[Supplementary Material] 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

Appendix

We supplement the main text by the following materials.

- **Appendix A** provides more details of the search space, the technique to scale the search space, and the search algorithm.
- **Appendix B** is about the neural architectures of MoViNets A0-A7.
- **Appendix C** reports additional results on the datasets studied in the main text along with ablation studies.

A. MoViNet Architecture Search

A.1 Scaling Algorithm for the Search Space

To produce models that scale well, we progressively expand the search space across width, depth, input resolution, and frame rate, like EfficientNet [25]. Specifically, we use a single scaling parameter ϕ to define the size of our search space. Then we define the following coefficients:

$$\begin{aligned} \text{depth: } d &= \alpha^\phi = 1.36^\phi \\ \text{base width: } w &= \beta^\phi = 1.18^\phi \\ \text{resolution: } r &= \gamma^\phi = 1.16^\phi \\ \text{frame-rate: } f &= \delta^\phi = 1.24^\phi \end{aligned}$$

such that $\alpha\beta^2\gamma^2\delta \approx 4$. This will ensure that an increase in ϕ by 1 will multiply the average model size in the search space by 4. Here we use a multiplier of 4 (instead of 2) to spread out our search spaces so that we can run the same search space with multiple efficiency targets and sample our desired target model size from it.

As a result, our parameters for a given search space is the following:

$$\begin{aligned} \text{depth: } L &\in \{d, \dots, 10d\} \\ \text{base width: } c^{\text{base}} &\in \{16w, 24w, 48w, 96w, 96w, 192w\} \\ \text{resolution: } S &= 224r \\ \text{frame-rate: } \tau &= 5f. \end{aligned}$$

We round each of the above parameters to the nearest multiple of 8. If $\phi = 0$, this forms the base search space for MoViNet-A2. Note that c^{expand} is defined relative to c^{base}, so we do not need coefficients for it.

We found coefficients $\alpha, \beta, \gamma, \delta$ using a random search over these parameters. More specifically, we select values in the range $[1.05, 1.40]$ at increments of 0.05 to represent possible values of the coefficients. We ensure that the choice of coefficients is such that $\alpha\beta^2\gamma^2\delta \approx 4$, where the initial computation target for each frame is 300 MFLOPs. For each combination, we scale the search space by the coefficients where $\phi = 1$, and randomly sample three architectures from each search space. We train models for a selected search space for 10 epochs, averaging the results of the accuracy for that search space. Then we select the coefficients that maximize the resulting accuracy. Instead of selecting a single set of coefficients, we average the top 5 candidates to produce the final coefficients. While the sample size of models is small and would be prone to noise, we find that the small averages work well in practice.

A.2 Search Algorithm

During search, we train a one-shot model using TuNAS [3] that overlaps all possible architectures into a hypernetwork. At every step during optimization, we alternate between learning the network weights and learning a policy π which we use to randomly sample a path through the hypernetwork to produce an initially random network architecture. π is learned using REINFORCE [26], optimized on the quality of sampled architectures, defined as the absolute reward consisting of the sampled network's accuracy and cost. At each stage, the RL controller must choose a

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

single categorical decision to select an architectural component. The network architecture is a result of binding a value to each decision. For example, the decision might choose between a spatial 1x3x3 convolution and a temporal 5x1x1 convolution. We use FLOPs as the cost metric for architecture search, and use Kinetics 600 as the dataset to optimize for efficient video networks. During search, we obtain validation set accuracies on a held-out subset of the Kinetics 600 training set, training for a total of 90 epochs.

The addition of SE [14] to our search space increases FLOPs by such a small amount ($< 0.1\%$) that the search enables it for all layers. SE plays a similar role as the feature gating in S3D-G [28], except with a nonlinear squeeze inside the projection operation.

A.3 Stream Buffers for NAS

We apply the stream buffers to MoViNets as a separate step after NAS in the main text. We can also leverage them for NAS to reduce memory usage during search. Memory poses one of the biggest challenges for NAS, as models are forced to use a limited number of frames and small batch sizes to be able to keep the models in memory during optimization. While this does not prevent us from performing search outright, it requires the use of accelerators with very high memory requirements, requiring a high cost of entry. To circumvent this, we can use stream buffers with a small clip size to reduce memory. As a result, we can increase the total embedded frames and increase the batch size to provide better model accuracy estimation while running NAS. Table 1 provides an example of an experiment where the use of a stream buffer can reduce memory requirements in this manner. Using a stream buffer, we can reduce the input size from a single clip of 16 frames to 2 clips of 8 frames each, and double the batch size. This results in a relatively modest increase in memory, compared to not using the buffer where we can run into out-of-memory (OOM) issues.

We note that the values of b in each layer influences the memory consumption of the model. This is dependent entirely on the temporal kernel width of the 3D convolution. If $k = 5$, then we only need to cache the last 4 frames. b could be larger, but but it will result in extra frames we will discard, so we set it to the smallest value to conserve memory. Therefore, it is not necessary to specify it directly with NAS, as NAS is only concerned with the kernel sizes. However, we can add an objective to NAS to minimize memory usage, which will apply pressure to reduce the temporal kernel widths and therefore will indirectly affect the value of b in each layer. Reducing memory consumption even further by keeping kernel sizes can be explored in future work.

CONFIG	FULL INPUT	BATCH SIZE	Memory	TOP-1
No Buffer	16×172^2	8	5.8 GB	55.9
Buffer (8 frames)	16×172^2	16	6.6 GB	58.5
No Buffer	16×172^2	16	10.4 GB	OOM

Table 1. **Effect of Stream Buffer on NAS (Kinetics 600)**. We measure the effect of NAS on MoViNet-A0 when using a stream buffer vs. without on the same input. By embedding half the input at a time, we can double the batch size to improve the average NAS held-out accuracy without significantly increasing GPU memory per device.

B. Architectures of MoViNets

See Tables 11, 12, 13, 14, 15, and 16 for the architecture definitions of MoViNet A0-A5 (we move the tables to the final pages of the Appendices to reduce clutter). For MoViNet-A6, we ensemble architectures A4 and A5 using the strategy described in the main text, i.e., we train both models independently and apply an arithmetic mean on the logits during inference. All layers of all models have SE layers enabled, so we remove this search hyperparameter from all tables for brevity.

B.1 More Details of the Architectures and Training

We apply additional changes to our architectures and model training to improve performance even further. To improve convergence speed in searching and training we use ReZero [2] by applying zero-initialized learnable scalar weights that are multiplied with features before the final sum in a residual block. We also apply skip connections that are traditionally used in ResNets, adding a 1x1x1 convolution in the first layer of each block which may change the base channels or downsample the input. However, we modify this to be similar to ResNet-D [12] where we apply 1x3x3 spatial average pooling before the convolution to improve feature representations.

We apply Polyak averaging [20] to the weights after every optimization step, using an Exponential Moving Average (EMA) with decay 0.99. We adopt the Hard Swish activation function, which is a variant of SiLU/Swish [9, 21] proposed by MobileNetV3 [13] that is friendlier to quantization and CPU inference. We use the RMSProp optimizer with momentum 0.9 and a base learning rate of 1.8. We train for 240 epochs with a batch size of 1024 with synchronized batch normalization on all datasets and decay the learning rate using a cosine learning rate schedule [17] with a linear warm-up of 5 epochs.

We use a softmax cross-entropy loss with label smoothing 0.1 during training, except for Charades where we apply sigmoid cross-entropy to handle the multiple-class labels per video. For Charades, we aggregate predictions across frames similar to AssembleNet [23], where we apply a soft-

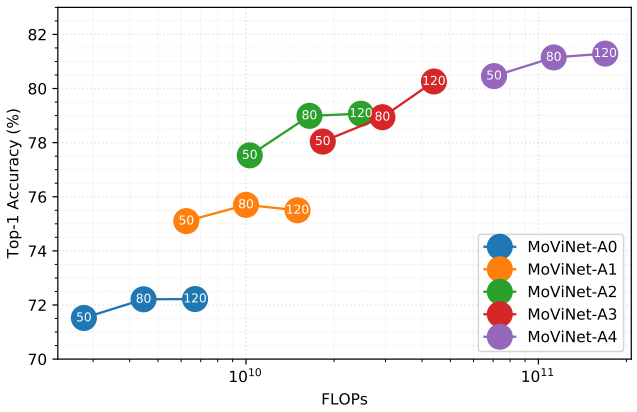

Figure 1. **Effect of Frame-Rate on Efficiency** on Kinetics 600. We train and evaluate each model with 50, 80, and 120 frames at 5fps, 8fps, and 12fps respectively.

max across frames before applying temporal global average pooling to find multiple action classes that may occur in different frames.

Some works also expand the resolution for inference. For instance, X3D-M trains with a 224^2 resolution while evaluating 256^2 when using spatial crops. We evaluate all of our models on the same resolution as training to make sure the FLOPs per frame during inference is unchanged from training.

Our choice of frame-rates can vary from model to model, providing different optimality depending on the architecture. We plot the accuracy of training various MoViNets on Kinetics 600 with different frame-rates in Figure 1. Most models have good efficiency at 50 frames (5fps) or 80 frames (8fps) per video. However, we can see MoViNet-A4 benefits from a higher frame-rate of 12fps. For Charades, we use 64 frames at 6fps for both training and inference.

C. More Implementation Details and Experiments

C.1 Hardware Benchmark

MoViNets A0, A1, and A2 represent the fastest models that would most realistically be used on mobile devices. We compare them with MobileNetV3 in Figure 2 with respect to both FLOPs and real-time latency on an x86 Intel Xeon W-2135 CPU at 3.70GHz. These models are comparable in per-frame computation cost, as we evaluate on 50 frames for all models. From these results we can conclude that streaming MoViNets can run faster on CPU while being more accurate at the same time, even with temporal modifications like TSM. While there is a discrepancy between FLOPs and latency, searching over a latency target explicitly in NAS can reduce this effect. However, we still see that FLOPs is a reasonable proxy metric for CPU latency, which would translate well for mobile devices.

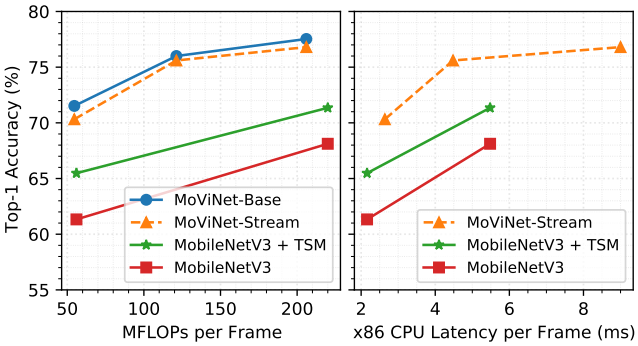

Figure 2. **CPU Latency Comparison** on Kinetics 600. We compare the efficiency of MoViNets (A0, A1, A2) vs. MobileNetV3 [13] using FLOPs and benchmarked latency on an x86 Xeon CPU at 3.70GHz.

Model	Streaming	Video (ms)	Frame (ms)	Top-1
MoViNet-A0-Stream	✓	183	3.7	70.3
MoViNet-A1-Stream	✓	315	6.3	75.6
MoViNet-A2-Stream	✓	325	6.5	76.5
MoViNet-A3-Stream	✓	1110	9.2	79.6
MoViNet-A4-Stream	✓	1130	14.1	80.5
MoViNet-A5-Stream	✓	2310	19.2	82.0
MobileNetV3-S*	✓	68	1.4	61.3
MobileNetV3-L*	✓	81	1.6	68.1
X3D-M* (Single Clip)	✗	345	6.9	76.9
X3D-XL* (Single Clip)	✗	943	18.9	80.3

Table 2. **Runtime on an Nvidia V100 GPU on Kinetics 600**. Latency is given for the entire video clip and per frame in ms.

We also show benchmarks for MoViNets running on an Nvidia V100 GPU in Table 2. Similar to mobile CPU, our streaming model latency is comparable to single-clip X3D models. However, we do note that MobileNetV3 can run faster than our networks on GPU, showing that the FLOPs metric for NAS has its limitations. MoViNets can be made more efficient by targeting real hardware instead of FLOPs, which we leave for future work.

C.2 Implementing Causal Convolution by Padding

To make a temporal convolution operation causal, we can apply a simple padding trick which shifts the receptive field forward such that the convolutional kernel is centered at the frame furthest into the future. Figure 3 illustrates this effect. With a normal 3D convolution operation with kernel size (k_t, k_h, k_w) and stride $s = 1$, the padding with respect to dimension i is given as:

$$p_i^{\text{left}}, p_i^{\text{right}} = \begin{cases} \left(\frac{k_i-1}{2}, \frac{k_i-1}{2}\right) & \text{if } x \text{ is odd} \\ \left(\frac{k_i-2}{2}, \frac{k_i}{2}\right) & \text{otherwise.} \end{cases} \quad (1)$$

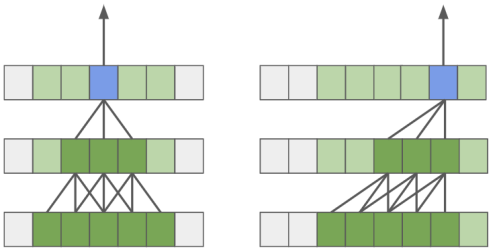

Figure 3. **Standard Convolution vs. Causal Convolution**. The figure illustrates the effective receptive field along a sequence of frames. The temporal kernel size is 3, with padding shown in white. Causal convolution can be performed efficiently by padding only on one side of the time axis thus to force the output causality.

DATASET	TRAIN	VALID	RELEASED
Kinetics 400	215,435 (87.5%)	17,686 (88.4%)	May 2017
Kinetics 600	364,305 (92.8%)	27,764 (92.5%)	Aug 2018
Kinetics 700	524,595 (96.2%)	33,567 (95.9%)	Jul 2019

Table 3. The number of examples available for each of the Kinetics dataset splits at the time of writing (Sept 20, 2020) along with the percentages compared to examples available on release. Each dataset loses about 4% of its examples per year.

where $p_i^{\text{left}}, p_i^{\text{right}}$ are the left and right padding amounts respectively. For causal convolutions, we transform p_t as:

$$p_t^{\text{left causal}}, p_t^{\text{right causal}} = (p_t^{\text{left}} + p_t^{\text{right}}, 0). \quad (2)$$

such that the effective temporal receptive field of a voxel at time position t only spans $(0, t]$.

C.3 Additional Details of Datasets

We note that for all the Kinetics datasets are gradually shrinking over time due to videos being taken offline, making it difficult to compare against less recent works. We report on the most recently available videos. While putting our work at a disadvantage compared to previous work, we wish to make comparisons more fair for future work. Nevertheless, we report the numbers as-is and report the reduction of examples in the datasets in Table 3.

We report full results of all models in the following tables: Table 4 for Kinetics 400, Table 5 for Kinetics 600 (with top-5 accuracy), Table 6 for Kinetics 700, Table 7 for Moments in Time, Table 8 for Charades, and Table 9 for Something-Something V2. For a table of results on Kinetics 600, see Table 5 and also the main text.

C.4 Single-Clip vs. Multi-Clip Evaluation

We report all of our results on a *single* view without multi-clip evaluation. Additionally, we report the total number of frames used for evaluation and the frame rate (note that the evaluation frames can exceed the total number of frames in the reference video when subclips overlap).

MODEL	TOP-1	GFLOPS	PARAM
MoViNet-A0	**65.8**	**2.71**	**3.1M**
MoViNet-A1	**70.2**	**6.02**	**4.6M**
MoViNet-A1-Stream	68.3	6.06	**4.6M**
MoViNet-A2	**74.1**	**10.3**	4.8M
X3D-XS [10]	69.5	23.3	**3.8M**
MoViNet-A3	**75.9**	**56.9**	5.3M
X3D-S [10]	73.5	76.1	**3.8M**
MoViNet-A4	**76.4**	**105**	5.9M
X3D-M [10]	**76.0**	**186**	**3.8M**
MoViNet-A5	**78.2**	**289**	15.7M
X3D-L [10]	77.5	744	**6.1M**
MoViNet-A6	**79.1**	**289**	31.4M
X3D-XL [10]	**79.1**	1452	**11.0M**
TimeSformer-HR [4]	**80.7**	**645**	-
X3D-XXL [10]	80.4	5600	**20.3M**

Table 4. **Accuracy of MoViNet on Kinetics 400**.

