# Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset

João Carreira[†] joaoluis@google.com

Andrew Zisserman[†,*] zisserman@google.com

[†]DeepMind [*]Department of Engineering Science, University of Oxford

## Abstract

*The paucity of videos in current action classification datasets (UCF-101 and HMDB-51) has made it difficult to identify good video architectures, as most methods obtain similar performance on existing small-scale benchmarks. This paper re-evaluates state-of-the-art architectures in light of the new Kinetics Human Action Video dataset. Kinetics has two orders of magnitude more data, with 400 human action classes and over 400 clips per class, and is collected from realistic, challenging YouTube videos. We provide an analysis on how current architectures fare on the task of action classification on this dataset and how much performance improves on the smaller benchmark datasets after pre-training on Kinetics.*

*We also introduce a new Two-Stream Inflated 3D ConvNet (I3D) that is based on 2D ConvNet inflation: filters and pooling kernels of very deep image classification ConvNets are expanded into 3D, making it possible to learn seamless spatio-temporal feature extractors from video while leveraging successful ImageNet architecture designs and even their parameters. We show that, after pre-training on Kinetics, I3D models considerably improve upon the state-of-the-art in action classification, reaching 80.2% on HMDB-51 and 97.9% on UCF-101.*

Figure 1. **A still from 'Quo Vadis' (1951).** Where is this going? Are these actors about to kiss each other, or have they just done so? More importantly, where is action recognition going? Actions can be ambiguous in individual frames, but the limitations of existing action recognition datasets has meant that the best-performing video architectures do not depart significantly from single-image analysis, where they rely on powerful image classifiers trained on ImageNet. In this paper we demonstrate that video models are best pre-trained on videos and report significant improvements by using spatio-temporal classifiers pre-trained on Kinetics, a freshly collected, large, challenging human action video dataset.

## 1. Introduction

One of the unexpected benefits of the ImageNet challenge has been the discovery that deep architectures trained on the 1000 images of 1000 categories, can be used for other tasks and in other domains. One of the early examples of this was using the fc7 features from a network trained on ImageNet for the PASCAL VOC classification and detection challenge [10]. Furthermore, improvements in the deep architecture, changing from AlexNet to VGG-16, immediately fed through to commensurate improvements in the PASCAL VOC performance [23]. Since then, there have been numerous examples of ImageNet trained architectures warm starting or sufficing entirely for other tasks, e.g. segmentation, depth prediction, pose estimation, action classification.

In the video domain, it is an open question whether training an action classification network on a sufficiently large dataset, will give a similar boost in performance when applied to a different temporal task or dataset. The challenges of building video datasets has meant that most popular benchmarks for action recognition are small, having on the order of 10k videos.

In this paper we aim to provide an answer to this question using the new Kinetics Human Action Video Dataset [16], which is two orders of magnitude larger than previous datasets, HMDB-51 [18] and UCF-101 [27]. Kinetics has

400 human action classes with more than 400 examples for each class, each from a unique YouTube video.

Our experimental strategy is to reimplement a number of representative neural network architectures from the literature, and then analyze their transfer behavior by first pre-training each one on Kinetics and then fine-tuning each on HMDB-51 and UCF-101. The results suggest that there is always a boost in performance by pre-training, but the extent of the boost varies significantly with the type of architecture. Based on these findings, we introduce a new model that has the capacity to take advantage of pre-training on Kinetics, and can achieves a high performance. The model termed a "Two-Stream Inflated 3D ConvNets" (I3D), builds upon state-of-the-art image classification architectures, but *inflates* their filters and pooling kernels (and optionally their parameters) into 3D, leading to very deep, naturally spatio-temporal classifiers. An I3D model based on Inception-v1 [13] obtains performance far exceeding the state-of-the-art, after pre-training on Kinetics.

In our model comparisons, we did not consider more classic approaches such as bag-of-visual-words representations [6, 19, 21, 30]. However, the Kinetics dataset is publicly available, so others can use it for such comparisons.

The next section outlines the set of implemented action classification models. Section 3 gives an overview of the Kinetics dataset. Section 4 reports the performance of models on previous benchmarks and on Kinetics, and section 5 studies how well the features learned on Kinetics transfer to different datasets. The paper concludes with a discussion of the results.

## 2. Action Classification Architectures

While the development of image representation architectures has matured quickly in recent years, there is still no clear front running architecture for video. Some of the major differences in current video architectures are whether the convolutional and layers operators use 2D (image-based) or 3D (video-based) kernels; whether the input to the network is just an RGB video or it also includes pre-computed optical flow; and, in the case of 2D ConvNets, how information is propagated across frames, which can be done either using temporally-recurrent layers such as LSTMs, or feature aggregation over time.

In this paper we compare and study a subset of models that span most of this space. Among 2D ConvNet methods, we consider ConvNets with LSTMs on top [5, 34], and two-stream networks with two different types of stream fusion [8, 25]. We also consider a 3D ConvNet [14, 28]: C3D [29].

