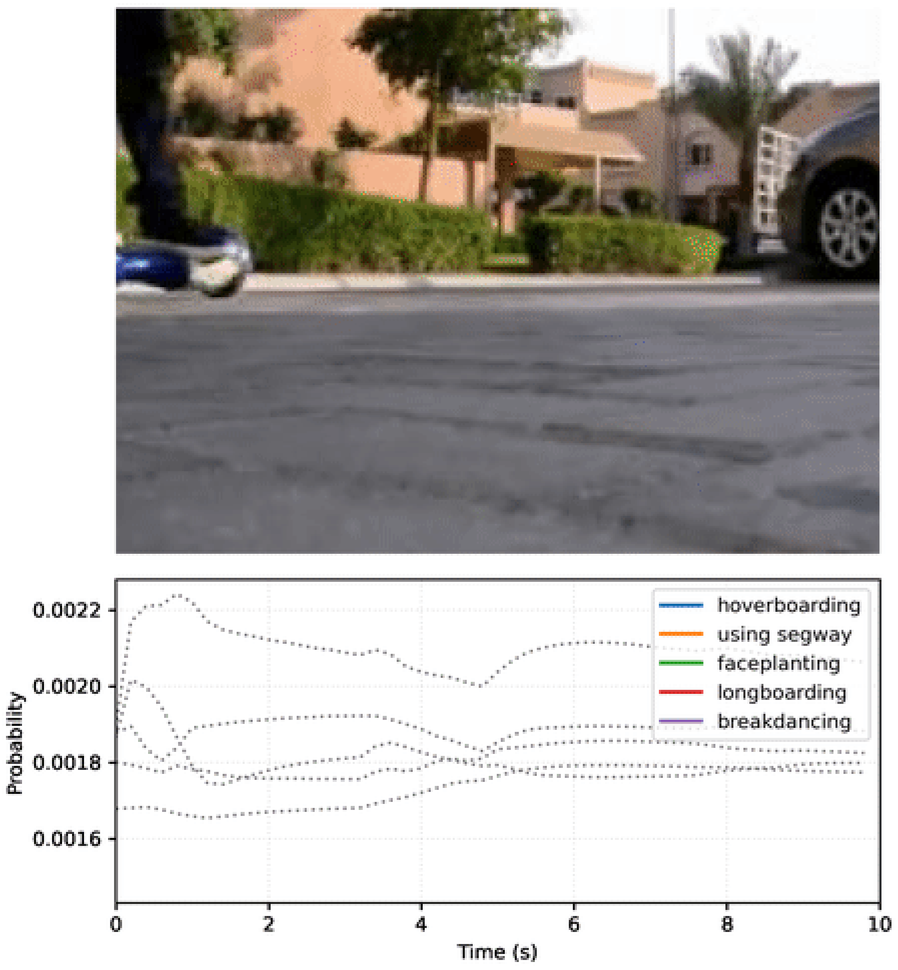
Mobile Video Networks (MoViNets)

Open in Colab | TF Hub Models | arXiv Paper arXiv.2103.11511

This repository is the official implementation of [MoViNets: Mobile Video Networks for Efficient Video Recognition](#).

- **[UPDATE 2022-03-14] Quantized TF Lite models [available on TF Hub](#) (also [see table](#) for quantized performance)**



Create your own video plot like the one above with this [Colab notebook](#).

Description

Mobile Video Networks (MoViNets) are efficient video classification models runnable on mobile devices. MoViNets demonstrate state-of-the-art accuracy and efficiency on several large-scale video action recognition datasets.

