Deep Analysis of CNN-based Spatio-temporal Representations for Action Recognition (Supplementary Material)

Chun-Fu (Richard) Chen[1,†], Rameswar Panda[1,†], Kandan Ramakrishnan[1],
Rogerio Feris[1], John Cohn[1], Aude Oliva[2], Quanfu Fan[1,†]
†: Equal Contribution
[1]MIT-IBM Watson AI Lab, [2]Massachusetts Institute of Technology

Summary. As part of the supplementary material, we first compare additional SOTA models trained with uniform sampling and discuss runtime speed/memory of different models in Section 1. We analyze the effects of pretrained models in Section 2. Moreover, we expand Table 7 of the main paper with the FLOPs and parameters for different temporal aggregations in Table 3. We then describe the datasets and benchmarks in Section 3 and Section 4, respectively. Section 5 describes the model implementation used in the Mini-Datasets benchmark. Figure 2 shows the results of all the models on Mini datasets.

1. More Results on SOTA Models

We argue in the main paper that the effect of input sampling (i.e., uniform or dense sampling) should be considered for fair comparison of action recognition models. Following this practice, we add TSM [7] and TAM [2] to Figure 8 of the main paper, which is shown here as Figure 1. Like I3D, TAM and TSM work well with both uniform and dense sampling. When uniform sampling is considered for TAM and TSM, they are slightly worse than SlowFast in accuracy but at the same efficiency in FLOPs.

Moreover, we also benchmark the runtime speed and memory consumption for a more comprehensive comparison among different models. Table 1 shows throughput, FLOPs and max batch size that can be fitted on a GPU. The numbers are conducted based on a NVIDIA V100 32G GPU with CUDA 10.1, cudnn 7.6.5, and PyTorch 1.5; and the throughput is measured under maximum batch size. Slowfast achieved better speed and memory usage; on the other hand, even though TAM-R50 has only half FLOPs of I3D-R50, their throughputs are similar. This is because the 3D-depthwise convolution used in TAM is not yet optimized[1].

[1] https://github.com/pytorch/pytorch/pull/40801

Models	Accuracy (%)	FLOPs (G)	Throughput (clips/sec)	Maximum batch size
TSN-ResNet50-tp	74.9	73.9	53.5	40
TAM-ResNet50	76.2	171.5	14.2	48
I3D-ResNet50	76.6	335.3	16.1	44
SlowFast-ResNet50-8×8	76.4	65.7	48.9	96

Table 1: Speed and memory complexity of models.

Model	Pretrain	
	ImageNet	None
I3D-ResNet50	76.61	76.54
TAM-ResNet50	76.18	75.61

Table 2: Top-1 Accuracy (%) of I3D and TAM models trained with and without ImageNet weights on *Kinetics*.

2. Effects of Pretrained Models

It is shown in [5] that when there is enough training data and the training is sufficiently long, then pretraining from ImageNet is not necessary for producing competitive performance in a downstream task like object detection. Here we conduct a similar experiment to train I3D-ResNet50 and TAM-ResNet50 from scratch for 196 epoches. As can be seen from Table 2, I3D-ResNet50 does not benefit much from a pretrained ImageNet model. For TAM-ResNet50, when trained from scratch, it degrades its accuracy by ~ 0.5, which is not significant. Thus our results seem to echo a similar observation that pretraining is not crucial for large-scale video action recognition as long as the training is allowed to be sufficiently long.

3. Datasets

Table 4 illustrates the characteristics of the datasets used in the paper. The *SSV2* dataset contains a total of 192K videos of 174 human-object interactions, captured in a sim-

