

# 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

**E**SSENTIA



Universitat  
Pompeu Fabra  
Barcelona

**MTG**  
Music Technology  
Group

# About us



**Albin Correya**

[@albincorreya](https://twitter.com/albincorreya)

Research collaborator at **MTG**

Senior ML Engineer at

**Moodagent A/S, Denmark**

Lead developer of Essentia.js



**Dmitry Bogdanov**

[@di\\_bogdanov](https://di_bogdanov.github.io)  
<https://dbogdanov.com>

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



**Jorge Marcos**

[@MaferGeorge](https://twitter.com/MaferGeorge)

Researcher and developer at  
**MTG UPF**

Working on Essentia.js  
applications



**Luis Joglar-Ongay**

[@luisjoglar](https://twitter.com/luisjoglar)

PhD Candidate at **MTG UPF**

Research Scientist at **SonoSuite**

Collaborator of Essentia.js



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 Essentia.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
- 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
- A few MIR tutorials:
  - [A Basic Introduction to Audio-Related Music Information Retrieval](#)
  - [Music Information Retrieval: Overview, Recent Developments and Future Challenges](#)
- 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*



SongExplorer for Reactable

# Some examples of music and sound retrieval tasks

*Query by example, sound/music similarity*

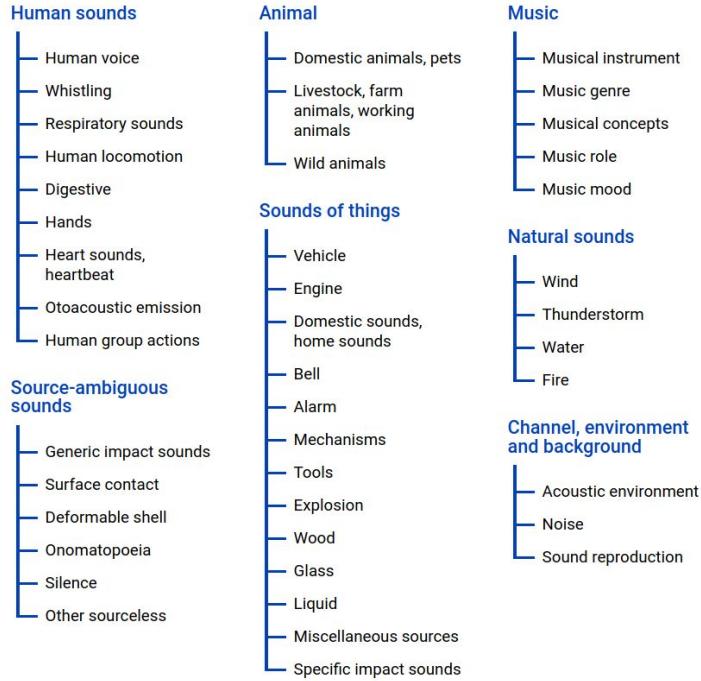
The screenshot shows the freesound.org website. At the top, there is a navigation bar with links for Register, Log In, and Upload Sounds. Below the navigation is a search bar labeled "search sounds". The main content area is titled "Query sound" and displays a waveform preview and metadata for a sound file named "Amen Break D (200 BPM)". The file has a rating of 5 stars and was uploaded by "Kevcio" on February 11th, 2015, with 1726 downloads and 9 comments. Below this section, there is a heading "Similar sounds" followed by a list of four more sound files with their waveforms, titles, ratings, uploaders, and brief descriptions.

Sound File	Uploader	Rating	Description
170_scorpio_A.wav	OaSyntax	5★	the main part of the "scorpio" break by Dennis Coffey
dnb_beat_2_Normal_175.wav	dahovv	5★	Typical drum'n'bass loop with punchy kick & snare with amen break in the back.
Amen Break E (170 BPM)	Kevcio	5★	Cutup Amenbreak
Mecha-jazz.wav	shart69	5★	Funky amen break created by sequencing individual hits.

<https://freesound.org>

# Some examples of music and sound retrieval tasks

## Music/sound classification



AcousticBrainz

Recording "A la fin cette bergère" by Anthoine Boesset

Metadata	value
MBID	0297fad5-e273-4c56-99d6-fee93d574227
title	A la fin cette bergère
artist	Anthoine Boesset
release	Je meurs sans mourir
track number	13 / 20
track length	3:57

Submission #1 out of 4 →

High-level information

Voice, timbre, gender, etc.	value	probability
Voice	Voice	93.4%
Gender	Female	95.2%
Danceability	Not Danceable	97.3%
Tonal	Tonal	54.7%
Timbre	Bright	85.0%
ISMIR04 Rhythm	Tango	85.1%

Moods	value	probability
Electronic	Not Electronic	78.2%
Party	Not Party	99.2%
Aggressive	Not Aggressive	94.0%
Acoustic	Acoustic	94.0%
Happy	Not Happy	94.5%
Sad	Sad	94.4%
Relaxed	Relaxed	97.1%
Mirex method	Rolling, Cheery, Fun, Sweet, Amiable/Good Natured	98.6%

# Some examples of music and sound retrieval tasks

## Semantic/tag-based retrieval

Jamendo Licensing

250,000+ Royalty Free Music Tracks

Music library for any project. All rights included.

What are you looking for? (eg: happy, rock...)