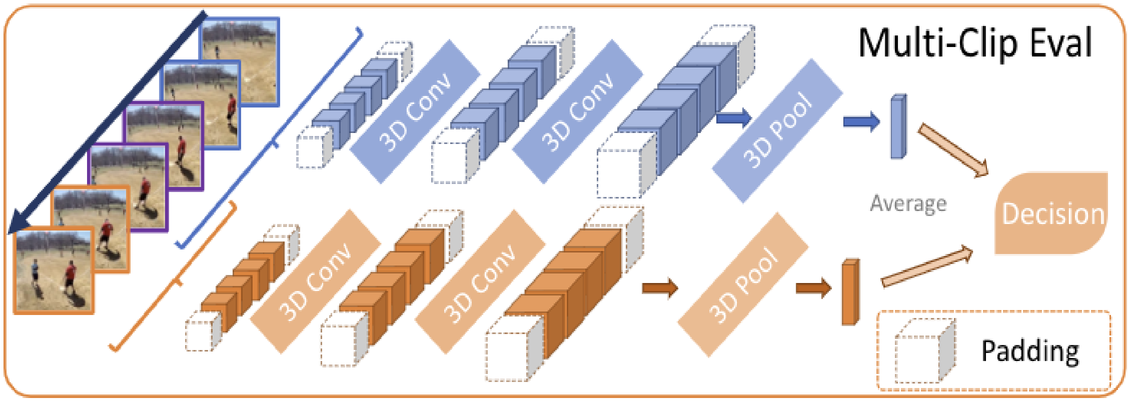
On [Kinetics 600](#), MoViNet-A6 achieves 84.8% top-1 accuracy, outperforming recent Vision Transformer models like [ViViT](#) (83.0%) and [VATT](#) (83.6%) without any additional training data, while using 10x fewer FLOPs. And streaming MoViNet-A0 achieves 72% accuracy while using 3x fewer FLOPs than MobileNetV3-large (68%).

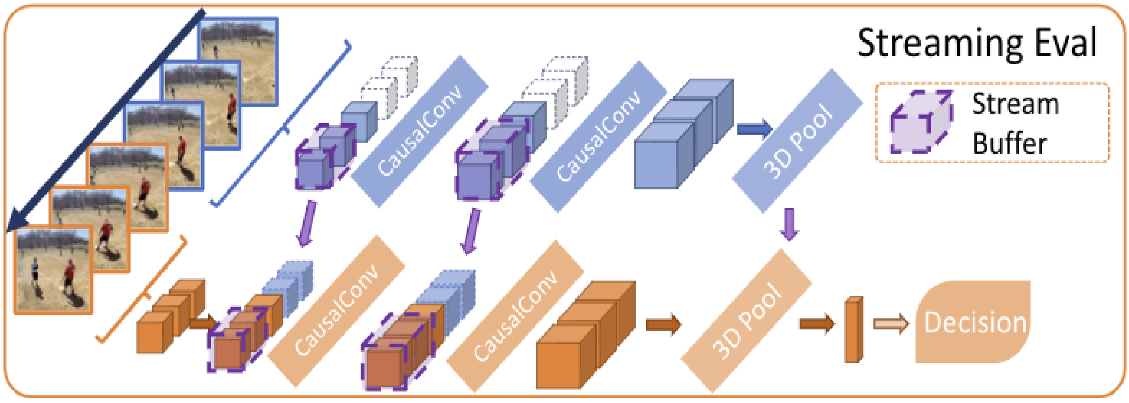
There is a large gap between video model performance of accurate models and efficient models for video action recognition. On the one hand, 2D MobileNet CNNs are fast and can operate on streaming video in real time, but are prone to be noisy and inaccurate. On the other hand, 3D CNNs are accurate, but are memory and computation intensive and cannot operate on streaming video.

MoViNets bridge this gap, producing:

- State-of-the art efficiency and accuracy across the model family (MoViNet-A0 to A6).
- Streaming models with 3D causal convolutions substantially reducing memory usage.
- Temporal ensembles of models to boost efficiency even higher.

MoViNets also improve computational efficiency by outputting high-quality predictions frame by frame, as opposed to the traditional multi-clip evaluation approach that performs redundant computation and limits temporal scope.





History

- **2022-03-14** Support quantized TF Lite models and add/update Colab notebooks.
- **2021-07-12** Add TF Lite support and replace 3D stream models with mobile-friendly (2+1)D stream.
- **2021-05-30** Add streaming MoViNet checkpoints and examples.
- **2021-05-11** Initial Commit.