As the main technical contribution, we introduce Two-Stream Inflated 3D ConvNets (I3D). Due to the high-dimensionality of their parameterization and the lack of labeled video data, previous 3D ConvNets have been relatively shallow (up to 8 layers). Here we make the observation that very deep image classification networks, such as Inception [13], VGG-16 [26] and ResNet [12], can be trivially inflated into spatio-temporal feature extractors, and that *their pre-trained weights provide a valuable initialization*. We also find that a two-stream configuration is still useful.

A graphical overview of the five types of architectures we evaluate is shown in figure 2 and the specification of their temporal interfaces is given in table 1.

Many of these models (all but C3D) have an Imagenet pre-trained model as a subcomponent. Our experimental strategy assumes a common ImageNet pre-trained image classification architecture as back bone, and for this we chose Inception-v1 with batch normalization [13], and morph it in different ways. The expectation is that with this back bone in common, we will be able to tease apart those changes that benefit action classification the most.

### 2.1. The Old I: ConvNet+LSTM

The high performance of image classification networks makes it appealing to try to reuse them with as minimal change as possible for video. This can be achieved by using them to extract features independently from each frame then pooling their predictions across the whole video [15]. This is in the spirit of bag of words image modeling approaches [19, 21, 30]; but while convenient in practice, it has the issue of entirely ignoring temporal structure (e.g. models can't potentially distinguish opening from closing a door).

In theory, a more satisfying approach is to add a recurrent layer to the model [5, 34], such as an LSTM, which can encode state, and capture temporal ordering and long range dependencies. We position an LSTM layer with batch normalization (as proposed by Cooijmans *et al.* [4]) after the last average pooling layer of Inception-V1, with 512 hidden units. A fully connected layer is added on top for the classifier.

The model is trained using cross-entropy losses on the outputs at all time steps. During testing we consider only the output on the last frame. Input video frames are subsampled by keeping one out of every 5, from an original 25 frames-per-second stream. The full temporal footprint of all models is given in table 1.

### 2.2. The Old II: 3D ConvNets

3D ConvNets seem like a natural approach to video modeling, and are just like standard convolutional networks, but with spatio-temporal filters. They have been explored several times, previously [14, 28, 29]. They have a very important characteristic: they directly create hierarchical representations of spatio-temporal data. One issue with these models is that they have many more parameters than 2D ConvNets because of the additional kernel dimension, and

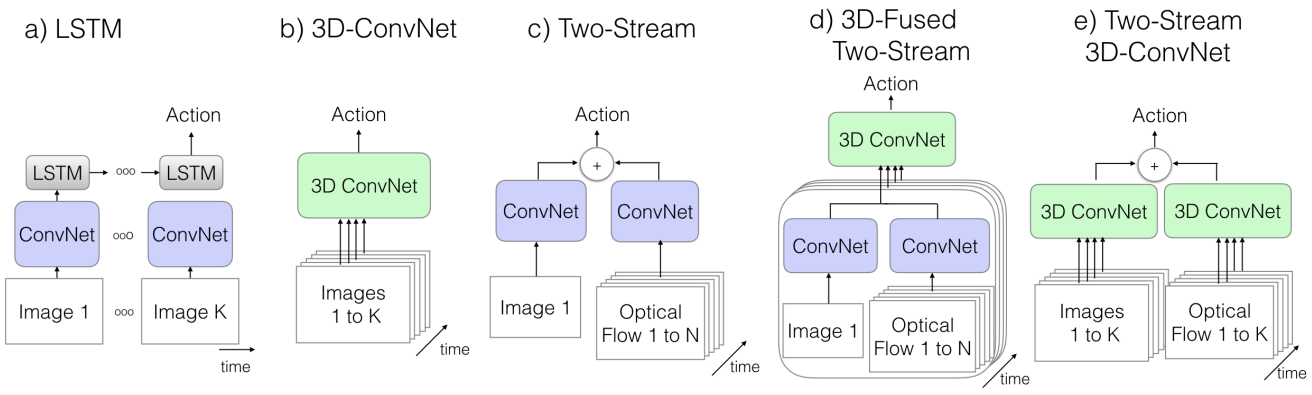


Figure 2. Video architectures considered in this paper. K stands for the total number of frames in a video, whereas N stands for a subset of neighboring frames of the video.

this makes them harder to train. Also, they seem to preclude the benefits of ImageNet pre-training, and consequently previous work has defined relatively shallow custom architectures and trained them from scratch [14, 15, 28, 29]. Results on benchmarks have shown promise but have not been competitive with state-of-the-art, making this type of models a good candidate for evaluation on our larger dataset.