FEATURED THEMES & MOODS GENRES INSTRUMENTS ALL

Instrument	Count
Piano	29158 tracks
Ukulele	3267 tracks
Percussion	9774 tracks
String	3617 tracks
Synthesizer	36699 tracks
Guitar	16114 tracks
Drums	9774 tracks
Violin	3617 tracks
Orchestral	2045 tracks
Trumpet	2800 tracks

freesound

Register Log In Upload Sounds

search sounds

Tags

80s+ abstract+ acoustic+ air+ alien- alien-sound-effects+ ambiance+ ambience+ ambient analog+ analogique+ application+ astral+ atmosphere+ background+ background+ barely-audible+ bass+ cartoon+ cinematic+ creepy+ curieux+ dark+ deep+ delay+ drama+ echo+ eerie+ effect+ effects+ efx+ electro+ electronic+ extra+ extraterrestre+ fear+ feedback+ film+ filter+ floating+ futur+ future+ futuristic+ fx+ galaxy+ horror+ illbient+ low+ machine+ mfb+ monster+ movie+ mutant+ outhier+ ovni+ pad+ porte+ processed+ psychedelic+ pulse+ remix+ reverb+ scary+ sci-fi+ science-fiction+ score+ sky+ sound-design+ sound-effects+ sounddesign+ soundscape+ soundtrack+ space+ spacecraft+ spooky+ strange+ suspense+ synth+ synthesized+ synthesizer+ synthetiseur+ synthizer+ talking+ terror+ texture+ thrill+ time-machine+ tone+ trip+ ufo- vocoder+ voice+ voyage+ weird+

Sounds with these tags

alien ambient synth ufo

suonho\_ScaryScape\_01.wav ★★★★☆

stereo feedback drone for cinematic use  
alien ambience ambient background creepy dark drama drone effect fear feedback film filter fx

suonho  
Number 14th, 2008  
55326 downloads  
100 comments

Dystopian Future - FX So... ★★★★★

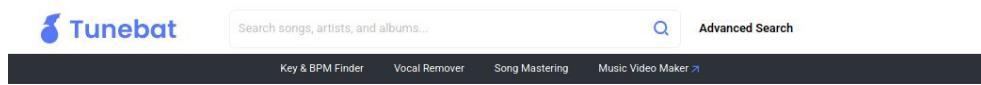
BurghRecords Presents The Dystopian Future - Sound Effects Pack Download link : <https://www.editedburghrecords.com/products/new-arrivals/dystopian-future/> The dystopian future effects pack is sampled ...  
atmospheric ambience alien spacecraft sound-effects ambient alien-sound-effects analog

ERH\_p1.2o2bb12\_14\_alien... ★★★★★

remix of "p1.2o2bb12\_14 alien sounds" added by ERH (see remix link above) slow down, edits, vocode, filter, reverb, delay, and gently compression

suonho  
August 9th, 2008  
4990 downloads  
13 comments

# Some examples of music and sound retrieval tasks



The image shows the Tunebat website. At the top is a header with the Tunebat logo, a search bar containing "Search songs, artists, and albums...", a magnifying glass icon, and an "Advanced Search" link. Below the header is a dark navigation bar with links for "Key & BPM Finder", "Vocal Remover", "Song Mastering", and "Music Video Maker".

## Key & BPM Database and Music Finder

Browse harmonic data and recommendations for over 70 million songs.



A search bar with a magnifying glass icon and the placeholder text "Search songs, artists, and albums...". To the right is a blue "Search" button.

### *Key & tempo estimation*

#### Top Tracks

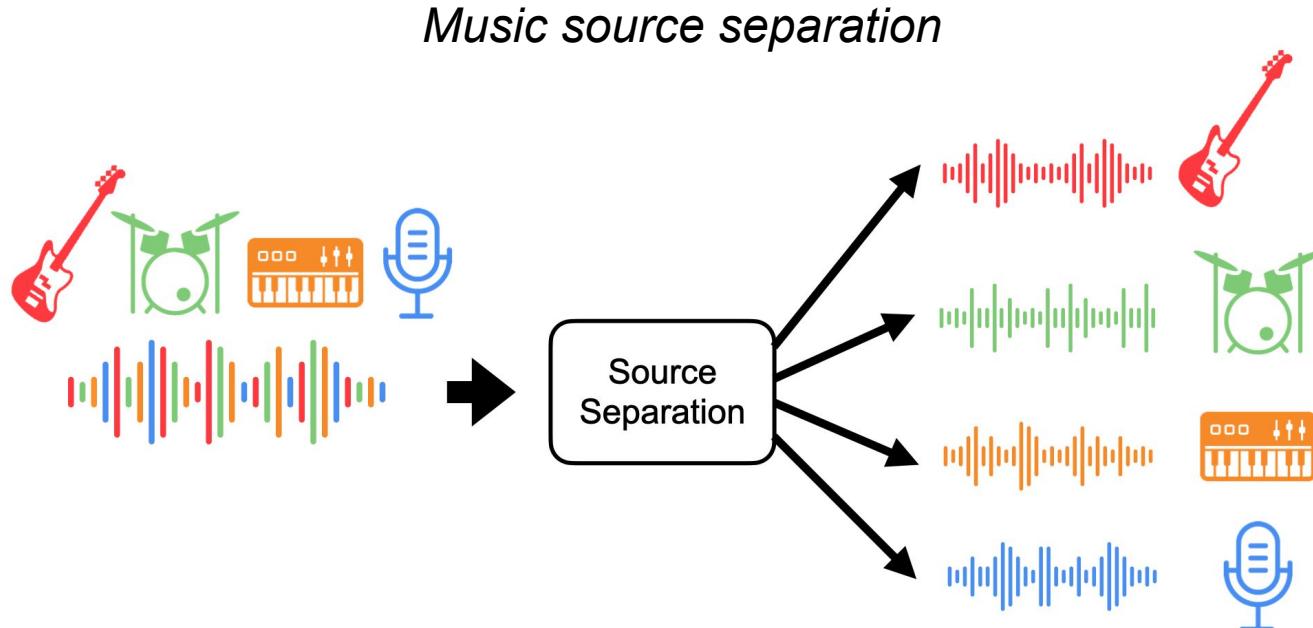


A table listing four top tracks with their details:

Image	Artist(s)	Title	Key	BPM	4B	Link
	BTS	Butter	A♭ Major Key	110 BPM	4B Camelot	
	Olivia Rodrigo	good 4 u	F# Minor Key	169 BPM	11A Camelot	
	Dua Lipa, DaBaby	Levitating (feat. DaBaby)	F# Minor Key	103 BPM	11A Camelot	
	Doja Cat, SZA	Kiss Me More (feat. SZA)	A♭ Major Key	111 BPM	4B Camelot	

9

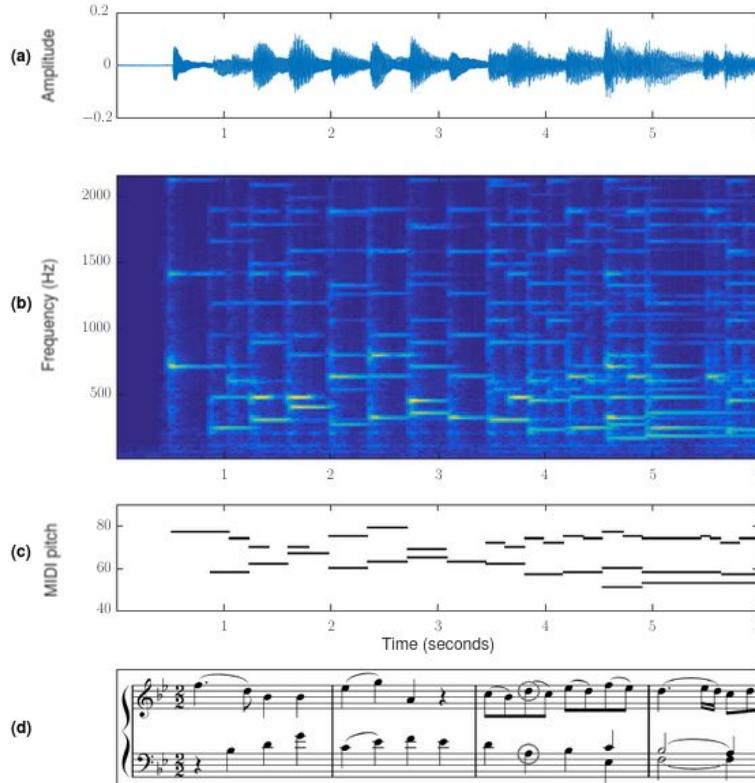
# Some examples of music and sound retrieval tasks



Source: <https://source-separation.github.io/tutorial>

# Some examples of music and sound retrieval tasks

## *Automatic music transcription*

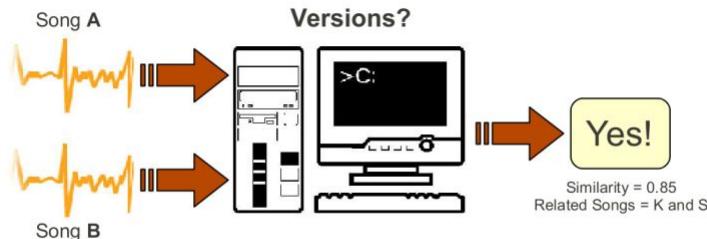


# Some examples of music and sound retrieval tasks

*Music identification, cover song identification*



Source: <https://acoustid.biz/>



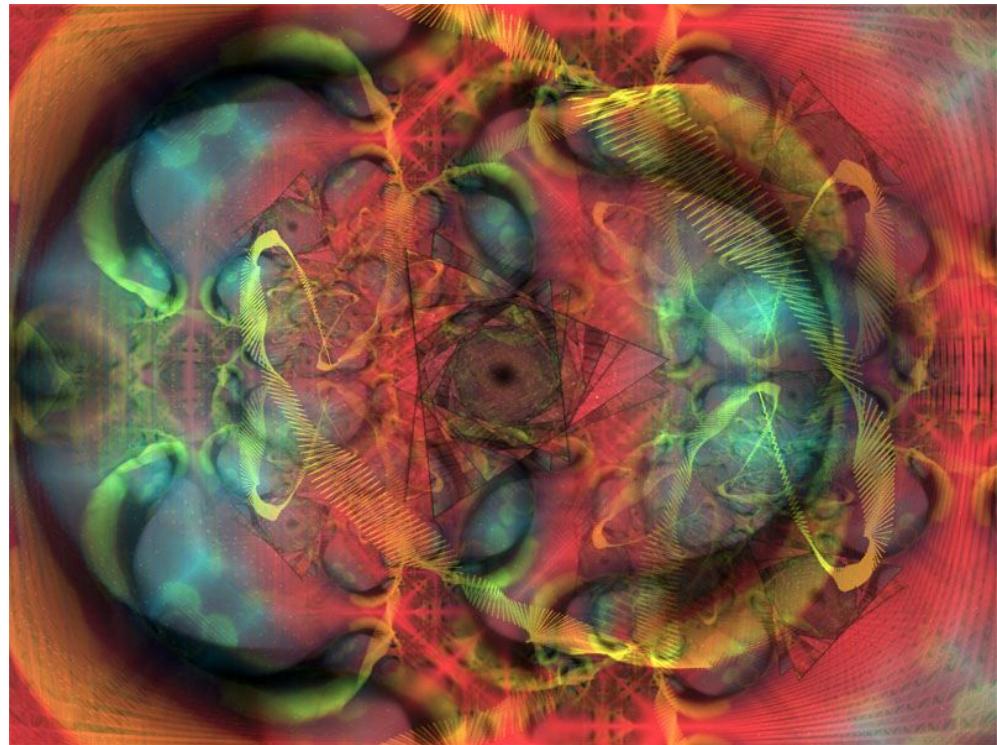
Furkan et al. 2020. Version identification in the 20s.

