EN.601.805 (42) Graduate Independent Study
Project Report

UNIVERSAL VOICE ACTIVITY DETECTION

May 8, 2024

Arnav Shah
Whiting School of Engineering
Johns Hopkins University
ashah108@jhu.edu

Advised by Prof. Leibny Paola Garcia and Prof. Benjamin Langmead

1 Abstract

In the realm of speech processing, the majority of tasks necessitate clean audio data during the training phase. To achieve this, voice activity detection (VAD) becomes indispensable. This process involves eliminating periods of silence, reducing noise in the audio, and selectively supplying the essential segments containing speech to the designated model. Consequently, researchers face the challenge of either developing their own VAD model or relying on pre-existing models that may not exhibit optimal performance with their specific datasets. The objective of this work is to address these challenges by introducing a universal voice activity detector that boasts speed, efficiency in memory usage, and seamless integration into widely-used speech processing tools employed by researchers. This innovation aims to streamline the preprocessing stage, enhancing the accessibility and effectiveness of speech processing techniques across various applications. We open-source the code and models at https://github.com/arnavsshah/universal-voice-activity-detection

2 Introduction

Voice Activity Detection involves identifying whether human speech is present or absent in audio data. This crucial process plays a fundamental role in various speech processing and recognition workflows, serving to clean the data by distinguishing speech from background noise or silent segments. By implementing VAD, downstream tasks can effectively concentrate on their specific objectives without being hindered by extraneous elements such as ambient noise or silent intervals in the audio stream.

This project aspires to streamline VAD for researchers across diverse projects by developing a universal VAD tool as a Python package. Designed for easy integration, researchers can effortlessly import the Python package into their projects, enabling efficient VAD implementation on any audio file or dataset. This emphasis on the Python package underscores the project's commitment to providing a convenient and adaptable solution for the research community. By eliminating the need for researchers to train models from scratch to suit specific datasets, the tool aims to offer a versatile solution capable of seamlessly working across a variety of acoustic conditions.

This tool integrates with Lhotse [1], a popular data preparation library for speech processing tasks and thus, can output data that is easily consumed by Lhotse and its dependents such as Icefall [2] and k2 [3].

3 Literature Review

Voice Activity Detection is a crucial component in numerous speech processing applications, ranging from automatic speech recognition to speaker identification. Accurate identification of speech segments within audio streams is fundamental for enhancing the

performance and efficiency of these applications. In recent years, there has been a growing interest in leveraging self-supervised learning techniques to extract discriminative features for voice activity detection. This section reviews key literature in the field, highlighting advancements and methodologies employed in incorporating self-supervised learning for VAD.

Early VAD methods often relied on handcrafted features and statistical models, such as energy-based thresholds [5] [6] and Gaussian Mixture Models [7]. While effective in certain scenarios, these approaches faced challenges in handling diverse acoustic conditions and noise environments.

The advent of deep learning revolutionized VAD by enabling the automatic extraction of hierarchical representations from raw audio data. Convolutional Neural Networks (CNNs) [9] and Recurrent Neural Networks (RNNs) [8] have been successfully applied to capture temporal and spectral patterns, improving VAD performance across different domains.

Representation Learning for Audio Features is the foundation of any speech processing task and is one of the primary determinants of the model's performance. Prior to the deep learning era, traditional audio feature extraction methods played a significant role in signal processing. Mel-Frequency Cepstral Coefficients (MFCCs) [10] and Filter Banks [11] were widely used for characterizing the spectral and temporal aspects of audio signals. These methods, while effective, often required manual tuning and were less adaptable to diverse acoustic conditions.

CNN based feature extractors such as SincNet [14] have shown significant improvements in tasks such as speaker recognition and are being used by libraries such as Pyannote [31] as part of their VAD implementation. SincNet produces features by analyzing the raw waveform, providing streamlined feature computation tailored specifically to the task it's trained for in conjunction with other components.

Recent studies have explored the use of self-supervised learning techniques to learn rich representations from audio signals. Contrastive learning, autoencoders, and other pretext tasks have been employed to train models that can capture meaningful and robust features from audio data. Wav2Vec 2.0, an advanced self-supervised learning model, has demonstrated remarkable success in learning contextualized representations from large amounts of unlabeled speech data [12]. Its ability to capture hierarchical features and contextual information has shown promise in enhancing the performance of downstream tasks, including VAD. Studies integrating Wav2Vec 2.0 into VAD frameworks have reported improved accuracy and robustness, especially in real-world, diverse acoustic environments [13].

Dataset Preparation has always been an inconvenience in the speech processing community. Datasets tend to have different audio formats, task-specific labels and variability in recording conditions. Researchers end up writing & re-writing data pre-processing pipelines for different frameworks and tasks so that their model can accept the data in the desired

format. To solve this, Lhotse [1] was introduced, a Python library aimed at easing the hassle and inconvenience of speech and audio data preparation.

