Searching for Two-Stream Models in Multivariate Space for Video Recognition

Xinyu Gong[†‡*], Heng Wang[†], Zheng Shou[†], Matt Feiszli [†], Zhangyang Wang[‡], Zhicheng Yan[††]
[†]Facebook AI, [‡]The University of Texas at Austin

Abstract

Conventional video models rely on a single stream to capture the complex spatial-temporal features. Recent work on two-stream video models, such as SlowFast network and AssembleNet, prescribe separate streams to learn complementary features, and achieve stronger performance. However, manually designing both streams as well as the in-between fusion blocks is a daunting task, requiring to explore a tremendously large design space. Such manual exploration is time-consuming and often ends up with sub-optimal architectures when computational resources are limited and the exploration is insufficient. In this work, we present a pragmatic neural architecture search approach, which is able to search for two-stream video models in giant spaces efficiently. We design a multivariate search space, including 6 search variables to capture a wide variety of choices in designing two-stream models. Furthermore, we propose a progressive search procedure, by searching for the architecture of individual streams, fusion blocks and attention blocks one after the other. We demonstrate two-stream models with significantly better performance can be automatically discovered in our design space. Our searched two-stream models, namely Auto-TSNet, consistently outperform other models on standard benchmarks. On Kinetics, compared with the SlowFast model, our Auto-TSNet-L model reduces FLOPS by nearly $11\times$ *while achieving the same accuracy* 78.9%. *On Something-Something-V2, Auto-TSNet-M improves the accuracy by at least* 2% *over other methods which use less than 50 GFLOPS per video.*

1. Introduction

Video recognition requires to learn both spatial and temporal features, which is arguably more challenging than image recognition. Many efforts have been made to extend single-stream image architectures for video recognition, such as C3D [32], I3D [2], S3D [45], R(2+1)D [34], TSN [37], and TSM [16]. However, such single-stream models often underperform two-stream models where each stream takes a separate input and learns spatial-temporal representations that are complementary to each other [29, 9, 6]. In the pioneering two-stream ConvNet [29], a separate temporal stream is added which takes multi-frame optical flow as input to better learn temporal information. Recently, SlowFast network [9] adds a fast pathway, which operates at a high frame rate, and captures temporal information at a finer granularity.

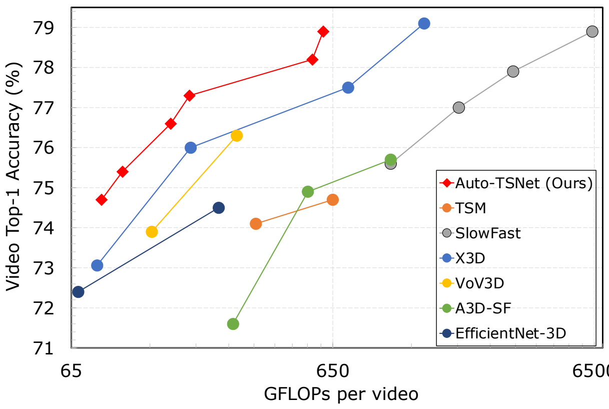


Figure 1: **Results on Kinetics-400.** Comparing the FLOPs and accuracy with state-of-the-art models, our Auto-TSNet models achieve better accuracy-to-complexity trade-off. For a fair comparison, we report the FLOPs for each video at inference time, taking into account the different number of views used by each method.

Compared with the single-stream model, the number of design choices grows exponentially for the two-stream models as we need to take into account the additional complexity from the second stream, the feature fusion between the streams. Prior hand-crafted two-stream models mitigate such challenges by largely reusing existing single-stream architectures, and only explore a limited number of customized design choices for each stream. For example, in two-stream ConvNet [29], the second temporal stream shares the same architecture as the spatial stream, which doubles the overall computational cost of the model. In the SlowFast network [9], the fast pathway only differs from the slow pathway by using 1D temporal convolutions and uniformly reducing the feature channels to balance the accuracy-to-complexity (ATC) trade-off. We hypothesize

*Work done during an internship at Facebook AI.
[†]Correspondence to Zhicheng Yan <zyan3@fb.com>.

that the existing two-stream models are sub-optimal, and pose the following question: *Can we more thoroughly explore the design space of two-stream video architectures and discover models with better performance?*

In this work, we present a pragmatic neural architecture search (NAS) approach, which can effectively explore large design spaces and discover high-performance two-stream models automatically. Unlike hand-crafted two-stream models where streams often use similar architectures, we encourage distinctive architectures at each stream, and jointly search both streams to learn complementary information. The core of our approach is a carefully prescribed multi-variate search space, which contains 6 search variables, including inter-stream fusion blocks, attention blocks, temporal/spatial kernel sizes, output channels, and expansion rates of building blocks. All of them have a substantial impact on both the accuracy and complexity of the learned model. Together they represent a wide variety of design choices for two-stream models.

It is computationally challenging to explore such gigantic search spaces efficiently. We propose *a multi-step progressive procedure* to decompose the large search space by only searching a smaller number of design choices at a time. For the basic search procedure, we adopt PARSEC [3] which is more memory efficient than other differential NAS methods by avoiding instantiating all choices of the search variables simultaneously, and only sampling one architecture at a time. Unlike the process of manually designing architectures which often favors uniform choices of search variables, searching in our space leads to the discovery of the *Auto-TSNet* models, which select *more nonuniform choices* for different components of the models. With extensive experiments, we demonstrate Auto-TSNet models substantially outperform others with more uniform choices on Kinetics-400 [13], shown in Figure 1, and Something-Something V2 [11] dataset.

Our main contributions are summarized below.