# Some examples of music and sound retrieval tasks

*Music/sound production*



*Music visualization*



# Extracting audio features

Many possibilities for applications operating with music/sound features retrieved from audio. Extracting these 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

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

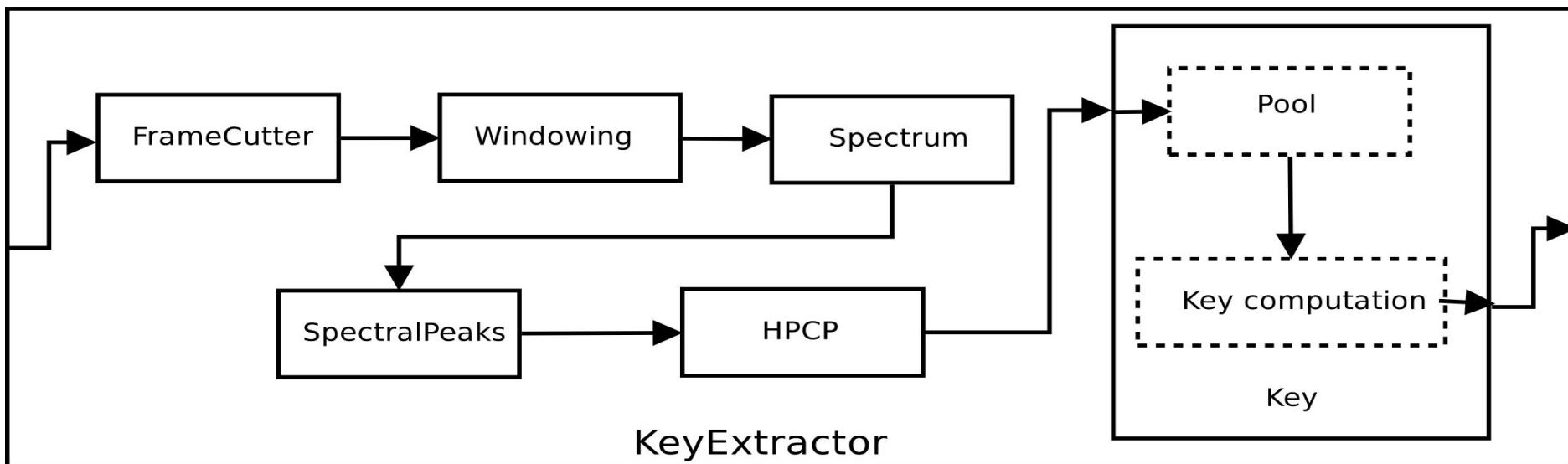
# Essentia

[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



<b>Similarity</b>	<b>Classification</b>	<b>Deep learning inference</b>	<b>Mood detection</b>
Analyze audio and compute features to find similar sounds or music tracks.	Classify sounds or music based on computed audio features.	Use data-driven TensorFlow models for a wide range applications from music annotation to synthesis.	Find if a song is happy, sad, aggressive or relaxed.
<b>Key detection</b>	<b>Onset detection</b>	<b>Segmentation</b>	<b>Beat tracking</b>
Find a key of a music piece.	Detect onsets (and transients) in an audio signal.	Split audio into homogeneous segments that sound alike.	Estimate beat positions and tempo (BPM) of a song.
<b>Melody extraction</b>	<b>Audio fingerprinting</b>	<b>Cover song detection</b>	<b>Spectral analysis</b>
Estimate pitch in monophonic and polyphonic audio.	Extract fingerprints from any audio source using the Chromaprint algorithm.	Identify covers and different versions of the same music piece.	Analyze spectral shape of an audio signal.
<b>Loudness metering</b>	<b>Audio problems detection</b>	<b>Voice analysis</b>	<b>Synthesis</b>
Use various loudness meters including algorithms compliant with the EBU R128 broadcasting standard.	Identify possible audio quality problems in music recordings.	Voice activity detection and characterization.	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



C++ library

<https://essentia.upf.edu>

A screenshot of the Essentia official website. The header includes the "ESSENTIA" logo and navigation links: Documentation, Algorithms reference, Demos, Applications, News, License, Download, Github, and Search. The main content area has a teal background with the title "Essentia" and a subtitle "Open-source library and tools for audio and music analysis, description and synthesis". It features three buttons: "GET STARTED" (red), "DOWNLOAD" (white), and "DEMONS" (white). Below this are four sections with icons: "Extensive collection of reusable algorithms" (calculator icon), "Cross-platform" (globe icon), "Fast prototyping" (code editor icon), and "Industrial applications" (factory icon).



An open-source JavaScript library,  
powered by Essentia WebAssembly

<https://essentia.upf.edu/essentiajs>

Extensive collection of  
reusable algorithms  
Flexible and easily extensible  
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.  
Optimized for computational  
speed, including real-time use  
cases.



