C3D User Guide

Du Tran (Last modified Mar 20, 2017)

C3D-v1.1 is released with new models (Mar 01, 2017).

- No documentation for v1.1 yet, but some examples for feature extraction, training, and fine-tuning are provided.

The below guide was written for C3D-v1.0

I. C3D Feature Extraction

If you have installed C3D successfully (same as install caffe and its dependences), following the following steps:

+ Download pre-trained model and save it to YOUR_C3D_HOME/examples/c3d_feature_extraction
+ Change directory to YOUR_C3D_HOME/examples/c3d_feature_extraction
+ Run: `sh c3d_sport1m_feature_extraction_frm.sh` or `sh c3d_sport1m_feature_extraction_video.sh`

If you can run the examples successfully, then you should find the extracted features in the output folders.

If you run to "*out of memory*" error, then you should consider to reduce the batch-size (see section I.B)

If you can run feature extraction with frames successfully, but fail with the video inputs. The cause may come from video codecs. Make sure you had compiled your OpenCV and Ffmpeg with shared-flags are on.

+ make sure that '`shuffle: false`' in your data layer when you use C3D as a feature extractor. This help us to keep the correspondences between the input clips and the output features.

I.A Extract C3D features for your own videos or frames

a. Prepare your input files

- C3D allows you to use video inputs as sequences of frames or video files. In the case of video files (.mp4, .avi, .mov), make sure that your machine has codecs, opencv, and ffmpeg installed properly. In the case of using frames, C3D assumes that each video is a folder with frames which are numbered starting from 1 to N (number of frames). The frame names are formatted as "`video_folder/%06d.jpg`".

- Note that: frame numbers starting from 1 (e.g. 1..N) for using frame as inputs, and starting from 0 (e.g. [0..N-1]) for using video as inputs.

b. Prepare your setting files

- There are two setting files you need to prepare: input-list and output prefix. In the provided example, they are: input_list_frm.txt, input_list_video.txt, and output_list_prefix.txt in YOUR_C3D_HOME/examples/c3d_feature_extraction/prototxt
- The input list file is a text file where each line contain information for a clip that you are inputting into C3D for extracting features. Each line has the following format:

  `<string_path> <starting_frame> <label>`

  where `<label>` is only used for training, testing, or fine-tuning, but **NOT for extracting features**, thus can be ignored (in the provided example, they are filled with 0s). For `<string_path>`, we have two cases. For the setting with video file inputs, `<string_path>` is the full path and filename of the video (e.g. `input/avi/v_ApplyEyeMakeup_g01_c01.avi`). For the setting with frame inputs, `<string_path>` is the full path to the folder containing frames of the video (e.g. `input/frm/v_ApplyEyeMakeup_g01_c01/`). Finally, the <starting_frame> is used to specify the starting frame of the clip. We note that C3D extract feature of 16-frame-long clips. For example, if starting frame is 1, then you are extracting features for the clip (from the video specified by <string_path>) from frame 1 to 16. If starting frame is 17, then the clip of interest is from frame 17 to 32. *Note that in the provided examples, we have sampled clips from videos with step size (or stride) of 16 frames. You can use different sampling step-size: e.g. as dense as every 1 frame or sparser e.g. every 32 frames.*
- The output prefix file is used to specify the locations for extracting features to be saved. Each line is formatted as

  `<output_prefix>`

  Each line in the prefix file is corresponded to a line in the input list file (in the same order, e.g. line 1 in prefix file is the output prefix for the clip of line 1 in the list file). C3D will save features are `output_prefix.[feature_name]` (e.g. `prefix.fc6`). It is recommend that for each video, you should create an output folder and the prefix lines are formatted as `sprintf("output_folder/%06d", starting_frame)`. That means each clip has its starting frame as identifier and file extensions are used for different features. Remember to create output folders, as C3D does not create them.

c. Extract C3D features

- Assume that you have prepared your setting files, then you need to modify the prototxt file to point to the input list file. In the prototxt file, looks for line:

```
source: "prototxt/input_list_frm.txt"
```

Also remember set the `use_image: true` if you use images as inputs or `false` if use videos as inputs.

- Use `extract_image_features` tool to extract features. The arguments used by this tools is follow:

```
extract_image_features.bin <feature_extractor_prototxt_file>
<c3d_pre_trained_model> <gpu_id> <mini_batch_size>
<number_of_mini_batches> <output_prefix_file> <feature_name1>
<feature_name2> ...
```

In which:

+ `<feature_extractor_prototxt_file>`: is prototxt file (provided in example) which points to your input list file.
+ `<c3d_pre_trained_model>`: is the C3D pre-trained model that you downloaded.
+ `<gpu_id>`: GPU ID you would like to run (starting from 0), if this is set to -1, then it will use CPU.
+ `<mini_batch_size>`: your mini batch size. Default is 50, but you can modify this number, depend on your GPU memory.
+ `<number_of_mini_batches>`: Number of mini-batches you want to extract features. For examples, if you have 100 clips to extract features and you are using mini-batch size of 50, then this parameter should be set to 2. However, if you have 101 clips to be extracted features, then this number should be set to 3.
+ `<output_prefix_file>`: Your output prefix file.
+ `<feature_name1>`: You can list as many feature names as possible as long as they are in the names of the output blobs of the network (see prototxt file for all layers, but they look like fc6-1, fc7-1, fc8-1, pool5, conv5b, prob,...).

