# MoViNets: Mobile Video Networks for Efficient Video Recognition

Dan Kondratyuk*, Liangzhe Yuan, Yandong Li, Li Zhang, Mingxing Tan, Matthew Brown, Boqing Gong
Google Research
{dankondratyuk,lzyuan,yandongli,zhl,tanmingxing,mtbr,bgong}@google.com

## Abstract

*We present Mobile Video Networks (MoViNets), a family of computation and memory efficient video networks that can operate on streaming video for online inference. 3D convolutional neural networks (CNNs) are accurate at video recognition but require large computation and memory budgets and do not support online inference, making them difficult to work on mobile devices. We propose a three-step approach to improve computational efficiency while substantially reducing the peak memory usage of 3D CNNs. First, we design a video network search space and employ neural architecture search to generate efficient and diverse 3D CNN architectures. Second, we introduce the Stream Buffer technique that decouples memory from video clip duration, allowing 3D CNNs to embed arbitrary-length streaming video sequences for both training and inference with a small constant memory footprint. Third, we propose a simple ensembling technique to improve accuracy further without sacrificing efficiency. These three progressive techniques allow MoViNets to achieve state-of-the-art accuracy and efficiency on the Kinetics, Moments in Time, and Charades video action recognition datasets. For instance, MoViNet-A5-Stream achieves the same accuracy as X3D-XL on Kinetics 600 while requiring 80% fewer FLOPs and 65% less memory. Code is available at https://github.com/google-research/movinet.*

## 1. Introduction

Efficient video recognition models are opening up new opportunities for mobile camera, IoT, and self-driving applications where efficient and accurate on-device processing is paramount. Despite recent advances in deep video modeling, it remains difficult to find models that run on mobile devices and achieve high video recognition accuracy. On the one hand, 3D convolutional neural networks (CNNs) [65, 69, 19, 18, 52] offer state-of-the-art accuracy, but consume copious amounts of memory and computation. On the other hand, 2D CNNs [40, 76] require far fewer resources suitable for mobile and can run online using frame-by-frame prediction, but fall short in accuracy.

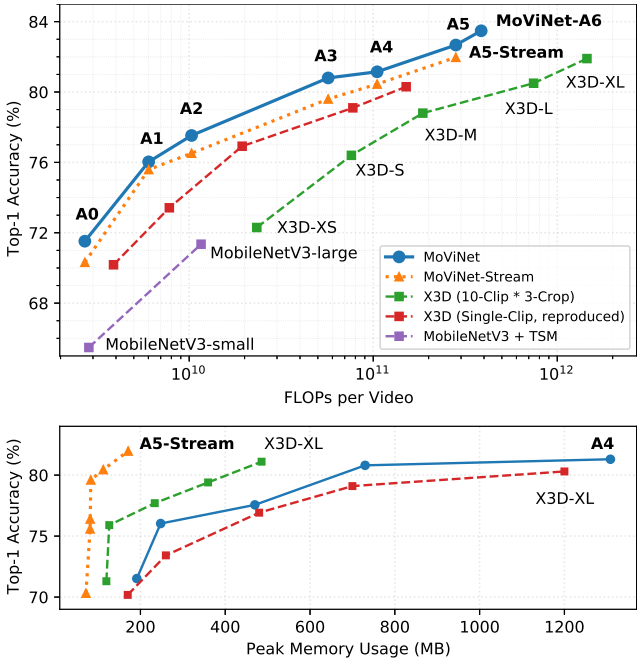


Figure 1. **Accuracy vs. FLOPs and Memory on Kinetics 600**. MoViNets are more accurate than 2D networks and more efficient than 3D networks. Top (log scale): MoViNet-A2 achieves **6% higher** accuracy than MobileNetV3 [26] at the same FLOPs while MoViNet-A6 achieves state-of-the-art 83.5% accuracy being **5.1x faster** than X3D-XL [18]. Bottom: Streaming MoViNets require **10x less memory** at the cost of 1% accuracy. Note that we only train on the 93% of Kinetics 600 examples that are available at the time of writing. Best viewed in color.

Many operations that make 3D video networks accurate (e.g., temporal convolution, non-local blocks [69], etc.) require all input frames to be processed at once, limiting the opportunity for accurate models to be deployed on mobile devices. The recently proposed X3D networks [18] provide a significant effort to increase the efficiency of 3D CNNs. However, they require large memory resources on large temporal windows which incur high costs, or small temporal windows which reduce accuracy. Other works aim

*Work done as a part of the Google AI Residency.

to improve 2D CNNs' accuracy using temporal aggregation [40, 17, 70, 43, 16], however their limited inter-frame interactions reduce these models' abilities to adequately model long-range temporal dependencies like 3D CNNs.

This paper introduces *three progressive steps* to design efficient video models which we use to produce Mobile Video Networks (**MoViNets**), a family of memory and computation efficient 3D CNNs.

1. We first define a **MoViNet search space** to allow Neural Architecture Search (NAS) to efficiently trade-off spatiotemporal feature representations.

2. We then introduce **Stream Buffers** for MoViNets, which process videos in small consecutive subclips, requiring constant memory without sacrificing long temporal dependencies, and which enable online inference.

3. Finally, we create **Temporal Ensembles** of streaming MoViNets, regaining the slightly lost accuracy from the stream buffers.

First, we design the MoViNet search space to explore how to mix spatial, temporal, and spatiotemporal operations such that NAS can find optimal feature combinations to trade-off efficiency and accuracy. Figure 1 visualizes the efficiency of the generated MoViNets. MoViNet-A0 achieves similar accuracy to MobileNetV3-large+TSM [26, 40] on Kinetics 600 [32] with 75% fewer FLOPs. MoViNet-A6 achieves state-of-the-art 83.5% accuracy, 1.6% higher than X3D-XL [18], requiring 60% fewer FLOPs.

Second, we create streaming MoViNets by introducing the stream buffer to reduce memory usage from linear to constant in the number of input frames for both training and inference, allowing MoViNets to run with substantially fewer memory bottlenecks. E.g., the stream buffer reduces MoViNet-A5's memory usage by 90%. In contrast to traditional multi-clip evaluation approaches [54, 67] which also reduce memory, a stream buffer carries over temporal dependencies between consecutive non-overlapping subclips by caching feature maps at subclip boundaries. The stream buffer allows for a larger class of operations to enhance online temporal modeling than the recently proposed temporal shift [40]. We equip the stream buffer with temporally unidirectional causal operations like causal convolution [46], cumulative pooling, and causal squeeze-and-excitation [27] with positional encoding to force temporal receptive fields to look only into past frames, enabling MoViNets to operate incrementally on streaming video for online inference. However, the causal operations come at a small cost, reducing accuracy on Kinetics 600 by 1% on average.

Third, we temporally ensemble MoViNets, showing that they are more accurate than single large networks while achieving the same efficiency. We train two streaming MoViNets independently with the same total FLOPs as a single model and average their logits. This simple technique gains back the loss in accuracy when using stream buffers.

