Auto-X3D: Ultra-Efficient Video Understanding via Finer-Grained Neural Architecture Search

Yifan Jiang[1]*, Xinyu Gong[1]*, Junru Wu[2], Humphrey Shi[3,5], Zhicheng Yan[4], Zhangyang Wang[1]
[1]University of Texas at Austin [2]Texas A&M University
[3]University of Oregon [4]Facebook AI [5]Picsart AI Research (PAIR)
{yifanjiang97,xinyu.gong,atlaswang}@utexas.edu

Abstract

*Efficient video architecture is the key to deploying video recognition systems on devices with limited computing resources. Unfortunately, existing video architectures are often computationally intensive and not suitable for such applications. The recent X3D work presents a new family of efficient video models by expanding a hand-crafted image architecture along multiple axes, such as space, time, width, and depth. Although operating in a conceptually large space, X3D searches one axis at a time, and merely explored a small set of 30 architectures in total, which does not sufficiently explore the space. This paper bypasses existing 2D architectures, and directly searched for 3D architectures in a fine-grained space, where block type, filter number, expansion ratio and attention block are jointly searched. A probabilistic neural architecture search method is adopted to efficiently search in such a large space. Evaluations on Kinetics and Something-Something-V2 benchmarks confirm our AutoX3D models outperform existing ones in accuracy up to **1.3**% under similar FLOPs, and reduce the computational cost up to ×**1.74** when reaching similar performance.*

1. Introduction

Video models are arguably living in a much larger design space than image models, as they are expected to abstract information in both spatial- and temporal dimensions, which requires a larger set of layer types and more complicated arrangements of the layers. However, many existing video models are simply built on top of image models, which are tailored to process video by applying image models to individual/a stack of frames (e.g. two-stream network [38]), replacing 2D- with 3D convolution (e.g. C3D [34], I3D [2], S3D [42], CSN [35]), or adding temporal convolution (e.g. R(2+1)D [36]). Such an image-to-video design process has served us as a fast path to build video architectures, but may also limit our exploration of the video architectures. On the other side, to design video architectures from scratch, one needs to take into account many more factors, than that of image recognition models. For example, one has to not only extend the inputs, features, and/or filter kernels into the temporal axis [2], but also strike a balance over many factors, such as the depth (number of layers), width (filter number), and spatial resolution, to achieve a good Accuracy-To-Complexity (ATC) trade-off [33].

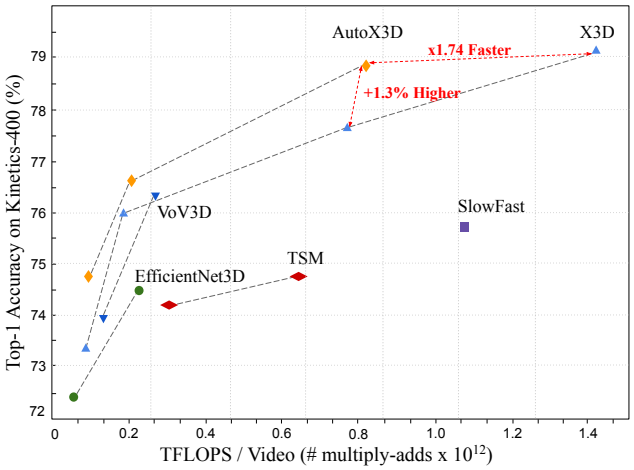


Figure 1: **Main results on Kinetics-400**. Compared to other SOTA models, the proposed AutoX3D network series achieve better Accuracy-To-Complexity (ATC) trade-off.

Recently, machine-designed architectures by Neural Architecture Search (NAS) have surpassed the human-designed ones for image recognition [28, 24, 33, 44]. For video action recognition, the latest work X3D [9] placed a new milestone in this line: it progressively expanded a hand-crafted 2D architecture into 3D spatial-temporal ones, by expanding along multiple axes, including space, time, width, and depth. As the joint search space of all axes is extremely large, X3D only searches over one axis at a time, progressively expanding the architectures to meet the

*The first two authors contribute equally.

target complexity. Although competitive ATC trade-off is achieved, those architectures are searched after only exploring 30 architectures in the space, which includes a total of 3,780 architectures. *Can we search video architectures with better ATC if the design space is more sufficiently explored?*

Moreover, X3D models still conform to many manual design choices in the hand-crafted 2D architecture based on MobileNet [31]. For example, the temporal-spatial kernel size is fixed to be 3×3^2 for depthwise convolution in the 3D MBConv (Mobile Inverted Bottleneck Conv) building block, the expansion rate is fixed to be 2.25, and all blocks in the same stage chooses the same number of filters while the difference in the number of filters between blocks from neighboring stages is always $2\times$. For the model architecture, X3D only explores to uniformly expand the number of filters and number of blocks to change the model complexity. *Can we go beyond hand-crafted 2D architectures, and directly search for 3D architectures while allowing different architectures in individual building blocks?*

This paper aims to address those questions, and takes steps further towards the direction of *efficient video recognition* . The goal is fundamentally challenging as it requires us to rethink the design space of the video architecture, relax many constraints which are made to facilitate manual exploration, and more aggressively explore the design space.