Authors and Maintainers

- Dan Kondratyuk (@hyperparticle)
- Liangzhe Yuan (@yuanliangzhe)

- Yeqing Li (@yeqingli)

Table of Contents

Requirements

TensorFlow 2.1 Python 3.6

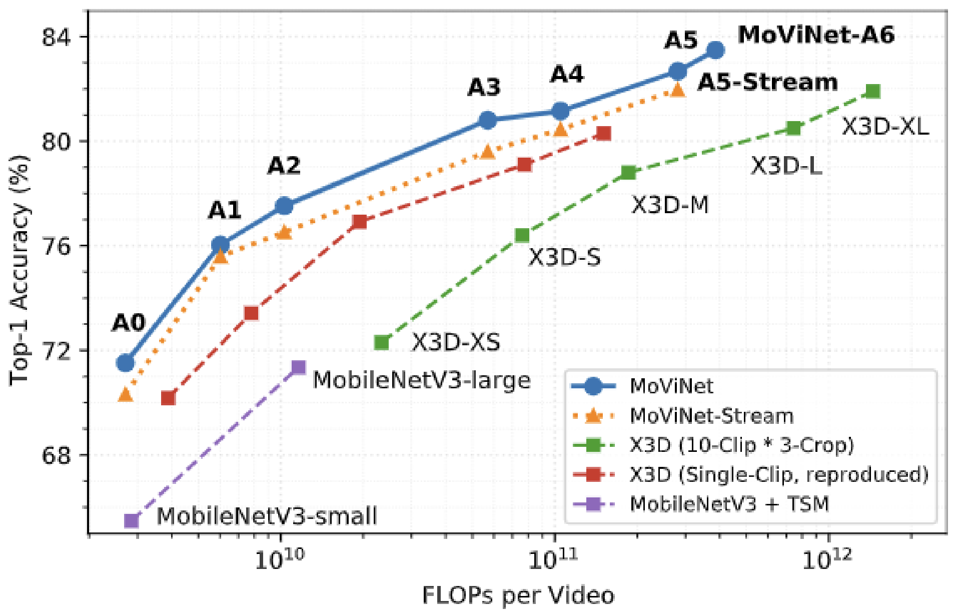
To install requirements:

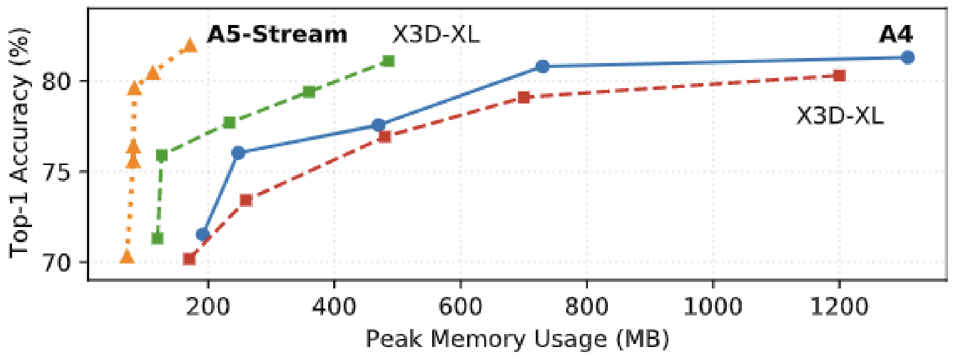
```
pip install -r requirements.txt
```

Results and Pretrained Weights

TF Hub Models TensorBoard dev

Kinetics 600





tensorboard.dev summary of training runs across all models.

The table below summarizes the performance of each model on Kinetics 600 and provides links to download pretrained models. All models are evaluated on single clips with the same resolution as training.

Note: MoViNet-A6 can be constructed as an ensemble of MoViNet-A4 and MoViNet-A5.

Base Models

Base models implement standard 3D convolutions without stream buffers. Base models are not recommended for fast inference on CPU or mobile due to limited support for `tf.nn.conv3d`. Instead, see the streaming models section.