Taken together, these three techniques create MoViNets that are high in accuracy, low in memory usage, efficient in computation, and support online inference. We search for MoViNets using the Kinetics 600 dataset [6] and test them extensively on Kinetics 400 [32], Kinetics 700 [7], Moments in Time [45], Charades [53], and Something-Something V2 [22].

## 2. Related Work

**Efficient Video Modeling.** Deep neural networks have made remarkable progress for video understanding [28, 54, 63, 68, 9, 69, 50, 18, 19]. They extend 2D image models with a temporal dimension, most notably incorporating 3D convolution [28, 62, 63, 72, 23, 49, 29, 52].

Improving the efficiency of video models has gained increased attention [19, 64, 20, 18, 40, 17, 3, 11, 37, 48]. Some works explore the use of 2D networks for video recognition by processing videos in smaller segments followed by late fusion [31, 15, 74, 68, 20, 58, 36, 39, 69, 75, 76]. The Temporal Shift Module [40] uses early fusion to shift a portion of channels along the temporal axis, boosting accuracy while supporting online inference.

**Causal Modeling.** WaveNet [46] introduces causal convolution, where the receptive field of a stack of 1D convolutions only extends to features up to the current time step. We take inspiration from other works using causal convolutions [8, 10, 13, 12, 14] to design stream buffers for online video model inference, allowing frame-by-frame predictions with 3D kernels.

**Multi-Objective NAS.** The use of NAS [77, 41, 47, 60, 5, 30] with multi-objective architecture search has also grown in interest, producing more efficient models in the process for image recognition [60, 5, 1] and video recognition [48, 52]. We make use of TuNAS [1], a one-shot NAS framework which uses aggressive weight sharing that is well-suited for computation intensive video models.

**Efficient Ensembles.** Deep ensembles are widely used in classification challenges to boost the performance of CNNs [4, 55, 59, 24]. More recent results indicate that deep ensembles of small models can be more efficient than single large models on image classification [33, 44, 56, 35, 21], and we extend these findings to video classification.

## 3. Mobile Video Networks (MoViNets)

This section describes our progressive three-step approach to MoViNets. We first detail the design space to search for MoViNets. Then we define the stream buffer and explain how it reduces the networks' memory footprints, followed by the temporal ensembling to improve accuracy.

| STAGE | NETWORK OPERATIONS | OUTPUT SIZE |
|---|---|---|
| data | stride $\tau$, RGB | $T \times S^2$ |
| $\text{conv}_1$ | $1 \times k_1^2$, $c_1$ | $T \times \frac{S^2}{2}$ |
| $\text{block}_2$ | $\left[ \begin{array}{c} k_2^{\text{time}} \times (k_2^{\text{space}})^2, c_2^{\text{base}}, c_2^{\text{expand}} \end{array} \right] \times L_2$ | $T \times \frac{S^2}{4}$ |
| | ... | |
| $\text{block}_n$ | $\left[ \begin{array}{c} k_n^{\text{time}} \times (k_n^{\text{space}})^2, c_n^{\text{base}}, c_n^{\text{expand}} \end{array} \right] \times L_n$ | $T \times \frac{S^2}{2^n}$ |
| $\text{conv}_{n+1}$ | $1 \times 1^2$, $c_{n+1}^{\text{base}}$ | $T \times \frac{S^2}{2^n}$ |
| $\text{pool}_{n+2}$ | $T \times \frac{S^2}{2^n}$ | $1 \times 1^2$ |
| $\text{dense}_{n+3}$ | $1 \times 1^2$, $c_{n+3}^{\text{base}}$ | $1 \times 1^2$ |
| $\text{dense}_{n+4}$ | $1 \times 1^2$, # classes | $1 \times 1^2$ |

Table 1. **MoViNet Search Space**. Given an input video with $T$ frames and resolution $S^2$, at stage $i$ we search over base widths $c_i^{\text{base}}$ and the number of layers $L_i$ in the block. Within each layer we search for expansion widths $c_i^{\text{expand}}$, along with 3D convolutional kernel sizes $k_i^{\text{time}} \times (k_i^{\text{space}})^2 \in \{1, 3, 5, 7\} \times \{1, 3, 5, 7\}^2$.

### 3.1. Searching for MoViNet

Following the practice of 2D mobile network search [60, 61], we start with the TuNAS framework [1], which is a scalable implementation of one-shot NAS with weight sharing on a supernetwork of candidate models, and repurpose it for 3D CNNs for video recognition. We use Kinetics 600 [32] as the video dataset to search over for all of our models, consisting of 10-second video sequences each at 25fps for a total of 250 frames.

**MoViNet Search Space.** We build our base search space on MobileNetV3 [26], which provides a strong baseline for mobile CPUs. It consists of several blocks of inverted bottleneck layers with varying filter widths, bottleneck widths, block depths, and kernel sizes per layer. Similar to X3D [18], we expand the 2D blocks in MobileNetV3 to deal with 3D video input. Table 1 provides a basic overview of the search space, detailed as follows.

We denote by $T \times S^2 = 50 \times 224^2$ and $\tau = 5$ (5fps) the dimensions and frame stride, respectively, of the input to the target MoViNets. For each block in the network, we search over the base filter width $c^{\text{base}}$ and the number of layers $L \leq 10$ to repeat within the block. We apply multipliers $\{0.75, 1, 1.25\}$ over the feature map channels within every block, rounded to a multiple of 8. We set $n = 5$ blocks, with strided spatial downsampling for the first layer in each block except the 4th block (to ensure the last block has spatial resolution $7^2$). The blocks progressively increase their feature map channels: $\{16, 24, 48, 96, 96, 192\}$. The final convolution layer's base filter width is 512, followed by a 2048D dense layer before the classification layer.

With the new time dimension, we define the 3D kernel size within each layer, $k^{\text{time}} \times (k^{\text{space}})^2$, to be chosen as one of the following: {1x3x3, 1x5x5, 1x7x7, 5x1x1, 7x1x1, 3x3x3, 5x3x3} (we remove larger kernels from consideration). These choices enable a layer to focus on and aggregate different dimensional representations, expanding the network's receptive field in the most pertinent directions while reducing FLOPs along other dimensions. Some kernel sizes may benefit from having different numbers of input filters, so we search over a range of bottleneck widths $c^{\text{expand}}$ defined as multipliers in $\{1.5, 2.0, 2.5, 3.0, 3.5, 4.0\}$ relative to $c^{\text{base}}$. Each layer surrounds the 3D convolution with two 1x1x1 convolutions to expand and project between $c^{\text{base}}$ and $c^{\text{expand}}$. We do not apply any temporal downsampling to enable frame-wise prediction.

Instead of applying spatial squeeze-and-excitation (SE) [27], we use SE blocks to aggregate spatiotemporal features via 3D average pooling, applying it to every bottleneck block as in [26, 61]. We allow SE to be searchable, optionally disabling it to conserve FLOPs.

