International Journal of Computer Vision (2020) 128:420–437
https://doi.org/10.1007/s11263-019-01225-w

Deep Insights into Convolutional Networks for Video Recognition

Christoph Feichtenhofer[1] · **Axel Pinz**[1] · **Richard P. Wildes**[2] · **Andrew Zisserman**[3]

Received: 26 September 2018 / Accepted: 29 August 2019 / Published online: 29 October 2019

Abstract

As the success of deep models has led to their deployment in all areas of computer vision, it is increasingly important to understand how these representations work and what they are capturing. In this paper, we shed light on deep spatiotemporal representations by visualizing the internal representation of models that have been trained to recognize actions in video. We visualize multiple two-stream architectures to show that local detectors for appearance and motion objects arise to form distributed representations for recognizing human actions. Key observations include the following. First, cross-stream fusion enables the learning of true spatiotemporal features rather than simply separate appearance and motion features. Second, the networks can learn local representations that are highly class specific, but also generic representations that can serve a range of classes. Third, throughout the hierarchy of the network, features become more abstract and show increasing invariance to aspects of the data that are unimportant to desired distinctions (e.g. motion patterns across various speeds). Fourth, visualizations can be used not only to shed light on learned representations, but also to reveal idiosyncrasies of training data and to explain failure cases of the system.

Keywords Computer vision · Machine learning · Deep learning · Video recognition · Neural network visualization · Action recognition

1 Motivation

Principled understanding of how deep networks operate and achieve their strong performance significantly lags behind their realizations. Since these models are being deployed to all fields from medicine to transportation, this issue becomes of ever greater importance. Previous work has yielded great advances in effective architectures for recognizing actions in video, with especially significant strides towards higher accuracies made by deep spatiotemporal networks (Tran et al. 2015; Feichtenhofer et al. 2016b; Simonyan and Zisserman 2014; Wang et al. 2016; Carreira and Zisserman 2017). However, what these models actually learn remains unclear, since their compositional structure makes it difficult to reason explicitly about their learned representations. In this paper we use spatiotemporally regularized activation maximization (Simonyan et al. 2014; Mordvintsev et al. 2015; Yosinski et al. 2015; Mahendran and Vedaldi 2016b) to visualize deep two-stream representations (Simonyan and Zisserman 2014) and better understand what the underlying models have learned.

As an example, in Fig. 1 we highlight a single interesting unit at the last convolutional layer of the VGG-16 Two-Stream Fusion model (Feichtenhofer et al. 2016b), which fuses appearance and motion features. This unit shows the strongest activation at this layer across all videos corresponding to the Billiards action in the test set of UCF101 (Soomro et al. 2012). We visualize the appearance and motion inputs

Communicated by Greg Mori (retired editor).

This document is best viewed offline[1] where figures play as animation.

✉ Christoph Feichtenhofer
cfeichtenhofer@gmail.com

Axel Pinz
axel.pinz@tugraz.at

Richard P. Wildes
wildes@cse.yorku.ca

Andrew Zisserman
az@robots.ox.ac.uk

1 Graz University of Technology, Graz, Austria

2 York University, Toronto, Canada

3 University of Oxford, Oxford, UK

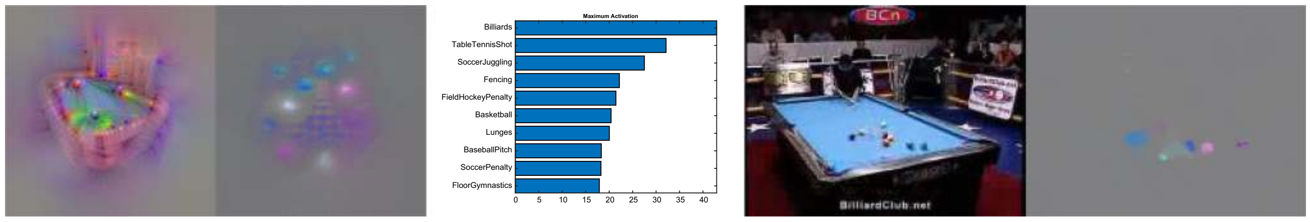

Fig. 1 Strongest Billiards unit at layer conv5_fusion: **a**, **b** show what maximizes the unit at the input: multiple coloured blobs in the appearance input (**a**) and moving circular objects at the motion input (**b**). **d** A sample clip from the test set, and **e** the corresponding optical flow (where the RGB channels correspond to the horizontal, vertical and magnitude flow components respectively). Note that **a**, **b** are optimized from white noise under regularized spatiotemporal variation. Best viewed in the Animated Manuscript with Adobe Reader where **b**, **d**, **e** should play as videos

that highly activate this filter in Fig. 1a, b, respectively.[1] We display the optical-flow visualizations of this paper as video clips with the RGB channels corresponding to the horizontal, vertical and magnitude flow components, respectively. This is similar to separate grayscale images showing the components of the flow, displayed in a single image. In the visualizations shown, red colour indicates positive horizontal direction (to the right), green color positive vertical direction (to the bottom) and blue color the magnitude of both flow components.

When looking at the inputs, we observe that this filter is activated by differently coloured blobs in the appearance input and by linear motion of circular regions in the motion input. Thus, this unit could support recognition of the Billiards class in UCF101, as well as other ball sports as can be seen in the maximum test set activity of this unit in Fig. 1c. Finally, we show in Fig. 1d a sample Billiards clip from the test set with the corresponding optical flow shown in Fig. 1e. Similar to emergence of object detectors for static images (Zhou et al. 2014; Bau et al. 2017), here we see the emergence of a spatiotemporal representation for an action. While Zhou et al. (2014) and Bau et al. (2017) automatically assigned concept labels to learned internal representations by reference to a large collection of labelled input samples, our work instead is concerned with visualizing the network's internal representations without appeal to any signal at the input and thereby avoids biasing the visualization via appeal to a particular set of samples.

Generally, we can understand deep networks from various viewpoints. An *architectural viewpoint* considers a network as a computational structure (e.g. a directed acyclic graph) of mathematical operations in feature space (e.g. affine scaling and shifting, local convolution and pooling, nonlinear activation functions, etc.). In previous work, architectures [such as Inception (Szegedy et al. 2015), VGG16 (Simonyan and Zisserman 2015), ResNet (He et al. 2016)] have been designed by composing such computational structures with a principle in mind (e.g. a direct path for backpropagation in ResNet). We can thus reason about their expected predictions for given input and the quantitative performance for a given task justifies their design, but this does not explain how a network actually arrives at these results. Another way to understand deep networks is the *representational viewpoint* that is concerned with the learned representation embodied in the network parameters. Understanding these representations is inherently hard as recent networks consist of a large number of parameters with a vast space of possible functions they can model. The hierarchical nature in which these parameters are arranged makes the task of understanding complicated, especially for ever deeper representations. Due to their compositional structure it is difficult to explicitly reason about what these powerful models actually have learned.

In this paper we shed light on deep spatiotemporal networks by visualizing what excites the learned models using activation maximization by backpropagating on the input. We concentrate our studies on Two-Stream networks (Simonyan and Zisserman 2014; Feichtenhofer et al. 2016a, b; Wang et al. 2016; Carreira and Zisserman 2017), aiming to reveal what information is conveyed by the learned representation of the motion stream, in order to find what makes it complementary to networks operating only on RGB information.

Our visual explanations are highly intuitive and provide qualitative support for the benefits of separating into two pathways of appearance (RGB image) and motion (optical-flow) information when processing spatiotemporal information—a principle that has also been found in nature where numerous studies suggest a corresponding separation into ventral and dorsal pathways of the brain (Mishkin et al. 1983; Goodale and Milner 1992; Felleman and van Essen 1991) as well as the existence of cross-pathway connections (Saleem et al. 2000; Kourtzi and Kanwisher 2000).

The present paper is an extended version of our related conference publication (Feichtenhofer et al. 2018), with additional contributions as follows. First, we augment our earlier activation maximization analysis of what the networks learn with response histograms across test data associated with

[1] An animated version of this document can be downloaded at http://feichtenhofer.github.io/pubs/Feichtenhofer_IJCV19.pdf where figures play as videos.

training labels to lend additional insight into what has been learned by units within the networks. Second, we show experiments on the evolution of filters throughout training. Third, we investigate several additional architectures, Spatiotemporal Residual Networks (Feichtenhofer et al. 2016a) using ResNet50 (He et al. 2016) streams, Temporal Segment Networks (Wang et al. 2016) using BN-Inception (Ioffe and Szegedy 2015) streams, and Two-Stream networks (Simonyan and Zisserman 2014) using Inception_v3 (Szegedy et al. 2015) streams, as well as the originally studied VGG16 two-stream fusion model (Feichtenhofer et al. 2016b). Fourth, we investigate the impact of training on additional datasets, UCF101 (Soomro et al. 2012), HMDB51(Kuehne et al. 2011) and Kinetics (Carreira and Zisserman 2017) to discuss our analysis in broader context, compared to the single architecture and dataset used in the earlier conference publication (Feichtenhofer et al. 2018).

2 Related Work on Visualization

The current approaches to visualization can be grouped into three types, and we review each of them in turn.

2.1 Visualizations for Given Inputs

Several approaches have used a large body of input images to increase the understanding of deep networks. A straightforward approach is to record the network activities and sample over a large set of input images for finding the ones that maximize the unit of interest (Zeiler and Fergus 2013; Zhou et al. 2014, 2016; Bau et al. 2017). Another strategy is to use backpropagation to highlight salient structure in the hidden units (Simonyan et al. 2014; Mahendran and Vedaldi 2016a; Zhang et al. 2016; Selvaraju et al. 2016) for given input. The drawback of these methods is that they use an image as prior for reasoning about network computations and thus can not provide direct samples from the model, but rather samples from the model under some specific input signal.

2.2 Activation Maximization

The Activation Maximization (AM) technique allows to inspect models without using an input. AM performs gradient ascent on the randomly initialized input to find an image that increases the activity of some unit of interest by backpropagation (Erhan et al. 2009). The method was employed to visualize units of Deep Belief Networks (Hinton et al. 2006; Erhan et al. 2009) and adopted for deep auto-encoder visualizations in Le et al. (2012). The AM idea was first applied to visualizing ConvNet representations trained on ImageNet (Simonyan et al. 2014). That work also showed that the AM techniques generalize the deconvolutional network reconstruction procedure introduced earlier (Zeiler and Fergus 2013), which can be viewed as a special case of one iteration in the gradient based activation maximization.

In an unconstrained setting, these methods can exploit the full dimensionality of the input space; therefore, plain gradient based optimization on the input can generate images that do not reflect natural signals. Regularization techniques can be used to compensate for this deficit. In the literature, different regularizers have been applied to the inputs to make them perceptually more interpretable: $L2$ norms (Simonyan et al. 2014), total-variation norms (Mahendran and Vedaldi 2016b), Gaussian blurring, and suppressing of low values and gradients (Yosinski et al. 2015), spatial shifting (jittering) of the input during optimization (Mordvintsev et al. 2015), or weight decay (Galloway et al. 2018).

Backpropagation on the input has also been used to find salient regions for a given input (Springenberg et al. 2015; Zhang et al. 2016; Mahendran and Vedaldi 2016a), or to “fool” networks by applying a perturbation to the input that is hardly perceptible to humans (Szegedy et al. 2014; Nguyen et al. 2015).

2.3 Generative Adversarial Networks

Generative Adversarial Networks (GANs) (Goodfellow et al. 2014) provide even stronger natural image priors, for visualizing class level representations (Nguyen et al. 2016, 2017) in the activation maximization framework. These methods optimize a high-dimensional code vector (typically fc_6 in AlexNet) that serves as an input to the generator which is trained with a perceptual loss (Dosovitskiy and Brox 2016) that compares the generater features to those from a pre-trained comparator network (typically AlexNet trained on ImageNet). The approach induces strong regularization on the possible signals produced. In other words, GAN-based activation maximization does not start the optimization process from scratch, but from a generator model that has been trained for the same or a similar task (Dosovitskiy and Brox 2016). More specifically, the work in (Nguyen et al. 2016) trains the generator network on ImageNet and activation maximization in some target (ImageNet) network is achieved by optimizing a high-level feature (i.e. fc_6) of this generator network.