For this paper we implemented a small variation of C3D [29], which has 8 convolutional layers, 5 pooling layers and 2 fully connected layers at the top. The inputs to the model are short 16-frame clips with $112 \times 112$-pixel crops as in the original implementation. Differently from [29] we used batch normalization after all convolutional and fully connected layers. Another difference to the original model is in the first pooling layer, we use a temporal stride of 2 instead of 1, which reduces the memory footprint and allows for bigger batches – this was important for batch normalization (especially after the fully connected layers, where there is no weight tying). Using this stride we were able to train with 15 videos per batch per GPU using standard K40 GPUs.

### 2.3. The Old III: Two-Stream Networks

LSTMs on features from the last layers of ConvNets can model high-level variation, but may not be able to capture fine low-level motion which is critical in many cases. It is also expensive to train as it requires unrolling the network through multiple frames for backpropagation-through-time.

A different, very practical approach, introduced by Simonyan and Zisserman [25], models short temporal snapshots of videos by averaging the predictions from a single RGB frame and a stack of 10 externally computed optical flow frames, after passing them through two replicas of an ImageNet pre-trained ConvNet. The flow stream has an adapted input convolutional layer with twice as many input channels as flow frames (because flow has two channels, horizontal and vertical), and at test time multiple snapshots are sampled from the video and the action prediction is averaged. This was shown to get very high performance on existing benchmarks, while being very efficient to train and test.

A recent extension [8] fuses the spatial and flow streams after the last network convolutional layer, showing some improvement on HMDB while requiring less test time augmentation (snapshot sampling). Our implementation follows this paper approximately using Inception-V1. The inputs to the network are 5 consecutive RGB frames sampled 10 frames apart, as well as the corresponding optical flow snippets. The spatial and motion features before the last average pooling layer of Inception-V1 ($5 \times 7 \times 7$ feature grids, corresponding to time, x and y dimensions) are passed through a $3 \times 3 \times 3$ 3D convolutional layer with 512 output channels, followed by a $3 \times 3 \times 3$ 3D max-pooling layer and through a final fully connected layer. The weights of these new layers are initialized with Gaussian noise.

Both models, the original two-stream and the 3D fused version, are trained end-to-end (including the two-stream averaging process in the original model).

### 2.4. The New: Two-Stream Inflated 3D ConvNets

With this architecture, we show how 3D ConvNets can benefit from ImageNet 2D ConvNet designs and, optionally, from their learned parameters. We also adopt a two-stream configuration here – it will be shown in section 4

that while 3D ConvNets can directly learn about temporal patterns from an RGB stream, their performance can still be greatly improved by including an optical-flow stream.

**Inflating 2D ConvNets into 3D.** A number of very successful image classification architectures have been developed over the years, in part through painstaking trial and error. Instead of repeating the process for spatio-temporal models we propose to simply convert successful image (2D) classification models into 3D ConvNets. This can be done by starting with a 2D architecture, and *inflating* all the filters and pooling kernels – endowing them with an additional temporal dimension. Filters are typically square and we just make them cubic – $N \times N$ filters become $N \times N \times N$.

**Bootstrapping 3D filters from 2D Filters.** Besides the architecture, one may also want to bootstrap parameters from the pre-trained ImageNet models. To do this, we observe that an image can be converted into a (boring) video by copying it repeatedly into a video sequence. The 3D models can then be implicitly pre-trained on ImageNet, by satisfying what we call the boring-video fixed point: the pooled activations on a boring video should be the same as on the original single-image input. This can be achieved, thanks to linearity, by repeating the weights of the 2D filters $N$ times along the time dimension, and rescaling them by dividing by $N$. This ensures that the convolutional filter response is the same. Since the outputs of convolutional layers for boring videos are constant in time, the outputs of pointwise non-linearity layers and average and max-pooling layers are the same as for the 2D case, and hence the overall network response respects the boring-video fixed point.

**Pacing receptive field growth in space, time and network depth.** The boring video fixed-point leaves ample freedom on how to inflate pooling operators along the time dimension and on how to set convolutional/pooling temporal stride – these are the primary factors that shape the size of feature receptive fields. Virtually all image models treat the two spatial dimensions (horizontal and vertical) equally – pooling kernels and strides are the same. This is quite natural and means that features deeper in the networks are equally affected by image locations increasingly far away in both dimensions. A symmetric receptive field is however not necessarily optimal when also considering time – this should depend on frame rate and image dimensions. If it grows too quickly in time relative to space, it may conflate edges from different objects breaking early feature detection, while if it grows too slowly, it may not capture scene dynamics well.

In Inception-v1, the first convolutional layer has stride 2, then there are four max-pooling layers with stride 2 and a $7 \times 7$ average-pooling layer preceding the last linear classification layer, besides the max-pooling layers in parallel Inception branches. In our experiments the input videos were processed at 25 frames per second; we found it helpful to not perform temporal pooling in the first two max-pooling layers (by using $1 \times 3 \times 3$ kernels and stride 1 in time), while having symmetric kernels and strides in all other max-pooling layers. The final average pooling layer uses a $2 \times 7 \times 7$ kernel. The overall architecture is shown in fig. 3. We train the model using 64-frame snippets and test using the whole videos, averaging predictions temporally.

