X3D: Expanding Architectures for Fast Video Recognition

Christoph Feichtenhofer

Facebook AI Research (FAIR)

Abstract

This paper presents X3D, a family of efficient video networks that progressively expand a tiny 2D image classification architecture along multiple network axes, in space, time, width and depth. Inspired by feature selection methods in machine learning, a simple stepwise network expansion approach is employed that expands a single axis in each step, such that good accuracy to complexity trade-off is achieved. To expand X3D to a specific target complexity, we perform progressive forward expansion followed by backward contraction. X3D achieves state-of-the-art performance while requiring 4.8× and 5.5× fewer multiply-adds and parameters for similar accuracy as previous work. Our most surprising finding is that networks with high spatiotemporal resolution can perform well, while being extremely light in terms of network width and parameters. We report competitive accuracy at unprecedented efficiency on video classification and detection benchmarks.

1. Introduction

Neural networks for video recognition have been largely driven by *expanding* 2D image architectures [29, 47, 64, 71] into spacetime. Naturally, these expansions often happen along the temporal axis, involving extending the network inputs, features, and/or filter kernels into spacetime (*e.g.* [7, 13, 17, 42, 56, 75]); other design decisions—including depth (number of layers), width (number of channels), and spatial sizes—however, are typically inherited from 2D image architectures. While expanding along the temporal axis (while keeping other design properties) generally increases accuracy, it can be sub-optimal if one takes into account the computation/accuracy *trade-off*—a consideration of central importance in applications.

In part because of the direct extension of 2D models to 3D, video recognition architectures are computationally heavy. In comparison to image recognition, typical video models are significantly more compute-demanding, *e.g.* an image ResNet [29] can use around 27× fewer multiply-add operations than a temporally extended video variant [81].

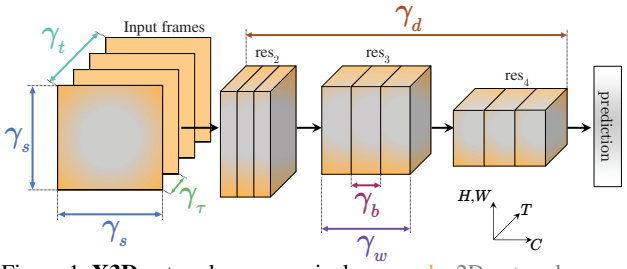

Figure 1. **X3D** networks progressively expand a 2D network across the following axes: Temporal duration γ_t, frame rate γ_τ, spatial resolution γ_s, width γ_w, bottleneck width γ_b, and depth γ_d.

This paper focuses on the low-computation regime in terms of computation/accuracy trade-off for video recognition. We base our design upon the "mobile-regime" models [31, 32, 61] developed for image recognition. Our core idea is that while expanding a small model along the temporal axis can increase accuracy, the computation/accuracy trade-off may not always be best compared with expanding *other axes*, especially in the low-computation regime where accuracy can increase quickly along different axes.

In this paper, we progressively "expand" a tiny base 2D image architecture into a spatiotemporal one by expanding *multiple* possible axes shown in Fig. 1. The candidate axes are temporal duration γ_t, frame rate γ_τ, spatial resolution γ_s, network width γ_w, bottleneck width γ_b, and depth γ_d. The resulting architecture is referred as **X3D** (Expand 3D) for expanding from the 2D space into 3D spacetime domain.

The 2D base architecture is driven by the MobileNet [31, 32, 61] core *concept* of channel-wise[1] separable convolutions, but is made tiny by having over 10× fewer multiply-add operations than mobile image models. Our expansion then progressively increases the computation (*e.g.*, by 2×) by expanding only *one axis at a time*, train and validate the resultant architecture, and select the axis that achieves the best computation/accuracy trade-off. The process is repeated until the architecture reaches a desired computational budget. This can be interpreted as a form of *coordinate descent* [83] in the hyper-parameter space defined by those axes.

[1] Also referred as "depth-wise". We use the term "channel-wise" to avoid confusions with the network depth, which is also an axis we consider.

Our progressive network expansion approach is inspired by the history of image ConvNet design where popular architectures have arisen by expansions across depth, [8, 29, 47, 64, 71, 94], resolution [35, 70, 73] or width [88, 93], and classical feature selection methods [25, 41, 44] in machine learning. In the latter, progressive feature selection methods [25, 44] start with either a set of minimum features and aim to find relevant features to improve in a greedy fashion by including (*forward selection*) a single feature in each step, or start with a full set of features and aim to find irrelevant ones that are excluded by repeatedly deleting the feature that reduces performance the least (*backward elimination*).

We evaluate our method on Kinetics-400 [43], Kinetics-600 [4], Charades [62] and AVA [24]. To allow systematic future comparison, we classify our models into different levels of complexity for small, medium and large models.

Overall, our expansion produces a sequence of spatiotemporal architectures, covering a wide range of computation/accuracy trade-offs. They can be used under different computational budgets that are application-dependent in practice. For example, across different computation and accuracy regimes X3D performs favorably to state-of-the-art while requiring 4.8× and 5.5× fewer multiply-adds and parameters for similar accuracy as previous work. Further, expansion is simple and cheap *e.g.* our low-compute model is completed after only training 30 *tiny* models that *accumulatively* require over 25× fewer multiply-add operations for training than one large state-of-the-art network [15, 81, 84].

Conceptually, our most surprising finding is that *very thin* video models that are created by only expanding spatiotemporal resolution and depth can perform well, while being extremely light in terms of network width and parameters. X3D networks have a significantly lower width than image-design [29, 64, 71] based video architectures. We hope these advances will facilitate future research and applications.

Code and models will be made publicly available.

2. Related Work

Spatiotemporal (3D) networks. Video recognition architectures are favorably designed by extending image classification networks with a temporal dimension, and preserving the spatial properties. These extensions include direct transformation of 2D models [29, 47, 64, 71] such as ResNet or Inception to 3D [7, 26, 58, 74, 75, 89], adding RNNs on top of 2D CNNs [13, 49, 50, 56, 67, 92], or extending 2D models with an optical flow stream that is processed by an identical 2D network [7, 18, 63, 80] . While starting with a 2D image based model and converting it to a spatiotemporal equivalent by inflating filters [7, 17] allows pretraining on image classification tasks, it makes video architectures inherently biased towards their image-based counterparts.

The SlowFast [15] architecture has explored the resolution trade-off across several axes, different temporal, spatial, and channel resolution in the Slow and Fast pathway. Interestingly the Fast pathway can be very thin and therefore only adds a small computational overhead; however, performs low in isolation. Further, these explorations were performed with the architecture of the computationally heavy Slow pathway held constant to a temporal extension of an image classification design [29]. In relation to this previous effort, our work investigates whether the heavy Slow pathway is required, or if a lightweight network can be made competitive.

Efficient 2D networks. Computation-efficient architectures have been extensively developed for the image classification task, with MobileNetV1&2 [32, 61] and ShuffleNet [95] exploring channel-wise separable convolutions and expanded bottlenecks. Several methods for neural architecture search in this setting have been proposed, also adding Squeeze-Excitation (SE) [33] attention blocks to the design space in [72] and more recently, MobileNetV3 [31] Swish non-linearities [59]. MobileNets [32, 61, 72] were scaled up and down by using a multiplier for width and input size (resolution). Recently, MnasNet [72] is used to apply liner scaling factors to spatial, width and depth axes for creating a set of EfficientNets [73] for image classification.

Our expansion is related to this, but requires fewer samples and handles more axes as we only train a single model for each axis in each step, while [73] performs a grid-search on the initial regime which requires k^d models to be trained where k is the gridsize and d the number of axes. Moreover, the model used for this search, MnasNet was found by sampling around 8000 models [72]. For video, this is prohibitive as datasets can have orders of magnitude more images than image classification *e.g.* the largest version of Kinetics [5] has ≈195M frames, 162.5× more images than ImageNet. By contrast, our approach only requires to train 6 models, one for each expansion axis, until a desired complexity is reached, *e.g.* for 5 steps, it requires 30 models to be trained.

Efficient 3D networks. Several innovative architectures for efficient video classification have been proposed, *e.g.* [3, 6, 10, 12, 14, 19, 36, 45, 48, 55, 57, 68, 69, 76, 78, 79, 85, 89, 97–99]. Channel-wise separable convolution as a key building block for efficient 2D ConvNets [31, 32, 61, 73, 95] has been explored for video classification in [45, 76], where 2D architectures are extended to their 3D counterparts, *e.g.* ShuffleNet and MobileNet in [45], or ResNet in [76] by using a 3×3×3 channel-wise separable convolution in the bottleneck of a residual stage. Earlier, [10] adopt 2D ResNets and MobileNets from ImageNet and sparsifies connections inside each residual block similar to separable or group convolution. A temporal shift module (TSM) is introduced in [51] that extends a ResNet to capture temporal information using memory shifting operations. There is also active research on adaptive frame sampling techniques, *e.g.* [2, 46, 65, 86, 87, 91], which we think can be complementary to our approach.

In relation to most of these works, our approach does not assume a fixed inherited design from 2D networks, but expands a tiny architecture across several axes in space, time, channels and depth to achieve a good efficiency trade-off.

3. X3D Networks

Image classification architectures have gone through an evolution of architecture design with progressively *expanding* existing models along network depth [8,29,47,64,71,94], input resolution [35,70,73] or channel width [88,93]. Similar progress can be observed for the mobile image classification domain where *contracting* modifications (shallower networks, lower resolution, thinner layers, separable convolution [31,32,37,61,95]) allowed operating at lower computational budget. Given this history in image ConvNet design, a similar progress has not been observed for video architectures as these were customarily based on direct temporal extensions of image models. However, is single expansion of a *fixed* 2D architecture to 3D ideal, or is it better to *expand* or *contract* along different axes?

For video classification the temporal dimension exposes an additional dilemma, increasing the number of possibilities but also requiring it to be dealt differently than the spatial dimensions [15,63,77]. We are especially interested in the trade-off between different axes, more concretely:

- What is the best temporal sampling strategy for 3D networks? Is a *long* input duration and sparser sampling preferred over *faster* sampling of short duration clips?
- Do we require *finer* spatial resolution? Previous works have used lower resolution for video classification [42, 75, 77] to increase efficiency. Also, videos typically come at *coarser* spatial resolution than Internet images; therefore, is there a maximum spatial resolution at which performance saturates?
- Is it better to have a network with high frame-rate but *thinner* channel resolution, or to slowly process video with a *wider* model? *E.g.* should the network have heavier layers as typical image classification models (and the Slow pathway [15]) or rather lighter layers with lower width (as the Fast pathway [15]). Or is there a better trade-off, possibly between these extremes?
- When increasing the network width, is it better to globally expand the network width in the ResNet block design [29] or to expand the inner ("*bottleneck*") width, as is common in mobile image classification networks using channel-wise separable convolutions [61,95]?
- Should going *deeper* be performed with expanding input resolution in order to keep the receptive field size large enough and its growth rate roughly constant, or is it better to expand into different axes? Does this hold for both the spatial and temporal dimension?

stage	filters	output sizes $T\times S^2$
data layer	stride γ_τ, 1^2	$1\gamma_t\times(112\gamma_s)^2$
conv_1	1×3^2, 3×1, $24\gamma_w$	$1\gamma_t\times(56\gamma_s)^2$
res_2	$\begin{bmatrix}1\times1^2, 24\gamma_b\gamma_w\\ 3\times3^2, 24\gamma_b\gamma_w\\ 1\times1^2, 24\gamma_w\end{bmatrix}\times\gamma_d$	$1\gamma_t\times(28\gamma_s)^2$
res_3	$\begin{bmatrix}1\times1^2, 48\gamma_b\gamma_w\\ 3\times3^2, 48\gamma_b\gamma_w\\ 1\times1^2, 48\gamma_w\end{bmatrix}\times2\gamma_d$	$1\gamma_t\times(14\gamma_s)^2$
res_4	$\begin{bmatrix}1\times1^2, 96\gamma_b\gamma_w\\ 3\times3^2, 96\gamma_b\gamma_w\\ 1\times1^2, 96\gamma_w\end{bmatrix}\times5\gamma_d$	$1\gamma_t\times(7\gamma_s)^2$
res_5	$\begin{bmatrix}1\times1^2, 192\gamma_b\gamma_w\\ 3\times3^2, 192\gamma_b\gamma_w\\ 1\times1^2, 192\gamma_w\end{bmatrix}\times3\gamma_d$	$1\gamma_t\times(4\gamma_s)^2$
conv_5	1×1^2, $192\gamma_b\gamma_w$	$1\gamma_t\times(4\gamma_s)^2$
pool_5	$1\gamma_t\times(4\gamma_s)^2$	$1\times1\times1$
fc_1	1×1^2, 2048	$1\times1\times1$
fc_2	1×1^2, #classes	$1\times1\times1$

Table 1. **X3D architecture**. The dimensions of kernels are denoted by $\{T\times S^2, C\}$ for temporal, spatial, and channel sizes. Strides are denoted as $\{$temporal stride, spatial stride$^2\}$. This network is expanded using factors $\{\gamma_\tau, \gamma_t, \gamma_s, \gamma_w, \gamma_b, \gamma_d\}$ to form **X3D**. Without expansion (all factors equal to one), this model is referred to as **X2D**, having 20.67M FLOPS and 1.63M parameters.

This section first introduces the basis X2D architecture in Sec. 3.1 which is expanded with operations defined in Sec. 3.2 by using the progressive approach in Sec. 3.3.

3.1. Basis instantiation

We begin by describing the instantiation of the basis network architecture, X2D, that serves as baseline to be expanded into spacetime. The basis network instantiation follows a ResNet [29] structure and the Fast pathway design of SlowFast networks [15] with degenerated (single frame) temporal input. X2D is specified in Table 1, if all expansion factors $\{\gamma_\tau, \gamma_t, \gamma_s, \gamma_w, \gamma_b, \gamma_d\}$ are set to 1.

We denote spatiotemporal size by $T\times S^2$ where T is the temporal length and S is the height and width of a square spatial crop. The X2D architecture is described next.

Network resolution and channel capacity. The model takes as input a raw video clip that is sampled with frame-rate $1/\gamma_\tau$ in the data layer stage. The basis architecture only takes a single frame of size $T\times S^2{=}1\times112^2$ as input and therefore can be seen as an image classification network. The width of the individual layers is oriented at the Fast pathway design in [15] with the first stage, conv_1, filters the 3 RGB input channels and produces 24 output features. This width is increased by a factor of 2 after every spatial sub-sampling with a stride $= 1, 2^2$ at each deeper stage from res_2 to res_5. Spatial sub-sampling is performed by the center ("bottleneck") filter of the first res-block of each stage.

Similar to the SlowFast pathways [15], the model preserves the temporal input resolution for all features throughout the network hierarchy. There is no temporal downsampling layer (neither temporal pooling nor time-strided convolutions) throughout the network, up to the global pooling layer before classification. Thus, the activations tensors contain all frames along the temporal dimension, maintaining full temporal frequency in all features.

Network stages. X2D consists of a stage-level and bottleneck design that is inspired by recent 2D mobile image classification networks [31, 32, 61, 95] which employ channel-wise separable convolution that are a key building block for efficient ConvNet models. We adopt stages that follow MobileNet [31, 61] design by extending every spatial 3×3 convolution in the bottleneck block to a 3×3×3 (*i.e.* 3×3^2) spatiotemporal convolution which has also been explored for video classification in [45, 76]. Further, the 3×1 temporal convolution in the first conv_1 stage is channel-wise.

Discussion. X2D can be interpreted as a Slow pathway since it only uses a single frame as input, while the network width is similar to the Fast pathway in [15] which is much lighter than typical 3D ConvNets (*e.g.*, [7, 15, 17, 75, 81]) that follow an ImageNet design. Concretely, it only requires 20.67M FLOPs which amounts to only 0.0097% of a recent state-of-the-art SlowFast network [15].

As shown in Table 1 and Fig. 1, **X2D** is expanded across 6 axes, $\{\gamma_\tau, \gamma_t, \gamma_s, \gamma_w, \gamma_b, \gamma_d\}$, described next.

3.2. Expansion operations

We define a basic set of expansion operations that are used for sequentially expanding **X2D** from a tiny spatial network to **X3D**, a spatiotemporal network, by performing the following operations on temporal, spatial, width and depth dimensions.

- ***X-Fast*** expands the temporal activation size, γ_t, by increasing the frame-rate, $1/\gamma_\tau$, and therefore temporal resolution, while holding the clip duration constant.
- ***X-Temporal*** expands the temporal size, γ_t, by sampling a longer temporal clip and increasing the frame-rate $1/\gamma_\tau$, to expand both duration *and* temporal resolution.
- ***X-Spatial*** expands the spatial resolution, γ_s, by increasing the spatial sampling resolution of the input video.
- ***X-Depth*** expands the depth of the network by increasing the number of layers per residual stage by γ_d times.
- ***X-Width*** uniformly expands the channel number for all layers by a global width expansion factor γ_w.
- ***X-Bottleneck*** expands the inner channel width, γ_b, of the center convolutional filter in each residual block.

3.3. Progressive Network Expansion

We employ a simple progressive algorithm for network expansion, similar to forward and backward algorithms for feature selection [25, 39, 41, 44]. Initially we start with X2D, the basis model instantiation with a set of unit expanding factors $\mathcal{X}_0$ of cardinality a. We use $a = 6$ factors, $\mathcal{X} = \{\gamma_\tau, \gamma_t, \gamma_s, \gamma_w, \gamma_b, \gamma_d\}$, but other axes are possible.

Forward expansion. The network expansion criterion function, which measures the goodness for the current expansion factors $\mathcal{X}$, is represented as $J(\mathcal{X})$. Higher scores of this measure represent better expanding factors, while lower scores would represent worse. In our experiments, this corresponds to the accuracy of a model expanded by $\mathcal{X}$. Furthermore, let $C(\mathcal{X})$ be a complexity criterion function that measures the cost of the current expanding factors $\mathcal{X}$. In our experiments, C is set to the floating point operations of the underlying network instantiation expanded by $\mathcal{X}$, but other measures such as runtime, parameters, or memory are possible. Then, the network expansion tries to find expansion factors $\mathcal{X}$ with the best trade-off $\mathcal{X} = \text{argmax}_{\mathcal{Z}, C(\mathcal{Z})=c} J(\mathcal{Z})$ where $\mathcal{Z}$ are the possible expansion factors to be explored and c is the target complexity. In our case we perform expansion that only changes a *single one* of the a expansion factors while holding the others constant; therefore there are only a different subsets of $\mathcal{Z}$ to evaluate, where each of them alters in only one dimension from $\mathcal{X}$. The expansion with the best computation/accuracy trade-off is kept for the next step. This is a form of *coordinate descent* [83] in the hyper-parameter space defined by those axes.

The expansion is performed in a progressive manner with an expansion-rate $\hat{c}$ that corresponds to the stepsize at which the model complexity c is increased in each expansion step. We use a multiplicative increase of $\hat{c} \approx 2$ of the model complexity in each step that corresponds to the complexity-increase for doubling the number of frames of the model. The stepwise expansion is therefore simple and efficient as it only requires to train a few models until a target complexity is reached, since we *exponentially* increase the complexity. Details on the expansion are in §A.3.

Backward contraction. Since the forward expansion only produces models in discrete steps, we perform a backward contraction step to meet a desired target complexity, if the target is exceeded by the forward expansion steps. This contraction is implemented as a simple reduction of the last expansion, such that it matches the target. For example, if the last step has increased the frame-rate by a factor of two, the backward contraction will reduce the frame-rate by a factor < 2 to roughly match the desired target complexity.

4. Experiments: Action Classification

Datasets. We perform our expansion on Kinetics-400 [43] (K400) with ~240k training, 20k validation and 35k testing

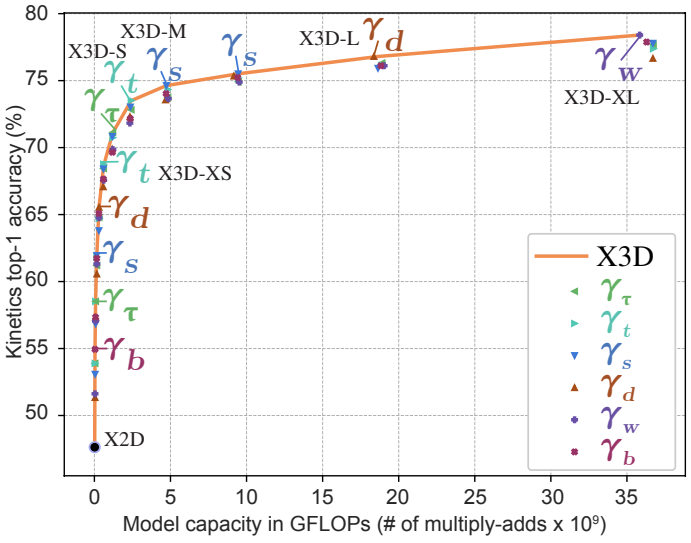

Figure 2. **Progressive network expansion of X3D.** The X2D base model is expanded 1st across bottleneck width (γ_b), 2nd temporal resolution (γ_τ), 3rd spatial resolution (γ_s), 4th depth (γ_d), 5th duration (γ_t), *etc*. The majority of models are trained for small computation cost, making the expansion economical in practice.

videos in 400 human action categories. We report top-1 and top-5 classification accuracy (%). As in previous work, we train and report ablations on the train and val sets. We also report results on test set as the labels have been made available [4]. We report the computational cost (in FLOPs) of a single, spatially center-cropped clip.[2]

Training. All models are trained *from random initialization* ("*from scratch*") on Kinetics, *without* using ImageNet [11] or other pre-training. Our training recipe follows [15]. All implementation details and dataset specifics are in §A.3.

For the temporal domain, we randomly sample a clip from the full-length video, and the input to the network are γ_t frames with a temporal stride of γ_τ; for the spatial domain, we randomly crop $112\gamma_s \times 112\gamma_s$ pixels from a video, or its horizontal flip, with a shorter side randomly sampled in $[128\gamma_s, 160\gamma_s]$ pixels which is a linearly scaled version of the augmentation used in [15, 64, 81].

Inference. To be comparable with previous work and evaluate accuracy/complexity trade-offs we apply two testing strategies: *(i) K-Center*: Temporally, uniformly samples K clips (*e.g.* K=10) from a video and spatially scales the shorter spatial side to $128\gamma_s$ pixels and takes a $\gamma_t \times 112\gamma_s \times 112\gamma_s$ center crop, comparable to [46, 51, 76, 84]. *(ii) K-LeftCenterRight* is the same as above temporally, but takes 3 crops of $\gamma_t \times 128\gamma_s \times 128\gamma_s$ to cover the longer spatial axis, as an approximation of fully-convolutional testing, following [15, 81]. We average the softmax scores for all individual predictions.

[2]We use single-clip, center-crop FLOPs as a basic *unit* of computational cost. Inference-time computational cost is roughly proportional to this, *if* a fixed number of clips and crops is used, as is for our all models.

model	top-1	top-5	regime FLOPs (G)	FLOPs (G)	Params (M)
X3D-XS	68.6	87.9	*X-Small* ≤ 0.6	0.60	3.76
X3D-S	72.9	90.5	*Small* ≤ 2	1.96	3.76
X3D-M	74.6	91.7	*Medium* ≤ 5	4.73	3.76
X3D-L	76.8	92.5	*Large* ≤ 20	18.37	6.08
X3D-XL	78.4	93.6	*X-Large* ≤ 40	35.84	11.0

Table 2. Expanded instances on K400-val. 10-Center clip testing is used. We show top-1 and top-5 classification accuracy (%), as well as computational complexity measured in GFLOPs (floating-point operations, in # of multiply-adds $\times 10^9$) for a single clip input. Inference-time computational cost is proportional to 10× of this, as a fixed number of 10 of clips is used per video.

4.1. Expanded networks

The accuracy/complexity trade-off curve for the expansion process on K400 is shown in Fig. 2. Expansion starts from X2D that produces 47.75% top-1 accuracy (vertical axis) with 1.63M parameters 20.67M FLOPs per clip (horizontal axis), which is roughly doubled in each progressive step. We use *10-Center* clip testing as our default test setting for expansion, so the overall cost per video is ×10. We will ablate different number of testing clips in Sec. 4.3. The expansion in Fig. 2 provides several interesting observations:

(i) First of all, expanding along *any* one of the candidate axes increases accuracy. This justifies our motivation of taking multiple axes (instead of just the temporal axis) into account when designing spatiotemporal models.

(ii) Surprisingly, the first step selected by the expansion algorithm is *not* along the temporal axis; instead, it is a factor that grows the "bottleneck" width γ_b in the ResNet block design [29]. This echoes the inverted bottleneck design in [61] (called "inverted residual" [61]). This is possibly because these layers are lightweight (due to the channel-wise design of MobileNets) and thus are economical to expand at first. Another interesting observation is that accuracy varies strongly, with the bottleneck expansion γ_b providing the highest top-1 accuracy of 55.0% and depth expansion γ_d the lowest with 51.3% at same complexity of 41.4M FLOPs.

(iii) The second step extends the temporal size of the model from one to two frames (expanding γ_τ and γ_t is identical for this step as there exists only a single frame in the previous one). This is what we expected to be the most effective expansion already in the first step as it enables the network to model temporal information for recognition.

(iv) The third step increases the spatial resolution γ_s and starts to show a pattern that is interesting. The expansion increases spatial and temporal resolution followed by depth (γ_d) in the fourth step. This is followed by multiple temporal expansions that increase temporal resolution (*i.e.* frame-rate) and input duration (γ_τ & γ_t), followed by two more expansions across the spatial resolution, γ_s, in steps 8 and 9, while step 10 increases the depth of the network, γ_d. An expansion of the depth after increasing input resolution is intuitive, as it allows to grow the filter receptive field resolution and size within each residual stage.

stage	filters	output sizes $T\times H\times W$
data layer	stride 6, 1^2	13×160×160
conv$_1$	1×3^2, 3×1, 24	13×80×80
res$_2$	$\begin{bmatrix}1\times1^2, 54\\3\times3^2, 54\\1\times1^2, 24\end{bmatrix}\times3$	13×40×40
res$_3$	$\begin{bmatrix}1\times1^2, 108\\3\times3^2, 108\\1\times1^2, 48\end{bmatrix}\times5$	13×20×20
res$_4$	$\begin{bmatrix}1\times1^2, 216\\3\times3^2, 216\\1\times1^2, 96\end{bmatrix}\times11$	13×10×10
res$_5$	$\begin{bmatrix}1\times1^2, 432\\3\times3^2, 432\\1\times1^2, 192\end{bmatrix}\times7$	13×5×5
conv$_5$	1×1^2, 432	13×5×5
pool$_5$	13×5×5	1×1×1
fc$_1$	1×1^2, 2048	1×1×1
fc$_2$	1×1^2, #classes	1×1×1

(a) **X3D-S** with 1.96G FLOPs, 3.76M param, and 72.9% top-1 accuracy using expansion of $\gamma_\tau=6$, $\gamma_t=13$, $\gamma_s=\sqrt{2}$, $\gamma_w=1$, $\gamma_b=2.25$, $\gamma_d=2.2$.

stage	filters	output sizes $T\times H\times W$
data layer	stride 5, 1^2	16×224×224
conv$_1$	1×3^2, 3×1, 24	16×112×112
res$_2$	$\begin{bmatrix}1\times1^2, 54\\3\times3^2, 54\\1\times1^2, 24\end{bmatrix}\times3$	16×56×56
res$_3$	$\begin{bmatrix}1\times1^2, 108\\3\times3^2, 108\\1\times1^2, 48\end{bmatrix}\times5$	16×28×28
res$_4$	$\begin{bmatrix}1\times1^2, 216\\3\times3^2, 216\\1\times1^2, 96\end{bmatrix}\times11$	16×14×14
res$_5$	$\begin{bmatrix}1\times1^2, 432\\3\times3^2, 432\\1\times1^2, 192\end{bmatrix}\times7$	16×7×7
conv$_5$	1×1^2, 432	16×7×7
pool$_5$	16×7×7	1×1×1
fc$_1$	1×1^2, 2048	1×1×1
fc$_2$	1×1^2, #classes	1×1×1

(b) **X3D-M** with 4.73G FLOPs, 3.76M param, and 74.6% top-1 accuracy using expansion of $\gamma_\tau=5$, $\gamma_t=16$, $\gamma_s=2$, $\gamma_w=1$, $\gamma_b=2.25$, $\gamma_d=2.2$.

stage	filters	output sizes $T\times H\times W$
data layer	stride 5, 1^2	16×312×312
conv$_1$	1×3^2, 3×1, 32	16×156×156
res$_2$	$\begin{bmatrix}1\times1^2, 72\\3\times3^2, 72\\1\times1^2, 32\end{bmatrix}\times5$	16×78×78
res$_3$	$\begin{bmatrix}1\times1^2, 162\\3\times3^2, 162\\1\times1^2, 72\end{bmatrix}\times10$	16×39×39
res$_4$	$\begin{bmatrix}1\times1^2, 306\\3\times3^2, 306\\1\times1^2, 136\end{bmatrix}\times25$	16×20×20
res$_5$	$\begin{bmatrix}1\times1^2, 630\\3\times3^2, 630\\1\times1^2, 280\end{bmatrix}\times15$	16×10×10
conv$_5$	1×1^2, 630	16×10×10
pool$_5$	16×10×10	1×1×1
fc$_1$	1×1^2, 2048	1×1×1
fc$_2$	1×1^2, #classes	1×1×1

(c) **X3D-XL** with 35.84G FLOPs & 10.99M param, and 78.4% top-1 acc. using expansion of $\gamma_\tau=5$, $\gamma_t=16$, $\gamma_s=2\sqrt{2}$, $\gamma_w=2.9$, $\gamma_b=2.25$, $\gamma_d=5$.

Table 3. Three instantiations of X3D with varying complexity. The top-1 accuracy corresponds to *10-Center* view testing on K400. The models in (a) and (b) only differ in spatiotemporal resolution of the input and activations (γ_t, γ_τ, γ_s), and (c) differs from (b) in spatial resolution, γ_s, width, γ_w, and depth, γ_d. See Table 1 for X2D. Surprisingly X3D-XL has a maximum width of 630 feature channels.

(v) Even though we start from a base model that is intentionally made tiny by having very few channels, the expansion does *not* choose to globally expand the width up to the 10$^{\text{th}}$ step of the expansion process, making X3D similar to the Fast pathway design [16] with high spatiotemporal resolution but low width. Finally, the last expansion step, shown in the top-right of Fig. 2, increases the width γ_w.

In the spirit of VGG models [8, 64] we define a set of networks based on their target complexity. We use FLOPs as this reflects a hardware agnostic measure of model complexity. Parameters are also possible, but as they would not be sensitive to the input and activation tensor size, we only report them as secondary metric. To cover the models from our expansion, Table 2 defines complexity regimes by FLOPs, ranging from extra small (XS) to extra large (XL).

Expanded instances. The smallest instance, **X3D-XS** is the output after 5 expansion steps. Expansion is simple and efficient as it requires to train few models that are mostly at a low compute regime. For **X3D-XS** each step trains models of around 0.04, 0.08, 0.15, 0.30, 0.60 GFLOPs. Since we train one model for each of the 6 axes the approximate cost for these five steps is roughly equal to training a single model of 6 ×1.17 GFLOPS (to be fair, this ignores overhead cost for data loading *etc*. as 6×5=30 models are trained overall).

The next larger model is **X3D-S** which is defined by one backward *contraction* step after the 7$^{\text{th}}$ expansion step. The contraction step simply reduces the expansion (γ_t) proportionally to roughly match the target regime of ≤ 2 GFLOPs. For this model we also tried to contract each other axis to match the target and found that γ_t is best among the others.

The next models in Table 2 is **X3D-M** (≤ 2 GFLOPs) that achieves 74.6% top-1 accuracy, **X3D-L** (≤ 20 GFLOPs) with 76.8% top-1 and **X3D-XL** (≤ 40 GFLOPs) with 78.4% top-1 accuracy by expansion in the consecutive steps.

Further speed/accuracy comparisons are provided in §B. Table 3 shows three instantiations of X3D with varying complexity. It is interesting to inspect the differences of the models, X3D-S in Table 3a is just a lower spatiotemporal resolution (γ_t, γ_τ, γ_s) version of Table 3b; therefore has the *same number of parameters*, and X3D-XL in Table 3c is created by expanding X3D-M 3b in spatial resolution (γ_s) and width (γ_w). See Table 1 for X2D.

4.2. Main Results

Kinetics-400. Table 4 shows the comparison with state-of-the-art results for three X3D instantiations.To be comparable to previous work, we use the same testing strategy, that is *10-LeftCenterRight* (*i.e*. 30-view) inference. For each model, the table reports (from-left-to-right) ImageNet pretraining (*pre*), top-1 and top-5 validation accuracy, average test accuracy as (top-1+ top-5)/2 (*i.e*. official test-server metric), inference cost (GFLOPs×views) and parameters.

In comparison to the state-of-the-art, SlowFast [15], **X3D-XL**, provides comparable (slightly lower) performance (-0.7% top-1 and identical top-5 accuracy), while requiring 4.8× fewer multiply-add operations (FLOPs) and 5.5× fewer parameters than SlowFast 16×8, R101 + NL blocks [81], and better accuracy than SlowFast 8×8, R101+NL with 2.4× fewer multiply-add operations and 5.5× fewer parameters. When comparing **X3D-L**, we observe similar performance as Channel-Separated Networks (ip-CSN-152) [76] and SlowFast 8×8, at 4.3× fewer FLOPs and 5.4× fewer parameters. Finally, in the lower compute regime **X3D-M** is comparable to SlowFast 4×16, R50 and Oct-I3D + NL [9] while having 4.7× fewer FLOPs and 9.1× fewer parameters. We observe consistent results on the test set with **X3D-XL** at 85.3% average top1/5, showing good generalization.

model	pre	top-1	top-5	test	GFLOPs×views	Param
I3D [7]	ImageNet	71.1	90.3	80.2	108 × N/A	12M
Two-Stream I3D [7]		75.7	92.0	82.8	216 × N/A	25M
Two-Stream S3D-G [89]		77.2	93.0		143 × N/A	23.1M
MF-Net [10]		72.8	90.4		11.1 × 50	8.0M
TSM R50 [51]		74.7	N/A		65 × 10	24.3M
Nonlocal R50 [81]		76.5	92.6		282 × 30	35.3M
Nonlocal R101 [81]		77.7	93.3	83.8	359 × 30	54.3M
Two-Stream I3D [7]	-	71.6	90.0		216 × NA	25.0M
R(2+1)D [77]	-	72.0	90.0		152 × 115	63.6M
Two-Stream R(2+1)D [77]	-	73.9	90.9		304 × 115	127.2M
Oct-I3D + NL [9]	-	75.7	N/A		28.9 × 30	33.6M
ip-CSN-152 [76]	-	77.8	92.8		109 × 30	32.8M
SlowFast 4×16, R50 [15]	-	75.6	92.1		36.1 × 30	34.4M
SlowFast 8×8, R101 [15]	-	77.9	93.2	84.2	106 × 30	53.7M
SlowFast 8×8, R101+NL [15]	-	78.7	93.5	84.9	116 × 30	59.9M
SlowFast 16×8, R101+NL [15]	-	**79.8**	**93.9**	**85.7**	234 × 30	59.9M
X3D-M	-	76.0	92.3	82.9	**6.2 × 30**	**3.8M**
X3D-L	-	77.5	92.9	83.8	24.8 × 30	6.1M
X3D-XL	-	79.1	**93.9**	85.3	48.4 × 30	11.0M

Table 4. **Comparison to the state-of-the-art on K400-val & test**. We report the inference cost with a single "view" (temporal clip with spatial crop) × the numbers of such views used (GFLOPs×views). "N/A" indicates the numbers are not available for us. The "test" column shows average of top1 and top5 on the Kinetics-400 testset.

model	pretrain	top-1	top-5	GFLOPs×views	Param
I3D [4]	-	71.9	90.1	108 × N/A	12M
Oct-I3D + NL [9]	ImageNet	76.0	N/A	25.6 × 30	12M
SlowFast 4×16, R50 [15]	-	78.8	94.0	36.1 × 30	34.4M
SlowFast 16×8, R101+NL [15]	-	81.8	95.1	234 × 30	59.9M
X3D-M	-	78.8	94.5	**6.2 × 30**	**3.8M**
X3D-XL	-	**81.9**	**95.5**	48.4 × 30	11.0M

Table 5. **Comparison with the state-of-the-art on Kinetics-600**. Results are consistent with K400 in Table 4 above.

Kinetics-600 is a larger version of Kinetics that shall demonstrate further generalization of our approach. Results are shown in Table 5. Our variants demonstrate similar performance as above, with the best model now providing slightly better performance than the previous state-of-the-art SlowFast 16×8, R101+NL [15], again for 4.8× fewer FLOPs (*i.e.* multiply-add operations) and 5.5× fewer parameter. In the lower computation regime, **X3D-M** is comparable to SlowFast 4×16, R50 but requires 5.8× fewer FLOPs and 9.1× fewer parameters.

Charades [62] is a dataset with longer range activities. Table 6 shows our results. **X3D-XL** provides higher performance (+0.9 mAP with K400 and +1.9mAP under K600 pretraining), while requiring 4.8× fewer multiply-add operations (FLOPs) and 5.5× fewer parameters than previous highest system, SlowFast [15] with+ NL blocks [81].

