tensorflow / **models**

Branch: **master** **models** / research / **audioset** /

Create new file | Upload files | Find file | History


qpleple Fixing wrong path — Latest commit d72df8b on 15 Oct

..		
README.md	Fixing wrong path	2 months ago
mel_features.py	Move the research models into a research subfolder (#2430)	3 months ago
vggish_inference_demo.py	Move the research models into a research subfolder (#2430)	3 months ago
vggish_input.py	Move the research models into a research subfolder (#2430)	3 months ago
vggish_params.py	Move the research models into a research subfolder (#2430)	3 months ago
vggish_postprocess.py	Move the research models into a research subfolder (#2430)	3 months ago
vggish_slim.py	Fix broken links in the models repo (#2445)	3 months ago
vggish_smoke_test.py	Move the research models into a research subfolder (#2430)	3 months ago
vggish_train_demo.py	Concatenate instead of add in vggish_train_demo.py.	2 months ago

README.md

Models for AudioSet: A Large Scale Dataset of Audio Events

This repository provides models and supporting code associated with AudioSet, a dataset of over 2 million human-labeled 10-second YouTube video soundtracks, with labels taken from an ontology of more than 600 audio event classes.

AudioSet was released in March 2017 by Google's Sound Understanding team to provide a common large-scale evaluation task for audio event detection as well as a starting point for a comprehensive vocabulary of sound events.

For more details about AudioSet and the various models we have trained, please visit the AudioSet website and read our papers:

- Gemmeke, J. et. al., AudioSet: An ontology and human-labelled dataset for audio events, ICASSP 2017
- Hershey, S. et. al., CNN Architectures for Large-Scale Audio Classification, ICASSP 2017

If you use the pre-trained VGGish model in your published research, we ask that you cite CNN Architectures for Large-Scale Audio Classification. If you use the AudioSet dataset or the released 128-D embeddings of AudioSet segments, please cite AudioSet: An ontology and human-labelled dataset for audio events.

VGGish

The initial AudioSet release included 128-dimensional embeddings of each AudioSet segment produced from a VGG-like audio classification model that was trained on a large YouTube dataset (a preliminary version of what later became YouTube-8M).

We provide a TensorFlow definition of this model, which we call ***VGGish***, as well as supporting code to extract input features for the model from audio waveforms and to post-process the model embedding output into the same format as the released embedding features.

Installation

VGGish depends on the following Python packages:

- `numpy`

- `scipy`
- `resampy`
- `tensorflow`
- `six`

These are all easily installable via, e.g., `pip install numpy` (as in the example command sequence below).

Any reasonably recent version of these packages should work. TensorFlow should be at least version 1.0. We have tested with Python 2.7.6 and 3.4.3 on an Ubuntu-like system with NumPy v1.13.1, SciPy v0.19.1, resampy v0.1.5, TensorFlow v1.2.1, and Six v1.10.0.

VGGish also requires downloading two data files:

- VGGish model checkpoint, in TensorFlow checkpoint format.
- Embedding PCA parameters, in NumPy compressed archive format.

After downloading these files into the same directory as this README, the installation can be tested by running `python vggish_smoke_test.py` which runs a known signal through the model and checks the output.

Here's a sample installation and test session:

```
# You can optionally install and test VGGish within a Python virtualenv, which
# is useful for isolating changes from the rest of your system. For example, you
# may have an existing version of some packages that you do not want to upgrade,
# or you want to try Python 3 instead of Python 2. If you decide to use a
# virtualenv, you can create one by running
#   $ virtualenv vggish   # For Python 2
# or
#   $ python3 -m venv vggish # For Python 3
# and then enter the virtual environment by running
#   $ source vggish/bin/activate  # Assuming you use bash
# Leave the virtual environment at the end of the session by running
#   $ deactivate
```

```
# Within the virtual environment, do not use 'sudo'.

# Upgrade pip first.
$ sudo python -m pip install --upgrade pip

# Install dependences. Resampy needs to be installed after NumPy and SciPy
# are already installed.
$ sudo pip install numpy scipy
$ sudo pip install resampy tensorflow six

# Clone TensorFlow models repo into a 'models' directory.
$ git clone https://github.com/tensorflow/models.git
$ cd models/research/audioset
# Download data files into same directory as code.
$ curl -O https://storage.googleapis.com/audioset/vggish_model.ckpt
$ curl -O https://storage.googleapis.com/audioset/vggish_pca_params.npz

# Installation ready, let's test it.
$ python vggish_smoke_test.py
# If we see "Looks Good To Me", then we're all set.
```

Usage

VGGish can be used in two ways:

- *As a feature extractor*: VGGish converts audio input features into a semantically meaningful, high-level 128-D embedding which can be fed as input to a downstream classification model. The downstream model can be shallower than usual because the VGGish embedding is more semantically compact than raw audio features.