**Scaling the Search Space.** Our base search space forms the basis for MoViNet-A2. For the other MoViNets, we apply a compound scaling heuristic similar to the one used in EfficientNet [61]. The major difference in our approach is that we scale the *search space* itself rather than a single model (i.e., search spaces for models A0-A5). Instead of finding a good architecture and then scaling it, we search over all scalings of all architectures, broadening the range of possible models.

We use a small random search to find the scaling coefficients (with an initial target of 300 MFLOPs per frame), which roughly double or halve the expected size of a sampled model in the search space. For the choice of coefficients, we resize the base resolution $S^2$, frame stride $\tau$, block filter width $c^{\text{base}}$, and block depths $L$. We perform the search on different FLOPs targets to produce a family of models ranging from MobileNetV3-like sizes up to the sizes of ResNet3D-152 [24, 23]. Appendix Aprovides more details of the search space, the scaling technique, and a description of the search algorithm.

The MoViNet search space gives rise to a family of versatile networks, which outperform state-of-the-art efficient video recognition CNNs on popular benchmark datasets. However, their memory footprints grow proportionally to the number of input frames, making them difficult to handle long videos on mobile devices. The next subsection introduces a stream buffer to reduce the networks' memory consumption from linear to constant in video length.

### 3.2. The Stream Buffer with Causal Operations

Suppose we have an input video $\boldsymbol{x}$ with $T$ frames that may cause a model to exceed a set memory budget. A common solution to reduce memory is multi-clip evaluation [54, 67], where the model averages predictions across $n$ overlapping subclips with $T^{\text{clip}} < T$ frames each, as seen in Figure 2 (left). It reduces memory consumption to $O(T^{\text{clip}})$. However, it poses two major disadvantages: 1) It limits the

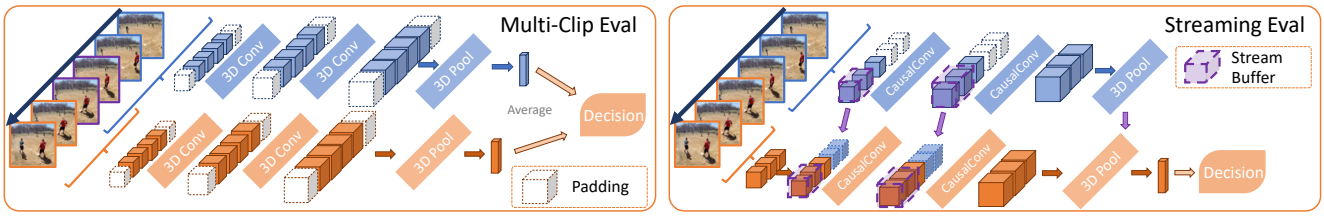


Figure 2. **Streaming Evaluation vs. Multi-Clip Evaluation**. In multi-clip evaluation, we embed overlapping subclips of an input video with 3D convolutions and average the logits. In streaming evaluation, we use the stream buffer to carry forward input features between non-overlapping subclips and apply causal operations, thereby allowing the temporal receptive field to cover the whole video. This stream buffer increases accuracy while retaining the benefits of reduced memory from multi-clip evaluation.

temporal receptive fields to each subclip and ignores long-range dependencies, potentially harming accuracy. 2) It recomputes frame activations which overlap, reducing efficiency.

**Stream Buffer.** To overcome the above mentioned limitations, we propose stream buffer as a mechanism to cache feature activations on the boundaries of subclips, allowing us to expand the temporal receptive field across subclips and requiring no recomputation, as shown in Figure 2 (right).

Formally, let $\boldsymbol{x}_i^{\text{clip}}$ be the current subclip (raw input or activation) at step $i < n$, where we split the video into $n$ adjacent *non-overlapping* subclips of length $T^{\text{clip}}$ each. We start with a zero-initialized tensor representing our buffer $B$ with length $b$ along the time dimension and whose other dimensions match $\boldsymbol{x}_i^{\text{clip}}$. We compute the feature map $F_i$ of the buffer concatenated ($\oplus$) with the subclip along the time dimension as:

$$F_i = f(B_i \oplus \boldsymbol{x}_i^{\text{clip}}) \tag{1}$$

where $f$ represents a spatiotemporal operation (e.g., 3D convolution). When processing the next clip, we update the contents of the buffer to:

$$B_{i+1} = (B_i \oplus \boldsymbol{x}_i^{\text{clip}})_{[-b:]} \tag{2}$$

where we denote $[-b:]$ as a selection of the last $b$ frames of the concatenated input. As a result, our memory consumption is dependent on $O(b + T^{\text{clip}})$, which is constant as the total video frames $T$ or number of subclips $n$ increase.

**Relationship to TSM.** The Temporal Shift Module (TSM) [40] can be seen as a special case of the stream buffer, where $b = 1$ and $f$ is an operation that shifts a proportion of channels in the buffer $B_t = \boldsymbol{x}_{t-1}$ to the input $\boldsymbol{x}_t$ before computing a spatial convolution at frame $t$.

### 3.2.1 Causal Operations

A reasonable approach to fitting 3D CNNs' operations to the stream buffer is to enforce *causality*, i.e., any features must not be computed from future frames. This has a number of advantages, including the ability to reduce a subclip $\boldsymbol{x}_i^{\text{clip}}$ down to a single frame without affecting activations or predictions, and enables 3D CNNs to work on streaming video for online inference. While it is possible to use non-causal operations, e.g., buffering in both temporal directions, we would lose online modeling capabilities which is a desirable property for mobile.

**Causal Convolution (CausalConv).** By leveraging the translation equivariant property of convolution, we replace all temporal convolutions with CausalConvs [46], effectively making them unidirectional along the temporal dimension. Concretely, we first compute padding to balance the convolution across all axes and then move any padding after the final frame and merge it with any padding before the first frame. See Appendix Cfor an illustration of how the receptive field differs from standard convolution, as well as a description of the causal padding algorithm.

When using a stream buffer with CausalConv, we can replace causal padding with the buffer itself, carrying forward the last few frames from a previous subclip and copying them into the padding of the next subclip. If we have a temporal kernel size of $k$ (and we do not use any strided sampling), then our padding and therefore buffer width becomes $b = k - 1$. Usually, $k = 3$ which implies $b = 2$, resulting in a small memory footprint. Stream buffers are only required before layers that aggregate features across multiple frames, so spatial and pointwise convolutions (e.g., 1x3x3, 1x1x1) can be left as-is, further saving memory.

**Cumulative Global Average Pooling (CGAP).** We use CGAP to approximate any global average pooling involving the temporal dimension. For any activations up to frame $T'$, we can compute this as a cumulative sum:

$$\text{CGAP}(\boldsymbol{x}, T') = \frac{1}{T'} \sum_{t=1}^{T'} \boldsymbol{x}_t, \tag{3}$$

where $\boldsymbol{x}$ represents a tensor of activations. To compute CGAP causally, we keep a single-frame stream buffer storing the cumulative sum up to $T'$.

