Freesound Audio Tagging 2019

Automatically recognize sounds and apply tags of varying natures

Proposed by: Cojocaru Gabriel-Codrin
Dinu Sergiu Andrei
Luncașu Bogdan Cristian
Racoviță Mădălina-Alina
Vîntur Cristian

Scientific coordinators: PhD Associate Professor Adrian Iftene
PhD Associate Professor Mihaela Elena Breaban

Contents

I. Introduction

I.1. Competition's description

One year ago, **Freesound** and **Google's Machine Perception** hosted an audio tagging competition challenging Kagglers to build a general-purpose auto tagging system. This year they're back and taking the challenge to the next level with multi-label audio tagging, doubled number of audio categories, and a noisier than ever training set. [1]
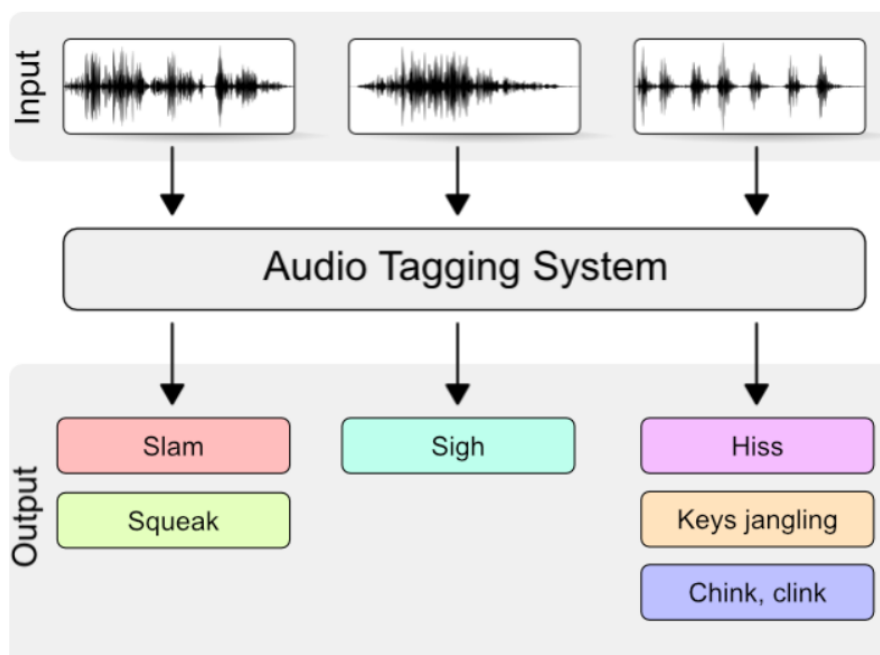
Here's the background: **Some sounds are distinct and instantly recognizable, like a baby's laugh or the strum of a guitar**. Other sounds are difficult to pinpoint. If you close your eyes, could you tell the difference between the sound of a chainsaw and the sound of a blender?

Because of **the vastness of sounds we experience**, *no reliable automatic general-purpose audio tagging systems exist*. A significant amount of manual effort goes into tasks like annotating sound collections and providing captions for non-speech events in audiovisual content.

To tackle this problem, Freesound (an initiative by MTG-UPF that maintains a collaborative database with over 400,000 Creative Commons Licensed sounds) and Google Research's Machine Perception Team (creators of AudioSet, a large-scale dataset of manually annotated audio events with over 500 classes) have teamed up to develop the dataset for this new competition.

To win this competition, Kagglers will develop an **algorithm to tag audio data automatically** using a diverse vocabulary of 80 categories.

If successful, these systems could be used for several applications, ranging from **automatic labelling of sound collections** to **the development of systems that automatically tag video content** or recognize sound events happening in real time.

I.2. Motivation

Current machine learning techniques require large and varied datasets in order to provide good performance and generalization. However, manually labelling a dataset is time-consuming, which limits its size. Websites like Freesound or Flickr host large volumes of user-contributed audio and metadata, and labels can be inferred automatically from the metadata and/or making predictions with pre-trained models. Nevertheless, these automatically inferred labels might include a substantial level of label noise.

The main research question addressed in this competition is **how to adequately exploit a small amount of reliable, manually-labeled data, and a larger quantity of noisy web audio data in a multi-label audio tagging task with a large vocabulary setting**. In addition, since the data comes from different sources, the task encourages domain adaptation approaches to deal with a potential domain mismatch.

I.3. Timeline

- **June 3, 2019 11:59 PM UTC** - Entry deadline. You must accept the competition rules before this date in order to compete.

- **June 3, 2019 11:59 PM UTC** - Team Merger deadline. This is the last day participants may join or merge teams.

- **June 11, 2019 11:59 AM UTC** - Final submission deadline.

II. Dataset description

II.1. Audio Dataset

The dataset used in this challenge is called **FSDKaggle2019**, and it employs audio clips from the following sources:

- Freesound Dataset (**FSD**): a dataset being collected at the **MTG-UPF** based on **Freesound** content organized with the AudioSet Ontology

- The soundtracks of a pool of Flickr videos taken from the **Yahoo Flickr Creative Commons 100M dataset (YFCC)**

The audio data is labeled using a vocabulary of 80 labels from **Google's AudioSet Ontology** [2], covering diverse topics: Guitar and other Musical instruments, Percussion, Water, Digestive, Respiratory sounds, Human voice, Human locomotion, Hands, Human group actions, Insect, Domestic animals, Glass, Liquid, Motor vehicle (road), Mechanisms, Doors, and a variety of Domestic sounds.

II.2. Ground Truth Labels

The ground truth labels are provided at the clip-level, and express the presence of a sound category in the audio clip, hence can be considered weak labels or tags. Audio clips have variable lengths (roughly from 0.3 to 30s, see more details below).

The audio content from **FSD** has been **manually labeled** by humans following a data labeling process using the **Freesound Annotator platform**. Most labels have inter-annotator agreement but not all of them. More details about the data labeling process and the Freesound Annotator can be found in [3].

The **YFCC soundtracks** were labeled using automated heuristics applied to the audio content and metadata of the original Flickr clips. Hence, a substantial amount of label noise can be expected. The label noise can vary widely in amount and type depending on the category, including in- and out-of-vocabulary noises. More information about some of the types of label noise that can be encountered is available in [4].

II.3. Format and License

All clips are provided as uncompressed PCM 16 bit, 44.1 kHz, mono audio files. All clips used in this competition are released under Creative Commons (CC) licenses, some of them requiring attribution to their original authors and some forbidding further commercial reuse. In order to be able to comply with the CC licenses terms, a full list of audio clips with their associated licenses and a reference to the original content (in Freesound or Flickr) will be published at the end of the competition. Until then, the provided audio files can only be used for the sole purpose of participating in the competition.

II.4. Train set

The train set is meant to be for system development. The idea is to limit the supervision provided (i.e., the manually-labeled data), thus promoting approaches to deal with label noise. The train set is composed of two subsets as follows:

II.4.1. Curated subset

The curated subset is a small set of manually-labeled data from FSD.

- **Number of clips/class**: 75 except in a few cases (where there are less)

- **Total number of clips**: 4970

- **Avge number of labels/clip**: 1.2

- **Total duration**: 10.5 hours

The duration of the audio clips ranges from 0.3 to 30s due to the diversity of the sound categories and the preferences of Freesound users when recording/uploading sounds. It can happen that a few of these audio clips present additional acoustic material beyond the provided ground truth label(s).

II.4.2. Noisy subset

The noisy subset is a larger set of noisy web audio data from Flickr videos taken from the **YFCC dataset** [6].

- **Number of clips/class**: 300

- **Total number of clips**: 19815

- **Avge number of labels/clip**: 1.2

- **Total duration**: 80 hours

The duration of the audio clips ranges from 1s to 15s, with the vast majority lasting 15s.

Considering the numbers above, per-class data distribution available for training is, for most of the classes, 300 clips from the noisy subset and 75 clips from the curated subset, which means 80

II.5. Test set

The test set is used for system evaluation and consists of manually-labeled data from FSD. Since most of the train data come from YFCC, some acoustic domain mismatch between the train and test set can be expected. All the acoustic material present in the test set is labeled, except human error, considering the vocabulary of 80 classes used in the competition.

The test set is split into two subsets, for the public and private leaderboards. In this competition, the submission is to be made through Kaggle Kernels. Only the test subset corresponding to the public leaderboard is provided (without ground truth).

