Random Forest-based Voice Activity Detector

1 Introduction

The *Voice Activity Detection* (VAD) module has emerged as the fundamental pre-processing block for speech processing devices. This has allowed for speech recognition products such as Kaldi and Amazon Alexa to thrive in a world where speech has become one of the primary ways of accessing and using this new technology. While speech and voice detection applications employ the use of both near and far field processing, the building blocks in both cases detail the use of sophisticated signal processing that allows for recognition and real-time processing of spoken words.

The main motivation for this project arises from the fact that the number of applications that use speech and voice detection tools have risen considerably over the last few years. An opportunity to analyze this segment of signal processing using the current tools of machine learning classification algorithms presents a unique opportunity into understanding how speech detection works and what are its strengths in contrast to other audio semantic tools.

The primary goal of this study is aimed at employing the *Random Forest Classifier*, a decision tree-based machine learning algorithm, for the detection of speech events in noisy urban environments. A number of analysis tools such as confusion matrices, *Receiver Operating Characteristic* (ROC) plots and *Signal-to-Noise ratio* (SNR) are used to determine the performance of the system. Towards the end of the paper, *three* improvements to the current method have been discussed.

2 Theoretical Background

This section is aimed at informing the reader about the various tools required for speech detection using the Random Forest classifier. The *first* sub-section details the feature extraction algorithms used as part the project while the *second* sub-section briefly describes the Random Forest classifier.

2.1 Feature Extraction

The process of feature extraction is vital to the success of the speech detection function. Speech and non-speech section can be distinguished by calculating "time domain, frequency domain and coefficient domain features." (Thambi et al. 2014) For each time domain feature, the input signal $S_w(i)$ is divided into 20 ms

windows with a 50% overlap between consecutive frames. For the frequency domain features, the signal is divided into 20 ms frequency bands with a 50% overlap, before taking the Fourier Transform of each frame. While many types of features can been used for semantic analysis, this project focuses on the extraction of *ten features* that have proven to be robust parameters for machine learning and speech detection projects. The following section describes the features briefly:

1. *Zero Crossing Rate:* The ZCR feature is a parameter used for calculating the noisiness level of a signal. It can be described as the number of times the audio signal changes signs in a given frame of length N. (Lavner et al. 2009)

$$ZCR = \frac{1}{2}\sum_{n=1}^{N-1} |sgn(x[n]) - sgn(x[n-1])| \tag{1}$$

2. *High Zero-Crossing Rate Ratio:* While Zero-Crossing Rate (ZCR) is a robust measure for detecting the noisiness in a given signal, the High Zero-Crossing Rate Ratio (HZCRR) presents a more discriminant value of the input signal. It measures the ratio of the number of frames whose ZCR value lies more than 1.5 times the average ZCR value within each frame. (Lu et al. 2002)

$$HZCRR = \frac{1}{2N}\sum_{n=0}^{N-1} [sgn(ZCR(n) - 1.5 * avZCR) + 1] \tag{2}$$

3. *Short-term Energy:* The short-time energy feature measures the sum of the squared samples in a given window of length N. This measure is then normalized by the dividing it by the frame length. (Thambi et al. 2014)

$$E = \left(\frac{1}{N} * \sum_{n=0}^{N-1} x^2[n]\right) \tag{3}$$

4. *Low Short-Time Energy Ratio:* Similar to the HZCRR, this measure is defined as the ratio of the number of frames whose short-time energy value is less than 0.5 of the short-time energy of the given frame. (Lu et al. 2002)

$$LSTER = \frac{1}{2N}\sum_{n=0}^{N-1} [sgn(0.5avSTE - STE(n)) + 1] \tag{4}$$

5. *Spectral Centroid:* It calculates the spectral shape of a given frame, N, and describes the center of gravity for a distribution. (Thambi et al. 2014)

$$C_t = \frac{\sum_{k=0}^{N-1}(k+1)X_t k}{\sum_{k=0}^{N-1} X_t k} \tag{5}$$

6. *Spectral Entropy:* The Spectral Entropy feature first divides the short-term spectrum into a number of sub-bands, then normalizes this sub-band energy in relation to the overall energy of the spectrum. This yields the normalized spectral energy of each frame. (Thambi et al. 2014)

$$H = -\sum_{i=0}^{L-1} n_i.log_2(n_i) \tag{6}$$

7. *Spectral Flatness:* The flatness variable of a distribution is measured simply by taking the ratio of the geometric mean and the arithmetic mean of the given distribution. (Bello, 2018)

$$SF(m) = \frac{\Sigma_k |X(m,k)|^{\frac{1}{K}}}{\frac{1}{K}\sum_k |X(m,k)|} \tag{7}$$

