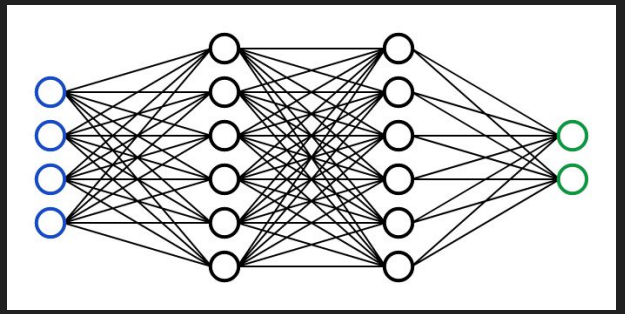
Audio Classification

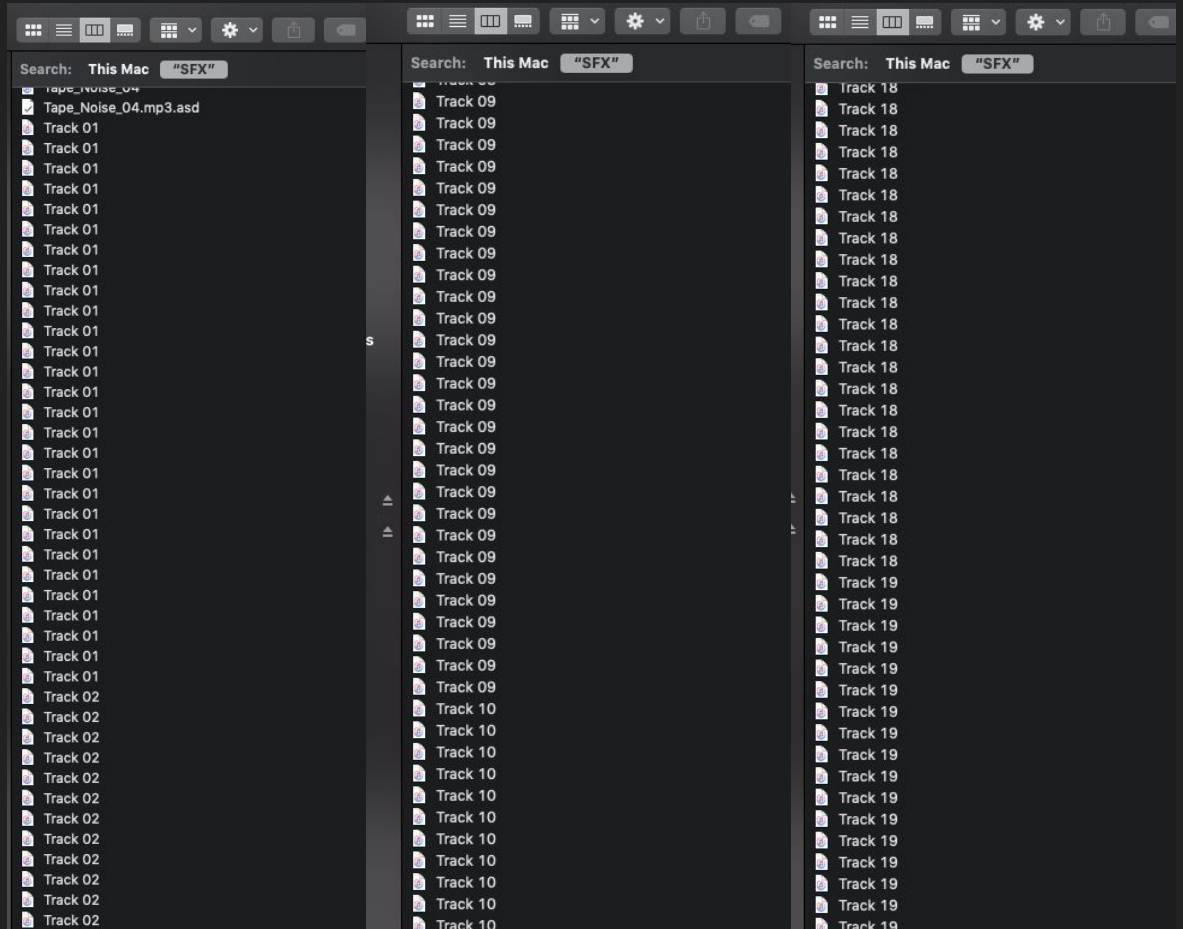
With Neural Networks



Why is Audio Classification Useful?

Organizing Sound FX Libraries

SFX Libraries can be enormous and very poorly labelled.