- We prescribe a multi-variate search space to accommodate the large variations in designing two-stream video models, by including 6 different search variables, each of which has a significant impact on the model accuracy and complexity.
- We decompose the search of two-stream models into multiple steps, and sequentially search different parts of the model, which renders the exploration in such a large space more efficient.
- The discovered Auto-TSNet models are distinct from the hand-crafted ones by selecting more nonuniform choices for different components. We evaluate them on two large action recognition benchmarks, and confirm their superior performance over other models.

2. Related Work

One-Stream Video Models. Video contains spatial-temporal signals and video recognition requires to extract both spatial and temporal features. One-stream video models, which are often built on top of image models, achieve such capability by various ways, such as replacing 2D with 3D convolution [32, 2], inserting 1D temporal convolution [34, 45], sampling temporal segments from the video [37], and shifting feature channels along temporal dimension [16].

From One-Stream to Two-Stream Video Models. Since the defining difference between video and image is video contains the temporal information between frames, a number of two-stream models are proposed where an extra stream is dedicated to capture more temporal information complementary to that from the existing stream [10, 38, 36, 43]. The Two-Stream ConvNet [29] augments the single-stream model by feeding optical flow to a separate 2D stream. Two-Stream Residual Network [6] improves it by introducing residual connections between streams. More recent SlowFast model [9] employs slow and fast pathways to capture spatial semantics and temporal motion separately.

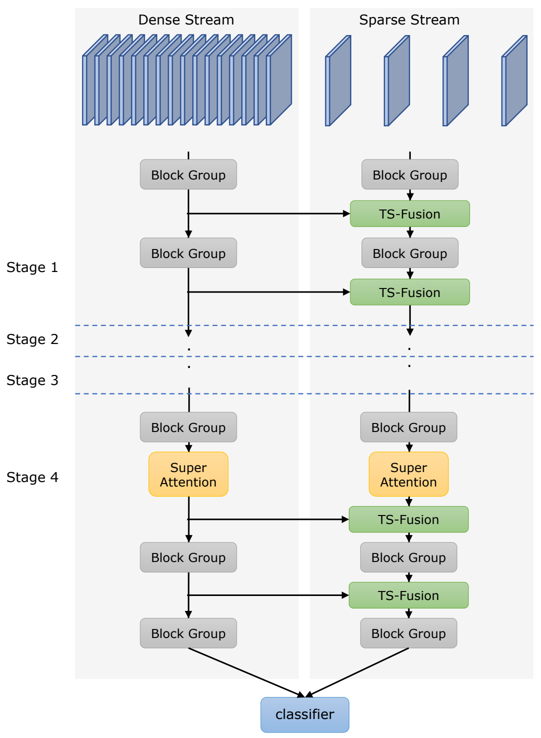
Neural Architecture Search. NAS methods automatically search models in a predefined space, and the searched 2D models has already surpassed the hand-crafted ones. NAS methods can be based on RL [30, 31, 49], evolution [24, 23], and differentiable search [17, 22, 3]. NAS has also been used to search video models. CAKES [46] searches channel-wise spatial/temporal kernels to improve model efficiency. X3D [8] gradually and uniformly expands a 2D model along 6 dimensions (*e.g.*, spatial resolution, model width) to derive efficient 3D models.

The design space of two-stream video models is significantly larger, and prior efforts have focused on searching fewer variables. AssembleNet [26] only searches the connections between streams, while keeping the architecture of building blocks fixed. AssembleNet++ [25] introduces the new object stream, but only searches the block connectivity for SqueezeExcite [12] module. Different from prior work, we search two-stream video models over 6 different variables, which is crucial for improving the ATC trade-off of the model. Our design space allows different architectures in individual streams and nonuniform design choices over different parts of the stream, and captures the variations in inter-stream fusion blocks, attention blocks, and the stream architectures. A progressive search process is proposed to efficiently search in such large spaces.

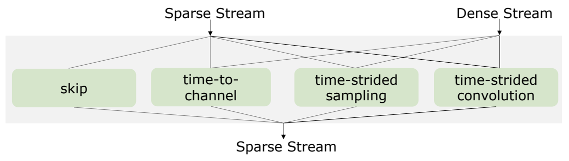
3. Two-Stream Multivariate Search Space

3.1. Overview

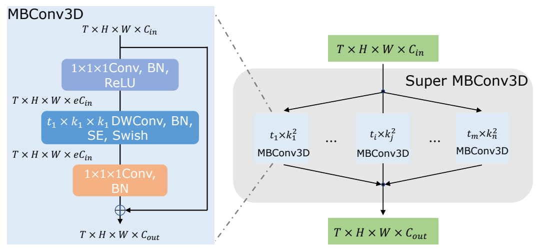
We define a multi-variate search space of two-stream models which use separate streams to capture complementary spatial-temporal signals. As shown in Figure 2a, a



(a) Multi-Variate Two-stream Super Model



(b) Super TS-Fusion Block


MBConv3D
$T \times H \times W \times C_{in}$
1×1×1Conv, BN, ReLU
$T \times H \times W \times eC_{in}$
$t_1 \times k_1 \times k_1$ DWConv, BN, SE, Swish
$T \times H \times W \times eC_{in}$
1×1×1Conv, BN
$T \times H \times W \times C_{out}$
$T \times H \times W \times C_{in}$
Super MBConv3D
$t_1 \times k_1^2$ MBConv3D
$t_i \times k_j^2$ MBConv3D
$t_m \times k_n^2$ MBConv3D
$T \times H \times W \times C_{out}$

(c) Super MBConv3D Block

Figure 2: **The multi-variate search space of two-stream models. (a)** The macro-architecture, where we define the layout of the various block groups in the model. **(b)** Super TS-Fusion block, where we search for the type of fusion operation. **(c)** Super MBConv3D block, where we search for the kernel size (t and k) of 3D depthwise convolution, the output channel C_{out}, and the expansion rate e.

sparse stream takes sparse frames as input which are sampled from the video with a larger temporal stride, while a *dense stream* takes dense frames with a smaller temporal stride. The proposed search space can be decomposed into three parts.