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

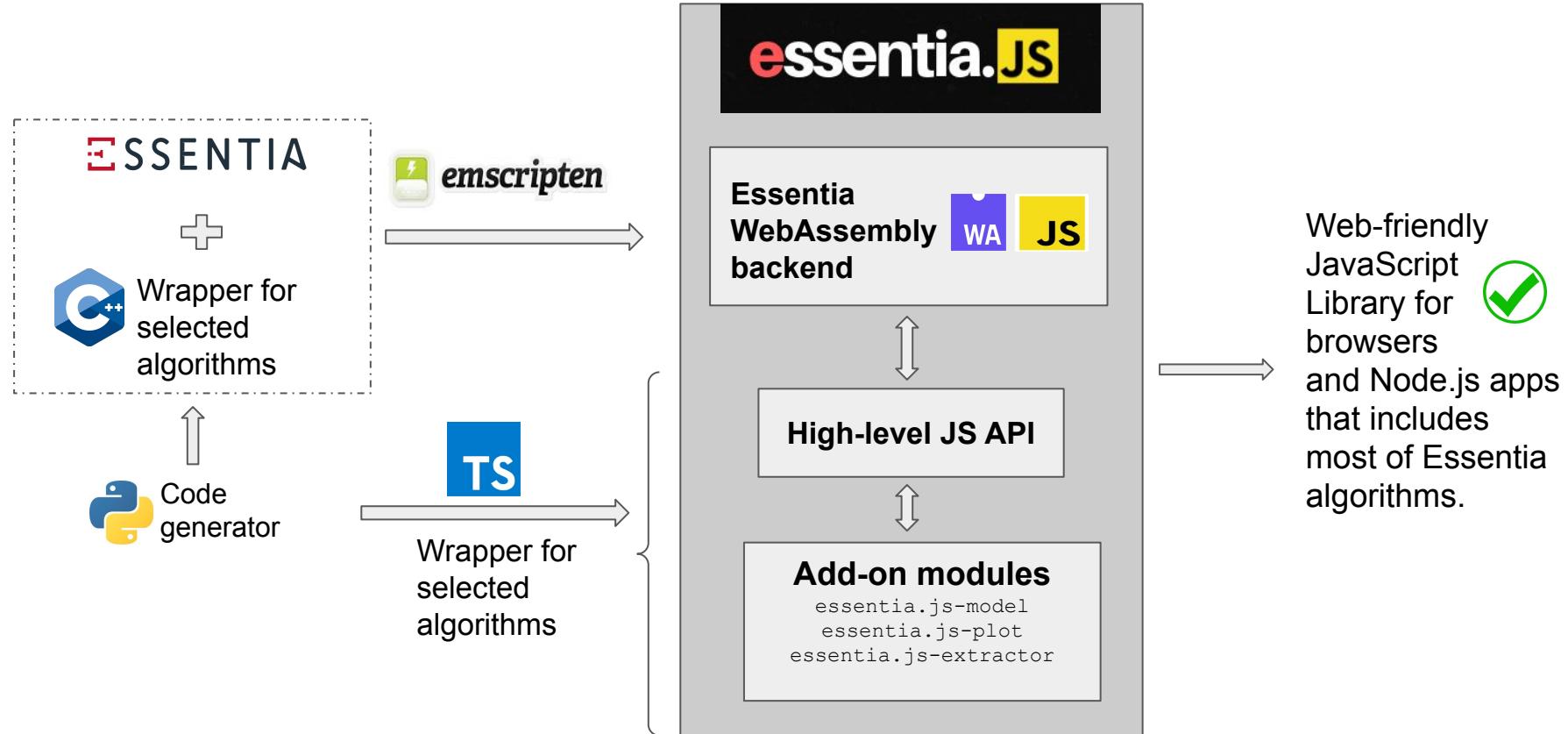
### Essentia JS/TS API

#### 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  
}



<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, backward compatibility is not guaranteed and not fully production-ready.

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

```
<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>
```