My personal SFX library is 413.52 GB and would not be considered a large SFX library

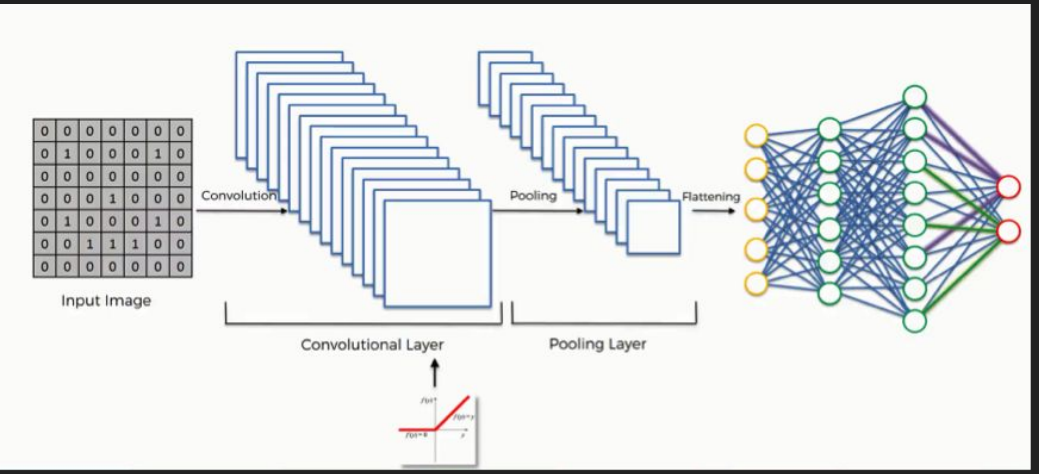
These are all titled Track 18

These have no meta data tags

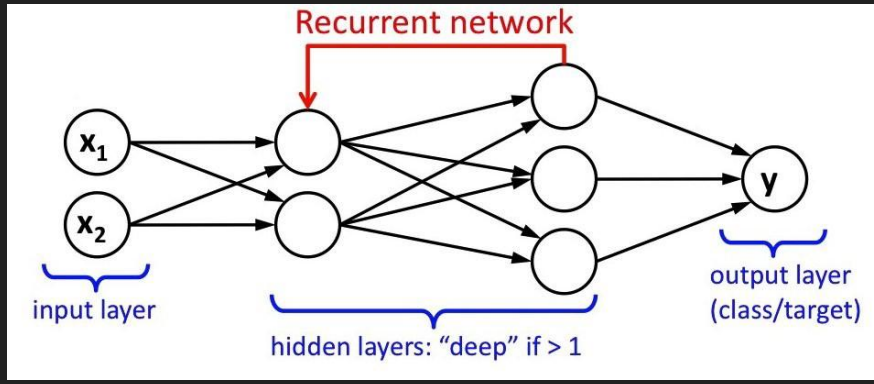
These are just a few examples of the audio files that have been sitting in my SFX library without anyway for them to come up in a search, simply wasting space.

Neural Networks for Audio Classification

Convolutional Neural Network



Recurrent Neural Network



Trained on 9,457 audio files corresponding to 41 classes

```
['Acoustic_guitar', 'Applause', 'Bark', 'Bass_drum', 'Burping_or_eructation', 'Bus', 'Cello', 'Chime', 'Clarinet', 'C
omputer_keyboard', 'Cough', 'Cowbell', 'Double_bass', 'Drawer_open_or_close', 'Electric_piano', 'Fart', 'Finger_snapp
ing', 'Fireworks', 'Flute', 'Glockenspiel', 'Gong', 'Gunshot_or_gunfire', 'Harmonica', 'Hi-hat', 'Keys_jangling', 'Kn
ock', 'Laughter', 'Meow', 'Microwave_oven', 'Oboe', 'Saxophone', 'Scissors', 'Shatter', 'Snare_drum', 'Squeak', 'Tamb
ourine', 'Tearing', 'Telephone', 'Trumpet', 'Violin_or_fiddle', 'Writing']
```

```
training convolutional model
Train on 1009704 samples, validate on 112190 sam
Epoch 1/1
1009704/1009704 [==============================]
val_acc: 0.6766
```

```
training recurrent model
Train on 1009704 samples, validate on 112190 sam
Epoch 1/1
1009704/1009704 [==============================]
val_acc: 0.6754
```

Cello

Predictions

CNN →

```
Meow
Laughter
Applause
```

RNN →

```
Applause
Cello
Violin_or_fiddle
```

Laughter

Predictions

CNN →

```
Laughter
Cough
Telephone
```

RNN →

```
Laughter
Cough
Writing
```

Bus

Predictions

CNN →

```
Gong
Scissors
Computer_keyboard
```

RNN →

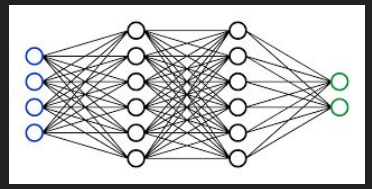
```
Drawer_open_or_close
Scissors
Writing
```

IBM Developer Model Asset Exchange: Audio Classifier

This repository contains code to instantiate and deploy an audio classification model. This model recognizes a signed 16-bit PCM wav file as an input, generates embeddings, applies PCA transformation/quantization, uses the embeddings as an input to a multi-attention classifier and outputs top 5 class predictions and probabilities as output. The model currently supports 527 classes which are part of the Audioset Ontology. The classes and the label_ids can be found in class_labels_indices.csv. The model was trained on AudioSet as described in the paper 'Multi-level Attention Model for Weakly Supervised Audio Classification' by Yu et al.