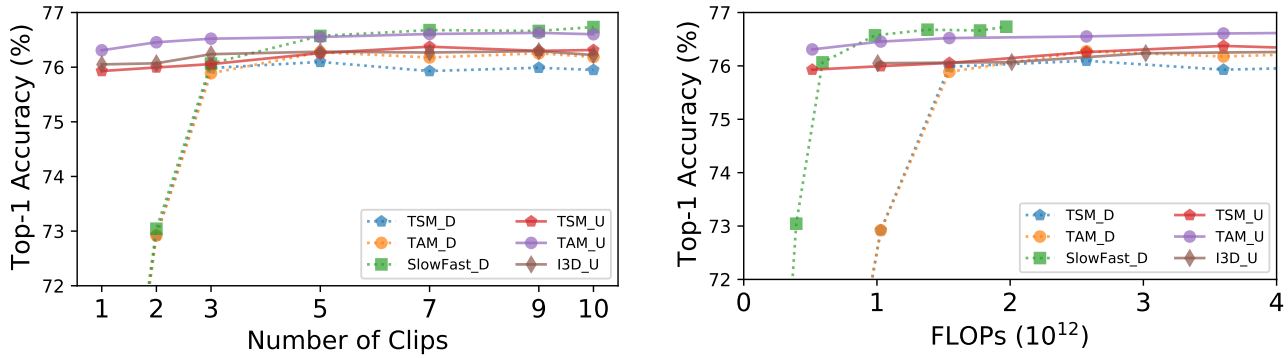

Figure 1: Model performance tested using 3 256×256 spatial crops and different number of clips. 'U': uniform sampling; 'D': dense sampling. Best viewed in color.

Dataset	Frames	InceptionV1				ResNet50					
		None	I3D	Conv.	TAM	None	I3D	Conv.	TAM	TSM	NLN
Mini-SSV2	f=8	33.1	56.4	58.2	**59.7**	33.9	62.6	61.6	**65.4**	64.1	53.0
	f=16	34.7	61.8	63.7	**63.9**	35.3	66.2	65.7	**68.6**	67.4	55.0
Mini-Kinetics	f=8	**70.4**	68.1	68.3	68.8	72.1	73.3	71.5	**74.1**	**74.1**	73.7
	f=16	70.5	70.9	**70.7**	70.0	72.5	75.5	73.4	**76.4**	75.6	74.5
FLOPs (G)	f=8	12.0	33.6	32.2	12.0	32.7	64.2	107.0	32.8	32.7	196
Parameters (M)	f=8	5.7	12.4	17.2	5.7	23.7	46.3	71.5	23.7	23.7	31.0

Table 3: Performance of different temporal aggregation strategies w/o temporal pooling. FLOPs of 16-frame are the double of 8-frame models, and number of parameters are the same.

ple setup without much background information. It has been shown that temporal reasoning is essential for recognition on this dataset [12]. *Kinetics* has been the most popular benchmark for deep-learning-based action approaches. It consists of 240K training videos and 20K validation videos of 400 action categories, with each video lasting 6-10 seconds. Interestingly, approaches without temporal modeling such as TSN [10] achieves strong results on this dataset, implying that modeling temporal information is not that important on this dataset. *MiT* is a recent collection of one million labeled videos, involving actions from people, animals, objects or natural phenomena. It has 339 classes and each clip is trimmed to 3 seconds long. These datasets cover a wide range of different types of videos, hence are suitable for studying various spatio-temporal representations.

We extract video frames via the FFMPEG packages and then resize the shorter side of an image to 256 while keeping the aspect ratio of the image.

4. Training and Evaluation

4.1. Mini-Datasets

Training. Table 5 illustrates the training protocol we use for all the models in our experiments. We train most of our models using a single compute node with 6 V100 GPUs and a total of 96G GPU memory with a batch size of 72 or the maximum allowed for a single node (a multiple of 6). For some of the large models (for example, I3D-ResNet50) using 32 or 64 frames, we limit the number of nodes to no more than 3, i.e. 18 GPUs, and apply synchronized batch normalization in training at a batch size of 36. We observe that such a setup generally leads to comparable model accuracy to the approaches studied in this work. Following the practice in TSN [10], we apply multi-scale augmentation and randomly crop the same 224×224 region of whole input images for training. In the meanwhile, temporal jittering is used to sample different frames from a video. Afterward, the input is normalized by the mean and standard deviation used in the original ImageNet-pretrained model.

Evaluation. In the *clip-level* accuracy setting, we sample f frames either with uniform sampling or dense sampling and then crop a 224×224 region centered at each image after resizing the shorter side of the image to 224. For uniforming sampling, we choose the middle frame of each segment to form a clip while for dense sample the first clip is used. In the *video-level* accuracy setting, m clips need to be prepared. For dense sampling, we uniformly select m points and then take f consecutive frames starting at each point.

Dataset	# of Images Train	Val	# of Classes	Duration
SSV2 [4]	168k	24k	174	3-5s@12fps
Mini-SSV2	81k	12k	87	
Kinetics [6]	240k	19k	400	6-10s@30fps
Mini-Kinetics	121k	10k	200	
MiT [8]	802k	34k	339	3s@30fps
Mini-MiT	100k	10k	200	

Mini-Something-Something and Mini-Kinetics400 are. created by randomly sampling half of classes.

Table 4: Overview of datasets.

	8-frame	16-frame	32-frame	64-frame
Weight Init.	ImageNet	8-frame	16-frame	32-frame
Epochs[1]	75 (100)	35 (45)	35 (45)	35 (45)
Learning rate	0.01			
LR scheduler[2]	cosine	multisteps	multisteps	multisteps
Weight decay	0.0005			
Optimizer	Synchronized SGD with moment 0.9			

[1]: Mini-Kinetics uses the epoch numbers mentioned in the bracket.

[2]: when the epoch number is 35, the learning rate drops 10× at the 10-th, 20-th, 30-th epoch; while drops 10× at the 15-th, 30-th, 40-th epoch when 45 epochs is used.

Table 5: Training protocol for Mini-datasets.

In the case of uniform sampling, we apply an offset i from the middle frame, where $-m/2 <= i < m/2$, to shift the sampling location at each segment. We use $m = 10$ to compute video-level accuracy in our analysis.

4.2. Full Datasets

To enable SOTA results on *Kinetics*, we follow the practices in the SlowFast paper [3] to train the models. During training, we take 64 consecutive frames from a video and sample every other frame as the input, i.e., 32 frames are fed to the model. The shorter side of a video is randomly resized to the range of [256, 320] while keeping aspect ratio, and then we randomly crop a 224×224 spatial region as the training input. We trained all models for 196 epochs, using a total batch size of 1024 with 128 GPUs, i.e. 8 samples per GPU. Batch normalization is computed on those 8 samples. We warm up the learning rate from 0.01 to 1.6 with 34 epochs linearly and then apply half-period cosine annealing schedule for the remaining epochs. We use synchronized SGD with momentum 0.9 and weight decay 0.0001. During the evaluation, we uniformly sample 10 clips from a video, and then take 3 256×256 crops from each clip whose shorter side of each clip is resized 256. The accuracy of a video is conducted by averaging over 30 predictions. On the other hand, for *SSV2* dataset, we only sample 2 clips since the video length of *SSV2* is shorter.

4.3. Transfer Learning

In transfer learning study, we used the models described in Section 6.2, i.e., 32-frame models trained with *Kinetics*, as the pretrained weights. Then, we finetuned 45 epochs with cosine annealing learning rate schedule starting with 0.01, and trained the models with a batch size of 48 with synchronized batch normalization on a 6 GPUs machine.

5. Model Implementation

Implementation. We follow the original published papers as much as we can to implement the approaches in our analysis. However, due to the differences in the backbones, some modifications are necessary to ensure a fair comparison under a common experimental framework. Here we describe how we build the networks including three backbones (InceptionV1, ResNet18 and ResNet50), four video architectures (I3D, S3D, TAM and TSN), and where to perform temporal pooling. For three backbones, we used those 2D models available on the torchvision repository (*googlenet, resnet18, resnet50*), and then used the weights in the model zoo for initializing the models either through inflation (I3D and S3D) or directly loading (TAM and TSN). Note that, for inflation, we simply copy the weights along the time dimension. Moreover, we always perform the same number of temporal pooling at the similar locations across all the backbones while training models with temporal pooling. For each backbone, there are five positions to perform *spatial pooling*, we add maximum *temporal pooling* along with the last three spatial poolings (kernel size is set to 3).

I3D. We follow I3D paper to re-implement the network [1]. We convert all 2D convolutional layer into 3D convolutions and set kernel size in temporal domain to 3 while using same spatial kernel size. For I3D-ResNet-50, we convert 3×3 convolution in bottleneck block into 3×3×3.

S3D. We follow the idea of the original S3D and R(2+1)D paper to factorize 3D convolution in the re-implemented models [11, 9]; thus, each 3D convolution in I3D becomes one 2D spatial convolution and one 1D temporal convolution. Nonetheless, the first convolution of the network is not factorized as the original papers. For InceptionV1 backbone, the difference from the original paper is the location of temporal pooling of backbone [11]. More specifically, in our implementation, we remove the temporal stride in the first convolutional layer and then add an temporal pooling layer to keep the same temporal downsampling ratio over the model. On the other hand, for ResNet backbone, we do not follow the R(2+1)D paper to expand the channels to have similar parameters to the corresponding I3D models, we simply set the output channels to the original output channel size [9] which helps us to directly load the ImageNet-pretrained weights into the model.

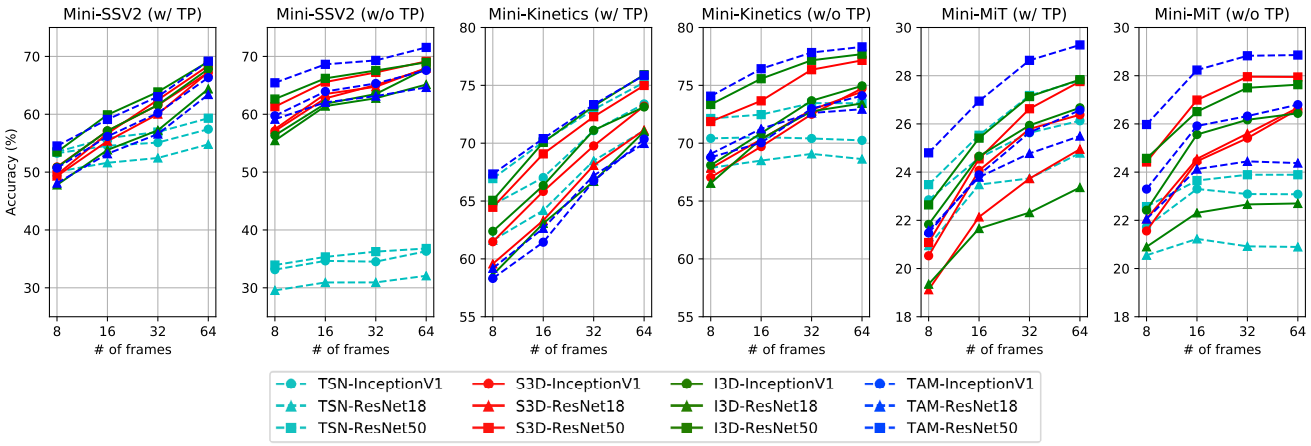

Figure 2: Top-1 accuracy of all the models with and without temporal pooling on three mini-datasets. The video architectures are separated by color while the backbones are separated by symbols. Best viewed in color.

TAM. We follow the original paper to build TAM-ResNet [2], the TAM module is inserted at the non-identity path of every residual block. For TAM-InceptionV1, we add TAM modules after the every inception module.

TSN. It does not have any temporal modeling, so it directly uses 2D models.

References

[1] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, pages 6299–6308, 2017. 3

[2] Quanfu Fan, Chun-Fu (Ricarhd) Chen, Hilde Kuehne, Marco Pistoia, and David Cox. More Is Less: Learning Efficient Video Representations by Temporal Aggregation Modules. In *NeurIPS*, 2019. 1, 4

[3] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. *arXiv:1812.03982*, 2018. 3

[4] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *ICCV*, 2017. 3

[5] Kaiming He, Ross Girshick, and Piotr Dollar. Rethinking imagenet pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 1

[6] 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:1705.06950*, 2017. 3

[7] Ji Lin, Chuang Gan, and Song Han. Temporal Shift Module for Efficient Video Understanding. In *ICCV*, 2019. 1

[8] Mathew Monfort, Alex Andonian, Bolei Zhou, Kandan Ramakrishnan, Sarah Adel Bargal, Yan Yan, Lisa Brown, Quanfu Fan, Dan Gutfreund, Carl Vondrick, et al. Moments in time dataset: one million videos for event understanding. *IEEE TPAMI*, 2019. 3

[9] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A Closer Look at Spatiotemporal Convolutions for Action Recognition. In *CVPR*, June 2018. 3

[10] 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 *ECCV*. Springer, 2016. 2

[11] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking Spatiotemporal Feature Learning: Speed-Accuracy Trade-offs in Video Classification. In *ECCV*, Sept. 2018. 3

[12] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *ECCV*, pages 803–818, 2018. 2