Lhotse works by creating a `Cut` which represents an individual instance of a dataset. Each `Cut` consists of a `Recording` and an optional `Supervision`. A `Recording` is a python dictionary object that represents an audio file with all its metadata such as audio path, channels, sampling frequency, duration, additional transformations/augmentations, etc. Likewise, a `Supervision` segment represents a time interval within a `Recording` along with any annotated labels or text, such as the transcription, speaker identity, language spoken, etc. Multiple cuts combine to form a `CutSet` which represents the entire dataset. 1 depicts a cut created by Lhotse with cuts spanning an entire supervision or with uniform duration spanning multiple supervisions.

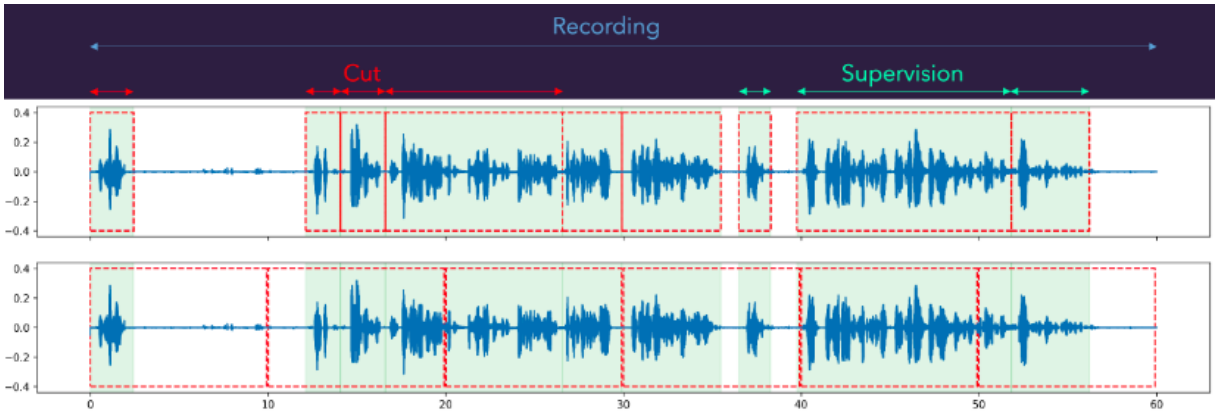

Figure 1: Cut in Lhotse

Lhotse offers a variety of data preparation 'recipes' that are specific to a dataset and/or task. Researchers can use these recipes out-of-the-box or add customizations to create datasets which in turn, can be fed into any model of their choice.

4 Implementation

The Python package we offer presents researchers with a complete speech processing pipeline. This pipeline encompasses the creation of datasets, whether they are new or existing, in a standardized format through the use of Lhotse. Subsequently, it involves extracting features from audio files utilizing self-supervised representation learning models. Following this, the package supports customized training tailored to both the model's specifications and the dataset's needs. Finally, it includes a comprehensive evaluation pipeline designed to assess the model's performance on previously unseen data.

As depicted in 2, the initial step involves providing the path to the audio files and their corresponding transcriptions/labels to the Lhotse data preparation function. This

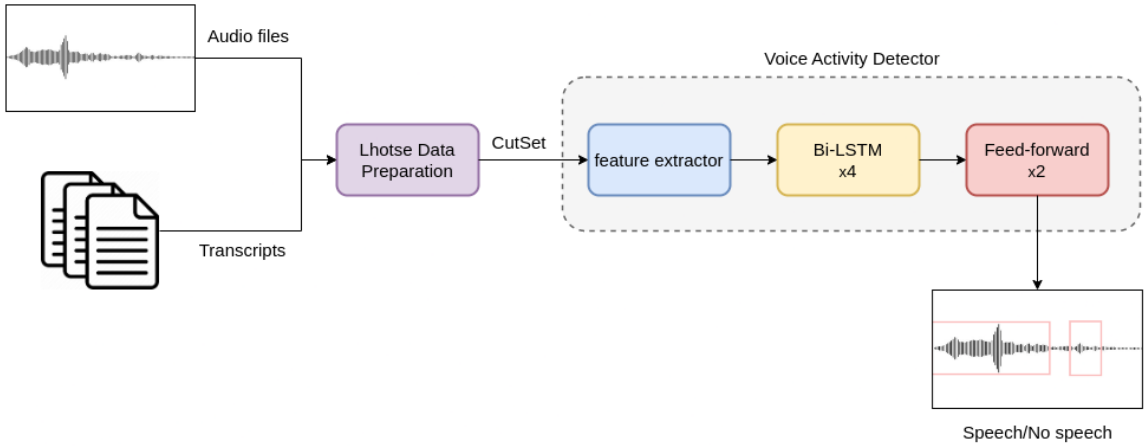

Figure 2: Voice Activity Detection Pipeline.

function, in return, produces a `CutSet`, which is utilized by the feature extractor such as SincNet, Filter banks, Wav2Vec 2.0 and Robust Hubert [28] to generate audio features. These features are then input into the voice activity detection model, which comprises Bi-directional LSTM layers followed by feed-forward layers. Ultimately, the model predicts one of two classes, namely speech or non-speech, for each audio frame.

