Introduction to Music Information Retrieval using Essentia.js

Albin Correya, Dmitry Bogdanov, Luis Joglar-Ongay*, Jorge Marcos-Fernández

Music Technology Group, Universitat Pompeu Fabra
*SonoSuite

https://github.com/MTG/essentia.js-tutorial-wac2021

Web Audio Conference 2021

About us

Albin Correya
@albincorreya
Research collaborator at **MTG**
Senior ML Engineer at
Moodagent A/S, Denmark
Dev of Essentia.js

Dmitry Bogdanov
@di_bogdanov
https://dbogdanov.com

Senior researcher and
lead developer of Essentia
at **MTG UPF**

Jorge Marcos
@MaferGeorge

Researcher and developer at
MTG UPF
Working on Essentia.js
applications

Luis Joglar-Ongay
@luisjoglar
PhD Candidate at **MTG UPF**
Research engineer at **SonoSuite**
Collaborator of Essentia.js

upf. Universitat Pompeu Fabra Barcelona

MTG Music Technology Group

https://www.upf.edu/web/mtg/

About this tutorial

1. Introduction to MIR and audio analysis. MIR applications and a typical analysis pipeline.
2. Using Essentia.js for music and audio analysis. Overview of available music audio features and application use-cases.
3. Getting started with Essentia.js. Writing your first “Hello world” application. Using Essenta.js for deferred-time vs. real-time analysis.
4. Available demos and template projects in a JavaScript playground.
5. Deep learning inference with Essentia.js using pre-trained machine learning models. Interface with TensorFlow.js.
6. Demos and examples of using Essentia.js for machine learning inference.
7. An industrial use-case example: audio problem detection for music distribution with Essentia.js.
8. QA and a playground session.

Audio analysis and Music Information Retrieval (MIR)

- Audio analysis (Wikipedia): "*Extraction of information and meaning from audio signals for analysis, classification, storage, retrieval, and synthesis*."

- Since 2000, we have a dedicated conference: International Society for Music Information Retrieval (ISMIR) conference

- ISMIR web page: "*processing, searching, organising and accessing music-related data*."
- Wikipedia: "*the interdisciplinary science of retrieving information from music*".
- Lots of MIR research is audio-based

- Good ISMIR tutorials:
  - A Basic Introduction to Audio-Related Music Information Retrieval
  - https://www.upf.edu/web/mtg/mir-overview-recent-developments-future

- Similarly, there is a lot of research on **sound information retrieval**

Some examples of music and sound retrieval tasks

Music/sound browsing interfaces
Music/sound search, exploration and recommendation

Music/sound classification
Semantic/tag-based retrieval

Query by example
Query by humming

Music source separation and instrument recognition
Music identification, cover song identification
Automatic music transcription
Music alignment (e.g., audio-to-score, audio-to-lyrics)

Music/sound generation, creative applications
Music/sound visualization

Extracting audio features

Many possibilities for applications operating with music/sound features retrieved from audio. Extracting this features is a fundamental step.

Low- and mid-level **sound and music features**

sound envelope, loudness, timbre, tonality, pitch/melody, onsets, rhythm, etc.

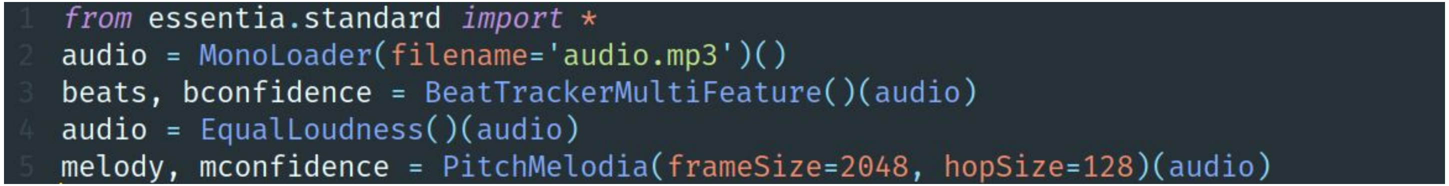
High-level **semantic descriptors**

type of sound, music genres, instrumentation, moods, danceability, etc.

Essentia https://essentia.upf.edu

Open-source library and tools for audio and music analysis, description and synthesis

- Extensive collection of reusable algorithms
- Written in **C++** and optimized for computational speed
- **Python** bindings for fast prototyping
- Feature extractors for **large-scale audio analysis**
- **Cross-platform** (Linux, Mac OS X, Windows, iOS, Android)
- Support for mobile platforms and **real-time** processing
- Developed at Music Technology Group, UPF



```python
from essentia.standard import *
audio = MonoLoader(filename='audio.mp3')()
beats, bconfidence = BeatTrackerMultiFeature()(audio)
audio = EqualLoudness()(audio)
melody, mconfidence = PitchMelodia(frameSize=2048, hopSize=128)(audio)
```

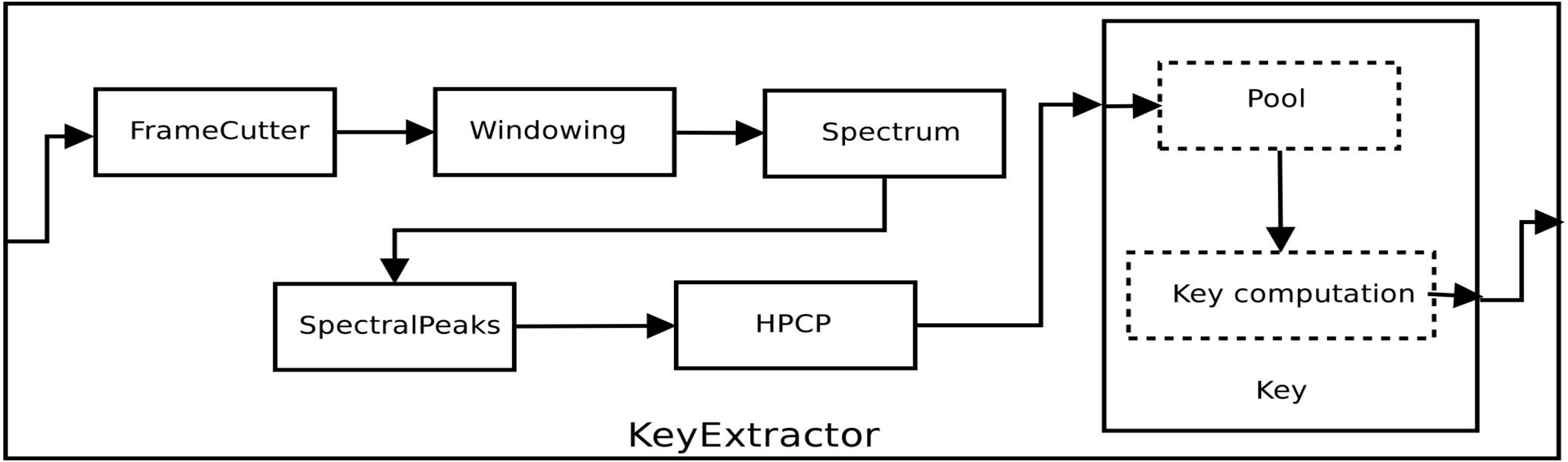
Essentia

260+ algorithms for audio signal processing and analysis, and sound and music description, developed at MTG

- Standard audio IO & DSP
- Sound and music descriptors
  - spectral features
  - rhythm and tempo
  - tonality, pitch and melody
  - loudness/dynamics
  - sound envelope
  - audio segmentation
  - fingerprinting
- Machine-learning based descriptors
  - genres, moods, instrumentation, …
  - SVM classifiers, inference with TensorFlow deep learning models

Modularity and processing chains

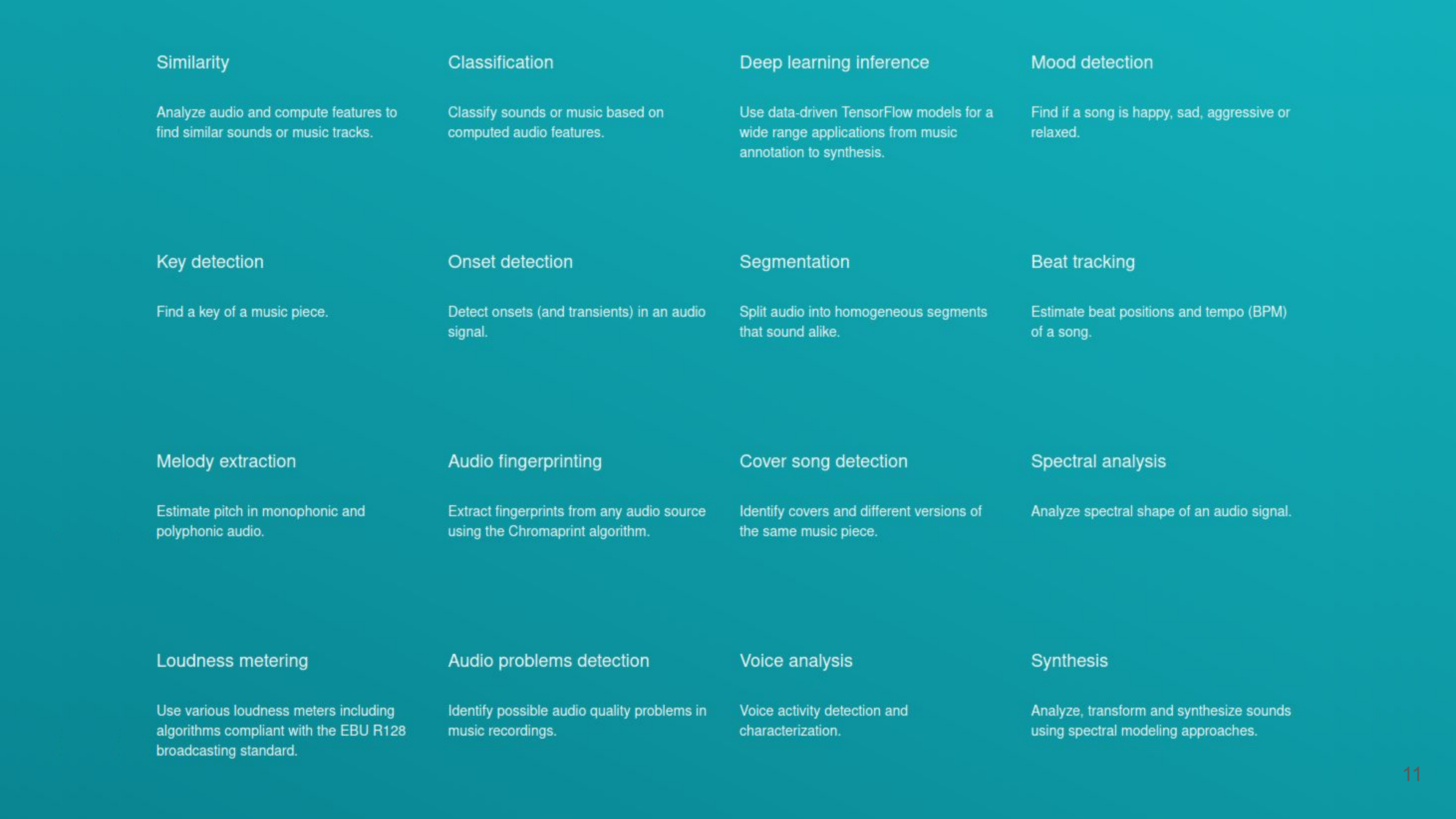
Users can build their own extractors for the descriptors they want to compute in a “data-flow” manner



Many applications in MIR and beyond

music
Music
classification
Detection
audio
recommendation
sound
analysis
Musical
Genre
Acoustic
similarity
based
using
content-based
detection
recordings
Melodic
Audio
Emotion
Computational
MediaEval
AcousticBrainz
Classification
performance
Analysis
Approach
browsing
Automatic
Recognition
Semantic
Identification
Using
Sound
preference
automatic
information
environmental
segmentation
learning
Learning
Task
examples
networks
system
research
evaluation
identification
visualization
Unsupervised
Rhythmic
feature
Feature
musical
Similarity
Robust
Monitoring
neural
Opera
Singing
Estimation
convolutional
Method
Multiple
transcription
real-time
Towards
makam
Visualization
Neural
Retrieval
Synthesis
Dataset
Indian
songs
Guitar
mood
Deep
Concatenative
metadata
corpus
Framework
bioacoustic
datasets
guitar
Playing
approach
melodic
Jingju
Key
user
art
jazz
effects
Tonic
deep
large
sounds
Goodness
Content-Based
Common
Features
Cover
urban
Tracking
pitch
Comparative
Turkey
indian
Beijing
Mood
Effects
Fusion
Wildlife
algorithms
intelligent
models
instrument
Networks
Dance
Recordings
Instrument
approaches
collections
Artificial
recognition
computational
Improving
high-level
Corpus
Sounds

ESSENTIA



Similarity

Analyze audio and compute features to find similar sounds or music tracks.

Classification

Classify sounds or music based on computed audio features.

Deep learning inference

Use data-driven TensorFlow models for a wide range applications from music annotation to synthesis.

Mood detection

Find if a song is happy, sad, aggressive or relaxed.

Key detection

Find a key of a music piece.

Onset detection

Detect onsets (and transients) in an audio signal.

Segmentation

Split audio into homogeneous segments that sound alike.

Beat tracking

Estimate beat positions and tempo (BPM) of a song.

Melody extraction

Estimate pitch in monophonic and polyphonic audio.

Audio fingerprinting

Extract fingerprints from any audio source using the Chromaprint algorithm.

Cover song detection

Identify covers and different versions of the same music piece.

Spectral analysis

Analyze spectral shape of an audio signal.

Loudness metering

Use various loudness meters including algorithms compliant with the EBU R128 broadcasting standard.

Audio problems detection

Identify possible audio quality problems in music recordings.

Voice analysis

Voice activity detection and characterization.

Synthesis

Analyze, transform and synthesize sounds using spectral modeling approaches.

Music and Audio analysis on the Web, so far

Plenty of web applications use audio analysis capabilities

But ...

Most of them do audio analysis on the servers

Since

- Limitations of the client devices and the web browsers itself.
- Web Audio API capabilities are limited for audio analysis.
- Lack of extensive software libraries for on-client computation.

Existing audio analysis APIs and libraries on the web

Spotify API - audio features for music in Spotify

Freesound API - audio analysis for sounds in Freesound with Essentia

AcousticBrainz API - database of audio features extracted with Essentia

Meyda https://meyda.js.org - written purely in JS

JS-Xtract https://github.com/nickjillings/js-xtract - JS conversion of LibXtract

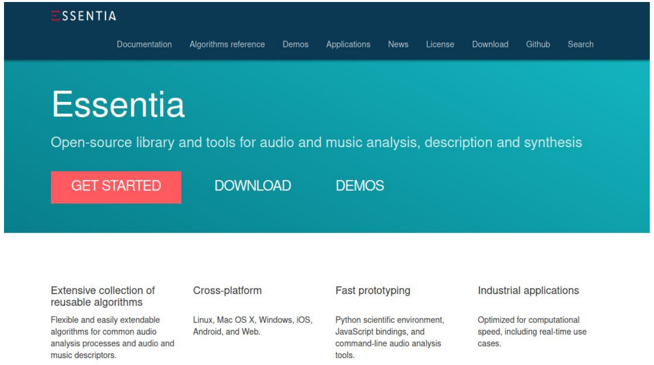
aubiojs https://qiuxiang.github.io/aubiojs - JS conversion of some Aubio algorithms

Music/audio analysis with Essentia and Essentia.js

ESSENTIA

C++ library

https://essentia.upf.edu



ESSENTIA

Documentation Algorithms reference Demos Applications News License Download Github Search

Essentia

Open-source library and tools for audio and music analysis, description and synthesis

GET STARTED DOWNLOAD DEMOS

Extensive collection of reusable algorithms

Flexible and easily extendable algorithms for common audio analysis processes and audio and music descriptors.

Cross-platform

Linux, Mac OS X, Windows, iOS, Android, and Web.

Fast prototyping

Python scientific environment, JavaScript bindings, and command-line audio analysis tools.

Industrial applications

Optimized for computational speed, including real-time use cases.

essentia.JS

An open-source JavaScript library, powered by Essentia WebAssembly

https://essentia.upf.edu/essentiajs

An open-source Javascript library for music/audio analysis and processing,
powered by WebAssembly

https://essentia.upf.edu/essentiajs

Correya, A. A., Bogdanov, D., Joglar-Ongay, L., & Serra, X. (2020). Essentia. js: a JavaScript library for music and audio analysis on the web. *Proceedings of the 21st International Society for Music Information Retrieval Conference; 2020 Oct 11-16; Montréal, Canada.[Canada]: ISMIR; 2020. p. 605-12.*. International Society for Music Information Retrieval (ISMIR).

License

AGPLv3 + commercial license under request

https://www.gnu.org/licenses/agpl-3.0.en.html

https://opensource.stackexchange.com/questions/4442/how-does-the-agpl-apply-to-javascript-libraries

Let's checkout essentia.js in action!

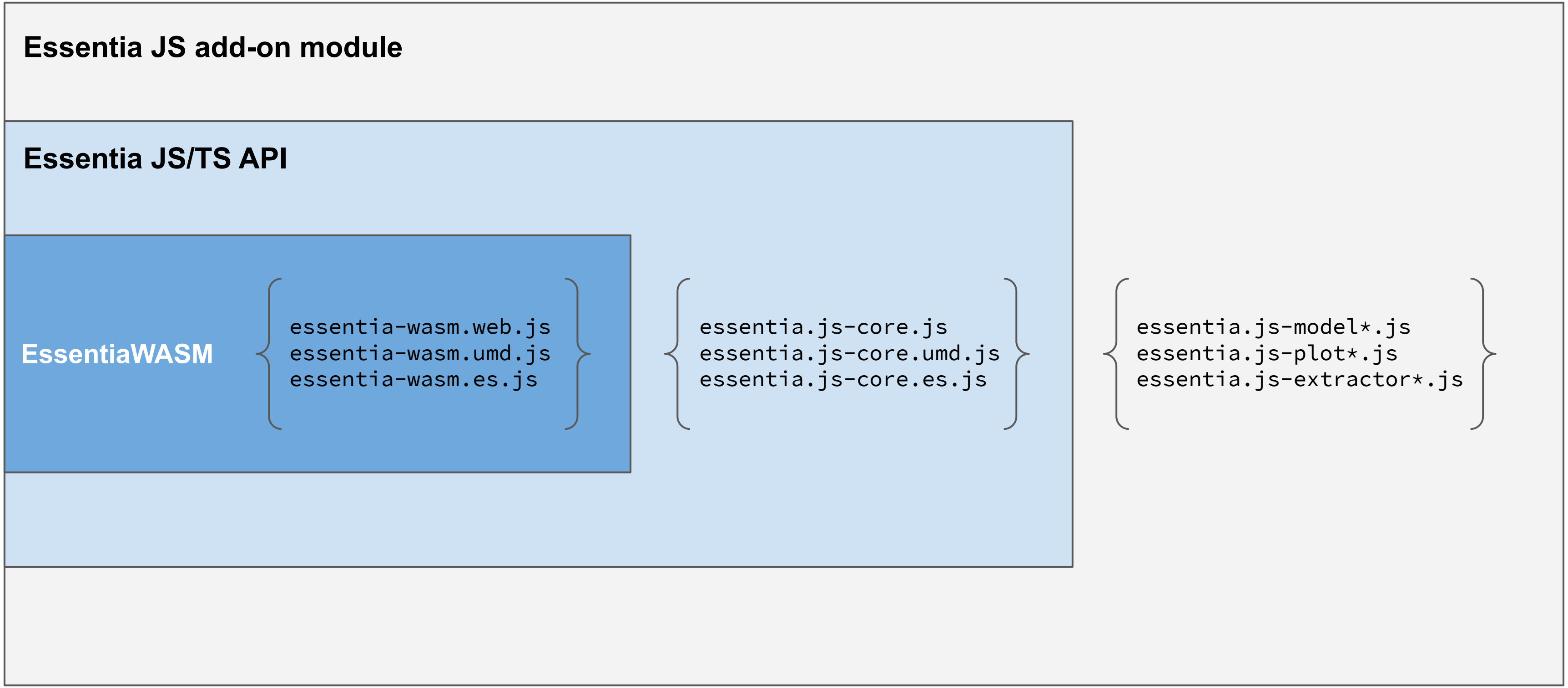
essentia.js examples

Design choices & functionalities

- Easy installation
  - NPM, CDN etc
- User friendly API & utility tools
  - Easy-to-use
- Modularity & Extensibility
  - Easy-to-customize and build
- Web standards compliance
  - Web Audio API, Audio Worklet, ES6 etc.

- Lightweight and lesser dependencies
  - Small as 2.7 MB ~
- Reproducibility across different platforms
  - Native and Web (C++, Python and JavaScript)
- Extensive documentation and examples

Abstractions



Essentia JS add-on module

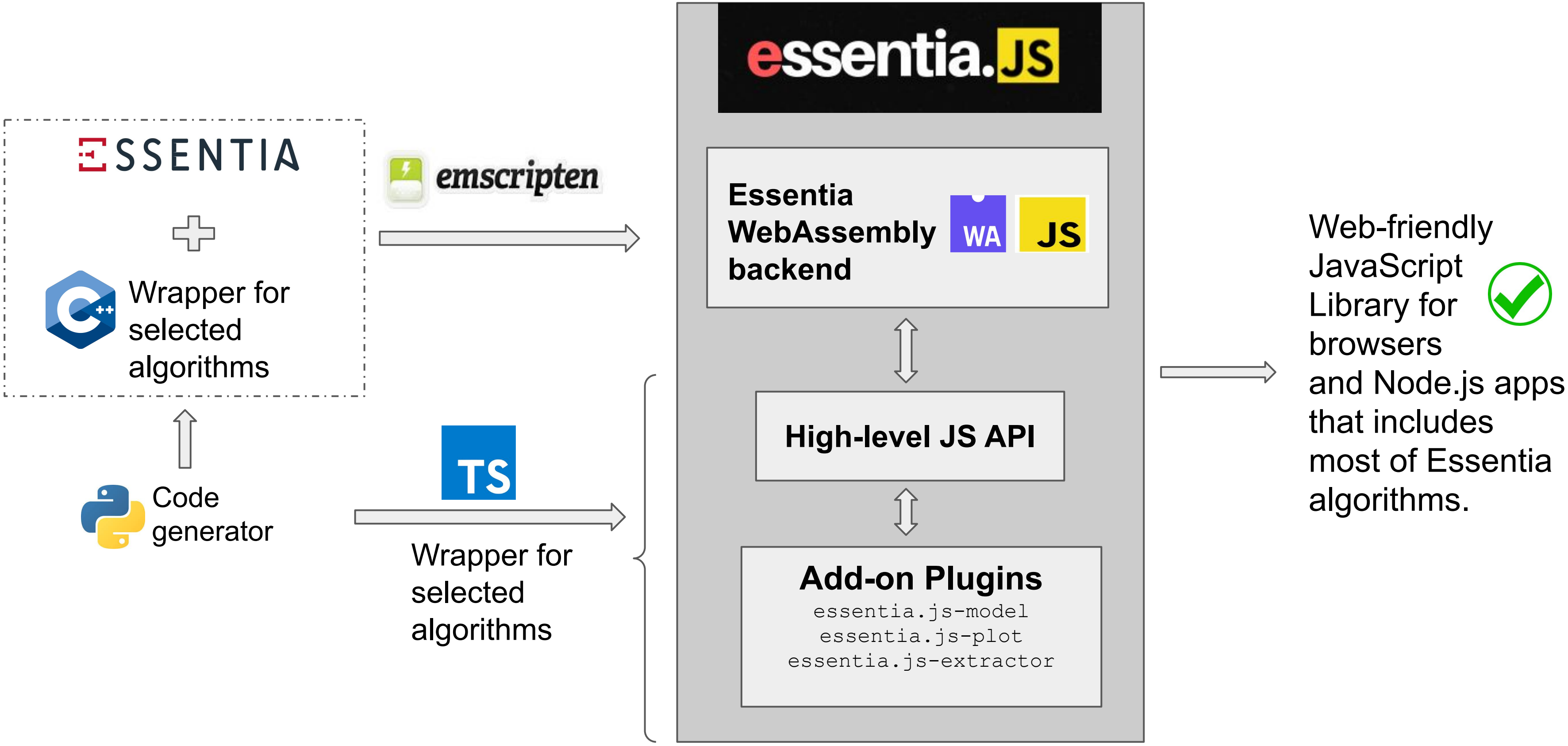
Essentia JS/TS API

EssentiaWASM

essentia-wasm.web.js
essentia-wasm.umd.js
essentia-wasm.es.js

essentia.js-core.js
essentia.js-core.umd.js
essentia.js-core.es.js

essentia.js-model*.js
essentia.js-plot*.js
essentia.js-extractor*.js



Wrapper for selected algorithms

Code generator

Wrapper for selected algorithms

Essentia WebAssembly backend

High-level JS API

Add-on Plugins

essentia.js-model
essentia.js-plot
essentia.js-extractor

Web-friendly JavaScript Library for browsers and Node.js apps that includes most of Essentia algorithms.

https://emscripten.org/
https://developer.mozilla.org/en-US/docs/WebAssembly
https://www.typescriptlang.org/

Known limitations

- As of now, only 200 essentia algorithms are supported (here).
- But all algorithms are available via custom C++ wrapper.
- Essentia.js has not been through rigorous QA tests. Hence, not fully production ready and backward compatible.

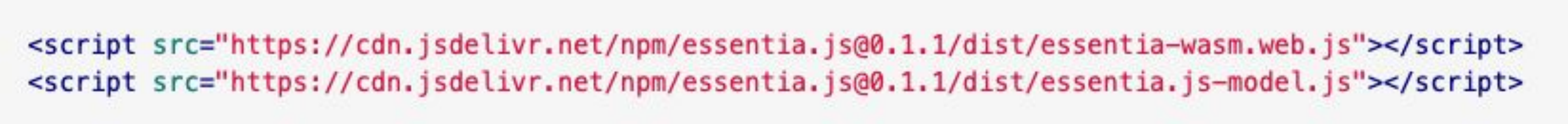
But, we are currently on active development on making towards a stable production-ready release.

You're most welcome to contribute!

