Byvshev, Petr; Xiao, Yu; Mettes, Pascal

Are 3D convolutional networks inherently biased towards appearance?

Published in:
Computer Vision and Image Understanding

DOI:
https://doi.org/10.1016/j.cviu.2022.103437

Published: 01/07/2022

Document Version
Publisher's PDF, also known as Version of record

Published under the following license:
CC BY

Please cite the original version:
https://doi.org/10.1016/j.cviu.2022.103437
Are 3D convolutional networks inherently biased towards appearance?

Petr Byvshev a,*, Pascal Mettes b, Yu Xiao a

a Aalto University, Espoo, Finland
b University of Amsterdam, Amsterdam, Netherlands

1. Introduction

Human activity recognition from videos is a long standing research problem in computer vision.

Foundational solutions performed activity recognition based on local invariant features such as spatio-temporal keypoints trajectories, skeleton detection and tracking, or dense trajectories (Sun et al., 2010; Wang et al., 2011; Li et al., 2016; Abdul-Azim and Hemayed, 2015; Messing et al., 2009).

Based on advances in deep learning, the current state-of-the-art performs recognition through deep spatio-temporal neural networks such as two-stream networks (Simonyan and Zisserman, 2014; Feichtenhofer et al., 2016), recurrent networks (Srivastava et al., 2015; Ullah et al., 2018) and 3D convolutional networks (Tran et al., 2019; Kay et al., 2017; Miech et al., 2019; Heilbron et al., 2015; Carreira and Zisserman, 2017; Tran et al., 2018) fueled by a growing collection of large-scale video datasets (Heilbron et al., 2015; Yansong Tang, 2019; Miech et al., 2019; Kay et al., 2017). These advances form the starting point of our research, where we seek to understand why deep networks and specifically 3D convolutional networks are effective for learning from videos.

Due to the effectiveness of 3D convolutional networks trained on large-scale video data, a number of works have performed empirical analysis to get insights into the working of such networks. One common finding is the bias in 3D convolutional networks towards scenes and objects (Hiley et al., 2020; Huang et al., 2018), which suggest that activity recognition happens not based on actor-centric dynamics but by recognizing the context in which activities occur. Others suggested various methods to measure and separate motion in videos and analyzed the model’s performance with respect to observed dynamics (Sevilla-Lara et al., 2021; Bertasius et al., 2021; Yang et al., 2020). In contrast, we directly look at the motion and appearance aspects in 3D models themselves and define new metrics to analyze temporality of the models and their layers.

Specifically, we question whether 3D convolutional networks are inherently biased towards static appearances and investigate whether such networks are able to capture both appearance and dynamic aspects given the right data.

As starting point for our investigation, we confirm the clear connection between activities and their static context by considering a large-scale dataset of human actions Kinetics (Kay et al., 2017) and locations Places365 (Zhou et al., 2017). We provide observations about...
the sparse relation between activities and their locations classes — the natural imbalance between actions and locations distributions. Additionally, we show the direct effect of activity-location exclusivity on the ability of the network to recognize an action class: a model better recognizes an action that takes place in a unique and consistent location.

Second, we propose new layer-wise and kernel-wise measurements that are designed to reveal the temporal properties of 3D convolutional models. The measurements use the weights of a network to determine how dynamically receptive individual kernels and layers are when trained on a large-scale video data. We observe a consistent drop of temporality for deeper convolutional layers in all 3D models trained on Kinetics. Said findings indicate that the trained models either reserve later layers for appearance only, or they are the result of the models adapting to the (lack of) discriminative motion statistics in the data.

Third, aiming to resolve the described disjunction, we investigate our hypothesis that the bias towards appearance and away from temporal dynamics is not inherent in 3D convolutional networks. To that end, we propose a new method to generate motion trajectories and construct two datasets in which motion and appearance are explicitly decoupled. Decoupling allows us to control the relation between these aspects in the temporality experiments. We find that 3D convolutional network are not biased by their design towards appearance. On the contrary, they successfully manage to learn complex motion patterns. Upon applying the proposed temporality measures on the trained networks, we find that increasing variability of motion patterns in the training data results in a significant rise in the temporality, especially in the deeper layer of the network. Additionally, compared to motion, increasing appearance variance stronger penalized models performance. Our investigation sheds new light on deep learning for video activity recognition by showing that appearance bias in modern 3D models reflects data properties rather than architectural limitations. We demonstrate that models can adapt to both motion and appearance content of the data, effortlessly learning complex movement trajectories.