**Two 3D Streams.** While a 3D ConvNet should be able to learn motion features from RGB inputs directly, it still performs pure feedforward computation, whereas optical flow algorithms are in some sense recurrent (e.g. they perform iterative optimization for the flow fields). Perhaps because of this lack of recurrence, experimentally we still found it valuable to have a two-stream configuration – shown in fig. 2, e) – with one I3D network trained on RGB inputs, and another on flow inputs which carry optimized, smooth flow information. We trained the two networks separately and averaged their predictions at test time.

## 2.5. Implementation Details

All models but the C3D-like 3D ConvNet use ImageNet-pretrained Inception-V1 [13] as base network. For all architectures we follow each convolutional layer by a batch normalization [13] layer and a ReLU activation function, except for the last convolutional layers which produce the class scores for each network.

Training on videos used standard SGD with momentum set to 0.9 in all cases, with synchronous parallelization across 32 GPUs for all models except the 3D ConvNets which receive a large number of input frames and hence require more GPUs to form large batches – we used 64 GPUs for these. We trained models on miniKinetics for up to 35k steps, and for 110k steps on Kinetics, with a 10x reduction of learning rate when validation loss saturated. We tuned the learning rate hyperparameter on the validation set of miniKinetics. Models were trained for up to 5k steps on UCF-101 and HMDB-51 using a similar learning rate adaptation procedure as for Kinetics but using just 16 GPUs. All the models were implemented in TensorFlow [1].

Data augmentation is known to be of crucial importance for the performance of deep architectures. During training we used random cropping both spatially – resizing the smaller video side to 256 pixels, then randomly cropping a $224 \times 224$ patch – and temporally, when picking the starting frame among those early enough to guarantee a desired number of frames. For shorter videos, we looped the video as many times as necessary to satisfy each model's input interface. We also applied random left-right flipping consistently for each video during training. During test time the models are applied convolutionally over the whole video

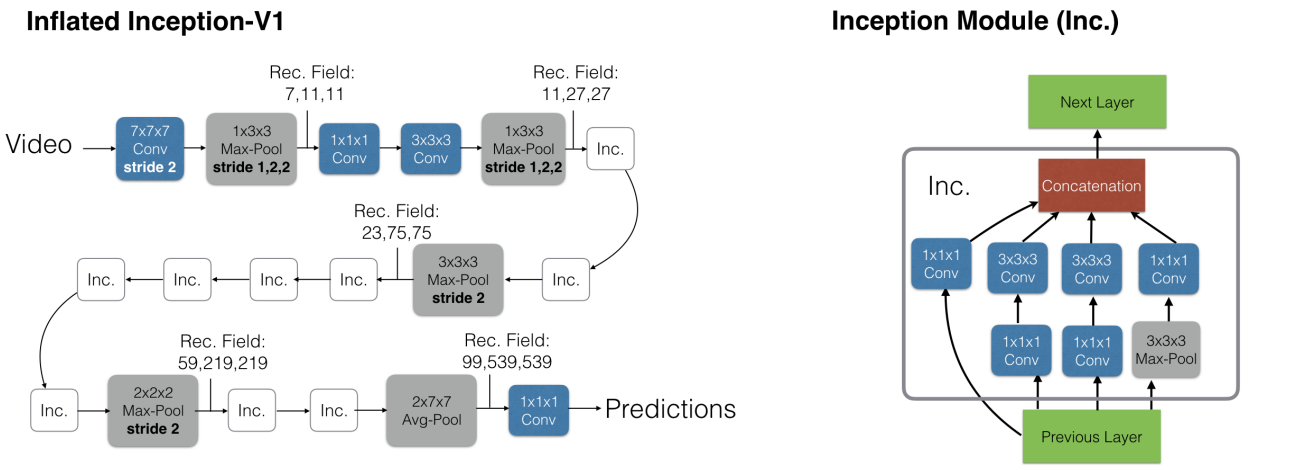


Figure 3. The Inflated Inception-V1 architecture (left) and its detailed inception submodule (right). The strides of convolution and pooling operators are 1 where not specified, and batch normalization layers, ReLu's and the softmax at the end are not shown. The theoretical sizes of receptive field sizes for a few layers in the network are provided in the format "time,x,y" – the units are frames and pixels. The predictions are obtained convolutionally in time and averaged.

| Method | #Params | Training | | Testing | |
|---|---|---|---|---|---|
| | | # Input Frames | Temporal Footprint | # Input Frames | Temporal Footprint |
| ConvNet+LSTM | 9M | 25 rgb | 5s | 50 rgb | 10s |
| 3D-ConvNet | 79M | 16 rgb | 0.64s | 240 rgb | 9.6s |
| Two-Stream | 12M | 1 rgb, 10 flow | 0.4s | 25 rgb, 250 flow | 10s |
| 3D-Fused | 39M | 5 rgb, 50 flow | 2s | 25 rgb, 250 flow | 10s |
| Two-Stream I3D | 25M | 64 rgb, 64 flow | 2.56s | 250 rgb, 250 flow | 10s |

Table 1. Number of parameters and temporal input sizes of the models.

taking $224 \times 224$ center crops, and the predictions are averaged. We briefly tried spatially-convolutional testing on the $256 \times 256$ videos, but did not observe improvement. Better performance could be obtained by also considering left-right flipped videos at test time and by adding additional augmentation, such as photometric, during training. We leave this to future work.

We computed optical flow with a TV-L1 algorithm [35].

## 3. The Kinetics Human Action Video Dataset

The Kinetics dataset is focused on human actions (rather than activities or events). The list of action classes covers: *Person Actions (singular)*, e.g. drawing, drinking, laughing, punching; *Person-Person Actions*, e.g. hugging, kissing, shaking hands; and, *Person-Object Actions*, e.g. opening presents, mowing lawn, washing dishes. Some actions are fine grained and require temporal reasoning to distinguish, for example different types of swimming. Other actions require more emphasis on the object to distinguish, for example playing different types of wind instruments.

The dataset has 400 human action classes, with 400 or more clips for each class, each from a unique video. The clips last around 10s, and there are no untrimmed videos. The test set consists of 100 clips for each class. A full description of the dataset and how it was built is given in [16].

For most of the experiments in this paper we use a smaller dataset than the full Kinetics, called miniKinetics. This is an early version of the dataset having only 213 classes with a total of 120k clips across three splits, one for training with 150–1000 clips per class, one for validation with 25 clips per class and one for testing with 75 clips per class. MiniKinetics enabled faster experimentation, and was available before the full Kinetics dataset.

## 4. Experimental Comparison of Architectures

In this section we compare the performance of the five architectures described in section 2 whilst varying the dataset used for training and testing.

| Architecture | UCF-101 | | | HMDB-51 | | | miniKinetics | | |
|---|---|---|---|---|---|---|---|---|---|
| | RGB | Flow | RGB + Flow | RGB | Flow | RGB + Flow | RGB | Flow | RGB + Flow |
| (a) LSTM | 81.0 | – | – | 36.0 | – | – | 69.9 | – | – |
| (b) 3D-ConvNet | 51.6 | – | – | 24.3 | – | – | 60.0 | – | – |
| (c) Two-Stream | 83.6 | 85.6 | 91.2 | 43.2 | 56.3 | 58.3 | 70.1 | 58.4 | 72.9 |
| (d) 3D-Fused | 83.2 | 85.8 | 89.3 | 49.2 | 55.5 | 56.8 | 71.4 | 61.0 | 74.0 |
| (e) Two-Stream I3D | **84.5** | **90.6** | **93.4** | **49.8** | **61.9** | **66.4** | **74.1** | **69.6** | **78.7** |

Table 2. Architecture comparison: (left) training and testing on split 1 of UCF-101; (middle) training and testing on split 1 of HMDB-51; (right) training and testing on miniKinetics. All models are based on Imagenet pre-trained Inception-v1, except 3D-ConvNet, a C3D-like [29] model which has a custom architecture and was trained here from scratch. Note that the Two-Stream architecture numbers on individual RGB and Flow streams can be interpreted as a simple baseline which applies a ConvNet independently on 25 uniformly sampled frames then averages the predictions.

Table 2 shows the classification accuracy when training and testing on either UCF-101, HMDB-51 or miniKinetics. We test on the split 1 test sets of UCF-101 and HMDB-51 and on the held-out test set of miniKinetics. There are several noteworthy observations. First, our new I3D models do best in all datasets, with either RGB, flow, or RGB+flow modalities. This is interesting, given its very large number of parameters and that UCF-101 and HMDB-51 are so small, and shows that the benefits of ImageNet pre-training can extend to 3D ConvNets.

Second, the performance of all models is far lower on miniKinetics than on UCF-101, an indication of the different levels of difficulty of the two datasets. It is however higher than on HMDB-51; this may be in part due to lack of training data in HMDB-51 but also because this dataset was purposefully built to be hard: many clips have different actions in the exact same scene (e.g. "drawing sword" examples are taken from same videos as "sword" and "sword exercise"). Third, the ranking of the different architectures is mostly consistent. Fourth, the LSTM and 3D-ConvNet models are much more competitive on miniKinetics than on the smaller datasets; these models seem quite data-hungry.

Additionally, two-stream architectures exhibit superior performance on all datasets, but the relative value of RGB and flow differs significantly between miniKinetics and the other datasets. The contribution from flow alone, is slightly higher than that of RGB on UCF-101, much higher on HMDB-51, and substantially lower on miniKinetics. Visual inspection of the datasets suggests that Kinetics has much more camera motion which may make the job of the motion stream harder. The I3D model seems able to get more out of the flow stream than the other models, however, which can probably be explained by its much longer temporal receptive field (64 frames vs 10 during training) and more integrated temporal feature extraction machinery. While it seems plausible that the RGB stream has more discriminative information – we often struggled with our own eyes to discern actions from flow alone in Kinetics, and this was rarely the case from RGB – there may be opportunities for future research on integrating some form of motion stabilization into these architectures.

## 5. Experimental Evaluation of Features

In this section we investigate the generalizability of the networks trained on Kinetics. We consider two measures of this: first, we freeze the network weights and use the network to produce features for the (unseen) videos of the UCF-101/HMDB-51 datasets. We then train multi-way soft-max classifiers for the classes of UCF-101/HMDB-51 (using their training data), and evaluate on their test sets; Second, we fine-tune each network for the UCF-101/HMDB-51 classes (using the UCF-101/HMDB-51 training data), and again evaluate on the UCF-101/HMDB-51 test sets.

The results are given in table 3. The clear outcome is that *all* architectures benefit from pre-training on the additional video data of miniKinetics, but some benefit significantly more than the others – notably the I3D-ConvNet and 3D-ConvNet (although the latter starting from a much lower base). Training just the last layers of the models after pretraining in miniKinetics (*Fixed*) also leads to much better performance than directly training on UCF-101 and HMDB-51 for I3D models.

One explanation for the significant better transferability of features of I3D models is their high temporal resolution – they are trained on 64-frame video snippets at 25 frames per second and process all video frames at test time, which makes it possible for them to capture fine-grained temporal structure of actions. Stated differently, methods with sparser video inputs may benefit less from training on this large video dataset because, from their perspective, videos do not differ as much from the images in ImageNet. The difference over the C3D-like model can be explained by our I3D models being much deeper, while having much fewer parameters, by leveraging an ImageNet warm-start, by being trained on $4\times$ longer videos, and by operating on $2\times$

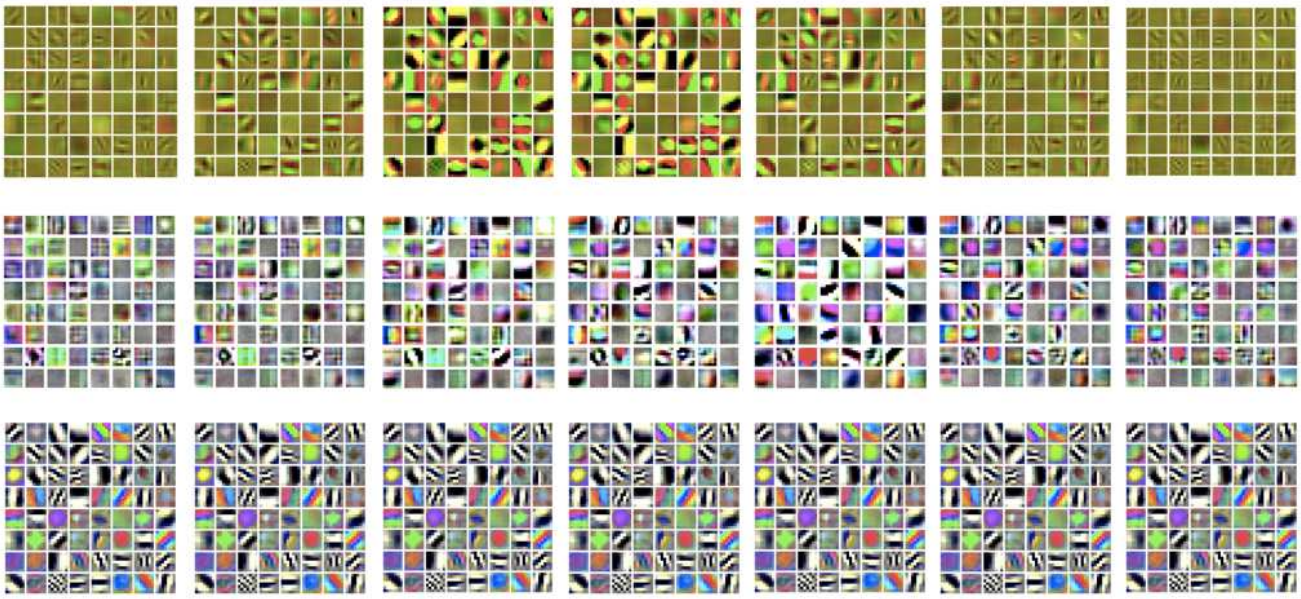

Figure 4. All 64 conv1 filters of each Inflated 3D ConvNet after training on Kinetics (the filter dimensions are $7 \times 7 \times 7$, and the 7 time dimensions are shown left-to-right across the figure). The sequence on top shows the flow network filters, the one in the middle shows filters from the RGB I3D network, and the bottom row shows the original Inception-v1 filters. Note that the I3D filters possess rich temporal structure. Curiously the filters of the flow network are closer to the original Inception-v1 filters, while the filters in the RGB I3D network are no longer recognizable. Best seen on the computer, in colour and zoomed in.

higher spatial resolution videos.

