The Journal of Open Source Software

Pydiogment: A Python package for audio augmentation

Ayoub Malek[1] **and Hasna Marwa Malek**[2]

1 Technical University of Munich **2** Grenoble Institute of Technology (Grenoble INP)

DOI: 00.00000/joss.00000

Software

- Review
- Repository
- Archive

Submitted: 25 February 2020
Published: 25 February 2020

Summary

Audio data augmentation is a key step in training Machine Learnign (ML) models to solve audio classification tasks. It is applied to increase the quality and size of the labeled training data-set, in order to improve the recognition accuracy. Data augmentation is simply a deformation technique, that helps stretch the data, and increase its size for a better training. Unlike image augmentation, audio augmentation is still limit explored by research and most deformation strategies manipulate the computed spectorams rather than the raw audio. With the exception of few libraries constrained to work with Pytorch (Paszke et al., 2019), most existing tools in this context either act on spectograms such as Google's Specaugment (Park et al., 2019), or are developed for music data augmentation like muda (McFee, Humphrey, & Bello, 2015). This paper describes version 0.1.0 of `Pydiogment`: a Python package for audio augmentation based on the scipy (Virtanen et al., 2019) and ffmpeg (FFmpeg Developers, 2019) libraries. `Pydiogment` implements various augmentation techniques that can be used to improve the accuracy of various recognition tasks (speaker recognition, spoken emotions recognition, speech recognition etc.) and avoid over-fitting when training models. The paper provides a brief overview of the library's functionality, along with a small emotions recognition experiment displaying the utility of the library.

Implementation and theory

`Pydiogment` includes 3 general categories of deformations / augmentations:

Amplitude based augmentations (`auga.py`)

- **Apply Gain:** This deformation can be described as an amplification of the signal and the noise by applying a given gain (in dB) to the input signal. Note that excessive gain application can result in clipping (Self et al., 2009).
- **Add Fade:** adds a fade-in and fade-out effects to the original signal. This is done by multiplying a hamming window with the original signal $y[n] = x[n] * w[n]$ where $x[n]$ is the original signal, $y[n]$ is the augmented signal and $w[n]$ is the computed hamming window (Poularikas, 1999).
- **Normalize:** Normalization refers to the practice of applying a uniform amount of gain across a signal, where signal-to-noise ratio and general dynamics levels remain unchanged (Shelvock, 2012). The normalization can be applied using the peak normalization method $y[n] = \frac{x[n]}{\max(x[n])}$ or the Root Mean Square (RMS) approach $y[n] = \sqrt{\frac{N.10^{(\frac{r}{20})}}{\sum_{n=0}^{N-1} x^2[n]}}.x[n]$ where $x[n]$ is the original signal, $y[n]$ is the augmented signal, N is the length of $x[n]$ and r is the input RMS level in dB.

- **Add Noise:** adds some random noise to the input signal based on a given signal to noise ratio (SNR).

Frequency based augmentation (`augf.py`)

- **Change tone:** changes the pitch of the audio (lowered or raised).
- **Convolve:** This is also called reverberating the audio and it consists of convolving the original signal with a given Room Impulse Response (RIR) to simulate an audio captured using far-field microphones in a different setup/channel $y[n] = x[n] * rir[n]$ where $x[n]$ is the original signal, $y[n]$ is the augmented signal and $rir[n]$ is the room impulse response (Raju, Panchapagesan, Liu, Mandal, & Strom, 2018).
- **Apply Filter:** apply various types of Butterworth filters (low-pass, high-pass or band-pass).

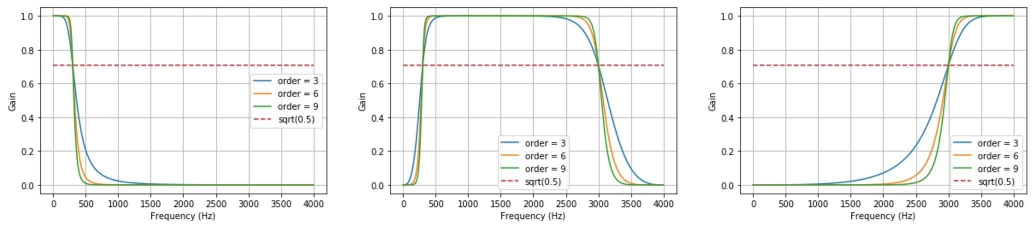

Figure 1: Frequency-Gain graphs of the implemented Butterworth filers (left: low-pass, middle: band-pass, right: high-pass) with a low cutoff frequency of 300 Hz and a high cutoff frequency of 3000 Hz.

Time based augmentation (`augt.py`)

- **Time Stretching:** slows down speeds up the original audio based on a given coefficient.
- **Time Shifting:** includes shifting the signal in a certain time direction or reversing the whole signal
- **Random Cropping:** generates a randomly cropped audio based on the original signal.
- **Eliminate Silence:** filters out silent frames from the input signal.
- **Resample:** resamples the the input signal given an input sampling rate.

It is very important to maintain the semantic validity when augmenting the data. *For example:* one cannot change tones when doing voice based gender classification and still expect tone to be a separating features of the predicted classes.

