Speech Enhancement and De-noising for Classification Tasks.

Preethi Natesan Sridhar, Samuel Merten, Shrivats Sudhir, Srivats Kumar Taralinath.

December 13, 2023.

Abstract

This report is intended to explore the use of unsupervised isolation- and density-based ensemble models for anomaly detection and cleaning using speech data. Furthermore, the paper also shows neural network classification implemented to quantify model performance improvement before and after cleaning. The dataset for this task contains $\approx$ 1.98 million audio files distributed among 34 languages. University of Wisconsin's Center of High-Throughput Computing (CHTC) clusters were utilized to conduct this task in a methodical and computationally efficient manner.

1 About the Dataset

Common Voice by Mozilla is a publicly-available voice dataset from volunteer contributors. The data was downloaded in parallel, with each file containing approximately 60,000 audio clips. For each language, shell scripts were written to randomize, split, chunk, and tar data to prepare them for staging. The end result was 1300 tar files, each of which contained 1500 audio files, totaling up to 65 gigabytes of data. Further details about Common Voice can be found here.

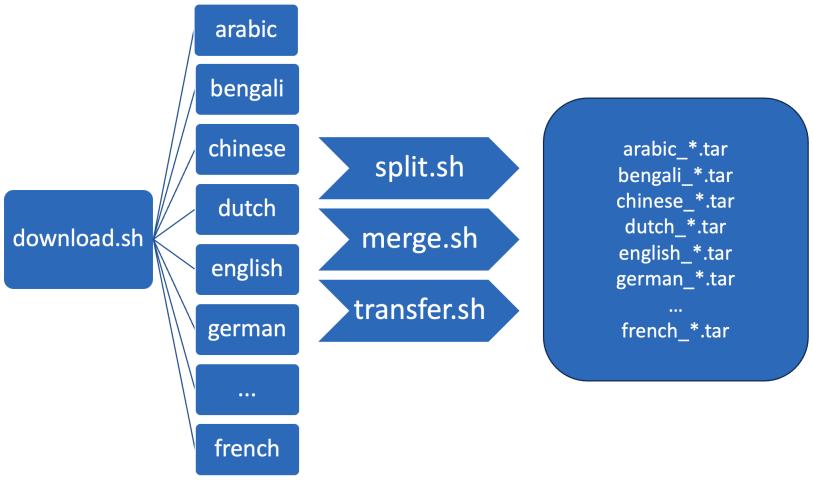

Figure 1: Flowchart describing downloading parallel processes used.

2 Data Engineering

Audio files were loaded, and sound features were extracted using the `librosa` package in python. The features used throughout the project include root mean squared energy, zero crossing rate, spectral flatness, mel-frequency cepstral coefficients, and chroma short-time fourier transforms. For each of those features, the mean, standard deviation, minimum, maximum, first quartile, third quartile, and skew statistics were chosen. To make the features usable for `sklearn` and `tensorflow`, relevant procedures such as standardization, feature stacking, dimension reduction, and dataframe creation were done using `numpy` and `pandas`. Furthermore, other packages like `os`, `glob`, and `shutil` were used to conduct OS and command line functionalities.

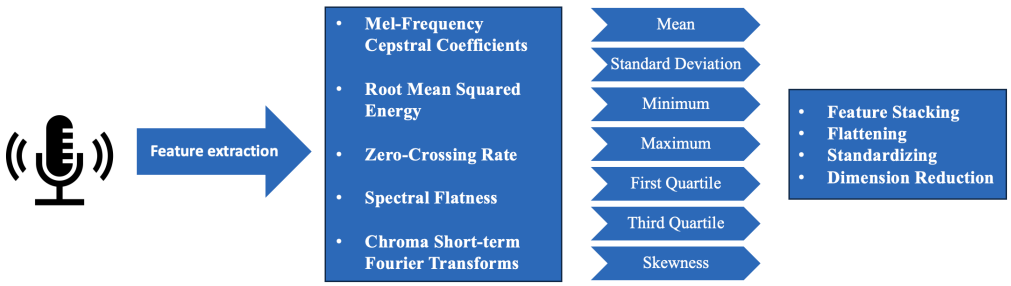

Figure 2: Flowchart describing engineering procedures used.

3 Outlier Detection to Find Anomalies.

A combination of two unsupervised outlier detection methods were used for this project. First, Isolation Forests (IF) detected anomalies by utilizing binary trees to recursively generate sub-samples to segment and search for outliers in the data, without employing any distance or density measures. Secondly, a Local Outlier Factor (LOF) was used to compute the local density deviation of a given data point with respect to its neighbors and identify outliers. `sklearn`'s IF and LOF functions were used for our project, and the intersection of these was used to classify audio as outliers or typical.

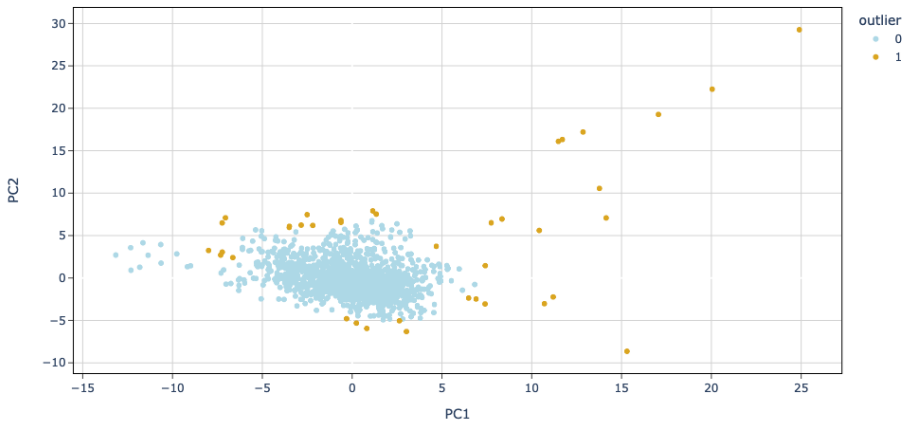

Figure 3: 2D PCA projection with anomalies for a single tar file.

4 Speech Detection

For the outlier audio found by the ensemble algorithm described above, we ran a preliminary detection procedure to determine whether speech was present in the audio file. To conduct this, a moving average (given parameters: window, stride, and threshold) was taken of the original sound signal waveform, and the following algorithm was run:

$$\xi(\vec{\mu}) = \begin{cases} \text{False} & \text{if } |\max_i - \min_i| < \text{threshold} \\ \text{True} & \text{Otherwise} \end{cases}$$

If $\xi(\vec{\mu}) = \text{False}$, no speech detected (and the audio is removed), otherwise it's de-noised.

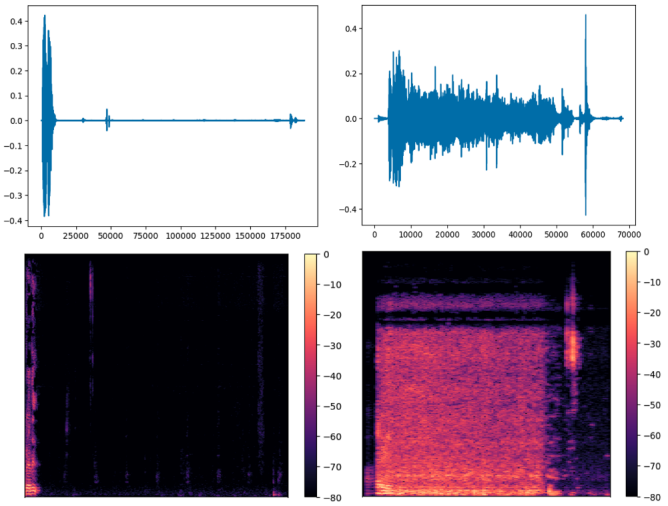

Figure 4: Examples of audio files with no speech.

5 De-noising using spectral gating

For outlier audio with detected speech, a de-noising algorithm was applied using spectral gating via the `noisereduce` package, for which our high-level understanding includes the following as a few main steps:

1. Calculate Fast-Fourier Transforms (FFT).
2. Calculate Statistics on Result.
    - Use resulting "power" vector and its statistics to calculate a threshold for filtering.
3. "Mask" is smoothed and reapplied, inverted, to the FFT of the signal to produce a "cleaned" result.

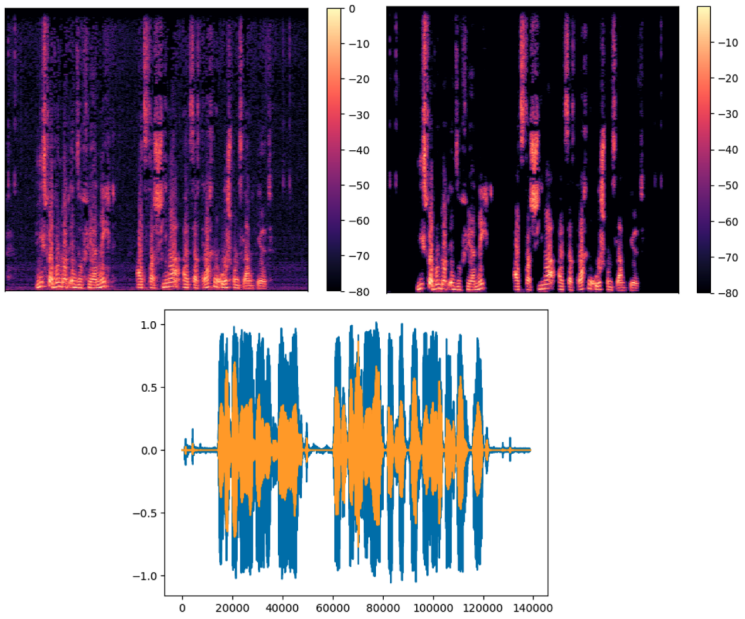

Figure 5: Examples of audio files that needed de-noising (before and after).

6 Metrics

To analyze the effectiveness of our speech enhancement process, we created two Convolutional Neural Network (CNN) Models trained on both clean or unclean data and compared their performances.