In short, our contributions are as follows:
- we confirm the clear connection between activities and their locations in 3D convolutional networks;
- we define new temporality measurements, designed to reveal the temporal properties of 3D convolutional networks;
- we present two new datasets, in which we explicitly decouple motion and appearance;
- with the new datasets and measurements, we reveal new findings on motion-appearance relation in 3D networks.

2. Background and related work

Initially, human recognition datasets focused on specific domains such as sports, cooking, household chores and outdoor activities. Therefore, background environment in such datasets generally remained consistent across different action categories (Damen et al., 2018; Soomro and Zamir, 2014; Zhou et al., 2018b; Yoshikawa et al., 2018). Deep video understanding requires a network that captures its visual aspects (i.e., actors, objects and locations) together with temporal dynamics, and perhaps, the most dominant trend in video understanding of the last decade is the successful conversion of 2D image networks to 3D and video networks. Well-known approaches include recurrent networks over frame-level features (Srivastava et al., 2015), tracking the relations of interest points (Kalita et al., 2018; Yi et al., 2020), focusing on human body poses and gestures, and localizing key frames (Ng et al., 2015; Wang et al., 2011; Laptev, 2005; Ahlara and Aoki, 2014; Ke et al., 2018; El-Ghaish et al., 2018; Agahian et al., 2020).

Recently, several large-scale action datasets, including COIN (Yan-song Tang, 2019), ActivityNet (Heilbron et al., 2015), Kinetics (Carreira and Zisserman, 2017), HowTo100M (Miech et al., 2019) and IG-video (Ghadiyaram et al., 2019), have been built from videos collected through public video sharing sites like YouTube and Instagram. With the introduction of large datasets, deep 3D networks have thrived and shown an ability to learn effective video representations. Among the popular architectures are: inflated inception 3D models – I3D (Carreira and Zisserman, 2017), factorized 3D convolutional networks – R(2+1)D (Tran et al., 2018), separable 3D CNNs — S3D (Xie et al., 2017), and channel-separated CNNs — CSN (Tran et al., 2019).

These networks provide useful features for many downstream video understanding tasks, including video segmentation, activity localization, video retrieval and classification of long videos (Lai and Xie, 2019; Miech et al., 2020; Farha and Gall, 2019; Chao et al., 2018; Hussein et al., 2019b,a; Mithun et al., 2018; Chen et al., 2020). We choose I3D as our main experimental model as it represents the mainstream 3D architecture for video analytics.

With the quantitative success of deep 3D networks came the necessity for model interpretation and explainability. Visualization and analysis of deep networks helps to understand the nature of learned features, their hierarchy, level of abstraction and generalization. Due to the hybrid nature of 3D spatiotemporal features their study and interpretation is not straightforward and many approaches have been proposed to address that property. Gradient-based methods provide pixel-wise visual explanations, highlighting regions of high importance for the classification task, for example, Guided Backpropagation (Springenberg et al., 2015), GradCAM (Selvaraju et al., 2017). Similarly, visualizations of video-based solutions compute spatio-temporal heatmaps for input voxels. T. Nagarajan (2019) focused on visualizing human-object interactions, Meng et al. (2019) proposed interpretable attention mechanism that distinguishes the most relevant parts of the video, Class Feature Pyramids method discovers 3D kernels that are informative for a specific class (Stergiou et al., 2019). Such methods reveal local implied spatio-temporal properties of a given video that are relevant for the classification task. Shuffling and removing frames is one of the techniques to test how deep models capture motion information: Zhou et al. (2018a) discover static and temporal action categories depending on the frame order, Xie et al. (2017) report performance changes when reversing the order of frames and show that importance of ‘arrow of time’ varies in different datasets. Huang et al. (2018) propose to measure the amount of temporal information by isolating the appearance only in a video and training a model to generate temporal dynamics in a class-agnostic fashion. Sevilla-Lara et al. (2021) split videos into static and temporal categories using human annotations and observe statistical difference of the action categories. Choi et al. (2019) observe the scene bias problem and propose new training methods to mitigate it. Weinzaepfel and Rogez (2019) analyze popular 3D models and observe a superior performance of shallow human-centered models in contextless activity recognition.

Where current works focus on the temporality of action categories, we look at 3D networks themselves and focus on the key building blocks — 3D kernels. We find that 3D networks copy biases from biased dataset, but can adapt to varying amount of motion in the data.