4.3. Ablation Experiments

This section provides ablation studies on K400 val and test sets, comparing accuracy and computational complexity.

Comparison with EfficientNet3D. We first aim to compare X3D with a 3D extension of EfficientNet [73]. This architecture uses exactly the same implementation extras such as channel-wise separable convolution [32] as as X3D,

model	pretrain	mAP	GFLOPs×views	Param
Nonlocal [81]	ImageNet+Kinetics400	37.5	544 × 30	54.3M
STRG, +NL [82]	ImageNet+Kinetics400	39.7	630 × 30	58.3M
Timeception [36]	Kinetics-400	41.1	N/A×N/A	N/A
LFB, +NL [84]	Kinetics-400	42.5	529 × 30	122M
SlowFast, +NL [15]	Kinetics-400	42.5	234 × 30	59.9M
SlowFast, +NL [15]	Kinetics-600	45.2	234 × 30	59.9M
X3D-XL	Kinetics-400	**43.4**	**48.4 × 30**	**11.0M**
X3D-XL	Kinetics-600	**47.1**	**48.4 × 30**	**11.0M**

Table 6. **Comparison with the state-of-the-art on Charades**. SlowFast variants are based on $T\times\tau = 16\times8$.

model	data	top-1	top-5	FLOPs (G)	Params (M)
EfficientNet3D-B0	K400 val	66.7	86.6	0.74	3.30
X3D-XS		68.6 (+1.9)	87.9 (+1.3)	0.60 (−1.4)	3.76 (+0.5)
EfficientNet3D-B3		72.4	89.6	6.91	8.19
X3D-M		74.6 (+2.2)	91.7 (+2.1)	4.73 (−2.2)	3.76 (−4.4)
EfficientNet3D-B4		74.5	90.6	23.80	12.16
X3D-L		76.8 (+2.3)	92.5 (+1.9)	18.37 (−5.4)	6.08 (−6.1)
EfficientNet3D-B0	K400 test	64.8	85.4	0.74	3.30
X3D-XS		66.6 (+1.8)	86.8 (+1.4)	0.60 (−1.4)	3.76 (+0.5)
EfficientNet3D-B3		69.9	88.1	6.91	8.19
X3D-M		73.0 (+2.1)	90.8 (+2.7)	4.73 (−2.2)	3.76 (−4.4)
EfficientNet3D-B4		71.8	88.9	23.80	12.16
X3D-L		74.6 (+2.8)	91.4 (+2.5)	18.37 (−5.4)	6.08 (−6.1)

Table 7. **Comparison to EfficientNet3D**: We compare to a 3D version of EfficientNet on K400-val and test. 10-Center clip testing is used. EfficientNet3D has the same mobile components as X3D.

but was found by searching a large number of models for optimal trade-off on image-classification. This ablation studies if a direct extension of EfficientNet to 3D is comparable to X3D (which is expanded by only training few models). EfficientNet models are provided for various complexity ranges. We ablate three versions, B0, B3 and B4 that are extended in 3D using uniform scaling coefficients [73] for the spatial and temporal resolution.

In Table 7, we compare three X3D models of similar complexity to EfficientNet3D on two sets, K400-val and K400-test (from top-to-bottom). Starting with K400-val (top rows), our model **X3D-XS**, corresponding to only 4 expansion steps in Fig. 2. is comparable in FLOPs (slightly lower) and parameters (slightly higher), to EfficientNet3D-B0, but achieves 1.9% higher top-1 and 1.3% higher top-1 accuracy.

Next, comparing **X3D-M** to EfficientNet3D-B3 shows a gain of 2.0% top-1 and 2.1% top-5, despite using 32% fewer FLOPs and 54% fewer parameters. Finally, comparing **X3D-L** to EfficientNet3D-B4 shows a gain of 2.3% top-1 and 1.9% top-5, while having 23% and 50% fewer FLOPs and parameters, respectively. Seeing larger gains for larger models underlines the benefit of progressive expansion, as more expansion steps have been performed for these.

Since our expansion is measured by validation set performance, it is interesting to see if this provides a benefit for X3D. Therefore, we investigate potential differences on the K400-test set, in the lower half of Table 7, where similar, even slightly higher improvements in accuracy can be observed when comparing the same models as above, showing that our models generalize well to the test set.

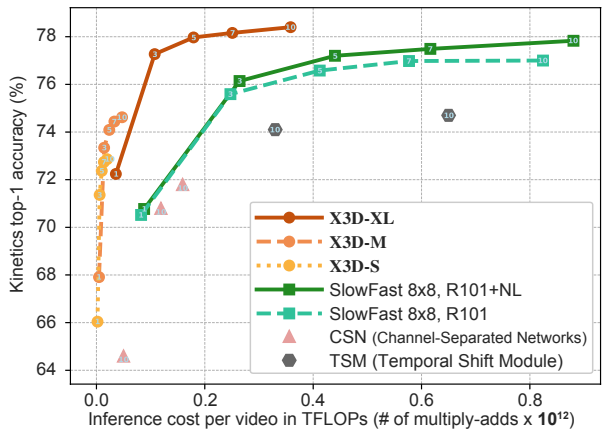

Figure 3. **Accuracy/complexity trade-off** on Kinetics-400 for different number of inference clips per video. The top-1 accuracy (vertical axis) is obtained by K-Center clip testing where the number of temporal clips $K \in \{1, 3, 5, 7, 10\}$ is shown in each curve. The horizontal axis shows the full inference cost per video.

Inference cost. In many cases, like the experiments before, the inference procedure follows a *fixed number of clips* for testing. Here we aim to ablate the effect of using *fewer* testing clips for video-level inference. In Fig. 3 we show the trade-off for the full inference of a video, when varying the number of temporal clips used. The vertical axis shows the top-1 accuracy on K400-val and the horizontal axis the overall inference cost in FLOPs for different models. Each model experiences a large performance increment when going from $K = 1$ clip to 3-clip testing (which triples the FLOPs); this is expected as the 1-clip only covers the temporal center of an input video, while 3-clip covers start, center and end. Increasing the number of clips beyond 3 only marginally increases performance, signaling that efficient video inference can be performed with *sparse clip sampling* if highest accuracy is not crucial. Lastly, when comparing different models we observe that X3D architectures can achieve similar accuracy as SlowFast [15], CSN [76] or TSM [51] (for the latter two, only 10-clip testing results are available to us), while requiring 3-20× fewer multiply-add operations. Notably we think of some of these ideas to be complementary to X3D (*e.g.* SlowFast) which can be explored in future work. A log-scale version of Fig. 3 and similar plots on K400-test are in §B.

5. Experiments: AVA Action Detection

Dataset. The AVA dataset [24] comes with bounding box annotations for spatiotemporal localization of (possibly multiple) human actions. There are 211k training and 57k validation video segments. We follow the standard protocol reporting mean Average Precision (mAP) on 60 classes [24].
Detection architecture. We exactly follow the detection architecture in [15] to allow direct comparison of X3D with SlowFast networks as a backbone. The detector is similar to Faster R-CNN [60] with minimal modifications adapted for video. Details on implementation and training are in §A.1.

model	AVA	pre	val mAP	GFLOPs	Param
LFB, R50+NL [84]	v2.1	K400	25.8	529	73.6M
LFB, R101+NL [84]			26.8	677	122M
X3D-XL			26.1	48.4	11.0M
SlowFast 4×16, R50 [15]	v2.2	K600	24.7	52.5	33.7M
SlowFast, 8×8 R101+NL [15]			27.4	146	59.2M
X3D-M			23.2	**6.2**	**3.1M**
X3D-XL			**27.4**	48.4	11.0M

Table 8. **Comparison with the state-of-the-art on AVA**. All methods use *single center crop* inference; full testing cost is directly proportional to the the floating point operations (GFLOPs) by multiplying with the number of validation segments (57k) in the dataset.

Inference. We perform inference on a single clip with γ_t frames sampled with stride γ_τ centered at the frame that is to be evaluated. Spatially we use a single center crop of $128\gamma_s \times 128\gamma_s$ pixels as in [84], to have a comparable measure for overall test costs, since fully-convolutional inference has variable cost depending on the input video size.

5.1. Main Results

We compare with state-of-the-art methods on AVA in Table 8. To be comparable to previous work, we report results on the older AVA version 2.1 and newer 2.2 (which provides more consistent annotations), for our models pre-trained on K400 or K600. We compare against long-term feature banks (LFB) [84] and SlowFast [15] as these are state-of-the art and use the same detection architecture as ours, varying the backbone from LFB, SlowFast and X3D. Note there are other recent works on AVA *e.g.*, [20, 66, 90, 96].

In the upper part of Table 8 we compare **X3D-XL** with LFB, that uses a heavy backbone architecture for short and long-term modeling. X3D-XL provides comparable accuracy (+0.3 mAP *vs*. LFB R50 and -0.7mAP *vs*. LFB R101) at greatly reduced cost by 10.9×/14× fewer multiply-adds and 6.7×/11.1× fewer parameters than LFB R50/R101.

Comparing to SlowFast [15] in the lower portion of the table we observe that **X3D-M** is lower than SlowFast 4×16, R50 by 1.5mAP, but requiring 8.5× less multiply-adds and 10.9×less parameters for this result. Comparing the larger **X3D-XL** to SlowFast 8×8 + NL we observe the same performance at 3× and 5.4× fewer multiply-adds and parameters.