Experiment & Results

To prove the utility of `Pydiogment`, we display its effect on a spoken emotions recognition task. We use the **Emo-DB** data-set (Burkhardt, F., Paeschke, A., Rolfes, M., A., Walter F. Sendlmeier & Weiss, B., 2005) as a starting point, which is a small German audio data-set simulating 7 different emotions (neutral, sadness, anger, boredom, fear, happiness, disgust). We choose the Mel-Frequency Cepstral Coefficients (MFCCs) (Milner & Shao, 2006) as the characterizing low-level audio features due to previous proved success on similar problems (Dahake, Shaw, & Malathi, 2016; Kandali, Routray, & Basu, 2008; Krishna Kishore & Krishna Satish, 2013; Sreeram, Geyasruti, Narayanan, & M, 2015).

The features are extracted using the python_speech_features library (Lyons et al., 2020). In a first phase and using the scikit-learn library (Pedregosa et al., 2011), we apply various recognition algorithms on the original data such as K-Nearest Neighbors (KNN), random forests, decision trees, Support Vector Machines (SVM) etc. In a second phase, we augment the data using `Pydiogment` by applying the following techniques:

- slow down samples using a coefficient of 0.8.
- speed up samples coefficient of 1.2.
- randomly crop samples with a minimum length of 1 second.
- add noise with an SNR = 10
- add a fade in and fade out effect.
- apply gain of -100 dB.
- apply gain of -50 dB.
- convolve with noise file using a level = 10**-2.75.
- shift time with one second (1 sec) to the right (direction = right)
- shift time with one second (1 sec) to the left (direction = left)
- change tone with tone coefficient equal to 0.9.
- change tone with tone coefficient equal to 1.1.

Then we re-run the same recognition algorithms on the augmented and original data. The following is a comparison of the results:

Machine learning Algorithm	Accuracy (no augmentation)	Accuracy (with augmentation)
KNN	0.588	0.622
Decision Tree	0.474	0.568
AdaBoost	0.258	0.429
Random Forest	0.639	0.753
Linear SVM	0.113	0.286
Extra Trees Classifier	0.680	0.768

Table 1: Accuracy comparison of results with and without data augmentation.

Conclusion

This paper introduced `Pydiogment`, a Python package for audio data augmentation, with diverse audio deformation strategies. These strategies aims to improve the accuracy of audio based recognition system by scaling the training data-set and increasing its quality/diversity. The utility of `Pydiogment` was proved by showing its effects when used in a spoken emotions recognition task. In the stated experiment, the augmentation using `Pydiogment` improved the accuracy up to 50%.

References

Burkhardt, F., Paeschke, A., Rolfes, M., A., Walter F. Sendlmeier & Weiss, B. (2005). A database of german emotional speech. *INTERSPEECH.*

Dahake, P. P., Shaw, K., & Malathi, P. (2016). Speaker dependent speech emotion recognition using mfcc and support vector machine. In *2016 international conference on automatic control and dynamic optimization techniques (icacdot)* (pp. 1080–1084). doi:10.1109/ICACDOT.2016.7877753

FFmpeg Developers. (2019). FFmpeg tool (version 4.2.2). Retrieved from http://ffmpeg.org/

Kandali, A. B., Routray, A., & Basu, T. K. (2008). Emotion recognition from assamese speeches using mfcc features and gmm classifier. In *TENCON 2008 - 2008 ieee region 10 conference* (pp. 1–5). doi:10.1109/TENCON.2008.4766487

Krishna Kishore, K. V., & Krishna Satish, P. (2013). Emotion recognition in speech using mfcc and wavelet features. In *2013 3rd ieee international advance computing conference (iacc)* (pp. 842–847). doi:10.1109/IAdCC.2013.6514336

Lyons, J., Wang, D. Y.-B., Gianluca, Shteingart, H., Mavrinac, E., Gaurkar, Y., Watcharawisetkul, W., et al. (2020). *Jameslyons/python_speech_features: Release v0.6.1*. Zenodo. doi:10.5281/zenodo.3607820

McFee, B., Humphrey, E., & Bello, J. (2015). A software framework for musical data augmentation. In *16th international society for music information retrieval conference*, ISMIR. Retrieved from https://github.com/bmcfee/muda

Milner, B., & Shao, X. (2006). Clean speech reconstruction from mfcc vectors and fundamental frequency using an integrated front-end. *Speech Communication*, *48*(6), 697–715. doi:10.1016/j.specom.2005.10.004

Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., & Le, Q. V. (2019). SpecAugment: A simple augmentation method for automatic speech recognition. In *INTERSPEECH*. doi:10.21437/interspeech.2019-2680

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., et al. (2011). Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, *12*, 2825–2830.

Poularikas, A. D. (1999). *The handbook of formulas and tables for signal processing* (1st ed., Vol. 1, pp. 137–138). Boca Raton, USA: CRC Press. doi:10.1201/9781315219707

Raju, A., Panchapagesan, S., Liu, X., Mandal, A., & Strom, N. (2018). Data augmentation for robust keyword spotting under playback interference. *ArXiv*, *abs/1808.00563*.

Self, D., Duncan, B., Sinclair, I., Brice, R., Hood, J. L., Singmin, A., Davis, D., et al. (2009). *Audio engineering: Know it all* (Vol. 1, pp. 277–278). Oxford, UK: Elsevier Inc.

Shelvock, M. (2012). *Audio Mastering as Musical Practice* (Master's thesis). The University of Western Ontario: The School of Graduate; Postdoctoral Studies, London, Ontario, Canada. Retrieved from https://ir.lib.uwo.ca/etd/530

Sreeram, L., Geyasruti, D., Narayanan, R., & M, S. (2015). Emotion detection using mfcc and cepstrum features. *Procedia Computer Science*, *70*, 29–35. doi:10.1016/j.procs.2015.10.020

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., et al. (2019). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 1–12. doi:10.1038/s41592-019-0686-2