3. I3D is biased towards appearance datasets

We first try to answer the following video understanding question: how is the temporality of videos encoded in the network’s parameters? Throughout our experiments, we use the I3D model pre-trained on the Kinetics dataset (Kay et al., 2017) as the main backbone, due to its canonical nature and widespread use in video recognition. All models are trained on four Tesla V100-PCIE-16gb GPUs, using the Pytorch framework (Paszke et al., 2019). Unless specified otherwise, we follow the data preprocessing and training from (Carreira and Zisserman, 2017).

As a starting point, we have quantified appearance bias in 3D models in Appendix A, before we investigate whether this bias is inherent. Choi et al. (2019), Girdhar and Ramanan (2019) have observed scene and object bias in action datasets: tasks and static surroundings are highly correlated. The results in the appendix show three things: (i) more than half of the Kinetics samples take place
in 16 location categories, (ii) I3D performs worse on video samples that take place in more common locations compared to more unique locations, and (iii) representations trained on actions also learn relevant scene representations. A network trained on Kinetics can be directly used to obtain effective representations for scene recognition. In fact, fine-tuning the entire network on training scenes does not improve results.

These experiments support that Kinetics dataset extensively covers scenes and enables learning useful features for scene recognition. Our next step is to find out which layers of 3D convolutional networks are responsible for temporal modeling rather than appearance modeling. We perform an examination at the layer level and the individual filter level and propose new measures to quantify the temporality of each convolutional kernel. We use these measures to analyze how much each layer is responsible for modeling dynamic representations of videos.

### 3.1. New measures for temporal dynamics in 3D filters

3D convolutions jointly learn visual and temporal representations, which raises the following question: how can we find correlates of temporal modeling in a 3D network and its parameters? We propose two novel metrics that aim to show the temporality of a convolutional layer and its kernels.

3D filters responsible for recognizing temporal changes should have weights receptive to those changes. Under this assumption, we define the first measure of temporality of a layer as an average standard deviation across the temporal dimension (ASDT), (see Fig. 1). A 3D convolution filter is defined by a 5-dimensional tensor \((H, W, T, M, N)\) denoting respectively height, width, temporal span, the number of channels in the previous layer, and the number of channels in the next layer. As an example, the second 3D convolutional layer of the I3D model is of size \((3, 3, 3, 64, 192)\). If \(w_{i,j,t,m,n}\) is the weights tensor and \(i, j, t, m, n\) are the indices that cover aforementioned dimensions, ASDT is defined as:

\[
\text{ASDT} = \sum_{i,j,m,n} \sqrt{\frac{\sum_{t=1}^{T} (w_{i,j,t,m,n} - \bar{w}_{i,j,m,n})^2}{T}},
\]

where \(w_{i,j,m,n}\) is the mean over the time dimension

ASDT can be applied to both fully convolutional networks (such as I3D) and spatiotemporal-separable 3D convolution networks as it only measures variance of weights across the temporal dimension. For example, we can similarly compute ASDT for the S3D model (Xie et al., 2017) by taking the weights from its 1D temporal convolutional layers.

For more details on calculation of ASDT for different architectures, please, refer to Appendix B.

ASDT defines the temporality of 3D convolutional networks for individual layers. To look at the feature maps of the model, we need to consider individual kernel of the convolution operations. For that purpose we propose a second measurement that aims to estimate which feature maps are more receptive to dynamics in the video and which are more responsible for appearance. In other words, to determine which channels of a layer are temporal and non-temporal. Each output channel represents a feature map that reflects a certain visual or dynamical property of the input, it is constructed as a sum of filtered feature maps of the previous layer. We aim to define an individual temporality measure for every 3D filter that together represent the temporality of the feature map. We notice that computing central self-convolution (\(sc()\)) for the filter (or strictly speaking, the sum of Hadamard products Horn, 1990) satisfies such requirement, see Fig. 2.

Indeed a fully non-temporal filter would have a number of identical 2D kernels stacked together and would have a high value of self-convolution. On the opposite, filters with high temporal discrepancy would have low (even negative) self-convolution values. If \(w_{i,j,t}\) is a single kernel of a layer, the self-convolution (sc) is defined as:

\[
sc(u) = \sum_{i,j,t} (w_{i,j,t} \odot w_{i,j,t}^2)
\]
The ASDT measure monotonically decreases with layer depth, which shows the limited response to video dynamics in the latter layers.