Activation maximization results produced by GANs offer visually impressive results, because the GAN enforces natural looking images and these methods do not have to use extra regularization terms to suppress extremely high input signals, high frequency patterns or translated copies of similar patterns that highly activate some unit. However, the produced result of this maximization technique is in direct correspondence to the generator, the data used to train this model, and not a random sample from the network under inspection (which serves as a condition for the learned generative prior).

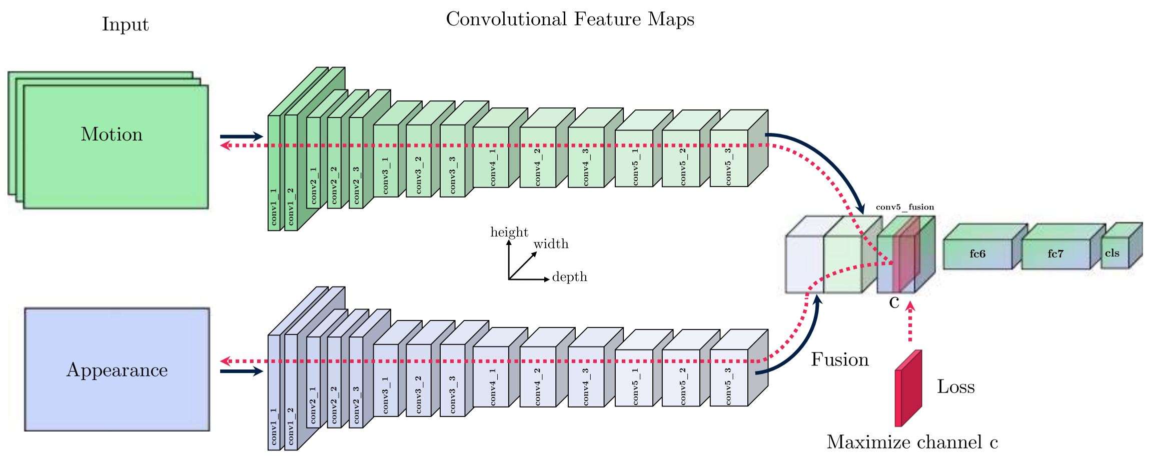

Fig. 2 Schematic of our two-stream activation maximization approach (see Sect. 3 for details)

Since we are interested in the raw input that excites our representations, we do not employ any generative priors in this paper. In contrast, our approach directly optimizes the spatiotemporal input of the models starting from randomly initialized image (appearance) and video (motion) inputs.

3 Approach

There are several techniques that perform activation maximization for image classification ConvNets (Szegedy et al. 2014; Simonyan et al. 2014; Mordvintsev et al. 2015; Yosinski et al. 2015; Mahendran and Vedaldi 2016b) which have shown that features become more abstract when approaching deeper layers of image-based networks. We build on these methods for visualizing the hierarchical features learned by a deep motion network. In particular, we optimize in the spacetime domain to find the preferred spatiotemporal input of individual units in a Two-Stream Fusion model (Feichtenhofer et al. 2016b). We formulate the problem as a (regularized) gradient-based optimization problem that searches in the input space.

An overview of our approach is shown in Fig. 2. A randomly initialized input is presented to the optical flow and the appearance pathways of our model. We compute the feature maps up to a particular layer that we would like to visualize. A single target feature channel, c, is selected and activation maximization is performed to generate the preferred input in two steps. First, the derivatives on the input that affect c are calculated by backpropagating the target loss, summed over all locations, to the input layer. Second, the propagated gradient is scaled by the learning rate and added to the current input. These operations are illustrated by the dotted red lines in Fig. 2. Gradient-based optimization performs these steps iteratively with an adaptively decreasing learning rate until the input converges. Importantly, during this optimization process the network weights are not altered, only the input receives changes. The detailed procedure is outlined in the remainder of this section.

3.1 Activation Maximization

To make the above more concrete, activation maximization of unit c at layer l seeks an input $\mathbf{x}^* \in \mathbb{R}^{H\times W\times T\times C}$, with H being the height, W the width, T the duration, and C the color and optical flow channels of the input. We find $\mathbf{x}^*$ by optimizing the following objective

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmax}} \frac{1}{\rho_l^2 \hat{\mathbf{a}}_{l,c}} \langle \mathbf{a}_l(\mathbf{x}), e_c \rangle - \lambda_r \mathcal{R}_r(\mathbf{x}) \tag{1}$$

where $\mathbf{a}_l$ are the activations at layer l, e_c is the natural basis vector corresponding to the cth feature channel, and $\mathcal{R}_r$ are regularization term(s) with weight(s) λ_r. To produce plausible inputs, the unit-specific normalization constant depends on ρ_l, which is the size of the receptive field at layer l (i.e. the input space), and $\hat{\mathbf{a}}_{l,c}$, which is the maximum activation of c recorded on a validation set.

Since the space of possible inputs that satisfy (1) is vast, and natural signals only occupy a small manifold of this high-dimensional space, we use regularization to constrain the input in terms of range and smoothness to better fit statistics of natural video signals. Specifically, we apply the following two regularizers, $\mathcal{R}_B$ and $\mathcal{R}_{TV}$, explicitly to the appearance and motion input of our networks.

3.2 Regularizing Local Energy

As first regularizer, $\mathcal{R}_B$, we enforce a local norm that penalizes large input values

$$\mathcal{R}_B(\mathbf{x}) = \begin{cases} N_B(\mathbf{x}) & \forall i,j,k : \sqrt{\sum_d \mathbf{x}(i,j,k,d)^2} \le B \\ +\infty, & \text{otherwise.} \end{cases} \tag{2}$$

with $N_B(\mathbf{x}) = \sum_{i,j} \left(\sum_d \mathbf{x}(i,j,k,d)^2\right)^{\frac{\alpha}{2}}$ and i, j, k are spatiotemporal indices of the input volume and d indexes either color channels for appearance input, or optical flow channels for motion input, B is the allowed range of the input, and α the exponent of the norm. Similar norms are also used in Simonyan et al. (2014), Yosinski et al. (2015), Mahendran and Vedaldi (2016b), with the motivation of preventing extreme input scales from dominating the visualization.

3.3 Regularizing Local Frequency

The second regularizer, $\mathcal{R}_{TV}$, penalizes high frequency content in the input, since natural signals tend to be dominated by low frequencies. We use a total variation regularizer based on spatiotemporal image gradients

$$\mathcal{R}_{TV}(\mathbf{x};\kappa,\gamma) = \sum_{ijkd} \left[\kappa \left((\nabla_x \mathbf{x})^2 + (\nabla_y \mathbf{x})^2\right) + \gamma (\nabla_t \mathbf{x})^2\right], \tag{3}$$

where i, j, k are used to index the spatiotemporal dimensions of input $\mathbf{x}$, d indexes the color and optical flow channels of the input, and ∇_x, ∇_y, ∇_t are the derivative operators in the horizontal, vertical and temporal direction, respectively. κ is used for weighting the degree of spatial variation and γ is an explicit slowness parameter that determines the regularization strength on the temporal frequency. By varying $0 \leq \gamma < \infty$ we can selectively penalize with respect to the slowness of the features at the input.

We now derive interesting special cases of (3) that we will investigate in our experiments:

- A spatially global regularizer, $\kappa > 0; \gamma = 0$ does not penalize variation over the temporal dimension, t. This choice produces reconstructions with unconstrained temporal frequency while only enforcing two-dimensional spatial smoothness in (3). This choice can be seen as an implicit low-pass filtering in the 2D spatial domain.
- An isotropic spatiotemporal regularizer, $\kappa = \gamma; \kappa, \gamma > 0$ equally penalizes variation in space and time. This can be seen as an implicit low-pass filtering in the 3D spatiotemporal domain.
- An anisotropic spatiotemporal regularizer, $\kappa \neq \gamma; \kappa, \gamma > 0$ allows balancing between space and time to e.g. visualize fast varying features in time that are smooth in space. The isotropic case above would bias the visualization to be smooth both in space and time, but not allow us to trade-off between the two.

3.4 Discussion

Purely spatial variation regularization is important to reconstruct natural images. Examples of application include image/video restoration (Zhang et al. 2010), feature inversion (Mahendran and Vedaldi 2016b), style transfer (Johnson et al. 2016), or activation maximization (Yosinski et al. 2015) where a 2D Gaussian filter was applied after each maximization iteration to achieve a similar effect. Isotropic spatiotemporal regularization relates to multiple hand-designed features that operate by derivative filtering of video signals, examples include HOG3D (Kläser et al. 2008), Cuboids (Dollar et al. 2005), or SOEs (Feichtenhofer et al. 2015). Finally, anisotropic spatiotemporal regularization relates to explicitly modelling the variation in the temporal dimension. Larger weights γ in (3) stronger penalize the temporal derivative of the signal and consequently enforce low-pass characteristic such that it varies slowly in time. This is a well studied principle in the literature. For learning general representations from video in an unsupervised manner, minimizing the variation across time is seen both in biological, e.g. Földiák (1991), Wiskott and Sejnowski (2002), and artificial, e.g. Goroshin et al. (2015) systems. The motivation for such an approach comes from how the brain solves object recognition by building a stable, slowly varying feature space with respect to time (Wiskott and Sejnowski 2002) in order to model temporally contiguous objects for recognition.

In summary, the regularization of the objective, (1), combines (2) and (3): $\mathcal{R}_r(\mathbf{x}) = \mathcal{R}_B(\mathbf{x}) + \mathcal{R}_{TV}(\mathbf{x};\kappa,\gamma)$. Thus, $\mathcal{R}_r(\mathbf{x})$ serves to bias the visualizations to the space of natural images in terms of their magnitudes and spatiotemporal rates of change. Note that the three different special cases of the variational regularizer for the motion input allow us to reconstruct signals that are varying slowly in space, uniformly in spacetime and non-uniformly in spacetime.

3.5 Implementation Details

For optimizing the overall objective, (1), we use ADAM (Kingma and Ba 2015) that adaptively scales the gradient updates on the input by its inverse square root, while aggregating the gradients in a sliding window over previous iterations. We use the same initializations as in Mahendran and Vedaldi (2016b). During optimization, we spatially shift (jitter) (Mordvintsev et al. 2015) the input randomly between 0 and the stride of the optimized layer. All preceding downsampling layers (pooling and strided convolutions) are accumulated into the layer stride that is used. The intention for jittering with a random stride that is less than the optimized layer is to interpolate between pixels in back-propagation. This generally results in increased reconstruction quality; an example can be found in Fig. 4 of Mahendran and Vedaldi (2016b). For all results shown in this paper, we chose the regularization/loss trade-off factors λ_r to provide similar weights for the different terms (2)–(3). We apply the regularizers separately to the optical flow and appearance input. The regularization terms for the appearance input are chosen to $\lambda_{B,\text{rgb}} = \frac{1}{HWB^\alpha}$ and $\lambda_{TV,\text{rgb}} = \frac{1}{HWV^2}$, with $V = B/6.5$, $B = 160$ and $\alpha = 3$, i.e. the default parameters in Mahendran and Vedaldi (2016b). Our work builds upon what has been used in Mahendran and Vedaldi (2016b) to invert image ConvNets; therefore, we use the same settings for reconstructing RGB images (appearance streams) and extend it for the 3D case to invert ConvNets trained on optical-flow (motion streams) as outlined below.