Model Name	Top-1 Accuracy	Top-5 Accuracy	Input Shape	GFLOPs*	Checkpoint	TF Hub SavedModel
MoViNet-A0-Base	72.28	90.92	50 x 172 x 172	2.7	checkpoint (12 MB)	tfhub
MoViNet-A1-Base	76.69	93.40	50 x 172 x 172	6.0	checkpoint (18 MB)	tfhub

MoViNet-A2-Base	78.62	94.17	50 x 224 x 224	10	checkpoint (20 MB)	tfhub
MoViNet-A3-Base	81.79	95.67	120 x 256 x 256	57	checkpoint (29 MB)	tfhub
MoViNet-A4-Base	83.48	96.16	80 x 290 x 290	110	checkpoint (44 MB)	tfhub
MoViNet-A5-Base	84.27	96.39	120 x 320 x 320	280	checkpoint (72 MB)	tfhub

*GFLOPs per video on Kinetics 600.

Streaming Models

Streaming models implement causal (2+1)D convolutions with stream buffers. Streaming models use (2+1)D convolution instead of 3D to utilize optimized `tf.nn.conv2d` operations, which offer fast inference on CPU. Streaming models can be run on individual frames or on larger video clips like base models.

Note: A3, A4, and A5 models use a positional encoding in the squeeze-excitation blocks, while A0, A1, and A2 do not. For the smaller models, accuracy is unaffected without positional encoding, while for the larger models accuracy is significantly worse without positional encoding.

Model Name	Top-1 Accuracy	Top-5 Accuracy	Input Shape*	GFLOPs**	Checkpoint	TF Hub SavedModel
MoViNet-A0-Stream	72.05	90.63	50 x 172 x 172	2.7	checkpoint (12 MB)	tfhub
MoViNet-A1-Stream	76.45	93.25	50 x 172 x 172	6.0	checkpoint (18 MB)	tfhub
MoViNet-A2-Stream	78.40	94.05	50 x 224 x 224	10	checkpoint (20 MB)	tfhub
MoViNet-A3-Stream	80.09	94.84	120 x 256 x 256	57	checkpoint (29 MB)	tfhub
MoViNet-A4-Stream	81.49	95.66	80 x 290 x 290	110	checkpoint (44 MB)	tfhub
MoViNet-A5-Stream	82.37	95.79	120 x 320 x 320	280	checkpoint (72 MB)	tfhub

*In streaming mode, the number of frames correspond to the total accumulated duration of the 10-second clip.

**GFLOPs per video on Kinetics 600.

Note: current streaming model checkpoints have been updated with a slightly different architecture. To download the old checkpoints, insert `_legacy` before `.tar.gz` in the URL. E.g., `movinet_a0_stream_legacy.tar.gz`.

TF Lite Streaming Models

For convenience, we provide converted TF Lite models for inference on mobile devices. See the TF Lite Example to export and run your own models. We also provide quantized TF Lite binaries via TF Hub.

For reference, MoViNet-A0-Stream runs with a similar latency to MobileNetV3-Large with +5% accuracy on Kinetics 600.

Model Name	Input Shape	Pixel 4 Latency*	x86 Latency*	TF Lite Binary
MoViNet-A0-Stream	1 x 1 x 172 x 172	22 ms	16 ms	TF Lite (13 MB)
MoViNet-A1-Stream	1 x 1 x 172 x 172	42 ms	33 ms	TF Lite (45 MB)
MoViNet-A2-Stream	1 x 1 x 224 x 224	200 ms	66 ms	TF Lite (53 MB)
MoViNet-A3-Stream	1 x 1 x 256 x 256	-	120 ms	TF Lite (73 MB)
MoViNet-A4-Stream	1 x 1 x 290 x 290	-	300 ms	TF Lite (101 MB)
MoViNet-A5-Stream	1 x 1 x 320 x 320	-	450 ms	TF Lite (153 MB)

*Single-frame latency measured on with unaltered float32 operations on a single CPU core. Observed latency may differ depending on hardware configuration. Measured on a stock Pixel 4 (Android 11) and x86 Intel Xeon W-2135 CPU.