Fusion. To learn complementary signals between two streams, we allow to fuse features from two streams at different layers of the model. Rather than manual designing, we search for the fusion operations and their locations between the two streams.

Attention. Attention block has been used to improve the accuracy [39, 44, 40]. Previous work has shown that the design of the attention blocks and their placement locations are critical to the final performance. We propose to search for these design choices to yield more competitive models.

Backbone. The model backbone comprises stacked building blocks and represents a significant portion of the computational cost. In our work, we adopt a hierarchical backbone search space. As shown in Figure 2a, each stream of the network backbone includes 4 stages, and each stage has multiple block groups, where blocks in the same group share the same architecture.

3.2. Searchable Two-Stream Fusion Block

Early two-stream models [38, 37] do not fuse features from individual streams in the middle of the model. Recent SlowFast work [9] adds a fusion block to fuse features from both streams at the end of each stage, and uses the same type of fusion operation at all places. We hypothesize such uniform design choice is sub-optimal, and propose to explore the design choices of how many fusion blocks to add, where to add and what type of fusion to use. At the end of each group, we add a *super two-stream fusion block*, as shown in Figure 2b, where we search for the type of fusion operation for combining the features from two streams and passing the fused feature to the next block in the sparse stream. The candidate fusion operations are adopted from [9], and details are included in the supplement.

3.3. Searchable Attention Block

Attention blocks, such as Non-Local [39] and GloRe blocks [5], can be inserted into the backbone to improve the performance. In the case of manual design, one has to carefully decide where to add attention blocks and how many to add [39, 5]. Different choices have a large impact on both the model recognition performance and model FLOPS. Therefore, we search for the number and location of attention blocks, in order to achieve a better ATC trade-off.

In particular, we choose to use GloRe [5] as an instantiation of the attention block. Other attention blocks, such as non-local block [39], can also be used. In our search space, *searchable attention blocks* are placed between block groups, which is denoted as Super Attention in Figure 2a. We decide whether the features from each stream should simply pass through it or enter the attention block to perform global reasoning.

3.4. Searchable Backbone Building Block

We design a hierarchical search space, where the backbone is composed of block groups of different architec-

Max Input size $C \times T \times S^2$	Stage	Operator	Block group Channels	Expansion	N	S	Fusion Block	Attention Block
$3 \times 4 \times 224^2$	0	conv 1×3^2	24	-	1	2	-	-
$3 \times 4 \times 112^2$		conv 3×1^2	24		1	1		
$48 \times 4 \times 112^2$	1	MBConv3D	(32, 48, 8)	(1.5, 6.0, 0.75)	1	2	TS-Fusion	-
$48 \times 4 \times 56^2$			(32, 48, 8)		2	1		
$88 \times 4 \times 56^2$	2	MBConv3D	(64, 88, 8)	(1.5, 6.0, 0.75)	1	2	TS-Fusion	GloRe
$88 \times 4 \times 28^2$			(64, 88, 8)		4	1		
$176 \times 4 \times 28^2$	3	MBConv3D	(128, 176, 16)	(1.5, 6.0, 0.75)	1	2	TS-Fusion	GloRe
$176 \times 4 \times 14^2$			(128, 176, 16)		3	1		
$176 \times 4 \times 14^2$			(128, 176, 16)		3	1		
$176 \times 4 \times 14^2$			(128, 176, 16)		4	1		
$344 \times 4 \times 14^2$	4	MBConv3D	(248, 344, 24)	(1.5, 6.0, 0.75)	1	2	TS-Fusion	GloRe
$344 \times 4 \times 7^2$			(248, 344, 24)		3	1		
$344 \times 4 \times 7^2$			(248, 344, 24)		3	1		

(a) Sparse stream

Max Input size $C \times T \times S^2$	Stage	Operator	Block group Channels	Expansion	N	S	Attention Block
$3 \times 32 \times 224^2$	0	conv 1×3^2	8	-	1	2	-
$3 \times 32 \times 112^2$		conv 3×1^2	8		1	1	
$8 \times 32 \times 112^2$	1	MBConv3D	8	(1.5, 6.0, 0.75)	1	2	-
$16 \times 32 \times 56^2$			(8, 16, 8)		2	1	
$24 \times 32 \times 56^2$	2	MBConv3D	(8, 24, 8)	(1.5, 6.0, 0.75)	1	2	GloRe
$24 \times 32 \times 28^2$			(8, 24, 8)		4	1	
$32 \times 32 \times 28^2$	3	MBConv3D	(16, 32, 8)	(1.5, 6.0, 0.75)	1	2	GloRe
$32 \times 32 \times 14^2$			(16, 32, 8)		3	1	
$32 \times 32 \times 14^2$			(16, 32, 8)		3	1	
$32 \times 32 \times 14^2$			(16, 32, 8)		4	1	
$56 \times 32 \times 14^2$	4	MBConv3D	(32, 56, 8)	(1.5, 6.0, 0.75)	1	2	GloRe
$56 \times 32 \times 7^2$			(32, 56, 8)		3	1	
$56 \times 32 \times 7^2$			(32, 56, 8)		3	1	

(b) Dense stream

Table 1: **Two-stream macro architecture**. **(a)** The macro-architecture of the sparse stream. Each row represents a block group, which includes multiple MBConv3D blocks with the same architecture. It also has a searchable GloRe attention block and a TS-Fusion block at the end. Columns *Channel* and *Expansion* denote the output channel and the expansion rate of the block. Their search choices are denoted as a range *(min, max, step)*. Column *N* denotes the repeating times of the MBConv3D block, and column *S* is the spatial stride of first block in the group. **(b)** The macro-architecture of the dense stream.

ture (Figure 2a). Each group consists of several blocks, which are stacked sequentially and share the same architecture. In this work, we adopt a 3D version of MBConv (Mobile inverted Bottleneck Conv), originating from MobileNetV2 [27], namely **MBConv3D** (Figure 2c), which has 4 search variables, including temporal & spatial kernel size t & k, output channel C_{out} and expansion rate e.