The motion input's regularization differs from that of appearance, as follows. In general, the optical flow is assumed to be smoother than appearance input; therefore, the total-variation regularization term of motion inputs has 10 times higher weight than the one for the appearance input. In order to visualize different speeds of motion signals, we use different weight terms for the variational regularizers of the motion input. In particular, to reconstruct different uniformly regularized spatiotemporal inputs we vary κ and set set $\gamma = \kappa$ for penalizing the degree of spatiotemporal variation for reconstructing the motion input (as we set $\gamma = \kappa$, we only list the values for κ in the experiments). For anisotropic spatiotemporal reconstruction, we vary the temporal slowness parameter, γ and fix $\kappa = 1$. The values in all visualizations are scaled to min-max over the whole sequence for effectively visualizing the full range of motion.

4 Experiments

In our detailed discussion in Sects. 4.1, 4.2 and 4.3, we focus our experimental studies on a VGG-16 two-stream fusion model (Feichtenhofer et al. 2016b) that is illustrated in Fig. 2 and trained on UCF-101. Our visualization technique, however, is generally applicable to any spatiotemporal architecture. In Sect. 4.4, we visualize various other architectures trained on multiple datasets.

We plot the appearance stream input directly by showing an RGB image and the motion input by showing the optical flow as a video that plays in the Animated Manuscript; the RGB channels of this video consist of the horizontal, vertical and magnitude of the optical flow vectors, respectively. It is our impression that the presented flow visualization is perceptually easier to understand than standard alternatives (e.g. HSV encoding).

4.1 Emergence of Spatiotemporal Features

We first study the conv5_fusion layer (i.e. the last local layer; see Fig. 2 for the overall architecture and Table 1 for the filter specification of the layers), which takes in features from the appearance and motion streams and learns a local fusion representation for subsequent fully-connected layers with global receptive fields. Therefore, this layer is of particular interest as it is the first point in the network's forward pass where appearance and motion information come together. At conv5_fusion we see the emergence of both class specific and class agnostic units (i.e. general units that form a distributed representation for multiple classes). We illustrate both of these by example in the following.

Table 1 Structure of the VGG-16 architecture showing the layer names, the receptive field sizes (RF size), and the feature stride and the output size of the feature maps, for an input of size 224 × 224×3. All convolutional layers use 3 × 3 filters with padding to preserve the input size

Layer	RF size	Stride	Output size
Data			$224 \times 224 \times 3$
conv1_1	3	1	$224 \times 224 \times 64$
conv1_2	5	1	$224 \times 224 \times 64$
pool1	6	2	$112 \times 112 \times 64$
conv2_1	10	2	$112 \times 112 \times 128$
conv2_2	14	2	$112 \times 112 \times 128$
pool2	16	4	$56 \times 56 \times 128$
conv3_1	24	4	$56 \times 56 \times 256$
conv3_2	32	4	$56 \times 56 \times 256$
conv3_3	40	4	$56 \times 56 \times 256$
pool3	44	8	$28 \times 28 \times 256$
conv4_1	60	8	$28 \times 28 \times 512$
conv4_2	76	8	$28 \times 28 \times 512$
conv4_3	92	8	$28 \times 28 \times 512$
pool4	100	16	$14 \times 14 \times 512$
conv5_1	132	16	$14 \times 14 \times 512$
conv5_2	164	16	$14 \times 14 \times 512$
conv5_3	196	16	$14 \times 14 \times 512$
pool5	212	32	$7 \times 7 \times 512$
fc6	404	32	$1 \times 1 \times 4096$
fc7	404	32	$1 \times 1 \times 4096$
fc8	404	32	$1 \times 1 \times N_{\text{cls}}$

N_{cls} denotes the number of output classes

4.1.1 Local Representation of Class Specific Units

In Fig. 1 we saw that some local filters might correspond to specific concepts that facilitate recognition of a single class (e.g. Billiards). We now reconsider that unit from Fig. 1 and visualize it under two further spatiotemporal regularization degrees, slow and fast temporal variation, in Fig. 3 (the visualization in Fig. 1 corresponds to medium speed, $\gamma = 1$).

Similar to Fig. 1, multiple coloured blobs as well as a billiards table with beige coloured structure on the top show up in the appearance (3a) input. Moving circular objects arise in the motion input (3b), but compared to Fig. 1, the motion is now varying differently in time, due to $\gamma = 5$. In Fig. 3c, d, we only regularize for spatial variation with unconstrained temporal variation, i.e. $\gamma = 0$ in (3). We observe that this unit is fundamentally different in the slow and the fast motion

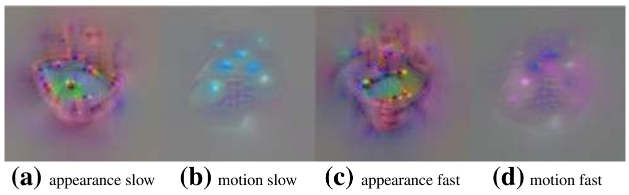

Fig. 3 Studying the Billiards unit at layer conv5_fusion from Fig. 1. We now show what highly activates the filter in the appearance and in the motion input space using high temporal variation regularization **a**, **b**: **c**, **d** show what excites the filter when there is no regularization on the temporal variation of the input: The appearance, **c** now shows different structure and the motion filter **d** now detects accelerating motion patterns e.g. when balls are accelerated

case: It looks for linearly moving circular objects in the slow spatiotemporal variation case, while it looks for an accelerating motion pattern into various directions in the temporally unconstrained (fast) motion case. It appears that this unit is able to detect a particular spatial pattern of motion, while allowing for a range of speeds and accelerations. Such an abstraction presumably has value in recognizing an action class with a degree of invariance to exact manner in which it unfolds across time.

Another interesting fact is that switching the regularizer for the motion input, also has an impact on the appearance input (Fig. 3a vs. c) even though the regularization for appearance is held constant. This fact empirically verifies that the fusion unit also expects specific appearance when confronted with particular motion signals.

We now consider unit f004 at conv5_fusion in Fig. 4. It seems to capture some drum-like structure in the center of the receptive field, with beige-colored structures in the upper region. This unit could relate to the PlayingTabla class. In Fig. 4 we show the unit under different spacetime regularizers and also show sample frames from three PlayingTabla videos from the test set. Interestingly, when stronger regularization is placed on both spatial and temporal change (e.g. $\kappa = 10$, top row) we see that a beige colour blob is highlighted in the appearance and a horizontal motion blob is highlighted in the motion in the same area, which combined could capture the characteristic head motions of a drummer. In contrast, with less constraint on motion variation (e.g. $\gamma = 0$, bottom row) we see that the appearance more strongly highlights the drum region, including hand and arm-like structures near and over the drum, while the motion is capturing high frequency oscillation where the hands would strike the drums. Significantly, we see that this single unit fundamentally links appearance and motion: We have the emergence of true spatiotemporal features.

4.1.2 Distributed Representation of General Units

In contrast to units that seem very class specific, we also find units that seem well suited for cross-class representation. To

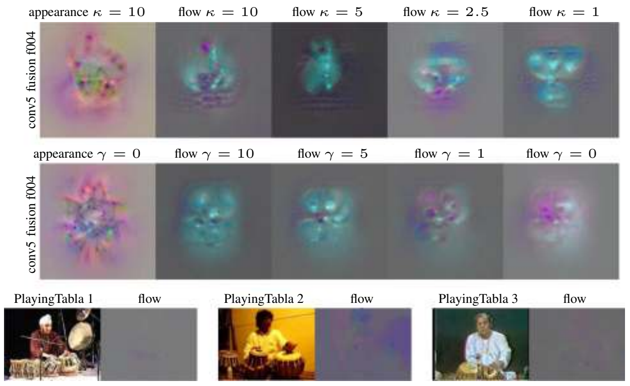

Fig. 4 Specific unit at conv5_fusion. Comparison between isotropic and anisotropic spatiotemporal regularization for a single filter at the last convolutional layer. The columns show the appearance and the motion input generated by maximizing the unit, under different degrees of isotropic spatiotemporal (κ) and anisotropic spatiotemporal TV regularization (γ). The last row shows frames from sample videos of appearance and optical flow from the PlayingTabla class

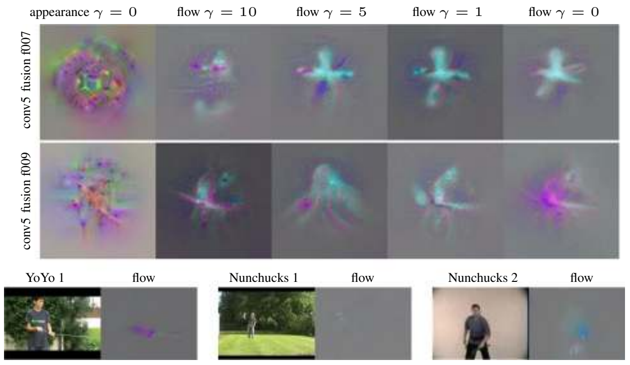

Fig. 5 Two general units at the convolutional fusion layer. The columns show the appearance and the motion input generated by maximizing the unit, under different degrees of anisotropic spatiotemporal regularization (γ). Note that $\kappa = 1$ for the anisotropic case. The last row shows frames from videos from the YoYo and Nunchucks classes

begin, we consider filters f006 and f009 at the conv5_fusion layer that fuses from the motion into the appearance stream, as shown in Fig. 5. These units seem to capture general spatiotemporal patterns for recognizing classes such as YoYo and Nunchucks, as seen when comparing the unit visualizations to the sample videos from the test set.

Next, in Fig. 6, we similarly show general feature examples for the conv5 fusion layer that seem to capture general spatiotemporal patterns for recognizing classes corresponding to multiple ball sport actions such as Soccer or TableTennis. These visualizations reveal that at the last convolutional layer the network builds a local representation that can be both distributed over multiple classes and quite specifically tuned to a particular class (e.g. Fig. 4 above).

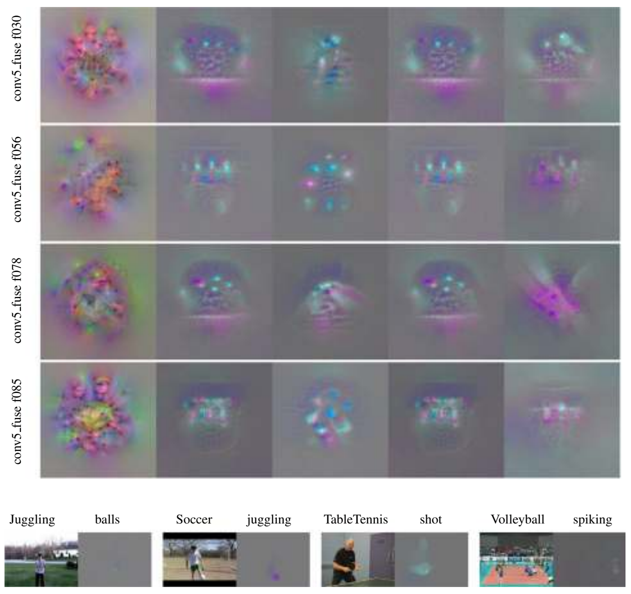

Fig. 6 General units at the convolutional fusion layer that could be useful for representing ball sports. The columns show the appearance and the motion input generated by maximizing the unit, under different degrees of temporal regularization (γ). The last row shows frames from sample videos from UCF101

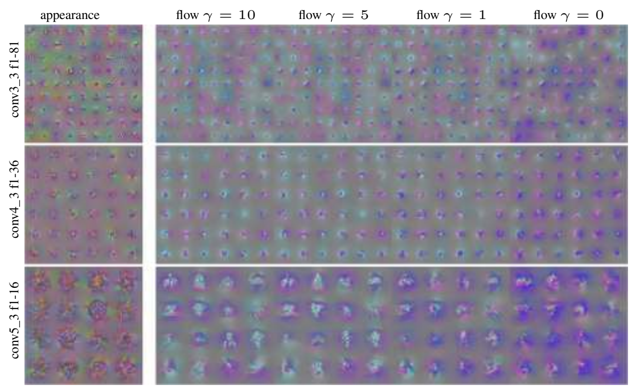

Fig. 7 Two-stream conv filters under anisotropic regularization. We show appearance and optical flow inputs for slowest $\gamma = 10$, slow $\gamma = 5$, fast $\gamma = 1$, and unconstrained (fastest) $\gamma = 0$, temporal variation regularization. Spatial regularization is kept constant. See the Animated Manuscript for animation

4.2 Progressive Feature Abstraction with Depth

4.2.1 Visualization of Early Layers

We now explore the layers of a VGG-16 Two-Stream architecture (Feichtenhofer et al. 2016b). Unlike in conv5_fusion layer where the inputs (appearance and flow) are produced by the same unit, inputs from early layers (i.e. conv1_1 to conv4_3) do not have correspondence as they are produced by two separate units in two different streams.