II.6. Files

The structure of the data	
train_curated.csv	ground truth labels for the curated subset of the training audio files (see Data Fields below)
train_noisy.csv	ground truth labels for the noisy subset of the training audio files (see Data Fields below)
sample_submission.csv	a sample submission file in the correct format, including the correct sorting of the sound categories; it contains the list of audio files found in the test.zip folder (corresponding to the public leaderboard)
train_curated.zip	a folder containing the audio (.wav) training files of the curated subset
train_noisy.zip	a folder containing the audio (.wav) training files of the noisy subset
test.zip	a folder containing the audio (.wav) test files for the public leaderboard

II.7. Columns

Each row of the train_curated.csv and train_noisy.csv files contains the following information:

- **fname**: the audio file name, eg, 0006ae4e.wav

- **labels**: the audio classification label(s) (ground truth). Note that the number of labels per clip can be one, eg, Bark or more, eg, ”Walk_and_footsteps,Slam”.

II.8. Dataset problems

Detected corrupted files in the curated train set The following 5 audio files in the curated train set have a wrong label, due to a bug in the file renaming process: **f76181c4.wav**, **77b925c2.wav**, **6a1f682a.wav**, **c7db12aa.wav**, **7752cc8a.wav**

The audio file **1d44b0bd.wav** in the curated train set was found to be corrupted (contains no signal) due to an error in format conversion.

IV. Sound classification predictive models

IV.1. Models brief description from different Kaggle approaches

Kaggle [9] is an online community of data scientists and machine learners. Since Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges, it **can be used also as a source of inspiration when solving different tasks machine learning related**. We extracted some of the predictive models that we have found in the public kernels that were attached to the subject competition.

IV.1.1. Bidirectional LSTM for audio labeling with Keras

Long short-term memory (LSTM) is an artificial **recurrent neural network (RNN)** architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as **unsegmented, connected handwriting recognition or speech recognition**. **LSTM** in its core, *preserves information from inputs that has already passed through it using the hidden state.*

Unidirectional LSTM only **preserves information of the past** because the only inputs it has seen are from the past. Using **bidirectional** will **run your inputs in two ways, one from past to future and one from future to past** and what differs this approach from unidirectional is that **in the LSTM that runs backwards you preserve information from the future** and using the two hidden states combined you are able in any point in time to preserve information from both past and future.

What they are suited for is a very complicated question but BiLSTMs show very good results as they can understand context better, but this can be explain through a simple example. Lets

say we try to predict the next word in a sentence, on a high level what a unidirectional LSTM will see is:

```
The boys went to ....
```

And will try to predict the next word only by this context, with bidirectional LSTM you will be able to see information further down the road for example

```
# Forward LSTM:

The boys went to ...

# Backward LSTM:

... and then they got out of the pool
```

