

---

## 3D ResNets for Action Recognition

### Update (2020/4/13)

We published a paper on arXiv.

Hirokatsu Kataoka, Tenga Wakamiya, Kensho Hara, and Yutaka Satoh,  
“Would Mega-scale Datasets Further Enhance Spatiotemporal 3D CNNs”,  
arXiv preprint, arXiv:2004.04968, 2020.

We uploaded the pretrained models described in this paper including ResNet-50 pretrained on the combined dataset with Kinetics-700 and Moments in Time.

### Update (2020/4/10)

We significantly updated our scripts. If you want to use older versions to reproduce our CVPR2018 paper, you should use the scripts in the CVPR2018 branch.

This update includes as follows: \* Refactoring whole project \* Supporting the newer PyTorch versions \* Supporting distributed training \* Supporting training and testing on the Moments in Time dataset. \* Adding R(2+1)D models \* Uploading 3D ResNet models trained on the Kinetics-700, Moments in Time, and STAIR-Actions datasets

### Summary

This is the PyTorch code for the following papers:

Hirokatsu Kataoka, Tenga Wakamiya, Kensho Hara, and Yutaka Satoh,  
“Would Mega-scale Datasets Further Enhance Spatiotemporal 3D CNNs”,  
arXiv preprint, arXiv:2004.04968, 2020.

Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh,  
“Towards Good Practice for Action Recognition with Spatiotemporal 3D Convolutions”,  
Proceedings of the International Conference on Pattern Recognition, pp. 2516-2521, 2018.

Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh,  
“Can Spatiotemporal 3D CNNs Retrace the History of 2D CNNs and ImageNet?”,  
Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6546-6555, 2018.

---

Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh,

“Learning Spatio-Temporal Features with 3D Residual Networks for Action Recognition”,  
Proceedings of the ICCV Workshop on Action, Gesture, and Emotion Recognition, 2017.

This code includes training, fine-tuning and testing on Kinetics, Moments in Time, ActivityNet, UCF-101, and HMDB-51.

## Citation

If you use this code or pre-trained models, please cite the following:

```
1 @inproceedings{hara3dcnns,  
2   author={Kensho Hara and Hirokatsu Kataoka and Yutaka Satoh},  
3   title={Can Spatiotemporal 3D CNNs Retrace the History of 2D CNNs and  
4     ImageNet?},  
5   booktitle={Proceedings of the IEEE Conference on Computer Vision and  
6     Pattern Recognition (CVPR)},  
7   pages={6546--6555},  
8   year={2018},  
9 }
```

## Pre-trained models

Pre-trained models are available here.

All models are trained on Kinetics-700 (*K*), Moments in Time (*M*), STAIR-Actions (*S*), or merged datasets of them (*KM*, *KS*, *MS*, *KMS*).

If you want to finetune the models on your dataset, you should specify the following options.

```
1 r3d18_K_200ep.pth: --model resnet --model_depth 18 --n_pretrain_classes  
2   700  
3 r3d18_KM_200ep.pth: --model resnet --model_depth 18 --  
4   n_pretrain_classes 1039  
5 r3d34_K_200ep.pth: --model resnet --model_depth 34 --n_pretrain_classes  
6   700  
7 r3d34_KM_200ep.pth: --model resnet --model_depth 34 --  
8   n_pretrain_classes 1039  
9 r3d50_K_200ep.pth: --model resnet --model_depth 50 --n_pretrain_classes  
10  700  
11 r3d50_KM_200ep.pth: --model resnet --model_depth 50 --  
12   n_pretrain_classes 1039  
13 r3d50_KMS_200ep.pth: --model resnet --model_depth 50 --  
14   n_pretrain_classes 1139  
15 r3d50_KS_200ep.pth: --model resnet --model_depth 50 --  
16   n_pretrain_classes 800  
17 r3d50_M_200ep.pth: --model resnet --model_depth 50 --n_pretrain_classes  
18  339
```

---

```
10 r3d50_MS_200ep.pth: --model resnet --model_depth 50 --
    n_pretrain_classes 439
11 r3d50_S_200ep.pth: --model resnet --model_depth 50 --n_pretrain_classes
    100
12 r3d101_K_200ep.pth: --model resnet --model_depth 101 --
    n_pretrain_classes 700
13 r3d101_KM_200ep.pth: --model resnet --model_depth 101 --
    n_pretrain_classes 1039
14 r3d152_K_200ep.pth: --model resnet --model_depth 152 --
    n_pretrain_classes 700
15 r3d152_KM_200ep.pth: --model resnet --model_depth 152 --
    n_pretrain_classes 1039
16 r3d200_K_200ep.pth: --model resnet --model_depth 200 --
    n_pretrain_classes 700
17 r3d200_KM_200ep.pth: --model resnet --model_depth 200 --
    n_pretrain_classes 1039
```

Old pretrained models are still available here.

However, some modifications are required to use the old pretrained models in the current scripts.