**CausalSE with Positional Encoding.** We denote CausalSE as the application of CGAP to SE, where we multiply the spatial feature map at frame $t$ with the SE computed from $\text{CGAP}(\boldsymbol{x}, t)$. From our empirical results,

CausalSE is prone to instability likely due to the SE projection layers have a difficult time determining the quality of the CGAP estimate, which has high variance early in the video. To combat this problem, we apply a sine-based fixed positional encoding (PosEnc) scheme inspired by Transformers [66, 42]. We directly use frame index as the position and sum the vector with CGAP output before applying the SE projection.

#### 3.2.2 Training and Inference with Stream Buffers

**Training.** To reduce the memory requirements during training, we use a recurrent training strategy where we split a given batch of examples into $n$ subclips, applying a forward pass that outputs a prediction for each subclip, using stream buffers to cache activations. However, we do not backpropagate gradients past the buffer so that the memory of previous subclips can be deallocated. Instead, we compute losses and accumulate computed gradients between subclips, similar to batch gradient accumulation. This allows the network to account for all $T = nT^{\text{clip}}$ frames, performing $n$ forward passes before applying the gradients. This training strategy allows the network to learn longer term dependencies thus results in better accuracy than a model trained with shorter video length (see Appendix C).

We can set $T^{\text{clip}}$ to any value without affecting accuracy. However, ML accelerators (e.g., GPUs) benefit from multiplying large tensors, so for training we typically set a value of $T^{\text{clip}} \in \{8, 16, 32\}$. This accelerates training while allowing careful control of memory cost.

**Online Inference.** One major benefit of using causal operations like CausalConv and CausalSE is to allow a 3D video CNN to work online. Similar to training, we use the stream buffer to cache activations between subclips. However, we can set the subclip length to a single frame ($T^{\text{clip}} = 1$) for maximum memory savings. This also reduces the latency between frames, enabling the model to output predictions frame-by-frame on a streaming video, accumulating new information incrementally like a recurrent network (RNN) [25]. But unlike traditional convolutional RNNs, we can input a variable number of frames per step to produce the same output. For streaming architectures with CausalConv, we predict a video's label by pooling the frame-by-frame output features using CGAP.

### 3.3. Temporal Ensembles

The stream buffers can reduce MoViNets' memory footprints up to an order of magnitude in the cost of about 1% accuracy drop on Kinetics 600. We can restore this accuracy using a simple ensembling strategy. We train two MoViNets independently with the same architecture, but halve the frame-rate, keeping the temporal duration the same (resulting in half the input frames). We input a video into both networks, with one network having frames offset by one frame and apply an arithmetic mean on the unweighted logits before applying softmax. This method results in a two-model ensemble with the same FLOPs as a single model before halving the frame-rate, providing prediction with enriched representations. In our observations, despite the fact that both models in the ensemble may have lower accuracy than the single model individually, together when ensembled they can have higher accuracy than the single model.

## 4. Experiments on Video Classification

In this section, we evaluate MoViNets' accuracy, efficiency, and memory consumption during inference on five representative action recognition datasets.

**Datasets.** We report results on all Kinetics datasets, including Kinetics 400 [9, 32], Kinetics 600 [6], and Kinetics 700 [7], which contain 10-second, 250-frame video sequences at 25 fps labeled with 400, 600, and 700 action classes, respectively. We use examples that are available *at the time of writing*, which is 87.5%, 92.8%, and 96.2% of the training examples respectively (see Appendix C). Additionally, we experiment with Moments in Time [45], containing 3-second, 75-frame sequences at 25fps in 339 action classes, and Charades [53], which has variable-length videos with 157 action classes where a video can contain multiple class annotations. We include Something-Something V2 [22] results in Appendix C.

**Implementation Details.** For each dataset, all models are trained with RGB frames from scratch, i.e., we do not apply any pretraining. For all datasets, we train with 64 frames (except when the inference frames are fewer) at various frame-rates, and run inference with the same frame-rate.

We run TuNAS using Kinetics 600 and keep 7 MoViNets each having a FLOPs target used in [18]. As our models get larger, our scaling coefficients increase the input resolution, number of frames, depth, and feature width of the networks. For the architectures of the 7 models as well as training hyperparameters, see Appendix B.

**Single-Clip vs. Multi-Clip Evaluation.** We evaluate all our models with a single clip sampled from input video with a fixed temporal stride, covering the entire video duration. When the single-clip and multi-clip evaluations use the same number of frames in total so that FLOPs are equivalent, we find that single-clip evaluation yields higher accuracy (see Appendix C). This can be due in part to 3D CNNs being able to model longer-range dependencies, even when evaluating on many more frames than it was trained on. Since existing models commonly use multi-clip evaluation, we report the total FLOPs per video, not per clip, for a fair comparison.

However, single-clip evaluation can greatly inflate a network's peak memory usage (as seen in Figure 1), which is

likely why multi-clip evaluation is commonly used in previous work. The stream buffer eliminates this problem, allowing MoViNets to predict like they are embedding the full video, and incurs less peak memory than multi-clip evaluation.

We also reproduce X3D [18], arguably the most related work to ours, to test its performance under single-clip and 10-clip evaluation to provide more insights. We denote 30-clip to be the evaluation strategy with 10 clips times three spatial crops for each video, while 10-clip just uses one spatial crop. We avoid any spatial augmentation in MoViNets during inference to improve efficiency.

### 4.1. Comparison Results on Kinetics 600

**MoViNets without Stream Buffers.** Table 2 presents the main results of seven MoViNets on Kinetics 600 *before applying the stream buffer*, mainly compared with various X3D models [18], which are recently developed for efficient video recognition. The columns of the table correspond to the Top-1 classification accuracy; GFLOPs per video a model incurs; resolution of the input video frame (where we shorten $224^2$ to 224); input frames per video, where $30 \times 4$ means the 30-clip evaluation with 4 frames as input in each run; frames per second (FPS), determined by the temporal stride $\tau$ in the search space for MoViNets; and a network's number of parameters.

MoViNet-A0 has fewer GFLOPs and is 10% more accurate than the frame-based MobileNetV3-S [26] (where we train MobileNetV3 using our training setup, averaging logits across frames). MoViNet-A0 also outperforms X3D-S in terms of both accuracy and GFLOPs. MoViNet-A1 matches the GFLOPs of X3D-S, but its accuracy is 2% higher than X3D-S.

Growing the target GFLOPs to the range between X3D-S and 30-clip X3D-XS, we arrive at MoViNet-A2. We can achieve a little higher accuracy than 30-clip X3D-XS or X3D-M by using almost half of their GFLOPs. Additionally, we include the frame-by-frame MobileNetV3-L and verify that it can benefit from TSM [40] by about 3%.