Getting started

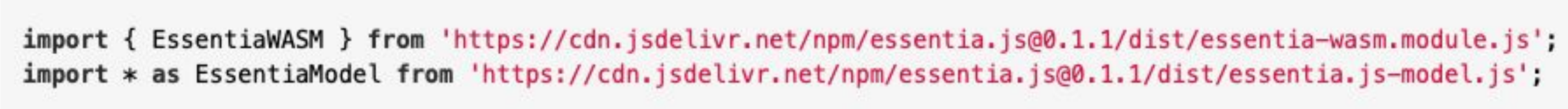
```
npm install essentia.js
```

script </>



```
<script src="https://cdn.jsdelivr.net/npm/essentia.js@0.1.1/dist/essentia-wasm.web.js"></script>
<script src="https://cdn.jsdelivr.net/npm/essentia.js@0.1.1/dist/essentia.js-model.js"></script>
```

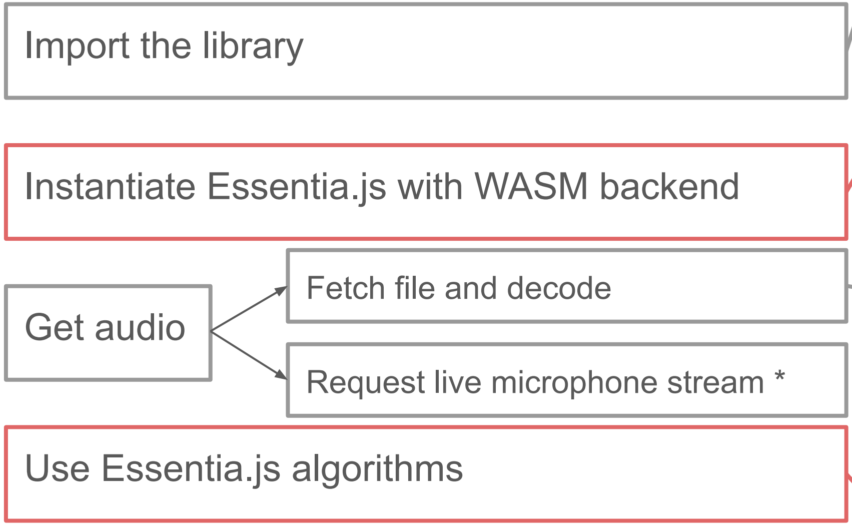
ES6 style imports



```
import { EssentiaWASM } from 'https://cdn.jsdelivr.net/npm/essentia.js@0.1.1/dist/essentia-wasm.module.js';
import * as EssentiaModel from 'https://cdn.jsdelivr.net/npm/essentia.js@0.1.1/dist/essentia.js-model.js';
```

Basic HTML use

Steps:



Import the library

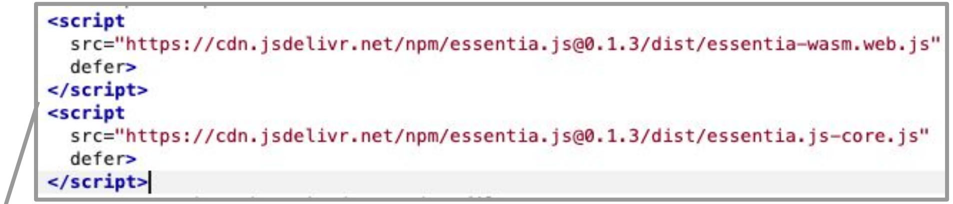
Instantiate Essentia.js with WASM backend

Get audio

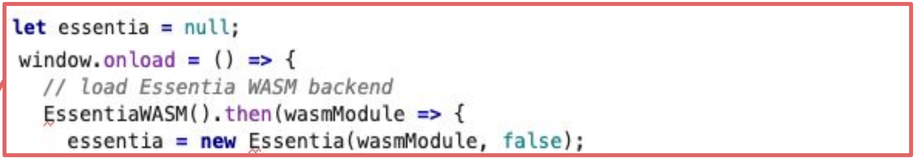
Fetch file and decode

Request live microphone stream *

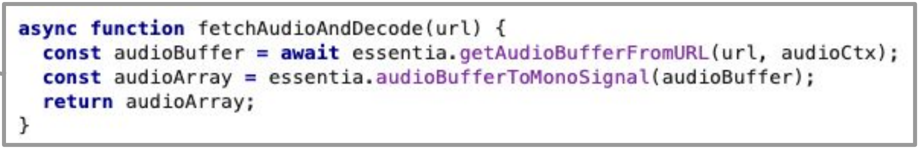
Use Essentia.js algorithms



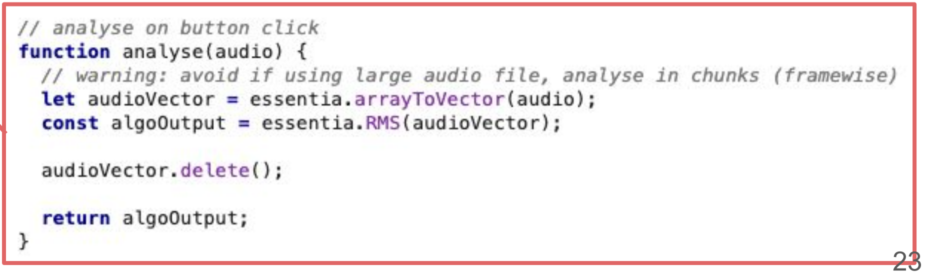
```html
<script
  src="https://cdn.jsdelivr.net/npm/essentia.js@0.1.3/dist/essentia-wasm.web.js"
  defer>
</script>
<script
  src="https://cdn.jsdelivr.net/npm/essentia.js@0.1.3/dist/essentia.js-core.js"
  defer>
</script>
```



```javascript
let essentia = null;
 window.onload = () => {
   // load Essentia WASM backend
   EssentiaWASM().then(wasmModule => {
     essentia = new Essentia(wasmModule, false);
```

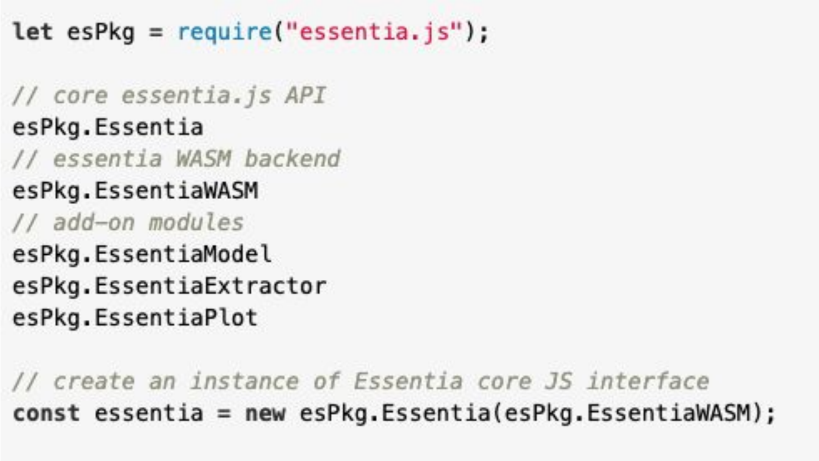


```javascript
async function fetchAudioAndDecode(url) {
  const audioBuffer = await essentia.getAudioBufferFromURL(url, audioCtx);
  const audioArray = essentia.audioBufferToMonoSignal(audioBuffer);
  return audioArray;
}
```



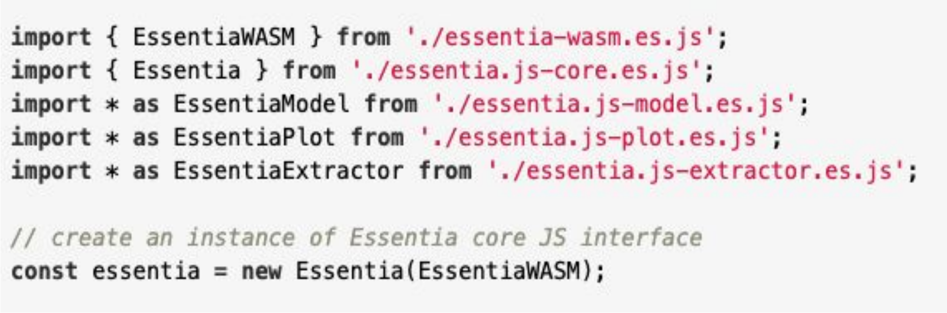
```javascript
// analyse on button click
function analyse(audio) {
  // warning: avoid if using large audio file, analyse in chunks (framewise)
  let audioVector = essentia.arrayToVector(audio);
  const algoOutput = essentia.RMS(audioVector);

  audioVector.delete();

  return algoOutput;
}
```

On Node.js



```
let esPkg = require("essentia.js");

// core essentia.js API
esPkg.Essentia
// essentia WASM backend
esPkg.EssentiaWASM
// add-on modules
esPkg.EssentiaModel
esPkg.EssentiaExtractor
esPkg.EssentiaPlot

// create an instance of Essentia core JS interface
const essentia = new esPkg.Essentia(esPkg.EssentiaWASM);
```