MODEL	TOP-1	Top-5	GFLOPS	Param
MoViNet-A0	**71.5**	**90.4**	**2.71**	**3.1M**
MoViNet-A0-Stream	70.3	90.1	2.73	**3.1M**
MoViNet-A1	**76.0**	92.6	**6.02**	**4.6M**
MoViNet-A1-Stream	75.6	**92.8**	6.06	**4.6M**
MoViNet-A2	**77.5**	**93.4**	**10.3**	**4.8M**
MoViNet-A2-Stream	76.5	93.3	10.4	**4.8M**
MoViNet-A3	**80.8**	**94.5**	**56.9**	**5.3M**
MoViNet-A4	**81.2**	**94.9**	**105**	4.9M
X3D-M [10]	78.8	94.5	186	**3.8M**
MoViNet-A5	**82.7**	**95.7**	**281**	15.7M
X3D-XL [10]	81.9	95.5	1452	**11.0M**
SlowFast-R50 [11]	78.8	94.0	1080	34.4M
SlowFast-R101 [11]	81.8	95.1	7020	59.9M

Table 5. **Accuracy of MoViNet on Kinetics 600** with additional top-5 data.

MODEL	TOP-1	GFLOPS	PARAM
MoViNet-A0	**58.5**	**2.71**	**3.1M**
MoViNet-A1	**63.5**	**6.02**	**4.6M**
MoViNet-A2	**66.7**	**10.3**	**4.8M**
MoViNet-A3	**68.0**	**56.9**	**5.3M**
MoViNet-A4	**70.7**	**105**	**4.9M**
MoViNet-A5	**71.7**	**281**	**15.7M**
MoViNet-A6	**72.3**	**386**	**31.4M**
SlowFast-R101 [11, 1]	70.2	3200	30M
SlowFast-R152 [11, 1]	71.6	9500	80M
EfficientNet-L2 (pretrain) [27, 1]	**76.2**	**15400**	**480M**

Table 6. **Accuracy of MoViNet on Kinetics 700**.

Model	Top-1	GFLOPs	Param
MoViNet-A0	**27.5**	**4.07**	**3.1M**
MoViNet-A1	**32.0**	**9.03**	**4.6M**
TVN-1 [19]	23.1	13.0	11.1M
MoViNet-A2	**34.3**	**15.5**	**4.8M**
MoViNet-A2-Stream	33.6	15.6	**4.8M**
TVN-2 [19]	24.2	17.0	110M
MoViNet-A3	**35.6**	**35.6**	**5.3M**
TVN-3 [19]	25.4	69.0	69.4M
MoViNet-A4	**37.9**	**98.4**	**4.9M**
TVN-4 [19]	27.8	**106**	44.2M
MoViNet-A5	**39.1**	**175**	**15.7M**
SRTG-R3D-34 [24]	28.5	220	-
MoViNet-A6	**40.2**	**274**	**31.4M**
ResNet3D-50 [23]	27.2	-	-
SRTG-R3D-50 [24]	30.7	300	-
SRTG-R3D-101 [24]	33.6	350	-
AssembleNet-50 (RGB+Flow) [23]	31.4	480	37.3M
AssembleNet-101 (RGB+Flow) [23]	34.3	760	53.3M

Table 7. **Accuracy of MoViNet on Moments in Time**. All MoViNets are evaluated on 75 frames at 25 fps.

Model	mAP	GFLOPs	Param
MoViNet-A2	**32.5**	**6.59**	**4.8M**
TVN-1 [19]	32.2	13.0	11.1M
TVN-2 [19]	**32.5**	**17.0**	**110M**
TVN-3 [19]	**33.5**	**69.0**	**69.4M**
MoViNet-A4	**48.5**	**90.4**	**4.9M**
TVN-4 [19]	35.4	106	44.2M
MoViNet-A6	**63.2**	**306**	**31.4M**
AssembleNet-50 (RGB+Flow) [23]	53.0	700	37.3M
AssembleNet-101 (RGB+Flow) [23]	58.6	1200	53.3M
AssembleNet++ (RGB+Flow+Seg)[22]	59.8	1300	-
SlowFast 16x8 R101 [11]	45.2	7020	59.9M

Table 8. **Accuracy of MoViNet on Charades**.

Model	Top-1	Top-5	GFLOPs	Param
MoViNet-A0	**61.3**	88.2	**2.71**	**3.1M**
MoViNet-A0-Stream	60.9	**88.3**	2.73	**3.1M**
TRN [29]	48.8	77.6	-	-
MoViNet-A1	**62.7**	**89.0**	**6.02**	**4.6M**
MoViNet-A1-Stream	61.6	87.3	6.06	**4.6M**
MoViNet-A2	**63.5**	**89.0**	**10.3**	4.8M
MoViNet-A2-Stream	63.1	89.0	10.4	4.8M
TSM [16]	63.4	88.5	33.0	24.3M
VoV3D-M (16 frame) [15]	63.2	88.2	34.2	**3.3M**
MoViNet-A3	**64.1**	**88.8**	**23.7**	**5.3M**
VoV3D-M (32 frame) [15]	65.2	**89.4**	**69.0**	**3.3M**
VoV3D-L (32 frame) [15]	**67.3**	90.5	125	5.8M

Table 9. **Accuracy of MoViNet on Something-Something V2**. All MoViNets are evaluated on 50 frames at 12 fps. For shorter clips with fewer than 50 frames, we repeat the video sequence from the beginning.

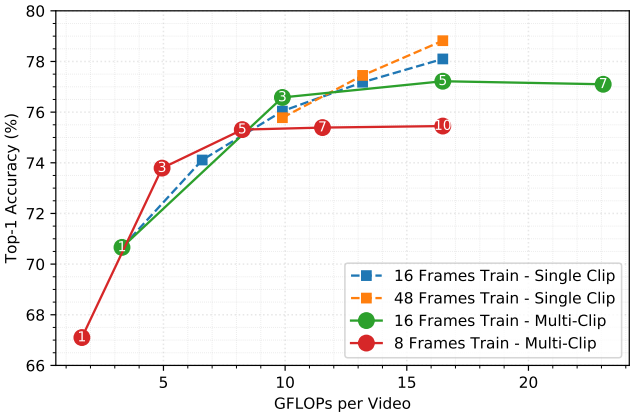

Figure 4. **Single vs. Multi-Clip Evaluation** on Kinetics 600. A comparison between the number of training frames and the number of eval frames and clips. The number of clips are shown inside each datapoint, where applicable. Other datapoints are evaluated on single clips. We use MoViNet-A2 with frame stride 3 for all datapoints.

As seen in Figure 1 and Table 2 (in the main text), switching from a multi-clip to single-clip X3D model on Kinetics 600 (where we cover the entire 10-second clip) results in much higher computational efficiency per video. Existing work typically factors out FLOPs in terms of FLOPs per subclip, but it can hide the true cost of computation, since we can keep adding more clips to boost accuracy higher.

We also evaluate the differences between training the same MoViNet-A2 model on smaller clips vs. longer clips and evaluating the models with multi-clip vs. single-clip, as seen in Figure 4. For multi-clip evaluation, we can see that accuracy improves when the number of clips fill the whole duration of the video (this can be seen at 5 clips for 8 training frames and at 3 clips for 16 training frames), and only very slightly improves as we add more clips. However, if we train MoViNet-A2 on 16 frames and evaluate on 80 frames (so that we cover all 10 seconds of the video), this results in higher accuracy than the same number of frames using multi-clip eval. Furthermore, we can boost this accuracy even higher if we use 48 frames to train our model. Using stream buffers, we can reduce memory usage of training so that we can train using 48 frames while only using the memory of embedding 16 frames at a time.

C.5 Streaming vs. Non-Streaming Evaluation

One question we have wondered is if the distribution of features learned is different from streaming and non-streaming architectures. In Figure 5, we plot the average accuracy across Kinetics 600 of a model evaluated on a single frame by embedding an entire video, pooling across spatial

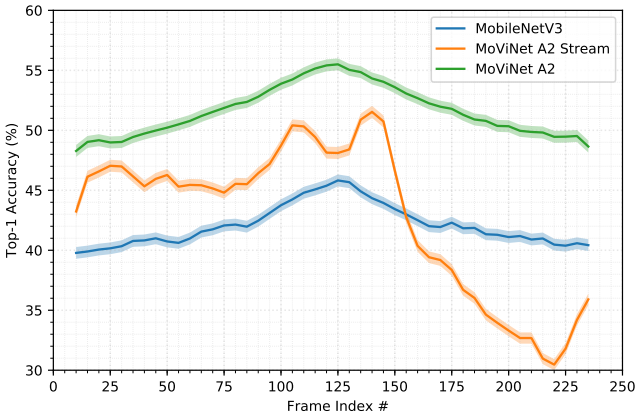

Figure 5. **Difference Between Streaming and Base MoViNets**. The plot displays the average accuracy across the Kinetics 600 dataset of an embedded model by applying the classification layers independently on each frame. Shading around each solid line indicates one standard deviation.

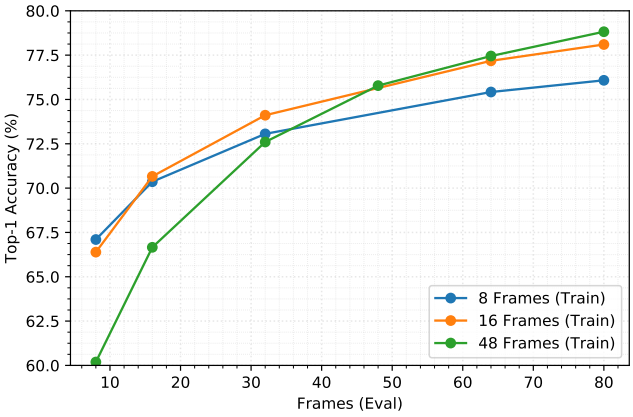

Figure 6. **Generalization to Longer Clips**. A display of how duration of a clip during training affects the evaluation accuracy of different clip durations during evaluation. We use MoViNet-A2 with frame stride 3 for all datapoints.

dimensions, and applying the classification layers independently on each frame.

We first notice that the accuracy MobileNetV3 and MoViNet-A2 exhibit a Laplace distribution, on average peaking at the center frame of each video. Since MobileNetV3 is evaluated on each frame independently, we can observe that the most salient part of the actions is on average in the video's midpoint. This is a good indicator that the videos in Kinetics are trimmed very well to center around the most salient part of each action. Likewise, MoViNet-A2, with balanced 3D convolutions, has the same characteristics as MobileNetV3, just with higher accuracy.

However, the dynamics of streaming MoViNet-A2 with causal convolutions is entirely different. The distribution of accuracy fluctuates and varies more than non-streaming architectures. By removing the ability for the network to see all frames as a whole with causal convolutions, the aggregation of features is not the same as when using balanced convolutions. Despite this difference, overall, the accuracy difference across all videos is only about 1%. And by looking at top-5 accuracy in Table 5, we can notice that streaming architectures nearly perform the same, despite the apparent information loss when transitioning to a model with a time-unidirectional receptive field.

C.6 Long Video Sequences

Figure 6 shows how training clip duration affects the accuracy of a model evaluated at different durations. We can see that MoViNet can generalize well beyond the original clip duration it was trained with, always improving in accuracy with more frames. However, the model does notably worse if evaluated on clips with shorter durations than it was trained on. Longer clip duration for training translates to better accuracy for evaluation on longer clips overall. And with a stream buffer, we can train on even longer sequences to boost evaluation performance even higher.

Model	Top-1	GFLOPs	Params
MoViNet-A2b-3D	79.0	17.1	4.8M
MoViNet-A2b-(2+1)D	79.4	16.8	5.0M

Table 10. **3D vs. (2+1)D** on Kinetics 600.

