BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2024
http://excel.fit.vutbr.cz

Data augmentation integration into PyTorch

Ladislav Vašina*

Abstract
Using audio augmentations which are by default part of the machine-learning library PyTorch can sometimes be insufficient because it contains just a relatively low number of mostly low complex augmentations. At that point, researchers need to look for publicly available tools for audio augmentation. Various of these tools are available, but each of them has distinct properties and can work with different types of data, have different interfaces, or work with various devices on which it can compute its augmentations. It means that to obtain a larger pool of different augmentations, researchers must use multiple of these tools in conjunction. This paper presents a tool that creates a unified and simple interface on top of the selected audio augmentation libraries and tools that can be used in conjunction with the PyTorch library. Python library called AudioAugmentor is created and published to PyPi, so that every researcher can easily and quickly start using it to augment their audio data. It allows easier augmentation of PyTorch's DataLoader, standalone audio recordings, or augmentation of the local datasets via just one interface with a wide range of audio augmentations.

*xvasin11@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Training of the artificial intelligence models needs a lot of data for their creation. In order to create robust models that enable high-quality speech recognition, not just in the laboratory environment, a diverse training dataset is needed. A common practice for gaining this diversity is data augmentation. The number of available audio augmentations in the most widely used machine-learning framework PyTorch is low and it does not contain more complex audio augmentations.

When you want to apply augmentations that are not included within the PyTorch library, you need to look for publicly available tools that can provide desired augmentations. There are multiple of these tools, but each of them has different properties so in order to use a wider range of audio augmentations, you need to use multiple of these tools and handle the differences between them. Precisely the differences can be, that some tools work with different data types, some can compute augmentations on CPU, some can compute them on GPU, and mainly each tool has a various pool of available audio augmentations and distinct interfaces. So in order to apply a larger range of various audio augmentations, you need to learn how to use multiple augmentation tools and while using them in combination, solve different characteristics of each of them.

At this moment the most integrated solution for audio augmentation within the PyTorch framework is its own library **torchaudio**[1]. It works great with tensors, can apply augmentations on GPU, and has SpecAugment [1] implemented, which is a modern approach to the Mel [2] spectrogram augmentation. The next existing library is **audiomentations**[2]. It provides the largest range of augmentations, but can only work on CPU and does not provide complex room simulation options or the option to apply various audio codecs. **Torch-audiomentations**[3] is the complementary library to audiomentations, which allows application of the augmentations on GPU, but it contains a lower number of the augmentations. For simulation of room impulse responses, there is a complex library **pyroomacoustics**[4], which allows the creation of various rooms by user definition. It also contains large lists of materials with different absorption coefficients,

[1]https://pytorch.org/audio/stable/index.html
[2]https://github.com/iver56/audiomentations
[3]https://github.com/asteroid-team/torch-audiomentations
[4]https://github.com/LCAV/pyroomacoustics

that can be used as floors, walls, or ceilings during room simulation. You can also place multiple speakers within the simulated room. Audio codecs can be applied using **ffmpeg-python**[5] library which provides a simple python interface over the FFmpeg[6] tool.

I have proposed a library that provides several classes and functions which are used to augment either PyTorch's DataLoader, standalone audio recordings, or local audio datasets. The user passes the augmentations (via their general names) and their parameters to the function, that handles the user's input and it returns a list of augmentations, where each augmentation is chosen from a specific library or tool that is the most optimal for that augmentation. I have also added an option that allows users to pass a range, from which parameter value will be randomly chosen, to torchaudio augmentations, which was not possible before. Users can also choose to define augmentations via pseudo-SoX command, which is parsed and handled using the torchaudio library. Room simulation is handled using the pyroomacoustics library and users can generate random rooms just by defining the number of vertices and the minimal and maximal lengths (in meters) of the X and Y axes within which the room is going to be generated.

This created library was named AudioAugmentor and was published on PyPi. AudioAugmentor provides a simple interface, which users can easily use to augment audio data while using various approaches. AudioAugmengtor's interface unites interfaces of multiple audio augmentation tools and libraries while maintaining their core properties and it also accepts augmentations definition via pseudo-SoX commands. This makes the augmentation of audio data an easy process, that does not require handling of different interfaces and specifics of various augmentation tools.

2. Poster description

Figure 1 shows one of the options for how the user can specify which augmentations should be applied. This option enables the usage of pseudo-SoX command while initializing the classes that handle the augmentations. Pseudo-SoX command saved to variable `sox1` shows the format of the command and means that the effects of normalization, a gain of 20 dB, a high-pass filter with a cutoff frequency of 300 Hz and phaser will be applied to the audio. Command saved to variable `sox2` shows the extension of this pseudo-Sox command, which allows also specification of the audio codec, that should be applied after the application of SoX effects.

Figure 2 shows two Mel spectrograms, where the top is without any augmentation and the bottom one shows an application of the convolution with a generated room impulse response which is internally generated using the pyroomacoustics library.

Figure 3 shows a code snippet, that tries to apply various audio augmentations from different augmentation libraries. You can see that in order to apply all the desired augmentations you have to use multiple library interfaces and handle the conversion to correct data types, data shapes, and other specifics of each library. This problem is solved by using the proposed AudioAugmentor library. Figure 4 shows the usage of this library and you can see right away that in order to apply the same audio augmentations as in Figure 3 you can simply use just one interface, which you can use to specify all the augmentations and apply them directly on your audio recordings.

3. Conclusions

Python library AudioAugmentor is proposed to solve the problem of a low number of audio augmentations within the PyTorch framework. It also solves the differences between various augmentation tools so they are easily usable though one simple interface. AudioAugmentor provides several classes and functions, that can be used to augment not just PyTorch's DataLoader, but also standalone recordings, or local audio datasets. AudioAugmentor is published on the PyPi platform so that everyone can easily download it and start using it straight away. It contains documentation of the usage and examples presented via Google Colab notebook. AudioAugmentor is tested positively for the ability to recreate experiments. It has also been tested on the task of fine-tuning the ASR system Whisper.

Acknowledgements

I would like to thank my supervisor Ing. Igor Szőke, Ph.D. for his help.

[5] https://github.com/kkroening/ffmpeg-python
[6] https://ffmpeg.org/

References

[1] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. SpecAugment: A simple data augmentation method for automatic speech recognition. In *Interspeech 2019*. ISCA, 9 2019.

[2] John C. Steinberg. Positions of Stimulation in the Cochlea by Pure Tones. *The Journal of the Acoustical Society of America*, 8(3):176–180, 01 1937.