There are more significant margins between larger MoViNets (A3–A6) and their counterparts in the X3D family. It is not surprising because NAS should intuitively be more advantageous over the handcrafting method for X3D when the design space is large. MoViNet-A5 and MoViNet-A6 outperform several state-of-the-art video networks (see the last 6 rows of Table 2). MoViNet-A6 achieves 83.5% accuracy (without pretraining) while still being substantially more efficient than comparable models. Even when compared to fully Transformer [66] models like TimeSformer-HR [2], MoViNet-A6 outperforms it by 1% accuracy and using 40% of the FLOPs.

**MoViNets with Stream Buffers.** Our base MoViNet architectures may consume lots of memory in the absence of modifications, especially as the model sizes and input frames grow. Using the stream buffer with causal operations, we can have an order of magnitude peak memory reduction for large networks (MoViNets A3-A6), as shown in the last column of Table 3.

| Model | Top-1 | GFLOPs | Res | Frames | FPS | Param |
|---|---|---|---|---|---|---|
| **MoViNet-A0** | **71.5** | **2.71** | 172 | 1×50 | 5 | 3.1M |
| MobileNetV3-S* [26] | 61.3 | 2.80 | 224 | 1×50 | 5 | 2.5M |
| MobileNetV3-S+TSM* [40] | 65.5 | 2.80 | 224 | 1×50 | 5 | 2.5M |
| X3D-XS* [18] | 70.2 | 3.88 | 182 | 1×20 | 2 | 3.8M |
| **MoViNet-A1** | **76.0** | **6.02** | 172 | 1×50 | 5 | 4.6M |
| X3D-S* [18] | 73.4 | 7.80 | 182 | 1×40 | 4 | 3.8M |
| X3D-S* [18] | 74.3 | 9.75 | 182 | 1×50 | 5 | 3.8M |
| **MoViNet-A2** | **77.5** | **10.3** | 224 | 1×50 | 5 | 4.8M |
| MobileNetV3-L* [26] | 68.1 | 11.0 | 224 | 1×50 | 5 | 5.4M |
| MobileNetV3-L+TSM* [40] | 71.4 | 11.0 | 224 | 1×50 | 5 | 5.4M |
| X3D-XS [18] | 72.3 | 23.3 | 182 | 30×4 | 2 | 3.8M |
| X3D-M* [18] | 76.9 | 19.4 | 256 | 1×50 | 5 | 3.8M |
| **MoViNet-A3** | **80.8** | **56.9** | 256 | 1×120 | 12 | 5.3M |
| X3D-S [18] | 76.4 | 76.1 | 182 | 30×13 | 4 | 3.8M |
| X3D-L* [18] | 79.1 | 77.5 | 356 | 1×50 | 5 | 6.1M |
| **MoViNet-A4** | **81.2** | **105** | 290 | 1×80 | 8 | 4.9M |
| X3D-M [18] | 78.8 | 186 | 256 | 30×16 | 5 | 3.8M |
| X3D-L* [18] | 80.7 | 187 | 356 | 1×120 | 2 | 6.1M |
| X3D-XL* [18] | 80.3 | 151 | 356 | 1×50 | 5 | 11.0M |
| I3D [6] | 71.6 | 216 | 224 | 1×250 | 25 | 12M |
| ResNet3D-50* | 78.7 | 390 | 224 | 1×250 | 25 | 34.0M |
| **MoViNet-A5** | **82.7** | **281** | 320 | 1×120 | 12 | 15.7M |
| X3D-L [18] | 80.5 | 744 | 356 | 30×16 | 5 | 6.1M |
| **MoViNet-A6** | **83.5** | **386** | 320 | 1×120 | 12 | 31.4M |
| TimeSformer-HR [2] | 82.4 | 645 | 224 | 3×8 | 1.5 | - |
| X3D-XL [18] | 81.9 | 1452 | 356 | 10×16 | 5 | 11.0M |
| ResNet3D-152* | 81.1 | 1400 | 224 | 1×250 | 25 | 80.1M |
| ResNet3D-50-G [38] | 82.0 | 3666 | 224 | 1×250 | 25 | - |
| SlowFast-R50 [19] | 78.8 | 1080 | 256 | 30×16 | 5 | 34.4M |
| SlowFast-R101 [19] | 81.8 | 7020 | 256 | 30×16 | 5 | 59.9M |
| LGD-R101 [50] | 81.5 | - | 224 | 15×16 | 25 | - |

Table 2. **Accuracy of MoViNet on Kinetics 600**. We measure total GFLOPs per video across all frames, and report the inference resolution (res), number of clips × frames per clip (frames), and frame rate (fps) of each video clip. * denotes our reproduced models. For X3D, we report *inference* resolution, which differs from training. We report all datapoints to the best knowledge available.

Moreover, Figure 3 visualizes the streaming architectures' effect on memory. From the left panel at the top, we see that our MoViNets are more accurate and more memory-efficient across all model sizes compared to X3D, which employs multi-clip evaluation. We also demonstrate constant memory as we scale the total number of frames in the input receptive field at the top's right panel. The bottom panel indicates that the streaming MoViNets remain efficient in terms of the GFLOPs per input video.

We also apply our stream buffer to ResNet3D-50 (see the last two rows in Table 3). However, we do not see as much of a memory reduction, likely due to larger overhead when using full 3D convolution as opposed to the depthwise

| Model | Top-1 | Res | Frames | FPS | GFLOPs | Mem (MB) |
|---|---|---|---|---|---|---|
| MobileNetV3-L* [26] | **68.1** | 224 | 1×50 | 5 | 11.0 | **23** |
| MoViNet-A0 | **71.5** | 172 | 1×50 | 5 | 2.71 | 173 |
| MoViNet-A0-Stream | 70.3 | 172 | 1×50 | 5 | 2.73 | **71** |
| MoViNet-A1 | **76.0** | 172 | 1×50 | 5 | 6.02 | 191 |
| MoViNet-A1-Stream | 75.6 | 172 | 1×50 | 5 | 6.06 | **72** |
| MoViNet-A1-Stream-Ens (x2) | 75.9 | 172 | 1×25 | 2.5 | 6.06 | **72** |
| MoViNet-A2 | **77.5** | 224 | 1×50 | 5 | 10.3 | 470 |
| MoViNet-A2-Stream | 76.5 | 224 | 1×50 | 5 | 10.4 | **85** |
| MoViNet-A2-Stream-Ens (x2) | 77.0 | 224 | 1×25 | 2.5 | 10.4 | **85** |
| MoViNet-A3 | 80.8 | 256 | 1×120 | 12 | 56.9 | 1310 |
| MoViNet-A3-Stream | 79.6 | 256 | 1×120 | 12 | 57.1 | **82** |
| MoViNet-A3-Stream-Ens (x2) | **80.4** | 256 | 1×60 | 6 | 57.1 | **82** |
| MoViNet-A4 | 81.2 | 290 | 1×80 | 8 | 105 | 1390 |
| MoViNet-A4-Stream | 80.5 | 290 | 1×80 | 8 | 106 | **112** |
| MoViNet-A4-Stream-Ens (x2) | **81.4** | 290 | 1×40 | 4 | 106 | **112** |
| MoViNet-A5 | 82.7 | 320 | 1×120 | 12 | 281 | 2040 |
| MoViNet-A5-Stream | 82.0 | 320 | 1×120 | 12 | 282 | **171** |
| MoViNet-A5-Stream-Ens (x2) | **82.9** | 320 | 1×60 | 6 | 282 | **171** |
| ResNet3D-50 | **78.7** | 224 | 1×250 | 25 | 390 | 3040 |
| ResNet3D-50-Stream | 76.9 | 224 | 1×250 | 25 | 390 | **2600** |
| ResNet3D-50-Stream-Ens (x2) | 78.6 | 224 | 1×125 | 12.5 | 390 | **2600** |

Table 3. **Base vs. Streaming Architectures** on Kinetics 600. We and report the inference resolution (res), number of clips × frames per clip (frames), and frame rate (fps) for each video. We measure the total GFLOPs per video across all frames. We denote "Stream" to be causal models using a stream buffer frame-by-frame, and "Ens" to be two ensembled models (with half the input frames so FLOPs are equivalent). Memory usage is measured in peak MB for a single video clip. * denotes our reproduced models.

convolution in MoViNets.

**MoViNets with Stream Buffers and Ensembling.** We see from Table 3 only a small 1% accuracy drop across all models after applying the stream buffer. We can restore the accuracy using the temporal ensembling without any additional inference cost. Table 3 reports the effect of ensembling two models trained at half the frame rate of the original model (so that GFLOPs remain the same). We can see the accuracy improvements in all streaming architectures, showing that ensembling can bridge the gap between streaming and non-streaming architectures, especially as model sizes grow. It is worth noting that, unlike prior works, the ensembling balances accuracy and efficiency (GFLOPs) in the same spirit as [33], not just to boost the accuracy.

## 4.2. Comparison Results on Other Datasets

Figure 4 summarizes the main results of MoViNets on all the five datasets along with state-of-the-art models that have results reported on the respective datasets. We compare MoViNets with X3D [18], MSNet [34], TSM [40], ResNet3D [24], SlowFast [19], EfficientNet-L2 [71], TVN [48], SRTG [57], and AssembleNet [52, 51]. Appendix Ctabulates the results with more details.

Despite only searching for efficient architectures on Kinetics 600, NAS yields models that drastically improve

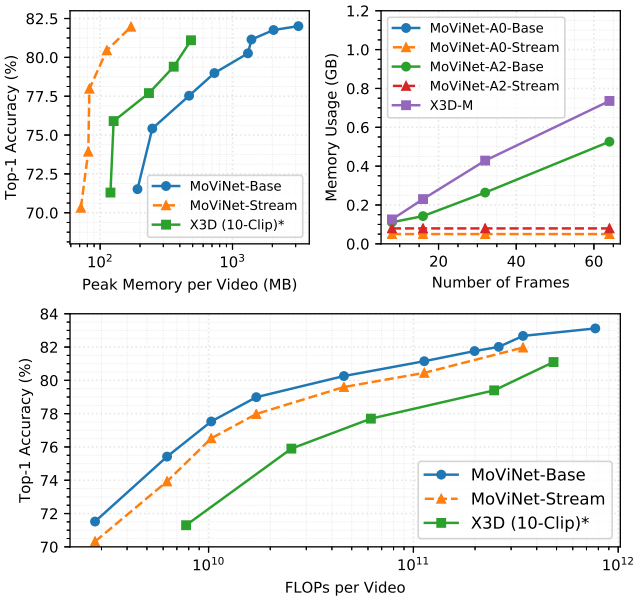


Figure 3. **Effect of Streaming MoViNets on Memory** on Kinetics 600. Top left: comparison of accuracy vs. max memory usage on a V100 GPU on our models, progressively increasing in size. We evaluate two versions of MoViNet: a base version without a stream buffer and a causal version with a stream buffer. Note that memory may be inflated due to padding and runtime overhead. Top right: comparison of max memory usage on a V100 GPU as a function of the number of input frames. Bottom: the classification accuracy. * denotes our reproduced models.

over prior work on other datasets as well. On Moments in Time, our models are 5-8% more accurate than Tiny Video Networks (TVNs) [48] at low GFLOPs, and MoViNet-A5 achieves 39.9% accuracy, outperforming AssembleNet [52] (34.3%) which uses optical flow as additional input (while our models do not). On Charades, MoViNet-A5 achieves the accuracy of 63.2%, beating AssembleNet++ [51] (59.8%) which uses optical flow and object segmentation as additional inputs. Results on Charades provide evidence that our models are also capable of sophisticated temporal understanding, as these videos can have longer duration clips than what is seen in Kinetics and Moments in Time.

## 4.3. Additional Analyses

**MoViNet Operations.** We provide some ablation studies about some critical MoViNet operations in Table 4. For the base network without the stream buffer, SE is vital for achieving high accuracy; MoViNet-A1's accuracy drops by 2.9% if we remove SE. We see a much larger accuracy drop when using CausalConv without SE than CausalConv with a global SE, which indicates that the global SE can take some of the role of standard Conv to extract information from future frames. However, when we switch to a fully streaming architecture with CausalConv and CausalSE, this

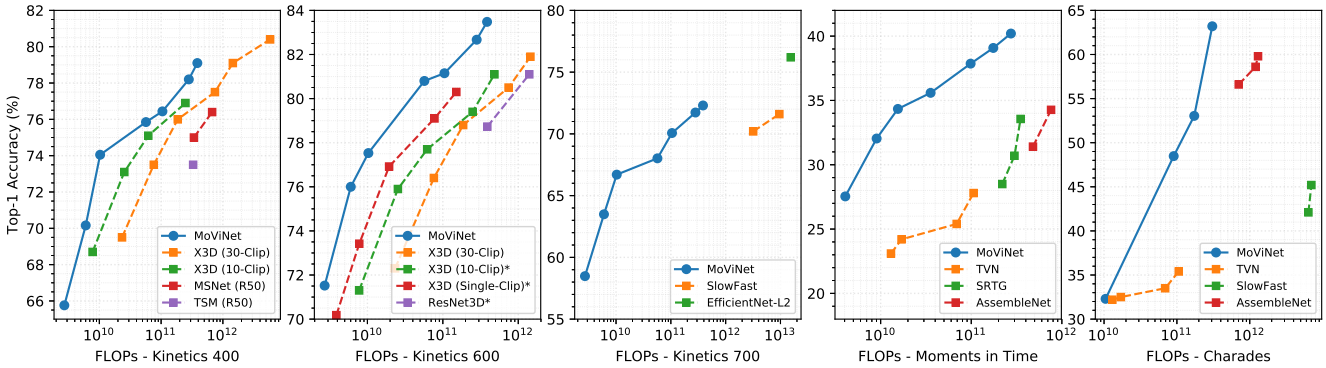


Figure 4. **Accuracy vs. FLOPs Comparison** across 5 large-scale action recognition datasets. Each series represents a model family, with points representing individual models ordered by FLOPs. We report FLOPs per video using single-clip evaluation for all MoViNets and compare with competitive multi-clip (and reproduced single-clip) models, using a log-scale on the x-axis. * denotes reproduced models. For Charades, we evaluate on MoViNet A1, A4, A5, and A6 only.

