Ladislav Vašina

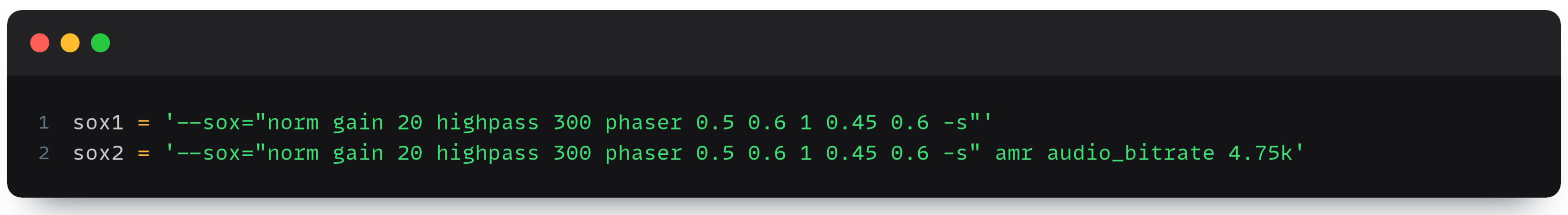
Data augmentation integration into PyTorch

supervisor: Ing. Igor Szőke, Ph.D.

Excel@FIT 2024

Objectives

- Integrate various audio augmentation tools into one, so it can be easily used with PyTorch.
- Design a simple interface for users to apply augmentations.



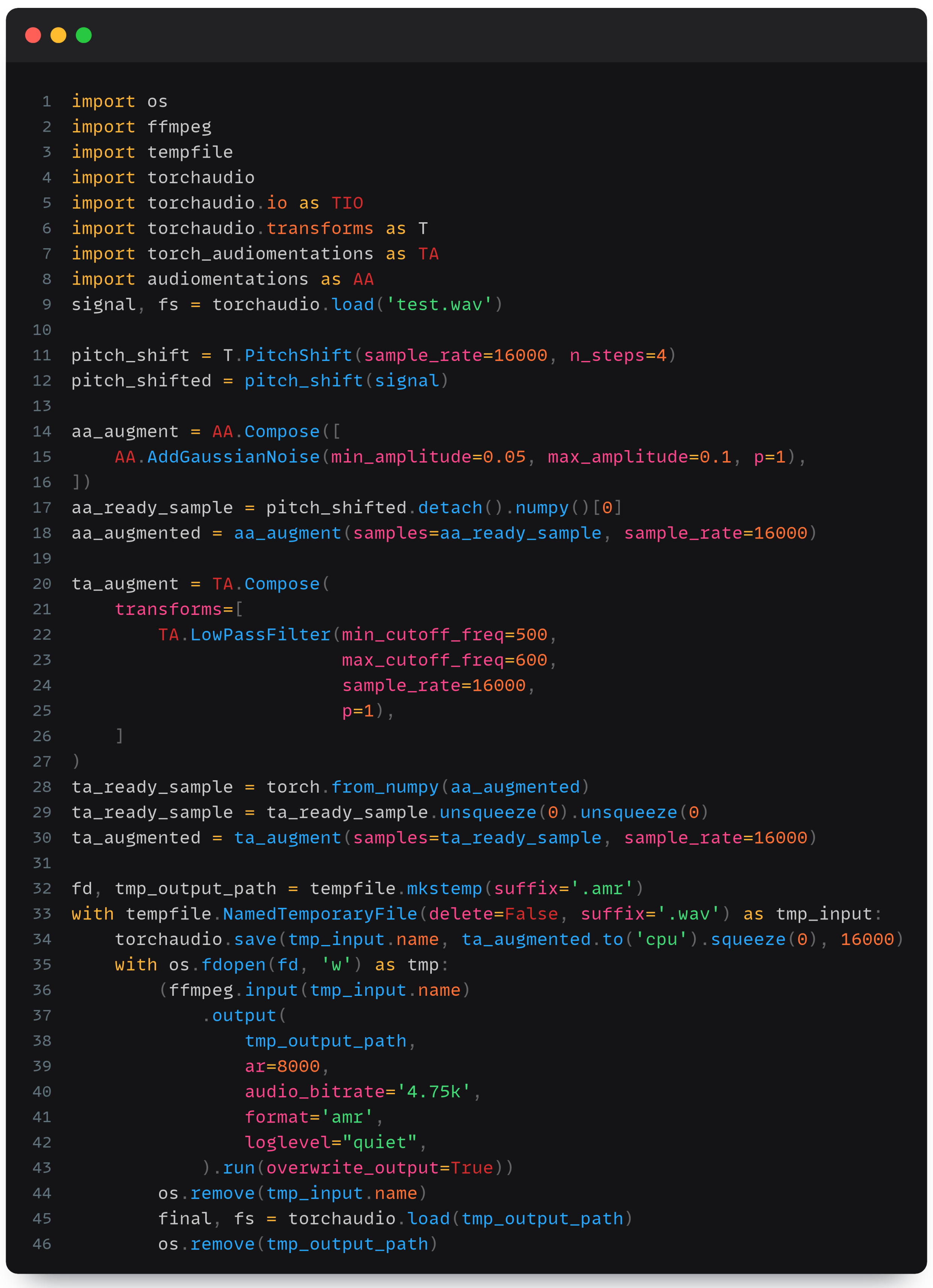
```
sox1 = '--sox="norm gain 20 highpass 300 phaser 0.5 0.6 1 0.45 0.6 -s"'
sox2 = '--sox="norm gain 20 highpass 300 phaser 0.5 0.6 1 0.45 0.6 -s" amr audio_bitrate 4.75k'
```