Trained on the google audio dataset:

2,084,320 YouTube Videos with 527 classes



Classification	Prob
Vehicle	.63
Whir	.47
Engine	.38
Rumble	.17
Earthquake	.06

Threshold = 0.25

Sound FX Library

Model Running on Docker Image

Classifications

Metadata Tags

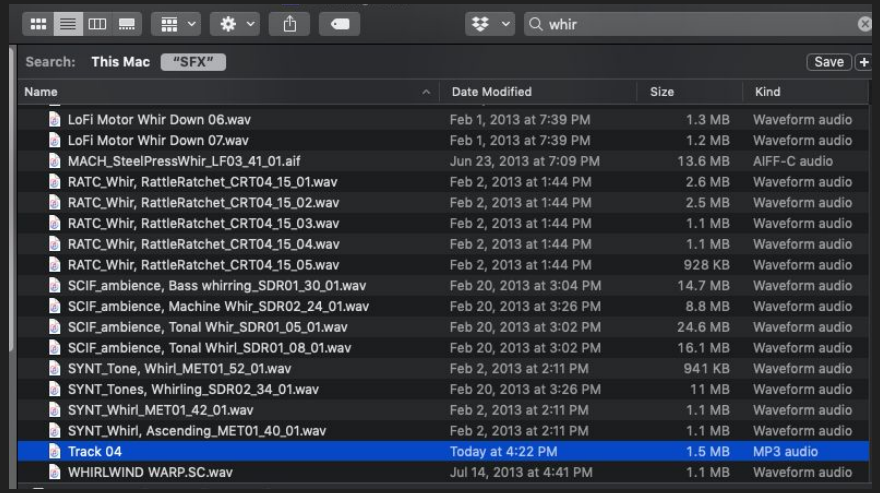
```
sending Track 24.wav to API
Adding these tags: Animal, Dog, Domestic animals, pets, Bow-wow, Bark
```

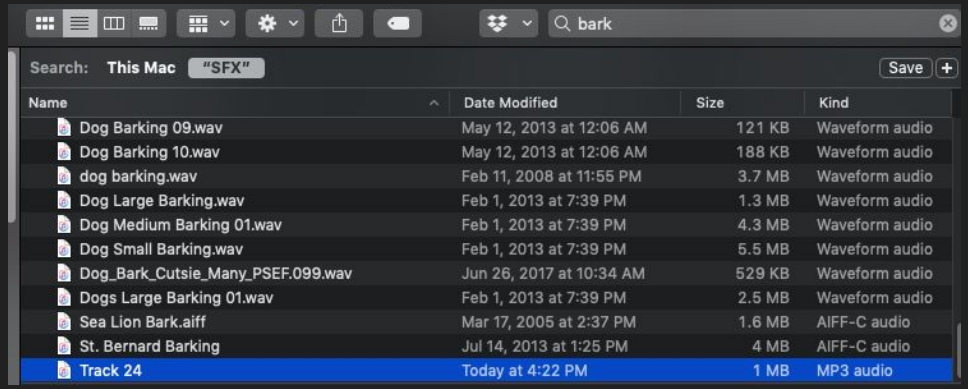
```
sending Track 35.wav to API
Adding these tags: Trumpet, Brass instrument, Music, Musical instrument
```

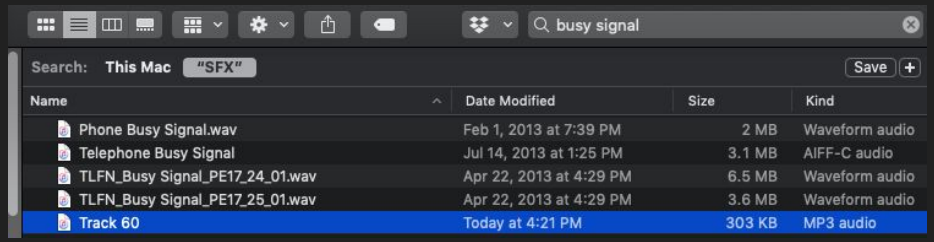
```
sending Track 60.wav to API
Adding these tags: Telephone, Busy signal, Music
```