8. *Spectral Rolloff:* The Spectral Rolloff of a frame of audio indicates the frequency below which a certain percentage of the spectral distribution lies. (Thambi et al. 2014) Given the DFT coefficient m and frame number t, the following equation can be solved for the spectral rolloff coefficient:

$$\sum_{k=0}^{m} X_t(k) = C \sum_{k=0} N - X_t(k) \tag{8}$$

9. *Mel Frequency Cepstral Coefficient:* The MFCC coefficients allow for the calculation of real cepstral sub-band information across the spectrum. It makes use of a Mel filterbank containing overlapping triangular windows, which is multiplied by the log magnitude spectrum to obtain the MFCCs for each frame. (Dave, 2013)

10. *Per-Channel Energy Normalization:* The PCEN method can be described as an iteration of the MFCC method wherein the log factorization is replaced by a form of dynamic normalized compression. Since the log function "uses a lot of its dynamic range on low-level less informative sections, and its dependency on loudness, the PCEN compression ratio allows for dynamic level stabilization." (Wang et al. 2016) Given the time and frequency indices, *t* and *f*, and sub-band energy, *E(t,f)*, the PCEN compression ratio can be defined as follows:

$$PCEN(t,f) = \left(\frac{E(t,f)}{(\epsilon + M(t,f))^{\alpha}}\right)^{r} - \delta^{r} \tag{9}$$

2.2 Random Forest Classifier

The Random Forest Classifier is a type of ensemble machine learning classification method that is based on the concept of decision trees. It is a form of a meta-estimator that places decision-tree classifiers on random subspaces and samples, and employs statistical mean tools to improve the prediction accuracy over time by controlling the phenomenon of overfitting. In a study published by Breiman (2001), this type of a classifier was defined as "consisting of a collection of tree-structured classifiers {*h(x, θk), k=1,...*} where the θk are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input x."

3 Implementation

3.1 Dataset Selection

The primary focus of this project relates to speech activity detection in a common street noise environment. This includes several urban environments wherein acoustic events are characterized by having both speech and non-speech events. As a result, one of the most important steps of this process involves the selection of an *appropriate dataset*, one that contains the required audio characteristics. The *three* sources of audio data selected as part of this study have been mentioned and described below:

1. The *Mozilla Common Voice* dataset was chosen as the corpus for this study. A corpus size of 15,000 audio files was prepared, organized and labelled as the training set for the process of binary classification.

2. A second set of 6000 soundscape audio files were extracted as the training data from the *UrbanSound SED* dataset by NYU MARL & CUSP groups. (Salamon et al. 2017) The latter was pre-sorted in classes, each representing a common urban acoustic event such as jackhammer, dog bark, etc. before being mixed with Brownian noise.

3. 8 traffic noise ambience samples were extracted from the commercial sound effects library *Sound Ideas Digieffects* series as part of the audio dataset.

3.2 Workflow Arrangement

Workflow arrangement is a critical step in the process of speech/non-speech binary classification. This step entails the processing of the raw audio dataset into a training class which is subject to the Random Forest classifier for the classification process. The process of speech detection, through the use of the workflow arrangement method, can be divided into two stages: *pre-processing* and *machine learning - processing* The details of each of the stages are as follows:

3.2.1 Pre-processing

The importance of the pre-processing stage cannot be undermined. Its main task is to iteratively implement all the features extraction algorithms on each audio file of the target folder. This is done by from reading the targeted audio files and compiling them into a *Pandas DataFrame* structure for further processing. Since this stage of the workflow arrangement involves reading and structuring the database files, the user only has to define parameters for these features extraction algorithms. These include the window size, hop size, minimum and maximum frequency range and DCT values among others. Once these parameters have been imported into the *Python* framework, the program runs all the feature extraction algorithms on the audio files and concatenates the processed arrays as a *Pandas DataFrame* structure. As an option, the program also allows the user to write the *DataFrame* structure as a .csv file that can be saved for future use. In addition to the *DataFrame* function, a function called *DataFrameSelector* was developed that can select or drop audio features required to train the model.

