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.

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
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.
It reduces memory consumption to provide a basic 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 $L_i$ in the block. Within each layer, we search for expansion widths $c_{\text{expand}}$ along with 3D convolutional kernel sizes $k_i^{\text{time}} \times (k_i^{\text{space}})^2 \in \{1, 3, 5, 7\} \times \{1, 3, 5, 7\}^2$.

### Table 1. MoViNet Search Space

<table>
<thead>
<tr>
<th>STAGE</th>
<th>NETWORK OPERATIONS</th>
<th>OUTPUT SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>$\tau$, RGB</td>
<td>$T \times S^2$</td>
</tr>
<tr>
<td>conv1</td>
<td>$1 \times k_1^2, c_1$</td>
<td>$T \times \frac{S^2}{T}$</td>
</tr>
<tr>
<td>block2</td>
<td>$[k_2^{\text{time}} \times (k_2^{\text{space}})^2, c_2^{\text{base}} \times c_2^{\text{expand}}] \times L_2$</td>
<td>$T \times \frac{S^2}{T}$</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>block_n</td>
<td>$[k_n^{\text{time}} \times (k_n^{\text{space}})^2, c_n^{\text{base}} \times c_n^{\text{expand}}] \times L_n$</td>
<td>$T \times \frac{S^2}{T}$</td>
</tr>
<tr>
<td>conv_{n+1}</td>
<td>$1 \times 1^2, c_{n+1}^{\text{base}}$</td>
<td>$T \times \frac{S^2}{T}$</td>
</tr>
<tr>
<td>pool_{n+2}</td>
<td>$T \times \frac{S^2}{T}$</td>
<td>$1 \times 1^2$</td>
</tr>
<tr>
<td>dense_{n+3}</td>
<td>$1 \times 1^2, c_{n+3}^{\text{base}}$</td>
<td>$1 \times 1^2$</td>
</tr>
<tr>
<td>dense_{n+4}</td>
<td>$1 \times 1^2, # \text{classes}$</td>
<td>$1 \times 1^2$</td>
</tr>
</tbody>
</table>

### 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.

### Table 1. MoViNet Search Space

<table>
<thead>
<tr>
<th>STAGE</th>
<th>NETWORK OPERATIONS</th>
<th>OUTPUT SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>$\tau$, RGB</td>
<td>$T \times S^2$</td>
</tr>
<tr>
<td>conv1</td>
<td>$1 \times k_1^2, c_1$</td>
<td>$T \times \frac{S^2}{T}$</td>
</tr>
<tr>
<td>block2</td>
<td>$[k_2^{\text{time}} \times (k_2^{\text{space}})^2, c_2^{\text{base}} \times c_2^{\text{expand}}] \times L_2$</td>
<td>$T \times \frac{S^2}{T}$</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>block_n</td>
<td>$[k_n^{\text{time}} \times (k_n^{\text{space}})^2, c_n^{\text{base}} \times c_n^{\text{expand}}] \times L_n$</td>
<td>$T \times \frac{S^2}{T}$</td>
</tr>
<tr>
<td>conv_{n+1}</td>
<td>$1 \times 1^2, c_{n+1}^{\text{base}}$</td>
<td>$T \times \frac{S^2}{T}$</td>
</tr>
<tr>
<td>pool_{n+2}</td>
<td>$T \times \frac{S^2}{T}$</td>
<td>$1 \times 1^2$</td>
</tr>
<tr>
<td>dense_{n+3}</td>
<td>$1 \times 1^2, c_{n+3}^{\text{base}}$</td>
<td>$1 \times 1^2$</td>
</tr>
<tr>
<td>dense_{n+4}</td>
<td>$1 \times 1^2, # \text{classes}$</td>
<td>$1 \times 1^2$</td>
</tr>
</tbody>
</table>

### 3.2. Searching for MoViNet

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 $x$ 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

---

16022
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 $x_i^{clip}$ 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 $T_{clip}$ each. We start with a zero-initialized tensor representing our buffer $B$ with length $b$ along the time dimension and whose other dimensions match $x_i^{clip}$. We compute the feature map $F_i$ of the buffer concatenated ($\oplus$) with the subclip along the time dimension as:

$$F_i = f(B_i \oplus x_i^{clip})$$  \hspace{1cm} (1)

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

$$B_{i+1} = (B_i \oplus x_i^{clip})_{[-b:]}$$  \hspace{1cm} (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 $O(b + T_{clip})$, which is constant as the total video frames $T$ or number of subclips $n$ increase.

Relationship to TSM. The Temporal Shift Module (TSM) \cite{Wang2017} 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 $B_i = x_{i-1}$ to the input $x_i$ 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 $x_i^{clip}$ 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 \cite{Kornblith2019}, 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 C for 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:

$$\text{CGAP}(x, T') = \frac{1}{T'} \sum_{t=1}^{T'} x_t,$$  \hspace{1cm} (3)

where $x$ 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 CGAP($x, t$). From our empirical results,
CausalSE is prone to instability likely due to the SE projection layers having 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 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 2242 to 224); input frames per video, where 30 × 4 means the 30-clip evaluation with 4 frames as input in each run; frames per second (FPS), determined by the temporal stride τ 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.

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...
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 C tabulates the results with more details.

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

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

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.

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.
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 B has 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.

<table>
<thead>
<tr>
<th>Model</th>
<th>CausalConv SE CausalSE PosEnc</th>
<th>Top-1</th>
<th>GFLOPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoViNet-A1</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MoViNet-A3</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Operation</th>
<th>Output size</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>stride 5, RGB</td>
<td>50 x 224^2</td>
</tr>
<tr>
<td>conv1</td>
<td>1 x 3^2, 16</td>
<td>50 x 112^2</td>
</tr>
<tr>
<td>block2</td>
<td>1 x 3^2, 16, 40</td>
<td>50 x 56^2</td>
</tr>
<tr>
<td></td>
<td>3 x 3^2, 16, 40</td>
<td>3 x 3^2, 16, 64</td>
</tr>
<tr>
<td>block3</td>
<td>3 x 3^2, 40, 96</td>
<td>3 x 3^2, 40, 120</td>
</tr>
<tr>
<td></td>
<td>3 x 3^2, 40, 96</td>
<td>3 x 3^2, 40, 96</td>
</tr>
<tr>
<td>block4</td>
<td>3 x 3^2, 72, 240</td>
<td>3 x 3^2, 72, 192</td>
</tr>
<tr>
<td></td>
<td>3 x 3^2, 72, 155</td>
<td>3 x 3^2, 72, 144</td>
</tr>
<tr>
<td></td>
<td>3 x 3^2, 72, 192</td>
<td>3 x 3^2, 72, 240</td>
</tr>
<tr>
<td>block5</td>
<td>5 x 3^2, 144, 480</td>
<td>1 x 5^2, 144, 384</td>
</tr>
<tr>
<td></td>
<td>1 x 5^2, 144, 384</td>
<td>1 x 5^2, 144, 480</td>
</tr>
<tr>
<td></td>
<td>1 x 5^2, 144, 480</td>
<td>1 x 5^2, 144, 480</td>
</tr>
<tr>
<td></td>
<td>1 x 5^2, 144, 480</td>
<td>1 x 5^2, 144, 576</td>
</tr>
</tbody>
</table>

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


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