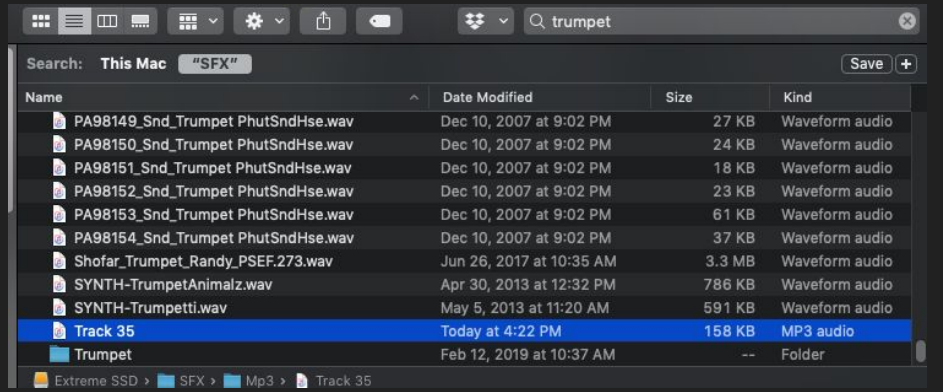
```
sending Track 04 4.wav to API
Adding these tags: Vehicle, Whir
```

Sample Searches









Dylan Case

dylancase@gmail.com
310-701-5537

dylancase Changed Title — Latest commit 90cefb3 1 minute ago

ClassifyAndTag.py	Added Script	20 hours ago
README.md	Changed Title	1 minute ago
models.py	Added Models	20 hours ago

README.md

Audio Classification and Metadata Tags

My Final Capstone Project from the Galvanize Data Science Immersive

Goals

Having been a recording engineer prior to enrolling in Galvanize's Data Science Immersive, I really wanted to use my new found data science skills to solve a problem I often encountered in the Post Production Audio world, poorly organized and labelled sound effect libraries. I set out to solve that problem by creating an audio classification model to help organize and tag Sound FX directories/libraries.

GitHub

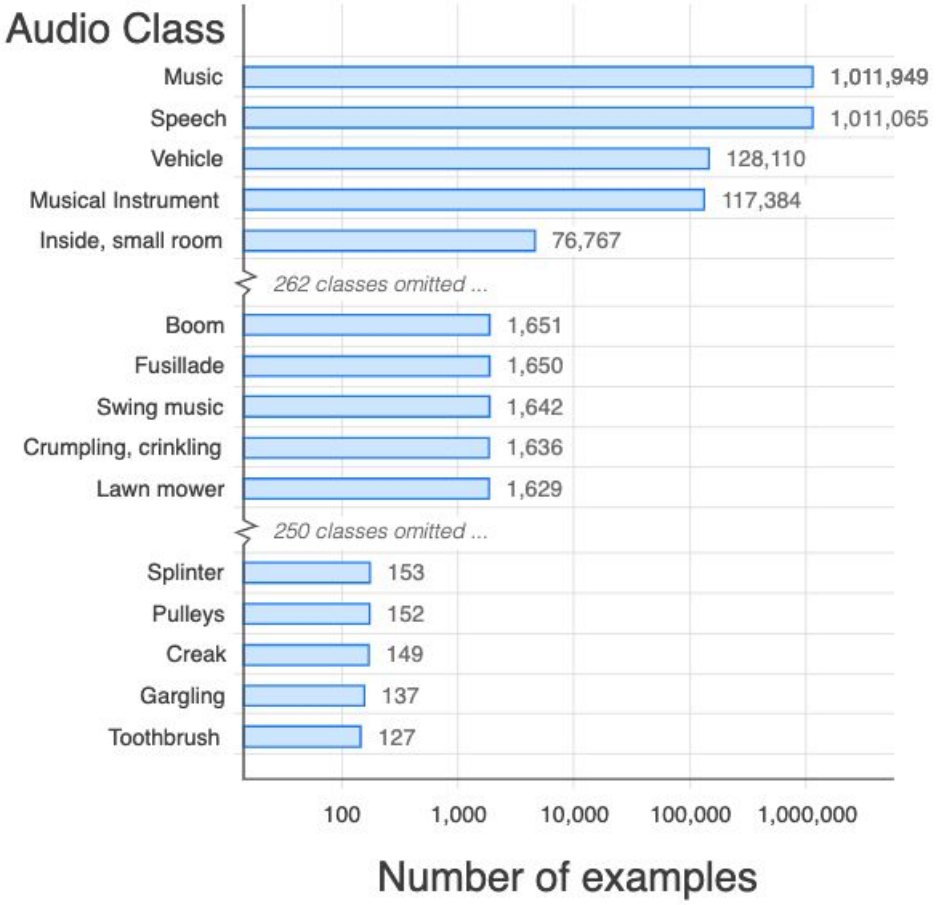
github.com/dylancase/

linkedin.com/in/dylancase/

Next Steps

- Train on labelled Sound FX Library
- Transfer Learn
- Add synonym tags (for example, when adding ‘clapping’ to metadata tags, also add ‘applause’, ‘cheering’, and/or ‘ovation’)

End



Music	100%	1,011,305
Speech	100%	1,010,480
Vehicle	100%	128,051
Musical instrument	100%	117,343
Plucked string instrument	100%	44,565
Singing	100%	42,493
Car	100%	41,554
Animal	100%	40,758
Outside, rural or natural	100%	35,731
Violin, fiddle	100%	28,125
Bird	100%	26,894
Drum	100%	20,246
Engine	100%	16,245
Narration, monologue	100%	15,590
Drum kit	100%	15,169
Acoustic guitar	100%	14,568
Dog	100%	13,705
Child speech, kid speaking	100%	11,816
Bass drum	100%	9,292
Rail transport	100%	9,052
Motor vehicle (road)	100%	9,044
Water	100%	8,994
Female speech, woman speaking	100%	8,513
Siren	100%	8,498
Railroad car, train wagon	100%	8,361

Cello


```
{
  "label_id": "/m/07pp_mv",
  "label": "Alarm",
  "probability": 0.41957834362983704
},
{
  "label_id": "/m/0c3f7m",
  "label": "Fire alarm",
  "probability": 0.3206864893436432
},
{
  "label_id": "/m/030rvx",
  "label": "Buzzer",
  "probability": 0.2251153439283371
},
```

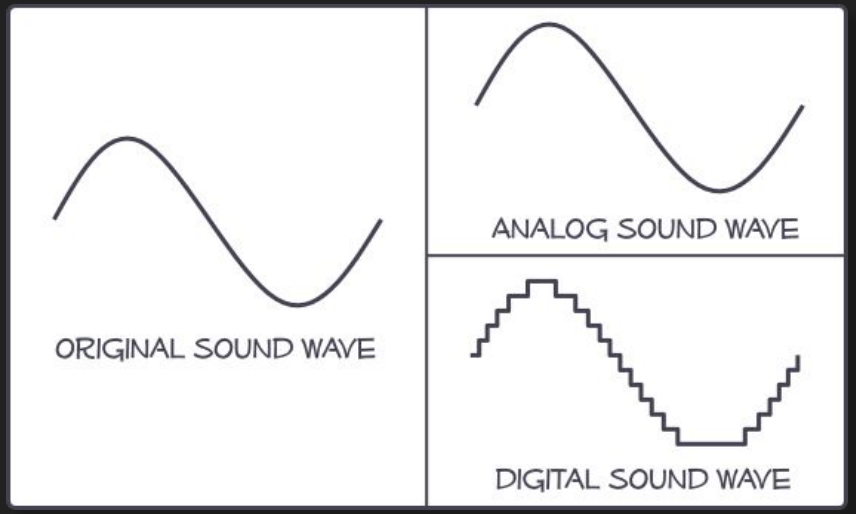

Laughter

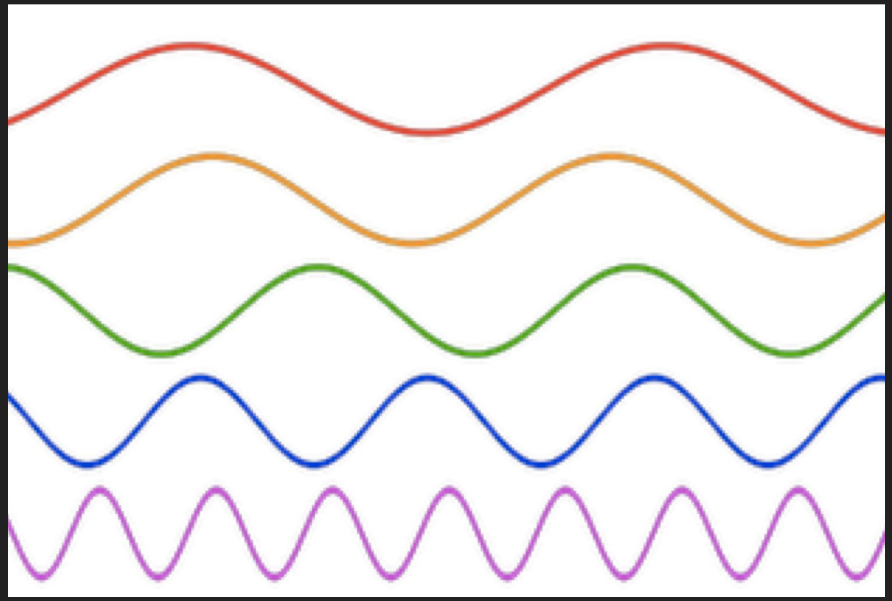

```
{
  "label_id": "/m/07qw_06",
  "label": "Wail, moan",
  "probability": 0.6524030566215515
},
{
  "label_id": "/m/02cz_7",
  "label": "Beatboxing",
  "probability": 0.387509822845459
},
{
  "label_id": "/t/dd00001",
  "label": "Baby laughter",
  "probability": 0.2391974925994873
```