## Requirements

- PyTorch (ver. 0.4+ required)

```
1 conda install pytorch torchvision cudatoolkit=10.1 -c soumith
```

- FFmpeg, FFprobe
- Python 3

## Preparation

### ActivityNet

- Download videos using the official crawler.
- Convert from avi to jpg files using `util_scripts/generate_video_jpgs.py`

```
1 python -m util_scripts.generate_video_jpgs mp4_video_dir_path
    jpg_video_dir_path activitynet
```

- Add fps information into the json file `util_scripts/add_fps_into_activitynet_json.py`

```
1 python -m util_scripts.add_fps_into_activitynet_json mp4_video_dir_path
    json_file_path
```

---

## Kinetics

- Download videos using the official crawler.
  - Locate test set in `video_directory/test`.
- Convert from avi to jpg files using `util_scripts/generate_video_jpgs.py`

```
1 python -m util_scripts.generate_video_jpgs mp4_video_dir_path
  jpg_video_dir_path kinetics
```

- Generate annotation file in json format similar to ActivityNet using `util_scripts/kinetics_json.py`
  - The CSV files (`kinetics_{train, val, test}.csv`) are included in the crawler.

```
1 python -m util_scripts.kinetics_json csv_dir_path 700
  jpg_video_dir_path jpg dst_json_path
```

## UCF-101

- Download videos and train/test splits here.
- Convert from avi to jpg files using `util_scripts/generate_video_jpgs.py`

```
1 python -m util_scripts.generate_video_jpgs avi_video_dir_path
  jpg_video_dir_path ucf101
```

- Generate annotation file in json format similar to ActivityNet using `util_scripts/ucf101_json.py`
  - `annotation_dir_path` includes `classInd.txt`, `trainlist0{1, 2, 3}.txt`, `testlist0{1, 2, 3}.txt`

```
1 python -m util_scripts.ucf101_json annotation_dir_path
  jpg_video_dir_path dst_json_path
```

## HMDB-51

- Download videos and train/test splits here.
- Convert from avi to jpg files using `util_scripts/generate_video_jpgs.py`

```
1 python -m util_scripts.generate_video_jpgs avi_video_dir_path
  jpg_video_dir_path hmdb51
```

- 
- Generate annotation file in json format similar to ActivityNet using `util_scripts/hmdb51_json.py`
    - `annotation_dir_path` includes `brush_hair_test_split1.txt`, ...

```
1 python -m util_scripts.hmdb51_json annotation_dir_path  
   jpg_video_dir_path dst_json_path
```

## Running the code

Assume the structure of data directories is the following:

```
1 ~/
2   data/
3     kinetics_videos/
4       jpg/
5         .../ (directories of class names)
6         .../ (directories of video names)
7         ... (jpg files)
8     results/
9       save_100.pth
10    kinetics.json
```

Confirm all options.

```
1 python main.py -h
```

Train ResNets-50 on the Kinetics-700 dataset (700 classes) with 4 CPU threads (for data loading).  
Batch size is 128.

Save models at every 5 epochs. All GPUs is used for the training. If you want a part of GPUs, use  
`CUDA_VISIBLE_DEVICES=....`

```
1 python main.py --root_path ~/data --video_path kinetics_videos/jpg --  
   annotation_path kinetics.json \  
2 --result_path results --dataset kinetics --model resnet \  
3 --model_depth 50 --n_classes 700 --batch_size 128 --n_threads 4 --  
   checkpoint 5
```

Continue Training from epoch 101. (`~/data/results/save_100.pth` is loaded.)

```
1 python main.py --root_path ~/data --video_path kinetics_videos/jpg --  
   annotation_path kinetics.json \  
2 --result_path results --dataset kinetics --resume_path results/save_100  
   .pth \  
3 --model_depth 50 --n_classes 700 --batch_size 128 --n_threads 4 --  
   checkpoint 5
```

---

Calculate top-5 class probabilities of each video using a trained model (~data/results/save\_200.pth.)  
Note that `inference_batch_size` should be small because actual batch size is calculated by `inference_batch_size * (n_video_frames / inference_stride)`.

```
1 python main.py --root_path ~/data --video_path kinetics_videos/jpg --
  annotation_path kinetics.json \
2 --result_path results --dataset kinetics --resume_path results/save_200
  .pth \
3 --model_depth 50 --n_classes 700 --n_threads 4 --no_train --no_val --
  inference --output_topk 5 --inference_batch_size 1
```

Evaluate top-1 video accuracy of a recognition result (~data/results/val.json).

```
1 python -m util_scripts.eval_accuracy ~/data/kinetics.json ~/data/
  results/val.json --subset val -k 1 --ignore
```

Fine-tune fc layers of a pretrained model (~data/models/resnet-50-kinetics.pth) on UCF-101.

```
1 python main.py --root_path ~/data --video_path ucf101_videos/jpg --
  annotation_path ucf101_01.json \
2 --result_path results --dataset ucf101 --n_classes 101 --
  n_pretrain_classes 700 \
3 --pretrain_path models/resnet-50-kinetics.pth --ft_begin_module fc \
4 --model resnet --model_depth 50 --batch_size 128 --n_threads 4 --
  checkpoint 5
```