Interpreting Video Features: A Comparison of 3D Convolutional Networks and Convolutional LSTM Networks

Joonatan Mänttäri*[0000−0003−2171−1429], Sofia Broomé*[0000−0001−5458−3473], John Folkesson[0000−0002−7796−1438], and Hedvig Kjellström[0000−0002−5750−9655]

Robotics, Perception and Learning, KTH Royal Institute of Technology, Sweden
{manttari,sbroome,johnf,hedvig}@kth.se

Abstract. A number of techniques for interpretability have been presented for deep learning in computer vision, typically with the goal of understanding what the networks have based their classification on. However, interpretability for deep video architectures is still in its infancy and we do not yet have a clear concept of how to decode spatiotemporal features. In this paper, we present a study comparing how 3D convolutional networks and convolutional LSTM networks learn features across temporally dependent frames. This is the first comparison of two video models that both convolve to learn spatial features but have principally different methods of modeling time. Additionally, we extend the concept of meaningful perturbation introduced by [1] to the temporal dimension, to identify the temporal part of a sequence most meaningful to the network for a classification decision. Our findings indicate that the 3D convolutional model concentrates on shorter events in the input sequence, and places its spatial focus on fewer, contiguous areas.

1 Introduction

Two standard approaches to deep learning for sequential image data are 3D convolutional neural networks (3D CNNs), e.g., the I3D model [2], and recurrent neural networks (RNNs). Among the RNNs, the convolutional long short-term memory network (hereon, C-LSTM) [3] is especially suited for sequences of images, since it learns both spatial and temporal dependencies simultaneously. Although both methods can capture aspects of the semantics pertaining to the temporal dependencies in a video clip, there is a fundamental difference in how 3D CNNs treat time compared to C-LSTMs. In 3D CNNs, the time axis is treated just like a third spatial axis, whereas C-LSTMs only allow for information flow in the direction of increasing time, complying with the second law of thermodynamics. More concretely, C-LSTMs maintain a hidden state that is continuously updated when forward-traversing the input video sequence, and are able to model non-linear transitions in time. 3D CNNs, on the other hand,

* Equal contribution

convolve (i.e., take weighted averages) over both the temporal and spatial dimensions of the sequence.

The question investigated in this paper is whether this difference has consequences for how the two models compute spatiotemporal features. We present a study of how 3D CNNs and C-LSTMs respectively compute video features: what do they learn, and how do they differ from one another?

As outlined in Section 2, there is a large body of work on evaluating video architectures on spatial and temporal correlations, but significantly fewer investigations of what parts of the data the networks have used and what semantics relating to the temporal dependencies they have extracted from them. Deep neural networks are known to be large computational models, whose inner workings are difficult to overview for a human. For video models, the number of parameters is typically significantly higher due to the added dimension, which makes their interpretability all the more pressing.

We will evaluate these two types of models (3D CNN and C-LSTM) on tasks where temporal order is crucial. The 20BN-Something-something-V2 dataset [4] (hereon, Something-something) will be central to our investigations; it contains time-critical classes, agnostic to object appearance, such as *move something from left to right* or *move something from right to left*. We additionally evaluate the models on the smaller KTH actions dataset [5].

Our contributions are listed as follows.

- We present the first comparison of 3D CNNs and C-LSTMs in terms of temporal modeling abilities. We point to essential differences between their assumptions concerning temporal dependencies in the data through qualitative and quantitative experiments.
- We extend the concept of meaningful perturbation introduced by [1] to the temporal dimension, to search for the most critical part of a sequence used by the networks for classification.

2 Related Work

The field of interpretability in deep learning is still young but has made considerable progress for single-image networks, owing to works such as [6–9]. One can distinguish between data centric and network centric methods for interpretability. *Activity maximization*, first coined by [10], is network centric in the sense that specific units of the network are studied. By maximizing the activation of a given unit by gradient ascent with reference to the input, one can compute its optimal input. In data centric methods, the focus is instead on the input to the network in order to reveal which patterns of the data the network has discerned.

Grad-CAM [11] and the meaningful perturbations explored in the work by [1] (Section 3), which form the basis for our experiments, belong to the data centric category. Layer-wise relevance propagation [9] (LRP) and Excitation backprop [12] are two other examples of data centric backpropagation techniques designed for interpretability, where Excitation backprop follows from a simpler parameter

setting of LRP. In Excitation backprop, saliency maps are produced without the use of gradients. Instead, products of forward weights and activations are normalized in order to be used as conditional probabilities, which are back-propagated. Building on Excitation backprop, [13] produce saliency maps for video RNNs. In our experiments, we produce spatial saliency maps using Grad-CAM, since it is efficient, easy to implement, widely used, and one of the saliency methods in [14] that passes the article's sanity checks.

