Supplemental material for

Recur, Attend or Convolve? On Whether Temporal Modeling Matters for Cross-Domain Robustness in Action Recognition

Sofia Broomé[1] Ernest Pokropek[1] Boyu Li[1] Hedvig Kjellström[1,2]
[1] KTH, Sweden sbroome,pokropek,boyul,hedvig@kth.se [2] Silo AI, Sweden

Model	S1/V ↑	S2/V ↑	T/V ↓	T/S1 ↓
3D CNN	**0.297**	**0.316**	0.244	**0.824**
ConvLSTM	0.245	0.279	0.238	0.973
TimeSformer	0.211	0.206	**0.182**	0.864

Table 1. Results for experiment (e): 14M parameters, 30% validation accuracy.

Model	S1/V ↑	S2/V ↑	T/V ↓	T/S1 ↓
3D CNN	**0.279**	**0.303**	0.216	**0.777**
ConvLSTM	0.216	0.270	0.199	0.919
TimeSformer	0.155	0.162	**0.150**	0.972

Table 2. Results for experiment (f): 14M parameters, 35% validation accuracy.

1. Supplemental figures regarding the model concepts

Figure 1 highlights the conceptual differences between 3D convolution, self-attention and recurrence in terms of temporal modeling.

2. Plots for each model size on the Temporal Shape dataset

In the main article (Figure 3), the shaded area of standard error is across both model sizes and repeated runs with different seeds (meaning $10*5 = 50$ runs per model and domain). Detailed plots for each model size with five repeated runs each are shown in Figures 2-3.

3. Robustness ratios for training both on 2Dot and MNIST-bg

In the main article, robustness ratios vs. model size are only plotted when training on 2Dot. In Figure 4, we include results when training on MNIST-bg as well. We show the two plots next to each other for comparison.

4. Detailed results on Diving48

Tables 1-4 show tabularized results corresponding to Figure 6 in the main article (experiments e-h).

Model	S1/V ↑	S2/V ↑	T/V ↓	T/S1 ↓
3D CNN	**0.259**	**0.263**	0.191	**0.737**
ConvLSTM	0.183	0.224	**0.142**	0.776

Table 3. Results for experiment (g): 14M parameters, 40% validation accuracy.

Model	S1/V ↑	S2/V ↑	T/V ↓	T/S1 ↓
3D CNN	**0.267**	**0.258**	0.190	0.713
ConvLSTM	0.219	0.245	**0.143**	**0.653**

Table 4. Results for experiment (h): 14M parameters, 45% validation accuracy.

5. Qualitative examples on Diving48

Here, we include the top-1 and top-5 accuracies tables corresponding to the qualitative examples of classes 12, 22 and 45 shown in Table 7 in the main article. In Tables 5, 6 and 7, the trends regarding the top-1 and top-5 accuracy on the different datasets are slightly less clear. We observe that in Tables 5 and 7, ConvLSTM and TimeSf drop the clearest in top-5 performance on T relative to S1 and S2. On the other hand, in Table 6 (Class 22), the top-5 accuracy is relatively improved on T compared to S1 and S2 for ConvLSTM and the 3D CNN, whereas TimeSf is unchanged. We inspected these clips, to verify that the segmentation had not failed, which it had not. However, the ConvLSTM is still the only one out of the three to have 20% in top-1 accuracy both for S1 and S2 on class 22, dropping to 0 in top-1 on T (Table 6). Last, for class 45, the ConvLSTM has the best results on S1 and S2 (20% top-5 accuracy) out of the three models, where the others have 0% accuracy, except for 20% top-5 accuracy for the 3D CNN on the texture dataset.

	S1		S2		T	
Model	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
ConvLSTM	0.0	0.2	0.0	0.2	0.0	0.0
3D CNN	0.2	0.2	0.2	0.2	0.2	0.2
TimeSf	0.2	0.6	0.2	0.4	0.0	0.2

Table 5. Qualitative example with predictions on five random clips from class 12, made by the model instances from experiment c) (38.3% acc.).