Bus

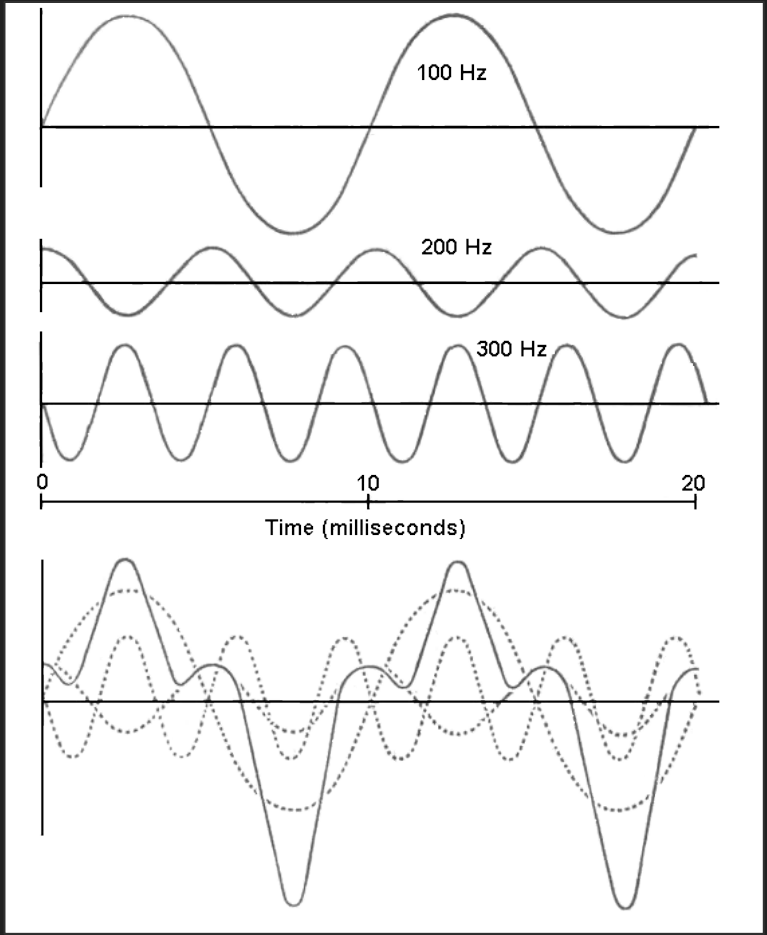

```
{
  "label_id": "/m/01m4t",
  "label": "Printer",
  "probability": 0.13082295656204224
},
{
  "label_id": "/m/02mk9",
  "label": "Engine",
  "probability": 0.09326490014791489
},
{
  "label_id": "/m/07yv9",
  "label": "Vehicle",
  "probability": 0.07092247158288956
```