To this end, this paper directly searches for efficient video architecture, leading to our work of **AutoX3D** . Different from X3D which ignores the variations of the block types and only uniformly expands the channels, we introduce a fine-grained search space for efficient video models, by including more efficient operations. Unlike the conventional differentiable NAS (e.g. DARTS [20]), which repeatedly stacks the cells of the same structure in the network, our search allows independent blocks at different layers. In our fine-grained space, we jointly search for the block types, filter numbers, expansion ratio, and attention blocks, substantially surpassing those in previous work on image NAS [20, 28, 47, 40] and video NAS [9, 30, 30, 29]. To support fast search, we adopt the probabilistic differentiable NAS (PARSEC) [4] as the basic search algorithm, which requires much less memory footprint than conventional DNAS (e.g. DARTS [20]). Furthermore, we extend it to support the joint search among different factors (depth, channel, expansion ratio etc.). For better sample efficiency in our search space, we introduce the fairness-aware channel selection strategy into the probabilistic search algorithm, which enables a more stable, accurate, and scalable search framework for our task.

To summarize, we view the contributions as following.

- A fine-grained design space for efficient video recognition is proposed, to jointly search the block types, filter numbers, expansion ratio, and attention blocks.
- A family of efficient AutoX3D models, which are directly searched without any architecture surrogate or the use of hand-crafted image architecture, are presented. Their architectures are distinctive from hand-crafted models, and provide new insights of designing video architectures.
- Extensive evaluations on Kinetics [16] and Something-Something [12] benchmarks confirm our AutoX3D models achieve significantly better accuracy-to-complexity trade-off than other existing models.

2. Related Work

2.1. Video Architecture Design

Early video models are built on top of image ones. As the name implies, Two-Stream Network [38] uses two streams of image architectures to process single-frame and multi-frame optical flow inputs, respectively. To capture temporal feature, R(2+1)D [36] adds 1D temporal convolution to the 2D ResNet model while I3D [2] replaces 2D convolution with more expensive 3D convolution. To model long-range temporal relations, [45, 8] runs LSTMs on top of image features extracted from video frames, while [39] uses non-local blocks. Those video architectures are computationally heavy, and not suitable to be deployed on devices with limited computing power.

Due to the prevalence of mobile devices, designing efficient video architectures has become an increasingly important task. Efficient blocks, such as Temporal Shift module [19] and Video Shuffling module [22], are proposed to capture the temporal dynamics at low cost. More fine-grained adaptation of 2D models is also proposed. For example, in [42] it reports for I3D backbone, top-heavy modeling achieves better ATC trade-off by placing expensive 3D convolutions in the top layers only. However, such adaptation is not effective for ResNet backbone [36], highlighting the limitation of manually designed video architectures. Recently, search-based video architecture design, such as Tiny Video Network [25] and X3D [9], has made significant progress towards efficient video architecture. In particular, X3D presents new insights for turning a 2D architecture into 3D one by progressively expanding it along multiple axes, such as width, depth, and time. Different from X3D, we completely search video architecture from scratch in a more fine-grained space, and demonstrate the discovered architectures can squeeze out more gain in terms of efficiency and accuracy.

2.2. Neural Architecture Search Methods

Neural architecture search (NAS) aims to replace the laborious human design of network architectures, as well as towards more efficient ones [32]. The conventional evolution or reinforcement learning-based search methods are

slow and take thousands of GPU days. For example, searching an image model for CIFAR-10 and ImageNet required 2000 GPU days of reinforcement learning (RL) [46] or 3150 GPU days of evolution [28]. ENAS [24] introduced a parameter-sharing strategy to reduce the search time. Recent differentiable NAS (DNAS) methods [20] introduced the softmax-based continuous relaxation of the architecture representation, allowing efficient search using gradient descent. However, they compute features for all layer choices, and uses much more memory than standalone model training. The probabilistic version of DNAS, such as PARSEC [4], describes the population of architectures in the design space by a distribution, and only requires to sample one architecture at a time, which uses memory only as much as in the standalone model training. In this work, we employ PARSEC as the search procedure to search efficient video architectures in our fine-grained search space due to its low memory footprint and fast search efficiency.

2.3. Neural Architecture Design Space

The success of searching efficient models depends not only on the efficiency of the search method, but also on the prescription of the architecture design space. In [26], a new comparison paradigm is proposed to compare different design spaces. NASNet [47] proposed a design space of a cell which is a directed acyclic graph. Only the connectivity between graph nodes and the operator type on each graph edge is searched. The final model is obtained by stacking cells with the same architecture. Similarly, DARTS [20] also only searches the cell architecture, and repeats cells to obtain the final model. Although sharing the architectures across cells can reduce the size of model design space and simplify the search, it can be detrimental to video architecture design. Recent hand-crafted video models, such as S3D [42] and SlowFast [10], only use expensive 3D convolution in the top layers to improve the ATC trade-offs, while leaving convolutions in the bottom layers to be 2D. Other design choices, such as the building block type and filter number, are still manually chosen and often uniform. To search for efficient video models, we relax such hard-coded constraints, allow individual building blocks to have separate architectures and filter numbers.

3. AutoX3D Design Space

In this section, we introduce the design space of AutoX3D architectures. We define a model as a stack of building blocks, and choose to use the 3D version of Mobile Inverted Bottleneck Conv block (MBConv) from MobileNetV2 [31] as our building block. The original MBConv block is proven to be more efficient than the ResNet building block [13] in prior work [31, 14] and its 3D version is suitable for building efficient video architectures. Unlike X3D models [9], in our space, the design of the video architecture is more flexible in the following aspects.