4.1 Datasets

First, one must decide which dataset to use and subsequently, check if the preparation recipe for that dataset exists in Lhotse or not. If yes, it is a simple procedure to add a dataset which includes calling `prepare_dataset(...)` to get its recordings and supervision manifests and then combining the two to create a CutSet which in turn, can be used to compute audio features.

If a recipe for the selected dataset doesn't exist, one must create it in accordance with Lhotse's attributes and objects. This includes creating recordings and combining them to create a RecordingSet. The same must be done for Supervisions which can be read from `.ctm` files or something similar (get timestamps, speaker information, etc.). Then, follow the procedure as mentioned above.

We only support single-channel audio and thus, only the first channel is used in CutSets with multi-channel audio. Additionally, the feature extractor only accepts 16KHz audio and thus, any 8KHz datasets are upsampled to 16KHz using utility functions provided by Lhotse.

For the purposes of this study, we utilize 8 datasets, the recipes for which are available in Lhotse. They include TED-LIUM [16] (TED talks), CHiME-6 [19] (multi-speaker

conversations in everyday home environments), VoxConverse [18] (multi-speaker clips of human speech, extracted from YouTube videos), AMI [15] (multi-party meeting conversations), Switchboard [17] (two-sided telephone conversations), Gale Arabic [21] (broadcast news in Arabic), Gale Mandarin [22] (broadcast news in Mandarin) DIHARD3 [20] (multi-speaker diarization dataset in diverse environments)and CALLHOME American English [23] (telephone conversations).

We measure the performance of our model on the test sets of all the datasets above along with those of DiPCo [24] (multi-speaker dinner party conversations) and the Santa Barbara Corpus of Spoken American English [25] (noisy real-world spoken interactions). In the Appendix, we also show the incremental performance of the model as each dataset is added to the training set.

4.2 Data augmentation

The datasets we use are of varying length. The training set size of these datasets are as follows — TED-LIUM - 540 hours, CHiME-6 - 40 hours, VoxConverse - 20 hours, AMI - 70 hours, Switchboard 387 hours, Gale Arabic - 960 hours, Gale Madarin - 821 hours and DIHARD3 (development set) - 34 hours, Callhome English - 38.

Due to this imbalance, we experiment with uniformly sampling 50 hours from each dataset (or the entire dataset if it's size is less than 50). We uniformly keep sampling 1 continuous hour to sum up to 50 hours across the entire training set. As shown in Table 1, this setting results in equal or better performance across most datasets and takes significantly less time.

In addition to the above sampling, we also augment the speech with the MUSAN noise dataset [26] (randomly chosen from noise, speech and music) with a probability of 50%.

4.3 Feature extraction

For the audio features, we use the following feature extraction methods namely Filter Banks, and pre-trained self-supervised models (Wav2Vec 2.0, Robust HuBERT).

4.3.1 Filter Banks

We use the implementation within Lhotse to generate filter banks with 80 filters and a frame shift of 10ms. The have a lower dimension compared to the self-supervised models and are thus quicker to compute and use. This comes at a cost of less information in the features but isn't a huge problem for the VAD task as the features do not need to model complex interactions in speech that are required by other tasks such as ASR or Speaker ID.

4.3.2 Self-supervised models

We experiment with pre-trained Wav2Vec 2.0 and Robust HuBERT (trained with speech distortions) models as part of existing feature computation scripts from Lhotse for all the

Dataset	Duration	Det Err	FA	MD
TED-LIUM	all	**0.07**	**0.05**	**0.02**
	limited	0.09	0.07	0.02
CHiME-6	all	0.17	0.07	**0.1**
	limited	**0.17**	**0.06**	0.11
VoxConverse	all	0.21	**0.08**	0.13
	limited	**0.17**	0.09	**0.08**
AMI	all	0.22	0.12	0.1
	limited	**0.22**	**0.12**	**0.1**
Switchboard	all	0.29	0.22	0.07
	limited	**0.29**	**0.22**	**0.07**
Gale Arabic	all	0.07	**0.03**	0.04
	limited	**0.05**	0.04	**0.01**
Gale Mandarin	all	0.14	0.06	0.08
	limited	**0.1**	**0.06**	**0.04**
DIHARD3	all	0.34	**0.17**	0.17
	limited	**0.32**	0.18	**0.14**
Callhome English	all	0.28	0.22	0.06
	limited	**0.26**	**0.22**	**0.04**
DiPCo	all	0.12	**0.06**	0.06
	limited	**0.12**	0.07	**0.05**
Santa Barbara	all	0.3	0.15	0.15
	limited	**0.29**	**0.15**	**0.14**