You can see that using the information from the future it could be easier for the network to understand what the next word is.
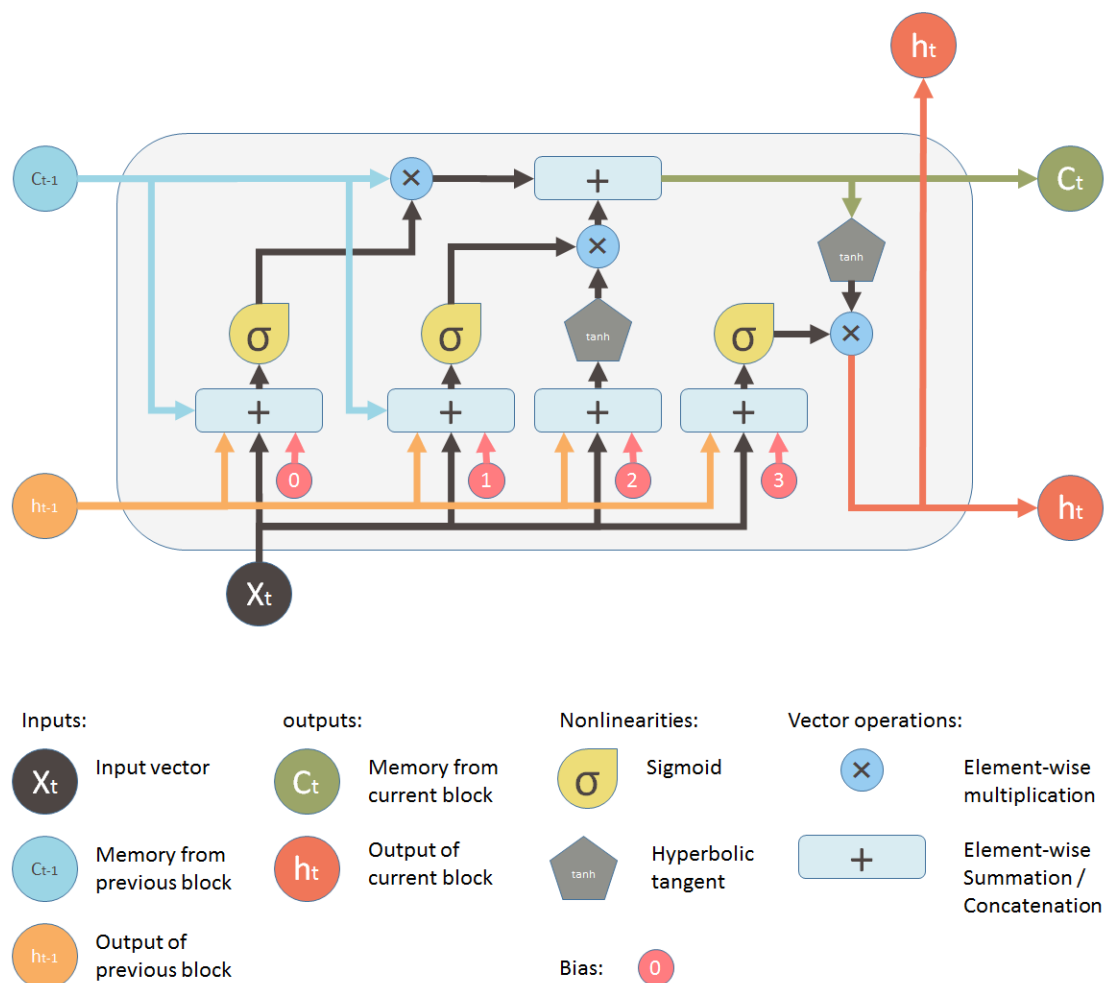

Fig. IV.1.1.a: *Overview LSTM*

In a public Kaggle kernel [8] it was used the following architecture for the **Freesound classification problem** inspired from an article named *Convolutional RNN: an Enhanced Model for Extracting Features from Sequential Data* [11].

```python
# Neural network model
input_shape = (636,128)
optimizer = Adam(0.005, beta_1=0.1, beta_2=0.001, amsgrad=True)
n_classes = 80

model = Sequential()
model.add(Bidirectional(CuDNNLSTM(256, return_sequences=True),
          input_shape=input_shape))
model.add(Attention(636))
model.add(Dropout(0.2))
model.add(Dense(400))
model.add(ELU())
model.add(Dropout(0.2))
model.add(Dense(n_classes, activation='softmax'))
model.compile(loss='categorical_crossentropy',
              optimizer=optimizer,
              metrics=['acc'])
```

Another likely model that develops a biird sound classification problem and uses bidirectional LSTM neural networks had the following architecture:
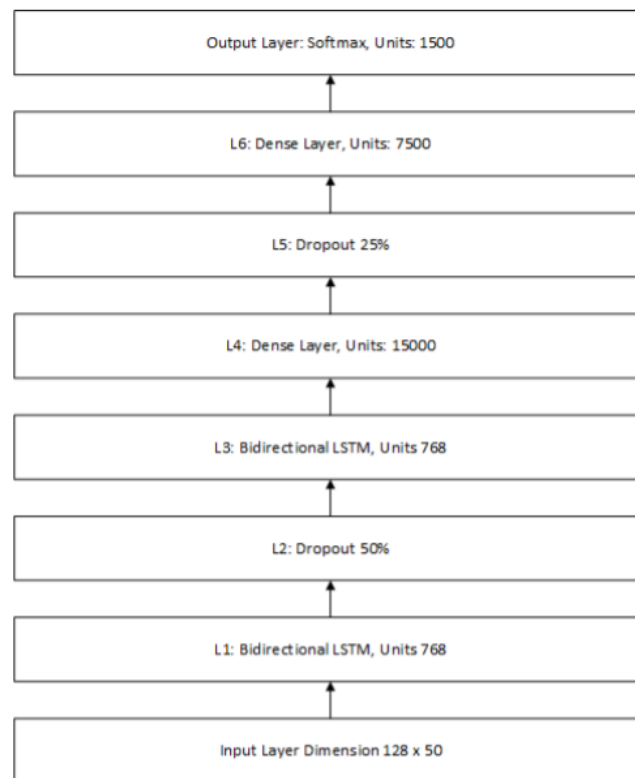