Type	Kernel
t1_s3	1×3^2
t1_s5	1×5^2
t3_s3	3×3^2
t3_s5	3×5^2
t5_s3	5×3^2
t5_s5	5×5^2

Table 1: **The choices of MBConv3D block micro architecture in our space.**

3.1. Nonuniform Block Micro-Architecture

In X3D models, the 3D MBConv building block always consists of a conv1×1 layer to expand the filter number, a depthwise convolution layer of kernel size 3×3^2 to convolve with spatial features, and another conv1×1 layer to shrink the filter number in the final output feature. Depthwise convolution with kernel 3×3^2 is more expensive than that with a 2D kernel of size 1×3^2, and such choice of uniform building block micro-architecture is sub-optimal to achieve good ATC. Many prior works [36, 42] have examined the progressively varying temporal-spatial feature patterns along with the model depth. For example, S3D [42] work has shown employing 3D convolution at bottom layers (close to input) is less cost-effective than using it at top layers (close to final prediction). On the other side, other convolution kernel sizes, such as 1×5^2 and 3×5^2, are not yet explored in hand-crafted models due to a large number of different combinations of spatial-temporal kernel size. In AutoX3D design space, we consider 3D MBConv block with a different spatial-temporal kernel size in depthwise convolution as a separate block micro-architecture.

In Table 1, we present the choices of the 3D MBConv block micro-architectures in our design space.

3.2. Searchable Filter Number

The choices of the filter number in the building blocks of the hand-crafted models often follow simple heuristics. For example, in I3D [2], S3D [42], R(2+1)D [36], and X3D [9] models, all of which have 4 stages and each stage has multiple building blocks, the filter number of the output feature from the building blocks is fixed within the stage, and is doubled in the next stage. The principle of this heuristic choice is originally from 2D ResNet work [13], whereby doubling the filter number when the spatial resolution of the 2D feature map is reduced by half in the next stage, all building blocks use a similar amount of FLOPS and the compute of the overall model is evenly distributed to the building blocks.

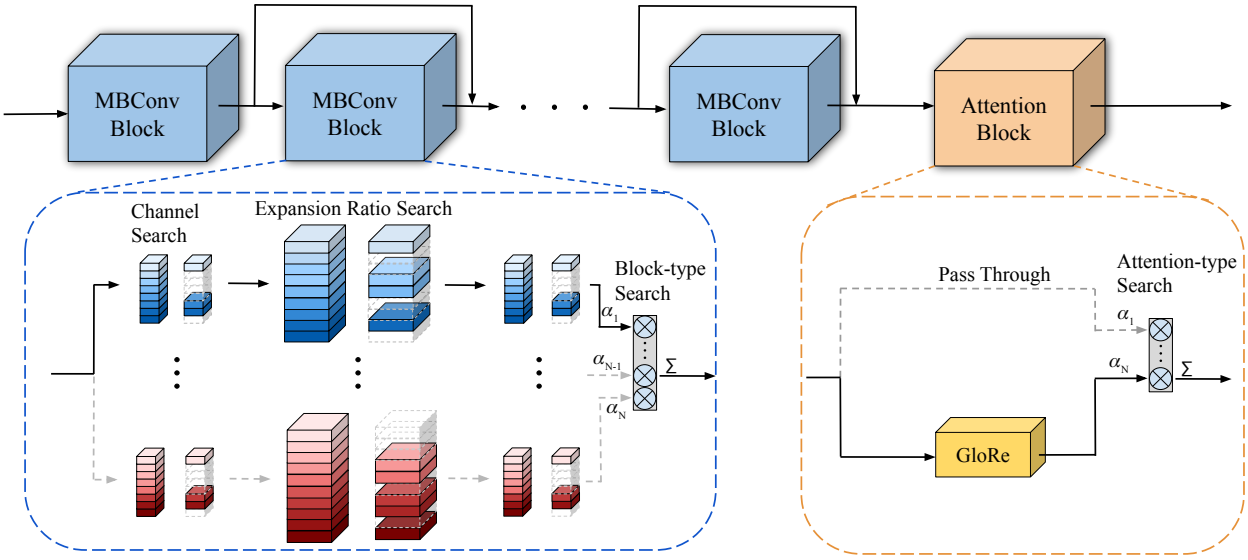


Figure 2: **The macro-architecture of AutoX3D design space.**

However, this principle does not hold for 3D video models, as the 3D feature map has both temporal- and spatial-resolution, and often only the spatial-resolution is reduced by half in the next stage while the temporal resolution is not reduced for achieving high recognition performance. For example, in SlowFast [10] and X3D [9] models, the temporal resolution is not reduced in all 4 stages.

Moreover, the choice of evenly distributing computation to all building blocks is quite ad-hoc. In our design space, we do not manually prescribe the filter number for the building blocks. Instead, we design a separate range of filter number choices for individual building blocks in the model, and even allow building blocks in the same stage to have different filter numbers. As the choices of filter number have a large impact on both the recognition performance and model computational cost, our flexible design and fine-grained choices allow the search methods to discover architectures with a better ATC trade-off.

The choices of filter numbers for individual building blocks are shown as part of the macro-architecture of our design space in Table 2.

3.3. Nonuniform Expansion Rate

In MBConv block, a conv1 $\times$ 1 layer is used to expand the filter number, and the expansion rate decides the channel number of the feature map where the following depth-wise convolution will operate on. For simplicity, a uniform choice of the expansion rate is often chosen by the hand-crafted models. For example, in the original MobileNet family of models [31, 14], all MBConv blocks use the expansion rate 6 while in X3D models [9], the expansion rate is always 2.25. The uniform choice of the expansion rate ignores the variations of feature representation needed at different model depths, and we hypothesize it leads to sub-optimal architectures.

