Rethinking Spatiotemporal Feature Learning For Video Understanding

Saining Xie[1,2]   Chen Sun[1]   Jonathan Huang[1]   Zhuowen Tu[2]   Kevin Murphy[1]
[1] Google Research   [2] UC San Diego

Abstract

In this paper we study 3D convolutional networks for video understanding tasks. Our starting point is the state-of-the-art I3D model of [3], which "inflates" all the 2D filters of the Inception architecture to 3D. We first consider "deflating" the I3D model at various levels to understand the role of 3D convolutions. Interestingly, we found that 3D convolutions at the top layers of the network contribute more than 3D convolutions at the bottom layers, while also being computationally more efficient. This indicates that I3D is better at capturing high-level temporal patterns than low-level motion signals. We also consider replacing 3D convolutions with spatiotemporal-separable 3D convolutions (i.e., replacing convolution using a $k_t \times k \times k$ filter with $1 \times k \times k$ followed by $k_t \times 1 \times 1$ filters); we show that such a model, which we call S3D, is 1.5x more computationally efficient (in terms of FLOPS) than I3D, and achieves better accuracy. Finally, we explore spatiotemporal feature gating on top of S3D. The resulting model, which we call S3D-G, outperforms the state-of-the-art I3D model by 3.5% accuracy on Kinetics and reduces the FLOPS by 34%. It also achieves a new state-of-the-art performance when transferred to other action classification (UCF-101 and HMDB-51) and detection (UCF-101 and JHMDB) datasets.

1. Introduction

There has been tremendous progress in computer vision over the past few years due to the success of deep convolutional neural networks (CNNs) for visual feature learning. To understand static images, novel CNN architectures are proposed every year, resulting in higher accuracy across a wide range of visual recognition tasks. Many of these models are "pretrained" on the ImageNet dataset [38], which contains over 1M labeled images.

However, progress on action recognition in videos has been comparably much slower. In particular, many approaches struggle to even beat the simple baseline of treating a video as a bag of frames, discarding temporal information completely. For example, temporal segment networks [53], which is the current state-of-the-art method,

use the 2D Inception-V1 architecture [45], which processes each frame separately.

Until recently, one of the likely major impediments to progress was the lack of a video equivalent of Imagenet. However, with the recent release of the Kinetics dataset [24], a dataset of temporally trimmed video clips with more than 200K examples covering 400 action classes, this is no longer a problem. By pre-training on the Kinetics dataset, Carreira and Zisserman [3] showed that they could achieve state-of-the-art performance by using a 3D convolutional adaptation of the Inception-V1 architecture [45]; they call their method "I3D", since it "inflates" the 2D convolutional filters of Inception to 3D.

Despite giving good performance, 3D convolutions have many more parameters than 2D convolutions, and 3D models are computationally much more intensive than 2D models. This prompts several questions, which we seek to address in this paper:

- Do we need 3D convolutions in all layers of the network? If not, should we replace them with 2D at the lowest layers (thus discarding potential pixel-level motion signals), or at the highest layers (thus discarding "semantic level" temporal signals)?

- Is it important that we convolve jointly over time and space? Or would it suffice to convolve over these dimensions independently?

- How can we use answers to the above questions to improve on prior methods in terms of accuracy, speed and memory footprint?

To answer the first question, we apply "network surgery" to obtain several variants of the I3D architecture. In one family of variants, which we call Bottom-heavy-I3D-K, we retain 3D temporal convolutions at the lowest K layers of the network (the ones closest to the pixels), and use 2D convolutions for the higher layers. However, deflating high level 3D layers is not particularly beneficial from a computational perspective, as the majority of FLOPS of the network are spent in the low-level layers, due to the larger spatial input. We therefore consider a second family of I3D variants, which we call Top-heavy-I3D-K, where we keep

3D temporal convolutions at the top K layers of the network (the most "abstract" ones), and use 2D convolutions for the lower layers.

We then investigate how to trade between accuracy and speed by varying K in these I3D variants. In particular, we show the somewhat surprising result that Top-heavy-I3D-K models significantly outperform Bottom-heavy-I3D-K models in terms of the accuracy-speed trade-off, at least when it comes to video classification tasks. This suggests that 3D convolution is more useful at the higher, abstract levels than it is at the lower levels.

To answer the second question (about separating space and time), we consider replacing 3D convolutions with temporally separable 3D convolutions, i.e., we replace filters of the form $k_t \times k \times k$ by $1 \times k \times k$ followed by $k_t \times 1 \times 1$, where k_t is the width of the filter in time, and k is the height/width of the filter in space We call the resulting model S3D, which stands for "separable 3D CNN". This obviously has many fewer parameters than models that use standard 3D convolution, and it is more computationally efficient. Surprisingly, we also show that it has slightly better accuracy than the original I3D model.

Finally, to answer the third question (about how to achieve state of the art), we combined what we have learned in answering the above two questions with a novel spatiotemporal gating mechanism to design a new model architecture which we call S3D-G. We show that this is over 3.5% better than I3D (the previous state-of-the-art) on the challenging Kinetics dataset, yet has fewer parameters and FLOPS. It also achieves a new state of the art on other video classification datasets, such as UCF-101 and HMDB, and even other tasks, such as action localization on JHMDB.

In summary, our main contributions are as follows:

- We conduct a thorough investigation on the speed and accuracy trade-offs for 3D convolution operations for video understanding, in the context of I3D.

- We propose to replace the standard 3D convolution operation used in video understanding models by factoring it along spatial and temporal dimensions. We show that this "separable conv3D" operation achieves a slightly higher accuracy with significantly fewer parameters and fewer FLOPS.

- We design a new video CNN architecture that combines separable conv3D with a form of feature gating. This new model, which we call S3D-G, outperforms previous models significantly on video classification and action detection tasks.

2. Related work

Early work on visual recognition in video focused on classification tasks with a limited number of categories and on small and highly-controlled data (*e.g.*, [2, 40]). Traditional methods, such as STIP [27] and HOG3D [25], extended hand-crafted local image features to 3D, and aggregated them as bags-of-words for classification. Since then, bigger and more realistic datasets have been proposed [37, 43, 19, 41], and hand-crafted features have also improved significantly [49, 44, 48].

There have also been several attempts to apply deep learning to video understanding. For example, Karpathy *et al.* [23] investigated several 2D CNN architectures with different temporal pooling strategies, but the quality of the learned features was much lower than state-of-the-art hand-crafted features at the time. Simonyan and Zisserman [42] introduced the first state-of-the-art deep learning system for action recognition. To capture motion information together with appearance, they proposed a two-stream architecture where one CNN stream handles raw RGB input, and the other handles pre-computed optical flow. It was shown that the pre-computed flow is crucial to achieve good performance. Since then, many works on video understanding follow the same multi-stream 2D CNN design, and have made improvements in terms of backbone architecture [10, 52, 30, 12], fusion of the streams [11, 8, 9, 58] and exploiting richer temporal structures [6, 54, 53].

On the other hand, attempts to learn motion features end-to-end, either by 3D convolutional filters (*e.g.* C3D [46]) or by applying FlowNet [20] style architectures for classification [29], often lead to inferior performance when compared with the two-stream frameworks that encode motion with optical flow. One dilemma for such approaches is that the models typically have more model parameters and need to be trained from scratch, while the existing datasets are either too small (*e.g.* UCF-101 [43]) or too loosely labeled (*e.g.* Sports-1M [43]). More recently, Tran *et al.* [47] proposed a C3D variant based on ResNet [15] and ablations with a mixture of 2D and 3D convolutions, including a *2.5D* architecture with separate spatial and temporal convolutions, but the study was conducted on the small-scale UCF-101 dataset and it is unclear if the observations will be consistent on large-scale video dataset. A set of ResNet-based *2.5D* architectures were also studied by [34], but without comparing their 3D counterparts.

The Inception 3D (I3D) architecture [3] proposed by Carreira and Zisserman significantly improved the performance of 3D CNNs; this model is the current state-of-the-art. There are three key ingredients for its success: first, they *inflate* all the 2D convolution filters used by the Inception V1 architecture [45] into 3D convolutions (see Figure 1(a), top left), and carefully choose the temporal kernel size in the earlier layers. Second, they initialize the inflated model weights by duplicating the pre-trained weights from ImageNet over the temporal dimension. Finally, they train the I3D network on the Kinetics dataset [24], which

(a) I3D

(b) I2D

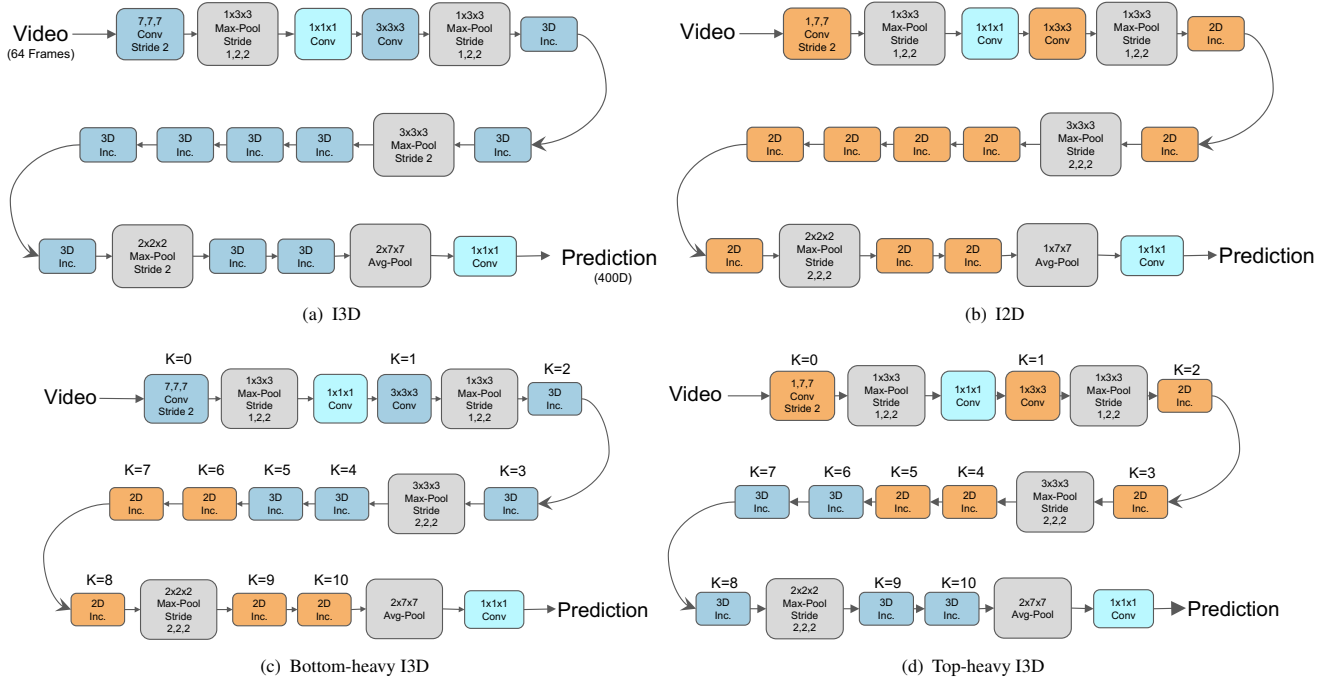**(c) Bottom-heavy I3D**

(d) Top-heavy I3D

Figure 1. Illustration of some of the networks we consider in this paper; (a) and (b) illustrate the two extremes, I3D and I2D, respectively; (c) and (d) illustrate bottom heavy and top heavy variants wherein 3D convolutions are replaced by 2D convolutions beginning from the top or bottom. K indexes the spatio-temporal convolutional layers. The "2D Inc." and "3D Inc." blocks refer to 2D and 3D inception blocks, defined in Figure 2.
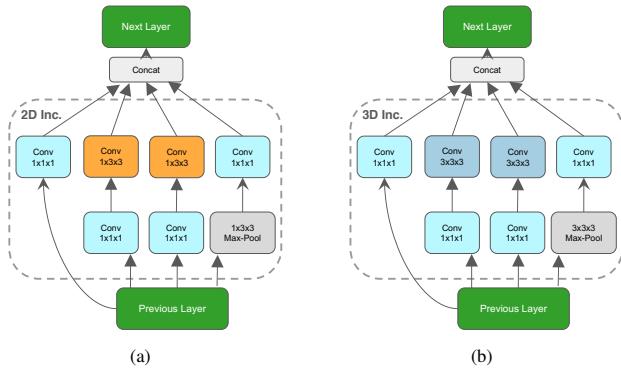

Figure 2. (a) 2D Inception block; (b) 3D Inception block.

is a large-scale video classification dataset collected from YouTube, containing 400 action classes and 240K training examples. Each example is temporally trimmed to be around 10 seconds.

3. Replacing 3D convolutions with 2D

In this section, we study the consequences (both in terms of accuracy and speed) of replacing 3D convolutions with 2D convolutions, either in every layer of the I3D model, or in a subset of layers.

3.1. Experimental Setup

The full Kinetics dataset is quite large, containing 240k video clips. To speed up experiments, it is helpful to have a smaller dataset. Unfortunately, the "mini-Kinetics" dataset used in [3] has been deprecated, since it contains videos that are no longer publicly available. In collaboration with the original authors, we have created a new split of the Kinetics dataset that we call Mini-Kinetics-200. This consists of the 200 categories with most training examples; for each category, we randomly sample 400 examples from the training set, and 25 examples from the validation set, resulting in 80K training examples and 5K validation examples in total. This split will be publicly released to enable future comparisons. We also report some results on the original Kinetics dataset, which we will call Full-Kinetics for clarity.

Our experimental setup largely follows [3]. During training, we densely sample 64 frames from a video and resize input frames to 256×256 and then take random crops of size 224×224. During evaluation, we use all frames and take 224×224 center crops from the resized frames. Our models are implemented with TensorFlow and optimized with a vanilla synchronous SGD algorithm with momentum of 0.9 and on 56 GPUs. We use batch of 6 per GPU, and train our model for 80k steps with an initial learning rate of 0.1. We decay the learning rate at step 60k to 0.01, and step 70k to

0.001. We report top-1 and top-5 accuracy as the performance metrics. All results are on Mini-Kinetics-200 unless noted otherwise.

To measure the computational efficiency of our models, we report theoretical FLOPS based on a single input video sequence of 64 frames and spatial size 224×224.

3.2. Replacing *all* 3D convolutions with 2D

In this section, we seek to determine how much value 3D convolution brings. We do this by replacing every 3D filter in the I3D model with a 2D filter. This yields what we will refer to as the *I2D* model (see Figure 1(b)). Note that we replace the 3D max-pooling layers in each inception block with 2D max-pooling too. However, to reduce the memory and time requirements, and to keep the training protocol identical to I3D (in terms of the number of clips we use for training in each batch, etc), we retain two max-pooling layers with stride 2 between Inception modules. Hence, strictly speaking, I2D is not a pure 2D model. However, it is very similar to a single-frame 2D classification model, since it is effectively a stack of point-wise (with respect to the temporal axis) convolution layers and temporal pooling layers.

Model	Normal (%)	Shuffled (%)	Reversed (%)
I3D	71.66	45.37	71.54
I2D	67.00	67.52	67.23

Table 1. Top-1 accuracy on Full-Kinetics dataset. For a pretrained model on normal frame orders, we manipulate the temporal ordering of testing frames by shuffling and reversing them. The results show that the performance of I2D baseline is not affected by shuffling and thus the model is order invariant. I3D performance drops severely if the input frames are shuffled. However, for both I2D ad I3D, it makes no difference in accuracy whether or not temporal ordering is reversed.

The whole I2D network is order invariant on the temporal ordering of input frames. To verify this, we train I2D and the original I3D model on the Full-Kinetics dataset with normal frame order, and apply the trained models on validation data in which the frames are in normal order, randomly shuffled order, and reversed temporal order. The results of the experiment are shown in Table 1. We see that I2D has the same performance on all three versions during testing, as is to be expected. We also see that I3D's performance on the randomly shuffled data is much worse than on the normal form of the data; however, its performance on the reversed form is the same, indicating that this model and/or dataset does not allow (or require) inferring the causal "arrow of time" [33].

3.3. Replacing *some* 3D convolutions with 2D

Although we have seen that 3D convolution can boost accuracy compared to 2D convolution, it is computation-ally very expensive. In this section, we investigate the consequences of only replacing some of the 3D convolutions with 2D. Specifically, starting with an I2D model, we gradually inflate 2D convolutions into 3D, from low-level to high-level layers in the network, to create what we call the Bottom-heavy-I3D model; if we inflate the first K temporal convolutional layers (either regular convolution or an inception block), we call the model Bottom-heavy-I3D-K, as shown in Figure 1(c). This is equivalent to what [47] call "MC" networks, which stands for "mixed 3D-2D convolution".

However, since the feature maps are much larger (spatially) at the lower layers, it makes more sense, from a computational perspective, to inflate the top layers of the model to 3D, but keep the lower layers 2D; we call such models Top-heavy-I3D models. Specifically, Top-heavy-I3D-K denotes a model in which the topmost K convolutional layers are 3D, and the rest are 2D, as shown in Figure 1(d).

We train and evaluate the Bottom-heavy-I3D-K and Top-heavy-I3D-K models on Mini-Kinetics-200 and show the results in Figure 3. Not surprisingly, Top-heavy-I3D models are faster, since the spatially large feature maps at the lower layers are only convolved in 2D. More surprising is the fact that such models are also significantly more accurate. This seems to indicate that temporal patterns amongst high level features are more useful (for this task) than low level motion patterns.

This result is also supported by our analysis of the weights of an I3D model which was trained on Full-Kinetics. Figure 4 shows the distribution of these weights across the 4 layers of our model, from low-level to high-level. In particular, each boxplot shows the distribution of $W_l(t, :, :, :)$ for temporal offset t and layer l. We use $t = 0$ to indicate no offset in time, i.e., the center in the temporal kernel. At initialization, all the filters started with the same set of (2D convolution) weights (derived from an Inception model pre-trained on Imagenet) for each value of $t \in \{-1, 0, 1\}$. After training, we see that the weights tend to concentrate on temporally centered ($t = 0$) filters in the lower layers, but have learned some interesting patterns in the higher layers. This suggests once again that the higher level temporal patterns are more useful, or alternatively, current spatiotemporal convolutions are more capable of modeling high-level feature representations, for the Kinetics action classification task.

4. Separating temporal convolution from spatial convolutions

In this section, we study the effect of replacing standard 3D convolution with a factored version which disentangles this operation into a temporal part and a spatial part.

In more detail, our method is to replace each 3D convolution with two consecutive convolution layers: one 2D
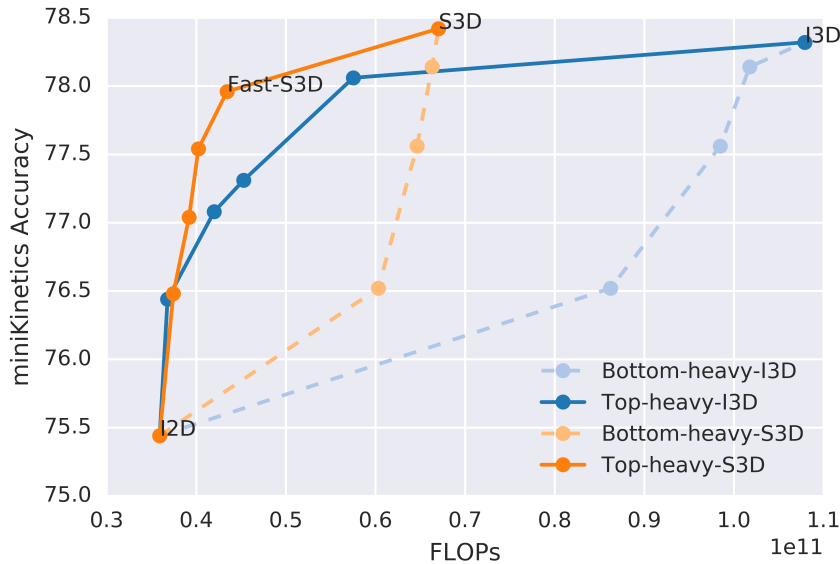
Figure 3. Accuracy (on Mini-Kinetics-200) vs number of FLOPS needed to perform inference on 64 frames. Solid-lines denote Top-heavy-I3D-K models for different K, and dotted lines denote Bottom-heavy-I3D-K models. Blue curves denote I3D based models, and Orange curves denote S3D based models. The point labeled "Fast-S3D" denotes a model on the "sweet spot" of the accuracy-speed trade-off curve. Figure is best viewed in color.
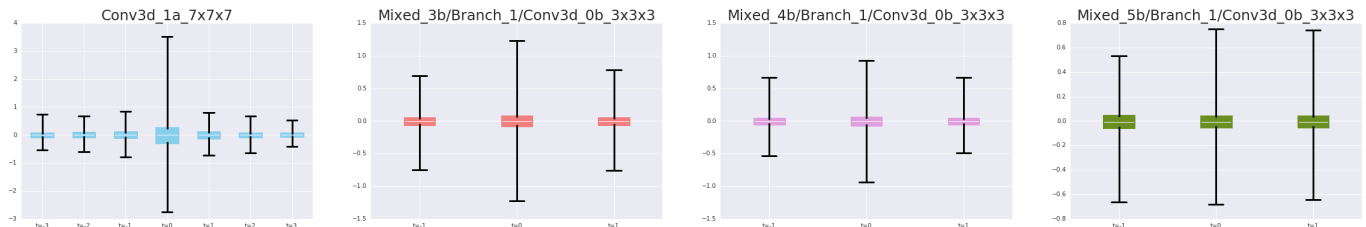

Figure 4. Statistics of convolutional filter weights of an I3D model trained on Full-Kinetics. Each boxplot shows the distribution of $W_l(t, :, :, :)$ for temporal offset t, with $t = 0$ being in the middle. Results for different layers l are shown in different panels, with lowest layers on the left. All filters with different temporal offset are initialized with the same set of weights. Low-level filters essentially ignore the temporal dimension, unlike higher level filters, where the weights distributed nicely across different temporal offsets.

convolution layer to learn spatial features, followed by a 1D convolution layer purely on the temporal axis. This can be implemented seamlessly in the I3D framework by running two 3D convolutions, where the first (spatial) convolution has filter shape $[1, k, k]$ and the second (temporal) convolution has filter shape $[k, 1, 1]$. By doing this factorization for every 3D convolution, we obtain a new model which we refer to as S3D. For a detailed illustration of the architecture, please refer to Figure 5(a).

This factorization is similar in spirit to the depth-wise separable convolutions used in [4, 16, 56], except that we apply the idea to the temporal dimension instead of the feature dimension. It is computationally more efficient than a full 3D convolution, but theoretically also less expressive, as the spatial processing and temporal processing are sequential and independent. So the important question is how much this will impact accuracy. We study this below.

Note that we can apply this transformation to any place where 3D convolution is used; thus this idea is orthogonal to the question of which layers should contain 3D convolution, which we discussed in Section 3. We denote the separable version of the Bottom-heavy-I3D models by Bottom-heavy-S3D, and the separable version of the Top-heavy-I3D models by Top-heavy-S3D, thus giving us 4 families of models.

4.1. Experimental results

Figure 3 compares the results of the S3D models (orange lines) against their corresponding I3D counterparts (blue lines). The results show that, despite a substantial compression in model size ($12.06M$ parameters for I3D reduced to $8.77M$ for S3D), and a large speed-up (107.9 GFLOPS for S3D reduced to 66.38 GFLOPS for I3D), the separable
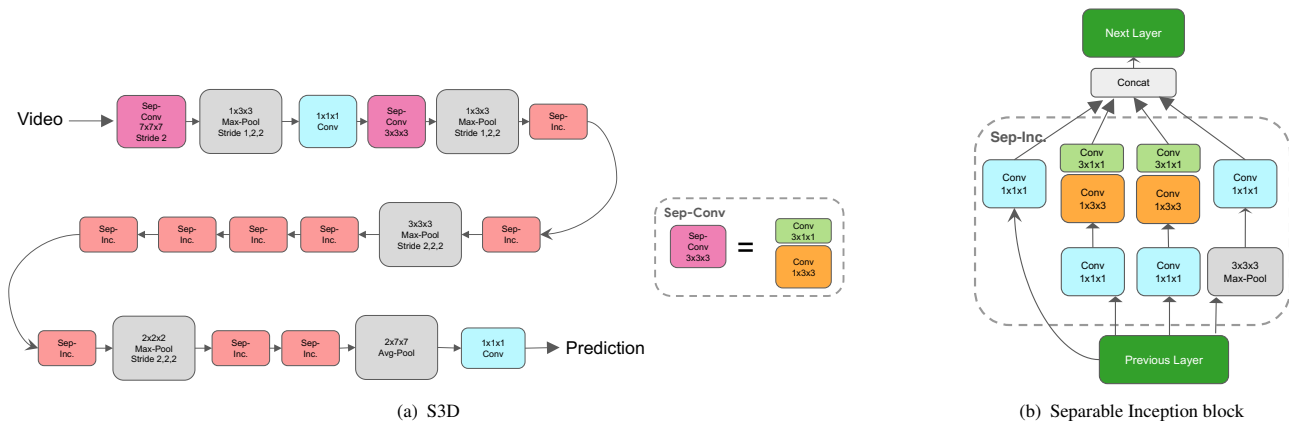
(a) S3D

(b) Separable Inception block

Figure 5. (a) An illustration of the S3D model. Dark red boxes are temporal separable convolutions (sep-conv), and pink boxes are temporal separable inception blocks, shown in (b).

model is even more accurate.

We believe the gain in accuracy is because the factorization reduces overfitting. As further evidence in support of this hypothesis, Table 2 compares an I3D and S3D model trained from scratch (i.e., without ImageNet pretraining) on the Full-Kinetics dataset. We see that S3D is significantly better.

Model	Top-1(%)	Top-5(%)	Params (M)	FLOPS (G)
I3D	68.40	88.00	12.06	107.9
S3D	69.44	89.11	8.77	66.38

Table 2. Accuracy on Full-Kinetics when training from scratch. We see that S3D has better accuracy, and also has much smaller model size and FLOPS.

To get even greater speed, we can apply temporal separable convolution to just the top K layers of the model, and replace the remaining 3D convolutions with 2D convolutions. In particular, based on Figure 3, we decide to keep the top 2 layers as separable 3D convolutions, and make the rest 2D convolutions. We denote this model as *Fast-S3D* in Figure 3. This model is 2.5 times more efficient than I3D (43.47 GFLOPS vs 107.9), and yet has comparable accuracy (78.0% vs 78.4% on Mini-Kinetics-200). We believe such lightweight models will be useful for processing very large video datasets, as well as for mobile applications.

4.2. A temporally separable version of an Inception block

There are 4 branches in an Inception block, but only two of them have 3x3 convolutions (the other two being pointwise 1x1 convolutions), as shown in Figure 2. As such, when I3D inflates the convolutions to 3D, only some of the features contain temporal information. However, by using separable temporal convolution, we can add temporal infor-

mation to all 4 branches. This improves the performance from 78.42% to 78.88% on Mini-Kinetics-200. We call this a "temporal inception block". In the following sections, whenever we refer to an S3D model, we mean S3D with a temporal inception block.

5. Spatiotemporal feature gating

In this section we further improve the accuracy of our model by weighting the features in each channel in every layer in an adaptive, data-dependent way. In contrast to prior work, we combine information across space and time in a novel way, and show that this significantly improves the accuracy.

Let $\mathbf{X}_{twh} \in \mathcal{R}^D$ be a feature vector at time frame t and spatial coordinate (w, h) at some layer of the network, and let $\mathbf{X}$ be the full tensor of features. We propose to replace the original features by a weighted version:

$$\mathbf{X}'_{twh} = \mathbf{A} \otimes \mathbf{X}_{twh} \qquad (1)$$

where $\otimes$ is elementwise multiplication, and $\mathbf{A} \in \mathcal{R}^D$ is an adaptive weight vector computed as follows:

$$\mathbf{A} = \sigma(\mathbf{W} \operatorname{pool}_{ST}(\mathbf{X})) \qquad (2)$$

where $\mathbf{W} \in \mathcal{R}^{D \times D}$ is a learnable weight matrix, $\sigma()$ is the sigmoid (logistic) function, and $pool_{ST} : \mathcal{R}^{T \times W \times H \times D} \rightarrow \mathcal{R}^D$ is an average pooling function that pools the input feature over space and time. The shape of $\mathbf{A}$ depends on the nature of the pooling function (pool), as we discuss below.

Feature gating captures dependencies between feature channels with a simple but effective multiplicative transformation. This can be viewed as an efficient approximation to second-order pooling as shown in [12]. Similar gated features have been used for other tasks, such as machine translation[5], VQA[32], reinforcement learning[7],

classification[35, 17], and action recognition[28]. Our formulation is different from previous works in the way we combine information across space and time when computing the attention mask, and in where we apply this technique. In particular, the context gating method proposed in [28] (which achieved first place in the Youtube-8M video classification challenge of 2017) only applies gating to the features on the output layer. In contrast, we place the feature gating module after each of the $[k, 1, 1]$ temporal convolutions in an S3D network.

When computing the attention map, we consider several variants of the pooling function: pooling over space and time using $pool_{ST} : \mathcal{R}^{T \times W \times H \times D} \rightarrow \mathcal{R}^D$; pooling over space using $pool_S : \mathcal{R}^{T \times W \times H \times D} \rightarrow \mathcal{R}^{T,D}$; pooling over time $pool_T : \mathcal{R}^{T \times W \times H \times D} \rightarrow \mathcal{R}^{W,H,D}$; or no pooling. Table 3 shows the performance of these variants on Mini-Kinetics-200. We see that pooling over time is more important than pooling over space, and that pooling over both gives the best results. (Note that gating without any form of pooling works worse than using no gating, presumably because of overfitting.)

Gating	Pooling	Accuracy
Y	**Space-time**	**79.88**
Y	Time	79.30
N	NA	78.88
Y	Space	78.20
Y	None	77.98

Table 3. Top-1 accuracy on Mini-Kinetics-200 for S3D models with different feature gating mechanisms. The "pooling" column describes whether and how features are combined across space and time before being passed as input to the gate. The row without gating corresponds to our baseline S3D model.

Based on the above results, our final model is an S3D model with feature gating with spatio-temporal pooling; we call this model S3D-G. On the full Kinetics dataset, we achieved 74.65% top-1 accuracy, which is a new record for RGB-only methods. Furthermore, our model uses 33% fewer FLOPS than the previous state of the art, I3D. See Table 4 for details.

Model	Top-1 (%)	Top-5 (%)	Params (M)	FLOPS (G)
I3D	71.66	90.21	12.06	107.89
S3D-G	**74.84**	**91.93**	11.56	71.38

Table 4. Results on Full-Kinetics validation set. We just use RGB inputs (no optical flow). S3D-G is our spatio-temporal-separable version of I3D with feature gating.

6. Performance of S3D-G on other video tasks

Finally, we evaluate the generality and robustness of the proposed S3D-G architecture by conducting transfer learning experiments on other input modalities, other video datasets and other tasks.

6.1. Training with optical flow features

We first verify if S3D-G also works with optical flow inputs. For these experiments, we follow the standard setup as described in [3] and extract optical flow features with the TV-L1 approach [57]. We truncate the flow magnitude at $[-20, 20]$ and store them as encoded JPEG files. Other experiment settings are the same as the RGB experiments. From Table 5, we can see that the improvement of S3D-G over I3D is consistent with the gains we saw with RGB inputs, bringing the performance up from 63.40% to 67.25%. By ensembling the two streams of RGB and flow, we obtain a performance of 77.16%, which is a 3% boost over the previous state-of-the-art.

Model	Top-1 (%)	Top-5 (%)
Flow-I3D [3]	63.91	85.02
Flow-S3D-G	**68.00**	**87.61**

Table 5. Results on the Full-Kinetics validation set with optical flow inputs.

6.2. Fine-tuning for action classification

Next we conduct transfer learning experiments from Kinetics to other video classification datasets, namely HMDB-51 [26] and UCF-101 [43].. HMDB-51 contains around 7,000 videos spanning over 51 categories, while UCF-101 has 13,320 videos spanning over 101 categories. Both datasets consist of short video clips that are temporally trimmed, and contain 3 training and validation splits. We follow the standard setup as used in previous work and report average accuracy across all splits.

For our transfer learning experiments, we use the same setup as training on Kinetics, but change the number of GPUs to 8 and lower the learning rate to 0.01 for 6K steps, and finally decay to 0.001 for another 2K steps. For simplicity, we only use RGB (no optical flow).

Table 7 shows the results of this experiment on both datasets. Our proposed S3D-G architecture clearly outperforms I3D when both are pre-trained on the Kinetics dataset. And both methods significantly outperform other recent works.

6.3. Fine-tuning for action detection

Finally, we demonstrate the effectiveness of S3D-G on action detection tasks, where the inputs are video frames,

Model	Inputs	Backbone	Top-1 (%)	Top-5 (%)
Shifting Attention Net [1]	RGB+Flow+Audio	Inception-ResNet-v2	77.7	93.2
Temporal Segment Net [53]	RGB+Flow	Inception	73.9	91.1
ARTNet w/ TSN [50]	RGB+Flow	ResNet-18	72.4	90.4
I3D [3]	RGB+Flow	Inception	74.1	91.6
S3D-G	RGB+Flow	Inception	**77.2**	**93.0**

Table 6. Comparing results on the Full-Kinetics validation set with optical flow inputs. We report I3D performance based on our implementation, as [3] report results on the held-out test set.

Model	Pre-train	Flow	UCF-101	HMDB-51
IDT [49]	N/A	✓	86.4	61.7
Two Stream [42]	ImageNet	✓	88.0	59.4
TDD + IDT [51]	ImageNet	✓	91.5	65.9
TSN [53]	ImageNet	✓	94.2	69.4
C3D [46]	Sports-1M		82.3	51.6
Res3D [47]	Sports-1M		85.8	54.9
I3D [3]	ImNet+Kinetics		95.6	74.8
S3D-G	ImNet+Kinetics		**96.8**	**75.9**

Table 7. Action classification comparisons between S3D-G, I3D and other state-of-the-art methods on UCF-101 and HMDB-51 datasets. All numbers are computed as the average accuracy across three splits.

and the outputs are bounding boxes associated with action labels on the frames. Similar to the framework proposed by Peng and Schmid [31], we use the Faster-RCNN [36] object detection algorithm to jointly perform actor localization and action recognition. We use the same approach as described in [14] to incorporate temporal context information via 3D neural networks. To be more specific, the model uses a 2D ResNet-50 [15] network that takes the annotated keyframe as input, and extract features for region proposal generation on the keyframe. We then use a 3D network (such as I3D or S3D-G) that takes the frames surrounding the keyframe as input, and extract feature maps which are then pooled for bounding box classification. The 2D region proposal network (RPN) and 3D action classification network are jointly trained end-to-end. Note that we extend the ROIPooling operation to handle 3D feature maps by simply pooling at the same spatial locations over all time steps.

We report performance on two widely adopted video action detection datasets: JHDMB [21] and UCF-101-24 [43]. JHMDB dataset is a subset of HMDB-51, it consists of 928 videos for 21 action categories, and each video clip contains 15 to 40 frames. UCF-101-24 is a subset of UCF-101 with 24 labels and 3207 videos; we use the cleaned bounding box annotations from [39]. We report performance using the standard frame-AP metric defined in [13], which is computed as the average precision of action detection over all individual frames, at the intersection-over-union (IoU)

threshold of 0.5. As commonly used by previous work, we report average performance over three splits of JHMDB and the first split for UCF-101-24.

Our implementation is based on the TensorFlow Object Detection API [18]. We train Faster-RCNN with asynchronous SGD on 11 GPUs for 600K iterations. We fix the input resolution to be 320 by 400 pixels. For both training and validation, we fix the size of temporal context to be 20 frames which gives the best performance. All the other model parameters are set based on the recommended values from [18], which were tuned for object detection. The ResNet-50 networks are initialized with ImageNet pre-trained models, and the I3D and S3D-G networks are pre-trained from Kinetics. We extract 3D feature maps at the so-called "*Mixed 4e*" layer which has a stride of 16 pixels on the input image.

Table 8 shows the comparison between I3D, S3D-G, and other state-of-the-art methods. We can see that both 3D networks outperform previous architectures by large margins, while S3D-G is consistently better than I3D, especially on UCF-101-24.

Model	Flow	JHMDB	UCF-101
Gkioxari and Malik [13]	✓	36.2	-
Weinzaepfel *et al.* [55]	✓	45.8	35.8
Peng and Schmid [31]	✓	58.5	65.7
Kalogeiton *et al.* [22]	✓	65.7	69.5
I3D		70.5	76.7
S3D-G		**72.1**	**80.1**

Table 8. Action detection performance comparison between I3D, S3D-G and other state-of-the-art methods. We report frame-mAP at IoU threshold of 0.5 on JHMDB (all splits) and UCF-101-24 (split 1) datasets.

7. Conclusion

We have shown that it is possible to significantly improve on the previous state of the art 3D CNN model, known as I3D, in terms of efficiency, by using temporally separable convolution. We can also improve the accuracy by using spatio-temporal feature gating. Our modifications are sim-

ple and can be applied to other 3D architectures. We believe this will boost performance on a variety of video understanding tasks.

References

[1] Y. Bian, C. Gan, X. Liu, F. Li, X. Long, Y. Li, H. Qi, J. Zhou, S. Wen, and Y. Lin. Revisiting the effectiveness of off-the-shelf temporal modeling approaches for large-scale video classification. *arXiv preprint arXiv:1708.03805*, 2017. 8

[2] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. In *ICCV*, 2005. 2

[3] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. *CVPR*, 2017. 1, 2, 3, 7, 8

[4] F. Chollet. Xception: Deep learning with depthwise separable convolutions. 2017. 5

[5] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. *ICML*, 2017. 6

[6] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 2

[7] S. Elfwing, E. Uchibe, and K. Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *arXiv preprint arXiv:1702.03118*, 2017. 6

[8] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal multiplier networks for video action recognition. In *CVPR*, 2017. 2

[9] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Spatiotemporal residual networks for video action recognition. In *NIPS*, 2016. 2

[10] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Temporal residual networks for dynamic scene recognition. In *CVPR*, 2017. 2

[11] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016. 2

[12] R. Girdhar and D. Ramanan. Attentional pooling for action recognition. In *NIPS*, 2017. 2, 6

[13] G. Gkioxari and J. Malik. Finding action tubes. In *CVPR*, 2015. 8

[14] C. Gu, C. Sun, S. Vijayanarasimhan, C. Pantofaru, D. A. Ross, G. Toderici, Y. Li, S. Ricco, R. Sukthankar, C. Schmid, and J. Malik. AVA: A video dataset of spatio-temporally localized atomic visual actions. *arXiv preprint arXiv:1705.08421*, 2017. 8

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 8

[16] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 5

[17] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 2017. 7

[18] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. *CVPR*, 2017. 8

[19] H. Idrees, A. R. Zamir, Y. Jiang, A. Gorban, I. Laptev, R. Sukthankar, and M. Shah. The THUMOS challenge on action recognition for videos "in the wild". *CVIU*, 2017. 2

[20] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. *arXiv:1612.01925*, 2016. 2

[21] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. Black. Towards understanding action recognition. In *ICCV*, 2013. 8

[22] V. Kalogeiton, P. Weinzaepfel, V. Ferrari, and C. Schmid. Action Tubelet Detector for Spatio-Temporal Action Localization. In *ICCV*, 2017. 8

[23] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 2

[24] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *CVPR*, 2017. 1, 2, 9

[25] A. Klaser, M. Marszalek, and C. Schmid. A Spatio-Temporal Descriptor Based on 3D-Gradients. In *BMVC*, 2008. 2

[26] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: A large video database for human motion recognition. In *ICCV*, 2011. 7

[27] I. Laptev and T. Lindeberg. Space-time interest points. In *ICCV*, 2003. 2

[28] A. Miech, I. Laptev, and J. Sivic. Learnable pooling with context gating for video classification. *arXiv preprint arXiv:1706.06905*, 2017. 7

[29] J. Y. Ng, J. Choi, J. Neumann, and L. S. Davis. Actionflownet: Learning motion representation for action recognition. *arXiv preprint arXiv:1612.03052*, 2016. 2

[30] J. Y. Ng, M. J. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015. 2

[31] X. Peng and C. Schmid. Multi-region two-stream r-cnn for action detection. In *ECCV*, 2016. 8

[32] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. *arXiv preprint arXiv:1709.07871*, 2017. 6

[33] L. C. Pickup, Z. Pan, D. Wei, Y. Shih, C. Zhang, A. Zisserman, B. Scholkopf, and W. T. Freeman. Seeing the arrow of time. In *CVPR*, 2014. 4

[34] Z. Qiu, T. Yao, and T. Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *ICCV*, 2017. 2

[35] P. Ramachandran, B. Zoph, and Q. V. Le. Swish: a self-gated activation function. *arXiv preprint arXiv:1710.05941*, 2017. 7

[36] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 8

[37] M. Rodriguez, J. Ahmed, and M. Shah. Action MACH: a spatio-temporal maximum average correlation height filter for action recognition. In *CVPR*, 2008. 2

[38] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015. 1

[39] S. Saha, G.Sing, and F. Cuzzolin. AMTnet: Action-microtube regression by end-to-end trainable deep architecture. In *ICCV*, 2017. 8

[40] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: a local SVM approach. In *ICPR*, 2004. 2

[41] G. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *ECCV*, 2016. 2

[42] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 2, 8

[43] K. Soomro, A. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. Technical Report CRCV-TR-12-01, 2012. 2, 7, 8

[44] C. Sun and R. Nevatia. Large-scale web video event classification by use of fisher vectors. In *WACV*, 2013. 2

[45] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 1, 2

[46] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. C3D: generic features for video analysis. *arXiv preprint arXiv:1412.0767*, 2014. 2, 8

[47] D. Tran, J. Ray, Z. Shou, S. Chang, and M. Paluri. Convnet architecture search for spatiotemporal feature learning. *arXiv preprint arXiv:1708.05038*, 2017. 2, 4, 8

[48] H. Wang, D. Oneata, J. Verbeek, and C. Schmid. A robust and efficient video representation for action recognition. *Intl. J. Computer Vision*, 2015. 2

[49] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 2, 8

[50] L. Wang, W. Li, W. Li, and L. V. Gool. Appearance-and-relation networks for video classification. *arXiv preprint arXiv:1711.09125*, 2017. 8

[51] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015. 8

[52] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards good practices for very deep two-stream convnets. *arXiv preprint arXiv:1507.02159*, 2015. 2

[53] 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 *ECCV*, 2016. 1, 2, 8

[54] X. Wang, A. Farhadi, and A. Gupta. Actions ˜ transformations. In *CVPR*, 2016. 2

[55] P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Learning to track for spatio-temporal action localization. In *ICCV*, 2015. 8

[56] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *CVPR*, 2017. 5

[57] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l1 optical flow. *Pattern Recognition,*, 2007. 7

[58] M. Zolfaghari, G. L. Oliveira, N. Sedaghat, and T. Brox. Chained multi-stream networks exploiting pose, motion, and appearance for action classification and detection. *arXiv preprint arXiv:1704.00616*, 2017. 2