In the *DataFrame*, each row denotes the frame number whose length can be assigned by the function. On the other hand, each column denotes the value of the audio feature for each frame. In addition to the audio features, two more attributes, *Label* and *FileID*, were attached. The former indicates if the sample frame, n, contains speech, and the latter denotes the audio file from which the sample frame has been extracted. Not only does this aid in the visual analysis of the feature extraction dataset but also allows for efficient computational processing since the binary label value of speech and non-speech for all frames and for all the audio files in the dataset are arranged in a single column vector.

3.2.2 Machine Learning

The second stage, machine learning, involves the process of training the pre-processed dataset as defined by the *DataFrame* parameter. This step was conducted using the Python machine learning package, *Scikit-Learn* (Pedregosa et al. 2011). The initial step of this stage involves the elimination of any bias that may

```
In [115]: 1 Dataset.head()
```

Out[115]:

	Label	FileID	Short Time Energy	Average Energy	Low Short Time Energy Ratio	ZCR	Norm ZCR	Spectral Flatness	Spectral Centroid	Spectral Rolloff	...	PCEN_MFCC_11	PCEN_MFCC_12	PCEN_MFCC_
0	1.0	1.0	0.008504	7.002382	-1.705410	0.009766	-0.143976	0.000791	1159.800169	1055.126953	...	-0.072515	0.012460	-0.0375
1	1.0	1.0	0.004754	7.002382	2.134166	0.019531	-0.134211	0.000083	617.819115	645.996094	...	0.663492	0.374699	-0.0460
2	1.0	1.0	0.002642	7.002382	4.296606	0.020508	-0.133234	0.000129	715.285524	882.861328	...	-0.529634	0.093191	0.3656
3	1.0	1.0	0.017012	7.002382	-10.418251	0.021484	-0.132258	0.000044	648.154309	710.595703	...	-0.280050	-0.921777	0.2030
4	1.0	1.0	0.033503	7.002382	-27.304811	0.020508	-0.133234	0.000036	668.845307	904.394531	...	-0.785219	-0.279951	0.6515

Figure 1: First 5 samples in the DataFrame by calling *DataFrame.head()*

have formed either during data collection or data processing. To eliminate any bias, the program needs to make sure that the amount of speech data is equal to the amount of test data in the system. Once this has been confirmed, the order of all the frames in the *DataFrame* are shuffle. The *GroupKFold* function is used to split this data into the training and test set. The study employed the use of the *GroupKFold* function for the following two reasons:

1. One, the *GroupKFold* allows for K-fold processing with the non-overlapping group. It splits dataset into K consecutive folds, and each fold is used once as a validation while the remaining folds form the training set. By doing this, all the data can be used iteratively as a training, test, and validation set and decreases the bias at each step of the training stage.

2. Two, since splitting of the dataset into the test and training set happens at the frame level, there is an underlying risk of distributing consecutive frames to both the test and training set. The *GroupKFold* function allows the program to avoid this type of error. For example, if two consecutive frames from the same audio file were added to the training and test sets, then the overall bias of the program increases. To avoid such a situation, the *GroupKFold* function annotates the groups using *FileID* and thus, avoid frames of the same group (fold) appearing in the two different folds.

Once the sample biased has been eliminated and the data is split into the training and test set, the Random Forest Classifier is fitted in each fold. Since the Random Forest classifier is a tree-based estimator, no scaling of the processed data is required.

4 Results and Discussion

4.1 Confusion Matrix

The evaluation phase of the system consists of evaluating the parameters in each fold. For each fold, a confusion matrix is computed that allows for a clear visual representation of the predicted and true class labels. The *rows* of the confusion matrix indicate the latter while the *columns* indicate the former. The diagonal of the confusion matrix represents correct classification while non-diagonal elements represent misclassifications made by the system.

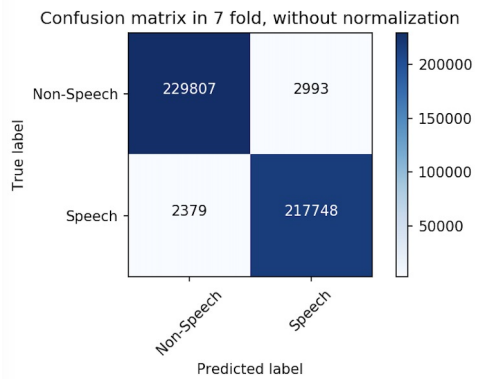


Figure 2: Confusion matrix for Fold 7 with no data normalization

The above figure shows an example of a confusion matrix generated for Fold 7 of the processed dataset. It shows that for a total of 452,927 frames of data, 232,800 are non-speech frames and the remaining 220,127 are speech frames. It states that 98.9% of the speech frames have been correctly classified. Of the 2379 speech frames, about 1.08% were wrongly classified as non-speech and, of the 2993 non-speech frames, 1.2% were recognized incorrectly as speech.

4.2 Receiver Operating Characteristic (ROC)

Similar to the precision and recall parameters, binary classification systems make use of a *Receiver Operating Characteristic* (ROC) plot that plots the *true positive rate* (TPR) against the *false positive rate* (FPR) for each fold computed as part of the machine learning process. One of prerequisite of using the ROC relates to an equal scaling of the dataset for both classes. Since our study ensures equal scaling of the datasets during the pre-processing stage, the use of the ROC plot fits this project.

Figure 3 is an example of an ROC plot of a Random Forest Classifier with 10 sub-trees in the forest and a maximum depth of 2. The plot highlights how the classifier is affected by the splitting-changing effect in

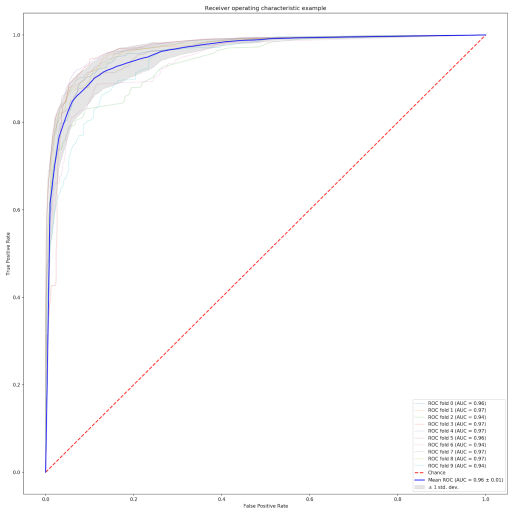


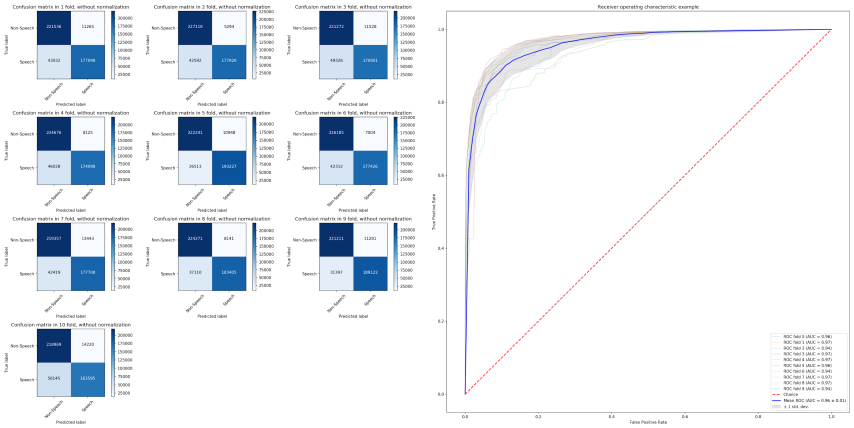
Figure 3: Receiver Operating Characteristic (ROC) plot for all Folds

each fold of the training set. Furthermore, it calculates the mean ROC value for each fold with a standard deviation value of +1/-1.

4.3 Machine Learning Evaluation

4.3.1 k-Fold

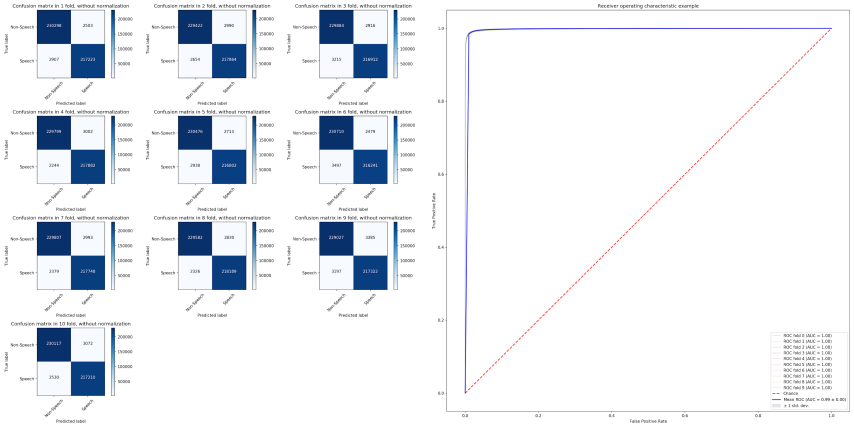
Now that we know what the confusion matrix and the ROC plot indicate, the analysis section can now proceed to discussing the results obtained during the machine learning process by employing 10 folds as part of the Random Forest Classifier. The results are as follows:



(a) Confusion Matrix (b) ROC Curve

Figure 4: Confusion Matrix and ROC plot for Classifier 1 (Max Depth = 2)

Figure 4 (above) plots the confusion matrix and ROC plot for a simple Random Forest Classifier, one that contains 10 trees in the forest and a maximum depth of 2. On the other hand, Figure 5 (below) shows a more complex implementation of the Random Forest Classifier that contains 75 trees in the forest but does not limit the depth value. Both these examples make use of *bootstrapping*, *gini criterion* and considers the square value of the features used. For example, if 50 features are employed in the system, the classifiers will consider using 250 features when looking for the best split.



(a) Confusion Matrix (b) ROC Curve

Figure 5: Confusion Matrix and ROC plot for Classifier 2 (Unlimited Max Depth)

4.3.2 Signal-to-Noise Ratio (SNR) Evaluation

Besides evaluating the classifier through the ROC plot, the study also uses a software called Scaper that allows for the creation of test audio by simulating speech in real-world urban noise environment. *Scaper* is an open source library for soundscape synthesis, developed by the MARL and CUSP group at NYU. (Salamon et al. 2017) It allows the mixing of a given foreground acoustic events with background acoustic events. Furthermore, it allows the user to conveniently define parameters such as event class, duration and signal-to-noise ratio (SNR) among others.

The speech corpus extracted from *Mozilla Common Voice* dataset was used as the foreground speech sound whereas the *Sound Idea Digieffects* dataset was employed as the background noise. The two sound events were mixed together using *Scaper* to generate the test audio set. This audio set was subject to the workflow arrangement wherein the frames of these audio files were pre-processed and passed through the Random Forest Classifier.

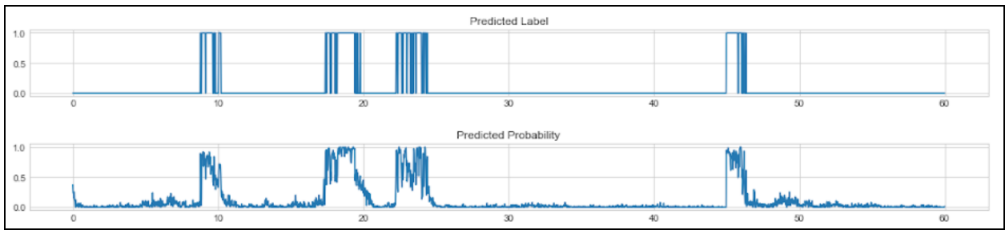


Figure 6: SNR = 15:1 with speech at 7-10 sec, 16-19 sec, 21-24 sec, 43-46 sec

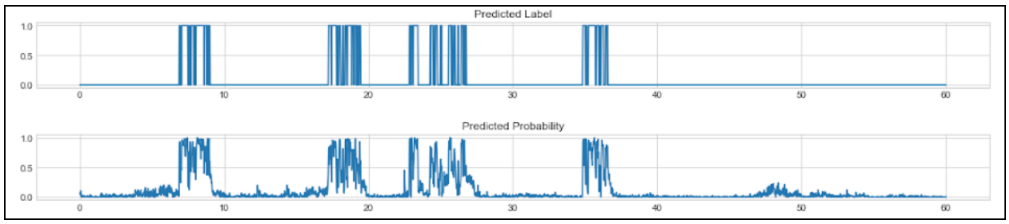


Figure 7: SNR = 10:1 with speech at 6-9 sec, 17-19 sec, 23-26 sec, 33-36 sec

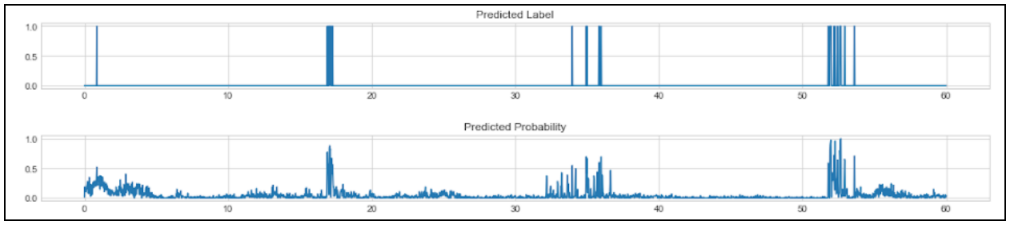


Figure 8: SNR = 5:1 with speech at 16-18 sec, 32-36 sec, 52-54 sec

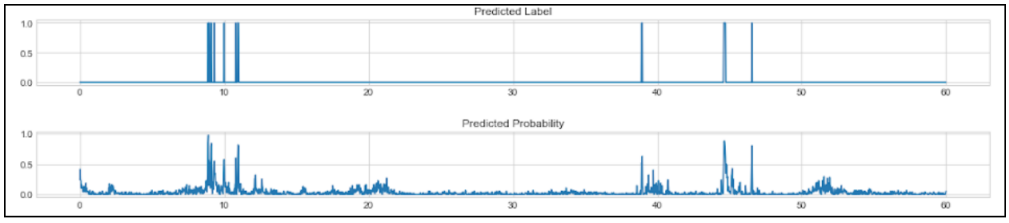


Figure 9: SNR = 0:1 with speech at 8-13 sec, 37-41 sec, 44-47 sec

The above mentioned figures indicate the prediction made by the classifier for a set of four audio files, where the speech data was mixed at fixed positions against the ambient background. A signal-to-noise ratio (SNR) value of 15:1, 10:1, 5:1 and 0:1 was assigned to the four files respectively. For example, a SNR value of 15:1 means the mixed speech signal is placed 15 LUFS above the background ambience.

The plots indicate that lower SNR values make the predictions *more intermittent* even when misclassified. By limiting the max depth of each tree and lowering the SNR value, the classification probability could be improved. The idea here is rather simple; *increasing* the depth of each tree or maximizing the number of leaves on each node *reduces* the bias of each sub-tree while increasing the size of the forest. As a result, the overall variance in the classification module is reduced without affecting the performance of a sub-tree, n.

5 Conclusion

This study was aimed at evaluating the performance of the Random Forest Classifier, a part of the *Scikit-Learn* library for *Python*, by detecting speech events in an urban street noise environment. The *UrbandSound SED* and *Mozilla Common Voice* datasets were used as the non-speech and speech corpus respectively, and the dataset for training the machine learning function was obtained by applying feature extraction algorithms, both in the time and frequency domain. Within each of the cross-validation fold, the classifier was fitted with the training set before evaluating the prediction accuracy in each fold. The confusion matrix and ROC curve plot was used to visualize and evaluate the results. To simulate how this classifier works in a real-world situation, the study used the open-source library, *Scaper*, to mix speech events in urban traffic ambience.