Temporal kernel of depthwise convolution. Temporal kernel size has a large impact on the model FLOPS and accuracy. When it is set to 1, the MBConv3D block only has 2D convolution, which is computationally cheaper but is not able to capture temporal information. When it is larger than 1, the MBConv3D block is doing convolution in 3D, which is more costly but can improve the recognition performance by capturing temporal signals. We search the temporal kernel size for each block group. This not only avoids the tedious manual tuning but also improves the ATC trade-off.

Spatial kernel of depthwise convolution. Spatial kernel size is always considered to be critical for the model complexity and performance [30, 41]. In hand-crafted video models, such as I3D [2], S3D, SlowFast, and X3D, the spatial kernel is fixed to be 3 in almost all convolutional layers. We hypothesize such simple choice is sub-optimal, and add spatial kernel size to our search space as well.

Output channel of MBConv3D block. In X3D [8], which also adopts MBConv3D block as the building block, the output channel of each block is manually prescribed following simple heuristics, such as the output channel number doubles when the spatial resolution is reduced by half. As the choices of output channels substantially impact both the computation cost and capacity of the model, we search for the output channel from a wider range of choices.

Expansion rate of the MBConv3D block. A MBConv3D block expands the feature channel through a point-wise convolution by an expansion rate, performs 3D depth-wise convolution, and finally shrinks the feature channel through an-

Search variable	Temporal kernel	Spatial Kernel
Choices	$\{1, 3, 5\}$	$\{3, 5\}$

Table 2: **The choices of kernel size for MBConv3D block.**

other point-wise convolution. The expansion rate affects the number of feature channels where the depth-wise convolution operates, and consequently the ATC trade-off of the model. Previous models [27, 8] uses a constant expansion rate for all blocks. In contrast, we search for a separate expansion rate for each block.

3.5. The Final Search Space

The full specification of the macro architecture of our search space is shown in Table 1. Our search space has 6 variables (temporal & spatial kernel size, channel width, expansion rate, fusion, and attention block), where each block group can have its unique choice. It contains over 2×10^{53} architectures, and poses a great challenge to efficiently search architectures within it. The choices of temporal and spatial kernel sizes for the MBConv3D block are shown in Table 2.

4. Search Method

4.1. Background of Search Algorithm

We adopt the PARSEC [3] method as our basic search procedure, which is a probabilistic version of the differentiable NAS method DARTS [17]. Unlike DARTS, which requires to simultaneously instantiate all the layer choices and has a large memory footprint, PARSEC only samples one architecture at a time, and uses the same memory as in the standard model training.

PARSEC constructs a supernet where we can sample architectures $\{A_i\}$ according to a distribution $P(A|\boldsymbol{\alpha})$. Architecture parameters $\boldsymbol{\alpha}$ denote the probabilities of choosing different operations. Leveraging Importance Weighted Monte-Carlo algorithm [1], we jointly optimize $\boldsymbol{\alpha}$ and model weights of the supernet to maximize the data likelihood of sampled architectures, which are weighted by a proxy architecture performance indicator (video-level accuracy on validation set in our paper). We also add a hinge-type regularization term in the loss function to penalize architectures exceeding the target FLOPs range.

4.2. Progressive Search

Directly searching for the architecture of all parts of the two-stream model in our large search space is challenging. Therefore, we consider a divide-and-conquer strategy by breaking down the search process into multiple steps, and searching for different parts of the model sequentially. Empirically, we found such progressive procedure can accelerate searching efficiency without sacrifice on the performance of the final discovered architecture, compared to searching for all model parts simultaneously.

Step 1: We first search the architecture of the sparse stream, including temporal/spatial kernel, the output channel, and expansion rate of the MBConv3D blocks, which lives in a greatly reduced search space of size 8×10^{24}.

Step 2: After that, we fix the architecture of the sparse stream and inherit the model weight from step 1, and further search the architecture of the dense stream as well as the fusion blocks to optimize the performance of *the overall two-stream model*. Therefore, the search of dense stream and fusion blocks favors the architectures which are more capable of learning complementary features from two streams. The search space in this step contains 6×10^{24} unique architectures.

Step 3: In the final step, we search for the location to add the attention blocks which can improve the performance of the overall two-stream model at low computational cost, while keeping previous searched architecture fixed and inheriting their model weights. The search space in this step has a small size of 4096.

5. Experiments

5.1. Datasets

We use two large-scale video benchmarks Kinetics-400 [13] and Something-Something-V2 [11], which capture different aspects of video recognition tasks. *(i) Kinetics-400:* It contains 240K training- and 20K validation trimmed videos in 400 action classes, and focuses on general-purpose action recognition. *(ii) Something-Something-V2:* Unlike Kinetics, it decouples human actions and the objects involved in the actions, and forces the model to learn temporal information instead of recognizing objects. It contains 169K and 25K videos in the training and validation set from 174 human action classes. Our proposed Auto-TSNet can generalize to datasets of different characteristics and achieve superior performance. Following the standard protocol, we report the top-1/top-5 validation accuracy for both datasets.

5.2. Implementation details

We briefly introduce the searching, training and evaluation setup below, and include more details in the supplement.

