

 **google** / **youtube-8m**

Starter code for working with the YouTube-8M dataset. <https://research.google.com/youtube8m/>

#youtube-8m

 **219** commits

 **4** branches

 **0** releases

 **14** contributors

 Apache-2.0

 Branch: **master** ▼














New pull request












Create new file

Upload files

Find file

Clone or download ▼

 samihaia Merge pull request #54 from google/fix-readme-link-1 ...	Latest commit 2a652b0 on 21 Oct
 feature_extractor	make top-level README point to feature_extractor/README + fix README ... 2 months ago
 CONTRIBUTING.md	Setup.py autogeneration is working again. Remove the setup.py file t... 10 months ago
 LICENSE	Setup.py autogeneration is working again. Remove the setup.py file t... 10 months ago
 README.md	Fix README.md link 2 months ago
 __init__.py	Setup.py autogeneration is working again. Remove the setup.py file t... 10 months ago
 average_precision_calculator.py	Add support for distributed TensorFlow. 10 months ago
 cloudml-4gpu.yaml	Add support for multiple GPUs (#42) 9 months ago
 cloudml-gpu-distributed.yaml	Fixing a bug that prevents model recovery when using multiple worker ... 10 months ago
 cloudml-gpu.yaml	Add support for multiple GPUs (#42) 9 months ago
 convert_prediction_from_json_to_c...	Add support for batch prediction. (#37) 9 months ago
 eval.py	Add support for multiple GPUs (#42) 9 months ago
 eval_util.py	Add support for distributed TensorFlow. 10 months ago

 export_model.py	Add support for multiple GPUs (#42)	9 months ago
 frame_level_models.py	Merge pull request #45 from rickymf4/master	8 months ago
 inference.py	Fix typo in flag-comment	7 months ago
 losses.py	Add support for distributed TensorFlow.	10 months ago
 mean_average_precision_calculato...	Add support for distributed TensorFlow.	10 months ago
 model_utils.py	Add Moe and DBoF models	10 months ago
 models.py	Add Moe and DBoF models	10 months ago
 readers.py	Add support for multiple GPUs (#42)	9 months ago
 train.py	Removed useless code, for the condition is always False.	9 months ago
 utils.py	Define xrange for Python 3	4 months ago
 video_level_models.py	Fix Regularization (#21)	10 months ago

README.md

YouTube-8M Tensorflow Starter Code

This repo contains starter code for training and evaluating machine learning models over the [YouTube-8M](#) dataset. The code gives an end-to-end working example for reading the dataset, training a TensorFlow model, and evaluating the performance of the model. Out of the box, you can train several [model architectures](#) over either frame-level or video-level features. The code can easily be extended to train your own custom-defined models.

It is possible to train and evaluate on YouTube-8M in two ways: on Google Cloud or on your own machine. This README provides instructions for both.

Table of Contents

- [Running on Google's Cloud Machine Learning Platform](#)
 - [Requirements](#)
 - [Testing Locally](#)
 - [Training on the Cloud over Video-Level Features](#)
 - [Evaluation and Inference](#)
 - [Accessing Files on Google Cloud](#)
 - [Using Frame-Level Features](#)
 - [Using Audio Features](#)
 - [Using Larger Machine Types](#)
- [Running on Your Own Machine](#)
 - [Requirements](#)
 - [Training on Video-Level Features](#)
 - [Evaluation and Inference](#)
 - [Using Frame-Level Features](#)
 - [Using Audio Features](#)
 - [Using GPUs](#)
 - [Ground-Truth Label Files](#)
- [Overview of Models](#)
 - [Video-Level Models](#)
 - [Frame-Level Models](#)
- [Create Your Own Dataset Files](#)
- [Overview of Files](#)
 - [Training](#)
 - [Evaluation](#)

- [Inference](#)
- [Misc](#)
- [About This Project](#)

Running on Google's Cloud Machine Learning Platform

Requirements

This option requires you to have an appropriately configured Google Cloud Platform account. To create and configure your account, please make sure you follow the instructions [here](#). If you are participating in the Google Cloud & YouTube-8M Video Understanding Challenge hosted on [kaggle](#), see [these instructions](#) instead.