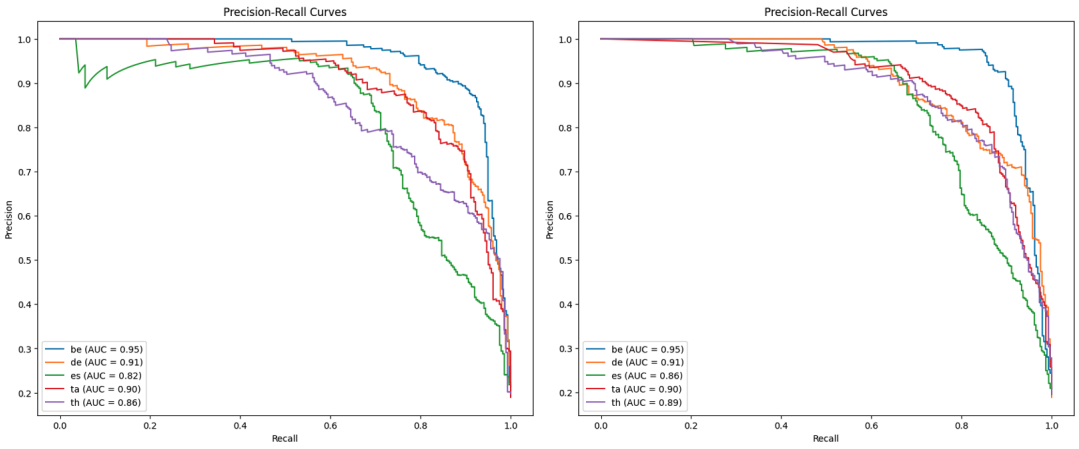

Figure 6: Precision Recall Curves (left: before cleaning, right: after cleaning).

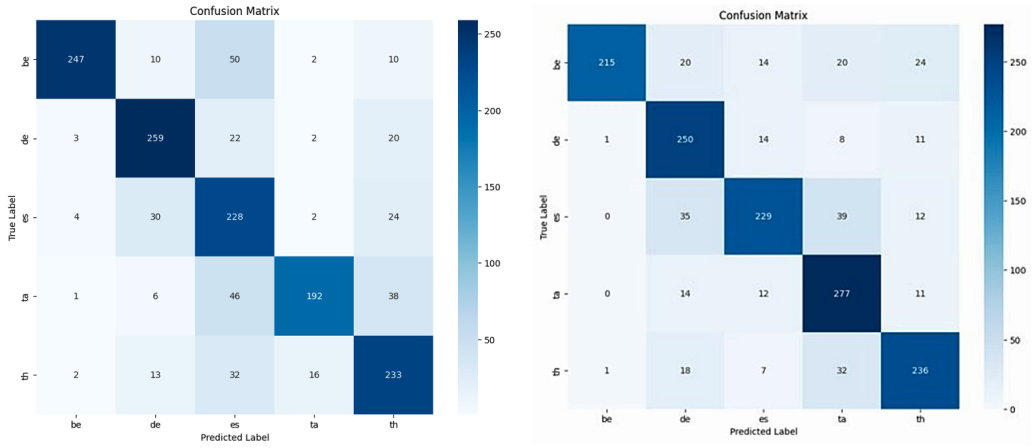

Figure 7: Confusion matrices (left: before cleaning, right: after cleaning).

7 Conclusion

For only 7500 audio files and 5 hand-picked languages - Bengali, German, Spanish, Tamil, and Thai - the overall test accuracy increased by 2%. Furthermore, the AUC score either remained unchanged or increased across the board (the curves also showed more consistent results and sharper trends) and the confusion matrix displayed a higher accuracy (as can be seen by consistently darker diagonal and consistently lighter non-diagonal elements). Subsequently, we conducted this speech enhancement algorithm for all audio files across every language by running 1320 parallel jobs for each tar file.

8 References and Appendix

Breunig, Markus M. and Kriegel, Hans-Peter and Ng, Raymond T. and Sander, J (2000). LOF: Identifying Density-Based Local Outliers. Association for Computing Machinery. New York, NY, USA. https://doi.org/10.1145/335191.335388.

Ji Wu, Fei Yang, Wenkai Hu (2023). Unsupervised anomalous sound detection for industrial monitoring based on ArcFace classifier and gaussian mixture model. Applied Acoustics. https://doi.org/10.1016/j.apacoust.2022.109188

Liu, Fei Tony and Ting, Kai Ming and Zhou, Zhi-Hua (2012). Isolation-Based Anomaly Detection. Association for Computing Machinery. New York, NY, USA. https://dl.acm.org/doi/10.1145/2133360.2133363.

Lo Scudo, F., Ritacco, E., Caroprese, L. et al. Audio-based anomaly detection on edge devices via self-supervision and spectral analysis. J Intell Inf Syst (2023). https://doi.org/10.1007/s10844-023-00792-2

Sainburg T. timsainb/noisereduce: v1.0.1; 2019. https://github.com/timsainb/noisereduce. Available from: https://doi.org/10.5281/zenodo.3243589.