Max Input $C \times T \times S^2$	Block Type	Expansion	Channel	Number	Spatial Stride	Att. Block
$3\times13\times160^2$	data	1	24	1	1	-
$3\times13\times80^2$	1×3^2	1	24	1	2	-
$28\times13\times40^2$	TBS	1.5 $\sim$ 6.0, 0.75	12 $\sim$ 28, 4	1	2	TBS
$28\times13\times40^2$	TBS		12 $\sim$ 28, 4	2	1	TBS
$64\times13\times20^2$	TBS		24 $\sim$ 64, 8	2	2	TBS
$64\times13\times20^2$	TBS		24 $\sim$ 64, 8	3	1	TBS
$132\times13\times10^2$	TBS		48 $\sim$ 132, 12	2	2	TBS
$132\times13\times10^2$	TBS		48 $\sim$ 132, 12	3	1	TBS
$132\times13\times10^2$	TBS		48 $\sim$ 132, 12	3	1	TBS
$132\times13\times10^2$	TBS		48 $\sim$ 132, 12	3	1	TBS
$264\times13\times5^2$	TBS		96 $\sim$ 264, 24	2	2	TBS
$264\times13\times5^2$	TBS		96 $\sim$ 264, 24	2	1	TBS
$264\times13\times5^2$	TBS		96 $\sim$ 264, 24	3	1	TBS
432	pool	-	432	1	-	-
2048	fc	-	2048	1	-	-

Table 2: **The macro-architecture of our AutoX3D design space. "TBS" means the operators that are to be searched.**

In our AutoX3D design space, we predefine a wide range of choices for the expansion ratio in individual building blocks, and allow different choices are chosen by the building blocks. The choices of the expansion ratio can be seen in the macro-architecture of our design space in Table 2.

3.4. Searchable Attention Block

Attention blocks, such as Nonlocal- [39] and GloRe block [6], can be easily plugged into the backbone model for improving the performance. One can choose to insert

more than one such blocks at different layers of the backbone. For example, in the GloRe work [6], for ResNet-50 backbone, it chooses to insert 3 GloRe blocks at Res4 stage for achieving a large gain in accuracy. Since modern deep models often contain dozens of layers and we can insert the attention block after any layer, a manual exhaustive exploration for selecting the best subset of layers to insert attention blocks is almost intractable. In our design space, we view the insertion of the attention block as a search problem, and define a set of choices of the attention block below to search the inserted position.

Pass through. The feature will simply pass through the block without any computation. In practice, it represents no attention block placed here.

GloRe. The original GloRe block, which includes a feature projection module, a graph convolution network (GCN) module, and an inverse feature projection module. The feature projection module projects spatial features from the coordinate space to the interaction space by aggregating spatial features with attention. The GCN module reasons over the projected features nodes in the interaction space. Finally, the inverse projection module re-projects the feature nodes back into the coordinate space.

3.5. Final AutoX3D Design Space

The final design space is illustrated in Figure 2. In Table 2, we prescribe a video macro-architecture, which mainly consists of a stack of 3D MBConv building blocks. We organize the blocks by groups, and blocks in each group share the same design choices. For each group, we search the micro-architecture, filter number, the expansion rate, and the type of attention block which will be inserted after each group. We allow the block groups to have different choices along those design axes. In total, our design space contains 2×10^{32} different video architectures, and represents a fine-grained space at a scale that has not been explored before.

4. Search Method

4.1. Probabilistic-based Architecture Search

As video models commonly require larger computation resources than image, directly applying the original differentiable NAS becomes unpractical. Here we adopt the PARSEC [4] approach, a probabilistic version of the differentiable NAS method, to guide our search procedure. Compared to DARTS [20], it only requires much memory as is needed to train a single architecture from our search space. This is because of a memory-efficient sampling procedure as we learn a probability distribution over high-performing neural network architectures. Here we define a unique discrete architecture as A, which is sampled from a prior distribution $P(A|\boldsymbol{\alpha})$. Architecture parameters $\boldsymbol{\alpha}$ denote the probabilities of choosing different operations. The goal of PARSEC is to optimize the architecture parameter $\boldsymbol{\alpha}$, in order to maximize the architecture accuracy. Concretely, for video recognition where we have video samples $\mathbf{X}$ and labels $\mathbf{y}$, probabilistic NAS can be formulated as optimizing the continuous architecture parameters α via an empirical Bayes Monte Carlo procedure [27]

$$\begin{aligned} P(\mathbf{y}|\mathbf{X},\omega,\alpha) &= \int P(\mathbf{y}|\mathbf{X},\omega,A)P(A|\boldsymbol{\alpha})\mathrm{d}A \\ &\approx \frac{1}{K}\sum_{k} P(\mathbf{y}|\mathbf{X},\omega,A_k), \end{aligned} \tag{1}$$

where ω denotes the model weights. The continuous integral of data likelihood is approximated by sampling K architectures and averaging the data likelihoods from them. We can jointly optimize architecture parameters $\boldsymbol{\alpha}$ and model weights ω by estimating gradients $\nabla_\alpha \log P(\mathbf{y}|\mathbf{X},\omega,\boldsymbol{\alpha})$ and $\nabla_w \log P(\mathbf{y}|\mathbf{X},\omega,\boldsymbol{\alpha})$ through the sampled architectures. Typically, the number of sampled architecture K is set to 13, which is sufficient to search for a good architecture empirically, according to our preliminary experiments.