Kinetics 400

We also have checkpoints for Kinetics 400 models available. See the Kinetics 600 sections for more details. To load checkpoints, set `num_classes=400`.

Base Models

Model Name	Top-1 Accuracy	Top-5 Accuracy	Input Shape	GFLOPs*	Checkpoint
MoViNet-A0-Base	69.40	89.18	50 x 172 x 172	2.7	checkpoint (12 MB)
MoViNet-A1-Base	74.57	92.03	50 x 172 x 172	6.0	checkpoint (18 MB)
MoViNet-A2-Base	75.91	92.63	50 x 224 x 224	10	checkpoint (20 MB)
MoViNet-A3-Base	79.34	94.52	120 x 256 x 256	57	checkpoint (29 MB)
MoViNet-A4-Base	80.64	94.93	80 x 290 x 290	110	checkpoint (44 MB)
MoViNet-A5-Base	81.39	95.06	120 x 320 x 320	280	checkpoint (72 MB)

*GFLOPs per video on Kinetics 400.

Prediction Examples

Please check out our Colab Notebook to get started with MoViNets.

This section provides examples on how to run prediction.

For **base models**, run the following:

```
import tensorflow as tf

from official.projects.movinet.modeling import movinet
from official.projects.movinet.modeling import movinet_model

# Create backbone and model.
backbone = movinet.Movinet(
    model_id='a0',
    causal=False,
    use_external_states=False,
)
model = movinet_model.MovinetClassifier(
    backbone, num_classes=600, output_states=False)

# Create your example input here.
# Refer to the paper for recommended input shapes.
inputs = tf.ones([1, 8, 172, 172, 3])

# [Optional] Build the model and load a pretrained checkpoint
model.build(inputs.shape)

checkpoint_dir = '/path/to/checkpoint'
checkpoint_path = tf.train.latest_checkpoint(checkpoint_dir)
checkpoint = tf.train.Checkpoint(model=model)
status = checkpoint.restore(checkpoint_path)
status.assert_existing_objects_matched()

# Run the model prediction.
output = model(inputs)
prediction = tf.argmax(output, -1)
```

For **streaming models**, run the following:

```
import tensorflow as tf

from official.projects.movinet.modeling import movinet
from official.projects.movinet.modeling import movinet_model

model_id = 'a0'
use_positional_encoding = model_id in {'a3', 'a4', 'a5'}

# Create backbone and model.
backbone = movinet.Movinet(
    model_id=model_id,
    causal=True,
    conv_type='2plus1d',
    se_type='2plus3d',
    activation='hard_swish',
    gating_activation='hard_sigmoid',
    use_positional_encoding=use_positional_encoding,
    use_external_states=True,
)

model = movinet_model.MovinetClassifier(
    backbone,
    num_classes=600,
    output_states=True)

# Create your example input here.
# Refer to the paper for recommended input shapes.
inputs = tf.ones([1, 8, 172, 172, 3])

# [Optional] Build the model and load a pretrained checkpoint.
model.build(inputs.shape)

checkpoint_dir = '/path/to/checkpoint'
checkpoint_path = tf.train.latest_checkpoint(checkpoint_dir)
checkpoint = tf.train.Checkpoint(model=model)
status = checkpoint.restore(checkpoint_path)
status.assert_existing_objects_matched()

# Split the video into individual frames.
# Note: we can also split into larger clips as well (e.g., 8-frame clips).
# Running on larger clips will slightly reduce latency overhead, but
# will consume more memory.
frames = tf.split(inputs, inputs.shape[1], axis=1)

# Initialize the dict of states. All state tensors are initially zeros.
init_states = model.init_states(tf.shape(inputs))

# Run the model prediction by looping over each frame.
states = init_states
predictions = []
for frame in frames:
  output, states = model({**states, 'image': frame})
  predictions.append(output)

# The video classification will simply be the last output of the model.
final_prediction = tf.argmax(predictions[-1], -1)

# Alternatively, we can run the network on the entire input video.
# The output should be effectively the same
# (but it may differ a small amount due to floating point errors).
non_streaming_output, _ = model({**init_states, 'image': inputs})
non_streaming_prediction = tf.argmax(non_streaming_output, -1)
```

TF Lite Example

This section outlines an example on how to export a model to run on mobile devices with TF Lite.

[Optional] For streaming models, they are typically trained with `conv_type = 3d_2plus1d` for better training throughpouts. In order to achieve better inference performance on CPU, we need to convert the `3d_2plus1d` checkpoint to make it compatible with the `2plus1d` graph. You could achieve this by running `tools/convert_3d_2plus1d.py`.

First, convert to TF SavedModel by running `export_saved_model.py`. For example, for `MoViNet-A0-Stream`, run:

```
python3 export_saved_model.py \
  --model_id=a0 \
  --causal=True \
  --conv_type=2plus1d \
  --se_type=2plus3d \
  --activation=hard_swish \
  --gating_activation=hard_sigmoid \
  --use_positional_encoding=False \
  --num_classes=600 \
  --batch_size=1 \
  --num_frames=1 \
  --image_size=172 \
  --bundle_input_init_states_fn=False \
  --checkpoint_path=/path/to/checkpoint \
  --export_path=/tmp/movinet_a0_stream
```

Then the SavedModel can be converted to TF Lite using the `TFLiteConverter`:

```
saved_model_dir = '/tmp/movinet_a0_stream'
converter = tf.lite.TFLiteConverter.from_saved_model(saved_model_dir)
tflite_model = converter.convert()

with open('/tmp/movinet_a0_stream.tflite', 'wb') as f:
  f.write(tflite_model)
```

To run with TF Lite using tf.lite.Interpreter with the Python API:

```
# Create the interpreter and signature runner
interpreter = tf.lite.Interpreter('/tmp/movinet_a0_stream.tflite')
runner = interpreter.get_signature_runner()

# Extract state names and create the initial (zero) states
def state_name(name: str) -> str:
  return name[len('serving_default_'):-len(':0')]

init_states = {
    state_name(x['name']): tf.zeros(x['shape'], dtype=x['dtype'])
    for x in interpreter.get_input_details()
}
del init_states['image']

# Insert your video clip here
```

```
video = tf.ones([1, 8, 172, 172, 3])
clips = tf.split(video, video.shape[1], axis=1)

# To run on a video, pass in one frame at a time
states = init_states
for clip in clips:
  # Input shape: [1, 1, 172, 172, 3]
  outputs = runner(**states, image=clip)
  logits = outputs.pop('logits')
  states = outputs
```

Follow the official guide to run a model with TF Lite on your mobile device.

Training and Evaluation

Run this command line for continuous training and evaluation.

```
MODE=train_and_eval  # Can also be 'train' if using a separate evaluator job
CONFIG_FILE=official/projects/movinet/configs/yaml/movinet_a0_k600_8x8.yaml
python3 official/projects/movinet/train.py \
    --experiment=movinet_kinetics600 \
    --mode=${MODE} \
    --model_dir=/tmp/movinet_a0_base/ \
    --config_file=${CONFIG_FILE}
```

Run this command line for evaluation.

```
MODE=eval  # Can also be 'eval_continuous' for use during training
CONFIG_FILE=official/projects/movinet/configs/yaml/movinet_a0_k600_8x8.yaml
python3 official/projects/movinet/train.py \
    --experiment=movinet_kinetics600 \
    --mode=${MODE} \
    --model_dir=/tmp/movinet_a0_base/ \
    --config_file=${CONFIG_FILE}
```

License

License Apache 2.0

This project is licensed under the terms of the **Apache License 2.0**.

Citation

If you want to cite this code in your research paper, please use the following information.

```
@article{kondratyuk2021movinets,
  title={MoViNets: Mobile Video Networks for Efficient Video Recognition},
  author={Dan Kondratyuk, Liangzhe Yuan, Yandong Li, Li Zhang, Matthew Brown, and
Boqing Gong},
  journal={arXiv preprint arXiv:2103.11511},
  year={2021}
}
```