Instantiate Essentia.js with WASM backend

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

Get audio

Fetch file and decode

Request live microphone stream \*

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

Use Essentia.js algorithms

```
// 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 synchronously

# Hands-on session

Go to: <https://glitch.com/@jmarcosfer/wac-21-essentiajs-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>

# Q/A Session

# 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](#)

# Break time

**ESSENTIA**



Universitat  
Pompeu Fabra  
Barcelona

**MTG**  
Music Technology  
Group

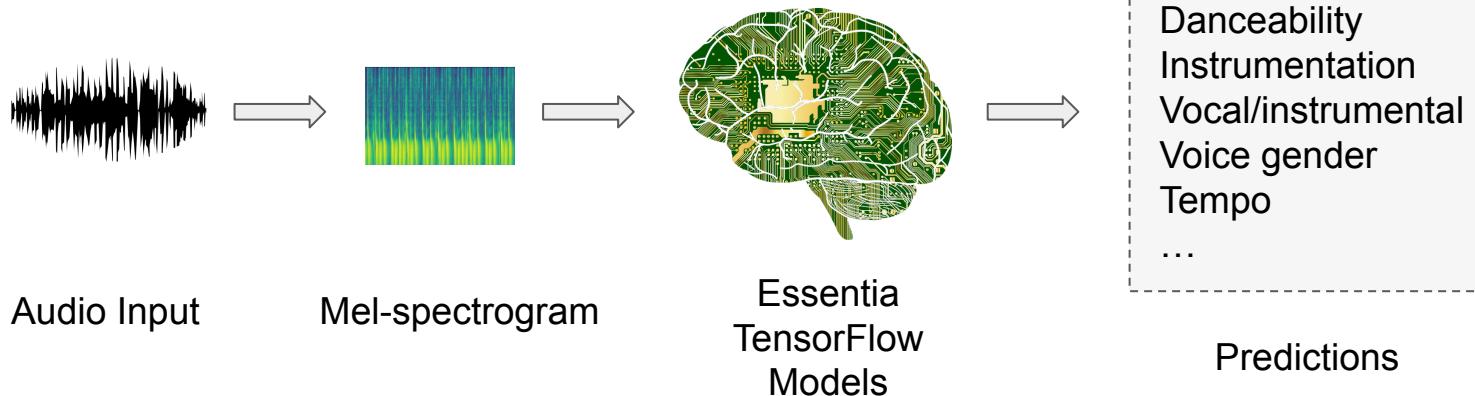
# Machine Learning on the Web



etc..

# Essentia TensorFlow models

A repository of publicly available deep learning models for music analysis tasks  
[https://essentia.upf.edu/machine\\_learning.html](https://essentia.upf.edu/machine_learning.html)



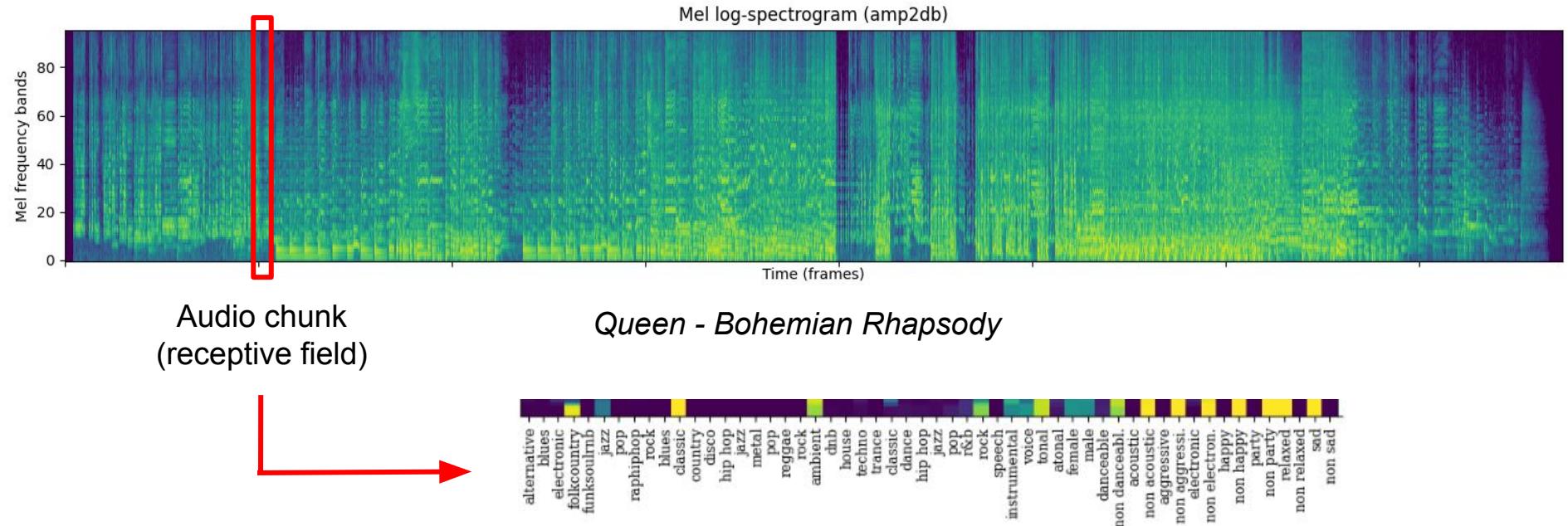
Now, we ported these optimized models to Tensorflow.js format for its usage on JS.

# Auto-tagging and classification



*Queen - Bohemian Rhapsody*

# Auto-tagging and classification

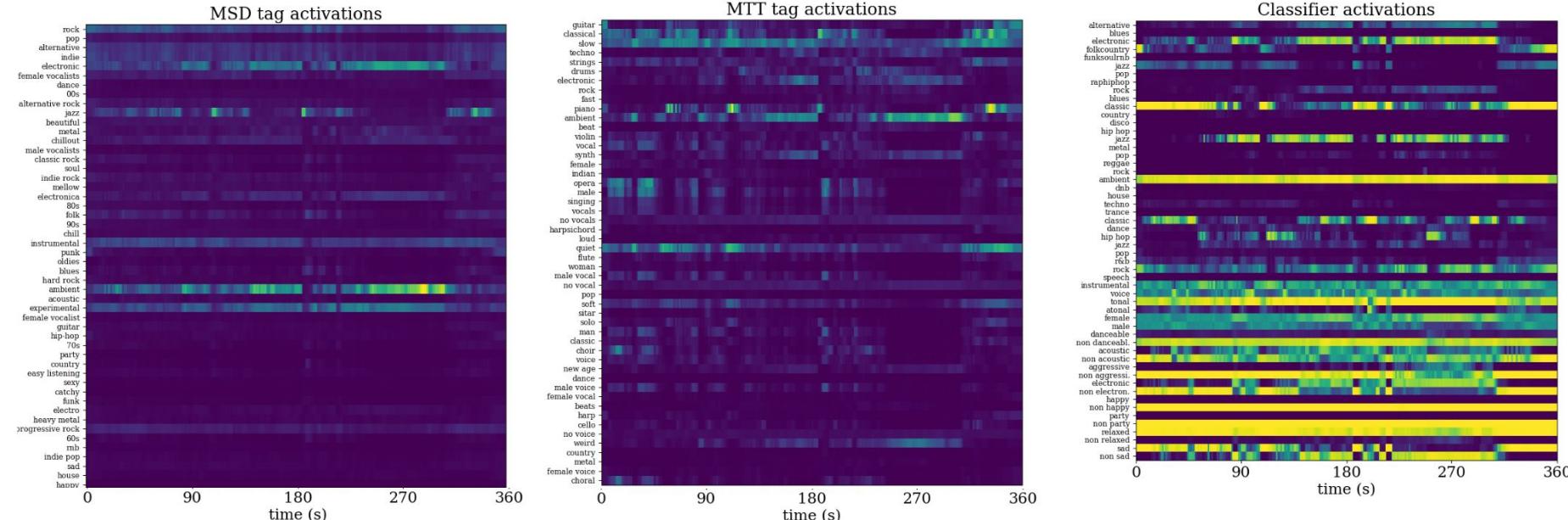