However, we also see we can operate frame-by-frame with stream buffers, substantially saving memory, showing better memory efficiency than multi-clip approaches and requiring constant memory as the number of input frames increase (and therefore temporal receptive field). Despite the accuracy reduction, we can see MoViNet-Stream models perform very well on long video sequences and are still more efficient than X3D which requires splitting videos into smaller subclips.

C.7 Stream Buffers with Other Operations

WaveNet [18] introduces causal convolution, where the receptive field on a stack of 1D convolutions is forced to only see activations up to the current time step, as opposed to balanced convolutions which expand their receptive fields in both directions. We take inspiration from causal convolutions [6, 5, 8] to design stream buffers. However, WaveNet only proposes 1D convolutions for generative modeling, using them for their autoregressive property. We generalize the idea of causal convolution to any local operation, and introduce stream buffers to be able to use causal operations for online inference, allowing frame-by-frame predictions. In addition, Transformer-XL [7] caches activations in a tem-

poral buffer much like our work, for use in long-range sequence modeling. However, the model is only causal across fixed sequences while our work can be causal across individual frames, and can even vary the number of frames in each clip (so long as frames are consecutive with no gaps or overlaps between clips). We can apply the same principle to other operations as well to generalize causal operations. Note that this approach is not inherently tied to any data type or modality. Stream buffers can also be used to model many kinds of temporal data, e.g., audio, text.

(2+1)D CNNs. Additionally, support for efficient 3D convolutions on mobile devices is currently fragmented, while 2D convolutions are well supported. We include the option to search for (2+1)D architectures, splitting up any 3D depthwise convolutions into a 2D spatial convolution followed by a 1D temporal convolution. We show that trivially changing a 3D architecture to (2+1)D decreases FLOPs while also keeping similar accuracy, as seen in table 10. Here we define MoViNet-A2b as a searched model similar to MoViNet-A2.

STAGE	OPERATION	OUTPUT SIZE
data	stride 5, RGB	50×172^2
conv_1	1×3^2, 8	50×86^2
block_2	$\left[\begin{array}{c} 1\times5^2, 8, 40 \end{array}\right]$	50×43^2
block_3	$\left[\begin{array}{c} 5\times3^2, 32, 80 \\ 3\times3^2, 32, 80 \\ 3\times3^2, 32, 80 \end{array}\right]$	50×21^2
block_4	$\left[\begin{array}{c} 5\times3^2, 56, 184 \\ 3\times3^2, 56, 112 \\ 3\times3^2, 56, 184 \end{array}\right]$	50×10^2
block_5	$\left[\begin{array}{c} 5\times3^2, 56, 184 \\ 3\times3^2, 56, 184 \\ 3\times3^2, 56, 184 \\ 3\times3^2, 56, 184 \end{array}\right]$	50×10^2
block_6	$\left[\begin{array}{c} 5\times3^2, 104, 344 \\ 1\times5^2, 104, 280 \\ 1\times5^2, 104, 280 \\ 1\times5^2, 104, 344 \end{array}\right]$	50×5^2
conv_7	1×1^2, 480	50×5^2
pool_8	50×5^2	1×1^2
dense_9	1×1^2, 2048	1×1^2
dense_{10}	1×1^2, 600	1×1^2

Table 11. **MoViNet-A0 Architecture**.

STAGE	OPERATION	OUTPUT SIZE
data	stride 5, RGB	50×172^2
conv_1	1×3^2, 16	50×86^2
block_2	$\left[\begin{array}{c} 1\times5^2, 16, 40 \\ 3\times3^2, 16, 40 \end{array}\right]$	50×43^2
block_3	$\left[\begin{array}{c} 3\times3^2, 40, 96 \\ 3\times3^2, 40, 120 \\ 3\times3^2, 40, 96 \\ 3\times3^2, 40, 96 \end{array}\right]$	50×21^2
block_4	$\left[\begin{array}{c} 5\times3^2, 64, 216 \\ 3\times3^2, 64, 128 \\ 3\times3^2, 64, 216 \\ 3\times3^2, 64, 168 \\ 3\times3^2, 64, 216 \end{array}\right]$	50×10^2
block_5	$\left[\begin{array}{c} 5\times3^2, 64, 216 \\ 3\times3^2, 64, 216 \\ 3\times3^2, 64, 216 \\ 3\times3^2, 64, 128 \\ 1\times5^2, 64, 128 \\ 3\times3^2, 64, 216 \end{array}\right]$	50×10^2
block_6	$\left[\begin{array}{c} 5\times3^2, 136, 456 \\ 1\times5^2, 136, 360 \\ 1\times5^2, 136, 360 \\ 1\times5^2, 136, 360 \\ 1\times5^2, 136, 456 \\ 3\times3^2, 136, 456 \\ 1\times3^2, 136, 544 \end{array}\right]$	50×5^2
conv_7	1×1^2, 600	50×5^2
pool_8	50×5^2	1×1^2
dense_9	1×1^2, 2048	1×1^2
dense_{10}	1×1^2, 600	1×1^2

Table 12. **MoViNet-A1 Architecture**.