Fig 1 – SoX command used for the augmentation

Results

- Python library **AudioAugmentor** which provides a simpler interface over the multiple audio augmentation tools.
- Reduced complexity while defining augmentations from different frameworks — You only need one library.
- Augment audio with classes that are usable with PyTorch's DataLoader, standalone waveform or with a local directory of recordings.

AudioAugmentor ❌



```
import os
import ffmpeg
import tempfile
import torchaudio
import torchaudio.io as TIO
import torchaudio.transforms as T
import torch_audiomentations as TA
import audiomentations as AA
signal, fs = torchaudio.load('test.wav')

pitch_shift = T.PitchShift(sample_rate=16000, n_steps=4)
pitch_shifted = pitch_shift(signal)

aa_augment = AA.Compose([
    AA.AddGaussianNoise(min_amplitude=0.05, max_amplitude=0.1, p=1),
])
aa_ready_sample = pitch_shifted.detach().numpy()[0]
aa_augmented = aa_augment(samples=aa_ready_sample, sample_rate=16000)

ta_augment = TA.Compose(
    transforms=[
        TA.LowPassFilter(min_cutoff_freq=500,
                         max_cutoff_freq=600,
                         sample_rate=16000,
                         p=1),
    ]
)
ta_ready_sample = torch.from_numpy(aa_augmented)
ta_ready_sample = ta_ready_sample.unsqueeze(0).unsqueeze(0)
ta_augmented = ta_augment(samples=ta_ready_sample, sample_rate=16000)

fd, tmp_output_path = tempfile.mkstemp(suffix='.amr')
with tempfile.NamedTemporaryFile(delete=False, suffix='.wav') as tmp_input:
    torchaudio.save(tmp_input.name, ta_augmented.to('cpu').squeeze(0), 16000)
    with os.fdopen(fd, 'w') as tmp:
        (ffmpeg.input(tmp_input.name)
            .output(
                tmp_output_path,
                ar=8000,
                audio_bitrate='4.75k',
                format='amr',
                loglevel="quiet",
            ).run(overwrite_output=True))
        os.remove(tmp_input.name)
        final, fs = torchaudio.load(tmp_output_path)
        os.remove(tmp_output_path)
```

Fig 3 – Application of various augmentations without AudioAugmentor

Implementation

- Integrated different augmentations from **torchaudio**, **audiomentations**, **torch-audiomentations**, **pyroomacoustics**, **ffmpeg-python** libraries.
- Handling of the miscellaneous properties and interfaces of the integrated libraries.
- Enabling easy usage of SoX (Sound eXchange) commands to augment audio data.
- Created random room generator so user can make the recording sounds like it's in a different room.