You can find the following command line provided in the example as below:

```
GLOG_logtosterr=1 ../../build/tools/extract_image_features.bin
prototxt/c3d_sport1m_feature_extractor_frm.prototxt
conv3d_deepnetA_sport1m_iter_1900000 0 50 1
prototxt/output_list_prefix.txt fc7-1 fc6-1 prob
```

I.B Extract C3D features with smaller or larger batch-size

In case you have more or less memory, you can adjust the mini-batch size (larger or smaller than 50). To do that, you need to change this parameter in the prototxt file of the network (find

line e.g. `batch_size: 50`). And you also need to input the newly-adjust parameters of `<mini_batch_size>` and `<number_of_mini_batches>` in the command line.

After extracted C3D features, you can use the provided MATLAB script (`read_binary_blob.m`) to read the features for further analysis.

II. Train 3D ConvNet

A. Compute volume mean from list

This tool allows you to compute volume mean for you own dataset which can be useful for both training from scratch or fine-tuning C3D on your own dataset.

Usage:

```
GLOG_logtostderr=1 compute_volume_mean_from_list input_chunk_list
length height width sampling_rate output_file [dropping rate]
```

Arguments:

`input_chunk_list`: the same as the list file used in feature extraction
`length`: the length of the clip used in training (e.g. 16)
`height, width`: size of frame e.g. 128, 171
`sampling_rate`: this is used to adjust the frame rate in you clip (e.g. clip length=16, sampling=1, then your clip is a 16-consecutive frame video chunk. Or if clip length=16, while sampling rate=2, then you clip is 32-frame long clips, but you sample 1 of every 2 frames)
`output_file`: the output mean file.
`dropping_rate`: In case you dataset is too large (e.g. 1M), you may want to compute the mean from a subset of your clips. Setting this to n, meaning the dropping rate is 1:n, choose 1 sample among every n clips for computing mean.

If you prefer to use `mean_value`, instead of `volume_mean` file, then you can set this `mean_value` field in your data layer. This is equivalent to the volume mean with all values are set to `mean_value`.

B. Train your own network from scratch

Assume you have your input_data_list, your train/test prototxt and your solver prototxt, you can use train_net to train the network.

C. An example of training from scratch on UCF101

+ Change directory to YOUR_C3D_HOME/examples/c3d_train_ucf101/
+ run `sh create_volume_mean.sh` to compute the volume mean file
+ run `sh train_ucf101.sh` to train, expect a couple days to finish
+ run `sh test_ucf101.sh` to test, expect about 15' to complete and you should have ~45% accuracy (this is **clip** accuracy)

III. Fine-tune C3D

Assume you have download the C3D pre-trained model. You can try the fine-tuning example, by:

+ Change directory to YOUR_C3D_HOME/examples/c3d_finetuning
+ Run: `sh ucf101_finetuning.sh`
+ When fine-tuning is done, you can test your fine-tuned model by running: `sh ucf101_testing.sh`
+ **[Added 05/10/2016]** In case you don't have time to fine-tune C3D on UCF101 yourself, here we provide the C3D model fine-tuned on UCF101: https://www.dropbox.com/s/mkc9q7g4wnqnmcv/c3d_ucf101_finetune_whole_iter_20000 Simply download this model to YOUR_C3D_HOME/examples/c3d_finetuning and run `sh ucf101_testing.sh` (assume that you have made sure your `test_01.lst` file points to your UCF101 frames). This will give an accuracy of 80.19% (clip accuracy). NOTE: this model is fine-tuned on UCF101 "train split 1", thus it is only valid to test on "test split 1".

FAQs

- Do we have MATLAB or Python wrappers for extracting C3D features?
  Unfortunately, we don't have them yet.
- Can I use C3D on a CPU?
  This version of C3D is built on an old caffe branch, so there is no 'CPU_ONLY' mode in Makefile. But you can do that by the following:

    - Compile C3D as normal (it requires CUDA driver to compile, but if you don't have GPU, you still can run on a CPU).
    - To train using CPU, you can modify solver file to solver_mode: CPU (see here https://github.com/facebook/C3D/blob/master/examples/c3d_train_ucf101/conv3d_ucf101_solver.prototxt#L19)
    - To test using CPU, in the command line use CPU instead of GPU (https://github.com/facebook/C3D/blob/master/examples/c3d_train_ucf101/test_ucf101.sh) and no GPU_ID is needed.
    - To extract features with CPU, use GPU_ID = -1, instead of 0 as in the example here (https://github.com/facebook/C3D/blob/master/examples/c3d_feature_extraction/c3d_sport1m_feature_extraction_frm.sh)

- Email me your questions? (trandu -at- fb.com) or post on github. This is more preferred because I sometime miss some emails. Github keeps tracks much better.