It was found that the **performance of the classifier improved** as the SNR of the speech signal increase. Furthermore, the performance also differed for changes in the input parameters such as window size, hop size, maximum depth for the forest and more.

6 Future Work and Improvements

The future work will be aimed at implementing multiple classification algorithms such as Support Vector Machine (SVM) and Decision Tree among others, and comparing the accuracy of these methods. Furthermore, different datasets will be employed as part of the training phase, a factor that increases the variety of sound events. While the Random Forest Classifier is a classification algorithm, clustering machine learning functions will also be explored in a future study. In conjunction to this study, several improvements to the current system could be achieved as future works:

1. First, since the estimator made prediction on the frame level, due to the instantaneous pauses between words, the predictions are usually in the form of a sequence of intermittent frames. Thus, adding a smooth decision module would help avoid this trend. One option is to run a moving average or median filter with a window size of 20-40 ms wherein smoothing could be achieved by considering the neighbouring predictions of a frame, N.

2. Second, the use of Principal Component Analysis (PCA) can help streamline the feature extraction algorithms used. Since these algorithms may correlate with each other, implementing a rank that denotes the contribution of each can aid in reducing the feature dimensionality. This can, indeed, suppress the noise level in the dataset and make the training process more efficient.

3. Third, as can be seen from the analysis section, the performance of the classifier is sensitive to changes in the SNR. One way of increasing its performance could be to fit the classifier with perturbed data, a process called Data Augmentation. If the classifier is trained under the synthetic condition that the system has annotated speech and non-speech events, then a simple comparison with the original data should increase its performance efficiency.

References

Atal, B., & Rabiner, L. (1976). "A pattern recognition approach to voiced-unvoiced-silence classification with applications to speech recognition." *IEEE Transactions on Acoustics, Speech, and Signal Processing, 24(3)*, pp. 201-212

Breiman, L. (2001). "Random forests." *Machine learning, 45(1)*, pp. 5 - 32.

Common Voice by Mozilla. (n.d.). Retrieved from *https://voice.mozilla.org/en/datasets.*

Dave, N. (2013). "Feature extraction methods LPC, PLP and MFCC in speech recognition." *International journal for advance research in engineering and technology, 1(6),* pp. 1 - 4.

Graf, S., Herbig, T., Buck, M., & Schmidt, G. (2015). "Features for voice activity detection: a comparative analysis." *EURASIP Journal on Advances in Signal Processing, 2015(1),* pp. 91.

Lavner, Y., & Ruinskiy, D. (2009). "A decision-tree-based algorithm for speech/music classification and segmentation. *EURASIP Journal on Audio, Speech, and Music Processing, 2009,* pp. 2 - 4.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Vanderplas, J. (2011). "Scikit-learn: Machine learning in Python." *Journal of machine learning research, 12(Oct),* pp. 2825 - 2830.

Renevey, P., & Drygajlo, A. (2001). "Entropy-based voice activity detection in very noisy conditions." *Seventh European Conference on Speech Communication and Technology.*

Saeedi, J., Ahadi, S. M., & Faez, K. (2015). "Robust voice activity detection directed by noise classification." S*ignal, Image and Video Processing, 9(3),* pp. 561-572

Salamon, J., MacConnell, D., Cartwright, M., Li, P., Bello, J. (2017). "Scraper: A Library for Soundscape Synthesis and Augmentation." *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics,* pp. 344 - 348.

Thambi, S., Sreekumar, T., Santosh, C., Reghu, R. (2014). "Random Forest Algorithm for Improving the Performance of Speech/Non-Speech Detection." *First International Conference on Computational Systems and Communications (ICCSC),* pp. 28 - 32.

Wang, Y., Getreuer, P., Hughes, T., Lyon, R. F., & Saurous, R. A. (2017). "Trainable frontend for robust and far-field keyword spotting." *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* pp. 5670- 5674.