Fig. IV.1.1: *Architecture bidirectional LSTM. [10]*

IV.2. Evaluation metric

The task consists of **predicting the audio labels (tags) for every test clip**. Some test clips bear one label while others bear several labels. The predictions are to be done at the clip level, i.e., no start/end timestamps for the sound events are required.

The primary competition metric will be **label-weighted label-ranking average precision** [7](lwlrap, pronounced "Lol wrap"). This **measures the average precision of retrieving a ranked list of relevant labels** for each test clip (i.e., the system ranks all the available labels, then the precisions of the ranked lists down to each true label are averaged). This is a generalization of the mean reciprocal rank measure for the case where there can be multiple true labels per test item. The novel "label-weighted" part means that the overall score is the average over all the labels in the test set, where each label receives equal weight (by contrast, plain lrap gives each test item equal weight, thereby discounting the contribution of individual labels when they appear on the same item as multiple other labels).

The formula for label-ranking average precision (LRAP) is as follows:

$$LRAP(y, \widehat{f}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} \frac{1}{\|y_i\|_0} \sum_{j:y_{ij}=1} \frac{|\mathcal{L}_{ij}|}{rank_{ij}} \tag{1}$$

```python
def calculate_overall_lwlrap_sklearn(truth, scores):
    """Calculate the overall lwlrap using sklearn.metrics.lrap."""
    # sklearn doesn't correctly apply weighting to samples with no labels
    # so just skip them.
    sample_weight = np.sum(truth > 0, axis=1)
    nonzero_weight_sample_indices = np.flatnonzero(sample_weight > 0)
    overall_lwlrap = label_ranking_average_precision_score(
        truth[nonzero_weight_sample_indices, :] > 0,
        scores[nonzero_weight_sample_indices, :],
        sample_weight=sample_weight[nonzero_weight_sample_indices])
    return overall_lwlrap
```

Bibliografie

[1] **Competition's overview from Kaggle platform**:
https://www.kaggle.com/c/freesound-audio-tagging-2019/overview

[2] **Google's AudioSet Ontology**: *https://research.google.com/audioset////////ontology/index.html*

[3] Eduardo Fonseca, Jordi Pons, Xavier Favory, Frederic Font, Dmitry Bogdanov, Andres Ferraro, Sergio Oramas, Alastair Porter, and Xavier Serra. **"Freesound Datasets: A Platform for the Creation of Open Audio Datasets."** In Proceedings of the International Conference on Music Information Retrieval, 2017. [PDF]

[4] Eduardo Fonseca, Manoj Plakal, Daniel P. W. Ellis, Frederic Font, Xavier Favory, and Xavier Serra. **"Learning Sound Event Classifiers from Web Audio with Noisy Labels."** In Proceedings of the International Conference on Acoustics, Speech and Signal Processing, 2019. [PDF]

[5] Frederic Font, Gerard Roma, and Xavier Serra. **"Freesound technical demo."** Proceedings of the 21st ACM international conference on Multimedia, 2013. https://freesound.org

[6] Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li, YFCC100M: **The New Data in Multimedia Research**, Commun. ACM, 59(2):64–73, January 2016

[7] **LRAP Python**:
https://scikit-learn.org/stable/modules/model_evaluation.html#label-ranking-average-precision

[8] **Bidirectional LSTM for audio labeling with Keras**:
https://www.kaggle.com/carlolepelaars/bidirectional-lstm-for-audio-labeling-with-keras#Birectional-LSTM-model-for-audio-labeling-with-Keras

[9] **About Kaggle**: *https://en.wikipedia.org/wiki/Kaggle*

[10] **Bird sound classification using a bidirectional LSTM**, Lukas Muller and Mario Marti: *http://ceur-ws.org/Vol-2125/paper_134.pdf*

[11] **Convolutional RNN: an Enhanced Model for Extracting Features from Sequential Data**: *https://arxiv.org/pdf/1602.05875v3.pdf*