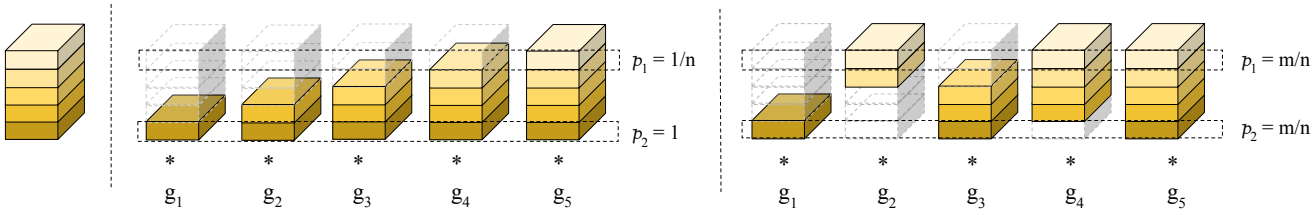
To target at efficient architecture discovering, we rebuild PARSEC to support cost-aware search that aiming to not only finding the best performance architecture but also under determined FLOPs constraint. We achieve this goal by adding another cost-aware hinge loss term to the original loss function. Given an architecture A_k, the cost-aware loss term c can be written as:

$$c(A_k) = \frac{1}{T}\max(FLOPs(A_k) - T, 0), \tag{2}$$

where T is our target FLOPs. All the search variables are jointly searched.

4.2. Fair Channel Selection

An important pitfall we found during channel search is the *unfairness* for the chances of candidate channels being selected. We show an example in Figure 3: on the left, when splitting the super kernel into N parts (N denotes the channel space size, with N = 5 in the figure just as an example) in the default order, the bottom part will be shared with all channel candidates and therefore always be updated. Meanwhile, the higher bottoms have fewer chances to be updated, with the top one only receiving sparse updates of $1/N$ probability. That unfairness often leads to the supernet training collapsing onto the most frequently updated channel(s) (e.g. the bottom one), and mislead the architecture selection to sub-optimal performance. To tackle this issue, we propose to *select channels with fairness*, i.e., the assignment of channel candidates shall ensure the chance of selecting/updating each super kernel part to be as equal as



(a) Unfair Channel Selection

(b) Fair Channel Selection

Figure 3: **Illustration of fairness-aware channel search**, using five channel parts for example. Where p_i is the initial probability of each filter and g_i represent the architecture weight of each candidates. We only activate one candidates in the forward pass. All channel candidates share weight from the original filters. For unfair channel selection, the small probability one gets fewer chance to update the corresponding filter compared to large probability one.

possible. Figure 3 right displays the fairness-aware selection pattern for $N = 5$, where each part now has the equal chance of 3/5 to be selected and updated. A more general discussion on how to design such fair patterns for different Ns can be found in the supplement.

5. Experiments

5.1. Implementation Details

5.1.1 Datasets

We adopt two benchmarks of video action recognition in the experiments, including Kinetics [16] and Something-Something-V2 [12].

Kinetics [16]. Kinetics-400 is a large-scale dataset that contains 400 action classes with 240K training examples and 20K validation examples. To accelerate searching, we adopt Mini-Kinetic200 [43] as our searching dataset, which is a subset of Kinetics-400 containing 200 classes of videos. As for the evaluation of the searched architecture, we use the full standard Kinetics-400 dataset for training and testing.

Something-Something-V2. Something-something-V2 dataset contains 108k videos with 174 classes of diverse actions, while each video lasting between 2 to 6 seconds. Different from Kinetics[16], the video sequence in something-something is more temporal related, as it focuses on humans performing predefined basic actions with different objects. Therefore, this dataset serves as a suitable benchmark to evaluate the effectiveness of temporal modeling.

5.1.2 Search and Derivation Setting

Architecture Search We implement PARSEC [4] in PyTorch, and use 64 Nvidia V100 GPUs to search architectures on Mini-Kinetics-200. By default, we use the input video clip of size $T \times S^2 = 13 \times 160^2$ with a spatial scale jittering range of [182, 228], and set the target FLOPS to $2G$ FLOPS, which is on par with the FLOPS of X3D-S model. We adopt Adam with 0.02 learning rate and 0 weight decay to optimize the architecture weights. SGD is adopted to update the supernet's parameters with a learning rate of 0.6, 0.9 momentum, and 5e-5 weight decay. The total search process takes 600 epoches with 3 days.

Architecture Training We evaluate the searched architectures by using the open-source ClassyVision [1] framework to train them from scratch on the benchmarks. We use 64 Nvidia V100 GPUs in the experiments. We adopt SGD with 0.9 momentum and 5e-5 weight decay. The learning rate is set to 0.4 following the half-period cosine decaying strategy and mini-batch size is 8 clips per GPU. The whole training process takes 300 epochs while we use linear warm-up strategy [11] for the first 34 epochs. Dropout is adopted at the head of the network with the probability of 0.5. We also apply AutoAugment [7] on each frame of input video clip. The data pre-processing part is the same as the searching part. More training detials are illustrated in the supplement.

5.1.3 Architecture Evaluation

To be comparable with previous work, we mainly follow the evaluation setting of X3D [9]. We evaluate the searched architecture on the validation set. We by default adopt 30-views evaluation, unless specifically stated. Concretely, in 30-views evaluation, 10 clips are uniformly sampled temporally from each video first. Each clip is cropped out using LeftCenterRight cropping strategy [9], which covers the longer axis of the original clip. The final prediction of a video is obtained by averaging the predictions for all corps of each clip. More details can be found in our supplement.