Architecture Search We randomly select 100 classes from Kinetics-400 dataset, denoted as *MiniKinetics-100*, and search architectures on it for fast search. The architecture of sparse stream is searched for 800 epochs. The search for dense stream and the locations of attention blocks takes 400, 200 epochs respectively. Adam optimizer is adopted to update the architecture parameters, with learning rate 0.025 and zero weight decay. Model weights in the supernet are optimized with SGD, which uses learning rate 0.4 with a schedule of cosine decaying.

The sparse and dense stream take 4 and 32 frames as input, respectively. We use a scale jittering range of [182, 228] and then take a random crop of size 160×160 from each frame of the input video.

Training the searched models. After the search, we take the most probable architecture and *train it from scratch with model weights randomly initialized*. We train the model for 300 epochs. We use SGD optimizer, and learning rate 0.4 with a schedule of cosine decaying.

Evaluating the searched models. The trained model is evaluated on the validation set. We uniformly sample 10 clips from each video, and use two different ways of taking crops to obtain results comparable to those from prior work. *(1) 10-Center:* we resize the clip to have a short edge 182. A single central crop of size 160^2 is taken. *(2) 10-LeftCenterRight:* we take 3 crops of size 182^2 to cover the longer axis of the clip. The predictions is averaged over all crops of the clips. By default, our results are obtained by 10-Center crop testing, unless explicitly stated. With regard to the comparison of method complexity, we consider to use total FLOPs (FLOPs per video) as the main metric.

5.3. Main Results

Here we present the main results of our Auto-TSNet models in Table 3, including S, M and L variants which stand for small, medium, and large model. We also compare them with other state-of-the-art video models whose architectures are searched. Auto-TSNet-S and Auto-TSNet-M shared the same architecture, and only differ in the input video spatial resolution (182^2 vs 256^2). Auto-TSNet-L model is obtained by naively stretching the depth of Auto-TSNet-S by $2\times$, and increasing the input spatial resolution to 356^2. We also report results of another three variants of our model, namely Auto-TSNet-S†, Auto-TSNet-M† and Auto-TSNet-L†, where the attention blocks are removed.

In the first section of Table 3 where we compare small models, Auto-TSNet-S† outperforms X3D-S by a significant margin of **1.3%**, while using similar FLOPs. When at-

Model	Params (M)	GFLOPS ×views	Total GFLOPs	Top-1 Acc (%)
X3D-S [8]	3.8	2.7 × 30	81	73.3[1]
Auto-TSNet-S†(ours)	7.7	2.8 × 30	84	74.6
Auto-TSNet-S (ours)	8.6	3.4 × 30	102	**75.4**
EfficientNet3D-B3 [31]	8.2	6.9 × 10	69	72.4
X3D-M [8]	3.8	6.2 × 30	186	76.0
Auto-TSNet-M†(ours)	7.7	5.2 × 30	156	76.6
Auto-TSNet-M (ours)	8.6	6.1 × 30	183	**77.3**
EfficientNet3D-B4 [31]	12.2	23.8 × 10	238	74.5
X3D-L [8]	6.1	24.8 × 30	744	77.5
Auto-TSNet-L†(ours)	12.2	18.1 × 30	543	78.3
Auto-TSNet-L (ours)	13.2	19.9 × 30	597	**78.9**

Table 3: **Comparisons with other NAS models on Kinetics-400.** Auto-TSNet and X3D models are evaluated using 10-LeftCenterRight testing. † denotes the model without attention blocks.

Model	FLOPS Ratio P	Top-1 Acc (%)
X3D-S	-	72.9
Manual-TSNet	85%	72.8
	70%	**73.2**
	55%	72.4

Table 4: **Comparing X3D-S model with our Manual-TSNet models on Kinetics-400.** All models use about 2.0G FLOPS. Ratio P denotes the ratio of FLOPS used by the sparse stream.

tention blocks are searched and added, Auto-TSNet-S further improves the accuracy by **0.8%**. In the second section where we compare medium-size models, Auto-TSNet-M improves X3D-M by a large gap of **1.3%** using similar FLOPs. The performance of Auto-TSNet-M is even on par with X3D-L (77.3% Vs 77.5%), while using 66% less FLOPs. In the last section of Table 3 for comparing big models, Auto-TSNet-L significantly surpasses X3D-L by 1.4% (78.9% Vs 77.5%), while using **20%** less FLOPs.

5.4. From One-Stream to Two-Stream Model

X3D [8] is a family of single-stream models, and achieve good performance on standard benchmarks. However, we hypothesize two-stream models can achieve higher performance than single-stream models. Before automatically searching for two-stream models, we manually build two-stream baseline models and compare with X3D models.

We fix the number of input frames of sparse and dense stream to be 4 and 32. For each stream, we reuse the macro structure of the X3D-S model and only modify it by uniformly scaling down the feature channel at each block under the following constraints. *(i)*: the total FLOPS of the two-stream model per crop is close to X3D-s's FLOPs. *(ii)*: the FLOPS of the sparse stream account for $P\%$ of the overall FLOPS, where P is a hyper-parameter. Following the design in SlowFast [9], we use time-strided convolution as fusion block between 2 streams, which are placed uniformly along the network. We experiment with 3 choices of $P \in \{85, 70, 55\}$, and denote the resulting models as *Manual-TSNet-P*$\%$. The results are shown in Table 4. On Kinetics-400, the Manual-TSNet-70$\%$ model achieves the best performance, and improves the X3D-S model by 0.3$\%$ top-1 accuracy using similar FLOPS.

Note our hand-crafted Manual-TSNet models only represent a fairly sparse set of data points in our multi-variate search space, and are not expected to be optimal in that space. Nevertheless, Manual-TSNet-70$\%$ already outperforms the delicately designed X3D-S model, which encourages us to more extensively explore the space at finer granularity beyond simple uniform channel scaling.

[*]Accuracy reported in the official repo. X3D paper doesn't report the result of X3D-S using 10-LeftCenterRight testing.