### 5.1. Comparison with the State-of-the-Art

We show a comparison of the performance of I3D models and previous state-of-the-art methods in table 4, on UCF-101 and HMDB-51. We include results when pre-training on miniKinetics and on the full Kinetics dataset. The conv1 filters of the trained models are shown in fig. 4.

Many methods get similar results, but the best performing method on these datasets is currently the one by Feichtenhofer and colleagues [7], which uses ResNet-50 models on RGB and optical flow streams, and gets 94.6% on UCF-101 and 70.3% on HMDB-51 when combined with the dense trajectories model [30]. We benchmarked our methods using the mean accuracy over the three standard train/test splits. Either of our RGB-I3D or RGB-Flow models alone, when pre-trained on Kinetics, outperforms all previous published performance by any model or model combinations. Our combined two-stream architecture widens the advantage over previous models considerably, bringing overall performance to 97.9 on UCF-101 and 80.2 on HMDB-51, which correspond to 57% and 33% misclassification reductions, respectively compared to the best previous model [7].

The difference between Kinetics pre-trained I3D models and prior 3D ConvNets (C3D) is even larger, although C3D is trained on more videos, 1M examples from Sports-1M plus an internal dataset, and even when ensembled and combined with IDT. This may be explainable by the better quality of Kinetics but also because of I3D simply being a better architecture.

Another noteworthy effect is the improvement of the I3D RGB stream on HMDB-51 when going from miniKinetics to Kinetics pre-training, which suggests that 3D ConvNets may require large amounts of data in order to learn robust motion features. After Kinetics pre-training, the two streams obtain similar performance but are still complementary: averaging their predictions brings results from 74.6% to 80.2%.

## 6. Discussion

We return to the question posed in the introduction, "is there a benefit in transfer learning from videos?". It is evident that there is a *considerable* benefit in pre-training on (the large video dataset) Kinetics, just as there has been such benefits in pre-training ConvNets on ImageNet for so many tasks. This demonstrates transfer learning from one dataset (Kinetics) to another dataset (UCF-101/HMDB-51)

| Architecture | UCF-101 | | | | HMDB-51 | | | |
|---|---|---|---|---|---|---|---|---|
| | Original | Fixed | Full-FT | Δ | Original | Fixed | Full-FT | Δ |
| (a) LSTM | 81.0 | 81.6 | 82.1 | -6% | 36.0 | 46.6 | 46.4 | -10.6% |
| (b) 3D-ConvNet | 49.2 | 76.0 | 79.9 | **-60.5%** | 24.3 | 47.5 | 49.4 | **-25.1%** |
| (c) Two-Stream | 91.2 | 90.3 | 91.5 | -3.4% | 58.3 | 64.0 | 58.7 | -5.7% |
| (d) 3D-Fused | 89.3 | 88.5 | 90.1 | -7.5% | 56.8 | 59.0 | 61.4 | -4.6% |
| (e) Two-Stream I3D | **93.4** | **95.7** | **96.5** | -47.0% | **66.4** | **74.3** | **75.9** | -9.5% |

Table 3. Performance on the UCF-101 and HMDB-51 test sets (splits 1 of both) for architectures pre-trained on miniKinetics. All except 3D-ConvNet are based on Inception-v1 and start off pre-trained on ImageNet. Original: train on UCF-101 / HMDB-51; Fixed: features from miniKinetics, with the last layer trained on UCF-101 / HMDB-51; Full-FT: miniKinetics pre-training with end-to-end fine-tuning on UCF-101 / HMDB-51; Δ shows the difference in misclassification as percentage between Original and the best of Full-FT and Fixed.

| Model | UCF-101 | HMDB-51 |
|---|---|---|
| Two-Stream [25] | 88.0 | 59.4 |
| IDT [30] | 86.4 | 61.7 |
| Dynamic Image Networks + IDT [2] | 89.1 | 65.2 |
| TDD + IDT [31] | 91.5 | 65.9 |
| Two-Stream Fusion + IDT [8] | 93.5 | 69.2 |
| Temporal Segment Networks [32] | 94.2 | 69.4 |
| ST-ResNet + IDT [7] | 94.6 | 70.3 |
| Deep Networks [15], Sports 1M pre-training | 65.2 | - |
| C3D one network [29], Sports 1M pre-training | 82.3 | - |
| C3D ensemble [29], Sports 1M pre-training | 85.2 | - |
| C3D ensemble + IDT [29], Sports 1M pre-training | 90.1 | - |
| RGB-I3D, miniKinetics pre-training | 91.8 | 66.4 |
| RGB-I3D, Kinetics pre-training | 95.4 | 74.5 |
| Flow-I3D, miniKinetics pre-training | 94.7 | 72.4 |
| Flow-I3D, Kinetics pre-training | 95.4 | 74.6 |
| Two-Stream I3D, miniKinetics pre-training | 96.9 | 76.3 |
| Two-Stream I3D, Kinetics pre-training | **97.9** | **80.2** |

Table 4. Comparison with state-of-the-art on the UCF-101 and HMDB-51 datasets, averaged over three splits. First set of rows contains results of models trained without labeled external data.

for a similar task (albeit for different action classes). However, it still remains to be seen if there is a benefit in using Kinetics pre-training for other video tasks such as semantic video segmentation, video object detection, or optical flow computation. We plan to make publicly available I3D models trained on the official Kinetics dataset's release to facilitate research in this area.