| Model | CausalConv | SE | CausalSE | PosEnc | Top-1 | GFLOPs |
|---|---|---|---|---|---|---|
| | | | | | 73.3 | **6.04** |
| | ✓ | | | | 72.1 | **6.04** |
| | ✓ | | ✓ | | 73.5 | 6.06 |
| MoViNet-A1 | ✓ | | ✓ | ✓ | 74.0 | 6.06 |
| | ✓ | ✓ | | | 74.9 | 6.06 |
| | | ✓ | | | **75.2** | 6.06 |
| | | | | | 77.7 | **56.9** |
| MoViNet-A3 | ✓ | | ✓ | | 79.0 | 57.1 |
| | ✓ | | ✓ | ✓ | 79.6 | 57.1 |
| | | ✓ | | | **80.3** | 57.1 |

Table 4. **MoViNet Operations Ablation** on Kinetics 600. We compare different configurations on MoViNet-A1, including Conv/CausalConv, SE/CausalSE/No SE, and PosEnc, and report accuracy and GFLOPs per video.

information from future frames is no longer available, and we see a large drop in accuracy, but still significantly improved from CausalConv without SE. Using PosEnc, we can gain back some accuracy in the causal model.

**MoViNet Architectures.** We provide the architecture description of MoViNet-A2 in Table 5 — Appendix Bhas the detailed architectures of other MoViNets. Most notably, the network prefers large bottleneck width multipliers in the range [2.5, 3.5], often expanding or shrinking them after each layer. In contrast, X3D-M with similar compute requirements has a wider base feature width with a smaller constant bottleneck multiplier of 2.25. The searched network prefers balanced 3x3x3 kernels, except at the first downsampling layers in the later blocks, which have 5x3x3 kernels. The final stage almost exclusively uses spatial kernels of size 1x5x5, indicating that high-level features for classification benefit from mostly spatial features. This comes at a contrast to S3D [73], which reports improved efficiency when using 2D convolutions at lower layers and 3D convolutions at higher layers.

**MoViNet Hardware Benchmarks.** For benchmarks running on real hardware, see Appendix C.

| Stage | Operation | Output size |
|---|---|---|
| data | stride 5, RGB | $50 \times 224^2$ |
| $conv_1$ | $1 \times 3^2$, 16 | $50 \times 112^2$ |
| $block_2$ | $\begin{bmatrix} 1\times5^2, 16, 40 \\ 3\times3^2, 16, 40 \\ 3\times3^2, 16, 64 \end{bmatrix}$ | $50 \times 56^2$ |
| $block_3$ | $\begin{bmatrix} 3\times3^2, 40, 96 \\ 3\times3^2, 40, 120 \\ 3\times3^2, 40, 96 \\ 3\times3^2, 40, 96 \\ 3\times3^2, 40, 120 \end{bmatrix}$ | $50 \times 28^2$ |
| $block_4$ | $\begin{bmatrix} 5\times3^2, 72, 240 \\ 3\times3^2, 72, 155 \\ 3\times3^2, 72, 240 \\ 3\times3^2, 72, 192 \\ 3\times3^2, 72, 240 \end{bmatrix}$ | $50 \times 14^2$ |
| $block_5$ | $\begin{bmatrix} 5\times3^2, 72, 240 \\ 3\times3^2, 72, 240 \\ 3\times3^2, 72, 240 \\ 3\times3^2, 72, 240 \\ 1\times5^2, 72, 144 \\ 3\times3^2, 72, 240 \end{bmatrix}$ | $50 \times 14^2$ |
| $block_6$ | $\begin{bmatrix} 5\times3^2, 144, 480 \\ 1\times5^2, 144, 384 \\ 1\times5^2, 144, 384 \\ 1\times5^2, 144, 480 \\ 1\times5^2, 144, 480 \\ 3\times3^2, 144, 480 \\ 1\times3^2, 144, 576 \end{bmatrix}$ | $50 \times 7^2$ |
| $conv_7$ | $1 \times 1^2$, 640 | $50 \times 7^2$ |
| $pool_8$ | $50 \times 7^2$ | $1 \times 1^2$ |
| $dense_9$ | $1 \times 1^2$, 2048 | $1 \times 1^2$ |
| $dense_{10}$ | $1 \times 1^2$, 600 | $1 \times 1^2$ |

Table 5. **MoViNet-A2 Architecture** searched by TuNAS, running 50 frames on Kinetics 600. See Table 1 for the search space definition detailing the meaning of each component.

## 5. Conclusion

MoViNets provide a highly efficient set of models that transfer well across different video recognition datasets. Coupled with stream buffers, MoViNets significantly reduce training and inference memory cost while also supporting online inference on streaming video. We hope our approach to designing MoViNets can provide improvements to future and existing models, reducing memory and computation costs in the process.

## References


[1] Gabriel Bender, Hanxiao Liu, Bo Chen, Grace Chu, Shuyang Cheng, Pieter-Jan Kindermans, and Quoc V Le. Can weight sharing outperform random architecture search? an investigation with tunas. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14323–14332, 2020. 2, 3

[2] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? *arXiv preprint arXiv:2102.05095*, 2021. 6

[3] Shweta Bhardwaj, Mukundhan Srinivasan, and Mitesh M Khapra. Efficient video classification using fewer frames. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 354–363, 2019. 2

[4] Yunlong Bian, Chuang Gan, Xiao Liu, Fu Li, Xiang Long, Yandong Li, Heng Qi, Jie Zhou, Shilei Wen, and Yuanqing Lin. Revisiting the effectiveness of off-the-shelf temporal modeling approaches for large-scale video classification. *arXiv preprint arXiv:1708.03805*, 2017. 2

[5] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018. 2

[6] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about kinetics-600. *arXiv preprint arXiv:1808.01340*, 2018. 2, 5, 6

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

[8] Joao Carreira, Viorica Patraucean, Laurent Mazare, Andrew Zisserman, and Simon Osindero. Massively parallel video networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 649–666, 2018. 2

[9] 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. 2, 5

[10] Shuo-Yiin Chang, Bo Li, Gabor Simko, Tara N Sainath, Anshuman Tripathi, Aäron van den Oord, and Oriol Vinyals. Temporal modeling using dilated convolution and gating for voice-activity-detection. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5549–5553. IEEE, 2018. 2