6. Conclusion

This paper presents X3D, a spatiotemporal architecture that is progressively expanded from a tiny spatial network. Multiple candidate axes, in space, time, width and depth are considered for expansion under good computation/accuracy trade-off. A surprising finding of our progressive expansion is that networks with thin channel dimension and high spatiotemporal resolution can be effective for video recognition. X3D achieves competitive efficiency, and we hope that it can foster future research and applications in video recognition.

Acknowledgements: I thank Kaiming He, Jitendra Malik, Ross Girshick, and Piotr Dollár for valuable discussions and encouragement.

Appendix

This appendix provides further details as referenced in the main paper: Sec. A contains additional implementation details for: AVA Action Detection (§A.1), Charades Action Classification (§A.2), and Kinetics Action Classification (§A.3). Sec. B contains further results and ablations on Kinetics-400.

A. Additional Implementation Details

A.1. Details: AVA Action Detection

Detection architecture. We exactly follow the detection architecture in [15] to allow direct comparison of X3D with SlowFast networks as a backbone. The detector is similar to Faster R-CNN [60] with minimal modifications adapted for video. Since our paper focuses on efficiency, by default, we do not increase the spatial resolution of res_5 by $2\times$ [15]. Region-of-interest (RoI) features [21] are extracted at the last feature map of res_5 by extending a 2D proposal at a frame into a 3D RoI by replicating it along the temporal axis, similar as done in previous work [24, 40, 66], followed by application of frame-wise RoIAlign [27] and temporal global average pooling. The RoI features are then max-pooled and fed to a per-class, sigmoid classifier for prediction.

Training. For direct comparison, the training procedure and hyper-parameters for AVA follow [15] without modification. The network weights are initialized from the Kinetics models and we use step-wise learning rate decay, that is reduced by $10\times$ when validation error saturates. We train for 14k iterations (68 epochs for ~211k data), with linear warm-up [23] for the first 1k iterations and use a weight decay of 10^{-7}, as in [15]. All other hyper-parameters are the same as in the Kinetics experiments. Ground-truth boxes, and proposals overlapping with ground-truth boxes by IoU > 0.9, are used as the samples for training. The inputs are instantiation-specific clips of size $\gamma_t \times 112 \gamma_s \times 112 \gamma_s$ with time stride γ_τ.

The region proposal extraction also follows [15] and is summarized here for completeness. We follow previous works that use pre-computed proposals [24, 40, 66]. Our region proposals are computed by an off-the-shelf person detector, *i.e.*, that is not jointly trained with the action detection models. We adopt a person-detection model trained with *Detectron* [22]. It is a Faster R-CNN with a ResNeXt-101-FPN [52, 88] backbone. It is pre-trained on ImageNet and the COCO human keypoint images [53]. We fine-tune this detector on AVA for person (actor) detection. The person detector produces 93.9 AP@50 on the AVA validation set. Then, the region proposals for action detection are detected person boxes with a confidence of > 0.8, which has a recall of 91.1% and a precision of 90.7% for the person class.

A.2. Details: Charades Action Classification

For **Charades**, we fine-tune the Kinetics models. All settings are the same as those of Kinetics, except the following. A per-class sigmoid output is used to account for the mutli-class nature. We train on a single machine for 24k iterations using a batch size of 16 and a base learning rate of 0.02 with $10\times$ step-wise decay if the validation error saturates. We use weight decay of 10^{-5}. We also increase the model temporal stride by $\times 2$ as this dataset benefits from longer clips. For inference, we temporally max-pool prediction scores [15, 81].

A.3. Details: Kinetics Action Classification

Training details. We use the initialization in [28]. We adopt synchronized SGD training on 128 GPUs following the recipe in [23]. The mini-batch size is 8 clips per GPU (so the total mini-batch size is 1024). We train with Batch Normalization (BN) [38], and the BN statistics are computed within each 8 clips, unless noted otherwise. We adopt a half-period cosine schedule [54] of learning rate decaying: the learning rate at the n-th iteration is $\eta \cdot 0.5[\cos(\frac{n}{n_{\text{max}}}\pi) + 1]$, where n_{max} is the maximum training iterations and the base learning rate η is set as 1.6. We also use a linear warm-up strategy [23] in the first 8k iterations. Unless specified, we train for 256 epochs (60k iterations with a total mini-batch size of 1024, in ~240k Kinetics videos).We use momentum of 0.9, weight decay of 5×10^{-5} and dropout [30] of 0.5 is used before the final classifier.

For **Kinetics-600**, we extend the training epochs (and schedule) of above by $2\times$. All other hyper-parameters are exactly as for Kinetics-400.

Implementation details. Non-Local (NL) blocks [81] are not used for X3D. For SlowFast results, we use exactly the same implementation details as in [15]. Specifically, for SlowFast models involving NL, we initialize them with the counterparts that are trained without NL, to facilitate convergence. We only use NL on the (fused) Slow features of res_4 (instead of res_3+res_4 [81]). For X3D and EfficientNet3D, we follow previous work on 2D mobile architectures [31, 72, 73], using SE blocks [33] (also found beneficial for efficient video classification in [89]) and swish non-linearity [59]. To conserve memory, we use SE with original reduction ratio of $1/16$ only in every other residual block after the 3×3^2 conv; swish is only used before and after these layers and all other weight layers are followed by ReLU non-linearity [47]. We do not employ the "linear-bottleneck" design used in mobile image networks [31, 61, 72, 73], as we found it to sometimes cause instability in distributed training, as it does not allow to zero-initialize the final BN scaling [23] of residual blocks.

Expansion details. To expand the model specified in Table 1, we set all initial expansion factors, $\mathcal{X}_0$, to one *i.e.* $\gamma_t{=}\gamma_s{=}\gamma_w{=}\gamma_b{=}\gamma_d{=}1$ resulting in the X2D base model.

A temporal sampling rate γ_τ is not defined for the X2D model as it does not have multiple frames. The smallest possible common expansion for this model is defined by increasing the number of frames from 1 to two; therefore we set the expansion-rate $\hat{c}$ to match the cost of increasing the temporal input length of the model by a factor of two (the smallest possible common increase in the first expansion step), which roughly doubles the cost of the model, $\hat{c} = 2$.

Then, in every step of our expansion we train a models, one for expanding each axis, such that its complexity doubles ($\hat{c} = 2$). For the individual axes this roughly[3] equals to the following operations:

- X-Fast: $\gamma_\tau \leftarrow 0.5\gamma_\tau$, reduces the sampling stride to double frame-rate while sampling the same input duration, this doubles the temporal size $\gamma_t \leftarrow 2\gamma_t$.
- X-Temporal: Increases frame-rate by $\gamma_\tau \leftarrow 0.75\gamma_\tau$ *and* input duration to double the input size $\gamma_t \leftarrow 2\gamma_t$ (*i.e.* 1.5× higher frame-rate and 1.5×longer input duration).
- X-Spatial: Expands the spatial resolution proportionally $\gamma_s \leftarrow \sqrt{2}\gamma_s$.
- X-Depth: Expands the depth of the network by around $\gamma_d \leftarrow 2.2\gamma_d$.
- X-Width: Expands the global width for all layers by $\gamma_w \leftarrow 2\gamma_w$.
- X-Bottleneck: Expands the bottleneck width by roughly $\gamma_b \leftarrow 2.25\gamma_b$.

The exact scaling factors slightly differ from one expansion step to the other due to rounding effects in network geometry (layers stride, activation size *etc*.).

Since the stepwise expansion also allows to elegantly integrate regularization (which is typically increased for larger models), we perform a regularization expansion if the training error of the previous expansion step starts to deviate from the validation error. Specifically, we start the expansion with double the batch-size and half learning schedule than described above, then the BN statistics are computed within each 16 clips which lowers regularization and improves performance on small models. The batch-size is then decreased by 2× at the 8$^{\text{th}}$ step of expansion which increases generalization. We perform another regularization expansion at the 11$^{\text{th}}$ step by using drop-connect with rate 0.1 [34].

B. Additional Results

Fig. A.4 shows a series of extra plots on Kinetics-400, analyzed next (this extends Sec. 4 of the main paper):

[3]The exact expansion factors slightly vary across steps to match the complexity increase, $\hat{c}$ (which is observable in Fig. 2 of the main paper).

Inference cost. Here we aim to provide further ablations for the effect of using *fewer* testing clips for efficient video-level inference. In Fig. A.4 we show the trade-off for the full inference of a video, when varying the number of temporal clips used. The vertical axis shows the top-1 accuracy on K400-val and the horizontal axis the overall inference cost in FLOPs for different models.

First, for comparison, the plot on top-left is the same as the one shown in Fig. 3. The plot on top-right shows this same plot with a logarithmic scale applied to the FLOPs axis. Using this scaling it is clearer to observe that smaller models (X3D-S and X3D-M) can provide up to 20× reduction in terms of multiply-add operations used during inference.

For example, 3-clip X3D-S produces 71.4% top-1 at 5.9 GFLOPs, whereas 10-clip CSN-50 [76] produces 70.8 top-1 at 119 GFLOPs (20.2× higher cost), or 10-clip X3D-S 72.9% top-1 at 19.6 GFLOPs, and 10-clip CSN-101 [76] 71.8% top-1 at 159 GFLOPs (8.1× higher cost).

The lower two plots in Fig. A.4 show the identical results on the test set of Kinetics-400 (which has been publicly released with Kinetics-600 [4]). Note that the test set is more challenging which leads to overall lower accuracy for all approaches [1]. We observe consistent results on the test set, illustrating good generalization of the models.

Mobile components. Finally, we ablate the effect of mobile components employed in X3D and EfficientNet3D. Since the components can have different effects of models from the small and large computation regime, we ablate the effects on a small (X3D-S) and a large model (X3D-XL).

First, we ablate channel-wise separable convolution [32], a key component in mobile ConvNets. We ablate two versions: (i) A version that reduces the bottleneck ratio (γ_b) accordingly, such that the overall architecture preserves the multiple-add operations (FLOPs), and (ii) a version that keeps the originally, expanded bottleneck ratio.

Table A.1 shows the results. For case (i) we see that performance drops significantly by 4% top-1 accuracy for X3D-S and by 2.4% for X3D-XL. For case (ii), we see that the performance of the baselines increases by 0.3% and 0.8% top-1 accuracy for X3D-S and X3D-XL, respectively. This shows that separable convolution is important for small-computation budgets, however, for best-performance a non-separable convolution can provide gains (at high cost).

Second, we ablate swish non-linearities [59] (that are only implemented before and after the "bottleneck" convolution, to conserve memory). We observe that removing swish has a smaller performance decrease of 0.9% for X3D-S and 0.4% for X3D-XL, and therefore could be changed to ReLU (which can be implemented in-place) if memory is priority.

Third, we ablate SE blocks [33] (that are only used in every other residual block, to conserve memory). We observe that removing SE has a larger effect on performance, decreasing accuracy by 1.6% for X3D-S and 1.3% for X3D-XL.

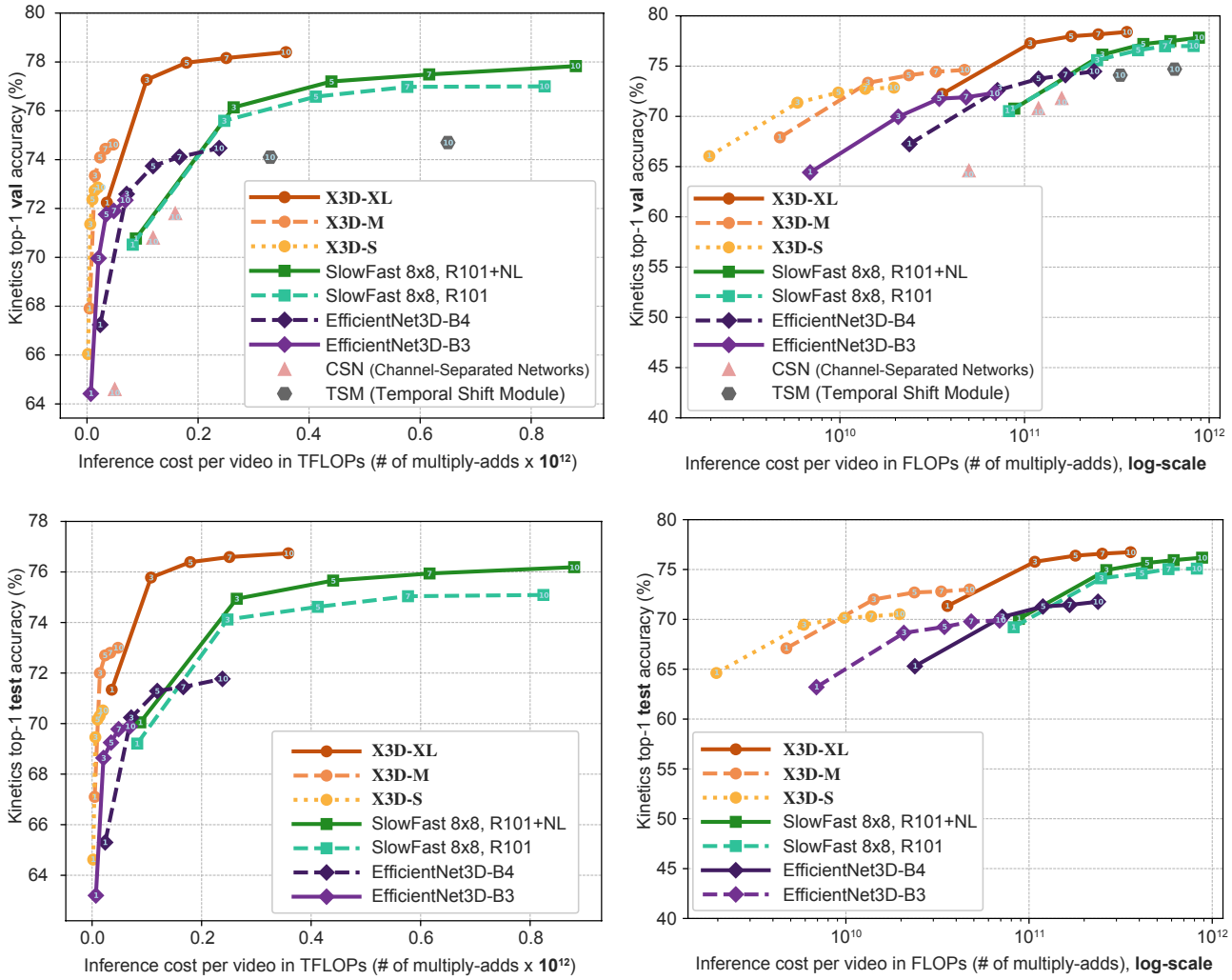

Figure A.4. **Accuracy/complexity trade-off** on K400-**val** (top) & **test** (bottom) for varying # of inference clips per video. The top-1 accuracy (vertical axis) is obtained by K-Center clip testing where the number of temporal clips $K \in \{1, 3, 5, 7, 10\}$ is shown in each curve. The horizontal axis measures the full inference cost per video. The left-sided plots show a linear and the right plots a logarithmic (**log**) scale.

These observed effects on performance as similar as the ones that have been shown Non-local (NL) attention blocks [81], and are also in line with [89], where SE attention blocks have been found beneficial for efficient video classification.

References

[1] ActivityNet-Challenge. http://activity-net.org/challenges/2017/evaluation.html, 2017. 10

[2] Humam Alwassel, Fabian Caba Heilbron, and Bernard Ghanem. Action search: Spotting actions in videos and its application to temporal action localization. In *Proc. ECCV*, 2018. 2

[3] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *Proc. CVPR*, 2016. 2

[4] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about Kinetics-600. *arXiv:1808.01340*, 2018. 2, 5, 7, 10

[5] Joao Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. A short note on the kinetics-700 human action dataset. *arXiv preprint arXiv:1907.06987*, 2019. 2

[6] Joao Carreira, Viorica Patraucean, Laurent Mazare, Andrew Zisserman, and Simon Osindero. Massively parallel video networks. In *Proc. ECCV*, 2018. 2

[7] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proc. CVPR*, 2017. 1, 2, 4, 7

[8] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *Proc. BMVC.*, 2014. 2, 3, 6

[9] Yunpeng Chen, Haoqi Fang, Bing Xu, Zhicheng Yan, Yannis Kalantidis, Marcus Rohrbach, Shuicheng Yan, and Jiashi Feng. Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution. *arXiv preprint arXiv:1904.05049*, 2019. 6, 7

[10] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng. Multi-fiber networks for video recognition.

model	top-1	top-5	FLOPs (G)	Param (M)
X3D-S	72.9	90.5	1.96	3.76
− CW conv γ_b= 0.6	68.9	88.8	1.95	3.16
− CW conv	73.2	90.4	17.6	22.1
− swish	72.0	90.4	1.96	3.76
− SE	71.3	89.9	1.96	3.60

(a) Ablating mobile components on a Small model.

model	top-1	top-5	FLOPs (G)	Param (M)
X3D-XL	78.4	93.6	35.84	11.0
− CW conv, γ_b= 0.56	76.0	92.6	34.80	9.73
− CW conv	79.2	93.5	365.4	95.1
− swish	78.0	93.4	35.84	11.0
− SE	77.1	93.0	35.84	10.4

(b) Ablating mobile components on an X-Large model.

Table A.1. Ablations of mobile components for video classification on K400-val. We show top-1 and top-5 classification accuracy (%), parameters, and computational complexity measured in GFLOPs (floating-point operations, in # of multiply-adds $\times 10^9$) for a single clip input of size $\gamma_t \times 112 \gamma_s \times 112 \gamma_x$. Inference-time computational cost is reported GFLOPs $\times 10$, as a fixed number of *10-Center* views is used. The results show that removing channel-wise separable convolution (CW conv) with unchanged bottleneck expansion ratio, γ_b, drastically increases mutliply-adds and parameters at slightly higher accuracy, while swish has a smaller effect on performance than SE.