5.5. Progressive Search of Two-Stream Models

5.5.1 Searching For Sparse Stream

As described in Section 4, we adopt a progressive search process that starts with searching the architecture of the sparse stream. We set the target FLOPS of the sparse stream to $1.4G$ FLOPS based on the design of our manually explored two-stream models Manual-TSNet-70% in Table 4. The search of the sparse stream takes 2.3 days. The searched sparse stream achieves 70.8$\%$ Top-1 accuracy on Kinetics-400, with 1.39G FLOPs, as shown in Table 5, which is a good starting point for our progressive search.

5.5.2 Searching For Dense Stream and Fusion Blocks

In the 2nd step of the progressive search, we fix the architecture of the sparse stream from the previous step as well as the model weights of the supernet, and further search for the architecture of the dense stream and TS-Fusion blocks. We set the target FLOPS of the overall two-stream model to 2.0G. The results are shown in Table 5. The discovered model includes sparse stream, dense stream, and fusion blocks. Compared with the searched sparse stream network from the previous step, the searched two-stream model achieves a performance boost of 3.3$\%$. The FLOPs of the searched model is 2.05 GFLOPs, which is close to the target 2 GFLOPs.

5.5.3 Searching for Attention Blocks

In the final step of the progressive search, we search for the insertion location of the attention block, where we choose to use GloRe as an instance of attention block. We consider inserting GloRe blocks at stage 2, 3, and 4. We uniformly pick 6 locations at stage 2, 3, and 4 for each stream as the candidate attention locations, which leads to a space of $2^6 \times 2^6 = 4096$ choices in total. We set the target FLOPs to 2.5G. The search only takes 0.9 days and the results are shown in the last row of Table 5.

The final searched Auto-TSNet model chooses to insert two GloRe blocks to the sparse stream of stage 3 (see Figure 3), which improves the accuracy of the searched archi-

# Streams	Sparse stream	Dense stream	Fusion block	Attention block	Search days	1-View GFLOPs	Top-1 Acc (%)
1	✓				2.3	1.39	70.8
2	✓	✓	✓		1.6	2.05	74.1 (+3.3)
2	✓	✓	✓	✓	0.9	2.46	74.6 (+0.5)

Table 5: **Results of progressive architecture search.**

Design Method	Search Days	1-View GFLOPs	Top-1 Acc (%)
Manual	-	1.40	70.2
Searched	2.3	1.39	**70.8**

Table 6: **Ablation on backbone search on Kinetics-400.**

Design Method	Search Days	Total Fusion Blocks	Fusion Blocks per Stage S1	S2	S3	S4	Top-1 Acc (%)
Manual	-	0	-	-	-	-	70.7
		5	+1	+1	+2	+1	73.2
		10	+2	+2	+4	+2	71.2
Searched	0.7	4	+1	+1	+2	-	**73.7**

Table 7: **Evaluating the choices of fusion locations on Manual-TSNet on Kinetics-400.**

tecture of step 2 by 0.5% (74.6% Vs 74.1%). The FLOPs of the searched architecture is also closed to our target FLOPs.

5.6. Ablation of Searchable Components

In this section, we consider isolating each searchable component (backbone, fusion, and attention), to verify the necessity of searching for their architectures.

Searching for Backbone Only. In the first step of progressive search of Auto-TSNet, we searched for a sparse stream, whose target FLOPs is set to 1.4 GFLOPS. We directly compare the performance of the searched sparse stream with the sparse stream of Manual-TSNet-70% in Table 6, which shows that the searched sparse stream outperforms the hand-crafted one by 0.6% top-1 accuracy (70.2% vs 70.8%).

Searching for Fusion Only. We conduct a toy study where we only search for the fusion location between two streams with pre-defined architectures, identical to those in Manual-TSNet-70%. The candidate fusion operations include time-strided-conv and no-connection. We create 3 variant models based on the backbone architecture of Manual-TSNet-70% with different number of fusion blocks, which are 0, 5 and 10 respectively. The candidate searchable fusion locations fully cover the fusion location of Manual-TSNet-70% and its two variants. The results are in Table 7. We observe the searched architecture has surpassed other manually designed baselines with a considerable accuracy boost 0.5%, demonstrating fusion search is non-trivial.

Searching for Attention Only. To demonstrate the need of searching attention blocks, we also conduct an experiment where we only search for the location of attention block, while keeping other searchable components fixed (identical to Manual-TSNet-70%). We adopt Manual-TSNet-70% as the baseline, along with its variants with different insertions

Design Method	Search Days	Use GloRe	GloRe per Stage S2	S3	S4	Params (M)	1-View GFLOPs	Top-1 Acc (%)
Manual-TSNet-70%	-	×	-	-	-	7.71	2.03	73.2
Manual	-	✓	+2	-	-	8.03	2.19	72.7
			+2	+2	-	8.24	2.41	73.2
			+2	+2	+2	8.37	2.51	73.3
Searched	0.9	✓	+1	+1	-	8.15	2.40	**73.6**

Table 8: **Comparing manually inserted and searched attention blocks in Manual-TSNet-70% model on Kinetics-400.**

Progressive search	Search days	End entropy	Params (M)	GFLOPs ×views	Top-1 Acc (%)
×	6.5	18.9	8.9	3.41×30	72.8
✓	4.8	9.7	8.6	3.25×30	75.4

Table 9: **Comparisons of one-step search and progressive search on Kinetics-400.** Models are evaluated using 10-LeftCenterRight testing. The entropy of progressive search is averaged across the search for each part.

of attention blocks. Results in Table 8 confirm the network with searched attention location outperforms other manual designed variations. Compared to Manual-TSNet-70%, the searched network obtains an accuracy boost of 0.4%.