[11] Chun-Fu Chen, Quanfu Fan, Neil Mallinar, Tom Sercu, and Rogerio Feris. Big-little net: An efficient multi-scale feature representation for visual and speech recognition. *arXiv preprint arXiv:1807.03848*, 2018. 2

[12] Changmao Cheng, Chi Zhang, Yichen Wei, and Yu-Gang Jiang. Sparse temporal causal convolution for efficient action modeling. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 592–600, 2019. 2

[13] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019. 2

[14] Divyanshu Daiya, Min-Sheng Wu, and Che Lin. Stock movement prediction that integrates heterogeneous data sources using dilated causal convolution networks with attention. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8359–8363. IEEE, 2020. 2

[15] 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

[16] Linxi Fan, Shyamal Buch, Guanzhi Wang, Ryan Cao, Yuke Zhu, Juan Carlos Niebles, and Li Fei-Fei. Rubiksnet: Learnable 3d-shift for efficient video action recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 2

[17] Quanfu Fan, Chun-Fu Richard Chen, Hilde Kuehne, Marco Pistoia, and David Cox. More is less: Learning efficient video representations by big-little network and depthwise temporal aggregation. In *Advances in Neural Information Processing Systems*, pages 2264–2273, 2019. 2

[18] 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, 5, 6, 7

[19] 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. 1, 2, 6, 7

[20] 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. 2

[21] Tommaso Furlanello, Zachary C Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. *arXiv preprint arXiv:1805.04770*, 2018. 2

[22] 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 *Proceedings of the IEEE International Conference on Computer Vision*, pages 5842–5850, 2017. 2, 5

[23] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6546–6555, 2018. 2, 3

[24] 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. 2, 3, 7

[25] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 5

[26] 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 Confer-*

*ence on Computer Vision*, pages 1314–1324, 2019. 1, 2, 3, 6, 7

[27] 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. 2, 3

[28] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2012. 2

[29] 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. 2

[30] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Neural architecture search with bayesian optimisation and optimal transport. In *Advances in neural information processing systems*, pages 2016–2025, 2018. 2

[31] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014. 2

[32] 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, 3, 5

[33] Dan Kondratyuk, Mingxing Tan, Matthew Brown, and Boqing Gong. When ensembling smaller models is more efficient than single large models. *arXiv preprint arXiv:2005.00570*, 2020. 2, 7

[34] Heeseung Kwon, Manjin Kim, Suha Kwak, and Minsu Cho. Motionsqueeze: Neural motion feature learning for video understanding. In *European Conference on Computer Vision*, pages 345–362. Springer, 2020. 7

[35] Stefan Lee, Senthil Purushwalkam Shiva Prakash, Michael Cogswell, Viresh Ranjan, David Crandall, and Dhruv Batra. Stochastic multiple choice learning for training diverse deep ensembles. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, pages 2119–2127. Curran Associates, Inc., 2016. 2

[36] Dong Li, Zhaofan Qiu, Qi Dai, Ting Yao, and Tao Mei. Recurrent tubelet proposal and recognition networks for action detection. In *Proceedings of the European conference on computer vision (ECCV)*, pages 303–318, 2018. 2

[37] 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. 2

[38] Yinxiao Li, Zhichao Lu, Xuehan Xiong, and Jonathan Huang. Perf-net: Pose empowered rgb-flow net. *arXiv preprint arXiv:2009.13087*, 2020. 6

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

[40] 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. 1, 2, 4, 6, 7

[41] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018. 2

[42] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *Advances in Neural Information Processing Systems*, pages 9605–9616, 2018. 5

[43] Zhaoyang Liu, Limin Wang, Wayne Wu, Chen Qian, and Tong Lu. Tam: Temporal adaptive module for video recognition. *arXiv preprint arXiv:2005.06803*, 2020. 2

[44] Ekaterina Lobacheva, Nadezhda Chirkova, Maxim Kodryan, and Dmitry Vetrov. On power laws in deep ensembles. *arXiv e-prints*, pages arXiv–2007, 2020. 2

[45] Mathew Monfort, Alex Andonian, Bolei Zhou, Kandan Ramakrishnan, Sarah Adel Bargal, Tom Yan, Lisa Brown, Quanfu Fan, Dan Gutfruend, Carl Vondrick, et al. Moments in time dataset: one million videos for event understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–8, 2019. 2, 5

[46] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016. 2, 4

[47] 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. 2

[48] AJ Piergiovanni, Anelia Angelova, and Michael S Ryoo. Tiny video networks: Architecture search for efficient video models. 2020. 2, 7

[49] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *proceedings of the IEEE International Conference on Computer Vision*, pages 5533–5541, 2017. 2

[50] Zhaofan Qiu, Ting Yao, Chong-Wah Ngo, Xinmei Tian, and Tao Mei. Learning spatio-temporal representation with local and global diffusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12056–12065, 2019. 2, 6

[51] Michael S Ryoo, AJ Piergiovanni, Juhana Kangaspunta, and Anelia Angelova. Assemblenet++: Assembling modality representations via attention connections-supplementary material. 2020. 7

[52] 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. 1, 2, 7

[53] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in

homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, pages 510–526. Springer, 2016. 2, 5

[54] 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. 2, 3

[55] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[56] Saurabh Singh, Derek Hoiem, and David Forsyth. Swapout: Learning an ensemble of deep architectures. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, pages 28–36. Curran Associates, Inc., 2016. 2

[57] Alexandros Stergiou and Ronald Poppe. Learn to cycle: Time-consistent feature discovery for action recognition. *arXiv preprint arXiv:2006.08247*, 2020. 7

[58] Lin Sun, Kui Jia, Kevin Chen, Dit-Yan Yeung, Bertram E Shi, and Silvio Savarese. Lattice long short-term memory for human action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2147–2156, 2017. 2

[59] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 2

[60] 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, 3

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

[62] Graham W Taylor, Rob Fergus, Yann LeCun, and Christoph Bregler. Convolutional learning of spatio-temporal features. In *European conference on computer vision*, pages 140–153. Springer, 2010. 2

[63] 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. 2

[64] 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. 2

[65] 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

[66] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 5, 6

[67] Limin Wang, Yuanjun Xiong, Zhe Wang, and Yu Qiao. Towards good practices for very deep two-stream convnets. *arXiv preprint arXiv:1507.02159*, 2015. 2, 3

[68] 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. 2

[69] 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. 1, 2

[70] 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. 2

[71] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698, 2020. 7

[72] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning for video understanding. *arXiv preprint arXiv:1712.04851*, 1(2):5, 2017. 2

[73] 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. 8

[74] 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

[75] 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. 2

[76] Linchao Zhu, Du Tran, Laura Sevilla-Lara, Yi Yang, Matt Feiszli, and Heng Wang. Faster recurrent networks for efficient video classification. In *AAAI*, pages 13098–13105, 2020. 1, 2

[77] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 2