Choir detection

Given an mp3 song detect audio fragments with a choir and generate a new mp3 file containing only these fragments.

Repository link

https://github.com/olegzinkevich/choir_audio_detection

Model training example in google colab

https://colab.research.google.com/drive/1bC-1DDdleqrSWw0twLYGM1FFCALHP0_v?usp=sharing

Model inference example in google colab

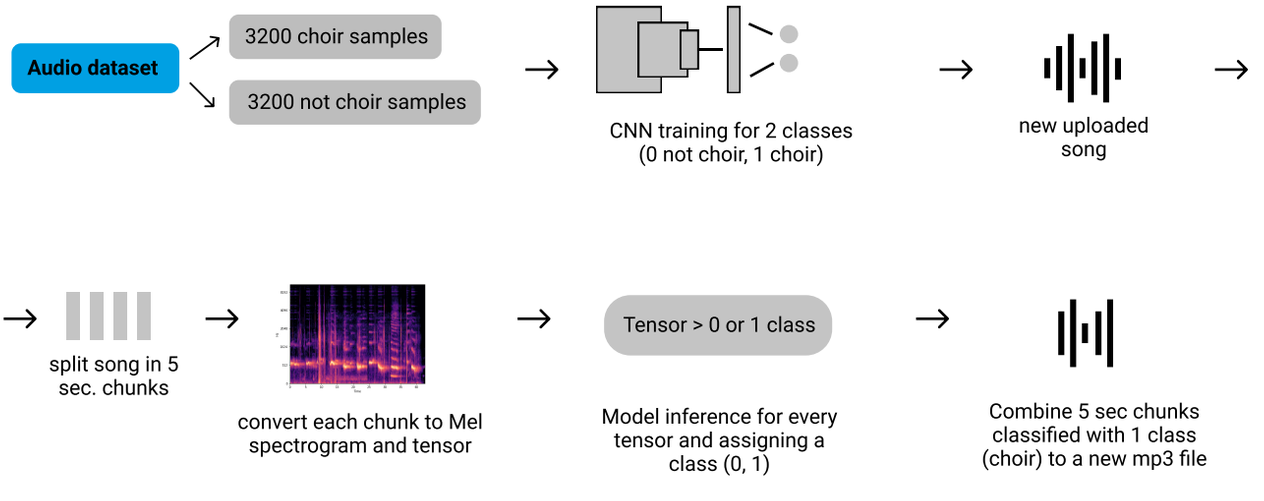
https://colab.research.google.com/drive/14Eb9H83R-b6MRzp62pnkvDRlL-BXmGr2?usp=sharing

Contents

Audio analysis hypothesis

We divide a given mp3 into 5 seconds audio chunks, normalize each chunk, generate a Mel Spectrogram for each audio segment. Then apply classification of every spectrogram tensor with a CNN. After this step the chunks which were classified with the 1 class (choir) are merged into the final mp3 file containing only choir samples.

General artchitecture



Unfortunately, existing available datasets with choir samples can't be used to properly detect choir:
1. https://github.com/HuiZhangDB/PMEmo - contains 794 choir mp3 samples. Though after analysis it turned out that most of the songs contain not a choir but a voice with a reverb, which was classified as the choir.
2. https://sites.google.com/site/unvoicedsoundseparation/mir-1k - contains only a few choir examples, so it can't be used for the lack of consistency.

Choir dataset generation and augmentation

So the decision was made to create a dataset from scratch. Audio Databanks for choral music, containing classical choir samples, Gospel samples without any background instruments were taken (these databanks were downloaded via torrents, so the final dataset can be applied only for research).

For not-choir sound examples GTZAN dataset was used (1000 audio tracks of different genres without any choir) (https://www.tensorflow.org/datasets/catalog/gtzan).

Audio normalization

- Then all audio samples were normalized (see the created **processing python class**: https://github.com/olegzinkevich/choir_audio_detection/tree/main/utils) - converted to mono, with a mean sample rate of 22050hz.

Audio augmentation

- After that, all normalized samples were chunked (divided into the 5-second parts) and saved as separate audio files.

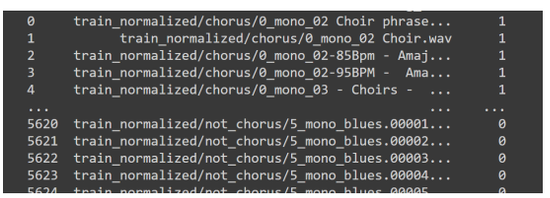
The final production dataset shape:

- 5992 not choir samples
- 3217 choir samples.

The dataset can be accessed and downloaded via this link:
https://drive.google.com/drive/folders/1kj5YwaznGRsJ7nFo6wWNz_j4BYjhUb8J?usp=sharing

Custom pytorch dataset creation

First we generate a csv dataframe with predefined classes length, links to the files and corresponding labels. The file is used as a relational reference for the **ChorusDataset class** (https://github.com/olegzinkevich/choir_audio_detection/blob/main/utils/chorus_dataset.py) __getitem__ method.

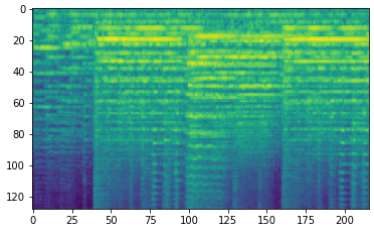


Each file is then processed with the **AudioProcessor class (**https://github.com/olegzinkevich/choir_audio_detection/blob/main/utils/audio_processor.py**)** (performing audio normalization - converting to 22050hz, rechanneling from stereo to mono; generating Mel Spectrogram and creating a tensor with a corresponding label for model training).

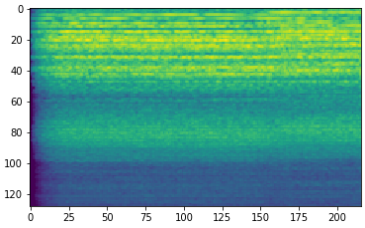
The Dataset is then propagated to the **Dataloader**, performing dataset batching, shuffling, etc. See model training pipeline (https://github.com/olegzinkevich/choir_audio_detection/blob/main/model_training.py)

Audio classification model

For choir detection, we interpret each five-second chunk of the audio file as a spectogram converted to a MEL scale. Each Mel spectrogram can be classified as an image by CNN model:



Not choir sample



Choir sample

3 different architectures of convolutional networks were trained (with different convolutional and pooling layers) with accuracy varying between 0.75 ~ 0.81. But the most significant results were achieved **fine-tuning pretrained Resnet model** (training only the last 2 layers - convolutional and Linear with 2 classes output) for our dataset. The model was trained during 10 epochs with a traditional train/validation split 0.7/0.3. As a loss function, a cross-entropy function was used. The **mean validation accuracy** with a Resnet model is: **0.95.**

Future enhancements

A rather high accuracy can be interpreted as possible overfitting, so for future pipeline enhancement, we plan to augment the dataset with more samples, use a learning rate scheduler for Adam optimizer, etc.

Model inference pipeline for the new song

For a model inference we follow the same steps as before:

- normalizing a new mp3 song.
- dividing each song into 5 seconds chunks,
- creating Mel spectrogram for every chunk,
- converting spectrograms to tensors with a proper NHCW dimension ([batch size, n channels, height, width]),
- loading pretrained model weights,
- perform classification of each audio chunk,
- merge chunks classified with 1 class to a final mp3 file.

Pytorch choir detection model weights/checkpoint can be downloaded via this link

If you want to reproduce the pipeline in your environment, **we highly recommend using classical, symphonic, or choral songs**, because the model was trained on clear choir samples without any background noise and accompanying instruments, so electronic or rock songs won't be classified properly.

Example cases

Example 1

Carl Orff "Carmina Burana" full 2 min. 34 sec. track:

https://drive.google.com/file/d/1Lf4uPufk4YUHpoFVwmyDKgP-tCw-GUV5/view?usp=sharing

Normalized and chunked audio segments from this track:

https://drive.google.com/file/d/1Lf4uPufk4YUHpoFVwmyDKgP-tCw-GUV5/view?usp=sharing

Final generated choir mp3 file from classified audio samples:

https://drive.google.com/file/d/1IO2TcUlJzs6aABNgFeY40hQX7g4FMYwL/view?usp=sharing

---

Example 2

3 min 15 sec track:

https://drive.google.com/file/d/1qwBOqVa2BNejJ28q36StFwGeZ50PFzHC/view?usp=sharing

Final generated choir mp3 file:

https://drive.google.com/file/d/12nKMSPDKu13lLkJJfZ4j0UaFxWbN4RXN/view?usp=sharing

Used python libraries

pydub - audio processing library.
pytroch - ML framework.
torchvision - image transformation package.
torchaudio - audio transformation package.
---
ffmpeg - command line audio/video toolbox for converting, encoding media files.

Oleg Zinkevich
+7 906 2621516
olegz@purebrand.ru