In *Proc. ECCV*, 2018. 2, 7

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proc. CVPR*, 2009. 5

[12] Ali Diba, Mohsen Fayyaz, Vivek Sharma, M Mahdi Arzani, Rahman Yousefzadeh, Juergen Gall, and Luc Van Gool. Spatio-temporal channel correlation networks for action classification. In *Proc. ECCV*, 2018. 2

[13] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proc. CVPR*, 2015. 1, 2

[14] Lijie Fan, Wenbing Huang, Chuang Gan, Stefano Ermon, Boqing Gong, and Junzhou Huang. End-to-end learning of motion representation for video understanding. In *Proc. CVPR*, 2018. 2

[15] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. SlowFast networks for video recognition. In *Proc. ICCV*, 2019. 2, 3, 4, 5, 6, 7, 8, 9

[16] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. SlowFast networks for video recognition in ActivityNet challenge 2019. http://static.googleusercontent.com/media/research.google.com/en//ava/2019/fair_slowfast.pdf, 2019. 6

[17] Christoph Feichtenhofer, Axel Pinz, and Richard Wildes. Spatiotemporal residual networks for video action recognition. In *NIPS*, 2016. 1, 2, 4

[18] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proc. CVPR*, 2016. 2

[19] Basura Fernando, Efstratios Gavves, Jose M Oramas, Amir Ghodrati, and Tinne Tuytelaars. Modeling video evolution for action recognition. In *IEEE PAMI*, pages 5378–5387, 2015. 2

[20] Rohit Girdhar, João Carreira, Carl Doersch, and Andrew Zisserman. Video action transformer network. In *Proc. CVPR*, 2019. 8

[21] Ross Girshick. Fast R-CNN. In *Proc. ICCV*, 2015. 9

[22] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. https://github.com/facebookresearch/detectron, 2018. 9

[23] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: training ImageNet in 1 hour. *arXiv:1706.02677*, 2017. 9

[24] Chunhui Gu, Chen Sun, David A. Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, Cordelia Schmid, and Jitendra Malik. AVA: A video dataset of spatio-temporally localized atomic visual actions. In *Proc. CVPR*, 2018. 2, 8, 9

[25] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003. 2, 4

[26] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proc. CVPR*, 2018. 2

[27] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proc. ICCV*, 2017. 9

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. CVPR*, 2015. 9

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016. 1, 2, 3, 5

[30] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, 2012. 9

[31] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for MobileNetV3. *arXiv:1905.02244*, 2019. 1, 2, 3, 4, 9

[32] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1, 2, 3, 4, 7, 10

[33] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proc. CVPR*, 2018. 2, 9, 10

[34] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *Proc. ECCV*, 2016. 10

[35] Yanping Huang, Yonglong Cheng, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, and Zhifeng Chen. Gpipe:

Efficient training of giant neural networks using pipeline parallelism. *arXiv preprint arXiv:1811.06965*, 2018. 2, 3

[36] Noureldien Hussein, Efstratios Gavves, and Arnold WM Smeulders. Timeception for complex action recognition. In *Proc. CVPR*, 2019. 2, 7

[37] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size. *arXiv:1602.07360*, 2016. 3

[38] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*, 2015. 9

[39] Anil Jain and Douglas Zongker. Feature selection: Evaluation, application, and small sample performance. *IEEE transactions on pattern analysis and machine intelligence*, 19(2):153–158, 1997. 4

[40] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *Proc. CVPR*, 2018. 9

[41] George H John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *Machine Learning Proceedings 1994*, pages 121–129. Elsevier, 1994. 2, 4

[42] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li 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. 1, 3

[43] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv:1705.06950*, 2017. 2, 4

[44] Ron Kohavi and George H John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324, 1997. 2, 4

[45] Okan Köpüklü, Neslihan Kose, Ahmet Gunduz, and Gerhard Rigoll. Resource efficient 3d convolutional neural networks. *arXiv preprint arXiv:1904.02422*, 2019. 2, 4

[46] Bruno Korbar, Du Tran, and Lorenzo Torresani. Scsampler: Sampling salient clips from video for efficient action recognition. In *Proc. ICCV*, 2019. 2, 5

[47] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2, 3, 9

[48] Myunggi Lee, Seungeui Lee, Sungjoon Son, Gyutae Park, and Nojun Kwak. Motion feature network: Fixed motion filter for action recognition. In *Proc. ECCV*, 2018. 2

[49] Dong Li, Zhaofan Qiu, Qi Dai, Ting Yao, and Tao Mei. Recurrent tubelet proposal and recognition networks for action detection. In *Proc. ECCV*, 2018. 2

[50] Zhenyang Li, Kirill Gavrilyuk, Efstratios Gavves, Mihir Jain, and Cees GM Snoek. VideoLSTM convolves, attends and flows for action recognition. *Computer Vision and Image Understanding*, 166:41–50, 2018. 2

[51] Ji Lin, Chuang Gan, and Song Han. Temporal shift module for efficient video understanding. In *Proc. ICCV*, 2019. 2, 5, 7, 8

[52] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proc. CVPR*, 2017. 9

[53] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Proc. ECCV*, 2014. 9

[54] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. *arXiv:1608.03983*, 2016. 9

[55] Chenxu Luo and Alan L. Yuille. Grouped spatial-temporal aggregation for efficient action recognition. In *Proc. ICCV*, 2019. 2

[56] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proc. CVPR*, 2015. 1, 2

[57] AJ Piergiovanni and Michael S. Ryoo. Representation flow for action recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[58] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *Proc. ICCV*, 2017. 2

[59] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017. 2, 9, 10

[60] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 8, 9

[61] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proc. CVPR*, 2018. 1, 2, 3, 4, 5, 9

[62] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *ECCV*, 2016. 2, 7

[63] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 2, 3

[64] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. ICLR*, 2015. 1, 2, 3, 5, 6

[65] Yu-Chuan Su and Kristen Grauman. Leaving some stones unturned: dynamic feature prioritization for activity detection in streaming video. In *Proc. ECCV*, 2016. 2

[66] Chen Sun, Abhinav Shrivastava, Carl Vondrick, Kevin Murphy, Rahul Sukthankar, and Cordelia Schmid. Actor-centric relation network. In *ECCV*, 2018. 8, 9

[67] Lin Sun, Kui Jia, Kevin Chen, Dit-Yan Yeung, Bertram E Shi, and Silvio Savarese. Lattice long short-term memory for human action recognition. In *Proc. ICCV*, 2017. 2

[68] Lin Sun, Kui Jia, Dit-Yan Yeung, and Bertram Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *Proc. ICCV*, 2015. 2

[69] Shuyang Sun, Zhanghui Kuang, Lu Sheng, Wanli Ouyang, and Wei Zhang. Optical flow guided feature: A fast and robust motion representation for video action recognition. In *Proc. CVPR*, 2018. 2

[70] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv:1602.07261*, 2016. 2, 3

[71] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proc. CVPR*, 2015. 1, 2, 3

[72] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. MnasNet: Platform-aware neural architecture search for mobile. In *Proc. CVPR*, 2019. 2, 9

[73] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. 2, 3, 7, 9

[74] Graham W Taylor, Rob Fergus, Yann LeCun, and Christoph Bregler. Convolutional learning of spatio-temporal features. In *Proc. ECCV*, 2010. 2

[75] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3D convolutional networks. In *Proc. ICCV*, 2015. 1, 2, 3, 4

[76] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *Proc. ICCV*, 2019. 2, 4, 5, 6, 7, 8, 10

[77] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proc. CVPR*, 2018. 3, 7

[78] Heng Wang, Du Tran, Lorenzo Torresani, and Matt Feiszli. Video modeling with correlation networks. *arXiv preprint arXiv:1906.03349*, 2019. 2

[79] Limin Wang, Wei Li, Wen Li, and Luc Van Gool. Appearance-and-relation networks for video classification. In *Proc. CVPR*, 2018. 2

[80] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Val Gool. Temporal segment networks: Towards good practices for deep action recognition. In *Proc. ECCV*, 2016. 2

[81] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proc. CVPR*, 2018. 1, 2, 4, 5, 6, 7, 9, 11

[82] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *Proc. ECCV*, 2018. 7

[83] Stephen J Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015. 1, 4

[84] Chao-Yuan Wu, Christoph Feichtenhofer, Haoqi Fan, Kaiming He, Philipp Krähenbühl, and Ross Girshick. Long-term feature banks for detailed video understanding. In *Proc. CVPR*, 2019. 2, 5, 7, 8

[85] Chao-Yuan Wu, Manzil Zaheer, Hexiang Hu, R Manmatha, Alexander J Smola, and Philipp Krähenbühl. Compressed video action recognition. In *CVPR*, 2018. 2

[86] Wenhao Wu, Dongliang He, Xiao Tan, Shifeng Chen, and Shilei Wen. Multi-agent reinforcement learning based frame sampling for effective untrimmed video recognition. In *Proc. ICCV*, 2019. 2

[87] Zuxuan Wu, Caiming Xiong, Chih-Yao Ma, Richard Socher, and Larry S Davis. Adaframe: Adaptive frame selection for fast video recognition. In *Proc. CVPR*, 2019. 2

[88] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proc. CVPR*, 2017. 2, 3, 9

[89] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning for video understanding. *arXiv:1712.04851*, 2017. 2, 7, 9, 11

[90] Xitong Yang, Xiaodong Yang, Ming-Yu Liu, Fanyi Xiao, Larry S Davis, and Jan Kautz. Step: Spatio-temporal progressive learning for video action detection. In *Proc. CVPR*, 2019. 8

[91] Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *Proc. CVPR*, 2016. 2

[92] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. In *ICML Workshop*, 2015. 2

[93] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Proc. BMVC.*, 2016. 2, 3

[94] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv:1212.5701*, 2012. 2, 3

[95] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proc. CVPR*, 2018. 2, 3, 4

[96] Yubo Zhang, Pavel Tokmakov, Martial Hebert, and Cordelia Schmid. A structured model for action detection. In *Proc. CVPR*, 2019. 8

[97] Linchao Zhu, Laura Sevilla-Lara, Du Tran, Matt Feiszli, Yi Yang, and Heng Wang. Faster recurrent networks for video classification. *arXiv preprint arXiv:1906.04226*, 2019. 2

[98] Yi Zhu, Zhenzhong Lan, Shawn Newsam, and Alexander G Hauptmann. Hidden two-stream convolutional networks for action recognition. *arXiv preprint arXiv:1704.00389*, 2017. 2

[99] Mohammadreza Zolfaghari, Kamaljeet Singh, and Thomas Brox. ECO: efficient convolutional network for online video understanding. In *Proc. ECCV*, 2018. 2