STAGE	OPERATION	OUTPUT SIZE
data	stride 5, RGB	50×224^2
conv_1	1×3^2, 16	50×112^2
block_2	$\begin{bmatrix} 1\times5^2, 16, 40 \\ 3\times3^2, 16, 40 \\ 3\times3^2, 16, 64 \end{bmatrix}$	50×56^2
block_3	$\begin{bmatrix} 3\times3^2, 40, 96 \\ 3\times3^2, 40, 120 \\ 3\times3^2, 40, 96 \\ 3\times3^2, 40, 96 \\ 3\times3^2, 40, 120 \end{bmatrix}$	50×28^2
block_4	$\begin{bmatrix} 5\times3^2, 72, 240 \\ 3\times3^2, 72, 160 \\ 3\times3^2, 72, 240 \\ 3\times3^2, 72, 192 \\ 3\times3^2, 72, 240 \end{bmatrix}$	50×14^2
block_5	$\begin{bmatrix} 5\times3^2, 72, 240 \\ 3\times3^2, 72, 240 \\ 3\times3^2, 72, 240 \\ 3\times3^2, 72, 240 \\ 1\times5^2, 72, 144 \\ 3\times3^2, 72, 240 \end{bmatrix}$	50×14^2
block_6	$\begin{bmatrix} 5\times3^2, 144, 480 \\ 1\times5^2, 144, 384 \\ 1\times5^2, 144, 384 \\ 1\times5^2, 144, 480 \\ 1\times5^2, 144, 480 \\ 3\times3^2, 144, 480 \\ 1\times3^2, 144, 576 \end{bmatrix}$	50×7^2
conv_7	1×1^2, 640	50×7^2
pool_8	50×7^2	1×1^2
dense_9	1×1^2, 2048	1×1^2
dense_{10}	1×1^2, 600	1×1^2

Table 13. **MoViNet-A2 Architecture**.

STAGE	OPERATION	OUTPUT SIZE
data	stride 5, RGB	120×256^2
conv_1	1×3^2, 16	120×128^2
block_2	$\begin{bmatrix} 1\times5^2, 16, 40 \\ 3\times3^2, 16, 40 \\ 3\times3^2, 16, 64 \\ 3\times3^2, 16, 40 \end{bmatrix}$	120×64^2
block_3	$\begin{bmatrix} 3\times3^2, 48, 112 \\ 3\times3^2, 48, 144 \\ 3\times3^2, 48, 112 \\ 1\times5^2, 48, 112 \\ 3\times3^2, 48, 144 \\ 3\times3^2, 48, 144 \end{bmatrix}$	120×32^2
block_4	$\begin{bmatrix} 5\times3^2, 80, 240 \\ 3\times3^2, 80, 152 \\ 3\times3^2, 80, 240 \\ 3\times3^2, 80, 192 \\ 3\times3^2, 80, 240 \end{bmatrix}$	120×16^2
block_5	$\begin{bmatrix} 5\times3^2, 88, 264 \\ 3\times3^2, 88, 264 \\ 3\times3^2, 88, 264 \\ 3\times3^2, 88, 264 \\ 1\times5^2, 88, 160 \\ 3\times3^2, 88, 264 \\ 3\times3^2, 88, 264 \\ 3\times3^2, 88, 264 \end{bmatrix}$	120×16^2
block_6	$\begin{bmatrix} 5\times3^2, 168, 560 \\ 1\times5^2, 168, 448 \\ 1\times5^2, 168, 448 \\ 1\times5^2, 168, 560 \\ 1\times5^2, 168, 560 \\ 3\times3^2, 168, 560 \\ 1\times5^2, 168, 448 \\ 1\times5^2, 168, 448 \\ 3\times3^2, 168, 560 \\ 1\times3^2, 168, 672 \end{bmatrix}$	120×8^2
conv_7	1×1^2, 744	120×8^2
pool_8	120×8^2	1×1^2
dense_9	1×1^2, 2048	1×1^2
dense_{10}	1×1^2, 600	1×1^2

Table 14. **MoViNet-A3 Architecture**.

STAGE	OPERATION	OUTPUT SIZE
data	stride 5, RGB	80×290^2
conv_1	1×3^2, 24	80×145^2
block_2	$\begin{bmatrix} 1\times5^2, 24, 64 \\ 3\times3^2, 24, 64 \\ 3\times3^2, 24, 96 \\ 3\times3^2, 24, 64 \\ 3\times3^2, 24, 96 \\ 3\times3^2, 24, 64 \end{bmatrix}$	80×72^2
block_3	$\begin{bmatrix} 5\times3^2, 56, 168 \\ 3\times3^2, 56, 168 \\ 3\times3^2, 56, 136 \\ 3\times3^2, 56, 136 \\ 3\times3^2, 56, 168 \\ 3\times3^2, 56, 168 \\ 3\times3^2, 56, 168 \\ 1\times5^2, 56, 136 \\ 3\times3^2, 56, 136 \end{bmatrix}$	80×36^2
block_4	$\begin{bmatrix} 5\times3^2, 96, 320 \\ 3\times3^2, 96, 160 \\ 3\times3^2, 96, 320 \\ 3\times3^2, 96, 192 \\ 3\times3^2, 96, 320 \\ 3\times3^2, 96, 152 \\ 3\times3^2, 96, 320 \\ 3\times3^2, 96, 256 \\ 3\times3^2, 96, 320 \end{bmatrix}$	80×18^2
block_5	$\begin{bmatrix} 5\times3^2, 96, 320 \\ 3\times3^2, 96, 320 \\ 3\times3^2, 96, 320 \\ 3\times3^2, 96, 320 \\ 1\times5^2, 96, 192 \\ 3\times3^2, 96, 320 \\ 3\times3^2, 96, 320 \\ 3\times3^2, 96, 192 \\ 3\times3^2, 96, 320 \\ 3\times3^2, 96, 320 \end{bmatrix}$	80×18^2
block_6	$\begin{bmatrix} 5\times3^2, 192, 640 \\ 1\times5^2, 192, 512 \\ 1\times5^2, 192, 512 \\ 1\times5^2, 192, 640 \\ 1\times5^2, 192, 640 \\ 3\times3^2, 192, 640 \\ 1\times5^2, 192, 512 \\ 1\times5^2, 192, 512 \\ 1\times5^2, 192, 640 \\ 1\times5^2, 192, 768 \\ 1\times5^2, 192, 640 \\ 3\times3^2, 192, 640 \\ 3\times3^2, 192, 768 \end{bmatrix}$	80×9^2
conv_7	1×1^2, 856	80×9^2
pool_8	80×9^2	1×1^2
dense_9	1×1^2, 2048	1×1^2
dense_{10}	1×1^2, 600	1×1^2

Table 15. **MoViNet-A4 Architecture**.