In Fig. 7 we show what excites the convolutional filters of a two-stream architecture at the early layers of the network hierarchy. We use the anisotropic regularization in space and time that penalizes variation at a constant rate across space and varies according to the temporal regularization strength, γ, over time. We observe that at earlier layers filters appear that perform differential spatial derivative filtering of multiple orders. For example the filter in row 2, column 2 from top right of the optical flow filters at conv3_3 in Fig. 7 exhibits centre-surround structure that operates across different motion directions to be matched to horizontal motion in the surround (indicated by the red outer region) and an accelerating vertical motion in the centre (indicated by the blue-cyan transition over time). Such a filter can be interpreted as akin to a spatiotemporal Laplacian that performs directional motion gradient filtering in spacetime. The emergence of such filters is interesting on its own as similar filters that show spatially differential structure across different channels appear in biological systems for the dimensions of colour (Gouras 1974; Livingstone and Hubel 1984), spatial orientation (Gorea and Papathomas 1993) and direction of motion (Stromeyer et al. 1984; Reichardt et al. 1983).

Moreover, we see that the spatial patterns are preserved throughout various temporal regularization factors γ, at all layers. From the temporal perspective, we see that, as expected, for decreasing γ the temporal variation increases; interestingly, however, the directions of the motion patterns are preserved while the optimal motion magnitude varies with γ. For example, consider the last shown unit f36 of layer conv4_3 (bottom right filter in the penultimate row of Fig. 7). This filter is matched to motion blobs moving in an upward direction. In the temporally regularized case, $\gamma > 0$, the motion is smaller compared to that seen in the temporally unconstrained case, $\gamma = 0$. Notably, all these motion patterns strongly excite the same unit. These observations suggest that the network has learned speed invariance, i.e. the unit can respond to the same direction of motion with robustness to speed. Such an ability is significant for recognition of actions irrespective of the speed at which they are executed, e.g. being able to recognize “running” without a concern for how fast the runner moves.

In Fig. 8 we first show what excites the convolutional filters of a two-stream architecture when varying the isotropic spatiotemporal regularization. The motion signals are reconstructed under spatiotemporal regularization with different regularization strengths, κ, and $\gamma = \kappa$. Varying the regularization in spacetime reveals interesting properties of the underlying representation. We discuss Fig. 8 from two perspectives: First, from the temporal perspective, we see that the early layer filters are more robust to regularization in spacetime, whereas higher layers show larger dependence on the regularization strength, κ. This dependence originates from the temporally consistent nature of the early filters that exhibit temporal low-pass characteristics. Second, from the spatial perspective, we see that with decreasing spacetime

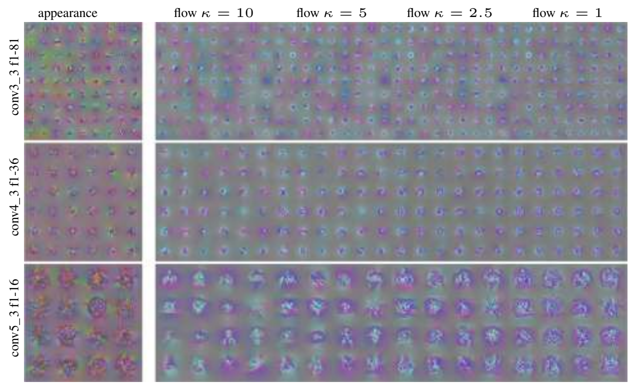

Fig. 8 Two-stream conv filters under isotropic ($\gamma = \kappa$) spatiotemporal variation regularization. We show appearance and the optical flow inputs for slowest $\kappa = 10$, slow $\kappa = 5$, fast $\kappa = 2.5$, and faster $\kappa = 1$ spatiotemporal variation. See the optical flow in the Animated Manuscript for video playback

regularization strength, κ, high-frequency inputs become dominant. Especially for low regularization factors $\kappa \leq 2.5$, we see high-frequency patterns dominating and reconstruction artifacts appearing in the background.

In Fig. 9 we show the evolution of internal filters of the VGG16 motion stream during training. After 10 epochs we see very minor temporal variation of the filters which increases when going towards the end of learning at epoch 60. Interestingly, the spatial shape of the early layer filters at conv2_2 does not change while the temporal dimension builds up over epochs. The higher layers, in contrast, undergo large changes of spatiotemporal structure from epoch 10 to epoch 60.

4.2.2 Visualization of Fusion Layers

We now briefly re-examine the convolutional fusion layer (as in the previous Sect. 4.1). In Fig. 10, we show filters at the conv5_fusion layer, which fuses from the motion into the appearance stream, while varying the temporal regularization and keeping the spatial regularization constant. This result is again achieved by varying the parameter γ in (3). The visualizations reveal that these fusion filters at this last convolutional layer show reasonable combinations of appearance and motion information, qualitative evidence that the fusion model in Feichtenhofer et al. (2016b) performs as desired. For example, the receptive field centre of conv5_fusion f002 seems matched to lip like appearance with a juxtaposed elongated horizontal structure, while the motion is matched to slight up and down motions of the elongation (e.g. flute playing). Once again, we also observe that the units are broadly tuned across temporal input variation (i.e. all the different inputs highly activate the same given unit).

In the last column of Fig. 10 we plot the maximum test set activity of corresponding filters at the fusion layer, for the 10

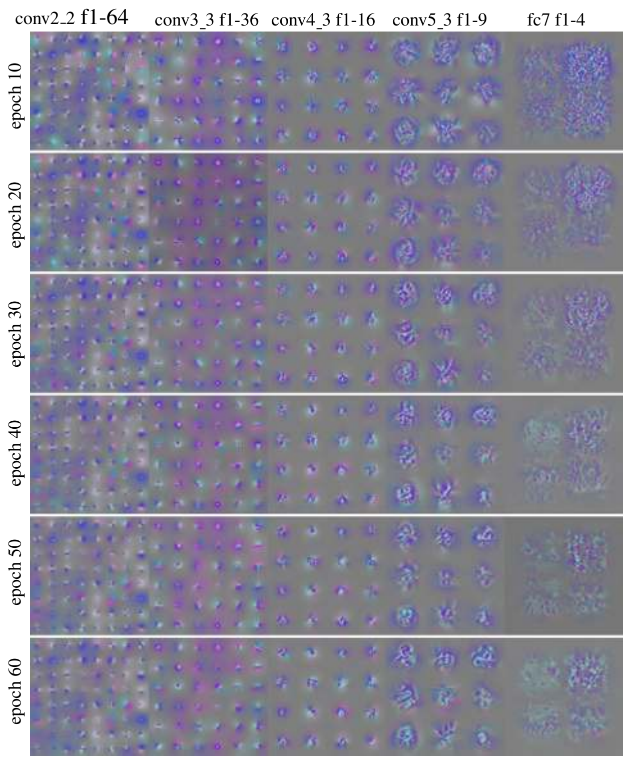

Fig. 9 Evolution of filters throughout training. We show the optical flow inputs for the motion stream filter maximization under spatiotemporal variation. Filters at early layers adapt quickly whereas filters at the higher layers emerge later on in training, especially over the temporal dimension

highest activating videos across the test set. Most of the filters fire for classes with similar appearance and motion, but also strong activity over a broad set of classes can be observed.

4.2.3 Visualization of Global Layers

We now visualize the layers that have non-local filters, i.e. fully-connected layers that operate on top of the convolutional fusion layer illustrated above. Figure 11 shows filters of the fully-connected layers 6 (fc_6) and 7 (fc_7) of the VGG-16 fusion architecture. We again show the maximum test set activity of corresponding units at the fully connected layers. In contrast to the local features above, we observe a holistic representation that consists of a mixture of the local units seen in the previous layer. For example, in the fc_6 units shown in the top three rows we observe features that could support prediction of Basketball and PlayingFlute. In the fc_7 units shown in the last three rows we see units that resemble Bowling and the Clean and Jerk actions, revealed by the maximum test set activity shown in the last column. Here, it is notable that these representations form something akin to a nonlinear (fc_6) and linear (fc_7) basis for the prediction layer; therefore, it is plausible that the filters resemble holistic classification patterns.

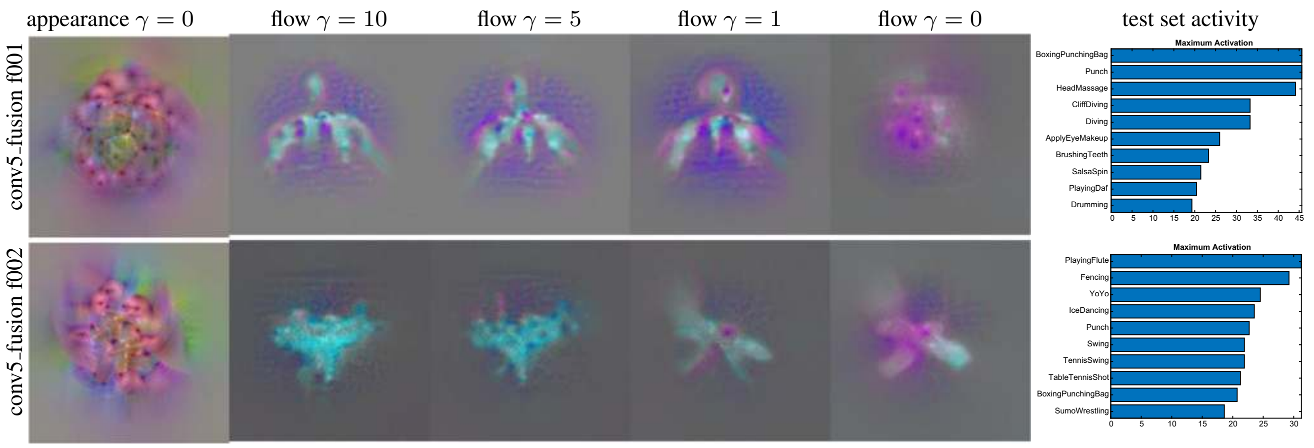

Fig. 10 Visualization of the conv5_fusion layer under different temporal TV regularization. We show the appearance input and the optical flow inputs for slowest $\gamma = 10$, slow $\gamma = 5$, fast $\gamma = 1$, and unconstrained $\gamma = 0$, temporal variation regularization. The last column shows the maximum activity observed for classes in the test set. Best viewed electronically