Table 1: Comparison of model trained on all available speech per dataset (denoted by "all") and model trained on <= 50 hours of speech per dataset (denoted by "limited").

datasets. Both models have a frame-shift 20ms and an embedding dimension of 768. The weights of these models are frozen during training and thus, are not updated, to avoid a larger model size. However, as these models are primarily optimized for speech recognition, the features aren't as robust for the VAD task where detecting noise and distortions is important.

Using the CutSet generated by Lhotse, we compute audio features with a sampling rate of 16Khz. The features are computed in batches and are lazily loaded later for efficient memory usage.

4.4 Model

We employ the same model architecture utilized by [31] which consists of a feature extractor followed by 4 bi-directional LSTM layers with dropout 0.5, 2 feed-forward layers and a binary classification head (speech or no speech). The input dimension to the LSTM changes as the feature extractor changes. Using filter banks as a feature extractor requires an input dimension of 80 while using self-supervised models such as Wav2Vec 2.0 or robust HuBERT requires an input dimension of 768. This primarily affects the time for training and inference but not the results as much (as the VAD task does not require as much information as part of the feature vectors).

4.5 Training

The computed audio features and their corresponding labels are combined to create a PyTorch dataloader. As there might be data imbalance between the variety of datasets we incorporate, we make use of Lhotse's multiplexing sampler. The audio features are fed in the model described above in batches of 400 seconds (80 samples) for 25 epochs with a learning rate of 0.001 and based on the evaluation metrics described below, the best model checkpoint is used. We use a simple binary cross entropy loss with the Adam optimizer and validate our model on the development split every 3 epochs.

4.6 Evaluation

To evaluate the performance of the model, we use many metrics which include but are not limited to:

- False alarm rate: ratio of false positives and the total number of samples
- Missed detection rate: ratio of false negatives and the total number of samples
- Detection error rate: Sum of the false alarm and missed detection rate.

The model with the most optimum values of the above metrics on the development split is selected to predict speech intervals for the test split and subsequently, create a CutSet.

This step is crucial as our system needs to detect speech intervals for any new dataset and then create a CutSet to then be fed to libraries like icefall [2] and k2 [3] for downstream tasks such as speech recognition, speaker diarization, speaker verification, etc.

4.7 Technologies Used

The project utilizes PyTorch (and PyTorch Lightning) to orchestrate training and testing pipelines. For logging and monitoring the training process, weights and biases is used. Finally, as mentioned previously, Lhotse is used to prepare datasets and can be seamlessly integrated into the application.

With the project's objective of ensuring ease of use through simple import statements, the code is organized to facilitate this accessibility. It features a modular structure where various components are decoupled, promoting a flexible and streamlined code base.

5 Results

Through the following results, we aim to compare the performance of our VAD (feature extractors used for comparison are filter banks, Wave2Vec 2.0 and Robust HuBERT) with those of other commercially available VADs such as that of SpeechBrain [29], Silero [30] and Pyannote.

As shown, the performance varies between datasets. Our models (across all feature extractors) have lower false alarm rates across most datasets and are thus, more precise in their prediction of speech boundaries. However, our models tend to have higher missed detection rates and thus, a lower recall compared to other VADs (especially Speechbrain). However, the caveat is that the other VADs tend to predict longer speech segments which generally consist of a lot of ground-truth segments which in turn, reduces their missed detection rate. Here, the ground-truth segments refer to the actual segments of speech present in the dataset that have been manually labeled or annotated by human experts. These segments represent the true instances of speech within the audio data, serving as the reference or "ground truth" against which the performance of the VAD models is evaluated. The predicted speech segments can have a duration of over 100 seconds, even though the ground-truth segments are $\sim$3 seconds. This is a trade-off and appropriate models can be chosen based on the task at hand. Our VAD can also be used in conjunction with other VADs to refine the longer duration predictions that might include multiple ground-truth speech segments.

We also see in Table 6 and Table 7 that using filter banks with our VAD significantly reduces the false alarm rate for 8KHz datasets (that are upsampled to 16KHz). This is useful for VADs like Speechbrain that do not accept 8KHz audio.