5.2. Ablation Study

5.2.1 Searching for Attention Block

To ablate the importance of attention block search, we conduct an ablation study based on the original X3D-S [9] backbone by 1) Manually insert attention block in each

Model	Attention location S1	S2	S3	S4	Params (M)	FLOPS (G)	Accuracy (%) Top-1
X3D-S	-	-	-	-	3.4	1.96	72.9
Manual	2	-	-	-	3.5	2.33	72.6 (**−0.3**)
	-	2	-	-	3.6	2.56	73.0 (**+0.1**)
	-	-	2	-	3.7	2.39	72.8 (**−0.1**)
	-	-	2	3	5.4	2.92	73.0 (**+0.1**)
Automatic	Searched				4.1	2.44	**73.3** (**+0.4**)

Table 3: **Ablation in attention block insertion location**. The number k under column Sn denotes k attention blocks are uniformly placed in stage n.

stage; 2) Automatically search the suitable position of attention block through probabilistic search algorithm [4]. We then compare them with the original baseline (X3D-S) on Kinetics-400 testbed.

As shown in Table 3, the first row represents the X3D-S baseline, reaching 72.9% top-1 accuracy with 1.96 GFLOPs cost. The second row includes four different architectures, where the attention blocks are inserted to different positions, ranging from the shallow layer (stage 1) to the deep layer (stage 4). These limited manual attempts bring minor improvement (+**0.1%**) based on the X3D-S backbone and sometimes even hurt the performance (−**0.3%**), motivating us to apply the automatic tools (*e.g.*, NAS) to reap the performance benefits of various position choices. We finally adopted the probabilistic-based architecture search on discovering the suitable position of attention blocks, where a block-wise position search is conducted with the rest of the backbone being fixed. To this end, the searched architecture reaches 73.3% top-1 accuracy, surpassing the original X3D-S baseline by a notable margin (+**0.4%**) with a comparable FLOPs cost.

5.2.2 Evaluation of Fairness-Aware Search

The default channel search simply split the super kernel into N splits and sample the candidate by default order. As we discussed in Sec. 4.2, this naive selecting strategy will make some parts of the super kernel to be over-sampled and updated unfairly. Thus the discovered model will easily collapse to the candidate which is updated more frequently.

To better understand the effectiveness of the proposed fairness-aware sample strategy, we conduct experiments that only search for the output channel and expansion ratio by separately using these two sampling strategies, while other factors are fixed. As shown in Figure 4, the blue curve represents the FLOPs cost of discovered architectures during the search process by using the naive channel selection strategy, and the red curve represents the fairness-aware one. Given a target FLOPs, the naive channel selection strategy easily collapses to the smaller subnets, due to the smaller channel candidates are updated more frequently than others in each super kernel. In contrast, the fairness-aware strategy can discover a more suitable subnet that is closer to the target FLOPs cost. The collapsed discovered architecture, shows poor performance after evaluation, with a large margin (−**2.6%**) behind the well-discovered fairness-based one. Details are shown in Table 4.

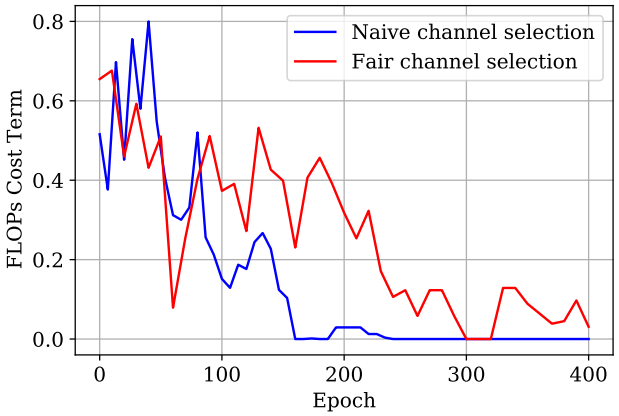


Figure 4: **FLOPs during searching.** For each epoch (x-axis), we derive the most probable architecture and measure its FLOPs (y-axis). The target FLOPs T (Equation 2) is set to 2.0 GFLOPs.

Search Strategy	Target FLOPs (G)	Discovered FLOPs (G)	Accuracy (%) Top-1
Naive Channel Selection	2.0	1.7(**−0.3**)	70.5 (**−2.6**)
Fair Channel Selection (**ours**)		**2.0**	**73.1**

Table 4: **Architecture search with different channel selection mechanisms.**

5.3. Comparing AutoX3D Models with Others

In this section, we compare our searched AutoX3D models with other SOTA models.

5.3.1 Results on Kinetics

We mainly follow the comparison convention of X3D [9] on popular large-scale benchmarks, and compare the performance of our AutoX3D with current state-of-the-art efficient architectures. The Top-1 accuracy as well as the FLOPS cost are reported on Kinetics-400 dataset. We divide the models into four categories according to their FLOPs cost, shown in Table 5. The first group includes previous high-performance yet inefficient methods. Those methods either cost more FLOPs and parameters compared to ours, or get worse performance. In the second group, the proposed AutoX3D-S architecture achieves 74.7% accuracy, better than TSM with mobilenet backbone (+**5.2%**), VoV3D-M (+**0.8**%), and X3D-S (+**1.4**%). Regarding FLOPs, AutoX3D-S costs ×**2** smaller FLOPs than TSM with mobilnet backbone, ×**1.5** smaller FLOPs than

