PALA - Python Audio Loudness Analysis

HBB-ThinkTank

February 2025

Contents

1 Introduction

PALA (Python Audio Loudness Analysis) is a Python package designed for loudness analysis of audio files. It aims to compute LUFS values, as well as other relevant loudness metrics such as RMS, Peak, True Peak, LRA, DR, and frequency distributions.

2 Motivation and Objectives

The primary goal of PALA is to provide a comprehensive analysis tool for measuring audio loudness within Python, integrating various measurement methods into a single package.

3 Planned Metrics

PALA is designed to compute the following loudness metrics:

- **LUFS (Loudness Units Full Scale)**
  - Momentary LUFS (M) - computed over a 400-ms window
  - Short-Term LUFS (S) - computed over a 3-second window
  - Integrated LUFS (I) - measured over the entire file
- **Other Loudness Metrics**
  - RMS (Root Mean Square)
  - Peak (Sample Peak Level, dBFS)
  - True Peak (Interpolated Peak Value)
  - LRA (Loudness Range, LU)
  - PLR (Peak-to-Loudness Ratio, dB)
  - DR (Dynamic Range, dB)
  - Frequency Distribution Analysis

4 Technical Implementation

PALA follows a modular structure with the following core components:

```
pala/
|--- __init__.py
|--- analysis.py  # Main analysis functions
|--- lufs.py  # LUFS calculations
|--- dynamics.py  # RMS, Peak, True Peak, PLR, DR
|--- frequency.py  # Frequency analysis
|--- utils.py  # Utility functions
|--- io.py  # File handling

tests/
|--- test_analysis.py
|--- test_lufs.py
|--- test_dynamics.py
|--- test_frequency.py
```

5 Available Functions in dynamics.py

The ‘dynamics.py‘ module provides various functions for analyzing dynamic properties of audio files. These functions are available in two forms: one using a waveform array as input and another accepting a file path as input.

5.1 Functions using waveform as input

- **compute_rms_aes(waveform)** - Computes RMS level using the AES standard.
- **compute_rms_itu(waveform)** - Computes RMS level using the ITU-R BS.1770 standard.[1]
- **compute_peak(waveform)** - Determines the sample peak level in dBFS.
- **compute_true_peak(waveform)** - Computes the true peak value using oversampling.
- **compute_dynamics(waveform, norm)** - Computes various dynamic range metrics, such as DR, Crest Factor, and Headroom.

5.2 Functions using file path as input

- **compute_rms_aes_load(filepath)** - Computes AES RMS level from an audio file.
- **compute_rms_itu_load(filepath)** - Computes ITU RMS level from an audio file.
- **compute_peak_load(filepath)** - Determines the sample peak level from an audio file.
- **compute_true_peak_load(filepath)** - Computes the true peak level from an audio file.
- **compute_dynamics_load(filepath)** - Computes dynamic range metrics from an audio file.

5.3 Function Aliases

For convenience, the following alternative function names (aliases) are provided:

- **rmsaes** → compute_rms_aes
- **rmsitu** → compute_rms_itu
- **peak** → compute_peak

[1] The accuracy of this function has not been fully tested due to the lack of reference audio files and validated implementations.

- **true_peak** → compute_true_peak
- **dynamics** → compute_dynamics
- **lrmsaes** → compute_rms_aes_load
- **lrmsitu** → compute_rms_itu_load
- **lpeak** → compute_peak_load
- **ltrue_peak** → compute_true_peak_load
- **ldynamics** → compute_dynamics_load

Additionally, pre-configured versions for specific norms are available:

- **dynaes** → compute_dynamics(norm='aes')
- **dynitu** → compute_dynamics(norm='itu')
- **ldynaes** → compute_dynamics_load(norm='aes')
- **ldynitu** → compute_dynamics_load(norm='itu')

6 Data Storage Format

PALA uses JSON as the primary format for storing analysis results:

```
{
 "filename": "track1.wav",
 "lufs": {
 "momentary": [-18.5, -17.8, ...],
 "short_term": [-17.0, -16.5, ...],
 "integrated": -16.0
 },
 "rms": -15.5,
 "true_peak": -2.3,
 "plr": 13.7,
 "dr": 10.2,
 "lra": 5.8,
 "frequency_analysis": {
 "low": -12.3,
 "mid": -10.5,
 "high": -9.2
 }
}
```

Other formats such as CSV and YAML might also be supported in future updates.

7 Integration of Equal-Loudness Contours

Fletcher-Munson curves are used as references for frequency weighting, enabling calculation of LUFS values for specific frequency bands.

8 Licensing and Open-Source Strategy

PALA is released under the MIT License, encouraging open-source contributions and public use.

9 Publication Strategy

9.1 GitHub Repository

The project is hosted on GitHub under HBB-ThinkTank/PALA. The repository contains:

- `.gitignore`
- `pala/__init__.py`
- `pala/analysis.py`
- `pyproject.toml`
- `setup.cfg`

Versioning is managed using Git tags.

9.2 PyPI Package Deployment

To build and upload the package:

```
python -m build
python -m pip cache purge
rmdir /s /q build dist pala.egg-info

# Upload to PyPI
twine upload --repository pala dist/*
```

10 Future Steps

- Implement LUFS calculations.
- Conduct tests with real audio files.
- Finalize the first stable release (0.1.0).

- Integrate additional metrics such as PLR, DR, and LRA.
- Incorporate equal-loudness contours.
- Publish the package on PyPI and engage with the open-source community.