Table 12 shows results for the Santa Barbara corpus which is a noisy dataset of spoken interactions in 8KHz. Our model isn't robust to noisy environments and thus, does not do as well as the other VADs.

Model	Det Err	FA	MD
speechbrain	**0.07**	0.07	**0**
silero	0.18	0.05	0.13
pyannote	0.18	**0.04**	0.14
ours (fbank)	0.09	0.07	0.02
ours (hubert)	0.09	0.06	0.03
ours (wav2vec2)	0.09	0.07	0.02

Table 2: Performance on TED-LIUM

Model	Det Err	FA	MD
speechbrain	0.22	0.08	0.14
silero	0.35	0.04	0.31
pyannote	0.25	**0.01**	0.24
ours (fbank)	0.19	0.07	0.12
ours (hubert)	0.19	0.05	0.14
ours (wav2vec2)	**0.17**	0.06	**0.11**

Table 3: Performance on ChiME-6

Model	Det Err	FA	MD
speechbrain	0.07	0.06	**0.01**
silero	0.08	0.03	0.05
pyannote	**0.05**	**0.03**	0.02
ours (fbank)	0.16	0.09	0.07
ours (hubert)	0.18	0.09	0.09
ours (wav2vec2)	0.17	0.09	0.08

Table 4: Performance on VoxConverse

Model	Det Err	FA	MD
speechbrain	0.5	0.5	**0**
silero	0.24	0.2	0.04
pyannote	0.32	0.31	0.01
ours (fbank)	0.24	0.14	0.1
ours (hubert)	0.23	0.12	0.11
ours (wav2vec2)	**0.22**	**0.12**	0.1

Table 5: Performance on AMI

Model	Det Err	FA	MD
speechbrain	-	-	-
silero	0.58	0.55	**0.03**
pyannote	0.58	0.53	0.05
ours (fbank)	**0.21**	**0.13**	0.08
ours (hubert)	0.31	0.24	0.07
ours (wav2vec2)	0.29	0.22	0.07

Table 6: Performance on Switchboard

Model	Det Err	FA	MD
speechbrain	-	-	-
silero	0.77	0.77	**0**
pyannote	0.75	0.75	0
ours (fbank)	**0.16**	**0.11**	0.05
ours (hubert)	0.35	0.32	0.03
ours (wav2vec2)	0.26	0.22	0.04

Table 7: Performance on Calhome English

5.1 DIHARD3 Domain results

We also measure the performance of our VAD (using the wav2vec 2.0 feature extractor) and that of Pyannote's across all the domains in the DIHARD-3 test set in Table 13. As Pyannote is optimized for the DIHARD-3 dataset, it does considerably well as compared to ours and other VADs.

6 Future Work

As shown, our VAD model is precise and diverse, but has a lower recall compared to a few other commercially available ones such as Speechbrain. Thus, in the future, we aim

Model	Det Err	FA	MD
speechbrain	0.06	0.06	**0**
silero	0.13	0.06	0.07
pyannote	0.08	0.06	0.02
ours (fbank)	0.06	0.04	0.02
ours (hubert)	0.06	0.04	0.02
ours (wav2vec2)	**0.05**	**0.04**	0.01

Table 8: Performance on Gale Arabic

Model	Det Err	FA	MD
speechbrain	0.15	0.14	**0.01**
silero	0.18	0.14	0.04
pyannote	0.19	0.14	0.05
ours (fbank)	0.13	0.07	0.06
ours (hubert)	0.11	0.06	0.05
ours (wav2vec2)	**0.1**	**0.06**	0.04

Table 9: Performance on Gale Mandarin

Model	Det Err	FA	MD
speechbrain	0.19	0.18	**0.01**
silero	0.15	0.08	0.07
pyannote	**0.07**	**0.03**	0.04
ours (fbank)	0.32	0.18	0.14
ours (hubert)	0.31	0.17	0.14
ours (wav2vec2)	0.32	0.18	0.14

Table 10: Performance on DIHARD-3

Model	Det Err	FA	MD
speechbrain	0.6	0.6	**0**
silero	0.47	0.44	0.03
pyannote	0.5	0.44	0.06
ours (fbank)	0.15	0.08	0.07
ours (hubert)	0.2	0.14	0.06
ours (wav2vec2)	**0.12**	**0.07**	0.05

Table 11: Performance on DiPCo

Model	Det Err	FA	MD
speechbrain	-	-	-
silero	0.17	0.12	0.05
pyannote	**0.14**	**0.08**	**0.06**
ours (fbank)	0.29	0.15	0.04
ours (hubert)	0.3	0.15	0.15
ours (wav2vec2)	0.29	0.15	0.14

Table 12: Performance on SB Corpus

to improve this shortcoming by relaxing and/or changing the loss function, implementing collar-aware training and experimenting with multi-task training to provide more context.

We also aim to add more diverse datasets, especially ones in noisy environments and add support for multiple channels, to further improve the model's robustness in real-world settings.

Furthermore, we aim to create an easy-to-use python package that can be installed and used on-the-fly, similar to it's counterparts. We would also improve upon the current functionality of providing manifest files as outputs (that can be easily consumed by Icefall and k2) to account for diverse combinations of recordings and supervisions.

Split	Det Err		FA		MD	
	Pyannote	*Ours*	*Pyannote*	*Ours*	*Pyannote*	*Ours*
audio books	**0.03**	0.3	**0.02**	0.17	**0.01**	0.13
broadcast interviews	**0.04**	0.3	**0.03**	0.17	**0.01**	0.13
clinical interviews	**0.07**	0.43	**0.02**	0.24	**0.05**	0.19
court proceedings	**0.03**	0.26	**0.02**	0.13	0.01	0.13
telephone conversations	**0.04**	0.16	**0.02**	0.08	**0.02**	0.08
map-based tasks	**0.03**	0.39	**0.02**	0.19	**0.01**	0.2
meetings	**0.09**	0.21	**0.04**	0.14	**0.05**	0.07
restaurant conversations	**0.1**	0.25	**0.05**	0.07	**0.05**	0.18
socio-linguistic field recs	**0.08**	0.31	**0.01**	0.17	**0.07**	0.14
socio-linguistic lab recs	**0.04**	0.26	**0.02**	0.17	**0.02**	0.09
web videos	**0.15**	0.36	**0.07**	0.18	**0.08**	0.17

Table 13: Results across different splits in DIHARD-3

7 Conclusion

In conclusion, our endeavor involves the release of an open-source universal voice activity detection model designed to seamlessly accommodate a range of speech settings. For this, we train models with a diverse mix of datasets which include single and multi-speaker environments. We provide extensive results and comparative studies across different datasets, feature extraction methods and training settings. We also compare the absolute performance of our models and that of commercially available VAD models and outline our strengths and weaknesses.

This model is intended for integration with widely used speech processing libraries such as icefall and k2 and is meticulously trained on varied datasets curated through the utilization of Lhotse.

References

[1] P. Zelasko, D. Povey, J. Y. Trmal, and S. Khudanpur, "Lhotse: a speech data representation library for the modern deep learning ecosystem," 2021.

[2] "Icefall." [Online]. Available: https://github.com/k2-fsa/icefall

[3] "K2: Fsa/fst algorithms, differentiable, with pytorch compatibility." [Online]. Available: https://github.com/k2-fsa/k2.

[4] "Depiction of cut with recordings and supervisions in lhotse." [Online].

Available: https://raw.githubusercontent.com/lhotse-speech/lhotse/master/docs/lhotse-cut-illustration.png

[5] A. Davis, S. Nordholm, and R. Togneri, "Statistical voice activity detection using low-variance spectrum estimation and an adaptive threshold," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 2, pp. 412–424, 2006.

[6] S. G. Tanyer and H. Ozer, "Voice activity detection in nonstationary noise," *IEEE Transactions on speech and audio processing*, vol. 8, no. 4, pp. 478–482, 2000.

[7] J. Sohn, "A statistical model-based voice activity detection," *IEEE Signal Processing Letters*, vol. 6, no. 1, p. 1 â 3, 1999. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-0032762471&doi=10.1109%2f97.736233&partnerID=40&md5=ae137e0d162c4d728cfd056084cd8346

[8] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng, "Deep speech: Scaling up end-to-end speech recognition," 2014.

[9] F. Eyben, F. Weninger, S. Squartini, and B. Schuller, "Real-life voice activity detection with lstm recurrent neural networks and an application to hollywood movies," 2013, Conference paper, p. 483 â 487, cited by: 197; All Open Access, Green Open Access. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84890443834&doi=10.1109%2fICASSP.2013.6637694&partnerID=40&md5=d2156ec9f04f44c73ee19dea5832f4fb

[10] L. Muda, M. Begam, and I. Elamvazuthi, "Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques," 2010.

[11] D. Vlaj, B. Kotnik, B. Horvat, and Z. Kačič, "A computationally efficient mel-filter bank vad algorithm for distributed speech recognition systems," *EURASIP Journal on Advances in Signal Processing*, vol. 2005, pp. 1–11, 2005.

[12] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," 2020.

[13] M. Kunsova and Z. Zajic, "Multitask detection of speaker changes, overlapping speech and voice activity using wav2vec 2.0," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, Jun. 2023. [Online]. Available: http://dx.doi.org/10.1109/ICASSP49357.2023.10094972

[14] M. Ravanelli and Y. Bengio, "Speaker recognition from raw waveform with sincnet," in *2018 IEEE spoken language technology workshop (SLT)*. IEEE, 2018, pp. 1021–1028.