Model	Pretrain	Top-1	Frames	GFLOPs×Views	Param.
I3D [3]	ImgNet	71.1	64	108 × N/A	12.0
2-Stream I3D [3]	ImgNet	75.7	64	216 × N/A	25.0
MF-Net [5]	ImgNet	72.8	16	11.1 × 50	8.0
TSM R50 [19]	ImgNet	74.7	16	65.0 × 10	24.3
S3D-G [43]	-	74.7	64	71.0 × N/A	N/A
2-Stream I3D [3]	-	71.6	64	216 × N/A	25.0
R(2+1)D [37]	-	72.0	32	152 × 115	63.6
2-Stream R(2+1)D [37]	-	73.9	32	304 × 115	127
ip-CSN-50 [35]	-	70.8	8	11.9 × 30	14.3
ip-CSN-101 [35]	-	71.8	8	15.9 × 30	24.5
TSM Mb-V2 [19]	ImgNet	69.5	16	6.0 × N/A	3.9
VoV3D-M [17]	-	73.9	16	4.4 × 30	3.8
X3D-S [9]	-	73.3	13	2.7 × 30	3.8
AutoX3D-S	-	**74.7**	13	2.9 × 30	3.5
MobileNetV2-3D (impl.)	-	71.2	16	6.5 × 30	4.4
SlowFast 4×16, R50 [10]	-	75.6	16	36.1 × 30	34.4
VoV3D-L [17]	-	76.3	16	9.3 × 30	6.2
X3D-M [9]	-	76.0	16	6.2 × 30	3.8
AutoX3D-M	-	**76.7**	16	6.8 × 30	3.5
SlowFast 8×8, R101 [10]	-	77.9	8	106 × 30	53.7
X3D-L	-	77.5	16	24.8 × 30	6.1
X3D-XL	-	79.1	16	48.4 × 30	11.1
AutoX3D-L	-	**78.8**	16	**27.8** × 30	6.2

Table 5: **Comparison with the state-of-the-art methods on Kinetics-400 dataset**. We use (GFLOPs × views) to represents inference cost × number of views).

VoV3D-M, and comparable FLOPs compared to X3D-S. In the third group, we evaluate a larger architecture, named AutoX3D-M, which is derived from AutoX3D-S by increase the input frame number to 16 and input video spatial resolution to 256. It achieves 76.7% accuracy, largely outperforms our implemented MobileNetV2-3D (+**5.5%**) and SlowFast 4x16 (+**1.1%**), respectively. Compared to X3D-M, AutoX3D-M obtains +**0.7%** higher accuracy with similar FLOPS cost. In the fourth group, we present our biggest architecture AutoX3D-L by scaling the network depth of AutoX3D-M to ×2 deeper and increasing the input video spatial resolution to 356. AutoX3D-L surpasses X3D-L by **1.3%** accuracy with similar FLOPs cost, and reaches comparable performance compared to X3D-XL with much smaller (×**1.74**) FLOPs cost.

5.3.2 Results on Something-Something-V2

In addition, we directly transfer the discovered architectures on Something-something v2 dataset. We include current state-of-the-art methods in Table 6, where AutoX3D model family achieves competitive performance **without** the necessity of pre-training. For example, AutoX3D-S achieves similar accuracy (62.1% vs 62.3%) with STM_{8F} while the FLOPs cost is ×**15** smaller. The Top-1 accuracy of AutoX3D-M is on par with SmallBigNet_{16F} (63.4% vs 63.8%), while the FLOPs cost is ×**20** fewer.

5.4. Understanding AutoX3D Architectures

Our searched AutoX3D architectures are visualized in Figure 5. Regarding to the design of the backbone network,

Model	Pretrain	Top-1	Frames	GFLOPs	Param. (M)
TSM_{8F} [19]	ImageNet	59.1	8	32.9	23.9
TSM_{16F} [19]		63.4	16	65.8	23.9
STM_{8F} [15]		62.3	8	33.3	24.0
STM_{16F} [15]		64.2	16	66.5	24.0
$\text{GST}_{8}F$ [21]		61.6	8	35.4	-
$\text{GST}_{1}6F$ [21]		62.6	16	35.4	-
I3D + STIN + OIE [23]		60.2	-	-	-
Dynamic Inference [41]		58.2	-	35.4	-
SmallBigNet_{8F} [18]		61.6	8	52	-
SmallBigNet_{16F} [18]		63.8	16	105	-
X3D-S	-	61.3	13	2.0	3.8
X3D-M		62.7	16	4.7	3.8
AutoX3D-S		62.1	13	**2.2**	3.5
AutoX3D-M		**63.4**	16	**5.3**	3.5

Table 6: **Comparison with the state-of-the-art methods on Something-Something-V2 dataset**.

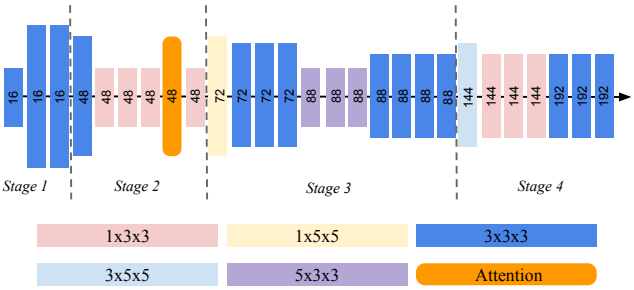


Figure 5: Visualization of the searched architecture. The number in each block represents the base channel of 3D MBConv block. The wider width of the block denotes the higher expansion ratio.

we find that 3×3^2 convolution is widely adopted, while temporal kernel size 5 is hardly used. We also observe that 5×5^2 convolution is never used in our network, due to its high complexity. Interestingly, We also find that the expansion ratio is often higher at the first block of each stage (except stage 1).