Now for every 3D layer, we classify each channel as temporal if more than half of its kernels are temporal and static otherwise as defined by the self-convolution measurement (we use $\delta = 0$ for our measurement). In Fig. 4, we plot the distribution of the output channels and observe similar dynamics, i.e. the number of temporal channels drops towards the outer segments of the model. This finding clarifies the ASDT measurement of I3D (the average drop of temporality cannot be explained by lateral growth of the layers) and both measurements suggest that motion related features are more likely to be learned in the earlier layers of I3D, when trained on the Kinetics dataset. To summarize, the location experiments showed how the I3D model relies on the static information when classifying Kinetics actions. The ASTD and self-convolution metrics show that the average temporality of the I3D layers decreases with depth and the absolute number of temporal feature maps also drops with layer depth. In short: I3D is biased towards scenes and static appearance in general when trained on Kinetics.

4. I3D is not inherently biased towards appearance

The results of the previous sections show that the I3D model trained on Kinetics is biased towards appearance and that temporal dynamics is only marginally relevant, especially in the final 3D layers. In the following section, we show that this bias is not inherent to I3D model itself, but rather reflects the properties of the training data. For this purpose we need to decouple motion and appearance in the videos — to directly access the variability of movement and background information and its effect on the resulting model. To deepen the analysis of temporality in 3D models we introduce two new datasets with controlled differentiation of motion and appearance.

4.1. Decoupled motion-appearance datasets

BrushMNIST is a synthetic dataset we composed to study the ability of 3D convolutional networks to learn motion and appearance. Variable appearances are achieved by using different instances of the MNIST dataset that move in the video clip, and variable motion patterns are introduced as different drawing trajectories of the MNIST digits. Simply speaking we use one digit as a brush and another digit as a trajectory example - shape. This corresponds to 100 different classes corresponding to 10 brushes and 10 shapes. Each sample consists of black and white video clips with $224 \times 224$ pixels resolution and 150 frames each. In every video clip a random MNIST digit ($28 \times 28$ pixels) is moving in a trajectory that corresponds to another random MNIST digit that is resized to cover the $224 \times 224$ frame (See Fig. 5 for examples). The generation of random trajectories from a digit sample is described in detail in Appendix D. Since MNIST consists of 60,000 training samples, BrushMNIST dataset theoretically consists of 60,000 $\times$ 60,000 training samples. Our dataset differs from Moving MNIST (Srivastava et al., 2015), which consists of short video clips of groups of digits moving in straight lines following geometrical trajectories and bouncing off the frame edges (no labels). Moving MNIST is used for predicting future trajectories by extrapolating learned representations. The BrushMNIST is designed for a video sequence classification task, where trajectories' shapes define the video categories.

Places+Faces+MNIST (P+F+M) is our second synthetic dataset that is composed of Places365, Labeled Faces in the Wild (LFW) and MNIST datasets (Huang et al., 2007; Zhou et al., 2017; Deng, 2012).
Fig. 5. Examples from BrushMNIST. Depicted are 4 examples (1 example per row), 10 out of 150 frames for each sample. Red color highlights the footprint of the drawn shape. In the actual samples only the white numbers (i.e. brushes) are present in each frame and consecutively draw over the footprint. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 6. Examples from Places+Faces+MNIST. Depicted are 3 frames from 2 samples, the class of the sample depends of the location from Places365 and face trajectory. Yellow dotted-line represents the movement trajectory of the face and is not present in the actual video sample. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

It is similar to BrushMNIST, but uses Places365 images as background and represents the appearance variance, LFW faces are used as brushes and MNIST samples as trajectories (Fig. 6 for examples). We define different instances of the dataset P+F+M\_x\_y, where \( x \) is the number of places categories and \( y \) is the number of trajectories, and the total number of classes is \( x \times y \). Trajectories beyond 10 are constructed by drawing number’s digits: one after another. For the better viewing the datasets and the samples GIFs are available at [https://github.com/Petr-Byv/Brush_MNIST](https://github.com/Petr-Byv/Brush_MNIST).

The benefit of using the datasets is two-fold: we know exactly how motion and appearance are tied for each sample, and we can also control the complexity by balancing dynamics and appearances.

4.2. Analysis of temporal dynamics on decoupled datasets

In the following section, we study the behavior of the I3D model on the previously defined decoupled datasets — BrushMNIST and Places+Faces+MNIST. By utilizing the temporality measures we show how varying the amount of dynamics in the training data effects the model parameters.

4.2.1. BrushMNIST experiments

For the experiments on BrushMNIST we train an I3D model with randomly initialized weights and changing the number of input channels from 3 to 1 to feed black-and-white videos.

First, we conduct 3 validation experiments:

- Basic setup - Table 1(a). It shows motion and appearance related errors: motion is learned successfully and the majority of errors comes from the appearance miss-classification. This indicates that motion patterns of writing digits add information which helps to a better recognize BrushMNIST shapes.
Table 1
BrushMNIST experiments for the I3D model. (a) Basic setup experiment: appearance related errors dominate the misclassification. (b) Footprint experiment: removing the temporal information by feeding only the footprint of motion balances the error distribution. (c) Frames order experiment: reversing the order of frames in test samples of BrushMNIST reduces the recognition of motion to the chance level.

<table>
<thead>
<tr>
<th>BrushMNIST errors across motion/appearance (a)</th>
<th>Footprint experiment error distribution (b)</th>
<th>Frames order experiment (c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appearance</td>
<td>Brush</td>
<td>Shape</td>
</tr>
<tr>
<td>2.49%</td>
<td>0.11%</td>
<td>0.03%</td>
</tr>
</tbody>
</table>

Fig. 8. ASDT for I3D trained on BrushMNIST. Compared to Kinetics ASDT we do not observe decrease in temporality with layer depth — deeper layers also learned dynamics of the BrushMNIST samples.

Fig. 9. Self-convolution metrics for I3D trained on the P+F+M. The model was trained on 16 motions and 8 locations. We observe more temporal channels compared to Kinetics training, especially in the later layers.

• Footprint test - Table 1(b). To clarify the importance of motion information we removed the temporal information and fed only the footprint of a motion. This modification of BrushMNIST worsened the performance: errors are now equally distributed across Brush and Shape digits.

• Frame order - Table 1(c). It shows that opposite to the Kinetics set, when feeding reversed samples of the BrushMNIST the recognition of motion is dropped to the chance level — 11.3% (since the Brush digits are still recognized correctly).

Fig. 7 shows how the model learns temporal dynamics. Each row shows footprint of the samples with the same shape, but different brushes. The red highlights the frames (and therefore individual small numbers) that correspond to the maximum activation in the feature space along the time dimension. This shows the consistency of learned motion patterns across the shapes: the model learns to attend the
signature segments of the number shapes, for example, drawing a line from left to right in the middle of a ‘5’ or the intersection of the lines of a ‘4’.

Fig. 8 shows ASDT measure for I3D trained on BrushMNIST. For this dataset temporality of layers does not continuously drop with depth, which proves that directly infusing dynamics into the training samples changes the temporal responsibility of models layers. The appearance bias observed in previous experiments is caused by the training data itself and not the model architecture.

4.2.2. Places+Faces+MNIST experiments

We can tune the motion and appearance aspects of a P+F+M dataset by changing the number of background classes and the number of motion patterns. The total combinatorial number of classes would be the multiplication of these to numbers. For the constancy we experiment with all the combinations of $2^1, 2^2, 2^3$ numbers of motion and appearance classes. This way the simplest dataset has 4 classes ($2^1 \times 2^1$) and the most complex $2^4 \times 2^8 = 65536$ classes. This allows us to answer the question: does I3D better scale up to motions or appearances variability?

Table 2 shows the behavior of the I3D depending on different motion/appearance factors of a P+F+M set. We plot the model’s accuracy depending on the number of motion and appearance classes. The table shows that the model can deal with the growing number of motion patterns, and vice-versa performance deteriorates faster when adding more appearance classes. This finding suggests that I3D model is not intrinsically biased towards appearance, but rather flexible — dynamical-motion information is learned to the extend presented by the training dataset.

We also analyze the temporality of I3D when trained on P+F+M compared to training on Kinetics with help of measures proposed in Section 3.1. Fig. 10 shows an example ASDT measure for different layers after training on P+F+M with 16 motions and 8 locations. Compared to I3D trained on Kinetics, we observe a very different picture: the temporality of a layer does not decrease with depth, but the opposite trend is clear. Similarly, compared to I3D trained on Kinetics the self-convolution measure shows a drastic difference: more temporal channels are observed in all the layers, with the deeper ones effected the most compared to the Kinetics weights (Fig. 9). This result proves the utility of previously proposed measurements as they reveal how the nature of training data affects the temporality of model’s parameters.