ES6 import



```
import { EssentiaWASM } from './essentia-wasm.es.js';
import { Essentia } from './essentia.js-core.es.js';
import * as EssentiaModel from './essentia.js-model.es.js';
import * as EssentiaPlot from './essentia.js-plot.es.js';
import * as EssentiaExtractor from './essentia.js-extractor.es.js';

// create an instance of Essentia core JS interface
const essentia = new Essentia(EssentiaWASM);
```

Here, `EssentiaWASM` backend is loaded asynchronously

Hands-on session

Go to: https://glitch.com/@jmarcosfer/wac-21-essentia-js-tutorial

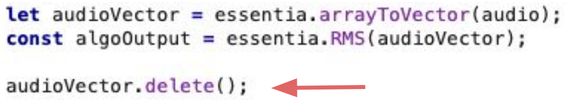
We will cover:

- Non-realtime use
  - Main UI thread: https://glitch.com/~essentiajs-core-non-rt
  - Workers: https://glitch.com/~essentiajs-core-non-rt-worker
  - Node.js: https://glitch.com/~essentiajs-core-node
- Realtime use
  - ScriptProcessorNode: https://glitch.com/~essentiajs-core-realtime-spn
  - AudioWorklets: https://glitch.com/~essentiajs-core-realtime-aw

Implementation Details

Some implementation details (mostly RT):

- Non-JS object clean-up:



```
let audioVector = essentia.arrayToVector(audio);
const algoOutput = essentia.RMS(audioVector);

audioVector.delete();
```

- Code injection in AudioWorklets → ES6 imports unsupported for Worklets on Firefox
    - P. Adenot's URLFromFiles (thanks!)
    - used here
- Frame-size matching
    - using ChromeLabs wasm-audio-helper.js
- Getting analysis results from Worklet to UI thread with SharedArrayBuffer
    - P. Adenot's ringbuf.js

Q/A Session

Break time

ESSENTIA

2021 web audio conference wac

upf. Universitat Pompeu Fabra *Barcelona*

MTG Music Technology Group

Machine Learning on the Web

JS

TensorFlow.js

ONNX

ml5

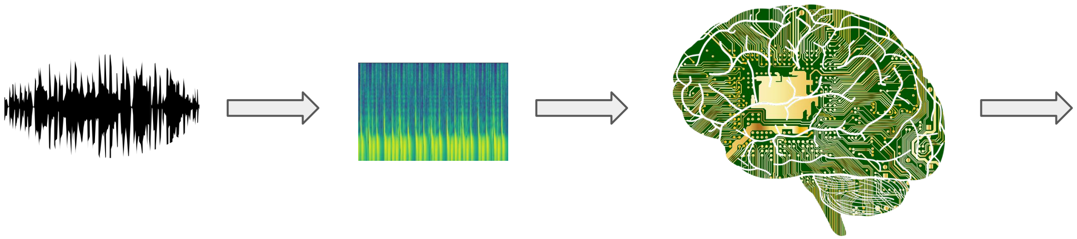
magenta

etc..

Essentia TensorFlow models

A repository of publicly available deep learning models for music analysis tasks

https://essentia.upf.edu/machine_learning.html



Audio Input

Mel-spectrogram

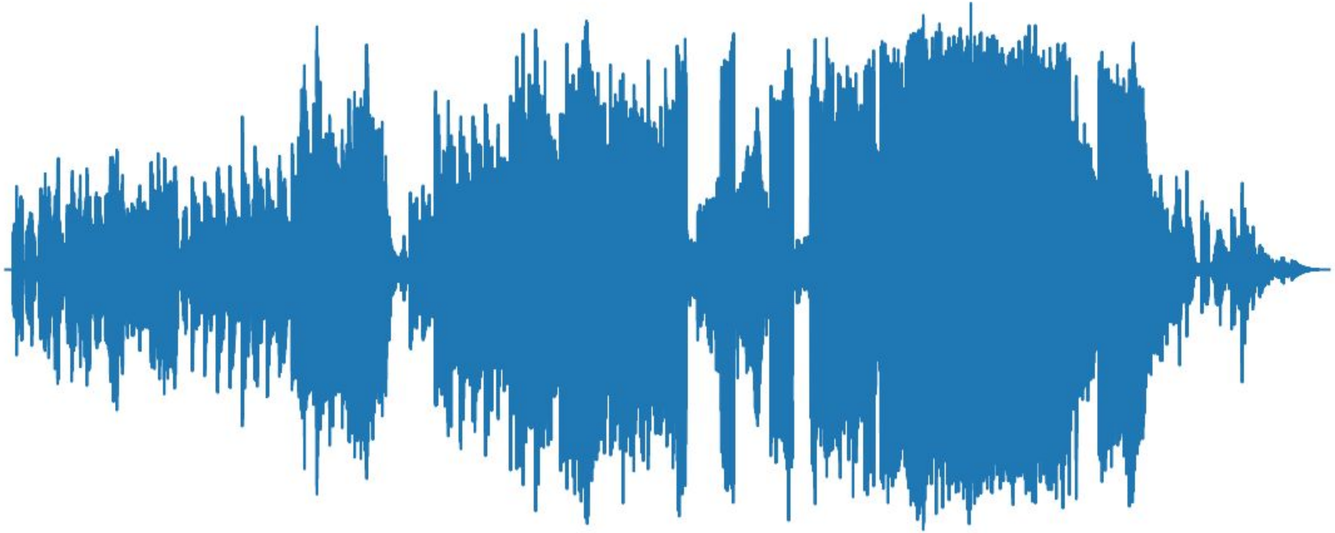
Essentia TensorFlow Models

Music tags
Genres and styles
Moods
Danceability
Instrumentation
Vocal/instrumental
Voice gender
Tempo
…

Predictions

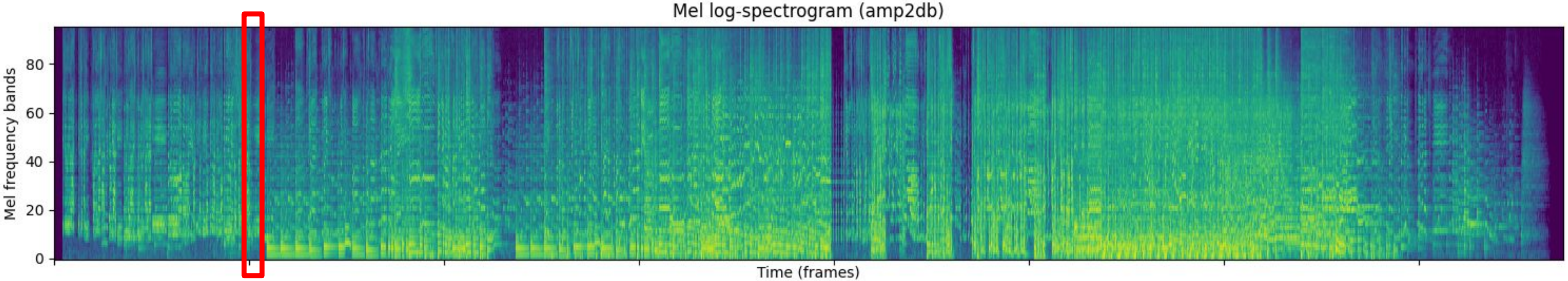
Now, we ported these optimized models to Tensorflow.js format for its usage on JS.

Auto-tagging and classification



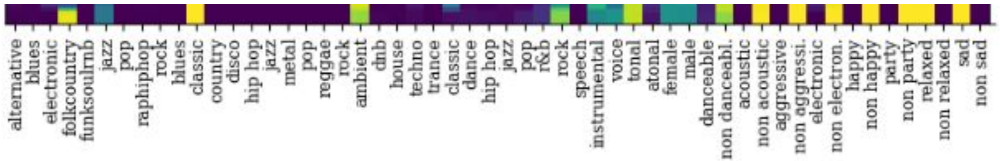
Queen - Bohemian Rhapsody

Auto-tagging and classification



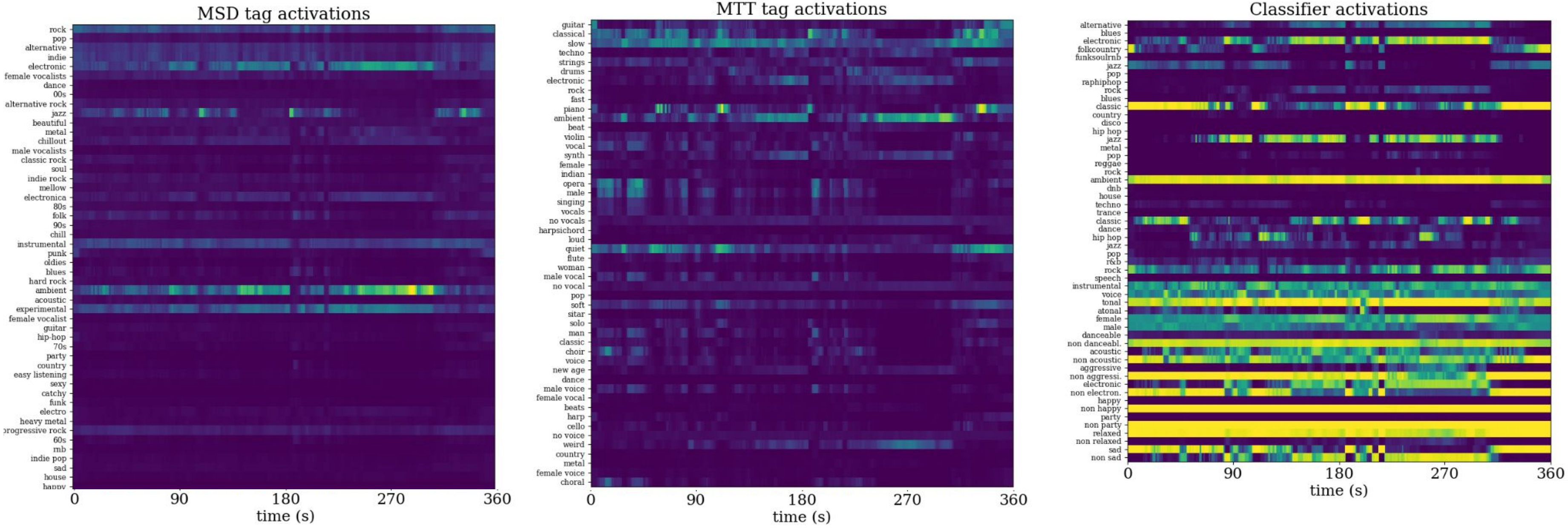
Audio chunk
(receptive field)

Queen - Bohemian Rhapsody



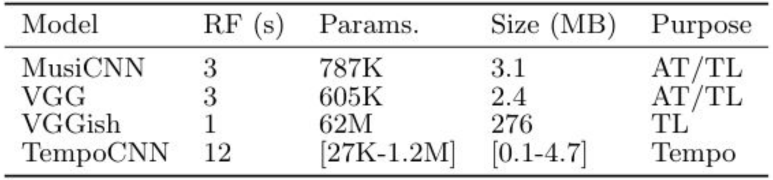
Predictions vector (activation values)

Auto-tagging and classification



Queen - Bohemian Rhapsody

Auto-tagging / embeddings / tempo models

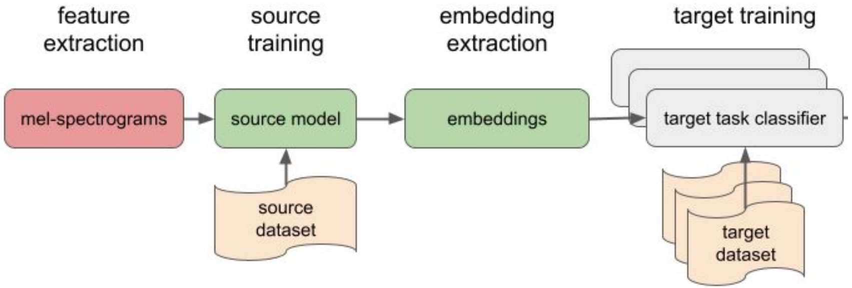


Model	RF (s)	Params.	Size (MB)	Purpose
MusiCNN	3	787K	3.1	AT/TL
VGG	3	605K	2.4	AT/TL
VGGish	1	62M	276	TL
TempoCNN	12	[27K-1.2M]	[0.1-4.7]	Tempo

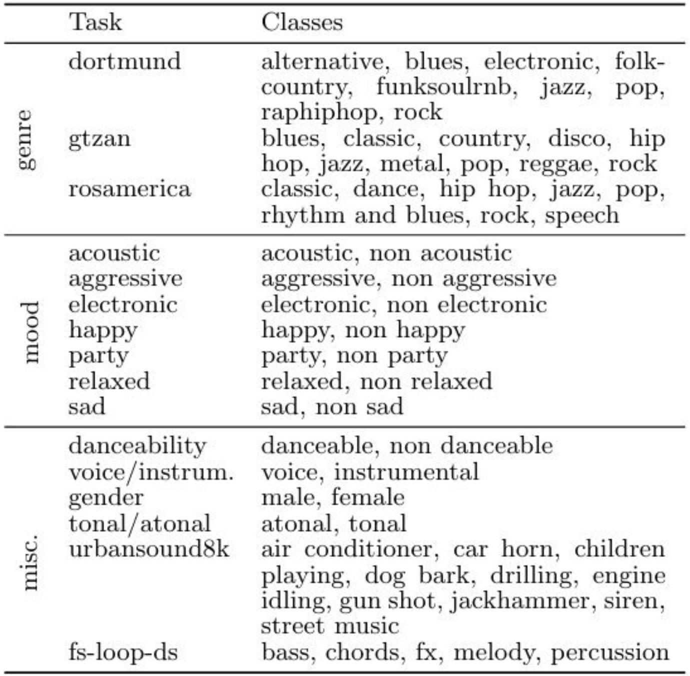
RF = receptive field
AT = auto-tagging
TL = transfer learning (embeddings)

Transfer learning classifiers

How we train the models



MusiCNN + MSD/MTT
VGG + MSD/MTT
VGGish + AudioSet



	Task	Classes
genre	dortmund	alternative, blues, electronic, folk-country, funksoulrnb, jazz, pop, raphiphop, rock
	gtzan	blues, classic, country, disco, hip hop, jazz, metal, pop, reggae, rock
	rosamerica	classic, dance, hip hop, jazz, pop, rhythm and blues, rock, speech
mood	acoustic	acoustic, non acoustic
	aggressive	aggressive, non aggressive
	electronic	electronic, non electronic
	happy	happy, non happy
	party	party, non party
	relaxed	relaxed, non relaxed
	sad	sad, non sad
misc.	danceability	danceable, non danceable
	voice/instrum.	voice, instrumental
	gender	male, female
	tonal/atonal	atonal, tonal
	urbansound8k	air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun shot, jackhammer, siren, street music
	fs-loop-ds	bass, chords, fx, melody, percussion

Essentia TensorFlow Models downloads

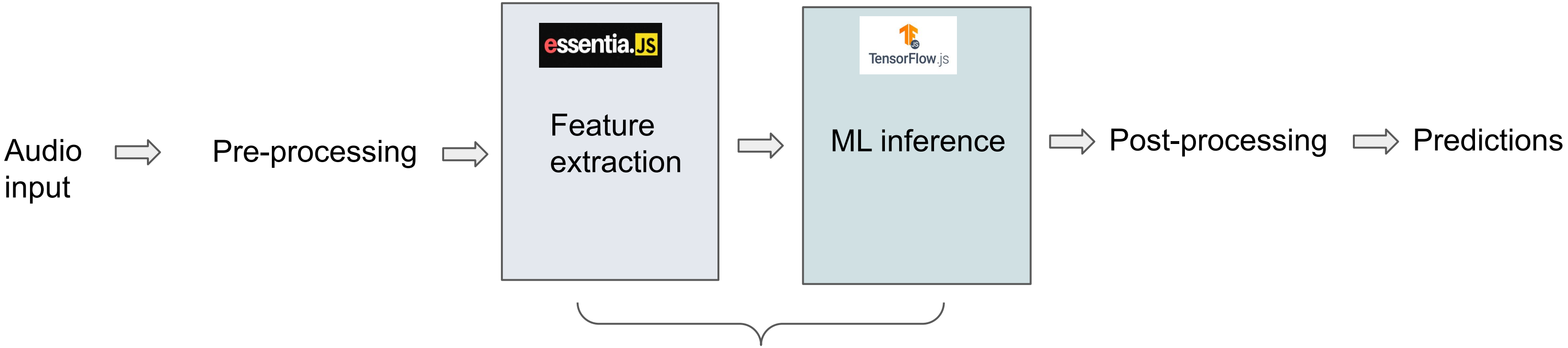
https://essentia.upf.edu/models/

https://essentia.upf.edu/machine_learning.html

Models available under the CC BY-NC-ND 4.0 license

Essentia.js + TensorFlow.js

Inference pipeline



Audio input

Pre-processing

Feature extraction

ML inference

Post-processing

Predictions

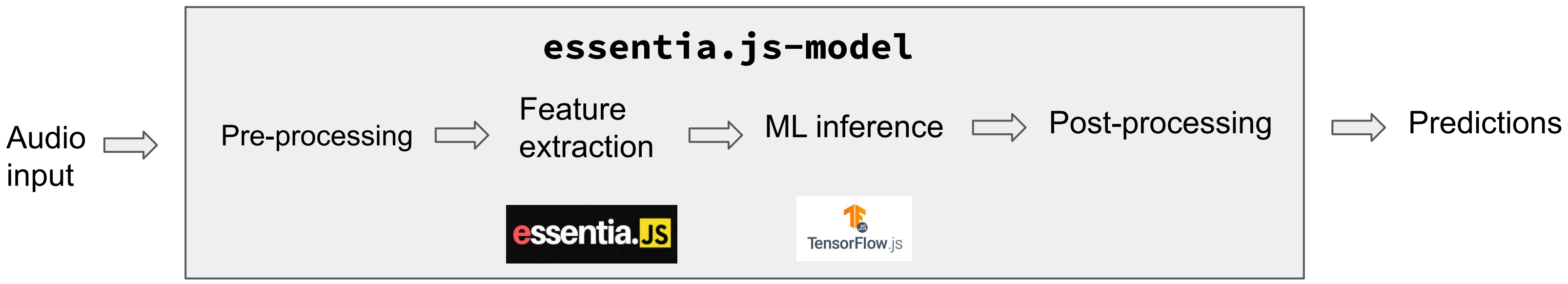
Often need some domain knowledge here

But how to make this more accessible and easy to use for everyone?

essentia.js-model

- Add-on module for the essentia.js library.
- Simple JS API
- Support Web Workers for feature extraction and inference separately.

Inference pipeline



essentia.js-model

Audio input

Pre-processing

Feature extraction

ML inference

Post-processing

Predictions

essentia.JS

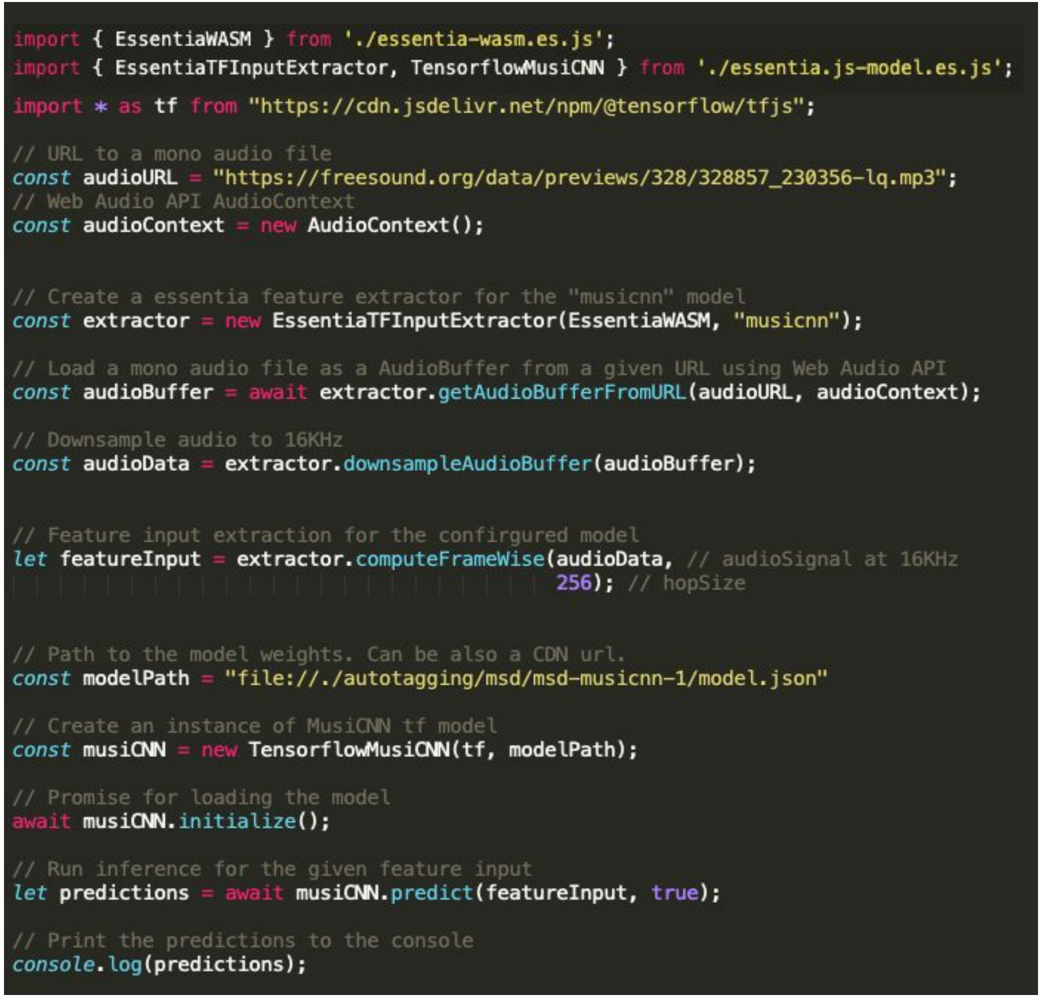
TensorFlow.js

A. Correya, P. Alonso-Jiménez, J. Marcos-Fernández, X. Serra, and D. Bogdanov. Essentia TensorFlow models for audio and music processing on the web. In Web Audio Conference (WAC 2021), 2021.

Let's checkout some models in action!

Autotagging with MusiCNN

Mood Classification

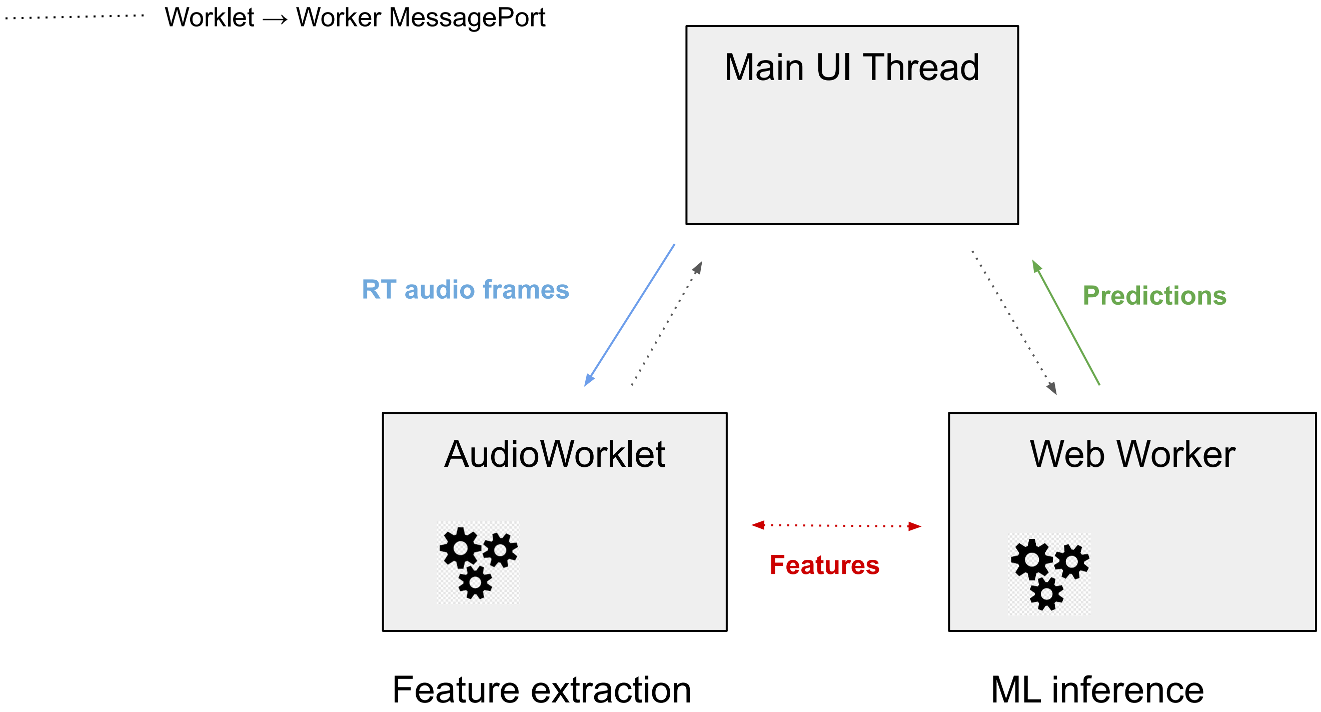


```
import { EssentiaWASM } from './essentia-wasm.es.js';
import { EssentiaTFInputExtractor, TensorflowMusiCNN } from './essentia.js-model.es.js';
import * as tf from "https://cdn.jsdelivr.net/npm/@tensorflow/tfjs";

// URL to a mono audio file
const audioURL = "https://freesound.org/data/previews/328/328857_230356-lq.mp3";
// Web Audio API AudioContext
const audioContext = new AudioContext();


// Create a essentia feature extractor for the "musicnn" model
const extractor = new EssentiaTFInputExtractor(EssentiaWASM, "musicnn");

// Load a mono audio file as a AudioBuffer from a given URL using Web Audio API
const audioBuffer = await extractor.getAudioBufferFromURL(audioURL, audioContext);

// Downsample audio to 16KHz
const audioData = extractor.downsampleAudioBuffer(audioBuffer);


// Feature input extraction for the confirgured model
let featureInput = extractor.computeFrameWise(audioData, // audioSignal at 16KHz
                                              256); // hopSize


// Path to the model weights. Can be also a CDN url.
const modelPath = "file://./autotagging/msd/msd-musicnn-1/model.json"

// Create an instance of MusiCNN tf model
const musiCNN = new TensorflowMusiCNN(tf, modelPath);

// Promise for loading the model
await musiCNN.initialize();

// Run inference for the given feature input
let predictions = await musiCNN.predict(featureInput, true);

// Print the predictions to the console
console.log(predictions);
```

Hands on

- Hands-on exercise using MusiCNN non-realtime: https://glitch.com/~essentiajs-models-non-rt
- Also can be used with Web Workers
- Realtime: https://glitch.com/~essentiajs-models-rt

Real-time implementation

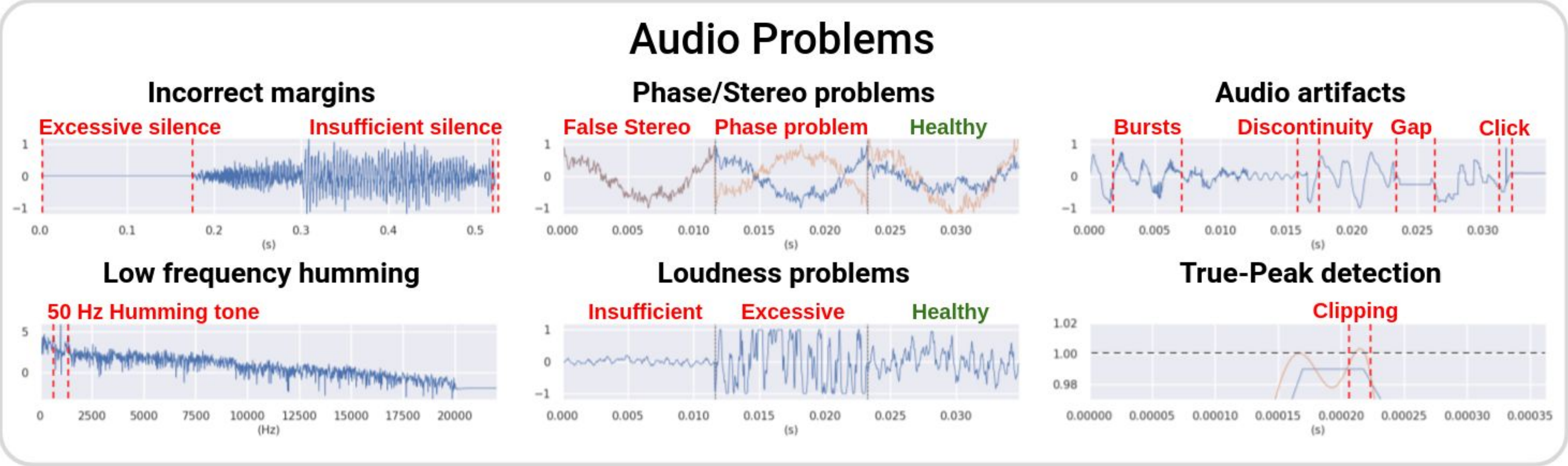


Industrial application

SonoSuite

Audio Problems

Example: https://glitch.com/~audio-problems-detection

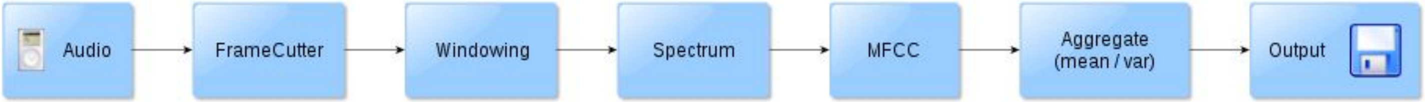


P. Alonso-Jiménez, L. Joglar-Ongay, X. Serra, and D. Bogdanov. Automatic Detection of Audio Problems for Quality Control in Digital Music Distribution. In AES Convention 146 (March 2019).

Industrial application

Custom extractor

- For better performance on JS
- Some algorithms are not yet working in Essentia.js



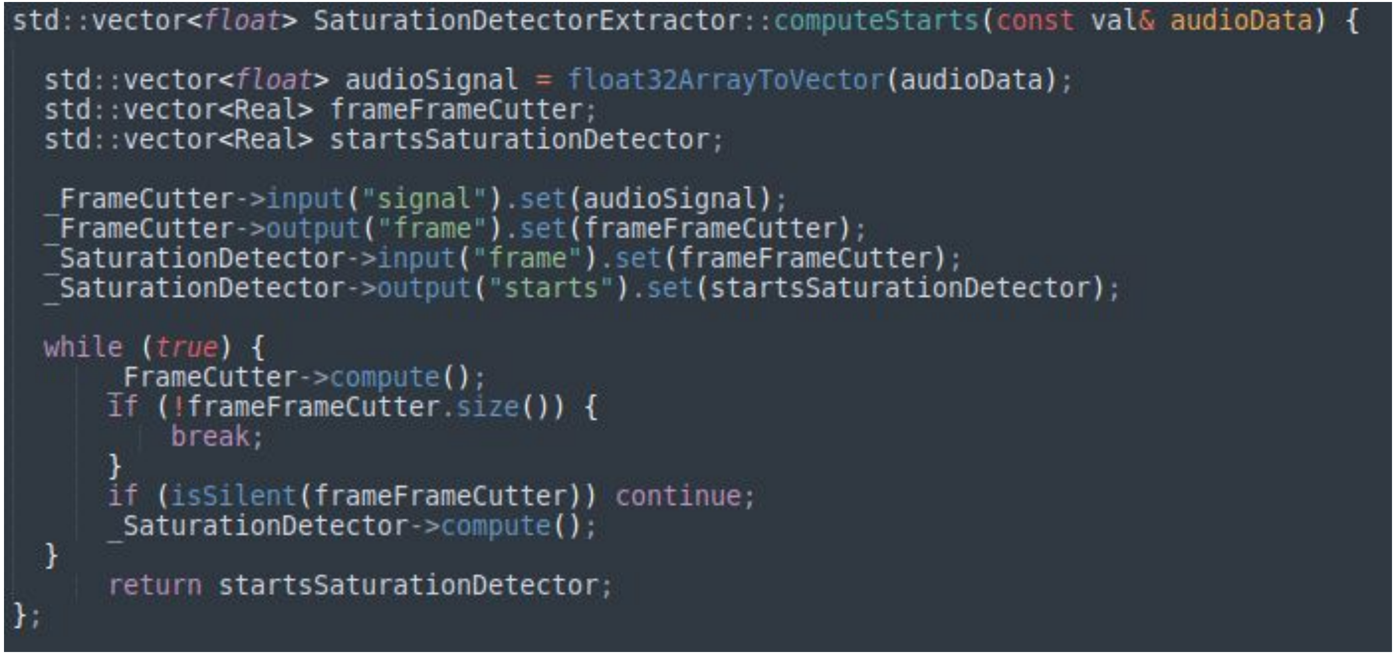
How to:

https://github.com/MTG/essentia.js/blob/master/src/cpp/custom/essentia_custom_extractor.h
https://github.com/MTG/essentia.js/blob/master/src/cpp/custom/essentia_custom_extractor.cpp
https://github.com/MTG/essentia.js/blob/master/src/cpp/custom/bindings_extractor.cpp

Then compile using the provided Makefile ensuring you have all requirements (or using the docker image)

Industrial application

Custom extractor



```
std::vector<float> SaturationDetectorExtractor::computeStarts(const val& audioData) {

  std::vector<float> audioSignal = float32ArrayToVector(audioData);
  std::vector<Real> frameFrameCutter;
  std::vector<Real> startsSaturationDetector;

  _FrameCutter->input("signal").set(audioSignal);
  _FrameCutter->output("frame").set(frameFrameCutter);
  _SaturationDetector->input("frame").set(frameFrameCutter);
  _SaturationDetector->output("starts").set(startsSaturationDetector);

  while (true) {
      _FrameCutter->compute();
      if (!frameFrameCutter.size()) {
          break;
      }
      if (isSilent(frameFrameCutter)) continue;
      _SaturationDetector->compute();
  }
      return startsSaturationDetector;
};
```

QA & Playground session

Build your own MIR web application :)

Instructions

- Use any JS playground of your choice eg: Glitch, Runkit, Codepen, jsfiddle etc.
- You can use any algorithms, models or their combinations to make a fun prototype.
- Most welcome to remix our examples on Glitch.

Optional,

- Tag your essentia.js prototypes on Twitter tagging @wac2021 with hashtag #essentiajs-wac2021

Thanks

If you have any more questions, feedbacks etc,
feel free to reach out to any of us

@albincorreya
@di_bogdanov
@MaferGeorge
@luisjoglar

ESSENTIA

2021 web audio conference wac

upf. Universitat Pompeu Fabra Barcelona

MTG Music Technology Group