6. Conclusions and Future Work

In this paper, we introduce a family of efficient AutoX3D models, by directly searching without using any architecture surrogate nor hand-crafted image architecture. The proposed efficient architectures are discovered through a finer-grained search space, where block type, filter number, expansion ratio and attention block are jointly searched. In addition, a fairness-aware search strategy is adopted to avoid collapse issue. Our best architecture achieves state-of-the-art performance both on Kinetics-400 dataset and Something-something-V2 dataset, *e.g.*, +**1.3%** accuracy compared to X3D-L, and ×**1.74** lower computational cost compared to X3D-XL, on Kinetics-400 validation set.

References

[1] A. Adcock, V. Reis, M. Singh, Z. Yan, van der Maaten L., K. Zhang, S. Motwani, J. Guerin, N. Goyal, I. Misra, L. Gustafson, C. Changhan, and P. Goyal. Classy vision. 2019. 6

[2] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 1, 2, 3

[3] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 8

[4] Francesco Paolo Casale, Jonathan Gordon, and Nicolo Fusi. Probabilistic neural architecture search. *arXiv preprint arXiv:1902.05116*, 2019. 2, 3, 5, 6, 7

[5] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng. Multi-fiber networks for video recognition. In *Proceedings of the european conference on computer vision (ECCV)*, pages 352–367, 2018. 8

[6] Yunpeng Chen, Marcus Rohrbach, Zhicheng Yan, Yan Shuicheng, Jiashi Feng, and Yannis Kalantidis. Graph-based global reasoning networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 433–442, 2019. 4, 5

[7] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018. 6

[8] Jeffrey 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015. 2

[9] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 203–213, 2020. 1, 2, 3, 4, 6, 7, 8

[10] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 6202–6211, 2019. 3, 4, 8

[11] 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 preprint arXiv:1706.02677*, 2017. 6

[12] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *ICCV*, volume 1, page 5, 2017. 2, 6

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[14] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1314–1324, 2019. 3, 4

[15] Boyuan Jiang, MengMeng Wang, Weihao Gan, Wei Wu, and Junjie Yan. Stm: Spatiotemporal and motion encoding for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2000–2009, 2019. 8

[16] 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 preprint arXiv:1705.06950*, 2017. 2, 6

[17] Youngwan Lee, Hyung-Il Kim, Kimin Yun, and Jinyoung Moon. Diverse temporal aggregation and depthwise spatiotemporal factorization for efficient video classification. *arXiv preprint arXiv:2012.00317*, 2020. 8

[18] Xianhang Li, Yali Wang, Zhipeng Zhou, and Yu Qiao. Smallbignet: Integrating core and contextual views for video classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1092–1101, 2020. 8

[19] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7083–7093, 2019. 2, 8

[20] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 2, 3, 5

[21] Chenxu Luo and Alan L Yuille. Grouped spatial-temporal aggregation for efficient action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5512–5521, 2019. 8

[22] Pingchuan Ma, Yao Zhou, Yu Lu, and Wei Zhang. Learning efficient video representation with video shuffle networks. *arXiv preprint arXiv:1911.11319*, 2019. 2

[23] Joanna Materzynska, Tete Xiao, Roei Herzig, Huijuan Xu, Xiaolong Wang, and Trevor Darrell. Something-else: Compositional action recognition with spatial-temporal interaction networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1049–1059, 2020. 8

[24] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018. 1, 3

[25] AJ Piergiovanni, Anelia Angelova, and Michael S Ryoo. Tiny video networks. *arXiv preprint arXiv:1910.06961*, 2019. 2

[26] Ilija Radosavovic, Justin Johnson, Saining Xie, Wan-Yen Lo, and Piotr Dollár. On network design spaces for visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1882–1890, 2019. 3

[27] Carl Edward Rasmussen and Zoubin Ghahramani. Bayesian monte carlo. *Advances in neural information processing systems*, pages 505–512, 2003. 5

[28] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019. 1, 2, 3

[29] Michael S Ryoo, AJ Piergiovanni, Juhana Kangaspunta, and Anelia Angelova. Assemblenet++: Assembling modality representations via attention connections. *arXiv preprint arXiv:2008.08072*, 2020. 2

[30] Michael S Ryoo, AJ Piergiovanni, Mingxing Tan, and Anelia Angelova. Assemblenet: Searching for multi-stream neural connectivity in video architectures. *arXiv preprint arXiv:1905.13209*, 2019. 2

[31] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 2, 3, 4

[32] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019. 2

[33] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. 1

[34] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015. 1

[35] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5552–5561, 2019. 1, 8

[36] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018. 1, 2, 3

[37] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018. 8

[38] Xuanhan Wang, Lianli Gao, Peng Wang, Xiaoshuai Sun, and Xianglong Liu. Two-stream 3-d convnet fusion for action recognition in videos with arbitrary size and length. *IEEE Transactions on Multimedia*, 20(3):634–644, 2017. 1, 2

[39] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. 2, 4

[40] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019. 2

[41] Wenhao Wu, Dongliang He, Xiao Tan, Shifeng Chen, Yi Yang, and Shilei Wen. Dynamic inference: A new approach toward efficient video action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 676–677, 2020. 8

[42] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 305–321, 2018. 1, 2, 3

[43] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 305–321, 2018. 6, 8

[44] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. Bignas: Scaling up neural architecture search with big single-stage models. *arXiv preprint arXiv:2003.11142*, 2020. 1

[45] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702, 2015. 2

[46] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 3

[47] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018. 2, 3