5. Does pre-training on millions of videos resolve the appearance bias in 3D convolutional networks?

So far, we have found that the ASDT profile of a model reflects the bias towards or away from temporality as a function of the layer depth in 3D convolutional networks. In the Appendix we also analyze state-of-the-art models pre-trained on Kinetics such as R(2+1)D, S3D, ip-CSN-152 and I3D — all show a similar behavior, their ASDT temporality drops with layer depth, as shown in Fig. B.13. We also analyzed the HMDB51 (Human Motion Database) — a dataset with a direct focus on the motion aspect. Nonetheless motion specialty does not resolve the appearance bias, as seen in Fig. 11(a) the temporality drop in the later layers persists. While Kinetics remains the main video set for pre-training, recently a few datasets have been introduced with millions of video clips, including Howto100 m (Miech et al., 2019) and IG-65M (Tran et al., 2019). They have shown competitive results when used for pre-training of deep 3D models. To see whether training on millions of videos alleviates the bias towards static appearance, we analyze the ASDT profile for 3D models trained on these datasets, i.e. the temporality of said datasets in Fig. 11. Interestingly, it shows that natural video collections can induce a different temporality profile in 3D models, for example, Howto100 m leads to a flat temporality in all the layers of the S3D model (Miech et al., 2020). This confirms authors’ assumption, that Howto100M dataset holds more dynamical information which makes it stand out from both Kinetics and IG-65M.

We conclude that ASDT measure helps to estimate the temporality of future video datasets and progress towards more dynamical video representations. Based on the ASDT measures, we recommend to pre-train 3D convolutional networks on Howto100 m if an appearance bias needs to be avoided.

6. Conclusions

In the paper we study motion and appearance aspects in 3D convolutional models for video analysis. We propose measurements to estimate the temporality of network’s weights, and these measures reveal a common pattern for different models trained on Kinetics - a clear decrease of temporality as a function of the layer depth. More to that, our experiments point to the training data, and not 3D architecture itself, for being responsible for previously observed appearance/scene bias in video classification solutions. With the proposal of decoupled motion-appearance datasets we observe that 3D convolutions adapt to videos visual statistics and are powerful enough to effectively recognize complex motion patterns. By modifying the amount of motion in the training data we managed to show the validity of proposed temporality measure. We believe that the presented results and proposed measures give useful insights for computer vision researchers, which can help in construction of future video datasets and engineering of deep video models. We will make the code and measures publicly available.

Table 2

<table>
<thead>
<tr>
<th>A</th>
<th>M</th>
<th>T</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.99</td>
<td>0.98</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>0.98</td>
<td>0.98</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.96</td>
<td>0.98</td>
<td>0.97</td>
<td>0.97</td>
<td>0.97</td>
<td>0.97</td>
<td>0.97</td>
<td>0.96</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.89</td>
<td>0.89</td>
<td>0.90</td>
<td>0.90</td>
<td>0.89</td>
<td>0.89</td>
<td>0.86</td>
<td>0.86</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>0.86</td>
<td>0.86</td>
<td>0.83</td>
<td>0.83</td>
<td>0.82</td>
<td>0.81</td>
<td>0.81</td>
<td>0.80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>0.78</td>
<td>0.78</td>
<td>0.77</td>
<td>0.77</td>
<td>0.76</td>
<td>0.76</td>
<td>0.75</td>
<td>0.75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>0.66</td>
<td>0.66</td>
<td>0.66</td>
<td>0.65</td>
<td>0.65</td>
<td>0.65</td>
<td>0.64</td>
<td>0.64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>0.52</td>
<td>0.51</td>
<td>0.50</td>
<td>0.50</td>
<td>0.49</td>
<td>0.49</td>
<td>0.49</td>
<td>0.48</td>
<td></td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>0.36</td>
<td>0.35</td>
<td>0.35</td>
<td>0.34</td>
<td>0.33</td>
<td>0.33</td>
<td>0.32</td>
<td>0.32</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 10. ASDT measure for I3D trained on the P+F+M. The model was trained on 16 motions and 8 locations. Contrary to training on Kinetics, we do not see the decrease of temporality with layer’s depth.

1 https://github.com/facebookresearch/vmz
Fig. 11. ASDT for state-of-the-art video models. Where training on HMDB51 or IG-65M still induces an appearance bias in later layers, this bias is mostly gone when training on Howto100m. (See Xie et al., 2017; Kuehne et al., 2011; Miech et al., 2020; Tran et al., 2019).