Please also verify that you have Python 2.7+ and Tensorflow 1.0.0 or higher installed by running the following commands:

```
python --version
python -c 'import tensorflow as tf; print(tf.__version__)'
```

Testing Locally

All gcloud commands should be done from the directory *immediately above* the source code. You should be able to see the source code directory if you run 'ls'.

As you are developing your own models, you will want to test them quickly to flush out simple problems without having to submit them to the cloud. You can use the `gcloud beta ml local` set of commands for that.

Here is an example command line for video-level training:

```
gcloud ml-engine local train \
  --package-path=youtube-8m --module-name=youtube-8m.train -- \
```

```
--train_data_pattern='gs://youtube8m-ml/1/video_level/train/train*.tfrecord' \
--train_dir=/tmp/yt8m_train --model=LogisticModel --start_new_model
```

You might want to download some training shards locally to speed things up and allow you to work offline. The command below will copy 10 out of the 4096 training data files to the current directory.

```
# Downloads 55MB of data.
gsutil cp gs://us.data.yt8m.org/1/video_level/train/traina[0-9].tfrecord .
```

Once you download the files, you can point the job to them using the 'train_data_pattern' argument (i.e. instead of pointing to the "gs://..." files, you point to the local files).

Once your model is working locally, you can scale up on the Cloud which is described below.

Training on the Cloud over Video-Level Features

The following commands will train a model on Google Cloud over video-level features.

```
BUCKET_NAME=gs://${USER}_yt8m_train_bucket
# (One Time) Create a storage bucket to store training logs and checkpoints.
gsutil mb -l us-east1 $BUCKET_NAME
# Submit the training job.
JOB_NAME=yt8m_train_$(date +%Y%m%d_%H%M%S); gcloud --verbosity=debug ml-engine jobs \
submit training $JOB_NAME \
--package-path=youtube-8m --module-name=youtube-8m.train \
--staging-bucket=$BUCKET_NAME --region=us-east1 \
--config=youtube-8m/cloudml-gpu.yaml \
-- --train_data_pattern='gs://youtube8m-ml-us-east1/1/video_level/train/train*.tfrecord' \
--model=LogisticModel \
--train_dir=$BUCKET_NAME/yt8m_train_video_level_logistic_model
```

In the 'gsutil' command above, the 'package-path' flag refers to the directory containing the 'train.py' script and more generally the python package which should be deployed to the cloud worker. The module-name refers to the specific python script which should be executed (in this case the train module).

It may take several minutes before the job starts running on Google Cloud. When it starts you will see outputs like the following:

```
training step 270| Hit@1: 0.68 PERR: 0.52 Loss: 638.453
training step 271| Hit@1: 0.66 PERR: 0.49 Loss: 635.537
training step 272| Hit@1: 0.70 PERR: 0.52 Loss: 637.564
```

At this point you can disconnect your console by pressing "ctrl-c". The model will continue to train indefinitely in the Cloud. Later, you can check on its progress or halt the job by visiting the [Google Cloud ML Jobs console](#).

You can train many jobs at once and use tensorboard to compare their performance visually.

```
tensorboard --logdir=$BUCKET_NAME --port=8080
```

Once tensorboard is running, you can access it at the following url: <http://localhost:8080>. If you are using Google Cloud Shell, you can instead click the Web Preview button on the upper left corner of the Cloud Shell window and select "Preview on port 8080". This will bring up a new browser tab with the Tensorboard view.

Evaluation and Inference

Here's how to evaluate a model on the validation dataset:

```
JOB_TO_EVAL=yt8m_train_video_level_logistic_model
JOB_NAME=yt8m_eval_$(date +%Y%m%d_%H%M%S); gcloud --verbosity=debug ml-engine jobs \
submit training $JOB_NAME \
--package-path=youtube-8m --module-name=youtube-8m.eval \
--staging-bucket=$BUCKET_NAME --region=us-east1 \
```

```
--config=youtube-8m/cloudml-gpu.yaml \
-- --eval_data_pattern='gs://youtube8m-ml-us-east1/1/video_level/validate/validate*.tfrecord' \
--model=LogisticModel \
--train_dir=$BUCKET_NAME/${JOB_TO_EVAL} --run_once=True
```