[15] J. e. a. Carletta, "The ami meeting corpus: A pre-announcement," 2006.

[16] F. Hernandez, V. Nguyen, S. Ghannay, N. Tomashenko, and Y. Esteve, *TED-LIUM 3: Twice as Much Data and Corpus Repartition for Experiments on Speaker Adaptation*. Springer International Publishing, 2018, p. 198â208. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-99579-3_21

[17] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "Switchboard: Telephone speech corpus for research and development," in *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech and Signal Processing - Volume 1*, ser. ICASSP'92. USA: IEEE Computer Society, 1992, p. 517â520.

[18] J. S. Chung, J. Huh, A. Nagrani, T. Afouras, and A. Zisserman, "Spot the conversation: speaker diarisation in the wild," in *INTERSPEECH*, 2020.

[19] S. Watanabe, M. Mandel, J. Barker, E. Vincent, A. Arora, X. Chang, S. Khudanpur, V. Manohar, D. Povey, D. Raj, D. Snyder, A. S. Subramanian, J. Trmal, B. B. Yair, C. Boeddeker, Z. Ni, Y. Fujita, S. Horiguchi, N. Kanda, T. Yoshioka, and N. Ryant, "Chime-6 challenge:tackling multispeaker speech recognition for unsegmented recordings," 2020.

[20] N. Ryant, P. Singh, V. Krishnamohan, R. Varma, K. Church, C. Cieri, J. Du, S. Ganapathy, and M. Liberman, "The third dihard diarization challenge," 2021.

[21] M. Glenn, H. Lee, S. Strassel, and K. Maeda, "GALE Phase 2 Arabic Broadcast News Transcripts Part 1."

[22] ——, "GALE Phase 4 Chinese Broadcast News Transcripts."

[23] G. Z. Alexandra Canavan, David Graff, "Callhome american english speech."

[24] M. V. Segbroeck, Z. Ahmed, K. Kutsenko, C. Huerta, T. Nguyen, B. Hoffmeister, J. Trmal, M. Omologo, and R. Maas, "Dipco - dinner party corpus," in *Interspeech 2020*, 2019. [Online]. Available: https://www.amazon.science/publications/dipco-dinner-party-corpus

[25] W. L. C. C. M. S. A. T. R. E. Du Bois, John W. and N. Martey, "Santa barbara corpus of spoken american english," 2000.

[26] D. Snyder, G. Chen, and D. Povey, "Musan: A music, speech, and noise corpus," *arXiv preprint arXiv:1510.08484*, 2015.

[27] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," 2020.

[28] K. P. Huang, Y.-K. Fu, Y. Zhang, and H.-y. Lee, "Improving distortion robustness of self-supervised speech processing tasks with domain adaptation," *arXiv preprint arXiv:2203.16104*, 2022.

[29] M. Ravanelli, T. Parcollet, P. Plantinga, A. Rouhe, S. Cornell, L. Lugosch, C. Subakan, N. Dawalatabad, A. Heba, J. Zhong, J.-C. Chou, S.-L. Yeh, S.-W. Fu, C.-F. Liao, E. Rastorgueva, F. Grondin, W. Aris, H. Na, Y. Gao, R. D. Mori, and Y. Bengio, "SpeechBrain: A general-purpose speech toolkit," 2021, arXiv:2106.04624.

[30] S. Team, "Silero vad: pre-trained enterprise-grade voice activity detector (vad), number detector and language classifier," https://github.com/snakers4/silero-vad, 2021.

[31] H. Bredin, R. Yin, J. M. Coria, G. Gelly, P. Korshunov, M. Lavechin, D. Fustes, H. Titeux, W. Bouaziz, and M.-P. Gill, "pyannote.audio: neural building blocks for speaker diarization," 2019.

8 Appendix

Through the following results, we aim to show the change in VAD performance as we add different datasets in the training set. For convenience, we number models based on their training set, as shown in Table 14, and then compare their results on each dataset's test split. We have used the Wav2Vec 2.0 base model as a feature extractor and not added any noise.

Model	tedlium	chime	voxconverse	ami	swbd	gale ar	gale man	dihard
1	✓							
2	✓	✓						
3	✓	✓	✓					
4	✓	✓	✓	✓				
5	✓	✓	✓	✓	✓			
6	✓	✓	✓	✓	✓	✓	✓	
7	✓	✓	✓	✓	✓	✓	✓	✓

Table 14: Models and the datasets included in their training set

Table 15 shows that the performance for TED-LIUM remains stable as we add different datasets. This might be because it is a relatively easier dataset to learn due to a majority of single-speaker audio samples as opposed to datasets like CHiME-6, VoxConverse & AMI, which include multi-speaker conversations.