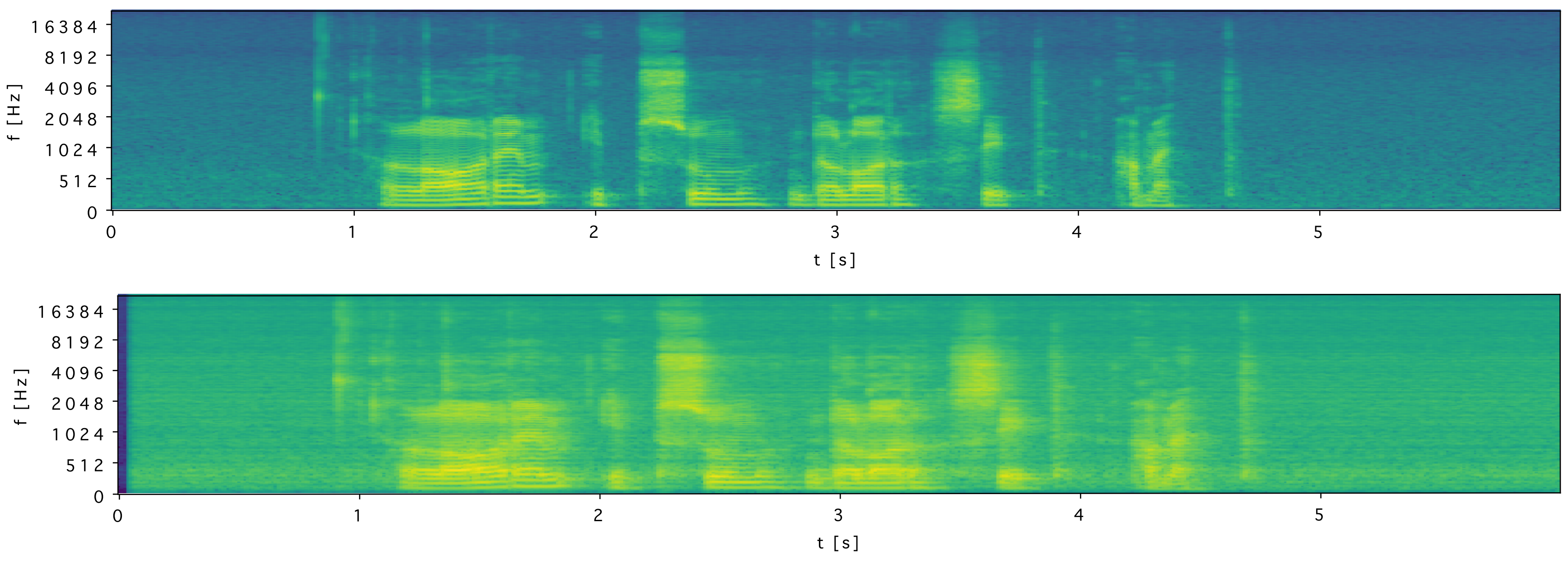
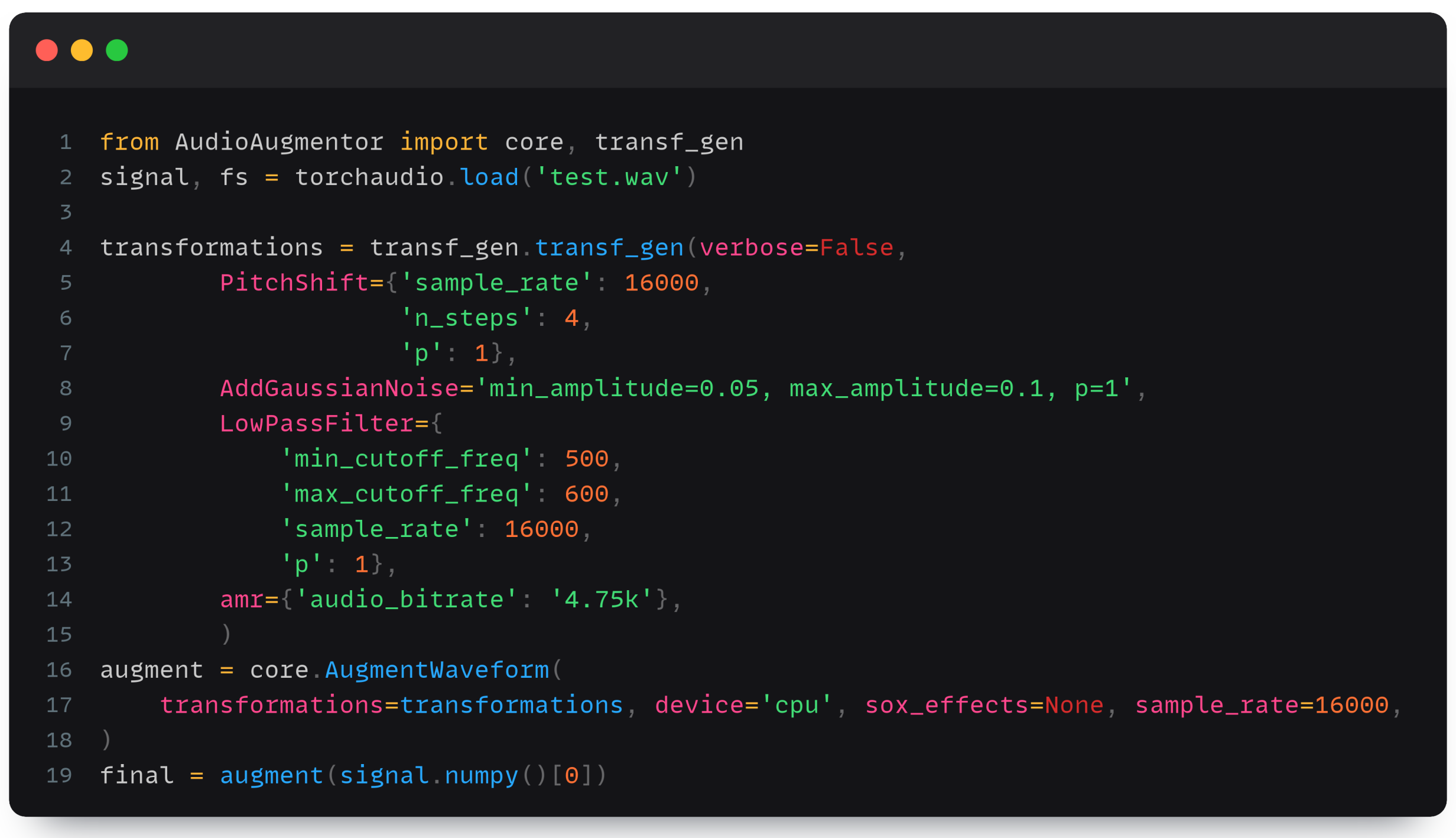


Fig 2 – Mel-Spectrograms of recording before (top) and after (bottom) applying room impulse response

AudioAugmentor ✅



```
from AudioAugmentor import core, transf_gen
signal, fs = torchaudio.load('test.wav')

transformations = transf_gen.transf_gen(verbose=False,
        PitchShift={'sample_rate': 16000,
                    'n_steps': 4,
                    'p': 1},
        AddGaussianNoise='min_amplitude=0.05, max_amplitude=0.1, p=1',
        LowPassFilter={
            'min_cutoff_freq': 500,
            'max_cutoff_freq': 600,
            'sample_rate': 16000,
            'p': 1},
        amr={'audio_bitrate': '4.75k'},
        )
augment = core.AugmentWaveform(
    transformations=transformations, device='cpu', sox_effects=None, sample_rate=16000,
)
final = augment(signal.numpy()[0])
```

Fig 4 – Application of various augmentations with AudioAugmentor

PyPi

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY