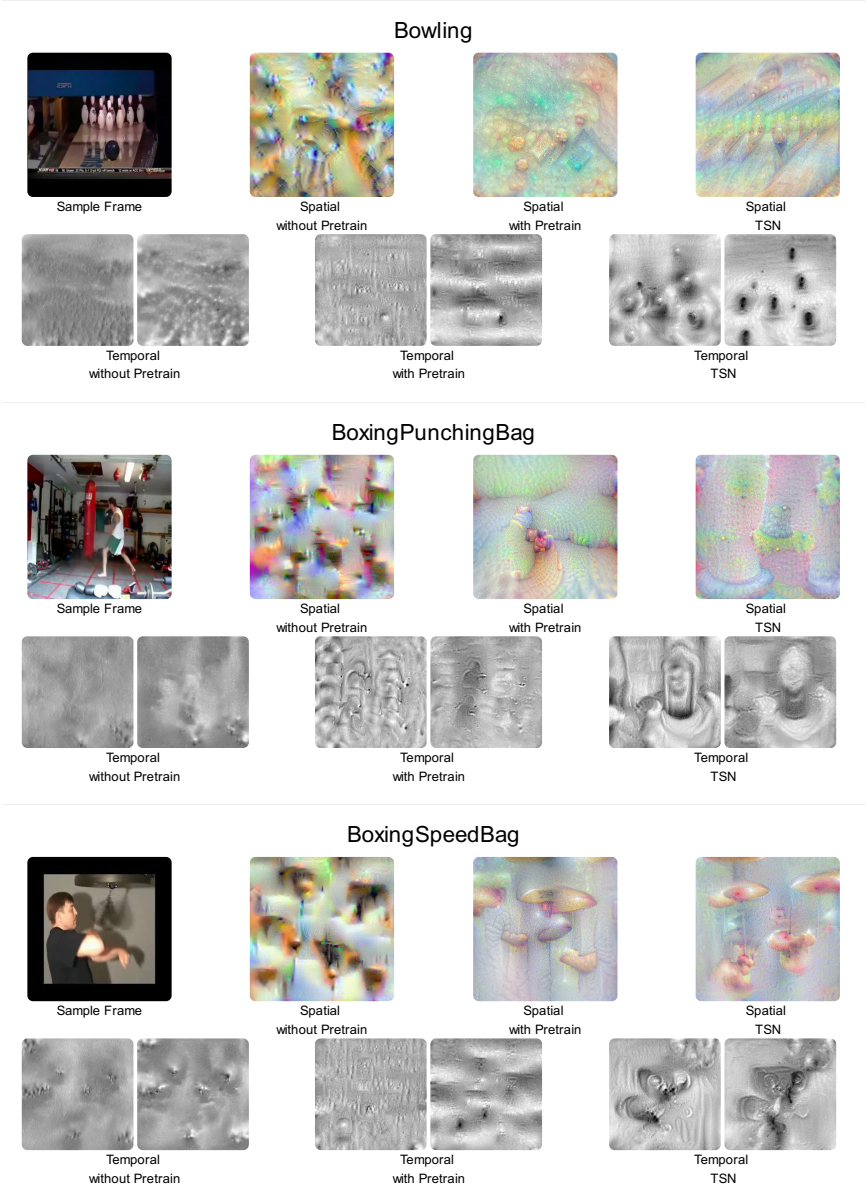

Fig. 6. Visualization.

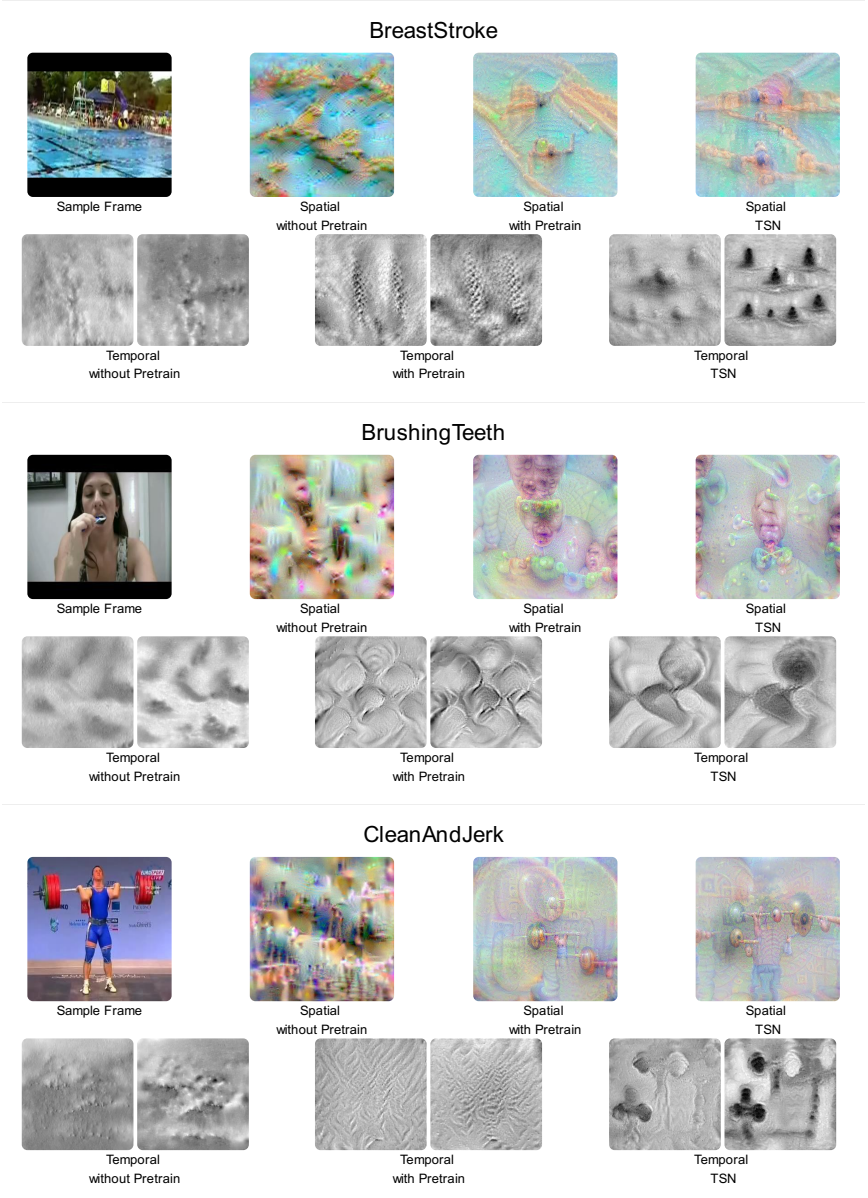

Fig. 7. Visualization.

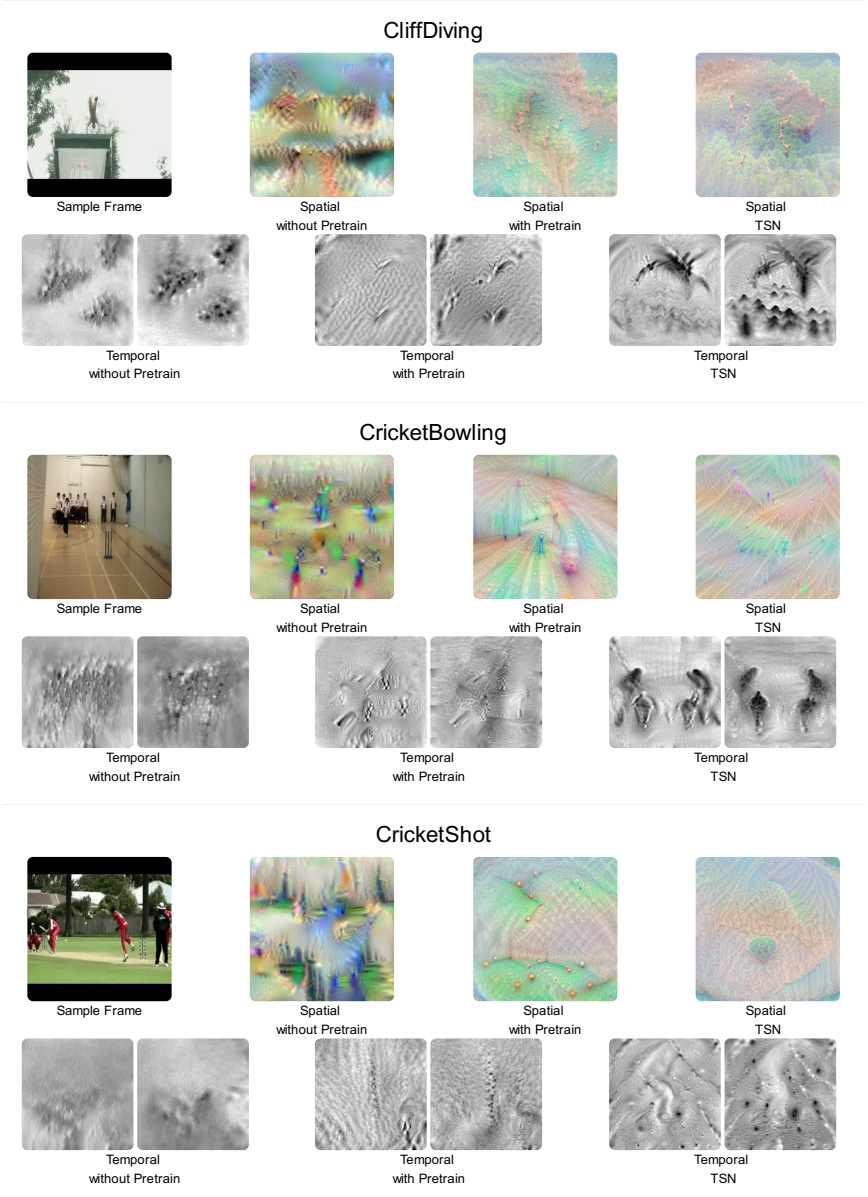

Fig. 8. Visualization.

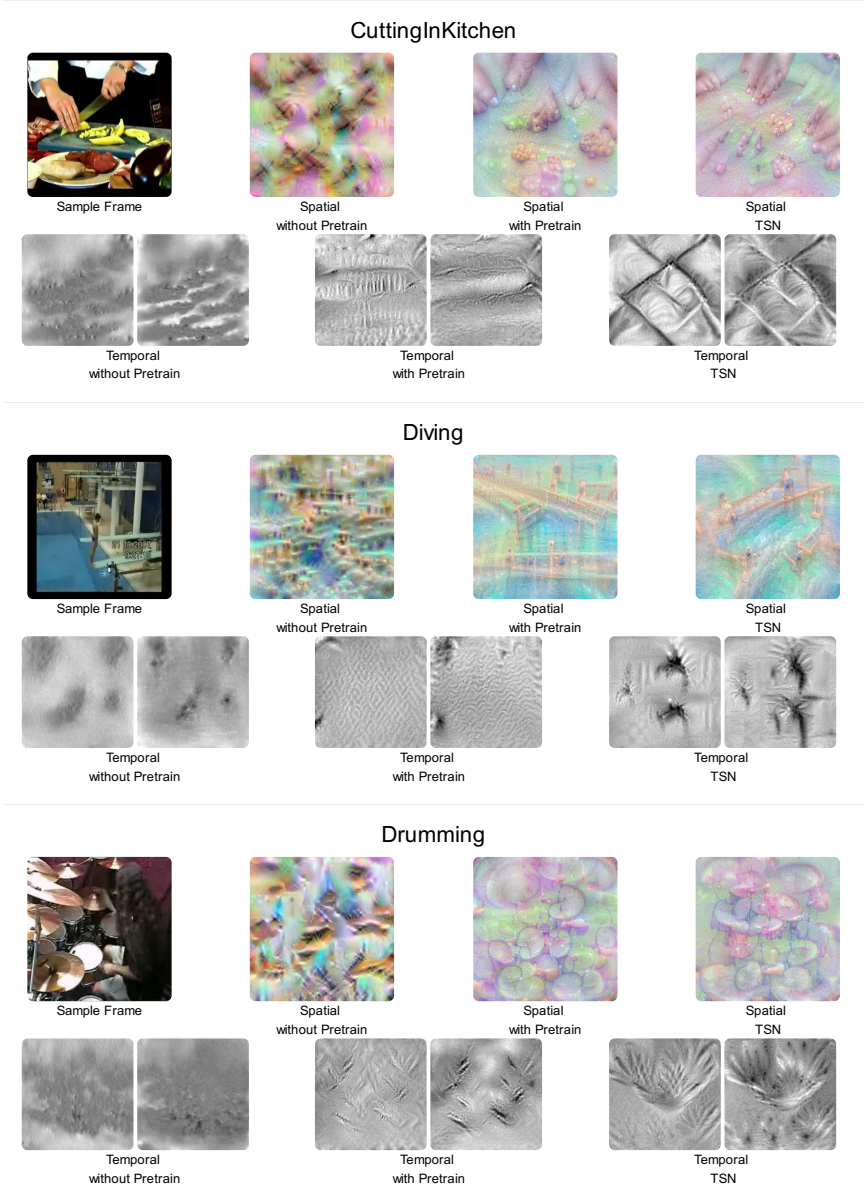

Fig. 9. Visualization.

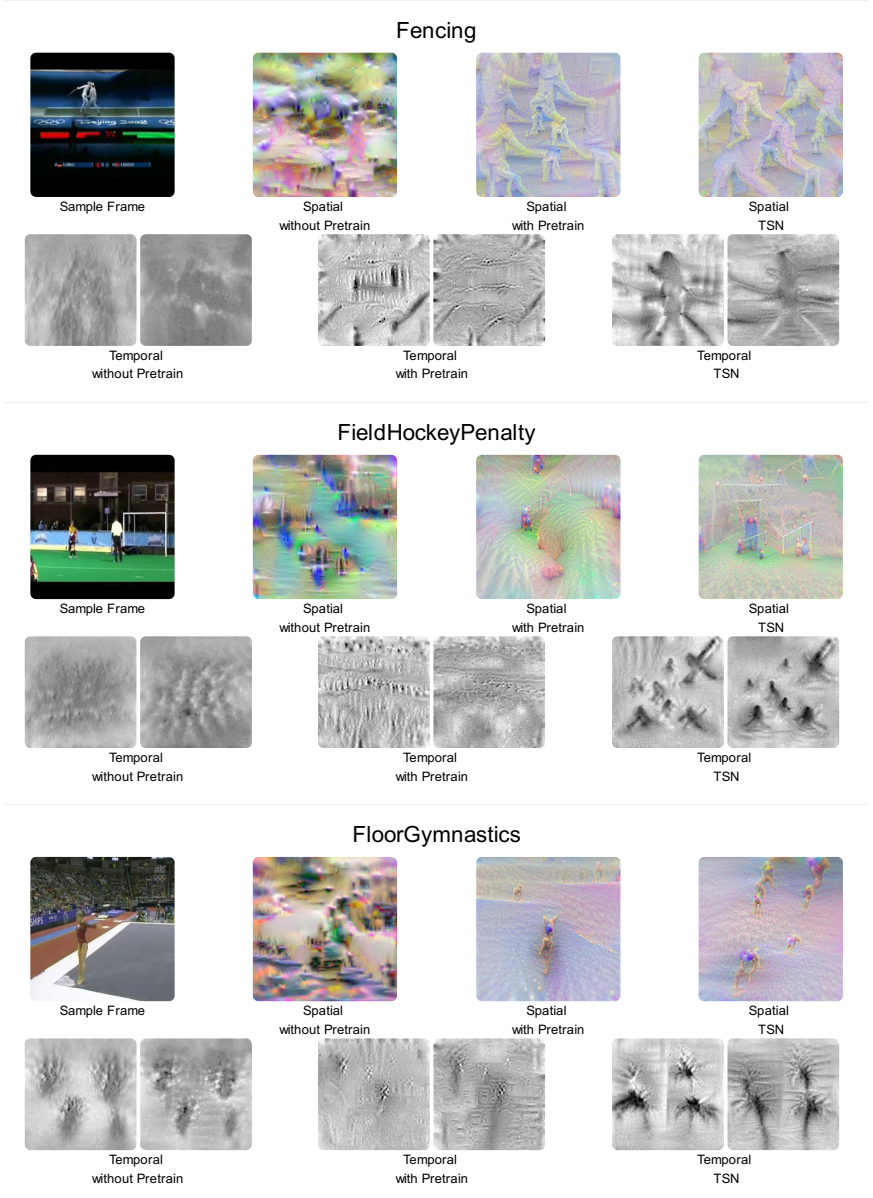

Fig. 10. Visualization.

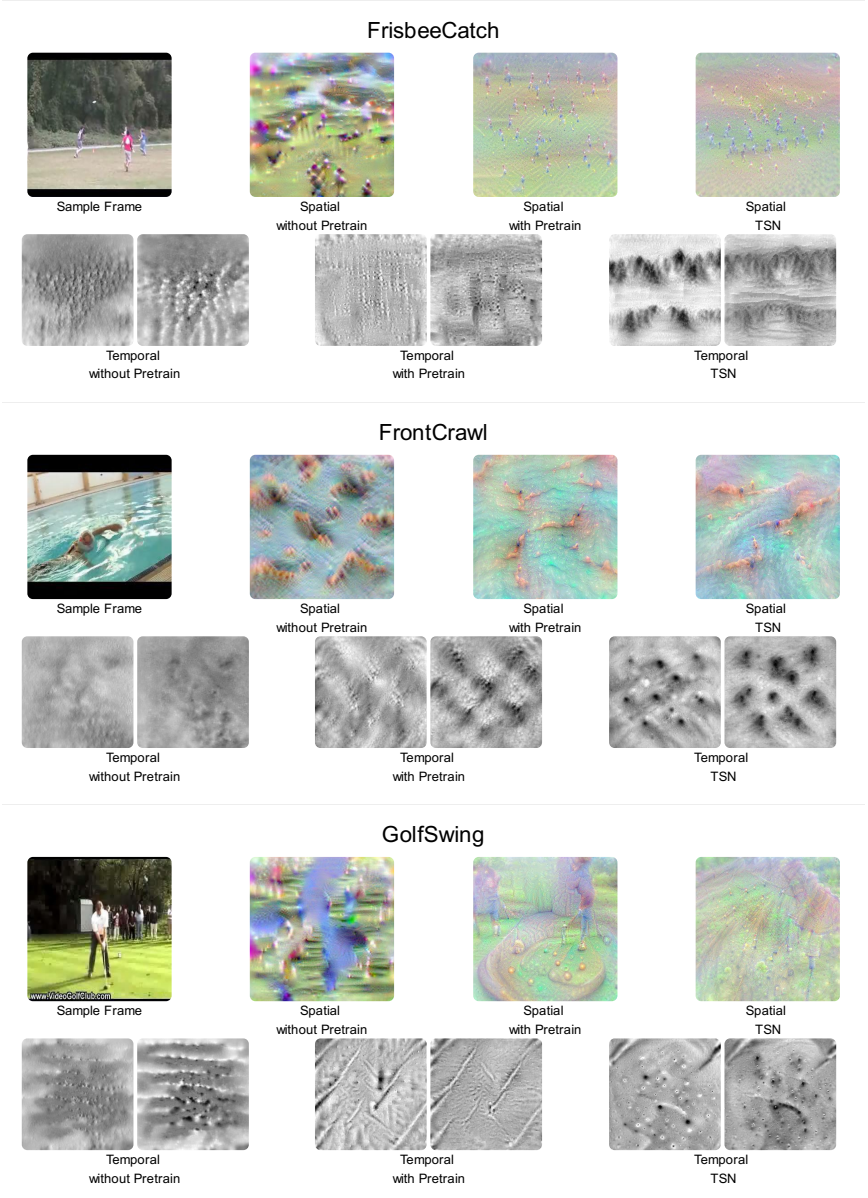

Fig. 11. Visualization.

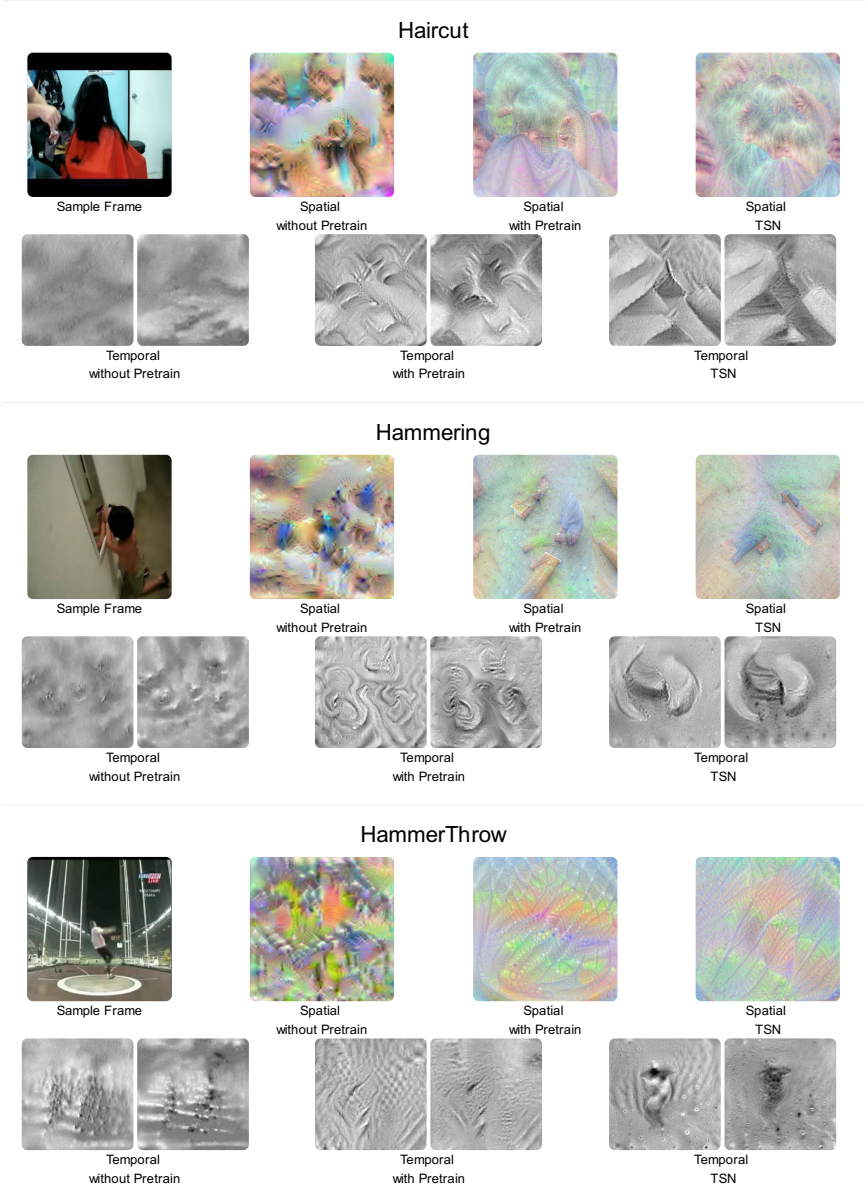

Fig. 12. Visualization.

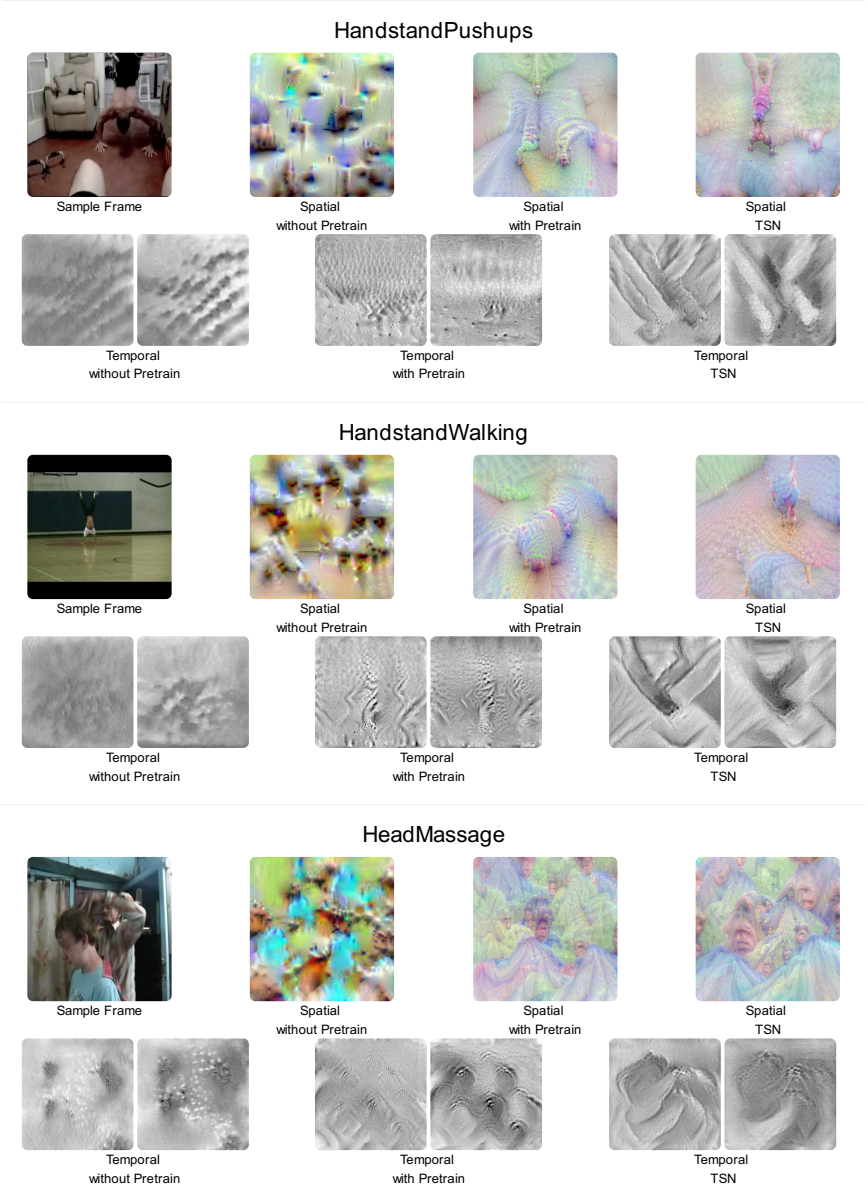

Fig. 13. Visualization.

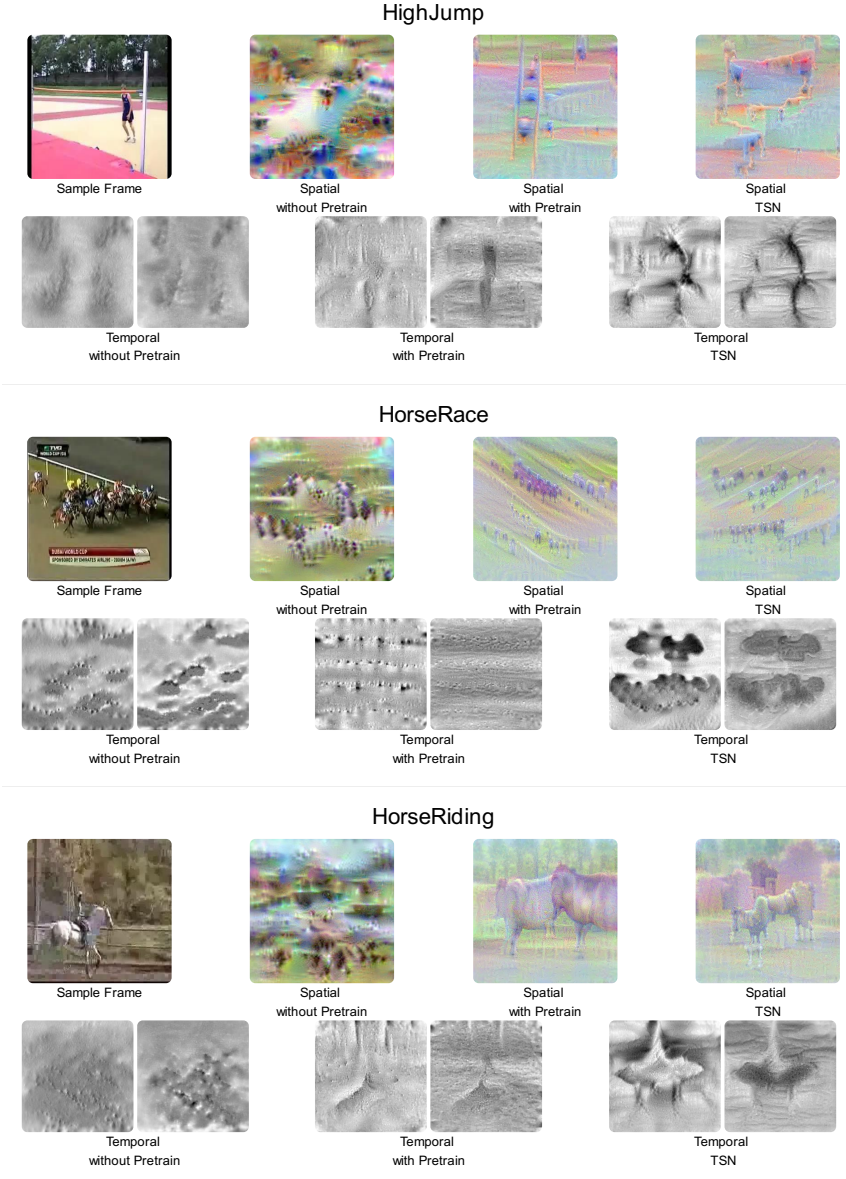

Fig. 14. Visualization.

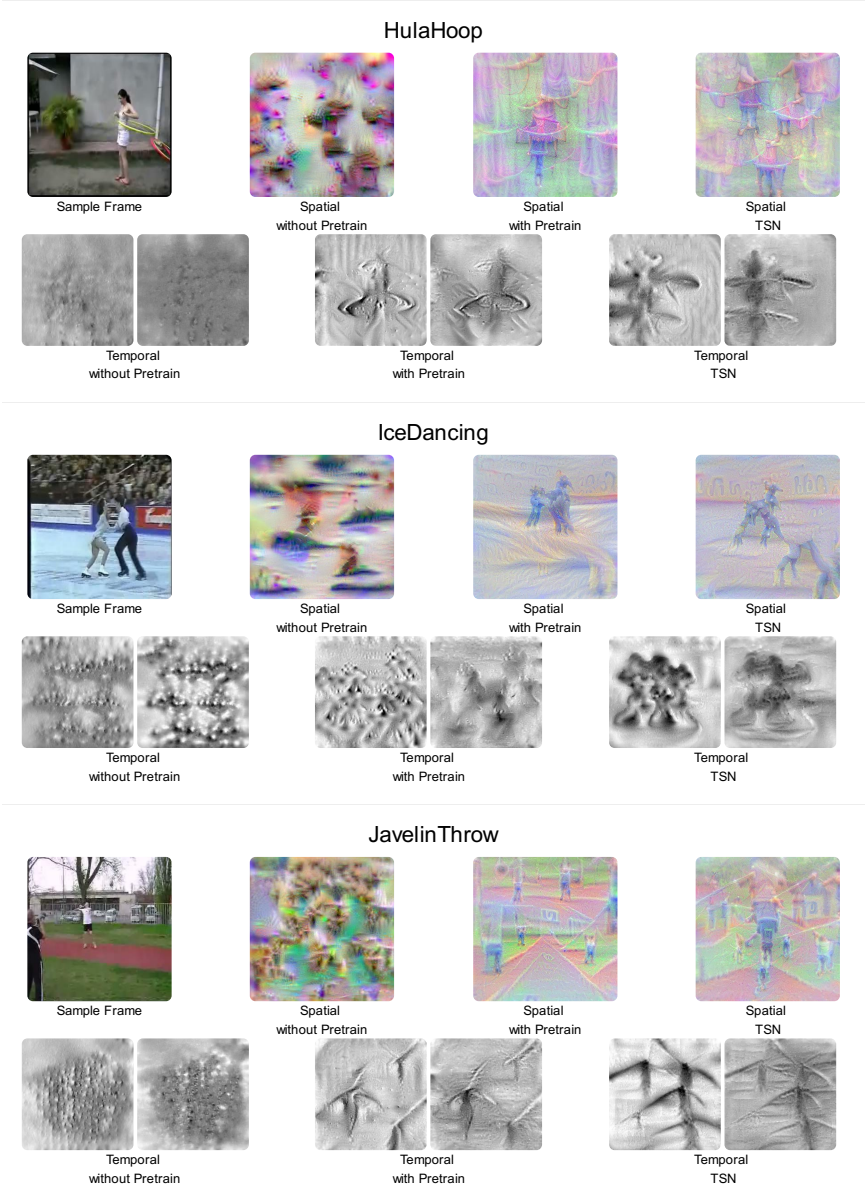

Fig. 15. Visualization.

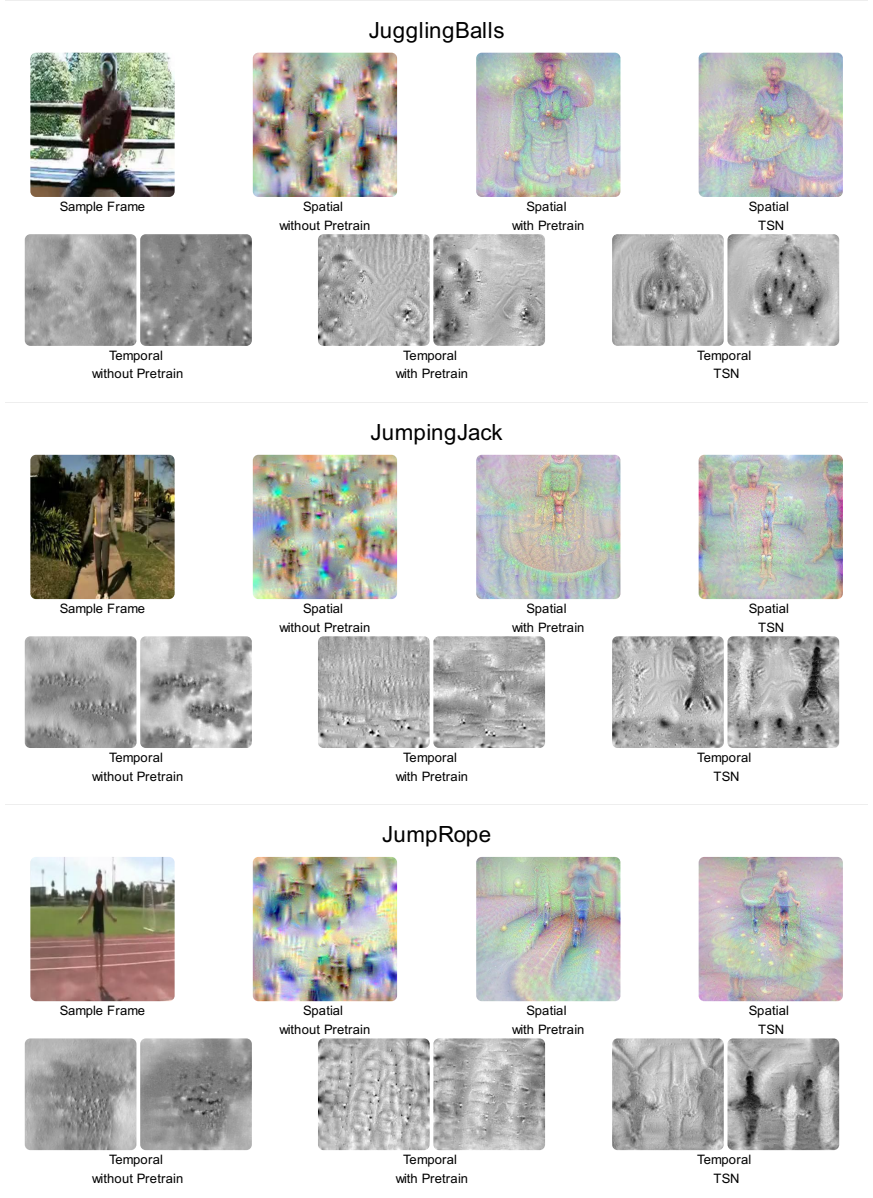

Fig. 16. Visualization.

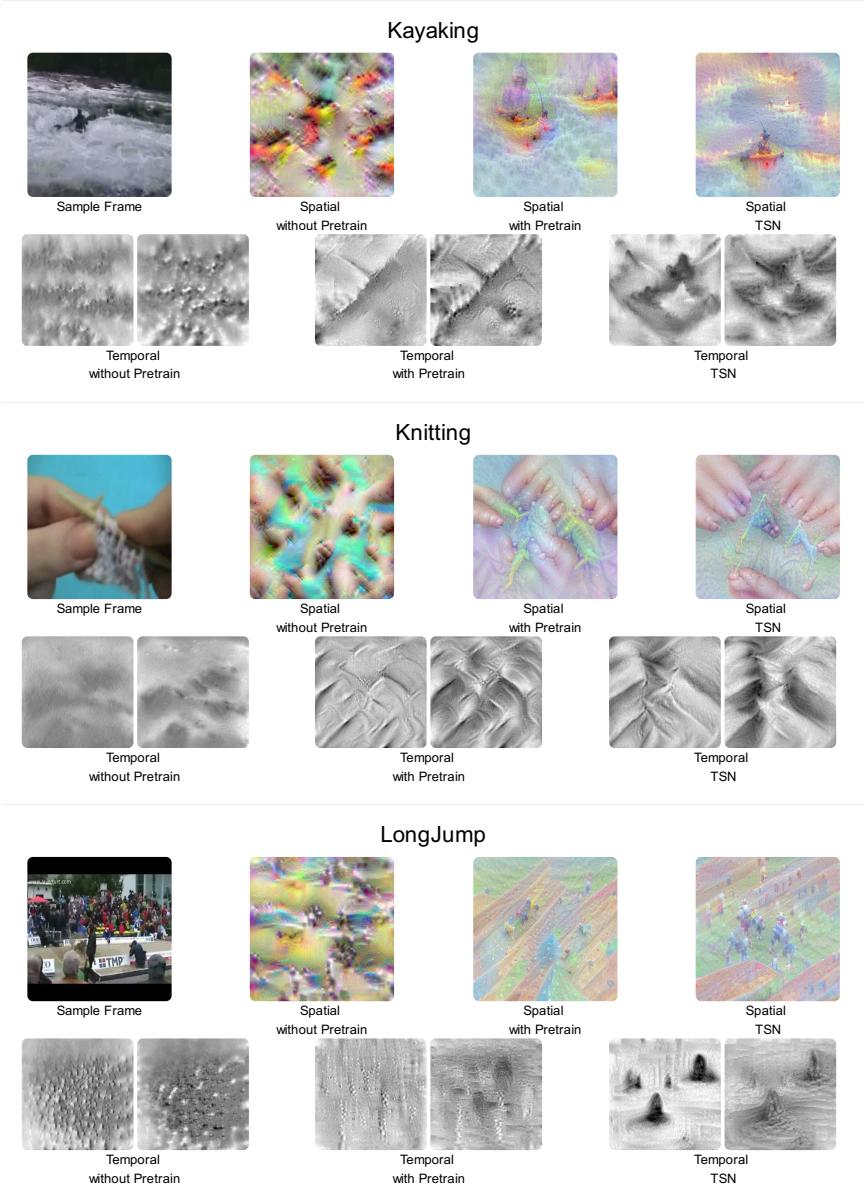

Fig. 17. Visualization.

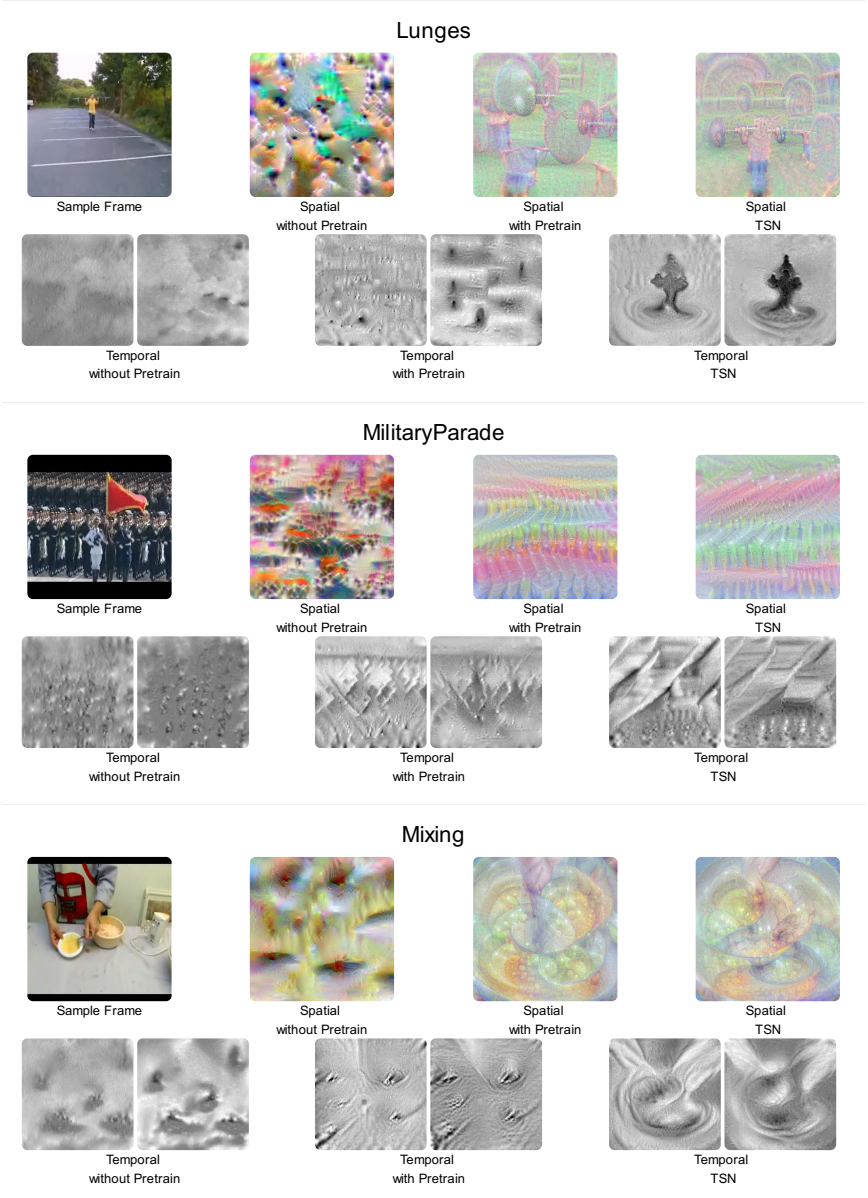

Fig. 18. Visualization.

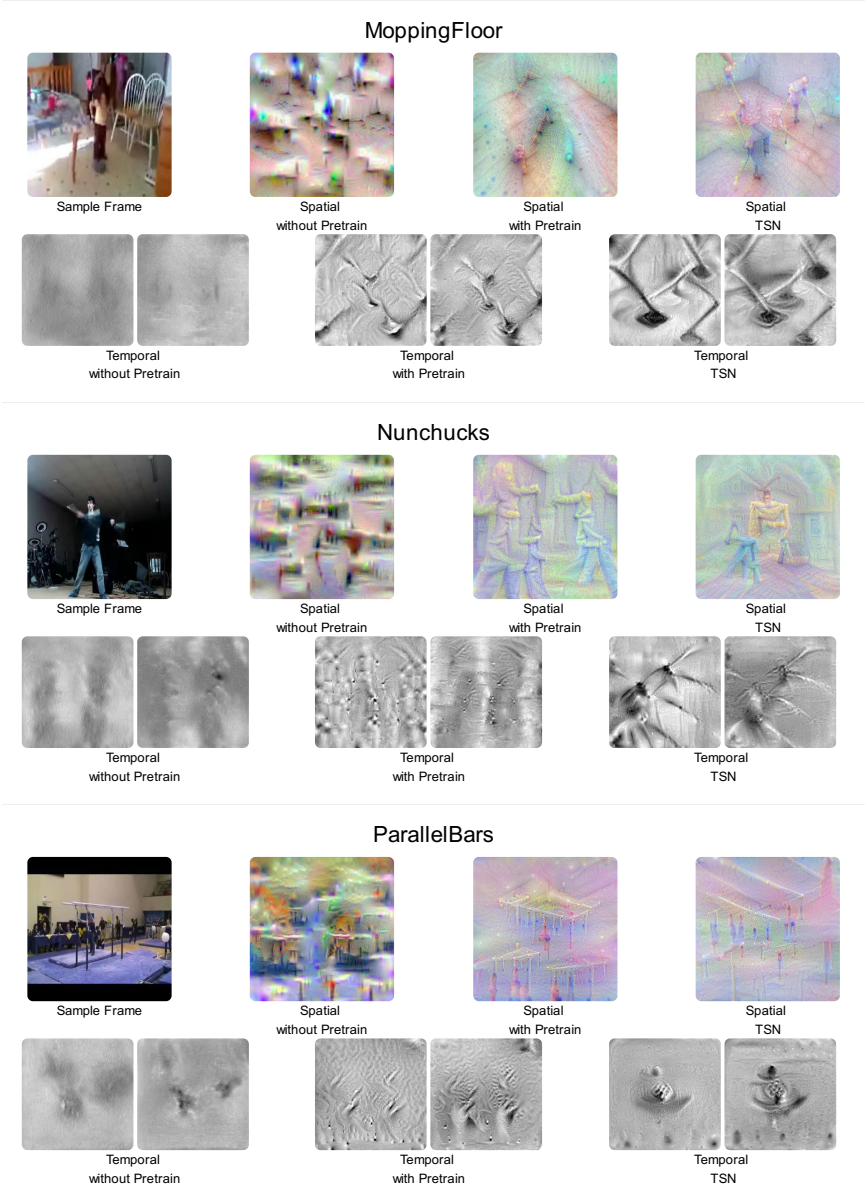

Fig. 19. Visualization.

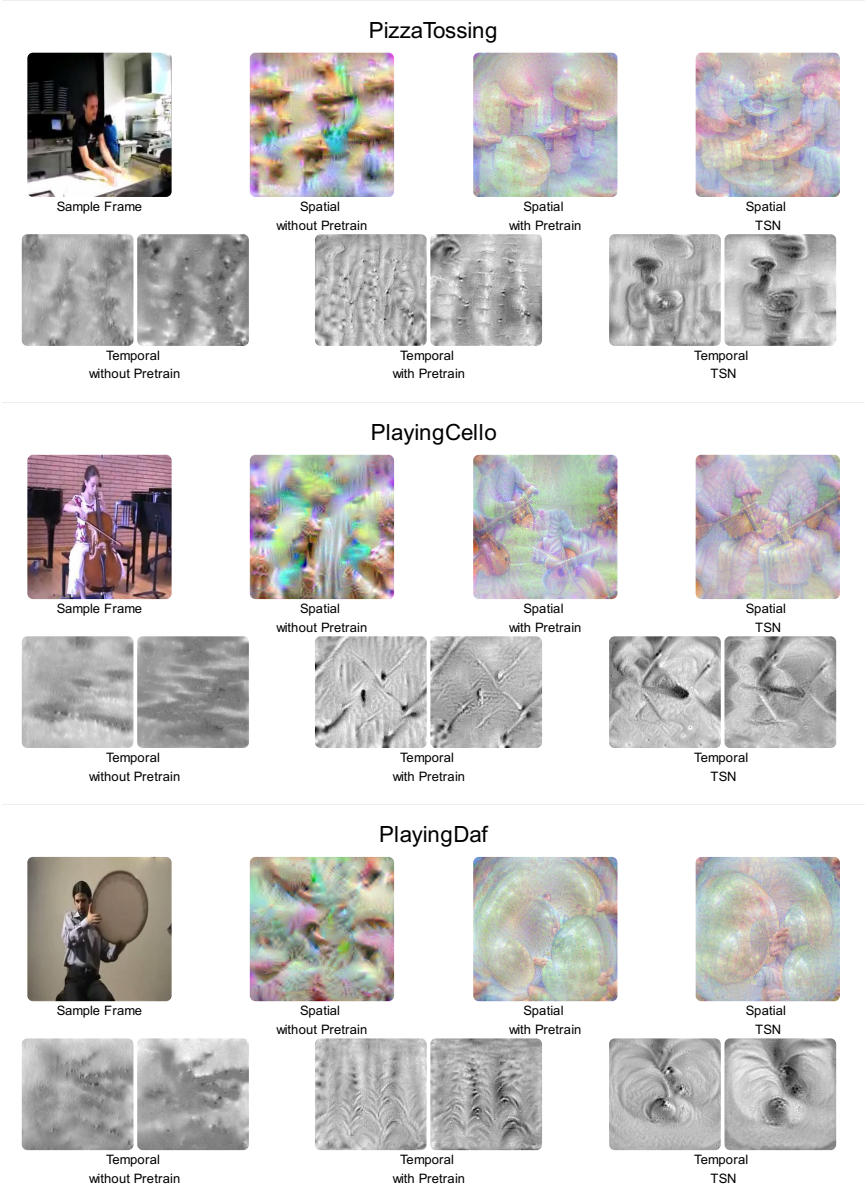

Fig. 20. Visualization.

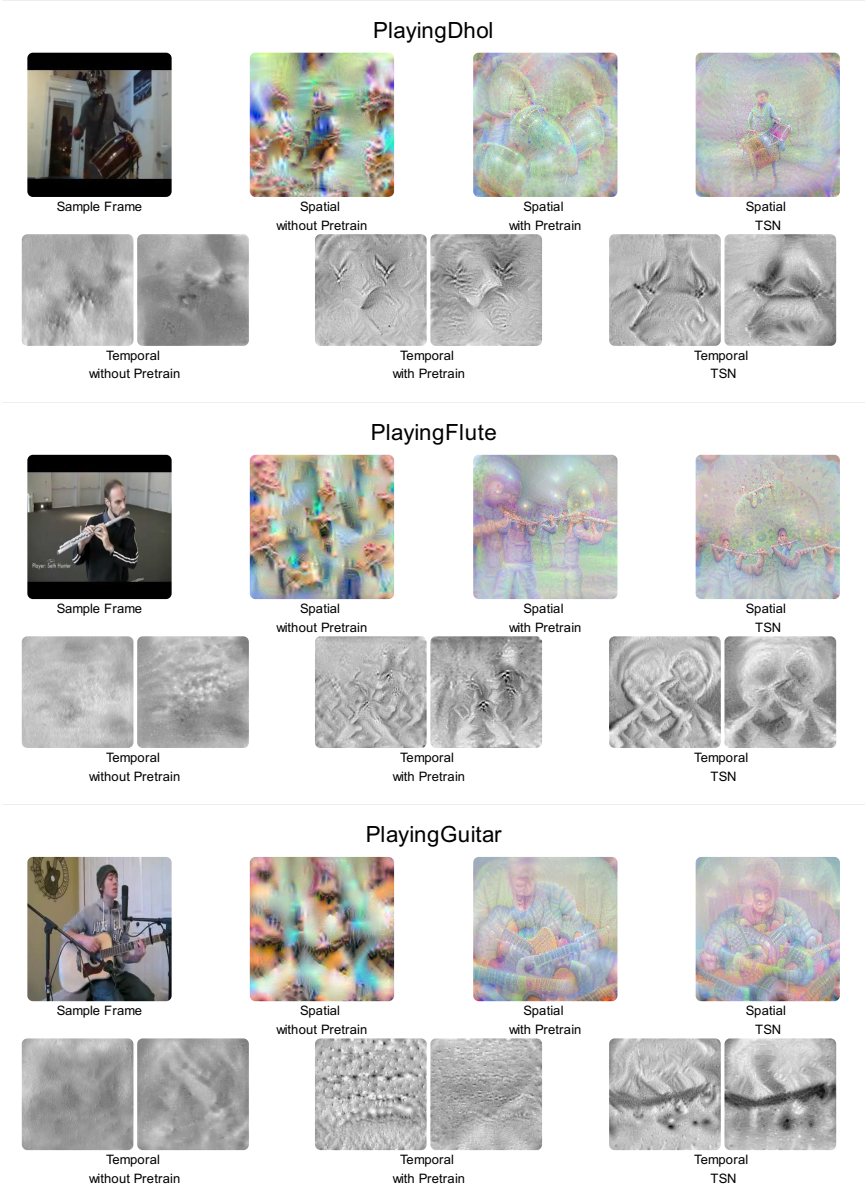

Fig. 21. Visualization.

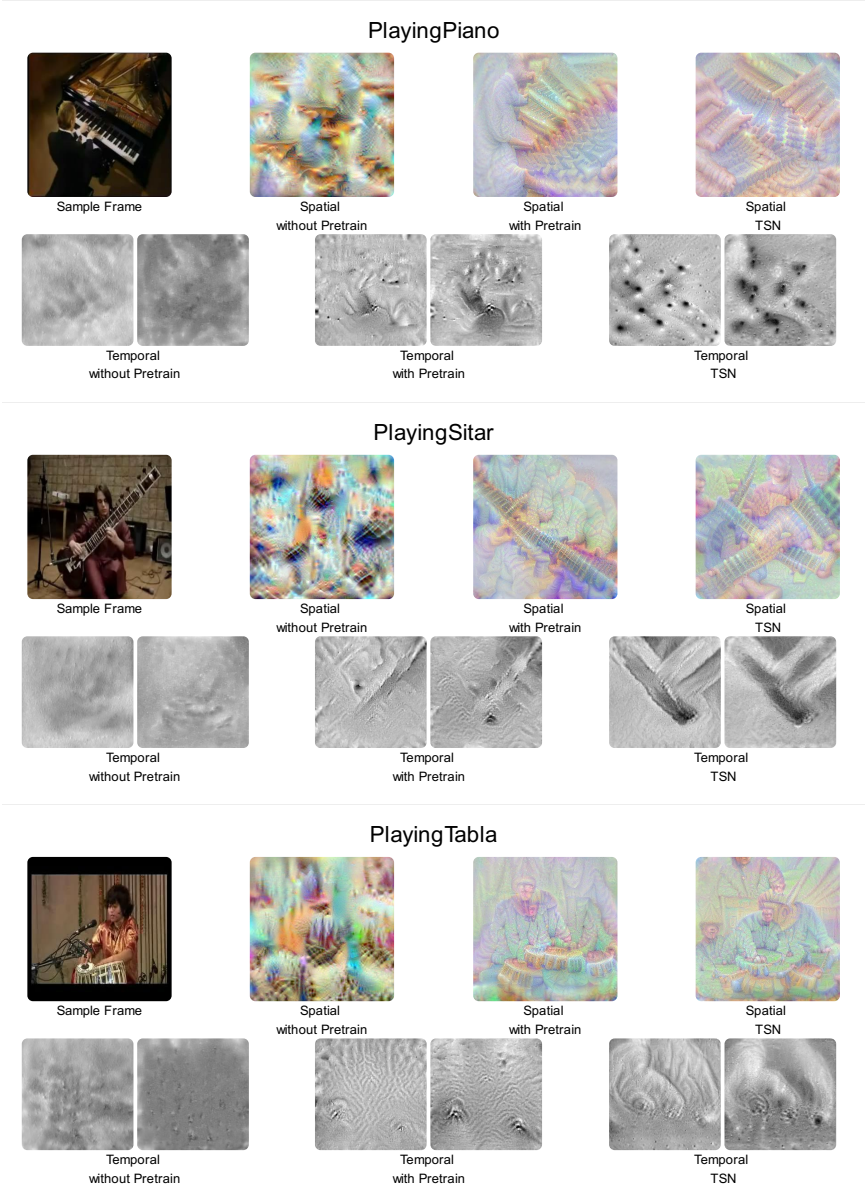

Fig. 22. Visualization.

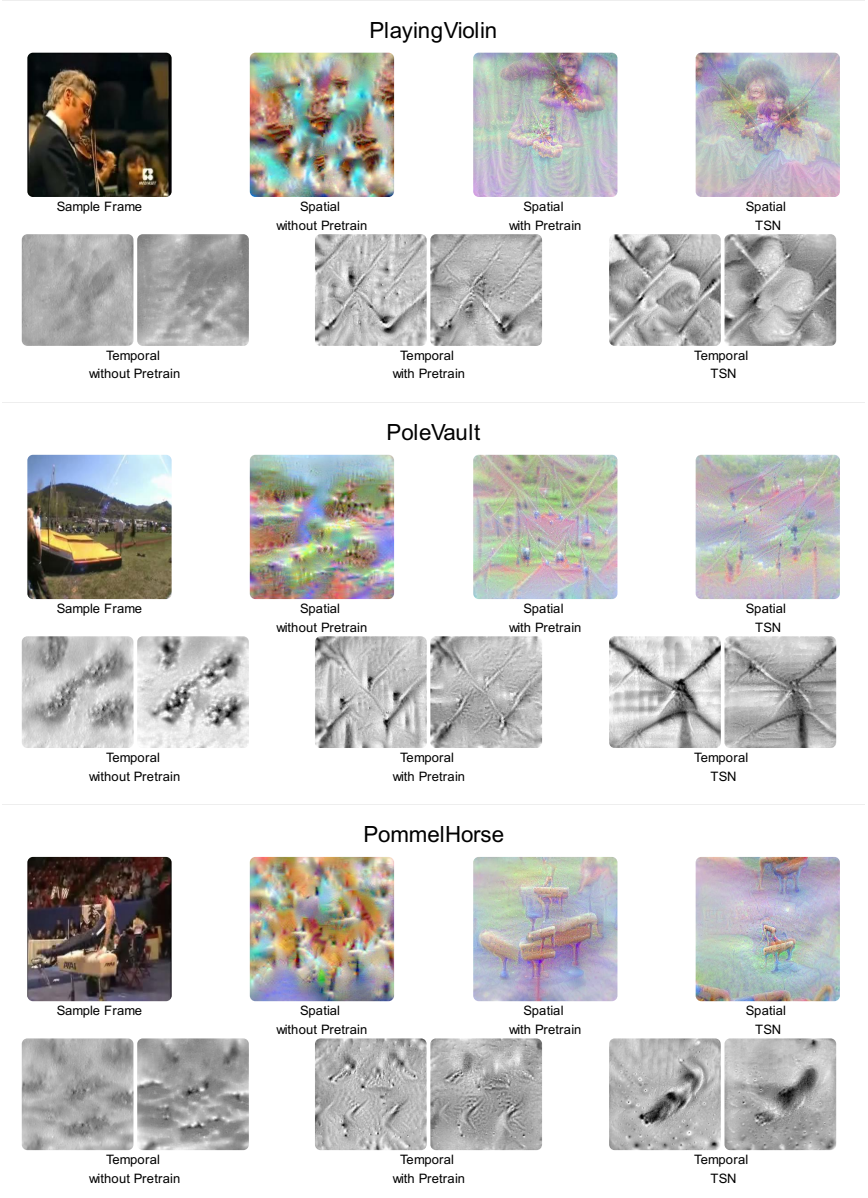

Fig. 23. Visualization.

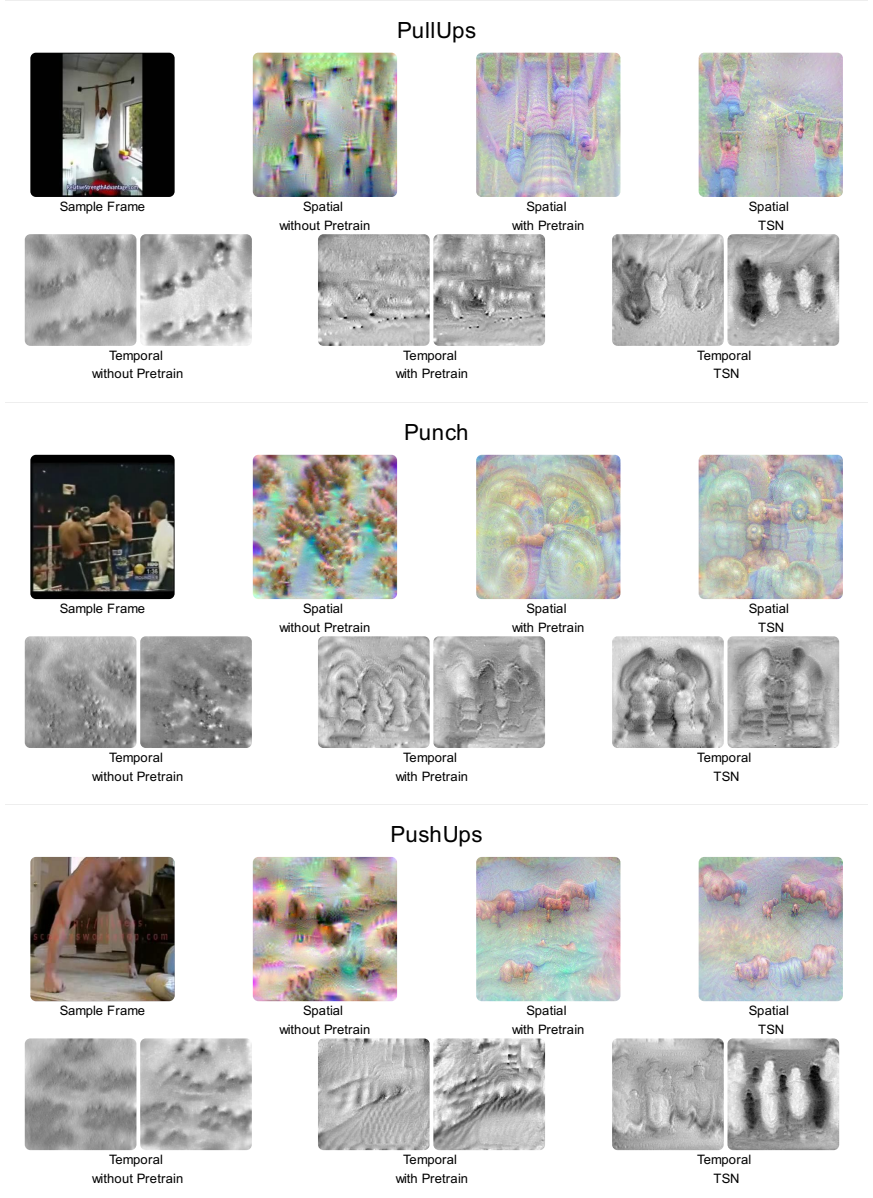

Fig. 24. Visualization.

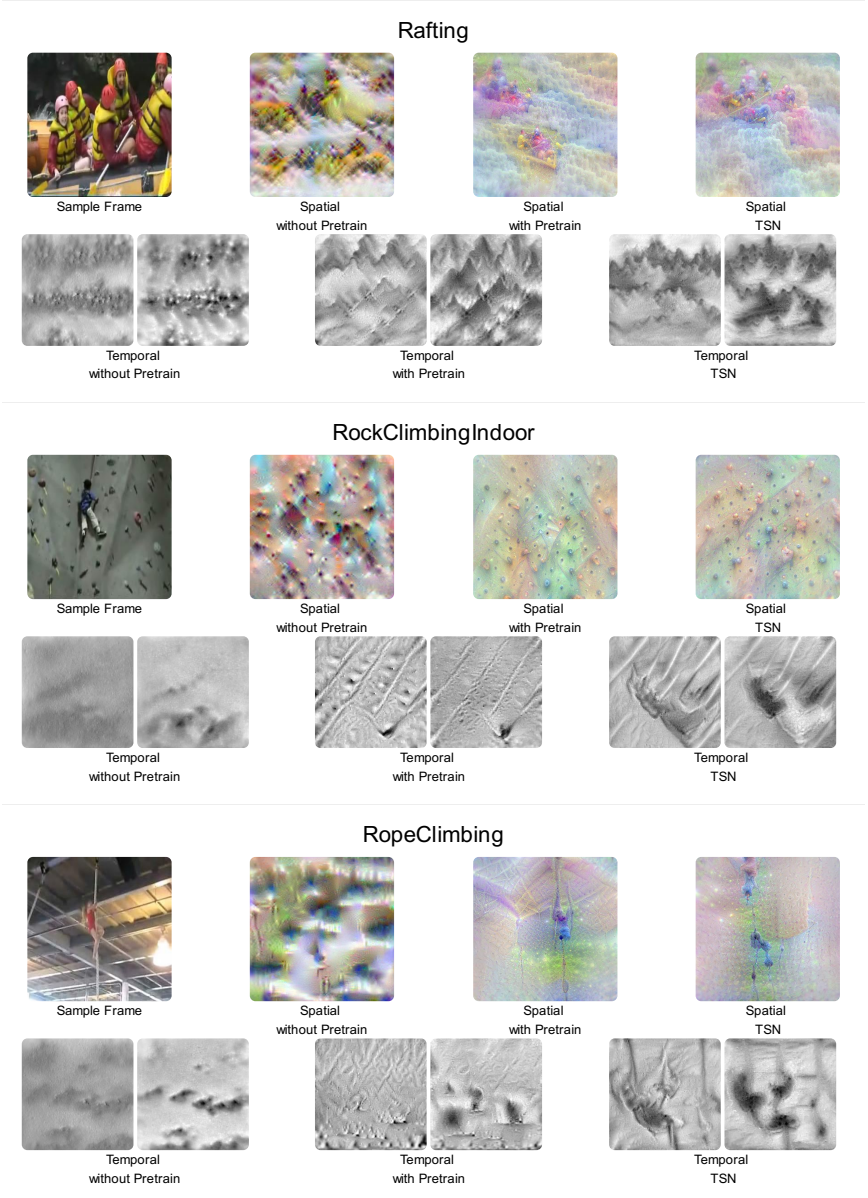

Fig. 25. Visualization.

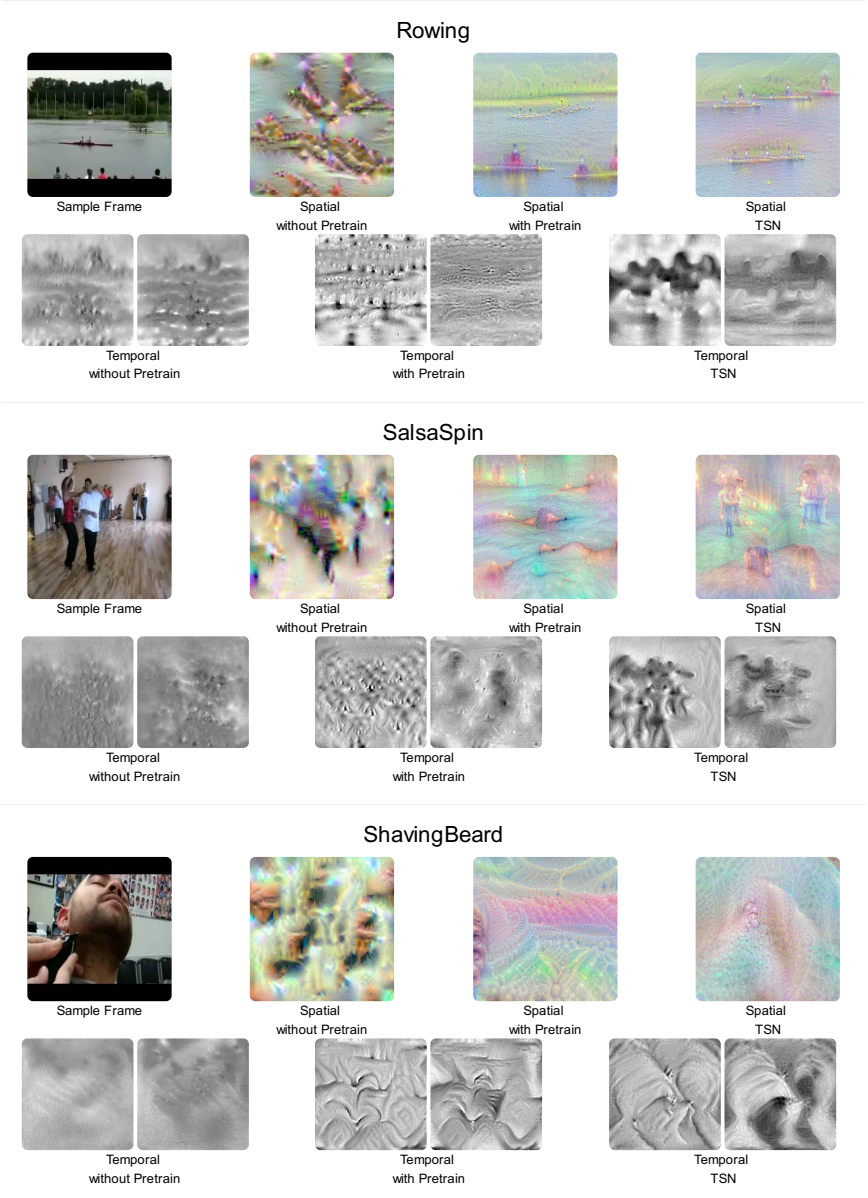

Fig. 26. Visualization.

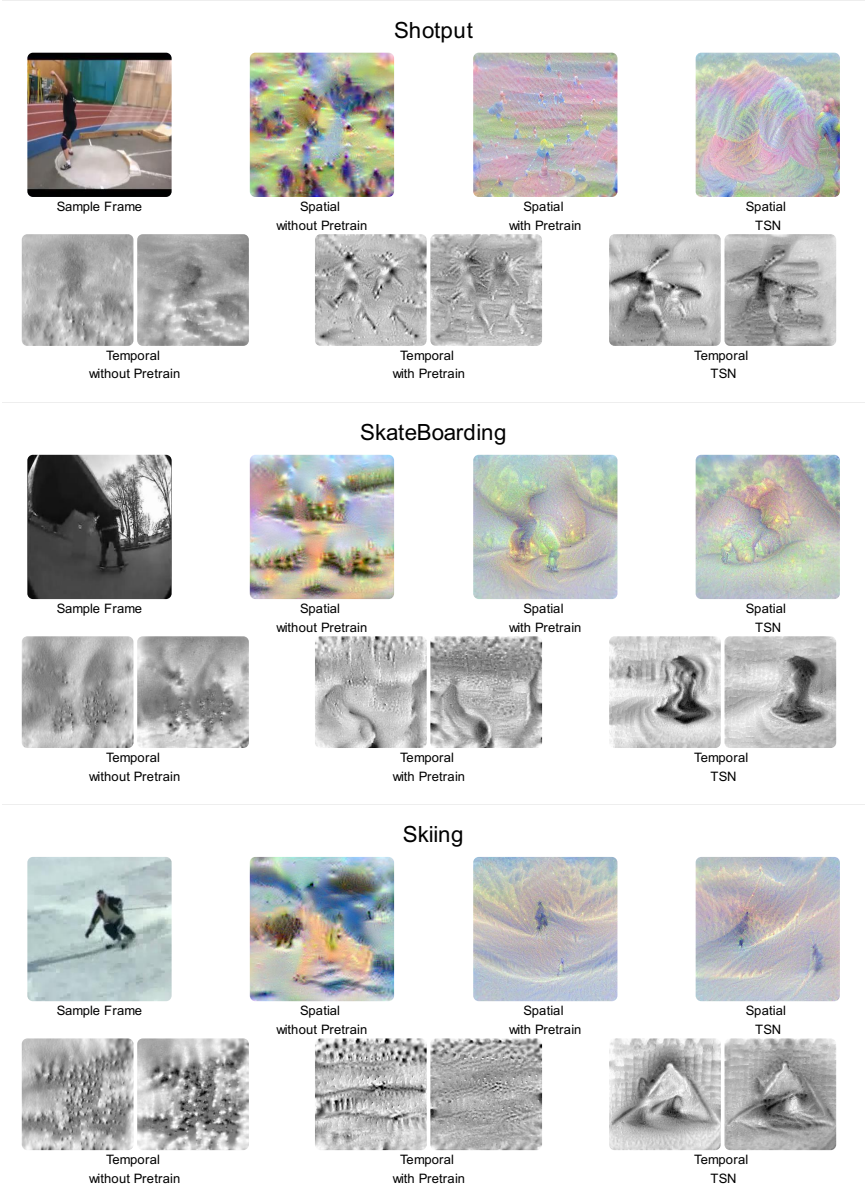

Fig. 27. Visualization.

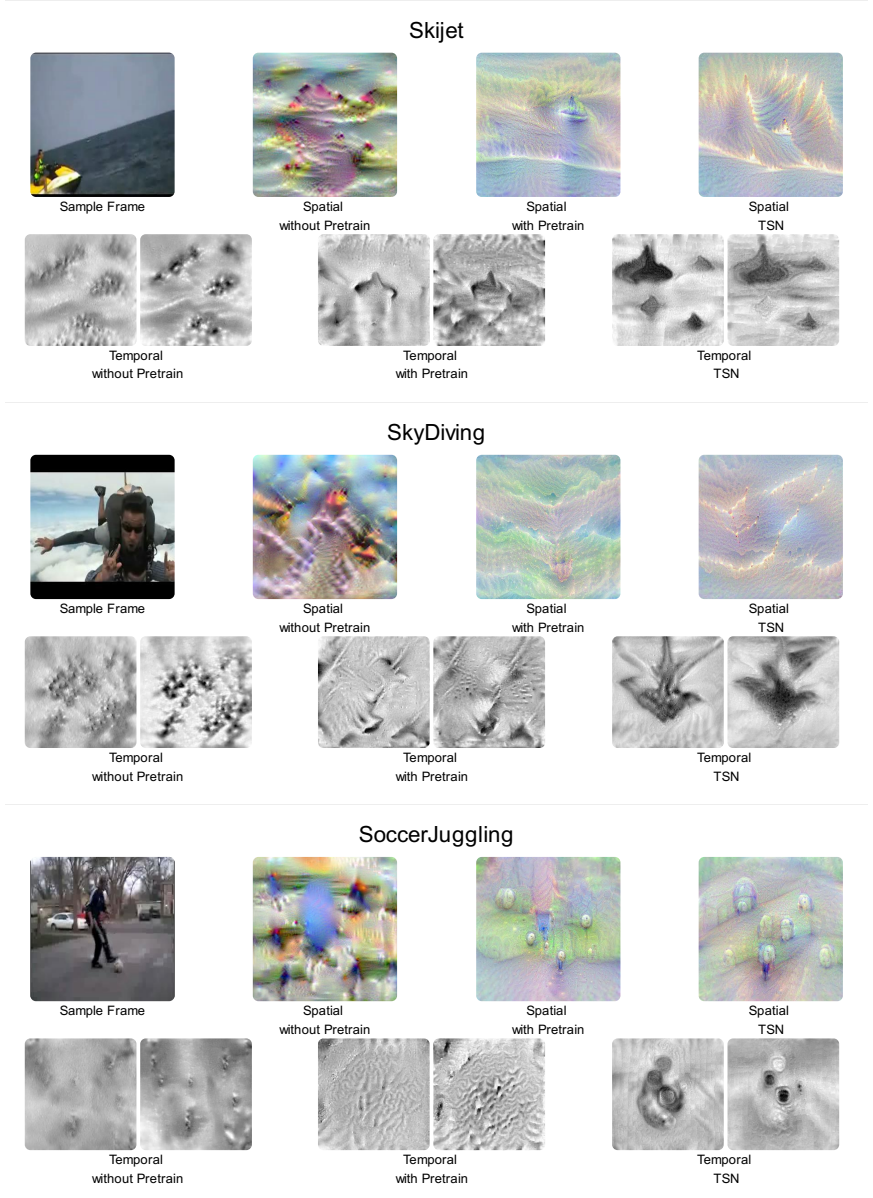

Fig. 28. Visualization.

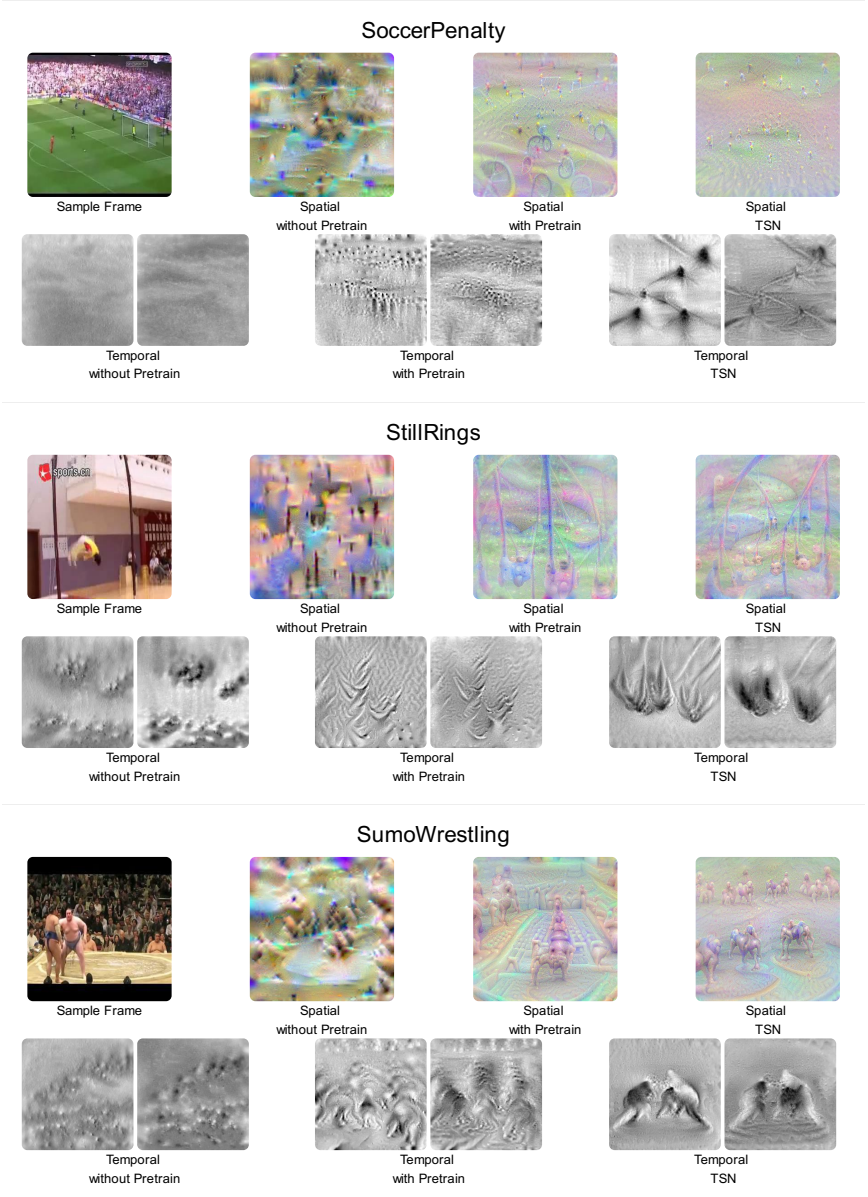

Fig. 29. Visualization.

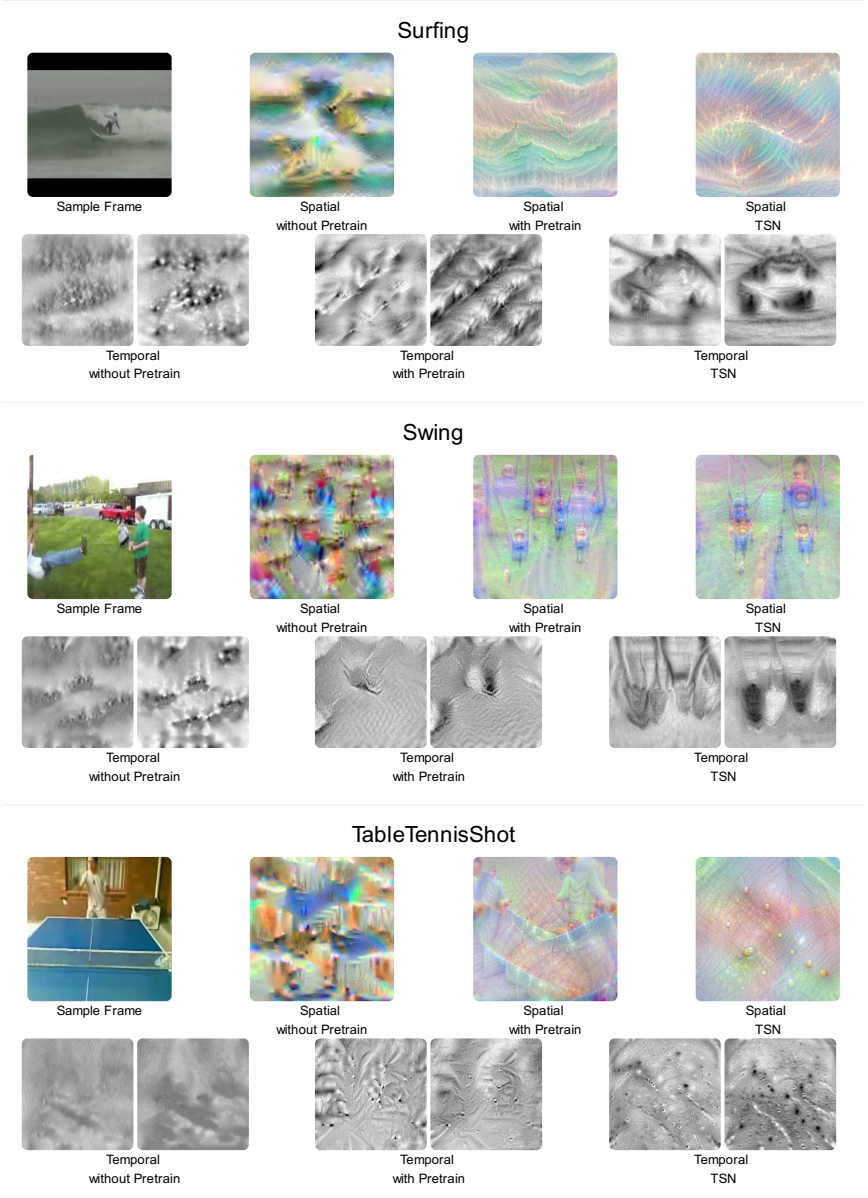

Fig. 30. Visualization.

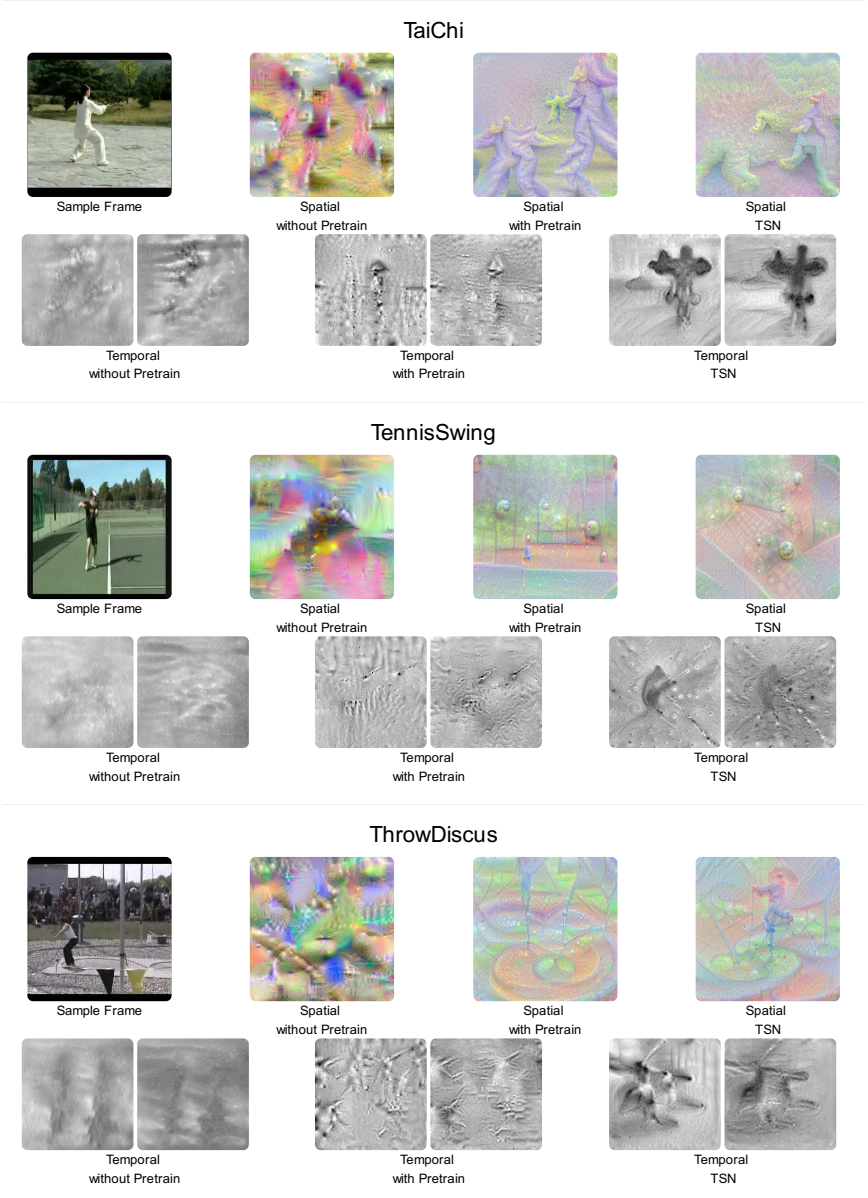

Fig. 31. Visualization.

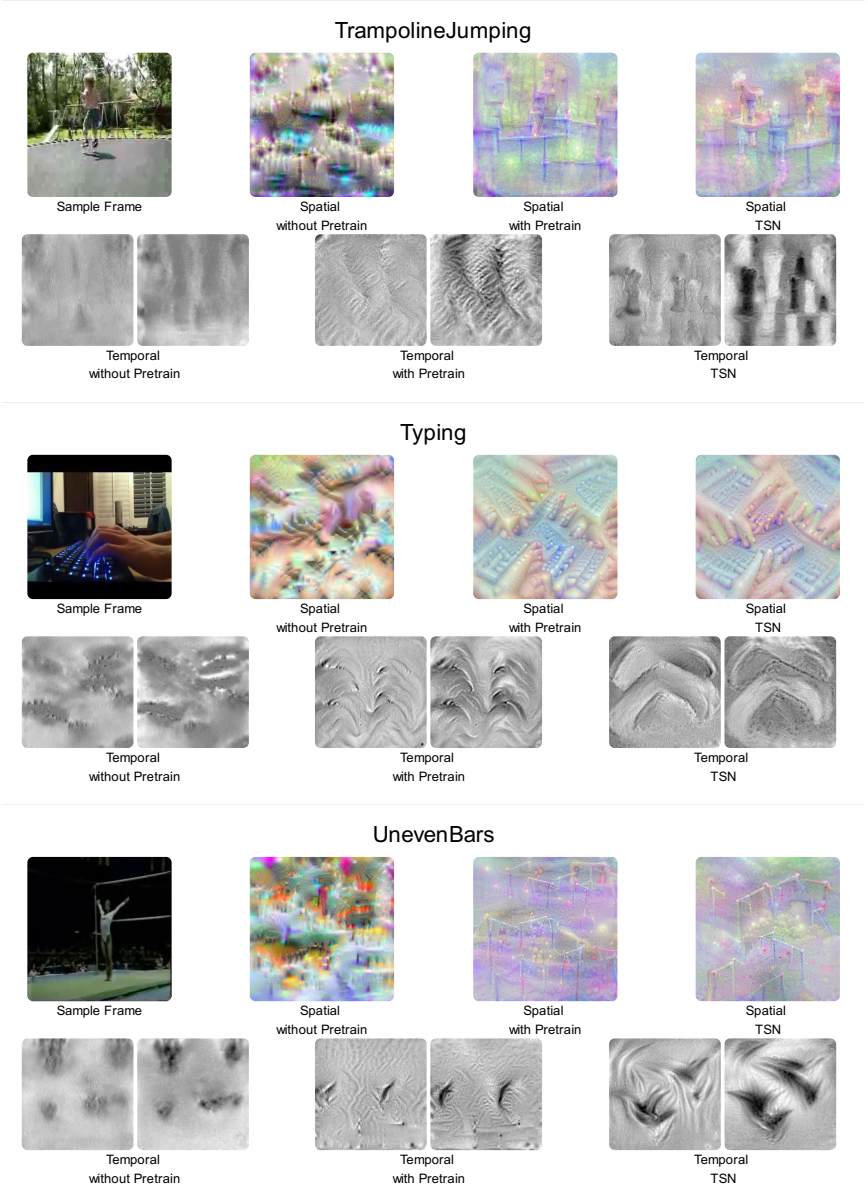

Fig. 32. Visualization.

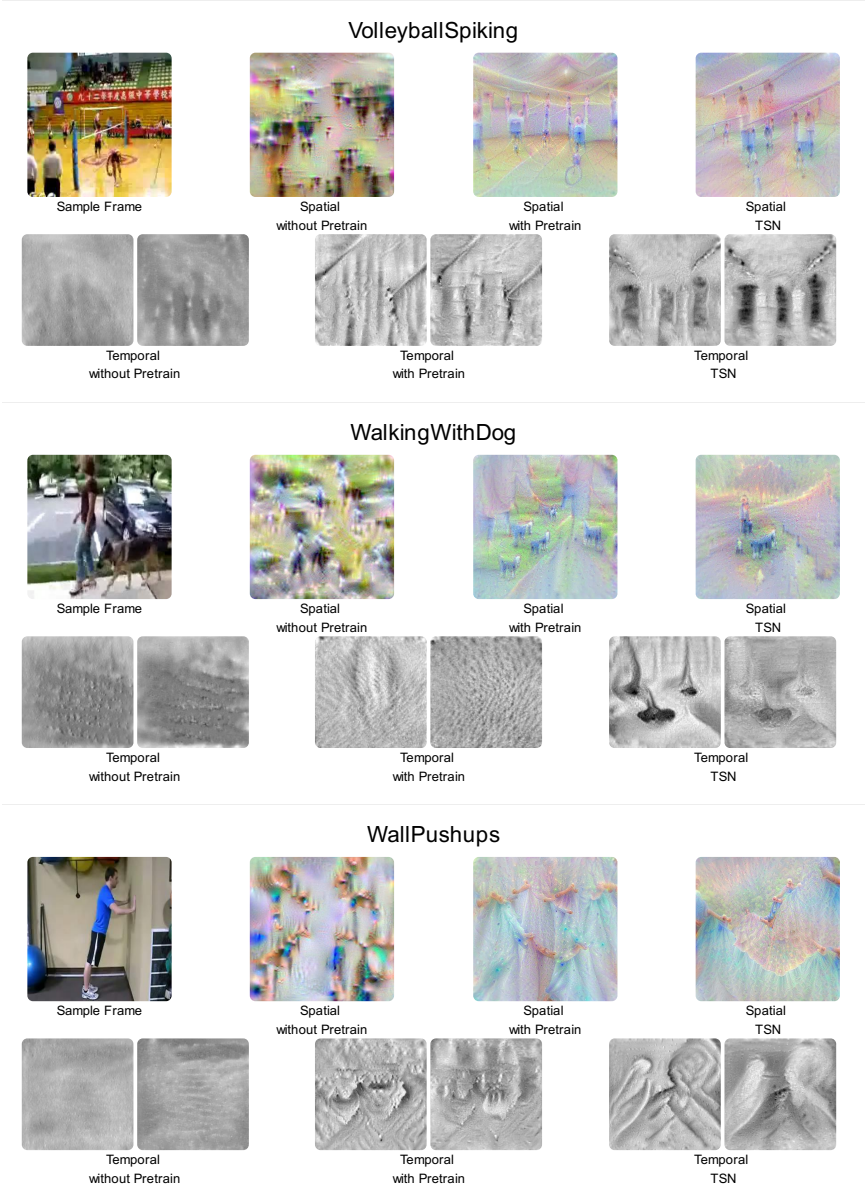

Fig. 33. Visualization.

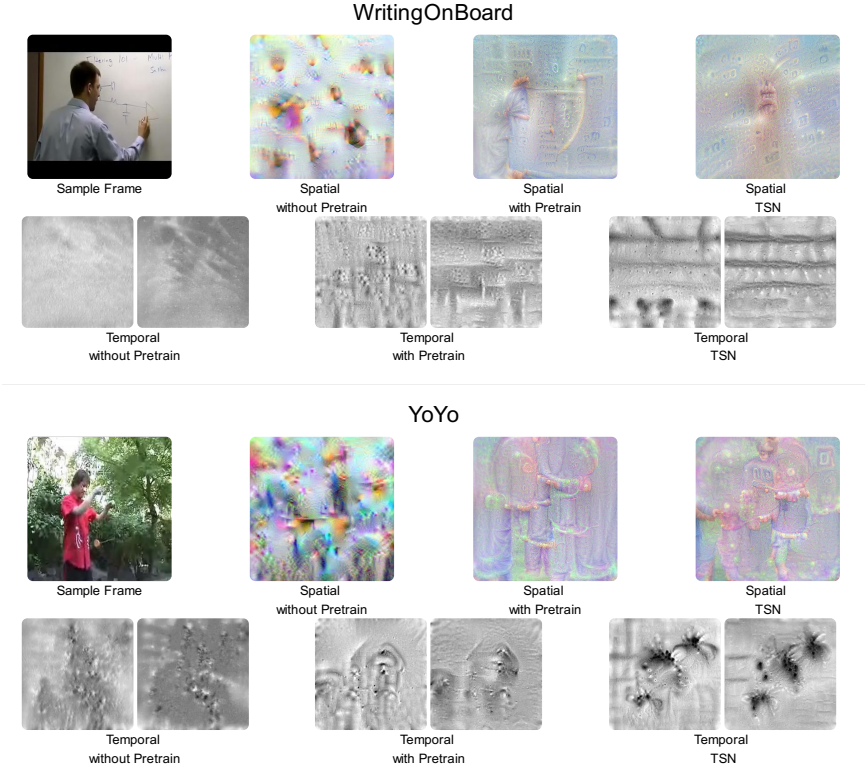

Fig. 34. Visualization.