Of course, we did not perform a comprehensive exploration of architectures – for example we have not employed action tubes [11, 17] or attention mechanisms [20] to focus in on the human actors. Recent works have proposed imaginative methods for determining the spatial and temporal extent (detection) of actors within the two-stream architectures, by incorporating linked object detections in time [22, 24]. The relationship between space and time is a mysterious one. Several very creative papers have recently gone out of the box in attempts to capture this relationship, for example by learning frame ranking functions for action classes and using these as a representation [9], by making analogies between actions and transformations [33], or by creating 2D visual snapshots of frame sequences [2] – this idea is related to the classic motion history work of [3]. It would be of great value to also include these models in our comparison but we could not, due to lack of time and space.

As future work, we plan to repeat all experiments using Kinetics instead of miniKinetics, with and without ImageNet pre-training, and explore inflating other state-of-the-art 2D ConvNets.

## Acknowledgements:

We would like to thank everyone on the Kinetics project and in particular Brian Zhang and Tim Green for help setting up the data for our experiments, and Karen Simonyan for comments, helpful clarifications and for sharing code.

## References


[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. 4

[2] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *IEEE International Conference on Computer Vision and Pattern Recognition CVPR*, 2016. 8

[3] A. F. Bobick and J. W. Davis. The recognition of human movement using temporal templates. *IEEE Transactions on pattern analysis and machine intelligence*, 23(3):257–267, 2001. 8

[4] T. Cooijmans, N. Ballas, C. Laurent, and A. Courville. Recurrent batch normalization. *arXiv preprint arXiv:1603.09025*, 2016. 2

[5] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2625–2634, 2015. 2

[6] A. Fathi and G. Mori. Action recognition by learning mid-level motion features. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. 2

[7] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Spatiotemporal residual networks for video action recognition. *arXiv preprint arXiv:1611.02155*, 2016. 7, 8

[8] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *IEEE International Conference on Computer Vision and Pattern Recognition CVPR*, 2016. 2, 3, 8

[9] B. Fernando, E. Gavves, J. M. Oramas, A. Ghodrati, and T. Tuytelaars. Modeling video evolution for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5378–5387, 2015. 8

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 1

[11] G. Gkioxari and J. Malik. Finding action tubes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 759–768, 2015. 8

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, 2016. 2

[13] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 2, 4

[14] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013. 2, 3

[15] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014. 2, 3, 8

[16] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman. The kinetics human action video dataset. *arXiv preprint*, 2017. 1, 5

[17] A. Kläser, M. Marszalek, C. Schmid, and A. Zisserman. Human focused action localization in video. In *International Workshop on Sign, Gesture, Activity, ECCV 2010*, 2010. 8

[18] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011. 1

[19] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. 2

[20] Z. Li, E. Gavves, M. Jain, and C. G. Snoek. VideoLSTM convolves, attends and flows for action recognition. *arXiv preprint arXiv:1607.01794*, 2016. 8

[21] J. C. Niebles, H. Wang, and L. Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. *International journal of computer vision*, 79(3):299–318, 2008. 2

[22] X. Peng and C. Schmid. Multi-region two-stream R-CNN for action detection. In *European Conference on Computer Vision*, pages 744–759. Springer, 2016. 8

[23] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1

[24] S. Saha, G. Singh, M. Sapienza, P. H. Torr, and F. Cuzzolin. Deep learning for detecting multiple space-time action tubes in videos. *British Machine Vision Conference (BMVC) 2016*, 2016. 8

[25] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576, 2014. 2, 3, 8

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 2

[27] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 1

[28] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler. Convolutional learning of spatio-temporal features. In *European conference on computer vision*, pages 140–153. Springer, 2010. 2, 3

[29] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497. IEEE, 2015. 2, 3, 6, 8

[30] H. Wang and C. Schmid. Action recognition with improved trajectories. In *International Conference on Computer Vision*, 2013. 2, 7, 8

[31] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Proceedings of the IEEE Conference on Computer Vision and*

*Pattern Recognition*, pages 4305–4314, 2015. 8

[32] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: towards good practices for deep action recognition. In *European Conference on Computer Vision*, 2016. 8

[33] X. Wang, A. Farhadi, and A. Gupta. Actions˜ transformations. *arXiv preprint arXiv:1512.00795*, 2015. 8

[34] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4694–4702, 2015. 2

[35] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime TV-L1 optical flow. *Pattern Recognition*, pages 214–223, 2007. 5