CRediT authorship contribution statement

Petr Byvshev: Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing – original draft, Writing – review & editing. Pascal Mettes: Conceptualization, Writing – review & editing, Supervision. Yu Xiao: Conceptualization, Writing – review & editing, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Scene bias

We quantified the bias towards scenes in I3D using a source and a target datasets:

Kinetics-400 (source) is a collection of approximately 300,000 video urls from a video hosting service that cover general human action classes. We use the first version of the dataset with 400 classes (Carreira and Zisserman, 2017).

Places365 (target) is a collection of 18 million images with 365 scene categories (Zhou et al., 2017).

Both Kinetics and Places365 are built upon extensive language corpora and aim to cover the visual world fully and precisely. For this reason we expect the scene categories from the Places365 dataset to adequately describe the locations of human activities from Kinetics and to test that we study the relations between actions from the Kinetics dataset and environments from the Places365 dataset.

First, we view the distribution of Places instances in the activities’ samples. Each sample of the video set is a sequence of frames that belong to a certain activity category from Kinetics. We extract the place information of every frame from that sequence, and to do so, we use a VGG model trained on Places365 (Zhou et al., 2017). This way each video sample produces a list of location predictions. Among all the frames we count the most common location prediction and assign it to the video sample. We call the proportion of frames that belong to the same location the common place score – it can be computed for every video sample from the Kinetics dataset. Fig. A.12 shows the percentile graph of the most common location scores for Kinetics video samples. For example, for 80% of the videos samples the common place score is 0.25 or greater, indicating that 25% or more frames of each video belong to the same place category. We assume that location categories for the video clips are reasonably consistent, meaning that from the Places365 perceptive the environment of a single Kinetics sample does not usually change. Table A.3 shows the distribution of places in all the samples from Kinetics. For the calculations we select the most dominant location category for every video sample. We see that, more than 50% of samples from Kinetics were captured in 16 categories of locations. Among the most popular are house, discotheque, outdoor cabin, library, barn, junkyard, garden and general store.

The I3D model performs better on samples with more exclusive place-activity relation — 56.3% classification accuracy for most common places and 63.2% for less common, which means that the model can more effectively utilize the surroundings to recognize activity category and videos that are related to the same environment are more likely to be confused.

The short study above indicates a relation between activities and scenes, further, we tested if representations learned on actions transfer well to scenes. We trained a classification layer on top of fixed I3D features to classify categories from the Places dataset and compare the performance to the I3D’s 2D backbone – the Inception model. The video model takes a sequence of frames as an input, therefore to get the prediction we repeat the frames — every input is a fully static video that represents a single place. We test two training regimes: fixed weights – where all the parameters up to the last convolutional layer are non-trainable and non-fixed weights – where we train all
the model’s weights. The results in Table A.4 show that I3D model can perform on par with 2D models pretrained on ImageNet (41.3% vs 44.3%). More interestingly, there is only marginal improvement when fine-tuning the entire network compared to freezing the backbone. In other words, the parameters are already adapted to scenes and do not require further tuning.

Appendix B. ASDT measure

B.1. ASDT calculation

In Section 3.1 we define the ASDT that measures temporality of the temporal layers. It takes deviation across the temporal component, and for more recent architectures such as ir-CSN, or S3D, where spatial and temporal convolutional components are separated we consider only the temporal subunits. Algorithm 1 describes the model agnostic ASDT calculation procedure.

B.2. ASDT for different 3D architectures

We plot the ASDT measure with respect to layer number (considering only layers with temporal component) for I3D, R3D, ir-CSN152 and S3D models in Fig. B.13. The pattern is similar across different models, temporality tends to drop towards later layers. Interestingly when taking an S3D trained on the Howto100m dataset (Miech et al., 2020) the pattern changes (see Fig. 11(b)) towards more temporal deeper layers. We also calculate and show ASDT for ir-CSN-152 trained on IG-65m and Sports-1m datasets2 (Figs. 11(c), B.14).

Algorithm 1 ASDT

| 1: for layer in model’s layers, ... do |
| 2: if layer belongs temporal, ... then |
| 3: if layer belongs spatial, ... then |
| 4: for kernel in layer, ... do |
| 5: Compute standard deviation for kernel along temporal dimension. |
| 6: Compute average along spatial dimension. |
| 7: end for |
| 8: else |
| 9: if layer does not belong spatial, ... then |
| 10: Compute standard deviation for kernel along temporal dimension. |
| 11: end if |
| 12: end if |
| 13: Compute average for all kernels in the layer |
| 14: end if |
| 15: end for |