Performance on multi-speaker datasets is shown in Table 16, Table 17, Table 18 where the model learns to adapt to multi-speaker conversations as datasets with those attributes are added to the training set. Without those datasets, the performance is significantly reduced.

Table 19 shows that Switchboard does not perform as well. This is because it was upsampled from 8KHz to 16KHz which adds artificial noise that the model cannot handle until it is trained on similar datasets. It misclassifies a lot of silence as speech in this process and thus, has a very high false alarm rate. This is why we add a similar dataset such as Callhome English to the training set to further improve the false alarm rate. This is clearly shown in Table 6 and Table 7 with improved performance.

Gale Arabic and Mandarin are added to provide the model with data from different languages to adapt to language-specific nuances (if any).

CHiME-6, VoxConverse, AMI and DIHARD3 are notably challenging and diverse datasets. Consequently, the inclusion of additional datasets yields either consistent results or a gradual improvement in performance, as illustrated in Table 16, Table 17, Table 18 and Table 22. This is an encouraging trend, suggesting that incorporating more datasets enables our VAD model to effectively adapt to new and intricate settings. As a result, this adaptability proves valuable for researchers seeking a versatile and robust voice activity detection model.

Model	Det Err	FA	MD
1	**0.07**	**0.05**	**0.02**
2	0.07	0.04	0.03
3	0.07	0.05	0.02
4	0.08	0.06	0.02
5	0.08	0.05	0.03
6	0.07	0.05	0.02
7	0.07	0.05	0.02

Table 15: Performance on TED-LIUM

Model	Det Err	FA	MD
1	0.78	**0**	0.78
2	0.2	0.05	0.15
3	0.19	0.06	0.13
4	0.17	0.07	0.1
5	0.17	0.06	0.11
6	**0.16**	0.07	**0.09**
7	0.17	0.07	0.1

Table 16: Performance on CHiME-6

Model	Det Err	FA	MD
1	0.31	0.07	0.24
2	0.34	**0.06**	0.28
3	**0.2**	0.08	**0.12**
4	0.21	0.08	0.13
5	0.25	0.07	0.18
6	0.2	0.08	0.12
7	0.21	0.08	0.13

Table 17: Performance on VoxConverse

Model	Det Err	FA	MD
1	0.26	0.12	0.14
2	0.3	0.13	0.17
3	0.3	0.19	0.11
4	0.22	0.12	**0.1**
5	**0.21**	**0.1**	0.11
6	0.21	0.11	0.1
7	0.22	0.12	0.1

Table 18: Performance on AMI

Model	Det Err	FA	MD
1	**0.31**	0.2	0.11
2	0.32	**0.19**	0.13
3	0.34	0.25	0.09
4	0.35	0.26	0.09
5	0.38	0.33	**0.05**
6	0.39	0.32	0.07
7	0.38	0.32	0.06

Table 19: Performance on Switchboard

Model	Det Err	FA	MD
1	0.55	0.01	0.54
2	0.62	0.01	0.61
3	0.2	0.04	0.16
4	0.33	0.03	0.3
5	0.43	**0.02**	0.41
6	**0.07**	0.03	**0.04**
7	0.07	0.03	0.04

Table 20: Performance on Gale Arabic

Model	Det Err	FA	MD
1	0.59	0.03	0.57
2	0.67	**0.02**	0.65
3	0.3	0.06	0.24
4	0.35	0.06	0.29
5	0.39	0.05	0.34
6	0.16	0.04	0.12
7	**0.14**	0.06	**0.08**

Table 21: Performance on Gale Mandarin

Model	Det Err	FA	MD
1	0.47	0.11	0.36
2	0.5	**0.1**	0.4
3	0.38	0.15	0.23
4	0.41	0.14	0.27
5	0.44	0.12	0.32
6	**0.34**	0.17	**0.17**
7	0.34	0.17	0.17

Table 22: Performance on DIHARD3

Model	Det Err	FA	MD
1	0.19	0.04	0.15
2	0.25	0.05	0.2
3	0.3	0.22	0.08
4	0.11	0.05	**0.06**
5	**0.09**	**0.02**	0.07
6	0.12	0.05	0.07
7	0.12	0.06	0.06

Table 23: Performance on DiPCo

Model	Det Err	FA	MD
1	0.5	**0.09**	0.41
2	0.47	0.09	0.38
3	0.36	0.13	0.23
4	0.35	0.13	0.22
5	0.42	0.11	0.31
6	0.31	0.15	0.16
7	**0.3**	0.15	**0.15**

Table 24: Performance on SB corpus