Machine Learning Engineer Nanodegree, Capstone
General-Purpose Audio Tagging

Salih Güngör
Jun 11th, 2018

Domain Background

There is no reliable application to automatically tag nested voices. Manual effort is required for the marking process. With this work, we want to make a deep learning application where sounds can be separated and labeled. In this way, unwanted sounds can be extracted in a sound file. It can also be used in music suggestion systems. For example, you can perceive a music with clarinet sound and you can suggest another music that has a clarinet in it. You can access the article describing the problem and its solution. The article tells the story of a human being. However, the general logic is the same. And another text extracts some features from the audio signal.

Problem Statement

It is a problem of today's artificial intelligence to recognize many different sources of sound in a pic, like recognizing many objects in a photo. Each sound source has its own signature. We aim to capture this pattern of permission and catch what source of sound is taken from it. We will then try to identify these tracks with mixed sound. I will use a previously marked dataset for this. I want to find out what audio sources are on which sound sources. For this, I chose a kaggle project with a classification problem.

The interference will initially be an audio signal, but I will extract the MFCC properties from this audio signal (pre-processing data). So I do not use RAW data for my CNN model.

The output will be 41 categories, so I have 41 outputs, but the input will actually try with 50 inputs at first depending on the performance (Extracted MFCC algorithm). And it will likely change due to hardware performance and accuracy. I will test these changes.

Dataset and Inputs

You can find the used data on the kaggle page. (The data set is taken from kaggle)

I will extract some features using the Mel Frequency Cepstral Coefficient (MFCC) algorithm. The anchor basically divides into fragments and summarizes the data framed. For more details, see Algorithm, Application.

File Descriptions

- train.csv - ground truth labels and other information relating to the training audio
- audio_train.zip - a folder containing the audio (.wav) training files
- audio_test.zip - a folder containing the audio (.wav) test files
- sample_submission.csv - a sample submission file in the correct format; contains the list of files found in the audio_test.zip folder

Data Fields
Each row of the train.csv file contains the following information:

- fname: the file name
- label: the audio classification label (ground truth)
- manually_verified: Boolean (1 or 0) flag to indicate whether or not that annotation has been manually verified; see below for additional background

About This Dataset (The data set is taken from kaggle)
Dataset Kaggle 2018 (or FSDKaggle2018 for short) is an audio dataset containing 18,873 audio files annotated with labels from Google's AudioSet Ontology.

All audio samples in this dataset are gathered from Freesound and are provided here as uncompressed PCM 16 bit, 44.1 kHz, mono audio files. The ground truth data provided in this dataset has been obtained after a data labeling process which is described in the Data labeling process section below. FSDKaggle2018 sounds are unequally distributed in the following 41 categories of the AudioSet.
Ontology:

> "Acoustic_guitar", "Applause", "Bark", "Bass_drum", "Burping_or_eructation", "Bus", "Cello", "Chime", "Clarinet", "Computer_keyboard", "Cough", "Cowbell", "Double_bass", "Drawer_open_or_close", "Electric_piano", "Fart", "Finger_snapping", "Fireworks", "Flute", "Glockenspiel", "Gong", "Gunshot_or_gunfire", "Harmonica", "Hi-hat", "Keys_jangling", "Knock", "Laughter", "Meow", "Microwave_oven", "Oboe", "Saxophone", "Scissors", "Shatter", "Snare_drum", "Squeak", "Tambourine", "Tearing", "Telephone", "Trumpet", "Violin_or_fiddle", "Writing"

Here are some other relevant characteristics of FSDKaggle2018:

- The dataset is split into a train set and a test set.
- The train set is meant to be for system development and includes ~9.5k samples unequally distributed among 41 categories. The minimum number of audio samples per category in the train set is 94, and the maximum 300. The duration of the audio samples ranges from 300ms to 30s due to the diversity of the sound categories and the preferences of Freesound users when recording sounds.
- MFCC algorithm that I use will normalize the duration of inputs. So I do not need padding the data.
- Out of the ~9.5k samples from the train set, ~3.7k have manually-verified ground truth annotations and ~5.8k have non-verified annotations. The non-verified annotations of the train set have a quality estimate of at least 65-70% in each category. Checkout the Data labeling process section below for more information about this aspect.
- Non-verified annotations in the train set are properly flagged in train.csv so that participants can opt to use this information during the development of their systems.
- The test set is composed of ~1.6k samples with manually-verified annotations and with a similar category distribution than that of the train set. The test set is complemented with ~7.8k padding sounds which are not used for scoring the systems.

- All audio samples in this dataset have a single label (i.e. are only annotated with one label). Checkout the Data labeling process section below for more information about this aspect.

Solution Statement

I will try to detect audio sources in pre-labeled audio files using CNN. I will teach CNN model data with audio features in tagged audio files. After teaching the model, I will test these learned weights in the test set divided into two parts, these predetermined data, training and test set. I will compare predicted classes and actual labeled test data. That's why I'm going to measure accuracy performance.

Benchmark Model

Random selection: I will estimate the probability that a voice belongs to any class of total labeled classes.

CNN: I will give the CNN model train data after reducing RAW output using Mel Frequency Cepstral Coefficient (MFCC) methods. So, I'll use my resources efficiently, giving less data and more focused features to the algorithm.

The MFCC mimics the pitch and logarithmic sense of the human auditory system and excludes the fundamental frequency and harmonics, thereby excluding the speaker-connected features. The underlying math is quite complex and I'll be sure to check the link for more details.

Once the attribute has been extracted, the CNN model will feed using the empirical properties and try to train the CNN model's weights. After that, I will compare the estimates and the original test data to determine how accurate my model is.

Evaluation Metrics

My evaluation metrics will be F1 score. I used it in our previous projects. A good advantage of F1 score: If there are crooked classes like this one, for example there are 41 different labels, but some tags are in other tags. If I guess a cluster that is smaller than it in a not too big main cluster, I cannot say that I made it with the right guess. However, if the entire recording is accurate, this is an excellent prediction. If I make a mistaken guess in a small cluster, it may not reduce the accuracy. Because the number of values in the cluster is small and it will be difficult to correct.

In such situations, precision and recall become very useful. These two measurements can be combined to obtain the F1 score; this is the weighted average of the sensitivity and recall scores (harmonic mean). This score can range from 0 to 1, with 1 being the best possible F1 score (we get the harmonic average for the ratios).

Project Design

Project: Freesound audio tagging description

Goal : Solving a Multi-label classification problem
Objectives : High Accuracy
Constraints : Relatively small data, Low Hardware

1. Preprocess in Data
    a. I will use data from Kaggle
    b. Feature Extraction by using MFCC for some hardware constraints
    c. Split the data test and train
2. Data Exploration
    a. Observing the labels
    b. Observing features that extracted
    c. Observing the statistics about the sounds
3. Define the accuracy functions
    a. F score
4. Define random choice predictor (for benchmarking)
5. Develop a classifier CNN model
    a. Developing a multi-layer CNN model which has all labels as an output and all features as an input
6. Tune the CNN model
    a. Training parameters
        i. Training length (number of epochs)
        ii. Solver type (Algorithm for learning)
        iii. Learning rate (how fast to learn)
        iv. Weight decay (prevents the model being dominated by a few “neurons”)
        v. Momentum (When the next step is calculated, the previous learning step is considered.)
    b. Neural network architecture
        i. Number of layers
        ii. Layer types (convolutional , fully-connected , or pooling)
        iii. Layer parameters
    c. Preprocessing parameters (see the Data Preprocessing tab)