5.7. Ablation of Progressive Search

In our paper, we adopted a progressive search algorithm to decompose the tremendous search space (2×10^{53}) into several small sub search space ($[8 \times 10^{24}, 6 \times 10^{24}, 4.096 \times 10^{3}]$), for more efficient search. We compare the searching time of progressive search and one-step search, as well as their search performances. Here one-step search denotes searching all variates simultaneously, instead of searching part by part in a progressive way. The only difference between them is that the one-step search algorithm search for all variables at once, and we keep other settings to be identical for fairness. We also introduce the entropy of architecture parameters $\boldsymbol{\alpha}$ as a convergence indicator for the search process. A lower entropy value indicates the searching is closer to convergence. The results are shown in Table 9. We confirm the progressive search mechanism reduces the search time by 26%, and also converges better (18.9 vs 9.7 in entropy). The searched architecture of progressive search also achieves better performance, outperforming the non-progressive one by 2.6%.

5.8. Comparisons with SOTA on Kinetics-400

In Table 10, we compare Auto-TSNet models with state-of-the-art results using *10-LeftCenterRight* testing setting. Compared with X3D-S, Auto-TSNet-S outperforms it with a significant margin of **2.4%**. Using similar FLOPs, Auto-TSNet-M surpasses X3D-M by **1.3%**. The performance of Auto-TSNet-M is even on par with X3D-L, while using 66% less FLOPs. Auto-TSNet-L surpasses X3D-L by a considerable gap of **1.4%**, while using **20%** fewer FLOPs. The performance of Auto-TSNet-L is even close

# streams	Model	Pre-training	Params (M)	Total GFLOPs	Top-1 Acc (%)
1	I3D [2]	ImageNet	12	-	71.1
	MF-Net [4]		8.0	555	72.8
	TSM [16]		24.3	650	74.7
	Nonlocal, R50 [39]		35.3	8,460	76.5
	Nonlocal, R101		54.3	10,770	77.7
	Oct-I3D+NL		33.6	867	75.7
	SmallBigNet [15]		-	5,016	77.4
1	R(2+1)D [34]	-	63.6	17,480	72.0
	ip-CSN-152 [33]		32.8	30,270	77.8
	VoV3D-M [14]		3.8	132	73.9
	VoV3D-L		6.2	279	76.3
	X3D-S [8]		3.8	75	73.3[1]
	X3D-M		3.8	186	76.0
	X3D-L		6.1	744	77.5
	X3D-XL		11.0	1,452	79.1
2	2-Stream I3D [2]	ImageNet	25	-	75.7
	2-Stream S3D-G[45]		23.1	-	77.2
	2-Stream TSN [37]		-	-	73.9
	2-Stream ARTNet, R18 [35]		-	-	72.4
2	2-Stream R(2+1)D[34]	-	127.2	34,960	73.9
	A3D-SF 4×16, R50 [48]		34.4	1,083	75.7
	SF 4×16, R50 [9]		34.4	1,083	75.6
	SF 8×8, R101		53.7	3,180	77.9
	SF 16×8, R101		53.7	6,390	78.9
2	**Auto-TSNet-S (ours)**	-	8.6	102	**75.4**
	Auto-TSNet-M (ours)		8.6	183	**77.3**
	Auto-TSNet-L (ours)		13.2	597	**78.9**

Table 10: **Comparing Auto-TSNet models with others on Kinetics-400.**

to that of X3D-XL (78.9% vs 79.1%), but saves FLOPS by 60%. Comparing with SF 16 × 8 R101, Auto-TSNet-L achieves the same performance with it, but it is incredibly **11**× smaller than SF 16 × 8 R101 w.r.t. FLOPs cost.

5.9. Transferability on Something-Something-V2

The next question we try to answer is: *Does the searched Auto-TSNet models overfit the dataset?* In other words, would the same model attain high performance, when we train it from scratch on a different dataset? To address this question, We further evaluate the searched Auto-TSNet models on Something-Something-V2 dataset. As mentioned before, the characteristic of Something-Something-V2 dataset is quite distinct from Kinetics-400, which focuses more on temporal modeling. The results are shown in Table 11, where we choose state-of-the-art efficient models under 50 GFLOPs to compare. Auto-TSNet models achieve highly competitive performance **without any pre-training**. Comparing with GST$_{8F}$, Auto-TSNet-S shows a significant performance gain of 0.7%, while using 20% fewer FLOPs. Auto-TSNet-M further boosts the accuracy to 63.6%, and outperform all other methods.

5.10. Visualization of Auto-TSNet Model

We visualize the discovered Auto-TSNet architecture in Figure 3. For backbone, we can observe that the spatial kernel size of 5 and temporal kernel size of 1 are widely used in both streams, which is quite different from the design in current hand-crafted models. Surprisingly, we notice that the attention blocks only appear on the sparse stream, indicating they are less effective on the dense stream. Unlike

# streams	Model	Pre-training	Params (M)	Total GFLOPs	Top-1 Acc (%)
1	TSN [37]	ImageNet	-	-	41.1
	TSN, Dual attention [44]		-	-	42.1
	TRN, Dual attention			-	51.6
	TRN [47]		-	-	48.8
	TSM, Dual attention		-	-	55.0
	I3D + STIN + OIE [21]		-	-	60.2
	Dynamic Inference [42]		-	35.4	58.2
	bLVNet-TAM [7]		40.2	32.1	60.2
	MSTNet [28]		24.3	33.2	59.5
	TANet [19]		30.4	33.0	60.5
	GST$_{8F}$ [20]		-	29.5	61.6
	TEINet [18]		-	33.0	61.3
2	2-Stream TRN [47]	ImageNet	-	-	55.5
	2-stream TRN, Dual attention [44]		-	-	58.4
2	**Auto-TSNet-S (ours)**	-	8.6	23.7	**62.3**
	Auto-TSNet-M (ours)		8.6	46.5	**63.6**