Algorithm 2 Self-convolution

| 1: for layer in model’s layers, ... do |
| 2: if layer ∈ 3D convolution, ... then |
| 3: for feature space in layer, ... do |
| 4: for kernel in feature space, ... do |
| 5: Compute and add pair-wise Hadamard’s products for kernel’s 2D slices |
| 6: if Sum of self-convolutions > δ ... then |
| 7: kernel → TEMPORAL |
| 8: end if |
| 9: if Number of kernels ∈ temporal > 1/2|kernels ∈ feature space| ... then |
| 10: feature space → TEMPORAL |
| 11: else |
| 12: feature space → STATIC |
| 13: end if |
| 14: end for |
| 15: end for |
| 16: end if |
| 17: end for |

Table A.4

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Pre-training</td>
<td>Fine-tuning</td>
<td>Fix weights</td>
</tr>
<tr>
<td>Inc.v2</td>
<td>Imagenet</td>
<td>Places365</td>
<td>X</td>
</tr>
<tr>
<td>Inc.v2</td>
<td>Imagenet</td>
<td>Places365</td>
<td>✓</td>
</tr>
<tr>
<td>I3D</td>
<td>Kinetics</td>
<td>Places365</td>
<td>X</td>
</tr>
<tr>
<td>I3D</td>
<td>Kinetics</td>
<td>Places365</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table A.3

| Performance of I3D on activities from popular places vs rare places. This result shows the bias of the kinetics dataset towards the common scenes and its inverse relation to activity performance: activity recognition is higher in rare scenes. |
| Places356 | 16 common places | 233 rare places |
| Kinetics400 | 15132 videos | 16434 videos |
| Activity accuracy | 56.3% | 63.2% |

Appendix C. Self-convolution measure

In Section 3.1 we defined the self-convolution measure for fully 3D convolutional layers. It uses Hadamard product to estimate the agreement between kernel’s 2D faces. Algorithm 2 describes the self-convolution calculation procedure that classifies every feature space of a layer as TEMPORAL or STATIC:

Fig. C.15 shows self-convolution profiles for models trained on different PFM instances.

Appendix D. Generating trajectories from MNIST digits

Fig. A.12. Common place frames score — percentile graph for Kinetics dataset. Each point represents the proportion of Kinetics samples that have the corresponding common place frame score or greater. This highlights the consistency of scenes in Kinetics activities samples.

2 https://github.com/facebookresearch/vmz
In Section 4.1 we introduce datasets with decoupled motion-appearance. The brush samples from BrushMNIST were directly sampled from MNIST training set. To generate random classifiable motion patterns — the shapes, we also used MNIST digits as movement templates. For a given random digit we define a function that maps a $28 \times 28$ pixels image to a list of coordinates $(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_n, y_n)$, where $n$ is the number of frames in the desired video and $(x_i, y_i)$ is the position of the brush in the $i$th frame. For every digit class $k$ we defined a principal direction of drawing: in practice, it describes the expected direction of drawing in every frame and is represented by an ordered list of $2d$ vectors $(\vec{p}_1^k, \vec{p}_2^k, \ldots, \vec{p}_n^k)$. We also use an anchor starting point for every digit from 0 to 9 - $(x_0^k, y_0^k)$. We used OpenCV (Bradski, 2000) function skeletonize to generate pixel-wide MNIST shapes. Algorithm 3 describes the procedure for generating the list of pixel positions:
Algorithm 3 BrushMNIST trajectory

\begin{algorithmic}
\For {I from MNIST\textsubscript{train} \ldots} 
  \State $I_s$ = skeletonize($I$)
  \State $I_s = I_s$ \text{} \text{list of non-zero pixel positions}
  \State $(x_1, y_1) = \text{argmin}(dist(I_s, (x_0, y_0)))$ \text{} \text{find pixel closest to anchor point}
  \State $I_s = I_s \setminus \{ (x_1, y_1) \}$
  \For {$i = 2, \ldots, n \ldots$}
    \State $(x_i, y_i) = \text{proj}_{p_i} (\text{argmin}(dist(I_s, (x_{i-1}, y_{i-1}))))$ \text{} \text{find pixel closest to}
    \State $(x_i, y_i)$ \text{} \text{along } $p_i$
    \State $I_s = I_s \setminus \{ (x_i, y_i) \}$
  \EndFor
\EndFor
\end{algorithmic}

Fig. C.15. Self-convolution profiles for I3D trained on various PFM instances. PFM\textsubscript{*}, \text{*}. 1st number - number of location categories, 2nd-number of trajectories.

References