Few works have been published with their focus on interpretability for video models [13, 15–18]. Other works have treated it, but with less extensive experimentation [19], while mainly presenting a new spatiotemporal architecture [20, 21]. We build on the work by [18], where the aim is to measure a network's ability to model *video time* directly, and not via the proxy task of action classification, which is most commonly seen. Three defining properties of video time are defined in the paper: temporal symmetry, temporal continuity and temporal causality, each accompanied by a measurable task. The third property is measured using the classification accuracy on Something-something. An important contribution of ours is that we compare 3D CNNs and C-LSTMs, whereas [18] compare 3D CNNs to standard LSTMs. Their comparison can be argued as slightly unfair, as standard LSTM layers only take 1D input, and thus need to vectorize each frame, which removes some spatial dependencies in the pixel grid. [20, 22, 23] all use variants of convolutional RNNs, but train them on CNN features. To the best of our knowledge, there has been no published convolutional RNNs trained on raw image data. This is crucial since information is lost when downsampling an image into CNN features, and we want to study networks having sufficient degrees of freedom to learn temporal patterns from scratch.

Similar to our work, [20] investigate the temporal modeling capabilities of convolutional gated recurrent units (ConvGRUs) trained on Something-something. The authors find that recurrent models perform well for the task, and present a qualitative analysis of the trained model's learned hidden states. For each class of the dataset, they obtain the hidden states of the network corresponding to the frames of one clip and display its nearest neighbors from other clips' per-frame hidden state representations. These hidden states had encoded information about the relevant frame ordering for the classes. [16] examine video architectures and datasets on a number of qualitative attributes. [17] investigate how much the motion contributes to the classification performance of a video architecture. To measure this, they vary the number of sub-sampled frames per clip and examine how much the accuracy changes as a result.

In a search-based precursor to our temporal mask experiments, [24] crop sequences temporally to obtain the most discriminative sub-sequence for a certain class. The crop corresponding to the highest classification confidence is selected as the most discriminative sub-sequence. This selection is done using an exhaustive search for crops across all frames, which increases in complexity with the sequence length according to $\frac{|f|^2}{2}$, where $|f|$ is the number of frames. Our proposed method, however, is gradient-descent based and has a fixed number of

iterations regardless of sequence length. Furthermore, our approach can identify more than one temporal sub-region in the sequence, in contrast to [24].

[15] present the first network centric interpretability work for video models. The authors investigate spatiotemporal features using activity maximization. [21] introduce the Temporal Relational Network (TRN), which learns temporal dependencies between frames through sampling the semantically relevant frames for a particular action class. The TRN module is put on top of a convolutional layer and consists of a fully connected network between the sampled frame features and the output. Similar to [20], they perform temporal alignment of clips from the same class, using the frames considered most representative for the clip by the network. They verify the conclusion previously made by [25], that temporal order is crucial on Something-something and also investigate for which classes it is most important.

3 Approach

3.1 Temporal Masks

The proposed temporal mask method aims to extend the interpretability of deep networks into the temporal dimension, utilizing meaningful perturbation of the input, as shown effective in the spatial dimension by [1]. When adopting this approach, it is necessary to define what constitutes a *meaningful* perturbation. In the mentioned paper, a mask that blurs the input as little as possible is learned for a single image, while still maximizing the decrease in class score. Our proposed method applies this concept of a learned mask to the temporal dimension. The perturbation, in this setting, is a noise mask approximating either a 'freeze' operation, which removes motion data through time, or a 'reverse' operation that inverses the sequential order of the frames. This way, we aim to identify which frames are most critical for the network's classification decision.

The temporal mask is defined as a vector of real numbers on the interval [0,1] with the same length as the input sequence. For the 'freeze' type mask, a value of 1 for a frame at index t duplicates the value from the previous frame at $t-1$ onto the input sequence at t. The pseudo-code for this procedure is given below.

```
for i in maskIndicesExceptFirst do
  originalComponent := (1-mask[i])*originalInput[i]
  perturbedComponent := mask[i]*perturbedInput[i-1]
  perturbedInput[i] := originalComponent + perturbedComponent
end for
```

For the 'reverse' mask type, all indices of the mask $\mathbf{m}$ that are activated are first identified (threshold 0.1). These indices are then looped through to find all contiguous sections, which are treated as sub-masks, m_i. For each sub-mask, the frames at the active indices in the sub-mask are reversed. For example (binary for clarity), an input indexed as $t_{1:16}$ perturbed with a mask with the value $[0,0,0,1,1,1,1,1,0,0,0,0,0,1,1,0]$ results in the sequence with frame indices $[1,2,3,8,7,6,5,4,9,10,11,12,13,15,14,16]$.