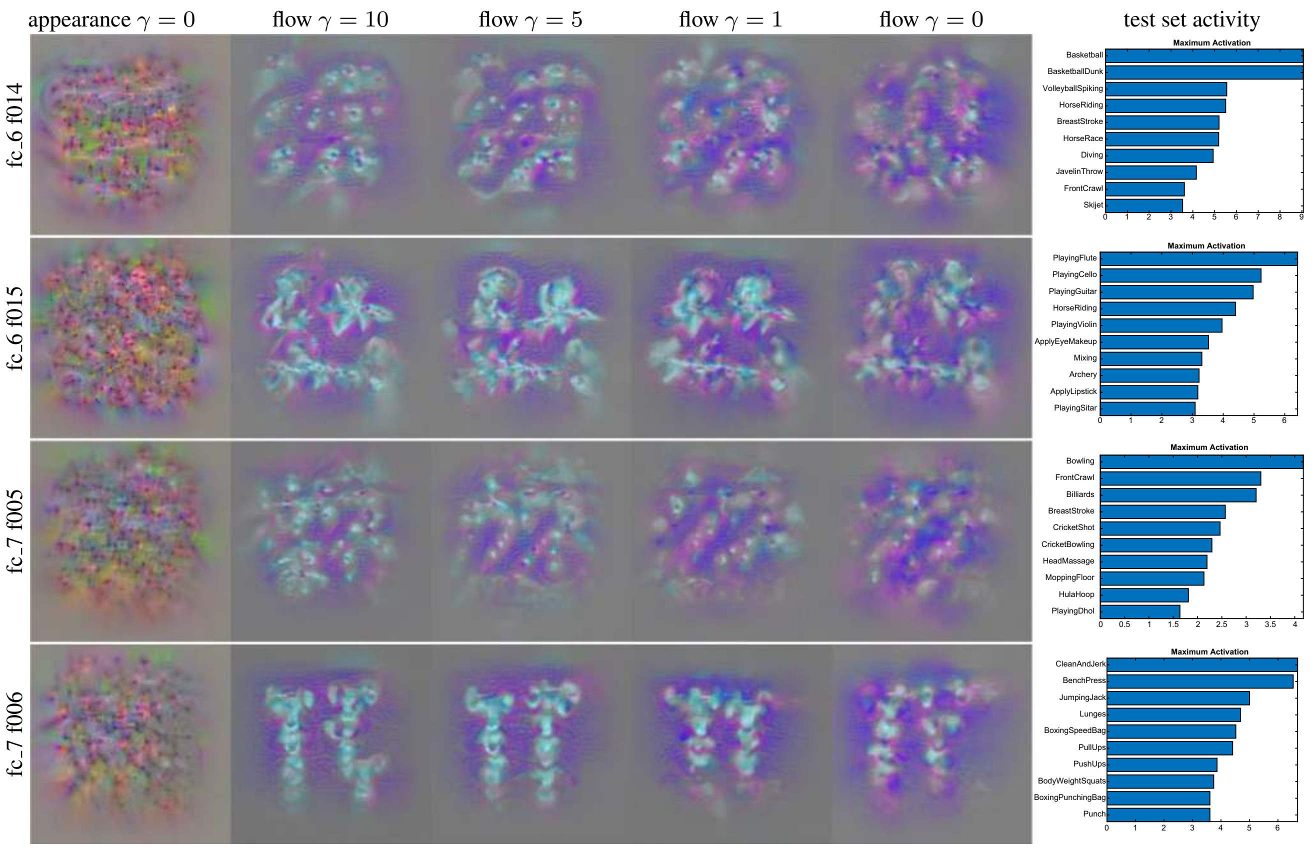

Fig. 11 Visualization of filters of the fc_6 and fc_7 layers under different temporal regularization. We show the appearance input and the optical flow inputs for slowest $\gamma = 10$, slow $\gamma = 5$, fast $\gamma = 1$, and unconstrained (fastest) $\gamma = 0$, temporal variation regularization. The last column shows the maximum activity observed for classes in the test set

Finally, we visualize the ultimate class prediction layers of the architecture, where the unit outputs correspond to different classes; thus, we know to what they should be matched. In Fig. 12, we show the fast motion activation of the classes Archery, BabyCrawling, PlayingFlute and CleanAndJerk and BenchPress. The learned features for archery (e.g., the elongated bow shape and positioning of the bow as well as the shooting motion of the arrow) are markedly

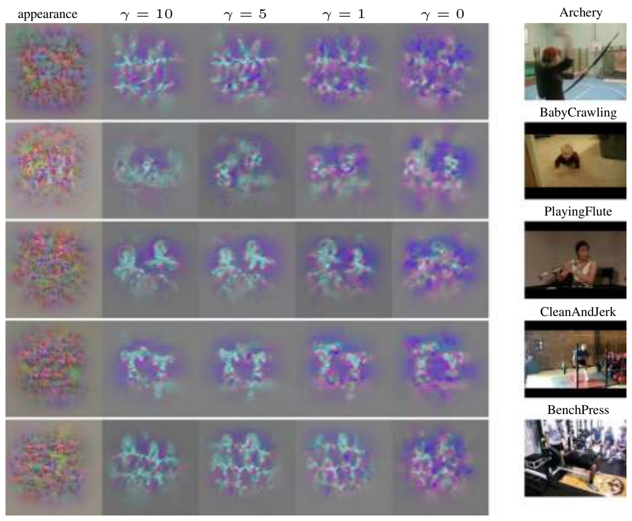

Fig. 12 Classification units at the last layer of the network. The first column shows the appearance and the second to fifth columns the motion input generated by maximizing the prediction layer output for the respective classes, with different degrees of temporal variation regularization (γ). The last column shows a sample frame from the first video of that class in the test set

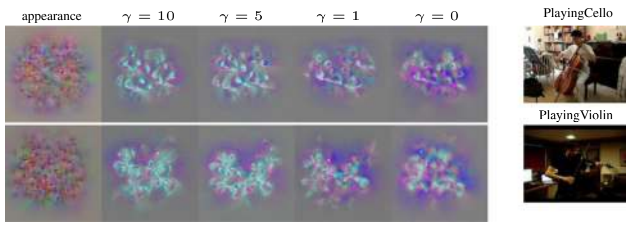

Fig. 13 Explaining confusion for PlayingCello and PlayingViolin. We see that the learned representation focuses on the vertical (Cello) and horizontal (Violin) alignment of the instrument, which could explain confusions for videos where this is less distinct

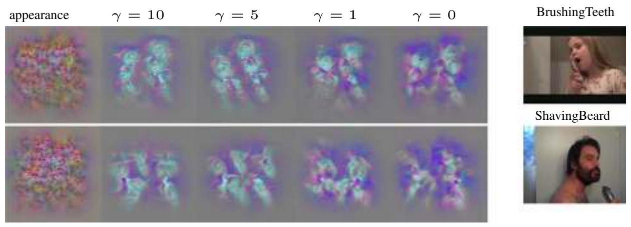

Fig. 14 Explaining confusion between BrushingTeeth and ShavingBeard. The representation focuses on the common local appearance of face and lips as well as the local motion of the tool

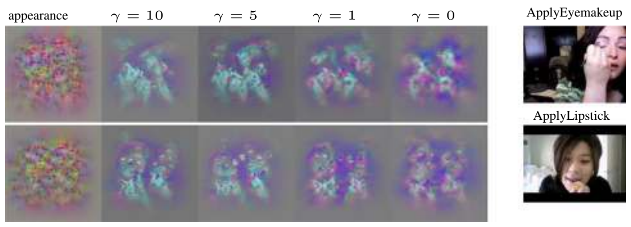

Fig. 15 Classification units for ApplyEyemakeup and ApplyLipstick. Surprisingly, the prediction unit for ApplyLipstick gets excited by moving eyes at the motion input. Presumably this activation reflects a peculiarity of the dataset which contains samples of the ApplyEyemakeup class with eyes appearing static

distinct from those of the baby crawling (e.g., capturing the facial parts of the baby appearance while focusing on the arm and head movement in the motion representation), and those of PlayingFlute (e.g. filtering eyes and arms (appearance) and moving arms below the flute (motion)), as well as those of CleanAndJerk and BenchPress (e.g. capturing barbells and human heads in the appearance with body motion for pressing ($\gamma = 0$) and balancing ($\gamma = 10$) the weight). Further, Clean and Jerk actions (where a barbell weight is pushed over the head in a standing position) contrasts to the Benchpress action (which is performed in lying position on a bench). Notice how the difference in relative body position is captured in the visualizations, e.g. the relatively vertical vs. horizontal orientations of the regions captured beneath the weights, especially in the motion visualizations. Thus, we find that the class prediction units have learned representations that are well matched to their classes.

4.3 Utilizing Visualizations for Understanding Failure Modes and Dataset Bias

Another use of our visualizations is to debug the model and reason about failure cases. In UCF101 15% of the PlayingCello videos get confused as PlayingViolin. In Fig. 13, we observe that the subtle differences between the classes are related to the alignment of the instruments. In fact, this is in concordance with the confused videos in which the Violins are not aligned in a horizontal position.

In UCF101 the major confusions are between the classes BrushingTeeth and ShavingBeard. In Fig. 14 we visualize the inputs that maximally activate these classes and find that they are quite similar, e.g. capturing a linear structure moving near the face, but not the minute details that distinguish them. This insight not only explains the confusion, but also can motivate remediation, e.g. focused training on the uncaptured critical differences (i.e. tooth brush vs shaver).

Dataset bias and generalization to unseen data is important for practical applications. Two classes, ApplyEyemakeup and ApplyLipstick are, even though being visually very similar, easily classified in the test set of UCF101 with classification rates above 90% (except for some obvious confusions with BrushingTeeth). This result makes us curious, so we inspect the visualizations in Fig. 15. The inputs are capturing facial features, such as eyes, and the motion of applicators. Interestingly, it seems that ApplyEyemakeup and ApplyLipstick are being distinguished, at least in part, by the fact that eyes tend to move in the latter case, while they are held static in the former case. Here, we see a benefit of our visualiza-

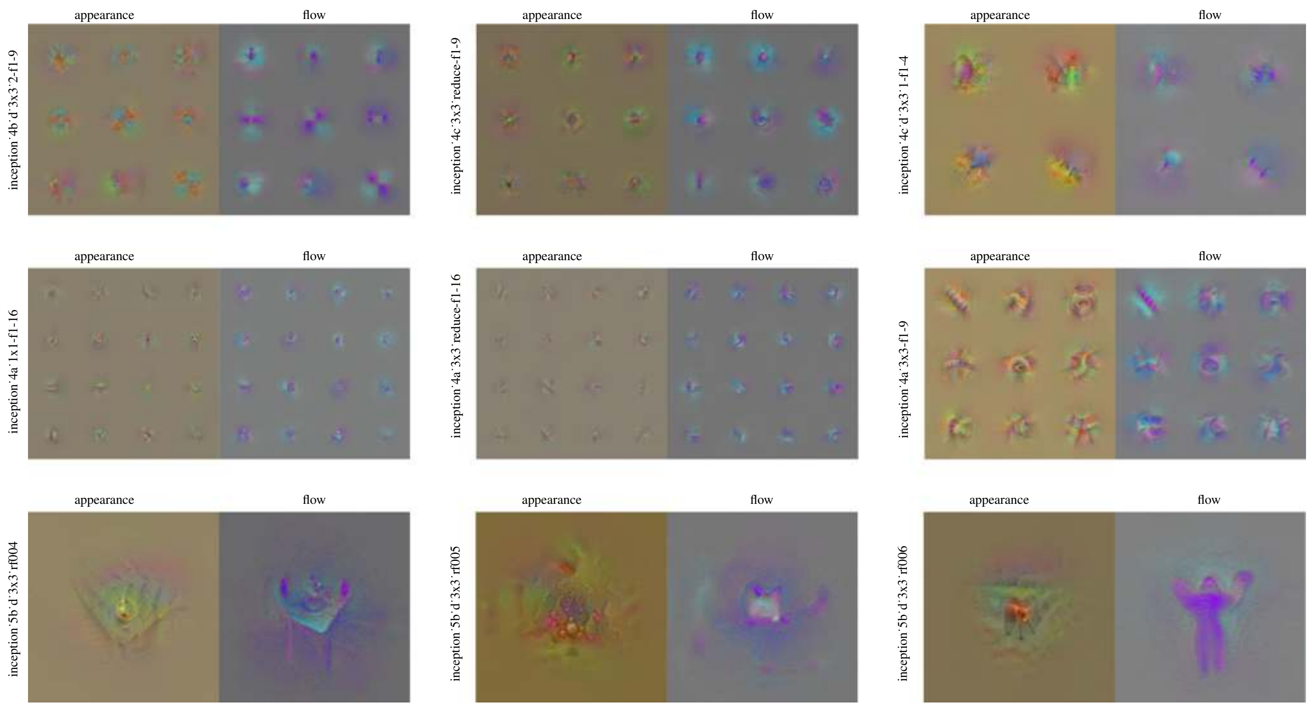

Fig. 16 Visualizations of convolutional layers of the BN-Inception (Ioffe and Szegedy 2015) two-stream network (Wang et al. 2016) trained on HMDB51 (Kuehne et al. 2011). The number of filters shown at each layer is inversely proportional to its receptive field

tions beyond revealing what the network has learned—they also can reveal idiosyncrasies of the data on which the model has been trained.