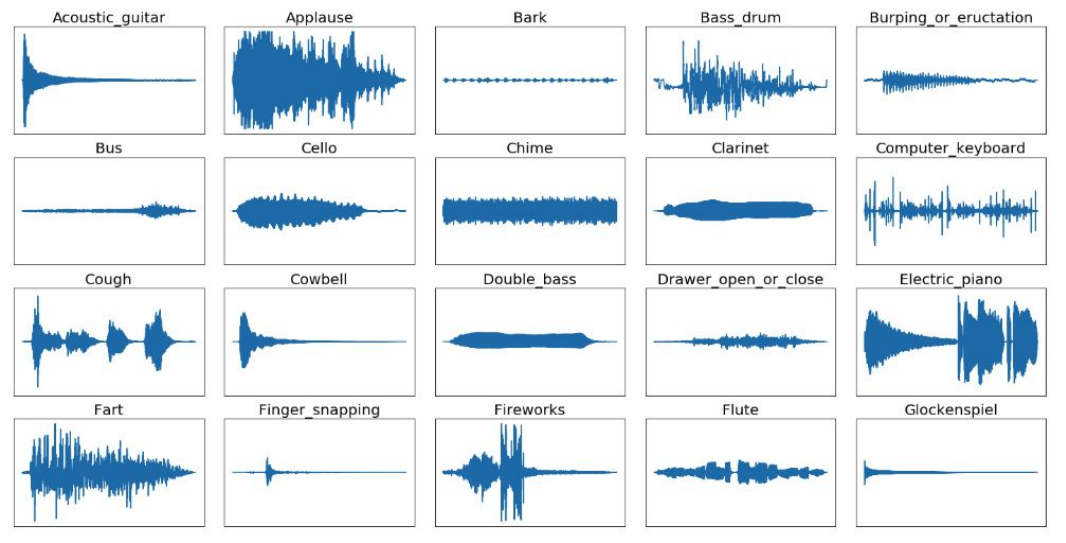
Audio Data





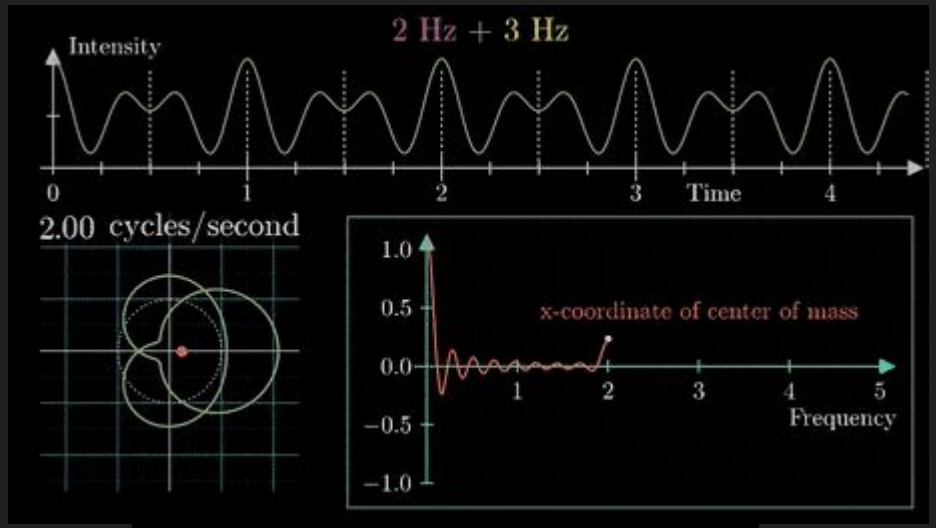
Complex Audio Waveforms

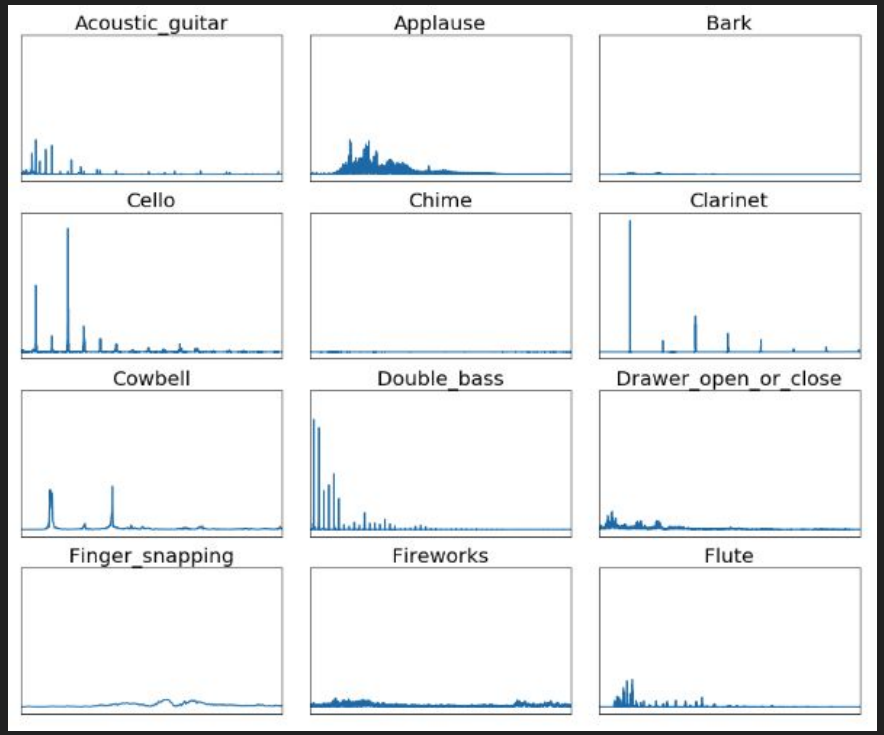




Fourier Transform

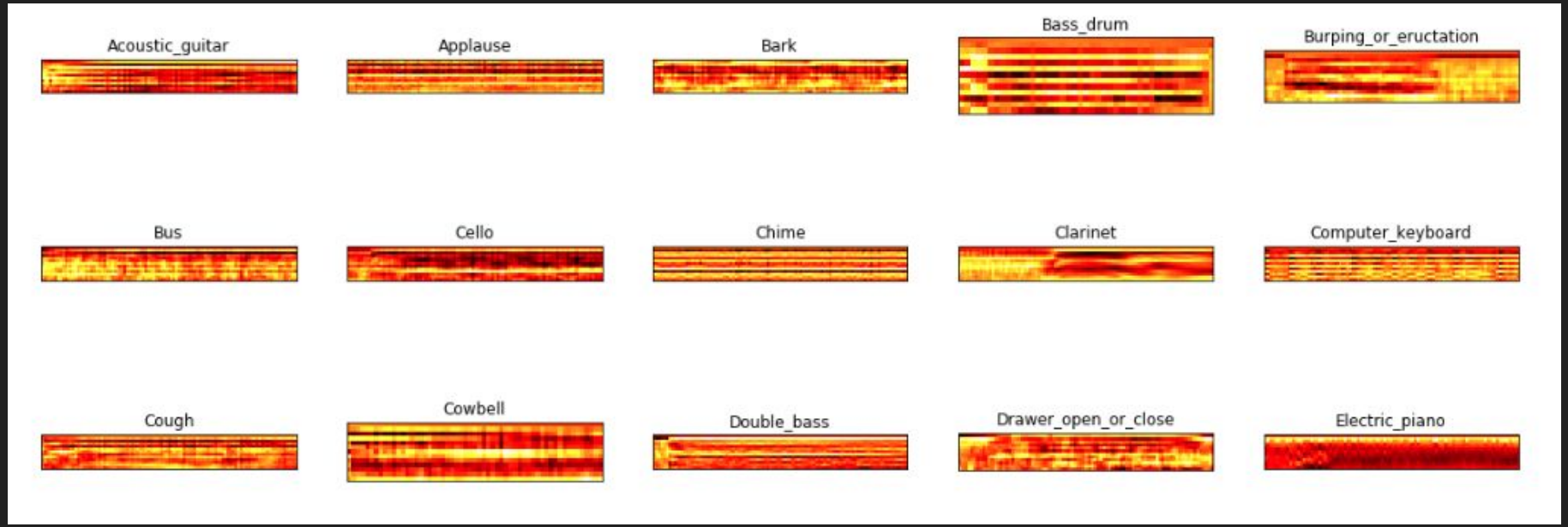
Extracts frequency data from sound data





Mel Cepstral Coefficients

Compensates for the logarithmic nature of loudness as well as binning frequency content so that small differences in pitch do not constitute entirely different sounds



Remember those files labelled Track 18

```
sending Track 18 copy.wav to API
Adding these tags: Clapping
```

```
sending Track 18.wav to API
Adding these tags: Music, Cacophony
```

```
sending Track 18 copy 2.wav to API
Adding these tags: Vehicle, Car
```

```
sending Track 18 copy 3.wav to API
Adding these tags: Typewriter
```

```python
def get_tags_and_add(directory, threshold = .25, verbose=0):
    for file in os.listdir(directory):
        if verbose > 0:
            print(f'sending {file} to API')
        sub = subprocess.check_output([f'curl -F "audio=@{directory}/{file};type=audio/wav" -XPOST http://localhost:5(
        response_dict = eval(sub)
        tags_to_add = []
        try:
            audio = ID3(f'Mp3/{file[0:-3]}mp3')
        except:
            audio = MP3(f'Mp3/{file[0:-3]}mp3')
            audio.add_tags()
            audio.save()
            audio = ID3(f'Mp3/{file[0:-3]}mp3')
        for pred in response_dict['predictions']:
            if pred['probability']>threshold:
                tags_to_add.append(pred['label'])
        if verbose > 0:
            print('Adding these tags: ' + ', '.join(tags_to_add))
            print('------------------------------')
        existing = audio.get('TCON')
        if existing:
            audio.add(TCON(text=' '.join(set(audio.get('TCON').text[0].split() + existing.text))))
        else:
            audio.add(TCON(text=' '.join(tags_to_add)))
        audio.add(TCON(text=' '.join(tags_to_add)))
        audio.save()
```

How much is it really worth to label audio files?

What is Soundminer?

DESCRIPTION

Advanced digital audio asset manager for Apple OSX 10.7 to 10.13. The v4.5 Standard Edition takes the core features of v4.5pro and distills them into a less expensive package designed for sound editors. Whether you are working with sound effects or music, commercial or original files, SoundminerV4.5 allows you to have advanced control over your organizational structure, searching, tracking and transferring. V4.5 is much more than a search engine and it builds on advances made in Version 3 and 4. It supports 4.5's expanded field structure designed with production music libraries in mind, new waveform editor, easy to use new Spotting Panel (Projects and transfers), easy import of Itunes, 64 bit Rewire and current versions of Pro Tools, simple metadata editing, multi-channel waveform display and a lot more.

V4.5 Standard

$599.00 USD

V5Pro

$899.00 USD

Deploy from Docker Hub

To run the docker image, which automatically starts the model serving API, run:

```
$ docker run -it -p 5000:5000 codait/max-audio-classifier
```