In order to learn the mask, we define a loss function (Eq. 1) to be minimized using gradient descent, similar to the approach in [1].

$$\mathcal{L} = \lambda_1 \|\mathbf{m}\|_1^1 + \lambda_2 \|\mathbf{m}\|_\beta^\beta + F_c, \tag{1}$$

where $\mathbf{m}$ is the mask expressed as a vector $m \in [0,1]^t$, $\|\cdot\|_1^1$ is the L^1 norm, $\|\cdot\|_\beta^\beta$ is the Total Variation (TV) norm, $\lambda_{1,2}$ are weighting factors, and F_c is the class score given by the model for the perturbed input. The L^1 norm punishes long masks, in order to identify only the most important frames in the sequence. The TV norm penalizes masks that are not contiguous. This approach allows our method to automatically learn masks that identify one or several contiguous sequences in the input. The mask is initialized centered in the middle of the sequence. To keep the perturbed input class score differentiable with respect to the mask, the optimizer operates on a real-valued mask vector. A sigmoid function is applied to the mask before using it for the perturbing operation in order to keep its values in the [0,1] range. The ADAM optimizer is then used to learn the mask through 300 iterations of gradient descent. After the mask has converged, its sigmoidal representation is thresholded for visualisation purposes.

3.2 Grad-CAM

Grad-CAM [11] is a method for producing visual explanations in the form of class-specific saliency maps for CNNs. One saliency map, L_t^c, is produced for each image input based on the activations from k filters, A_{ij}^k, at the final convolutional layer. In order to adapt the method to sequences of images, the activations for all timesteps t in the sequences are considered (Eq. 2).

$$L_{ijt}^c = \sum_k w_{kt}^c A_{ijt}^k \quad ; \qquad w_{kt}^c = \frac{1}{Z} \sum_{ij} \frac{\partial F^c}{\partial A_{ijt}^k}, \tag{2}$$

where Z is a normalizing constant and F^c is the network output for the class c. By up-sampling these saliency maps to the resolution of the original input image, the aim is to examine what spatial data in specific frames contributed most to the predicted class.

4 Experiments

4.1 Datasets

Something-something [4] contains over 220,000 sequences from 174 classes with a duration of more than 200 hours. The videos are recorded against varying backgrounds from different perspectives. The classes are action-oriented and object-agnostic. Each class is defined as performing some action with one or several arbitrary objects, such as *closing something* or *folding something*. This encourages the classifier to learn the action templates, since object recognition

does not give enough information for the classifying task. We train and validate according to the provided split, and use a frame resolution of 224x224.

The KTH Actions dataset [5] consists of 25 subjects performing six actions (*boxing, waving, clapping, walking, jogging, running*) in four different settings, resulting in 2391 sequences, and a duration of almost three hours (160x120 pixels at 25 fps). They are filmed against a homogeneous background with the different settings exhibiting varying lighting, distance to the subject and clothing of the participants. For this dataset, we train on subjects 1-16 and evaluate on 17-25.

Both datasets have sequences varying from one to almost ten seconds. As 3D CNNs require input of fixed sequence length, all input sequences from both datasets are sub-sampled to cover the entire sequence in 16 frames (Something-something) and 32 frames (KTH Actions). The same set of sub-sampled frames is then used as input for both architectures.

4.2 Architecture Details

Both models were trained from scratch on each dataset, to ensure that the learned models were specific to the relevant task. Pre-training on Kinetics can increase performance, but for our experiments, the models should be trained on the temporal tasks presented by the Something-something dataset specifically. It can be noted that our I3D model reached comparable performance to another I3D trained from scratch on Something-something presented in the work of [25]. Hyperparameters are listed on the project webpage. Any remaining settings can be found in the public code repository.

I3D consists of three 3D convolutional layers, nine Inception modules and four max pooling layers (Fig. 1). In the original setting, the temporal dimension of the input is down-sampled to $L/8$ frames by the final Inception module, where L is the original sequence length. In order to achieve a higher temporal resolution in the produced Grad-CAM images, the strides of the first convolutional layer and the second max pooling layer are reduced to 1x2x2 in our code, producing $L/2$ activations in the temporal dimension. The Grad-CAM images are produced from the gradients of the class scores with respect to the final Inception module.

We have not found any published C-LSTMs trained on raw pixels, and thus conducted our own hyperparameter search for this model. The model was selected solely based on classification performance; all feature investigations were conducted after this selection. The C-LSTM used for Something-something consists of three C-LSTM layers (two for KTH) with 32 filters, each followed by batch normalization and max pooling layers. The convolutional kernels used for each layer had size 5x5 and stride 2x2. The C-LSTM layers return the entire transformed sequence as input to the next layer. When calculating the Grad-CAM maps for the C-LSTM, the final C-LSTM layer was used.

There is a substantial difference in the number of parameters for each model, with $12,465,614$ for I3D and $1,324,014$ for the three-layer C-LSTM. Other variants of the C-LSTM with a larger number of parameters (up to five layers) were evaluated as well, but no significant increase in performance was observed. Also, due to the computational complexity of back-propagation through time (BPTT),

the C-LSTM variants were significantly more time demanding to train than their I3D counterparts.