4.4 Visualizing Multiple Architectures and Datasets

We focus all our experimental studies above on a VGG-16 two-stream fusion model (Feichtenhofer et al. 2016b) that is trained on UCF-101. Our visualization technique, however, is generally applicable to any spatiotemporal architecture. In this section, we visualize various other architectures: Spatiotemporal Residual Networks (Feichtenhofer et al. 2016a) using ResNet50 (He et al. 2016) streams, Temporal Segment Networks (Wang et al. 2016) using BN-Inception (Ioffe and Szegedy 2015) or Inception_v3 (Szegedy et al. 2015) streams, trained on multiple datasets: UCF101 (Soomro et al. 2012), HMDB51 (Kuehne et al. 2011) and Kinetics (Carreira and Zisserman 2017). Our visualization method is general and can also be applied to any other 3D ConvNet architecture such as C3D (Tran et al. 2015) or I3D (Carreira and Zisserman 2017) networks. All results in this Sect. 4.4 are shown for constant spatiotemporal regularization during reconstruction.

It is noteworthy that deeper models (e.g. ResNet & Inception_v3) are inherently harder to optimize, since we do not use batch normalization (BN) for activation maximization; essentially, we absorb the BN layers by projecting them into the preceding conv-layer weights. Moreover, for extremely deep nets, the jitter-based regularization becomes more important for filters that have large strides on the input. Deeper layers typically have large cumulative filter strides when backprojected to the input, which causes a subsampling effect when optimizing that can be ameliorated by the spatial jittering during optimization.

4.4.1 Visualization of Inception Networks

We show results for models based on a batch-normalized GoogLeNet Inception architecture (Ioffe and Szegedy 2015). An interactive web-based illustration of the GoogLeNet (ImageNet) architecture, showing the layer names with respective number of parameters (ch) and output resolution $(width \times height)$ for a 227×227 sized input, can be found here. The models are taken from the Temporal Segment Networks (TSN) approach (Wang et al. 2016), which pretrains the motion stream on TVL1 optical flow (Zach et al. 2007), IDT-flow (Wang and Schmid 2013) (termed warped flow in Wang et al. (2016)) and difference images of UCF101. This extra data yields highest accuracy of the motion stream in UCF101 and HMDB51. In Fig. 16, we show what the convolutional filters of the motion and appearance stream are capturing, for a TSN model trained on HMDB51 (and pretrained on ImageNet & UCF101). When going from low to higher layers, we observe two things: First, the filters become more task specific towards HMDB51 classes, which is as expected; for example, the motion filter f006 at a high-

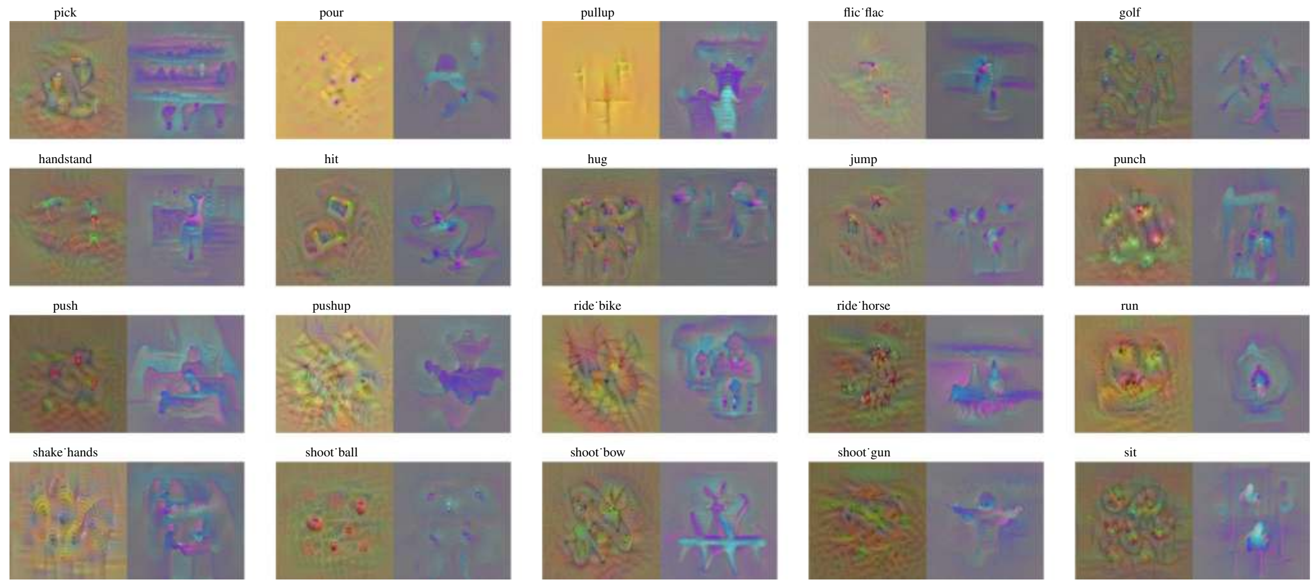

Fig. 17 Visualizations of classification units at the last layer of the BN-Inception (Ioffe and Szegedy 2015) two-stream network (Wang et al. 2016) trained on HMDB51 (Kuehne et al. 2011). We show pairs of appearance and motion input generated by maximizing the prediction layer output for the respective classes listed above

est conv-layer shown in the last visualization of Fig. 16 is activated by an optical flow pattern that could relate to the "pull-up" action in HMDB51. Second, we see that the inputs for the appearance and motion streams are spatially similar at early layers and become dissimilar later. This suggests that the prior of the (ImageNet) pretraining on the spatial shape of the filters is dominant in the lower layers of the networks and is similar to our observations made for the VGG-16 architecture reported in Sect. 4.2.

Interestingly, as with visualization of filters in the early layers of VGG16 (Sect. 4.2), we once again find the emergence of cross-channel differential filters in the Inception architecture. This point is exemplified through consideration of the first nine filters in the lowest layer in Fig. 16, inception_4b_d_3 $\times$ 3_2. For example, the units shown at row and column (2,2) and (3,3) resemble mixed first partial derivative filters, while the unit at (2,3) can be seen as a Gaussian second derivative filter across the horizontal direction and the filter at (2,1) is close to a Gaussian second derivative filter across the vertical direction; thus, in tandem these filters appear to provide a steerable basis set for a Gaussian second-derivative operator (cf. Freeman and Adelson 1991, Fig. 16), but here as taken across the horizontal and vertical motion directions.

When maximizing the class level units we see clearly recognizable pattern structure. Some of the most distinctive examples from the 20 class samples shown in Fig. 17 are pullup, flic-flac, golf, ride horse, shake hands, shoot bow, shoot gun. Here, an interesting observation is that the appearance and motion stream focus on different objects and scales; exemplarily the ride bike class is highly activated by spokes or bike frames, while the motion stream tends to fire for a frontal facing motion of a cyclist.

In general, we find that earlier filters (Fig. 16) seem to provide reasonable primitives for higher level abstractions (e.g. derivative filters), while classification level units (Fig. 17) show clear structure matched to their classes. In between, the intermediate units can capture incremental abstractions from the primitives to the class level representations. In contrast, other units, especially at the intermediate layers (Fig. 16), often seem less readily interpretable. One possible explanation for this pattern of results is that the capacity of the network exceeds that of the recognition task at hand and uninterpretable units may result simply from overfitting to the training data. Indeed, we observe similar overall patterns of abstraction across all visualized networks that we study in this paper.

4.4.2 Visualization of Spatiotemporal Residual Networks

We now show results of our approach applied for visualization of the filters in a Spatiotemporal Residual Network (Feichtenhofer et al. 2016a) trained on UCF101. This architecture uses two ResNet-50 (He et al. 2016) streams. An interactive web-based schematic of the ResNet-50 (ImageNet) architecture, showing the layer names with respective number of parameters (ch) and output resolution (width $\times$ height) for a 224 $\times$ 224 sized input, can be found here. It is noteworthy that the visualized Spatiotemporal Residual Network (Feichtenhofer et al. 2016a) architecture consists of a fusion stream that operates on appearance and motion information, as well as a pure motion stream that projects into the fusion stream after five layers (i.e. res2a, res3a, res4a, res5a) throughout the network hierarchy (for the detailed architecture and connections between the two ResNet50 streams see Feichtenhofer et al. 2016a). Therefore, it is interesting to study the filters in the fusion stream and compare them to the parallel motion filters in the other stream. We show results for multiple layers throughout the network hierarchy in Fig. 18. The Animated Manuscript shows the filters animated as a video; the motion stream filters should automatically play on repeat.

When looking at the filters, we see interesting abstract patterns appearing that show strong spatiotemporal correlation for the fusion filters, while the motion stream (right column), which projects as a residual connection into the parallel fusion stream, is maximized mostly by slightly different motion patterns, revealing a degree of complementarity between the fusion and motion stream features. The figure also qualitatively shows that the fusion stream learns spatially correlated patterns of flow and appearance. For example by clicking (in the Animated Manuscript) on the fusion stream visualizations of the 9 filters at the layer res4b-branch2b-spatial, shown in the third row and left column of Fig. 18, we observe motion units that have high energy at regions of uniform appearance structure (e.g. pink appearance correlating with vertical motion in the top left unit).

Also interesting is the fact that the lower fusion stream layers, shown in Fig. 18, exhibit only weaker motion information (noisy reconstructions); this result can be seen as evidence for the need of the parallel motion stream to build up these abstract, strong motion features later on in the hierarchy of the network. The reasons for this state of affairs is discussed below.

We observe that the visualized motion information right before the next fusion connection is very noisy (see fusion stream visualizations of res3a, res4a and res5a in Fig. 18), i.e. the fusion (which is performed after res2a, res3a, res4a and res5a) leads to very rudimentary motion features for layers that come several layers after the previous fusion layer. Contrarily, we observe stronger spacetime features when inspecting features from the fusion stream that come directly after a fusion connection, e.g. at res4b. This observation suggests that layers which are positioned several layers distant to the previous fusion connection are mostly activated by appearance information. An explanation could be that motion information is only fused into the residual units of the fusion stream and hence only locally affects the layers after the fusion. The local fusion, however, is important for good performance, as the work in Feichtenhofer et al. (2017) shows that a direct fusion connection of the streams produces clearly inferior results because it induces too large a change in the propagated signals, thereby disturbing the network's representation abilities; i.e. it would impair the appearance filters. Nevertheless, the low-energy motion filters in our visualizations illustrate the sub-optimality of this local fusion strategy, as the fusion network 'forgets' the motion information and provides insight into why a separate motion stream is required to build up abstract motion features. The observations can be used as a substrate for future work on designing a better fusion architecture that learns filters sensitive to both appearance and motion information.

4.4.3 Visualization of Inception_v3 Networks

We finally explore an Inception_v3 Two-Stream model that was trained on Kinetics (Carreira and Zisserman 2017). An interactive web-based schematic of the Inception_v3 (ImageNet) (Szegedy et al. 2015) architecture, showing the layer names with respective number of parameters (ch) and output resolution (width $\times$ height) for a 299×299 sized input, can be found at http://dgschwend.github.io/netscope/#/preset/inceptionv3.

Figure 19 shows the convolutional filters at the intermediate layers of this network. Interestingly, similar patterns appear in the appearance and the optical flow path, even though these streams were not connected during training (i.e. no fusion between the streams). We think that the similarities in the spatial structures of the filters for appearance and motion, despite being trained separately from different input modalities, are due to similar initialization (i.e. via ImageNet). We also observe that temporal variation increases when going deeper in the network hierarchy which is in accordance with other architectures and datasets shown above.

