YouTube-8M Tensorflow Starter Code

DISCLAIMER: This project is still under development. No support will be provided during the development phase.

This repo contains starter code (written in TensorFlow 2.x) for training and evaluating machine learning models over the YouTube-8M dataset. This is the Tensorflow2 version of the original starter code: YouTube-8M Tensorflow Starter Code which was tested on Tensorflow 1.14. (The code gives an end-to-end working example for reading the dataset, training a TensorFlow model, and evaluating the performance of the model). Functionalities are maintained, while necessary migrations were done to accomodate running on tf2 environment.

Requirements

The starter code requires Tensorflow. If you haven't installed it yet, follow the instructions on tensorflow.org. This code has been tested with Tensorflow 2.4.0. Going forward, we will continue to target the latest released version of Tensorflow.

Please verify that you have Python 3.6+ and Tensorflow 2.4.0 or higher installed by running the following commands:

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

Refer to the instructions here for using the model in this repo. Make sure to add the models folder to your Python path.

Using GPUs

If your Tensorflow installation has GPU support (which should have been provided with `pip install tensorflow` for any version above Tensorflow 1.15), this code will make use of all of your compatible GPUs. You can verify your installation by running

```
tf.config.list_physical_devices('GPU')
```

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

```
I tensorflow/core/common_runtime/gpu/gpu_device.cc:1720]
Found device 0 with properties:
pciBusID: 0000:00:04.0 name: Tesla P100-PCIE-16GB computeCapability: 6.0
coreClock: 1.3285GHz coreCount: 56 deviceMemorySize: 15.90GiB
deviceMemoryBandwidth: 681.88GiB/s
...
```

Train and inference

Train video-level model on frame-level features and inference at segment-level.

Train using the config file.

Create a YAML or JSON file for specifying the parameters to be overridden. Working examples can be found in yt8m/experiments directory.

```
task:
  model:
    cluster_size: 2048
```

```
    hidden_size: 2048
    add_batch_norm: true
    sample_random_frames: true
    is_training: true
    activation: "relu6"
    pooling_method: "average"
    yt8m_agg_classifier_model: "MoeModel"
  train_data:
    segment_labels: false
    temporal_stride: 1
    num_devices: 1
    input_path: 'gs://youtube8m-ml/2/frame/train/train*.tfrecord'
    num_examples: 3888919
...
```

The code can be run in different modes: `train / train_and_eval / eval`. Run `train.py` and specify which mode you wish to execute. Training is done using frame-level features with video-level labels, while inference can be done at segment-level. Setting `segment_labels=True` in your configuration forces the segment level labels to be used in the evaluation/validation phrase. If set to `False`, video level labels are used for inference.

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

```
python3 train.py --mode='train' \
    --experiment='yt8m_experiment' \
    --model_dir=$MODEL_DIR \
    --config_file=$CONFIG_FILE
```

In order to run evaluation after each training epoch, set the mode to `train_and_eval`. Paths to both train and validation dataset on Google Cloud are set as train: `input_path=gs://youtube8m-ml/2/frame/train/train*.tfrecord` validation: `input_path=gs://youtube8m-ml/3/frame/validate/validate*.tfrecord` as default.

```
python3 train.py --mode='train_and_eval' \
     --experiment='yt8m_experiment' \
     --model_dir=$MODEL_DIR \
     --config_file=$CONFIG_FILE \
```

Running on evaluation mode loads saved checkpoint from specified path and runs inference task.

```
python3 train.py --mode='eval' \
     --experiment='yt8m_experiment' \
     --model_dir=$MODEL_DIR \
     --config_file=$CONFIG_FILE
```

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

```
train | step:  15190 | training until step 22785...
train | step:  22785 | steps/sec:    0.4 | output:
    {'learning_rate': 0.0049961056,
```

```
     'model_loss': 0.0012011167,
     'total_loss': 0.0013538885,
     'training_loss': 0.0013538885}
```

and the following for evaluation:

```
eval | step:  22785 | running 2172 steps of evaluation...
eval | step:  22785 | eval time:  1663.4 | output:
    {'avg_hit_at_one': 0.5572835238737471,
     'avg_perr': 0.557277077999072,
     'gap': 0.768825760186494,
     'map': 0.19354554465020685,
     'model_loss': 0.0005052475,
     'total_loss': 0.0006564412,
     'validation_loss': 0.0006564412}
```