4.3 Comparison Method

To study the differences in the learned spatiotemporal features of the two models, we first compute spatial saliency maps using Grad-CAM and temporal masks using the proposed method. Once these are obtained for each model and dataset, we both examine them qualitatively and compute the quantitative metrics listed below. A 'blob' is defined as a contiguous patch within the Grad-CAM saliency map for one frame. The blobs were computed using the blob detection tool from OpenCV [26]. OS, FS and RS are the softmax scores for one class resulting from the original input, and from the freeze and reverse perturbed input, respectively.

- **Blob count:** The average number of blobs (salient spatial areas, as produced by the Grad-CAM method), per frame.
- **Blob size:** The average size of one salient spatial area (blob), in pixels, computed across all detected blobs.
- **Center distance:** The average distance in pixels to the center of the frame for one blob, computed across all detected blobs.
- **Mask length:** The average number of salient frames per sequence, as produced by the temporal mask method.
- **Drop ratio:** The average ratio between the drop in classification score using the freeze and reverse perturbations, defined as $\frac{OS-FS}{OS-RS}$, across all sequences.
- **Drop difference:** The average difference between the drop in classification score using the freeze and reverse perturbations, defined as $(OS - FS) - (OS - RS)$ (and equivalent to $RS - FS$), across all sequences.

We consider the difference and ratio between the freeze and reverse drops as the most relevant measures of how sensitive one model was for the reverse perturbation. FS and RS should not be compared in absolute numbers, since they depend on OS which might have been different for the two models. Moreover, using the same number of iterations for the optimization of the temporal mask, the two models typically reached different final losses (generally lower for I3D).

5 Results

For reference, the global validation F1-scores for both architectures and datasets are shown in Table 1. To emphasize the importance of temporal direction between the datasets, we first conduct a test where all the input validation sequences are entirely reversed. On Something-something, both C-LSTM and I3D were affected drastically, while on KTH, both performed well. Likely, this is because KTH's classes have distinct spatial features. As expected, Something-something is more time-critical than KTH. Overall, this shows that both models are indeed globally sensitive to temporal direction, when they need to be. In Sections 5.1-5.2, we examine in detail which spatiotemporal features are learned by the two models, and how they differ from one another.

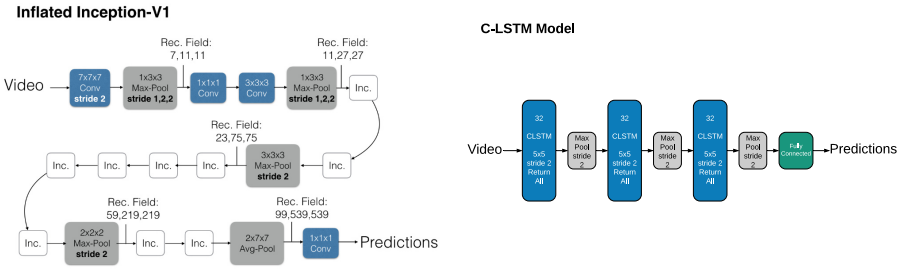


Fig. 1: I3D network (figure from [2]) and C-LSTM network (*right*).

Table 1: Validation F1-score per model on the two datasets. 'Rev.' indicates that the validation sequences were reversed at test time.

Model	**KTH Actions** (Top-1)	**Smth-Smth** (Top-1)	**Smth-Smth** (Top-5)
C-LSTM	0.84	0.23	0.48
C-LSTM (rev.)	0.78	0.05	0.17
I3D	0.86	0.43	0.73
I3D (rev.)	0.80	0.09	0.27

5.1 Interpretability Results on Something-something

The less widely used C-LSTM architecture could not reach the same global performance as the state-of-the-art I3D (Table 1), which also has an order of magnitude more parameters. The models were only compared on sequences from classes for which they had similar performance (Table 2). We include a variety of per-class F1-scores, ranging from approximately 0.1 to 0.9. All are, however, well above the random chance performance of $1/174 \approx 0.006$. The reason to include a variety of performance levels when studying the extracted features is to control for the general competence of the model. A well performing model might extract different features than a poor one.

In this section, we present an analysis of the Grad-CAM saliency maps and temporal masks generated for each architecture on the eleven classes. We evaluated the models on all validation sequences from these classes (1575 sequences in total). Quantitative results from the feature analysis are shown in Tables 3-4 and in Fig. 4. We display eight sample sequences in Figs. 2-3, but more on the project webpage.

Trends Regarding the Spatial Focus of the Two Models. We observe that the I3D generally focuses on contiguous, centered blobs, while the C-LSTM attempts to find relevant spatial features in multiple smaller areas (Table 3). Figs. 2a and 2c show examples of this, where I3D focuses on a single region covering both objects, while the C-LSTM has separate activations for the two objects, hands and the surface affected by the movement.