## Audio chunk (receptive field)

## *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

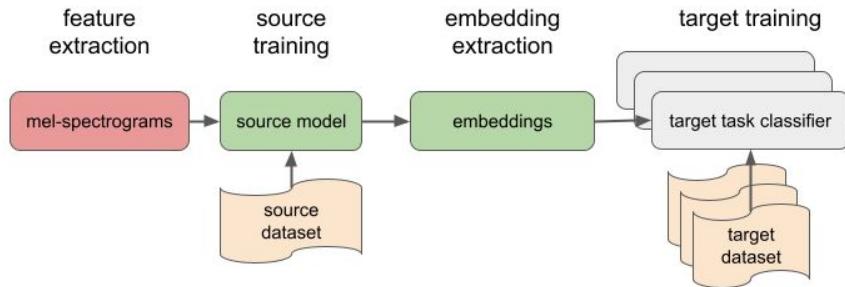
RF = receptive field

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, funksoulrbn, 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
misc.	sad	sad, non sad
	danceability	danceable, non danceable
	voice/instrum.	voice, instrumental
	gender	male, female
	tonal/atonal	tonal, atonal
fs-loop-ds	urbansound8k	air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun shot, jackhammer, siren, street music
		bass, chords, fx, melody, percussion

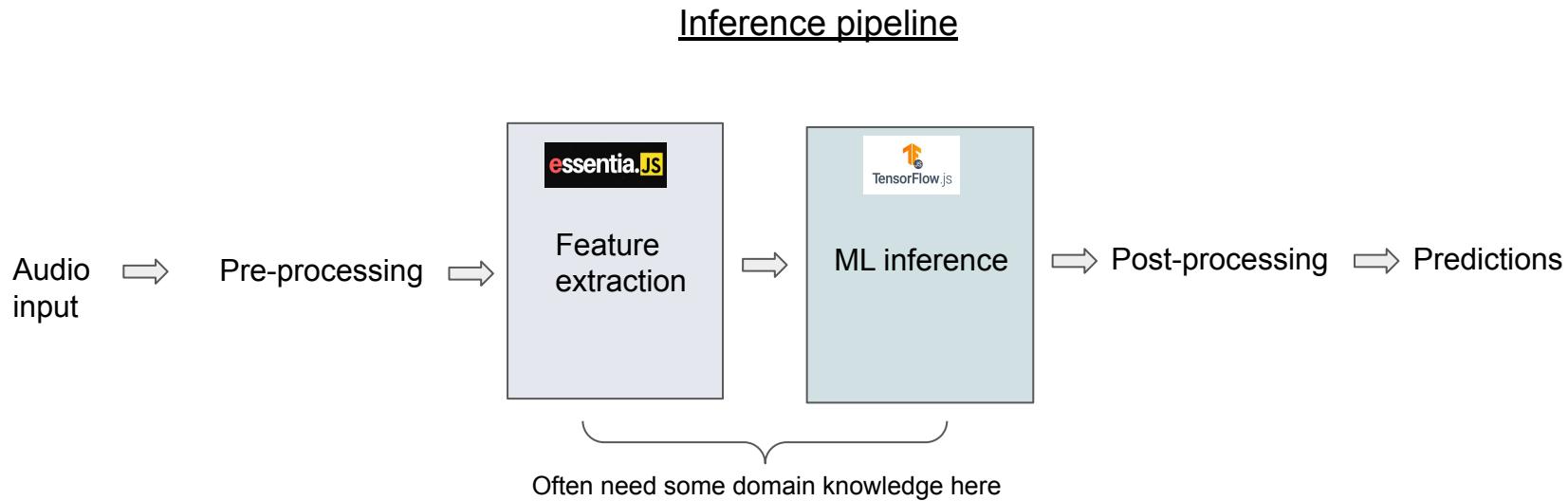
# Essentia TensorFlow Models downloads

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

[https://essentia.upf.edu/machine\\_learning.html](https://essentia.upf.edu/machine_learning.html)

Models available under [the CC BY-NC-ND 4.0 license](#)

# Essentia.js + TensorFlow.js

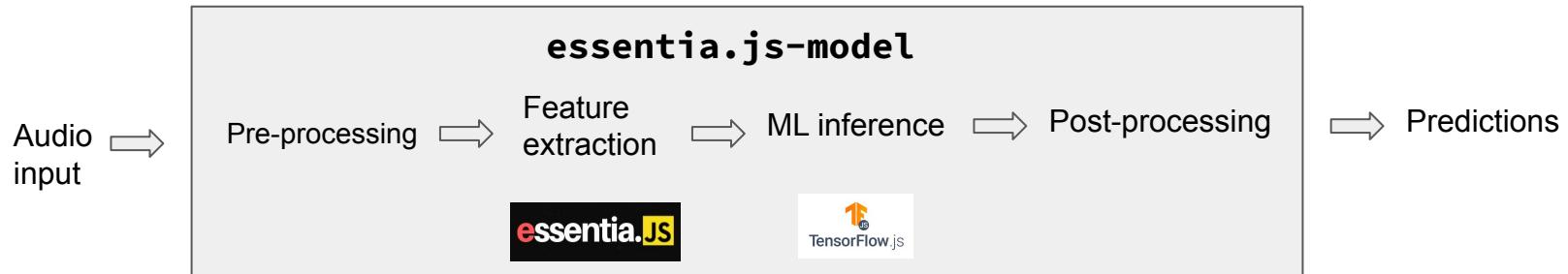


**But how to make this more accessible and easy to use for everyone?**

# 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



[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 configured 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();

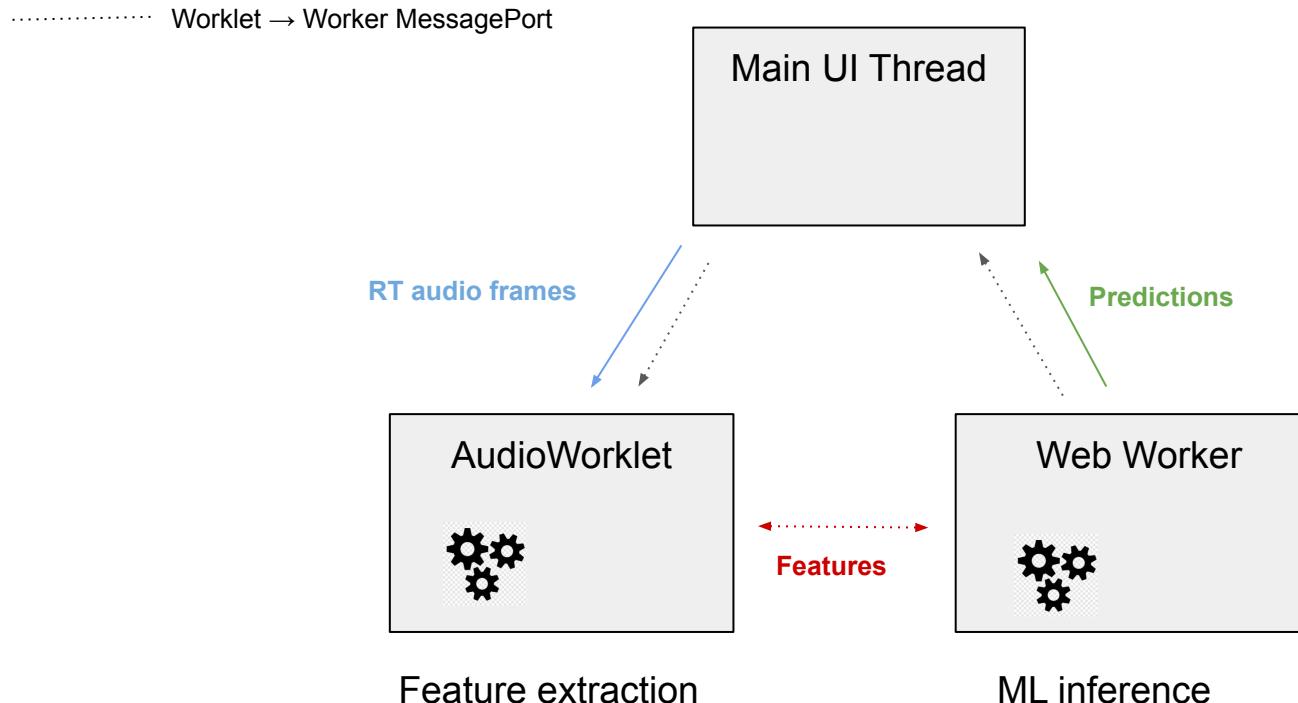
// Run inference for the given feature input
let predictions = await musiCNN.predict(featureInput, true);

// 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>
- Realtime: <https://glitch.com/~essentiajs-models-rt>

# Real-time implementation



# Industrial application

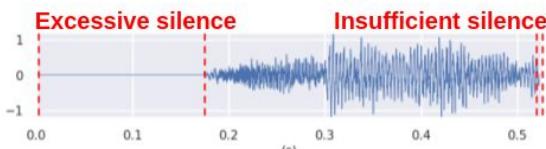


## Audio Problems

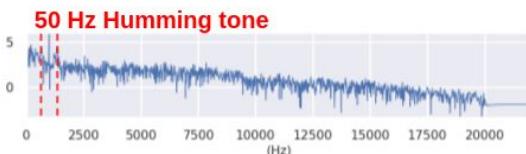
Example: <https://glitch.com/~audio-problems-detection>

### Audio Problems

#### Incorrect margins



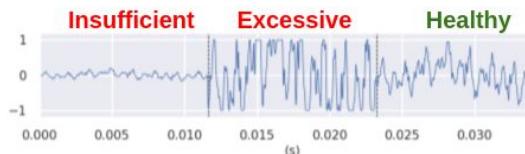
#### Low frequency humming



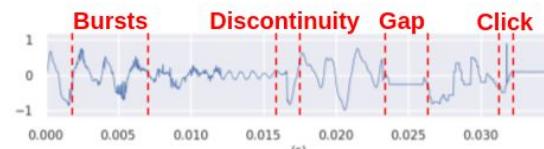
#### Phase/Stereo problems



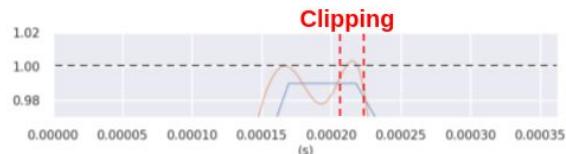
#### Loudness problems



#### Audio artifacts



#### True-Peak 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.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/essentia_custom_extractor.cpp)

[https://github.com/MTG/essentia.js/blob/master/src/cpp/custom/bindings\\_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

```
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**



Universitat  
Pompeu Fabra  
Barcelona

**MTG**  
Music Technology  
Group