And here's how to perform inference with a model on the test set:

```
JOB_TO_EVAL=yt8m_train_video_level_logistic_model
JOB_NAME=yt8m_inference_$(date +%Y%m%d_%H%M%S); gcloud --verbosity=debug ml-engine jobs \
submit training $JOB_NAME \
--package-path=youtube-8m --module-name=youtube-8m.inference \
--staging-bucket=$BUCKET_NAME --region=us-east1 \
--config=youtube-8m/cloudml-gpu.yaml \
-- --input_data_pattern='gs://youtube8m-ml/1/video_level/test/test*.tfrecord' \
--train_dir=$BUCKET_NAME/${JOB_TO_EVAL} \
--output_file=$BUCKET_NAME/${JOB_TO_EVAL}/predictions.csv
```

Note the confusing use of 'training' in the above gcloud commands. Despite the name, the 'training' argument really just offers a cloud hosted python/tensorflow service. From the point of view of the Cloud Platform, there is no distinction between our training and inference jobs. The Cloud ML platform also offers specialized functionality for prediction with Tensorflow models, but discussing that is beyond the scope of this readme.

Once these job starts executing you will see outputs similar to the following for the evaluation code:

```
examples_processed: 1024 | global_step 447044 | Batch Hit@1: 0.782 | Batch PERR: 0.637 | Batch Loss:
7.821 | Examples_per_sec: 834.658
```

and the following for the inference code:

```
num examples processed: 8192 elapsed seconds: 14.85
```

Accessing Files on Google Cloud

You can browse the storage buckets you created on Google Cloud, for example, to access the trained models, prediction CSV files, etc. by visiting the [Google Cloud storage browser](#).

Alternatively, you can use the 'gsutil' command to download the files directly. For example, to download the output of the inference code from the previous section to your local machine, run:

```
gsutil cp $BUCKET_NAME/${JOB_TO_EVAL}/predictions.csv .
```

Using Frame-Level Features

Append

```
--frame_features=True --model=FrameLevelLogisticModel --feature_names="rgb" \  
--feature_sizes="1024" --batch_size=128 \  
--train_dir=$BUCKET_NAME/yt8m_train_frame_level_logistic_model
```

to the 'gcloud' commands given above, and change 'video_level' in paths to 'frame_level'. Here is a sample command to kick-off a frame-level job:

```
JOB_NAME=yt8m_train_$(date +%Y%m%d_%H%M%S); gcloud --verbosity=debug ml-engine jobs \  
submit training $JOB_NAME \  
--package-path=youtube-8m --module-name=youtube-8m.train \  
--staging-bucket=$BUCKET_NAME --region=us-east1 \  
--config=youtube-8m/cloudml-gpu.yaml \  
-- --train_data_pattern='gs://youtube8m-ml-us-east1/1/frame_level/train/train*.tfrecord' \  
--frame_features=True --model=FrameLevelLogisticModel --feature_names="rgb" \  
--feature_sizes="1024" --batch_size=128 \  
--train_dir=$BUCKET_NAME/yt8m_train_frame_level_logistic_model
```


The 'FrameLevelLogisticModel' is designed to provide equivalent results to a logistic model trained over the video-level features. Please look at the 'video_level_models.py' or 'frame_level_models.py' files to see how to implement your own models.

Using Audio Features

The feature files (both Frame-Level and Video-Level) contain two sets of features: 1) visual and 2) audio. The code defaults to using the visual features only, but it is possible to use audio features instead of (or besides) visual features. To specify the (combination of) features to use you must set `--feature_names` and `--feature_sizes` flags. The visual and audio features are called 'rgb' and 'audio' and have 1024 and 128 dimensions, respectively. The two flags take a comma-separated list of values in string. For example, to use audio-visual Video-Level features the flags must be set as follows:

```
--feature_names="mean_rgb, mean_audio" --feature_sizes="1024, 128"
```

Similarly, to use audio-visual Frame-Level features use:

```
--feature_names="rgb, audio" --feature_sizes="1024, 128"
```

NOTE: Make sure the set of features and the order in which they appear in the lists provided to the two flags above match. Also, the order must match when running training, evaluation, or inference.