STAGE	OPERATION	OUTPUT SIZE
data	stride 5, RGB	120×320^2
conv_1	1×3^2, 24	120×160^2
block_2	$\begin{bmatrix} 1\times5^2, 24, 64 \\ 1\times5^2, 24, 64 \\ 3\times3^2, 24, 96 \\ 3\times3^2, 24, 64 \\ 3\times3^2, 24, 96 \\ 3\times3^2, 24, 64 \end{bmatrix}$	120×80^2
block_3	$\begin{bmatrix} 5\times3^2, 64, 192 \\ 3\times3^2, 64, 152 \\ 3\times3^2, 64, 152 \\ 3\times3^2, 64, 152 \\ 3\times3^2, 64, 192 \\ 3\times3^2, 64, 192 \\ 3\times3^2, 64, 192 \\ 3\times3^2, 64, 152 \\ 3\times3^2, 64, 152 \\ 3\times3^2, 64, 192 \\ 3\times3^2, 64, 192 \end{bmatrix}$	120×40^2
block_4	$\begin{bmatrix} 5\times3^2, 112, 376 \\ 3\times3^2, 112, 224 \\ 3\times3^2, 112, 376 \\ 3\times3^2, 112, 376 \\ 3\times3^2, 112, 296 \\ 3\times3^2, 112, 376 \\ 3\times3^2, 112, 224 \\ 3\times3^2, 112, 376 \\ 3\times3^2, 112, 376 \\ 3\times3^2, 112, 296 \\ 3\times3^2, 112, 376 \\ 3\times3^2, 112, 376 \\ 3\times3^2, 112, 376 \end{bmatrix}$	120×20^2
block_5	$\begin{bmatrix} 5\times3^2, 120, 376 \\ 3\times3^2, 120, 376 \\ 3\times3^2, 120, 376 \\ 3\times3^2, 120, 376 \\ 1\times5^2, 120, 224 \\ 3\times3^2, 120, 376 \\ 3\times3^2, 120, 376 \\ 3\times3^2, 120, 224 \\ 3\times3^2, 120, 376 \\ 3\times3^2, 120, 376 \\ 3\times3^2, 120, 376 \end{bmatrix}$	120×20^2
block_6	$\begin{bmatrix} 5\times3^2, 224, 744 \\ 3\times3^2, 224, 744 \\ 1\times5^2, 224, 600 \\ 1\times5^2, 224, 600 \\ 1\times5^2, 224, 744 \\ 1\times5^2, 224, 744 \\ 3\times3^2, 224, 744 \\ 1\times5^2, 224, 896 \\ 1\times5^2, 224, 600 \\ 1\times5^2, 224, 600 \\ 1\times5^2, 224, 896 \\ 1\times5^2, 224, 744 \\ 3\times3^2, 224, 744 \\ 1\times5^2, 224, 896 \\ 1\times5^2, 224, 600 \\ 1\times5^2, 224, 600 \\ 1\times5^2, 224, 744 \\ 3\times3^2, 224, 744 \end{bmatrix}$	120×10^2
conv_7	1×1^2, 992	120×10^2
pool_8	120×10^2	1×1^2
dense_9	1×1^2, 2048	1×1^2
dense_{10}	1×1^2, 600	1×1^2

Table 16. **MoViNet-A5 Architecture**.

References

[1] Activitynet task b: Kinetics challenge. http://activity-net.org/challenges/2020/tasks/guest_kinetics.html. 2020. 4

[2] Thomas Bachlechner, Bodhisattwa Prasad Majumder, Huanru Henry Mao, Garrison W Cottrell, and Julian McAuley. Rezero is all you need: Fast convergence at large depth. *arXiv preprint arXiv:2003.04887*, 2020. 2

[3] 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. 1

[4] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? *arXiv preprint arXiv:2102.05095*, 2021. 4

[5] 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. 6

[6] 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. 6

[7] 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. 6

[8] 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. 6

[9] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018. 2

[10] 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. 4

[11] 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. 4, 5

[12] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 558–567, 2019. 2

[13] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1314–1324, 2019. 2, 3

[14] 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

[15] Youngwan Lee, Hyung-Il Kim, Kimin Yun, and Jinyoung Moon. Diverse temporal aggregation and depthwise spatiotemporal factorization for efficient video classification. *arXiv preprint arXiv:2012.00317*, 2020. 5

[16] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7083–7093, 2019. 5

[17] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 2

[18] 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. 6

[19] AJ Piergiovanni, Anelia Angelova, and Michael S Ryoo. Tiny video networks: Architecture search for efficient video models. 2020. 5

[20] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992. 2

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

[22] Michael S Ryoo, AJ Piergiovanni, Juhana Kangaspunta, and Anelia Angelova. Assemblenet++: Assembling modality representations via attention connections-supplementary material. 2020. 5

[23] Michael S Ryoo, AJ Piergiovanni, Mingxing Tan, and Anelia Angelova. Assemblenet: Searching for multi-stream neural connectivity in video architectures. *arXiv preprint arXiv:1905.13209*, 2019. 2, 5

[24] Alexandros Stergiou and Ronald Poppe. Learn to cycle: Time-consistent feature discovery for action recognition. *arXiv preprint arXiv:2006.08247*, 2020. 5

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

[26] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992. 1

[27] 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. 4

[28] 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

[29] 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. 5