Table 11: **Comparing Auto-TSNet models with other efficient SOTA models (under 50 GFLOPs) on Something-Something-V2 dataset.**

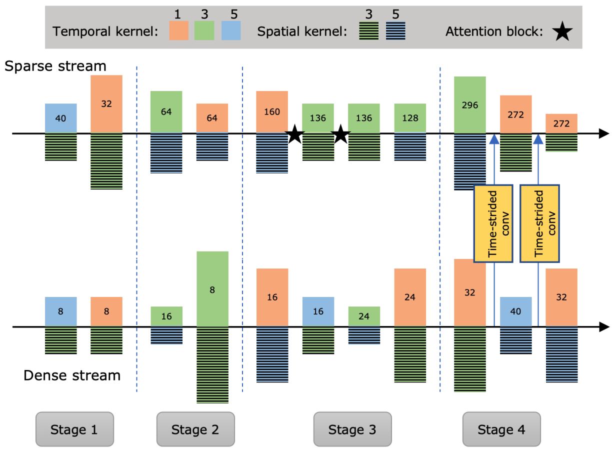


Figure 3: **The visualization of our searched Auto-TSNet-S**. Each rectangle box (except fusion blocks between the two streams) denotes a block group. We use height to denote the expansion rate, and the block output channel of the group is marked in the box. Different colors are used to denote kernel size, as shown in the legend. We use the shadow texture to distinguish spatial kernel size from temporal one. The star symbol is used to represent the attention block.

manual designed architectures which often favor the uniform choices of search variables, the searched model uses nonuniform architecture choices.

6. Conclusions

We present an approach to searching for high-performance two-stream models for video recognition. We meticulously prescribe a multivariate space with 6 search variables, which have a substantial impact on the model performance and complexity, and reflect the large variations in designing two-stream models. A progressive search process is proposed to efficiently search in the proposed large design space, and the discovered two-stream models outperform other models on two large-scale action recognition benchmarks.

References

[1] Bradley P Carlin and Thomas A Louis. Empirical bayes: Past, present and future. *Journal of the American Statistical Association*, 95(452):1286–1289, 2000.

[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.

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

[4] 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.

[5] 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.

[6] RPW Christoph and Feichtenhofer Axel Pinz. Spatiotemporal residual networks for video action recognition. *Advances in Neural Information Processing Systems*, pages 3468–3476, 2016.

[7] Quanfu Fan, Chun-Fu Chen, Hilde Kuehne, Marco Pistoia, and David Cox. More is less: Learning efficient video representations by big-little network and depthwise temporal aggregation. *arXiv preprint arXiv:1912.00869*, 2019.

[8] 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.

[9] 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.

[10] Christoph Feichtenhofer, Axel Pinz, and Richard P Wildes. Spatiotemporal multiplier networks for video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4768–4777, 2017.

[11] 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.

[12] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

[18] Zhaoyang Liu, Donghao Luo, Yabiao Wang, Limin Wang, Ying Tai, Chengjie Wang, Jilin Li, Feiyue Huang, and Tong Lu. Teinet: Towards an efficient architecture for video recognition. In *AAAI*, pages 11669–11676, 2020.

[19] Zhaoyang Liu, Limin Wang, Wayne Wu, Chen Qian, and Tong Lu. Tam: Temporal adaptive module for video recognition. *arXiv preprint arXiv:2005.06803*, 2020.

[20] 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.

[21] 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.

[22] Jieru Mei, Yingwei Li, Xiaochen Lian, Xiaojie Jin, Linjie Yang, Alan Yuille, and Jianchao Yang. Atomnas: Fine-grained end-to-end neural architecture search. *arXiv preprint arXiv:1912.09640*, 2019.

[23] 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.

[24] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc Le, and Alex Kurakin. Large-scale evolution of image classifiers. *arXiv preprint arXiv:1703.01041*, 2017.

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

[26] 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.

[27] 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.

[28] Kaiyu Shan, Yongtao Wang, Zhuoying Wang, Tingting Liang, Zhi Tang, Ying Chen, and Yangyan Li. Mixtconv:

Mixed temporal convolutional kernels for efficient action recogntion. *arXiv preprint arXiv:2001.06769*, 2020.

[29] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

[30] 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.

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

[32] 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.

[33] 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.

[34] 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.

[35] Limin Wang, Wei Li, Wen Li, and Luc Van Gool. Appearance-and-relation networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1430–1439, 2018.

[36] Limin Wang, Yuanjun Xiong, Zhe Wang, and Yu Qiao. Towards good practices for very deep two-stream convnets. *arXiv preprint arXiv:1507.02159*, 2015.

[37] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision*, pages 20–36. Springer, 2016.

[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.

[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.

[40] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[41] 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019.

[42] 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.

[43] Zuxuan Wu, Yu-Gang Jiang, Xi Wang, Hao Ye, and Xiangyang Xue. Multi-stream multi-class fusion of deep networks for video classification. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 791–800, 2016.

[44] Tete Xiao, Quanfu Fan, Dan Gutfreund, Mathew Monfort, Aude Oliva, and Bolei Zhou. Reasoning about human-object interactions through dual attention networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3919–3928, 2019.

[45] 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.

[46] Qihang Yu, Yingwei Li, Jieru Mei, Yuyin Zhou, and Alan L Yuille. Cakes: Channel-wise automatic kernel shrinking for efficient 3d network. *arXiv preprint arXiv:2003.12798*, 2020.

[47] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 803–818, 2018.

[48] Sijie Zhu, Taojiannan Yang, Matias Mendieta, and Chen Chen. A3d: Adaptive 3d networks for video action recognition. *arXiv preprint arXiv:2011.12384*, 2020.

[49] 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.