The most interesting finding here is unrelated to video. Consider the second row in Fig. 19, which shows three consecutive filterbanks at the mixed4 convolutional block that are located in the centre of the network hierarchy. The first illustration shows 9 filters at layer mixed_4_tower_1_conv_1, which is a filterbank with 128 filters of dimension 7×1 in width and height that receives its input from a 1×1 filterbank. The learned kernels of that layer show elongated horizontal structure in the visualizations. Another filterbank that also takes the output of a 1×1 kernel, and filters it with filters of dimension 1×7 in width and height, is shown in the second column of the second row in Fig. 19. The visualizations clearly show that the energy of these units is distributed across vertically elongated structures in the input. Finally, the filterbank shown in the last column of the second row takes the output of a 7×1 kernel and filters it with a 1×7 kernel. The visualizations show that filters emerge that distribute their energy at a large window of the input receptive field. Notably, the energy of these filters is distributed more evenly across the receptive field than it would be for a single spatial filter. This point is qualitatively shown in the filterbank of the mixed_4_tower_2_conv filter in the third row, first column,

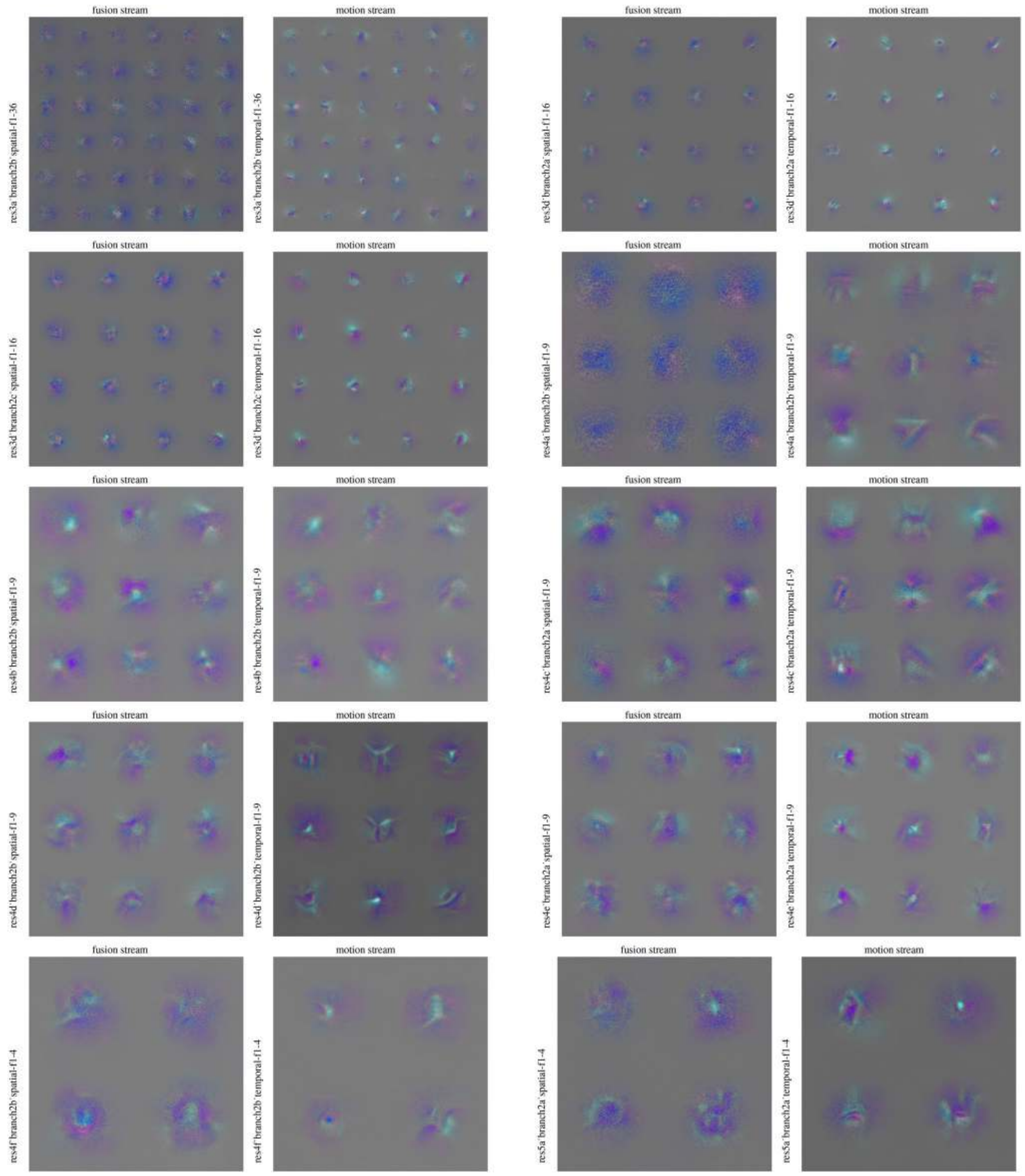
fusion stream
res3a_branch2b_spatial-f1-36
motion stream
res3a_branch2b_temporal-f1-36
fusion stream
res3d_branch2a_spatial-f1-16
motion stream
res3d_branch2a_temporal-f1-16
fusion stream
res3d_branch2c_spatial-f1-16
motion stream
res3d_branch2c_temporal-f1-16
fusion stream
res4a_branch2b_spatial-f1-9
motion stream
res4a_branch2b_temporal-f1-9
fusion stream
res4b_branch2b_spatial-f1-9
motion stream
res4b_branch2b_temporal-f1-9
fusion stream
res4c_branch2a_spatial-f1-9
motion stream
res4c_branch2a_temporal-f1-9
fusion stream
res4d_branch2b_spatial-f1-9
motion stream
res4d_branch2b_temporal-f1-9
fusion stream
res4e_branch2a_spatial-f1-9
motion stream
res4e_branch2a_temporal-f1-9
fusion stream
res4f_branch2b_spatial-f1-4
motion stream
res4f_branch2b_temporal-f1-4
fusion stream
res5a_branch2a_spatial-f1-4
motion stream
res5a_branch2a_temporal-f1-4

Fig. 18 Visualizations of convolutional layers of Spatiotemporal Residual Networks (Feichtenhofer et al. 2016a) using ResNet50 (He et al. 2016) streams trained on UCF101. The left column shows spatiotemporal filters of the fusion stream with motion and appearance playing sequentially (see Animated Manuscript for animation), whereas the right column shows the motion stream that projects into the fusion stream

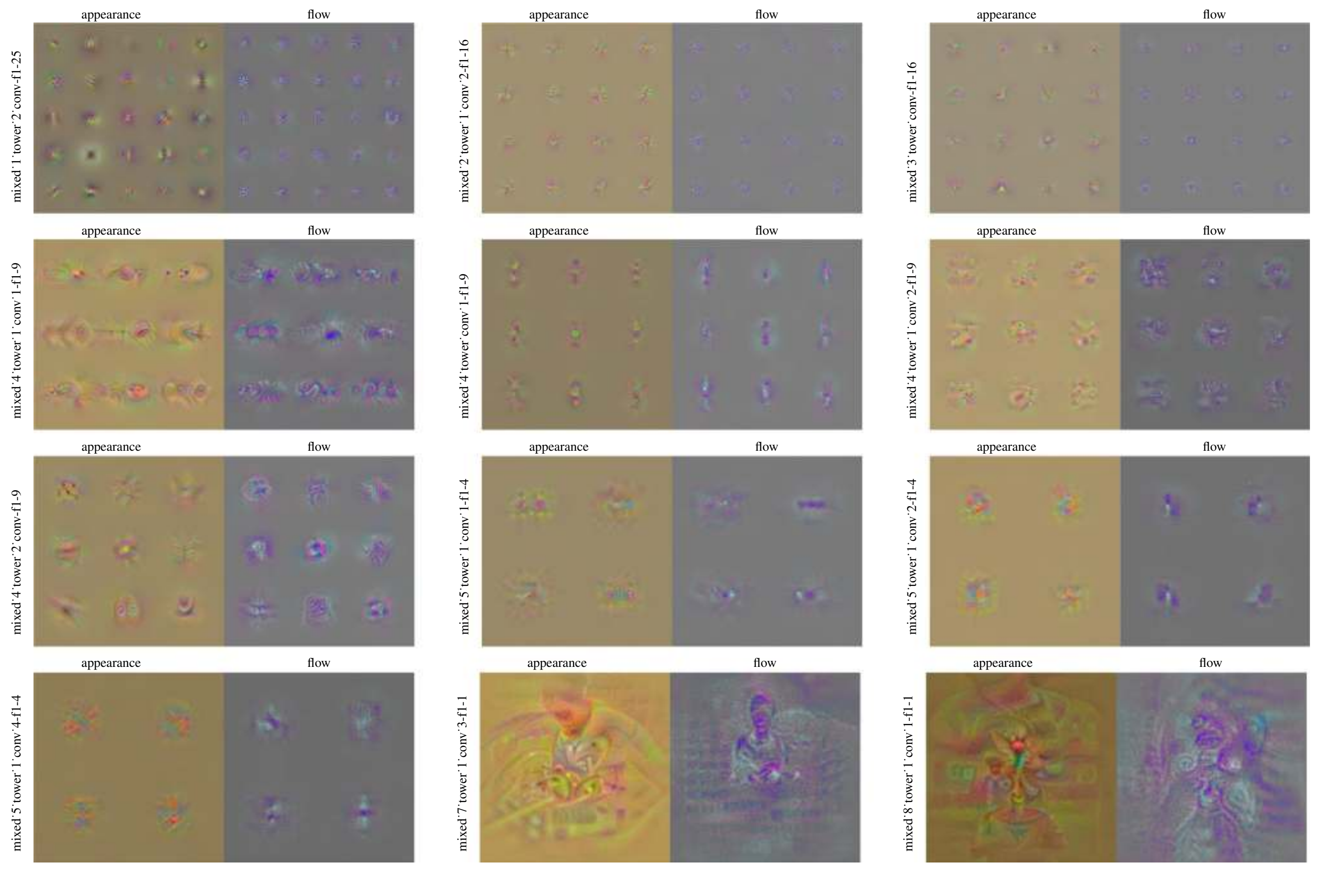
appearance
flow
mixed˙1˙tower˙2˙conv-f1-25
mixed˙2˙tower˙1˙conv˙2-f1-16
mixed˙3˙tower˙conv-f1-16
mixed˙4˙tower˙1˙conv˙1-f1-9
mixed˙4˙tower˙conv˙1-f1-9
mixed˙4˙tower˙1˙conv˙2-f1-9
mixed˙4˙tower˙2˙conv-f1-9
mixed˙5˙tower˙1˙conv˙1-f1-4
mixed˙5˙tower˙1˙conv˙2-f1-4
mixed˙5˙tower˙1˙conv˙4-f1-4
mixed˙7˙tower˙1˙conv˙3-f1-1
mixed˙8˙tower˙1˙conv˙1-f1-1

Fig. 19 Visualizations of convolutional layers of the Inception_v3 Two-Stream network trained on Kinetics (Carreira and Zisserman 2017). The number of filters shown at each layer is inversely proportional to its receptive field

which is a 1×1 kernel that succeeds a 3×3 average pooling layer and learns filters that center their energy in the receptive field.

In summary, two important observations can be made. First, separated filters kernels seem to learn explicit horizontal or vertical structure—a finding that is interesting on its own, because any equal sized (in height and width) filterbank could learn such structure, but does not as is illustrated in the circular filter visualizations of the other layers. Second, by combining a horizontal and a vertical kernel, the filters tend to broadly distribute their energy on the input receptive field. Overall, these findings qualitatively explain the quantitative performance gains for network architectures that explicitly use a separation into horizontal and vertical filters (Liu et al. 2017), as we observe structures that would not have been learned by uniform filters. This can trigger ideas for future work such as diagonal filters, or even a dictionary of learned kernel shapes. In general this observation illustrates the benefit of enforcing structure of filters, in order to force the system to learn filters of specific shape.