Using Larger Machine Types

Some complex frame-level models can take as long as a week to converge when using only one GPU. You can train these models more quickly by using more powerful machine types which have additional GPUs. To use a configuration with 4 GPUs, replace the argument to `--config` with `youtube-8m/cloudml-4gpu.yaml`. Be careful with this argument as it will also increase the rate you are charged by a factor of 4 as well.

Running on Your Own Machine

Requirements

The starter code requires Tensorflow. If you haven't installed it yet, follow the instructions on [tensorflow.org](https://www.tensorflow.org). This code has been tested with Tensorflow 1.0.0. Going forward, we will continue to target the latest released version of Tensorflow.

Please verify that you have Python 2.7+ and Tensorflow 1.0.0 or higher installed by running the following commands:

```
python --version
python -c 'import tensorflow as tf; print(tf.__version__)'
```

You can find complete instructions for downloading the dataset on the [YouTube-8M website](https://www.tensorflow.org/datasets/catalog/youtube8m). We recommend downloading the smaller video-level features dataset first when getting started. To do that, run:

```
mkdir -p features; cd features
curl data.yt8m.org/download.py | partition=1/video_level/train mirror=us python
```

This will download the full set of video level features, which takes up 31GB of space. If you are located outside of North America, you should change the flag 'mirror' to 'eu' for Europe or 'asia' for Asia to speed up the transfer of the files.

Change 'train' to 'validate'/'test' and re-run the command to download the other splits of the dataset.

Change 'video_level' to 'frame_level' to download the frame-level features. The complete frame-level features take about 1.71TB of space. You can set the environment variable 'shard' to 'm,n' to download only m/n-th of the data. For example, to download 1/100-th of the frame-level features from the training set, run:

```
curl data.yt8m.org/download.py | shard=1,100 partition=1/frame_level/train mirror=us python
```

Training on Video-Level Features

To start training a logistic model on the video-level features, run

```
MODEL_DIR=/tmp/yt8m
python train.py --train_data_pattern='/path/to/features/train*.tfrecord' --model=LogisticModel --train_dir=
```



Since the dataset is sharded into 4096 individual files, we use a wildcard (*) to represent all of those files.

By default, the training code will frequently write *checkpoint* files (i.e. values of all trainable parameters, at the current training iteration). These will be written to the `--train_dir`. If you re-use a `--train_dir`, the trainer will first restore the latest checkpoint written in that directory. This only works if the architecture of the checkpoint matches the graph created by the training code. If you are in active development/debugging phase, consider adding `--start_new_model` flag to your run configuration.

Evaluation and Inference

To evaluate the model, run

```
python eval.py --eval_data_pattern='/path/to/features/validate*.tfrecord' --model=LogisticModel --train_dir
```




As the model is training or evaluating, you can view the results on tensorboard by running

```
tensorboard --logdir=$MODEL_DIR
```

and navigating to <http://localhost:6006> in your web browser.

When you are happy with your model, you can generate a csv file of predictions from it by running

```
python inference.py --output_file=$MODEL_DIR/video_level_logistic_model/predictions.csv --input_data_patter
```



This will output the top 20 predicted labels from the model for every example to 'predictions.csv'.

Using Frame-Level Features

Follow the same instructions as above, appending `--frame_features=True --model=FrameLevelLogisticModel --feature_names="rgb" --feature_sizes="1024" --train_dir=$MODEL_DIR/frame_level_logistic_model` for the 'train.py', 'eval.py', and 'inference.py' scripts.

The 'FrameLevelLogisticModel' is designed to provide equivalent results to a logistic model trained over the video-level features. Please look at the 'models.py' file to see how to implement your own models.

Using Audio Features

See [Using Audio Features](#) section above.