Table 2: F1-score per class and model on the Something-something dataset.

Class	**I3D**	**C-LSTM**
burying something in something	0.1	0.06
moving something and something away from each other	0.76	0.58
moving something and something closer to each other	0.77	0.57
moving something and something so they collide with each other	0.16	0.03
moving something and something so they pass each other	0.37	0.31
moving something up	0.43	0.40
pretending to take something from somewhere	0.10	0.07
turning the camera downwards while filming something	0.67	0.56
turning the camera left while filming something	0.94	0.79
turning the camera right while filming something	0.91	0.8
turning the camera upwards while filming something	0.81	0.73

Table 3: Statistics for the Grad-CAM maps for each model on eleven classes from the validation set of Something-something (1575 sequences, 16 frames per sequence) and the whole test set of the KTH dataset (863 sequences, 32 frames per sequence). The 'blobs', i.e., the contiguous patches within each Grad-CAM map, were computed per frame, using the blob detection tool from OpenCV [26].

Model (Dataset)	**Blob count**	**Blob size**	**Center distance**
I3D (Smth-smth)	1.6 ± 0.97	33.7 ± 19.6	54.4 ± 33.6
C-LSTM (Smth-smth)	3.6 ± 1.9	26.7 ± 24.5	96.8 ± 34.9
I3D (KTH)	1.1 ± 0.5	44.0 ± 18.7	44.6 ± 19.4
C-LSTM (KTH)	32.6 ± 15.1	5.8 ± 7.0	49.9 ± 22.4

We further find that the I3D has a bias of starting its focus around the middle of the frame (Figs. 2-3), often even before the motion starts. This trend persists throughout the sequence, as the average distance to the center of the image for each blob in each frame is shorter for I3D (Table 3). The typical behavior for the C-LSTM is instead to remain agnostic until the action actually starts (e.g., Fig. 3a). In Fig. 3a, the I3D maintains its foveal focus even after the green, round object is out of frame. In Fig. 3b, the focus splits midway to cover both the moped and some features on the wall, while the C-LSTM focuses mainly on numerous features along the wall, as it usually does in classes where the camera turns. The C-LSTM also seems to pay more attention to hands appearing in the clips, rather than objects (Figs. 2a and 2c-e). Fig. 4 shows the normalized histograms of these spatial features. The distributions for the two models differ significantly for all three measures.

Trends of the Temporal Masks of the Two Models. The quantitative results from the temporal mask experiments are shown in Table 4[⋆]. We first

[⋆] For the drop ratio, if the denominator OS-RS ≤ 0.001, the sample was filtered out since its ratio would explode. The OS-FS ≤ 0.001 were also excluded for bal-

Table 4: Statistics for the temporal masks of both models for both datasets (1575 sequences for Something-something and 863 sequences for KTH).

Model (Dataset)	**Mask length**	**Drop ratio**	**Drop diff.**
I3D (Smth-smth)	6.2 ± 3.3	8.4 ± 47	0.2 ± 0.3
C-LSTM (Smth-smth)	9.9 ± 4.1	2.6 ± 6.9	0.08 ± 0.2
I3D (KTH)	10.6 ± 8.5	81.4 ± 174	0.57 ± 0.34
C-LSTM (KTH)	15.2 ± 5.7	17.4 ± 45.2	0.22 ± 0.18

note that the average temporal mask is shorter for the I3D. This suggests that it has learned to react to short, specific events in the sequences. As an example, its temporal mask in Fig. 2c is active only on the frames where the objects first pass each other, and in Fig. 2b, it is active on the frames leading to the objects touching (further discussed in Section 5.3). Second, we note that the drop ratio and drop difference are generally higher for the I3D compared to C-LSTM (Table 4), suggesting that I3D is less sensitive to the reverse perturbation.

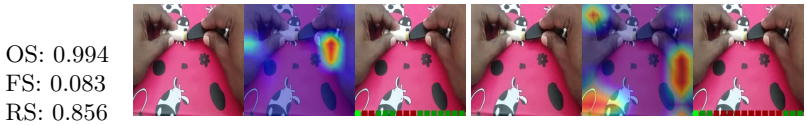
The normalized histograms of the three measures are shown in Fig. 4. The mask length distributions clearly have different means. For drop ratio and drop difference, the distributions have more overlap. A t-test conducted in Scipy [27] of the difference in mean between the two models assuming unequal variance gives a p-value $< 10^{-6}$ for both measures. We conclude that there is a significant difference in mean between the two models for drop ratio and drop difference.

Class Ambiguity of the Something-something Dataset. The Something-something classes can be ambiguous (one class may contain another class) and, arguably, for some samples, incorrectly labeled. Examining the spatiotemporal features may give insight as to how the models handle these ambiguities. Fig. 2e shows a case of understandable confusion, where I3D answers *taking one of many similar things on the table*. The surface seen in the image is a tiled floor, and the object is a transparent ruler. Once the temporal mask activates during the lifting motion in the last four frames, the Grad-CAM images show the model also focusing on two lines on the floor. These could be considered similar to the outline of the ruler, which could explain the incorrect classification. An example of ambiguous labeling can be seen for example in Fig. 2b, where I3D's classification is *moving something and something so they collide with each other* and the C-LSTM predicts *pushing something with something*. Although the two objects in the sequence do move closer to each other, they also touch at the end, making both predictions technically correct.

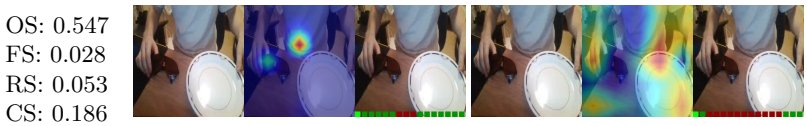
5.2 Interpretability Results on the KTH Actions Dataset

For the KTH dataset, we make similar observations regarding temporal and spatial features. In Fig. 5a, we observe results for the class 'handclapping'. In-

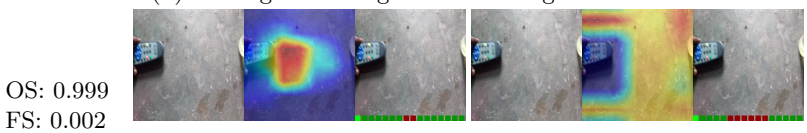
ance. When using 10^{-9} as threshold instead, the drop ratio results for Something-something were 215 ± 6346 (I3D) and 4.9 ± 47.6 (C-LSTM).



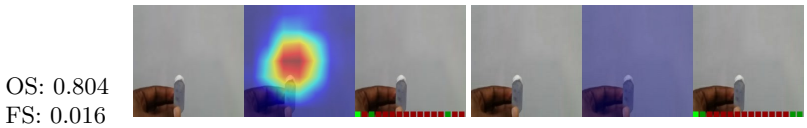
(a) Moving something and something away from each other.



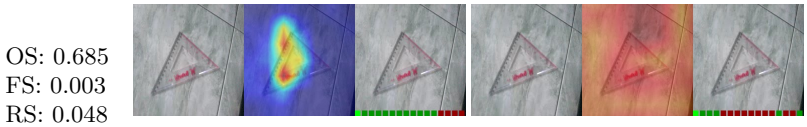
(b) Moving something and something closer to each other.



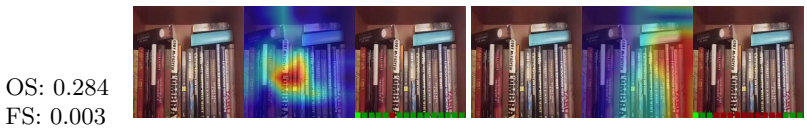
(c) Moving something and something so they pass each other.



(d) Moving something up.

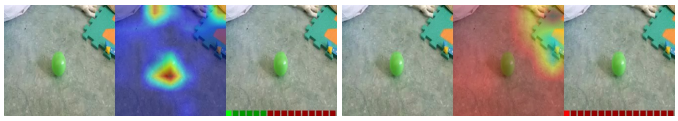


(e) Moving something up.



(f) Pretending to take something from somewhere.

Fig. 2: **Best displayed in Adobe Reader where the figures can be played as videos, or on the project webpage.** I3D (*left*) and C-LSTM (*right*) results for validation sequences from Something-something. The three columns show, from left to right, the original input, the Grad-CAM result, and the input as perturbed by the temporal freeze mask. The third column also visualizes when the mask is on (*red*) or off (*green*), with the current frame highlighted. *OS*: original score (softmax output) for the guessed class, *FS*: freeze score, *RS*: reverse score, *CS*: score for the ground truth class when there was a misclassification and *P*: predicted label, if different from ground truth.



(a) Turning the camera downwards while filming something.

(b) Turning the camera upwards while filming something.

Fig. 3: **Best displayed in Adobe Reader where the figures can be played as videos.** Same structure as Fig. 2.

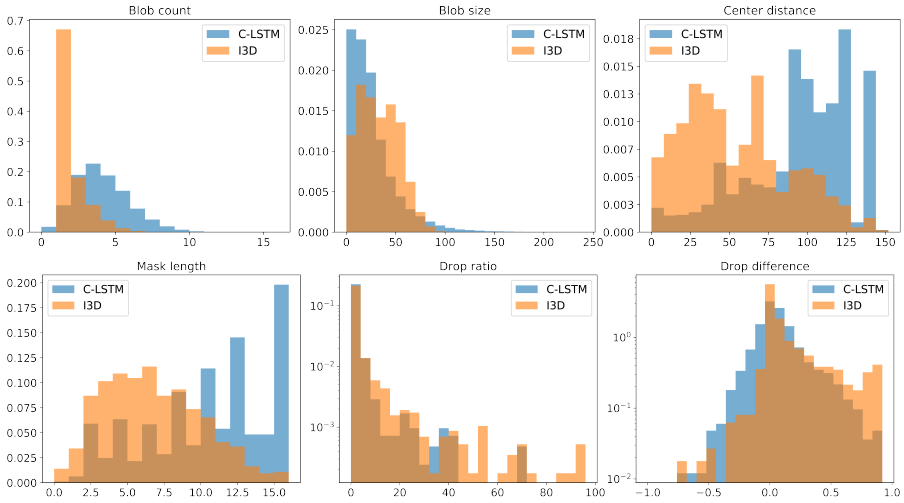
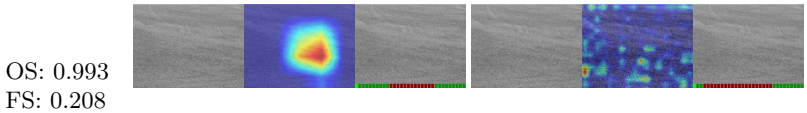


Fig. 4: Normalized histogram results for the Grad-CAM and temporal mask analysis for the I3D (*orange*) and C-LSTM (*blue*) networks. The histograms correspond to the results in Tables 3-4.

terestingly, the mask of each model covers at least one entire cycle of the action. The reverse perturbation affects both models very little since one action cycle is symmetrical in time. For the 'running' class (Fig. 5b), we see that the temporal mask identifies the frames in which the subject is in-frame as the most salient for both models, with I3D placing more focus on the subject's legs.

(a) Handclapping, subject 18.



(b) Running, subject 25.

Fig. 5: **The figures can be displayed as videos in Adobe Reader.** Same structure as Fig. 2.

5.3 Discussion

As stated in Section 1, 3D CNNs and C-LSTMs have fundamentally different ways of modeling time. In the following, we discuss two related observations: the shorter temporal masks of I3D and the fact that the classification scores after the freeze and reverse perturbations often are lower for I3D than for the C-LSTM.

For the I3D, all dimensions including the temporal axis of the input are progressively compressed through either convolutional strides or max pooling. The general understanding of CNNs are that later layers encode higher level features. In the deep video network examined in the work by [15], it is shown that the later layer units activate maximally for higher level actions. The representation that is input to the prediction layer in a 3D CNN has compressed high level motions or spatial relations through time to a shorter representation. The classification is then dependent on the presence or absence of these high level features in this representation. If perturbing the input would alter these critical high level features, the resulting prediction might be drastically affected.

For the C-LSTM, however, hidden states resulting from the entire input sequence are sent to the prediction layer. Ultimately, this means that it has a more temporally fine-grained feature space than its 3D CNN counterpart. We hypothesize that this is related to the two observed results. Due to this fine-grained and enveloping temporal feature space, the perturbation must remove

larger sub-sequences from the data to obscure enough information through time to cause a large change in prediction score, possibly accounting for the longer temporal masks observed for C-LSTM. Furthermore, as we penalize the length of the mask during optimization, the resulting converged mask is often too short to fully bring down the classification score of the C-LSTM method. Examples of where the freeze score is brought close to, or below, 0.1 are when the mask is nearly or fully active, as seen in Figs. 2b, 2d and 3a.

6 Conclusions and Future Work

We have presented the first comparison of the spatiotemporal information used by 3D CNN and C-LSTM based models in action recognition. We have presented indications that the difference in temporal modeling has consequences for what features the two models learn. Using the proposed temporal mask method, we presented empirical evidence that I3D on average focuses on shorter and more specific sequences than the C-LSTM. On average, our experiments showed that I3D also tends to focus on fewer or a single contiguous spatial patch closer to the center of the image, instead of smaller areas on several objects like the C-LSTM. Also, when comparing the effect of reversing the most salient frames or removing motion through 'freezing' them, the C-LSTM experiences a relatively larger decrease in prediction confidence than I3D upon reversal. We have also seen that the temporal mask is capable of identifying salient frames in sequences, such as one cycle of a repeated motion.

There is still much to explore in the patterns lying in temporal dependencies. It would be of interest to extend the study to other datasets where temporal information is important, e.g., Charades [28]. Other possible future work includes evaluating the effect of other noise types beyond 'freeze' and 'reverse'. We hope that this empirical study can guide future development and understanding of deep video models.

It is desirable that a model can be trained with as little data as possible. 3D CNNs do not represent video (time) in a physically sound way, treating it as a third spatial dimension. In our view, this is often made up for using large amounts of data and brute-force learning of its correlations, as most state-of-the-art video CNNs are from industry, trained on hundreds of GPUs, e.g., SlowFast [29]. For efficiency, it is important that the representation learned by the model should correspond to the world, and that variables that are uncorrelated in the world remain so in the model. With our evaluation framework it will be possible to gain further insight into what state-of-the-art video models have actually learned.

References

1. Fong, R.C., Vedaldi, A.: Interpretable explanations of black boxes by meaningful perturbation. In: The IEEE International Conference on Computer Vision (ICCV). (2017)

2. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2017)
3. Shi, X., Chen, Z., Wang, H.: Convolutional LSTM Network : A Machine Learning Approach for Precipitation Nowcasting arXiv : 1506 . 04214v1 [cs . CV] 13 Jun 2015. (2015) 1–11
4. Mahdisoltani, F., Berger, G., Gharbieh, W., Fleet, D.J., Memisevic, R.: Fine-grained video classification and captioning. CoRR **abs/1804.09235** (2018)
5. Schuldt, C., Laptev, I., Caputo, B.: Recognizing human actions: a local svm approach. In: Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004. Volume 3., IEEE (2004) 32–36
6. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **8689 LNCS** (2013) 818–833
7. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. (2014) 1–8
8. Kim, B., M., W., Gilmer, J., C., C., J., W., , Viegas, F., Sayres, R.: Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV) . ICML (2018)
9. Montavon, G., Samek, W., Müller, K.R.: Methods for interpreting and understanding deep neural networks. Digital Signal Processing: A Review Journal **73** (2018) 1–15
10. Erhan, D., Bengio, Y., Courville, A., Vincent, P.: Visualizing higher-layer features of a deep network. Bernoulli (2009) 1–13
11. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. Proceedings of the IEEE International Conference on Computer Vision **2017-Octob** (2017) 618–626
12. Zhang, J., Lin, Z., Brandt, J., Shen, X., Sclaroff, S.: Top-down neural attention by excitation backprop. CoRR **abs/1608.00507** (2016)
13. Adel Bargal, S., Zunino, A., Kim, D., Zhang, J., Murino, V., Sclaroff, S.: Excitation backprop for RNNs. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2018)
14. Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I.J., Hardt, M., Kim, B.: Sanity checks for saliency maps. CoRR **abs/1810.03292** (2018)
15. Feichtenhofer, C., Pinz, A., Wildes, R.P., Zisserman, A.: What have we learned from deep representations for action recognition? (2018) 1–64
16. Sigurdsson, G.A., Russakovsky, O., Gupta, A.: What Actions are Needed for Understanding Human Actions in Videos? (2017)
17. Huang, D.A., Ramanathan, V., Mahajan, D., Torresani, L., Paluri, M., Fei-Fei, L., Carlos Niebles, J.: What makes a video a video: Analyzing temporal information in video understanding models and datasets. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2018)
18. Ghodrati, A., Gavves, E., Snoek, C.G.M.: Video time: Properties, encoders and evaluation. In: British Machine Vision Conference. (2018)
19. Chattopadhyay, A., Sarkar, A., Howlader, P., Balasubramanian, V.N.: Grad-CAM++: Generalized Gradient-based Visual Explanations for Deep Convolutional Networks. (2017)
20. Dwibedi, D., Sermanet, P., Tompson, J.: Temporal reasoning in videos using convolutional gated recurrent units. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops. (2018)

21. Zhou, B., Andonian, A., Oliva, A., Torralba, A.: Temporal relational reasoning in videos. European Conference on Computer Vision (2018)
22. Ballas, N., Yao, L., Pal, C., Courville, A.C.: Delving deeper into convolutional networks for learning video representations. In: The International Conference on Learning Representations. (2016)
23. Li, Z., Gavrilyuk, K., Gavves, E., Jain, M., Snoek, C.G.: VideoLSTM convolves, attends and flows for action recognition. Computer Vision and Image Understanding **166** (2018) 41–50
24. Satkin, S., Hebert, M.: Modeling the temporal extent of actions. In: European Conference on Computer Vision. (2010)
25. Xie, S., Sun, C., Huang, J., Tu, Z., Murphy, K.: Rethinking spatiotemporal feature learning for video understanding. CoRR **abs/1712.04851** (2017)
26. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000)
27. Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Jarrod Millman, K., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C., Polat, İ., Feng, Y., Moore, E.W., Vand erPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., Contributors, S...: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods **17** (2020) 261–272
28. Sigurdsson, G.A., Varol, G., Wang, X., Farhadi, A., Laptev, I., Gupta, A.: Hollywood in homes: Crowdsourcing data collection for activity understanding. In: European Conference on Computer Vision. (2016)
29. Feichtenhofer, C., Fan, H., Malik, J., He, K.: SlowFast networks for video recognition. In: The IEEE International Conference on Computer Vision (ICCV). (2019)