  So, for example, you could train a classifier for 10 of the AudioSet classes by using the released embeddings as features. Then, you could use that trained classifier with any arbitrary audio input by running the audio through the audio feature extractor and VGGish model provided here, passing the resulting embedding features as input to your trained model. `vggish_inference_demo.py` shows how to produce VGGish embeddings from arbitrary audio.

- *As part of a larger model*: Here, we treat VGGish as a "warm start" for the lower layers of a model that takes audio features as input and adds more layers on top of the VGGish embedding. This can be used to fine-tune VGGish (or parts thereof) if you have large datasets that might be very different from the typical YouTube video clip.
  `vggish_train_demo.py` shows how to add layers on top of VGGish and train the whole model.

About the Model

The VGGish code layout is as follows:

- `vggish_slim.py` : Model definition in TensorFlow Slim notation.
- `vggish_params.py` : Hyperparameters.
- `vggish_input.py` : Converter from audio waveform into input examples.
- `mel_features.py` : Audio feature extraction helpers.
- `vggish_postprocess.py` : Embedding postprocessing.
- `vggish_inference_demo.py` : Demo of VGGish in inference mode.
- `vggish_train_demo.py` : Demo of VGGish in training mode.
- `vggish_smoke_test.py` : Simple test of a VGGish installation

Architecture

See `vggish_slim.py` and `vggish_params.py` .

VGGish is a variant of the VGG model, in particular Configuration A with 11 weight layers. Specifically, here are the changes we made:

- The input size was changed to 96x64 for log mel spectrogram audio inputs.
- We drop the last group of convolutional and maxpool layers, so we now have only four groups of convolution/maxpool layers instead of five.

- Instead of a 1000-wide fully connected layer at the end, we use a 128-wide fully connected layer. This acts as a compact embedding layer.

The model definition provided here defines layers up to and including the 128-wide embedding layer.

Input: Audio Features

See `vggish_input.py` and `mel_features.py`.

VGGish was trained with audio features computed as follows:

- All audio is resampled to 16 kHz mono.
- A spectrogram is computed using magnitudes of the Short-Time Fourier Transform with a window size of 25 ms, a window hop of 10 ms, and a periodic Hann window.
- A mel spectrogram is computed by mapping the spectrogram to 64 mel bins covering the range 125-7500 Hz.
- A stabilized log mel spectrogram is computed by applying log(mel-spectrum + 0.01) where the offset is used to avoid taking a logarithm of zero.
- These features are then framed into non-overlapping examples of 0.96 seconds, where each example covers 64 mel bands and 96 frames of 10 ms each.

We provide our own NumPy implementation that produces features that are very similar to those produced by our internal production code. This results in embedding outputs that are closely match the embeddings that we have already released. Note that these embeddings will *not* be bit-for-bit identical to the released embeddings due to small differences between the feature computation code paths, and even between two different installations of VGGish with different underlying libraries and hardware. However, we expect that the embeddings will be equivalent in the context of a downstream classification task.

Output: Embeddings

See `vggish_postprocess.py`.

The released AudioSet embeddings were postprocessed before release by applying a PCA transformation (which performs both PCA and whitening) as well as quantization to 8 bits per embedding element. This was done to be compatible with the YouTube-8M project which has released visual and audio embeddings for millions of YouTube videos in the same PCA/whitened/quantized format.

We provide a Python implementation of the postprocessing which can be applied to batches of embeddings produced by VGGish. `vggish_inference_demo.py` shows how the postprocessor can be run after inference.

If you don't need to use the released embeddings or YouTube-8M, then you could skip postprocessing and use raw embeddings.

Future Work

Below are some of the things we would like to add to this repository. We welcome pull requests for these or other enhancements, but please consider sending an email to the mailing list (see the Contact section) describing what you plan to do before you invest a lot of time, to get feedback from us and the rest of the community.

- An AudioSet classifier trained on top of the VGGish embeddings to predict all the AudioSet labels. This can act as a baseline for audio research using AudioSet.
- Feature extraction implemented within TensorFlow using the upcoming tf.contrib.signal ops.
- A Keras version of the VGGish model definition and checkpoint.
- Jupyter notebook demonstrating audio feature extraction and model performance.

Contact

For general questions about AudioSet and VGGish, please use the audioset-users@googlegroups.com mailing list.

For technical problems with the released model and code, please open an issue on the tensorflow/models issue tracker and ***assign to @plakal and @dpwe***. Please note that because the issue tracker is shared across all models released by Google, we won't be notified about an issue unless you explicitly @-mention us (@plakal and @dpwe) or assign the issue to us.

Credits

Original authors and reviewers of the code in this package include (in alphabetical order):

- DAn Ellis
- Shawn Hershey
- Aren Jansen
- Manoj Plakal