Using GPUs

If your Tensorflow installation has GPU support, this code will make use of all of your compatible GPUs. You can verify your installation by running

```
python -c 'import tensorflow as tf; tf.Session()'
```

This will print out something like the following for each of your compatible GPUs.

```
I tensorflow/core/common_runtime/gpu/gpu_init.cc:102] Found device 0 with properties:
name: Tesla M40
major: 5 minor: 2 memoryClockRate (GHz) 1.112
pciBusID 0000:04:00.0
Total memory: 11.25GiB
```

```
Free memory: 11.09GiB
```

```
...
```

If at least one GPU was found, the forward and backward passes will be computed with the GPUs, whereas the CPU will be used primarily for the input and output pipelines. If you have multiple GPUs, each of them will be given a full batch of examples, and the resulting gradients will be summed together before being applied. This will increase your effective batch size. For example, if you set `batch_size=128` and you have 4 GPUs, this will result in 512 examples being evaluated every training step.

Ground-Truth Label Files

We also provide CSV files containing the ground-truth label information of the 'train' and 'validation' partitions of the dataset. These files can be downloaded using 'gsutil' command:

```
gsutil cp gs://us.data.yt8m.org/1/ground_truth_labels/train_labels.csv /destination/folder/  
gsutil cp gs://us.data.yt8m.org/1/ground_truth_labels/validate_labels.csv /destination/folder/
```

or directly using the following links:

- http://us.data.yt8m.org/1/ground_truth_labels/train_labels.csv
- http://us.data.yt8m.org/1/ground_truth_labels/validate_labels.csv

Each line in the files starts with the video id and is followed by the list of ground-truth labels corresponding to that video. For example, for a video with id 'VIDEO_ID' and two labels 'LABEL1' and 'LABEL2' we store the following line:

```
VIDEO_ID, LABEL1 LABEL2
```

Overview of Models

This sample code contains implementations of the models given in the [YouTube-8M technical report](#).

Video-Level Models

- `LogisticModel` : Linear projection of the output features into the label space, followed by a sigmoid function to convert logit values to probabilities.
- `MoeModel` : A per-class softmax distribution over a configurable number of logistic classifiers. One of the classifiers in the mixture is not trained, and always predicts 0.

Frame-Level Models

- `LstmModel` : Processes the features for each frame using a multi-layered LSTM neural net. The final internal state of the LSTM is input to a video-level model for classification. Note that you will need to change the learning rate to 0.001 when using this model.
- `DbofModel` : Projects the features for each frame into a higher dimensional 'clustering' space, pools across frames in that space, and then uses a video-level model to classify the now aggregated features.
- `FrameLevelLogisticModel` : Equivalent to 'LogisticModel', but performs average-pooling on the fly over frame-level features rather than using pre-aggregated features.

Create Your Own Dataset Files

You can create your dataset files from your own videos. Our [feature extractor](#) code creates `tfrecord` files, identical to our dataset files. You can use our starter code to train on the `tfrecord` files output by the feature extractor. In addition, you can fine-tune your YouTube-8M models on your new dataset.

Overview of Files

Training

- `train.py` : The primary script for training models.
- `losses.py` : Contains definitions for loss functions.
- `models.py` : Contains the base class for defining a model.
- `video_level_models.py` : Contains definitions for models that take aggregated features as input.
- `frame_level_models.py` : Contains definitions for models that take frame- level features as input.
- `model_util.py` : Contains functions that are of general utility for implementing models.
- `export_model.py` : Provides a class to export a model during training for later use in batch prediction.
- `readers.py` : Contains definitions for the Video dataset and Frame dataset readers.

Evaluation

- `eval.py` : The primary script for evaluating models.
- `eval_util.py` : Provides a class that calculates all evaluation metrics.
- `average_precision_calculator.py` : Functions for calculating average precision.
- `mean_average_precision_calculator.py` : Functions for calculating mean average precision.

Inference

- `inference.py` : Generates an output file containing predictions of the model over a set of videos.

Misc

- `README.md` : This documentation.
- `utils.py` : Common functions.
- `convert_prediction_from_json_to_csv.py` : Converts the JSON output of batch prediction into a CSV file for submission.

About This Project

This project is meant help people quickly get started working with the [YouTube-8M](#) dataset. This is not an official Google product.