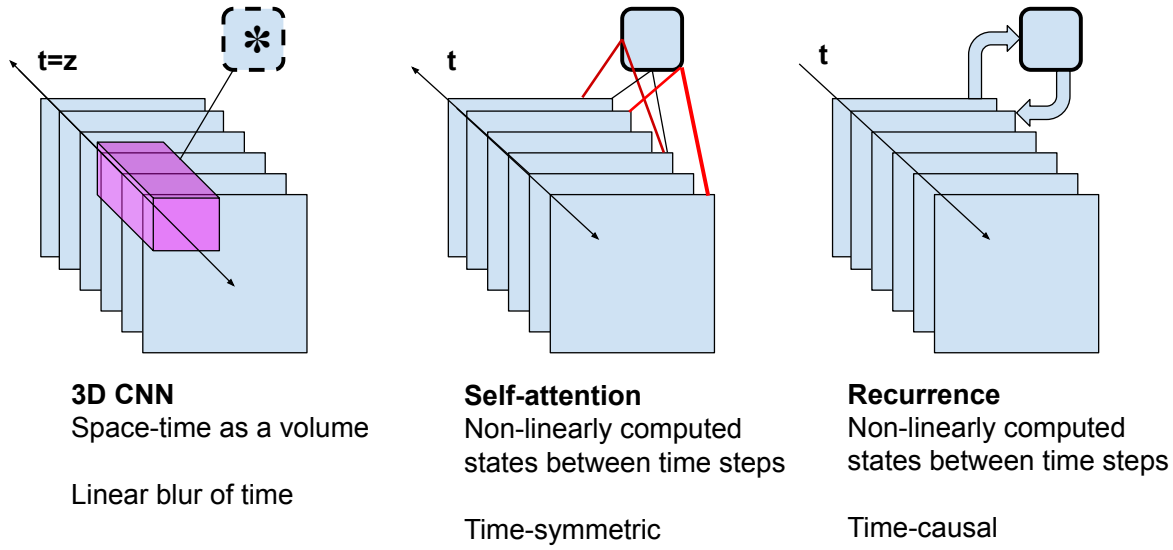

Figure 1. An overview of the conceptual differences in terms of frame dependency modeling between 3D convolutions, self-attention and recurrence.

	S1		S2		T	
Model	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
ConvLSTM	0.2	0.2	0.2	0.2	0.0	0.6
3D CNN	0.0	0.4	0.2	0.6	0.0	0.8
TimeSf	0.0	0.2	0.0	0.2	0.0	0.2

Table 6. Class 22, same table structure as Table 5.

	S1		S2		T	
Model	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
ConvLSTM	0.0	0.2	0.0	0.2	0.0	0.0
3D CNN	0.0	0.2	0.0	0.0	0.0	0.2
TimeSf	0.0	0.0	0.0	0.0	0.0	0.0

Table 7. Class 45, same table structure as Table 5.

6. Dataset details

6.1. Sampling with Replacement in the Temporal Shape Dataset

In the experiments, 4000 clips were used for training and 1000 for validation. The number of samples was chosen so as to be able to sample randomly with replacement, while still keeping the risk low that an identical clip occurs in both the training and the validation set. For the 2Dot-domain, each class has more than 30k possible variations (lower bounds: 31k circle, 34k line, 51k rectangle, 150k arc), except the spiral class which has 7200 as a lower bound on the possible variations. When the training set consists of 5000 samples in total, we generate around 1000 samples per class. For the spiral class, a frequentist estimation gives that $800/7200 = 0.11$ of the 200 spiral validation samples might be present in the training split (22 clips). However, this is still an over-estimation, since the spirals sometimes bounce against the sides of the frame which gives rise to extra variation. We decided to consider this as acceptable noise of the dataset. Some amount of data leakage can be considered interesting since this may occur in standard datasets as well.

6.2. Instance Segmentation of Diving48

To segment divers, it did not suffice to apply a pre-trained network and use the class "Person", which we first attempted (DeeplabV3 pre-trained on MS-COCO, provided by PyTorch). First of all, the off-the-shelf model could often not recognize the divers in the air as the "Person" class – they can be up-side down, or assume strange shapes in the air. Secondly, the model would often detect pixels of the "Person" class in the audience, when there was audience visible, which we, naturally, did not want to include.

Thus, we resorted to labelling our own segmented frames from the dataset (no segmentation masks were available online). We manually labelled 303 frames from the dataset containing one or two divers, picked from 303 randomly chosen videos of the training split. When there were two divers, we segmented each as its own instance. The segmentation masks will be made public.

We fine-tuned a MaskRCNN on our labeled dataset, using a random split of 290 frames as training set and 13 frames to validate, and monitored the bounding box IoU on the validation set. The best model achieved 93% validation bounding box IoU, which we used to segment the frames of the entire dataset (at 32 frames per clip). We used the confidence of the mask predictions as a threshold. The non-zero predictions were mostly confined to a bounding box surrounding the diver(s). When the threshold was $t = 0$, bounding boxes around the divers were used as crops (S2).

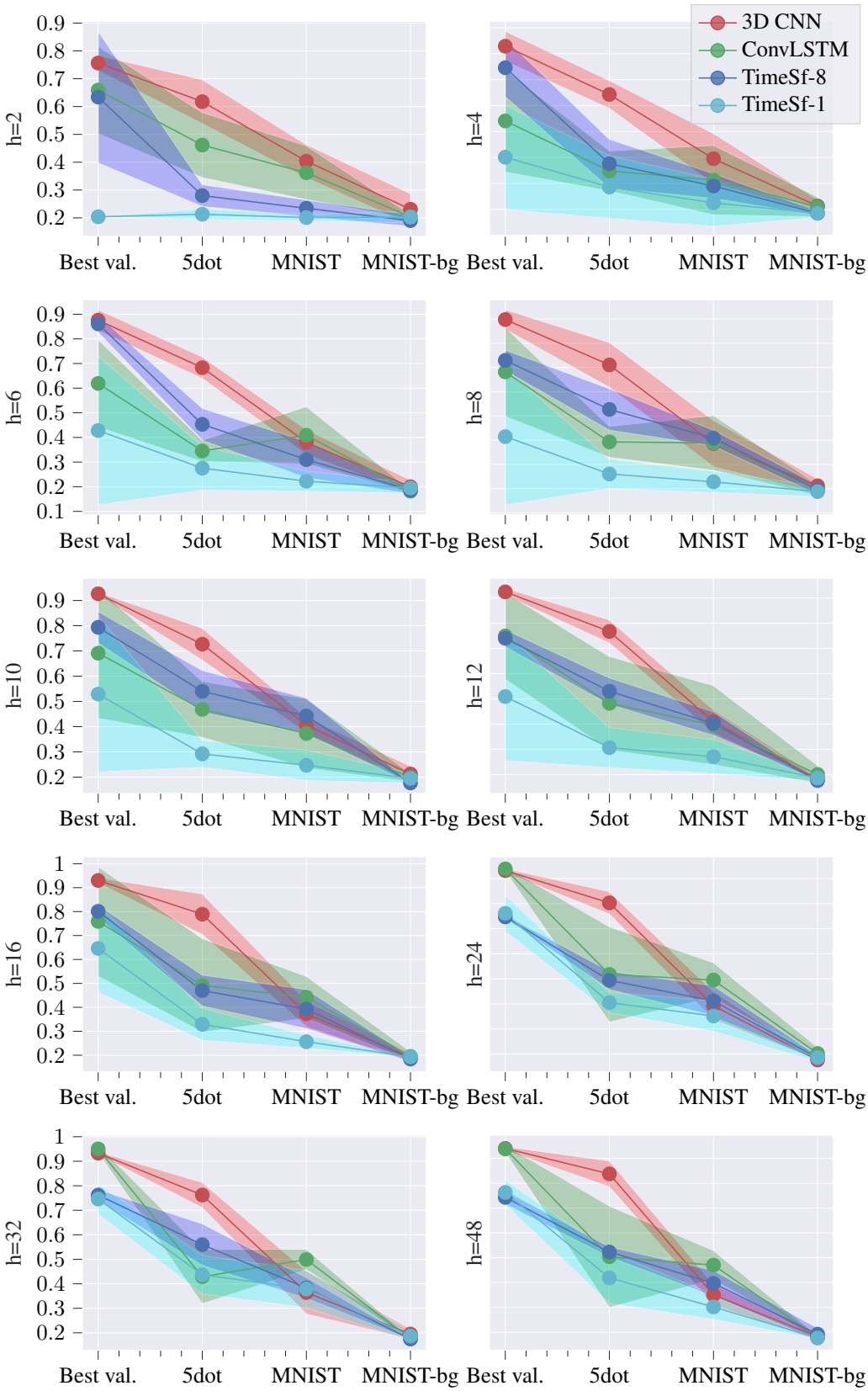

Figure 2. Average results (% acc.) across ten trials with varying numbers of hidden units per layer, repeated five times each. Training and validation on the 2Dot domain. The shaded area corresponds to standard deviation across the trials.

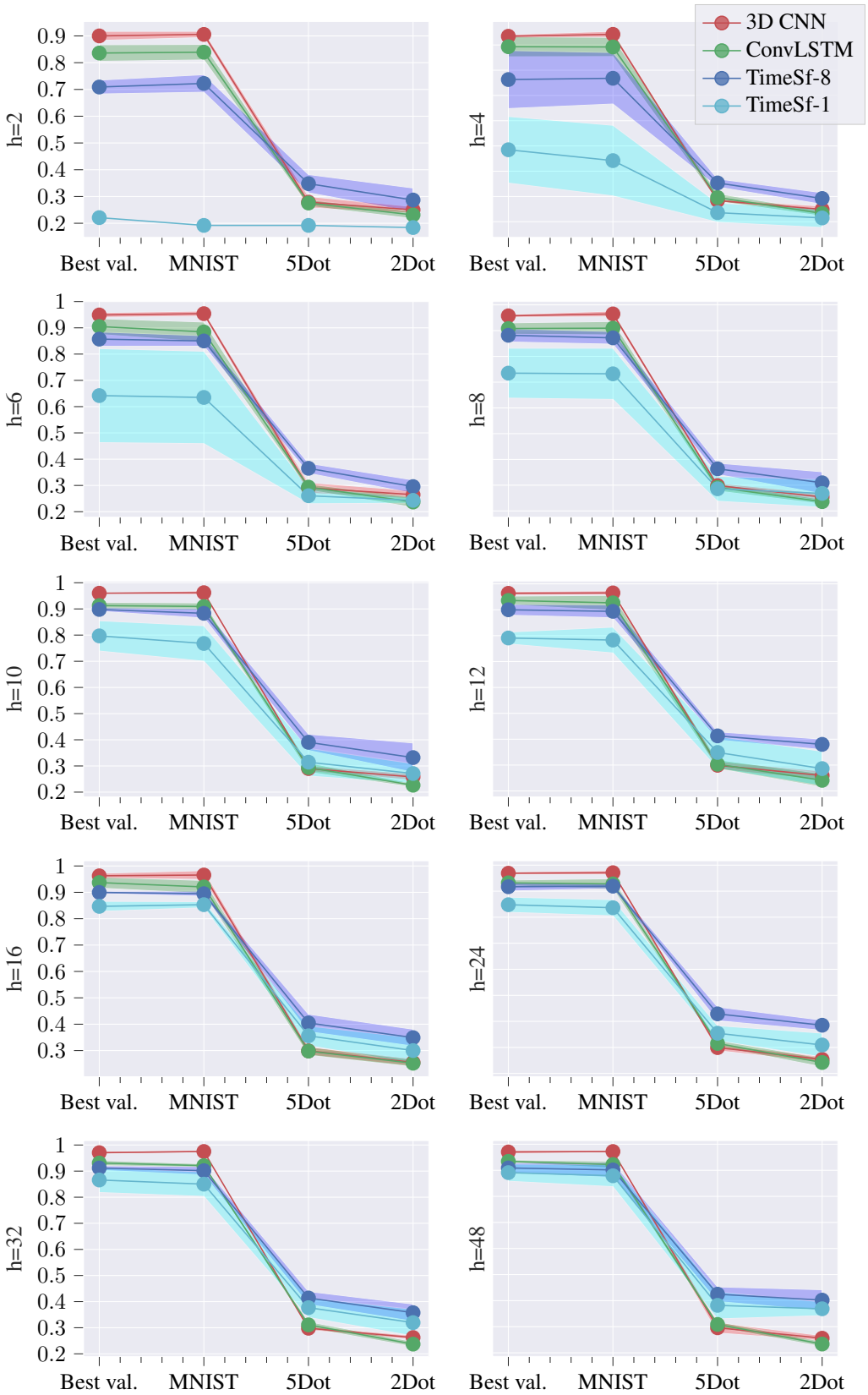

Figure 3. Average results (% acc.) across ten trials with varying numbers of hidden units per layer, repeated five times each. Training and validation on the MNIST-bg domain. The shaded area corresponds to standard deviation across the trials.

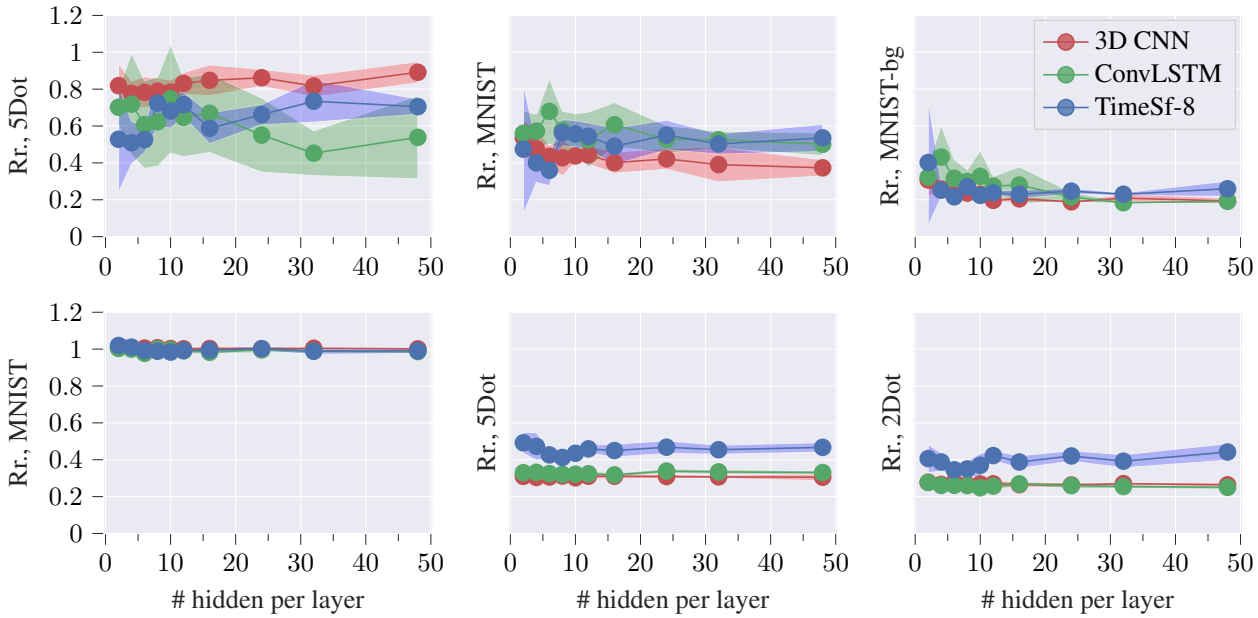

Figure 4. Robustness ratio (rr.) (↑) when training on 2Dot (top – same as in the main article for comparison) and on MNIST-bg (bottom), vs. number of hidden units per layer. The target domain is progressively further away from the source in subplots a-c. TimeSf-1 is excluded here due to its near random validation accuracy for small model sizes.

When increased to $t = 0.4$, we obtained proper segments of the diver shape (S1). The frames contain a lot of motion blur which made the segmentation more challenging, and the segmentation at $t = 0.4$ is not perfect – sometimes parts of for example an arm or foot is missing. The performance of the segmentation at $t = 0.4$ was deemed sufficient after manual inspection of 100 randomly chosen videos, where all videos had enough evidence to recognize the development of the dive. The segmentation at $t = 0$ (bounding boxes, S2) was satisfactory in all 100 clips inspected.

7. Parameter count

Table 8 shows the number of parameter for the various architectures used in the Temporal Shape experiments.

8. TimeSformer variants attempted for training

Table 10 lists the different variants we tested when training on Diving48 from scratch. In all variants, the number of heads was 8 ($\mathcal{A} = 8$), the patch size was 16×16, the learning rate was fixed at 0.001, and the weight decay was 0.00001. When SGD was used, the momentum was always 0.9.

9. Model specifications for the Diving48 experiments

Table 9 lists the different model specifications for each of the eight experiments a-h on Diving48 in the main article. For further details on the models, this is described in the main article and in the code repository.

References

[1] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is Space-Time Attention All You Need for Video Understanding? In *Proceedings of the International Conference on Machine Learning (ICML)*, July 2021.

Table 8. List of the number of trainable parameters for each model at each of the ten expreriments on Temporal Shape, where the model complexity was increased (the number of hidden units per layer, for three-layer models). TimeSformer-8 and TimeSformer-1 designates $\mathcal{A} = 8$ or $\mathcal{A} = 1$, respectively, i.e., the number of attention heads per layer.

	Nb. parameters			
# hidden per layer	**3D CNN**	**ConvLSTM**	**TimeSformer-8**	**TimeSformer-1**
2	1573	1497	20451	877
4	3573	4429	71557	2229
6	6005	8801	153413	4061
8	8869	14613	265989	6373
12	15893	30557	583301	12437
16	24645	52261	1023493	20421
24	47333	112949	2272517	42149
32	76933	196677	4013061	71557
48	156869	433253	8968709	153413

Table 9. List of the model variants used in the experiments a-h for Diving48. For the 3D CNN and ConvLSTM, the [x,y,z] lists designate the number of hidden units per layer (x for the first layer, y for the second, z for the third, etc), and the filter sizes lists similarly correspond to the filter size per layer.

Experiment	**3D CNN**	**ConvLSTM**	**TimeSformer**
a	Hidden [128,128,128,128], Filter sizes [7,7,5,3]	Hidden [128,128,128,128], Filter sizes [7,7,5,3]	Depth=4, $D = 1024$, $D_h = 128$
b	Hidden [128,128,128,128], Filter sizes [7,7,5,3]	Hidden [128,128,128,128], Filter sizes [7,7,5,3]	Depth=4, $D = 1024$, $D_h = 128$
c	Hidden [128,128,128,128], Filter sizes [7,7,5,3]	Hidden [128,128,128,128], Filter sizes [7,7,5,3]	Depth=4, $D = 1024$, $D_h = 128$
d	Hidden [32,64,128,128,128,256, 256,256,512,512,512] Filter sizes [5,3,3,3,3,3,3,3,3,3,3]	Hidden [128,128,128,128], Filter sizes [7,7,5,3]	-
e	Hidden [128,128,128,128,128,128], Filter sizes [7,7,7,5,3,3]	Hidden [128,128,128,128], Filter sizes [7,7,5,3]	Depth=11, $D = 256$, $D_h = 32$
f	Hidden [128,128,128,128,128,128], Filter sizes [7,7,7,5,3,3]	Hidden [128,128,128,128], Filter sizes [7,7,5,3]	Depth=11, $D = 256$, $D_h = 32$
g	Hidden [128,128,128,128,128,128], Filter sizes [7,7,7,5,3,3]	Hidden [128,128,128,128], Filter sizes [7,7,5,3]	-
h	Hidden [128,128,128,128,128,128], Filter sizes [7,7,7,5,3,3]	Hidden [128,128,128,128], Filter sizes [7,7,5,3]	-

Table 10. List of attempted TimeSformer variants, trained from scratch on Diving48. D and D_h are parameters in the TimeSformer [1] architecture, attn. do. and ff.do are attention dropout and feed-forward network dropout, T is the number of uniformly sampled frames that constitute the clip, and addditional ll. means an additional linear layer on top of the predictions output from the TimeSformer model.

Best val.	Ep.	D	D_h	Depth	Attn. do.	Ff. do.	T	Batch size	Optimizer	Additional ll.	Patience
32.7	88	512	64	12	0	0	8	8	SGD	1	30
31.5	84	512	64	12	0	0	8	8	SGD	0	30
36.1	78	512	64	3	0	0	32	8	SGD	0	30
39.7	122	1024	128	4	0	0	32	8	SGD	0	30
31.1	76	512	64	12	0.1	0.1	8	8	SGD	0	30
31.7	71	256	32	11	0	0	8	8	SGD	0	30
36.5	85	256	32	11	0	0	32	8	SGD	0	30
19.0	79	256	32	11	0	0	32	8	Adam	0	30
31.7	75	256	32	11	0	0	8	32	Adam	0	30
32.4	133	256	32	11	0	0	8	48	SGD	0	30
36.5	85	256	32	11	0	0	32	8	SGD	0	75