Lastly, we show class prediction units of the Inception-v3 architecture trained on Kinetics, for both the appearance and motion streams of a two-stream ConvNet. In Fig. 20 we show the class prediction units of these two streams for 20 sample classes. Notably, Kinetics includes many actions that are hard to predict just from optical flow information.[2] The first row shows classes that are easily classified by the appearance stream with recognition accuracies above 90% and the last row shows classes that have appearance stream recognition accuracies below 15%. The easily recognizable classes have clear visualizations of objects that unambiguously identify these, e.g., playing chess or squash are clearly visible in the appearance visualizations. On the other hand, the cases shown in the last row are not easily recognized from only appearance information.

5 Conclusion

The compositional structure of deep networks makes it difficult to reason explicitly about what these powerful systems actually have learned. In this paper, we have shed light on the learned representations of deep spatiotemporal net-

[2] https://deepmind.com/research/open-source/open-source-datasets/kinetics/.

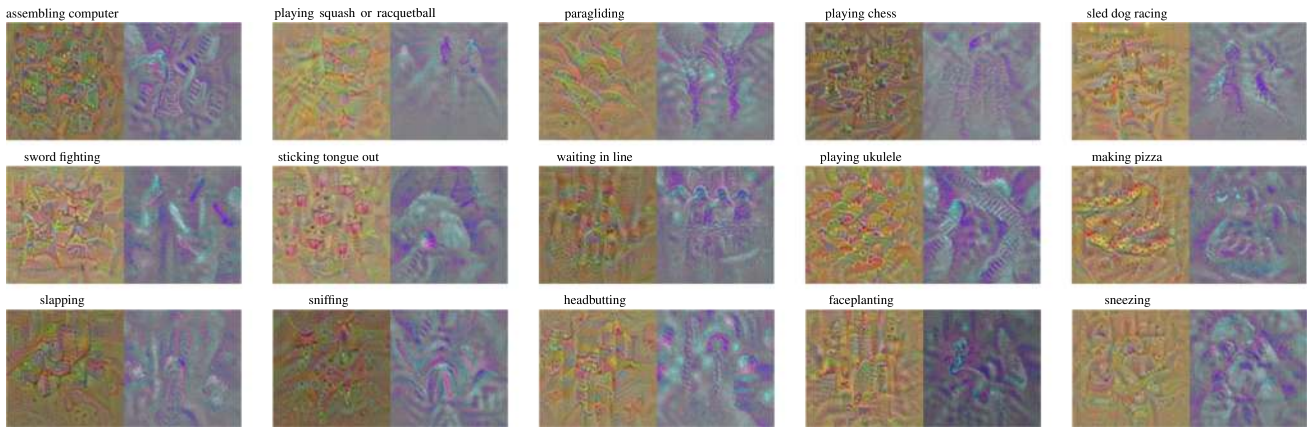

Fig. 20 Visualizations of classification units at the last layer of the Inception_v3 two-stream network (Szegedy et al. 2015) trained on Kinetics (Carreira and Zisserman 2017). We show pairs of appearance and motion input generated by maximizing the prediction layer output for the respective classes listed above. Please see the optical flow in the Animated Manuscript for video playback

works by visualizing what excites the models internally. We formulate our approach as a regularized gradient-based optimization problem that searches in the input space of a two-stream architecture by performing activation maximization. We have visualized the hierarchical features learned by deep spatiotemporal networks. Our visual explanations are intuitive and indicate the efficacy of processing appearance and motion in parallel pathways, as well as cross-stream fusion, for analysis of spatiotemporal information.

Acknowledgements Open access funding provided by Graz University of Technology. This work was partly supported by the Austrian Science Fund (FWF) under P27076, EPSRC Programme Grant Seebibyte EP/M013774/1, NSERC RGPIN 249910 and CFREF VISTA. Christoph Feichtenhofer is a recipient of a DOC Fellowship of the Austrian Academy of Sciences at the Institute of Electrical Measurement and Measurement Signal Processing, Graz University of Technology.

References

Animated Manuscript. http://feichtenhofer.github.io/pubs/Feichtenhofer_IJCV19.pdf.

Bau, D., Zhou, B., Khosla, A., Oliva, A., & Torralba, A. (2017). Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of CVPR*.

Carreira, J., & Zisserman, A. (2017). Quo vadis, action recognition? A new model and the kinetics dataset. In *Proceedings of CVPR*.

Dollar, P., Rabaud, V., Cottrell, G., & Belongie, S. (2005). Behavior recognition via sparse spatio-temporal features. In *ICCV VS-PETS*.

Dosovitskiy, A., & Brox, T. (2016). Generating images with perceptual similarity metrics based on deep networks. In *NIPS*.

Erhan, D., Bengio, Y., Courville, A., & Vincent, P. (2009). it Visualizing higher-layer features of a deep network. Technical report 1341, University of Montreal.

Feichtenhofer, C., Pinz, A., & Wildes, R. (2015). Dynamically encoded actions based on spacetime saliency. In *Proceedings of CVPR*.

Feichtenhofer, C., Pinz, A., & Wildes, R. (2016a). Spatiotemporal residual networks for video action recognition. In *NIPS*.

Feichtenhofer, C., Pinz, A., & Wildes, R. P. (2017). Spatiotemporal multiplier networks for video action recognition. In *Proceedings of CVPR*.

Feichtenhofer, C., Pinz, A., Wildes, R. P., & Zisserman, A. (2018). What have we learned from deep representations for action recognition? In *Proceedings of CVPR*.

Feichtenhofer, C., Pinz, A., & Zisserman, A. (2016b). Convolutional two-stream network fusion for video action recognition. In *Proceedings of CVPR*.

Felleman, D. J., & Van Essen, D. C. (1991). Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, *1*(1), 1–47.

Földiák, P. (1991). Learning invariance from transformation sequences. *Neural Computation*, *3*(2), 194–200.

Freeman, W., & Adelson, E. (1991). The design and use of steerable filters. *IEEE PAMI*, *13*(9), 891–906.

Galloway, A., Tanay, T., & Taylor, G. W. (2018). Adversarial training versus weight decay. arXiv preprint arXiv:1804.03308.

Goodale, M. A., & Milner, A. D. (1992). Separate visual pathways for perception and action. *Trends in Neurosciences*, *15*(1), 20–25.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In *NIPS*.

Gorea, A., & Papathomas, T. V. (1993). Double opponency as a generalized concept in texture segregation illustrated with stimuli defined by color, luminance, and orientation. *Journal of the Optical Society of America A*, *10*(7), 1450–1462.

Goroshin, R., Bruna, J., Tompson, J., Eigen, D., & LeCun, Y. (2015). Unsupervised feature learning from temporal data. In *Proceedings of ICCV*.

Gouras, P. (1974). Opponent-colour cells in different layers of foveal striate cortex. *The Journal of Physiology*, *238*(3), 583–602.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of CVPR*.

Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527–1554.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of ICML*.

Johnson, J., Alahi, A., & Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. In *Proceedings of ECCV*.

Kingma, D., & Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Kläser, A., Marszałek, M., & Schmid, C. (2008). A spatio-temporal descriptor based on 3D-gradients. In *Proceedings of BMVC*.

Kourtzi, Z., & Kanwisher, N. (2000). Activation in human MT/MST by static images with implied motion. *Journal of Cognitive Neuroscience*, *12*(1), 48–55.

Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., & Serre, T. (2011). HMDB: A large video database for human motion recognition. In *Proceedings of ICCV*.

Le, Q., Ranzato, M., Monga, R., Devin, M., Chen, K., Corrado, G., Dean, J., & Ng, A. (2012). Building high-level features using large scale unsupervised learning. In *Proceedings of ICML*.

Liu, C., Zoph, B., Shlens, J., Hua, W., Li, L. J., Fei-Fei, L., Yuille, A., Huang, J., & Murphy, K. (2017). Progressive neural architecture search. arXiv preprint arXiv:1712.00559.

Livingstone, M. S., & Hubel, D. H. (1984). Anatomy and physiology of a color system in the primate visual cortex. *Journal of Neuroscience*, *4*(1), 309–356.

Mahendran, A., & Vedaldi, A. (2016a). Salient deconvolutional networks. In *Proceedings of ECCV*.

Mahendran, A., & Vedaldi, A. (2016b). Visualizing deep convolutional neural networks using natural pre-images. *IJCV*, *120*(3), 233–255.

Mishkin, M., Ungerleider, L. G., & Macko, K. A. (1983). Object vision and spatial vision: Two cortical pathways. *Trends in Neurosciences*, *6*, 414–417.

Mordvintsev, A., Olah, C., & Tyka., M. (2015). *Inceptionism: Going deeper into neural networks*. Google Research Blog. Retrieved from June 20, https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html.

Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., & Clune, J. (2016). Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *NIPS*.

Nguyen, A., Yosinski, J., Bengio, Y., Dosovitskiy, A., & Clune, J. (2017). Plug & play generative networks: Conditional iterative generation of images in latent space. In *Proceedings of CVPR*.

Nguyen, A., Yosinski, J., & Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of CVPR*.

Reichardt, W., Poggio, T., & Hausen, K. (1983). Figure-ground discrimination by relative movement in the visual system of the fly. *Biological Cybernetics*, *46*(1), 1–30.

Saleem, K., Suzuki, W., Tanaka, K., & Hashikawa, T. (2000). Connections between anterior inferotemporal cortex and superior temporal sulcus regions in the macaque monkey. *Journal of Neuroscience*, *20*(13), 5083–5101.

Selvaraju, R. R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., & Batra, D. (2016). Grad-cam: Why did you say that? Visual explanations from deep networks via gradient-based localization. arXiv preprint arXiv:1610.02391.

Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR workshop*.

Simonyan, K., & Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. In *NIPS*.

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International conference on learning representations*.

Soomro, K., Zamir, A. R., & Shah, M. (2012). *UCF101: A dataset of 101 human actions classes from videos in the wild*. Technical report CRCV-TR-12-01.

Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2015). Striving for simplicity: The all convolutional net. In *ICLR workshop*.

Stromeyer, C., Kronauer, R., Madsen, J., & Klein, S. (1984). Opponent-movement mechanisms in human vision. *Journal of the Optical Society of America A*, *1*(8), 876–884.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of CVPR*.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2015). Rethinking the inception architecture for computer vision. arXiv preprint arXiv:1512.00567.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2014). Intriguing properties of neural networks. In *Proceedings of ICLR*.

Tran, D., Bourdev, L., Fergus, R., Torresani, L., & Paluri, M. (2015). Learning spatiotemporal features with 3D convolutional networks. In *Proceedings of ICCV*.

Wang, H., & Schmid, C. (2013). Action recognition with improved trajectories. In *Proceedings of ICCV*.

Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., & Van Gool, L. (2016). Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*.

Wiskott, L., & Sejnowski, T. J. (2002). Slow feature analysis: Unsupervised learning of invariances. *Neural computation*, *14*(4), 715–770.

Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., & Lipson, H. (2015). Understanding neural networks through deep visualization. In *ICML workshop*.

Zach, C., Pock, T., & Bischof, H. (2007). A duality based approach for realtime TV-L1 optical flow. In *Proceedings of DAGM*.

Zeiler, M. D., & Fergus, R. (2013). Visualizing and understanding convolutional networks. *CoRR* arXiv:1311.2901.

Zhang, H., Yang, J., Zhang, Y., & Huang, T. S. (2010). Non-local kernel regression for image and video restoration. In *Proceedings of ECCV*.

Zhang, J., Lin, Z., Brandt, J., Shen, X., & Sclaroff, S. (2016). Top-down neural attention by excitation backprop. In *Proceedings of ECCV*.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2014). Object detectors emerge in deep scene CNNs. In *Proceedings of ICLR*.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning deep features for discriminative localization. In *Proceedings of CVPR*.