BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2021
http://excel.fit.vutbr.cz

Personal Voice Activity Detection

Šimon Sedláček*

Abstract
This work aims to implement, test, and evaluate a speaker-conditioned voice activity detection method proposed by [1], originally called "Personal VAD". The method builds upon an LSTM based approach to voice activity detection and its purpose is to introduce a system that can reliably detect speech of a target speaker, while retaining the typical characteristics of a VAD system, mainly in terms of small model size, low latency, and low necessary computational resources. The system is trained to distinguish between three classes: non-speech, target speaker speech, and non-target speaker speech. For this purpose, the method utilizes speaker embeddings as a part of the input feature vector to represent the target speaker. Some of the more heavyweight personal VAD variants also make use of speaker verification scores issued to each frame based on the target embedding, resulting in a more robust system. In addition to the one scoring method presented in the original paper, two other scoring approaches are introduced, both outperforming the baseline method and improving the system's performance even for acoustically challenging conditions.

Keywords: Voice Activity Detection — Recurrent Neural Networks — Speaker Verification

Supplementary Material: N/A

*xsedla1h@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Many of today's personal devices such as mobile phones, tablets, etc., offer some voice command features to the user, most often in the form of a virtual personal assistant. To process the user's requests and commands, a speech recognition system is needed to process the incoming audio data and pass the transcribed speech to other components of the system.

These systems are however often quite complex and demanding in terms of computational resources. For personal devices such as mobile phones, where CPU time, memory, and especially battery life are limited, this can be a significant limitation if one wishes to run on-device speech recognition.

This problem can be addressed by simply triggering such components only when necessary – meaning only when the user is talking to the device. This is often realized by introducing a keyword detection model, however, in some applications and scenarios the users might benefit from a more seamless experience without the necessity of using a special keyword to trigger the system.

The method proposed by [1] aims to address this problem by expanding the classification capabilities of an LSTM-based voice activity detection (VAD) model, without necessarily increasing the resource demand. The system is trained to distinguish not only between speech and non-speech audio frames but also to detect and identify speech frames belonging to a particular target speaker. This method (originally called "Personal VAD") is proposed to be a potential replacement of the mentioned keyword detection systems, by only allowing the device owner's spoken voice commands to trigger the speech recognition system.

Such an approach has multiple advantages, one being that most speech recognition systems have to use a voice activity detection component anyway. Typically, it is used as a pre-processing/gating module that is responsible for filtering out the non-speech frames of the incoming audio data in order to help the more complex and resource-heavy downstream components achieve better accuracy. Additionally, by discarding non-speech audio frames, the overall computation time and power consumption of the whole system can be greatly reduced.

In order to use the personal VAD approach for on-device speech recognition scenarios, it would be desirable for it to retain some of the characteristics of a typical VAD system:

- A lightweight, accurate model with minimal latency and minimum computational resource requirements, so that battery life can be saved.
- A model that is able to operate accurately in acoustically challenging environments, including noisy and reverberant conditions.

Additionally, as we are dealing with an online classification scenario (and to minimize latency), it would be best if the resulting system could operate as a *streaming* model. This is why using a VAD architecture based on LSTM networks might be desirable. LSTM-based VAD architectures have become increasingly popular for sequential modeling of the VAD task, all that while showing state-of-the-art performance even in acoustically challenging conditions [2].

An LSTM-based model architecture is a good fit for the presented personal VAD task for multiple reasons:

- Recurrent neural networks have the implicit ability to work with temporal contexts in the input features.
- If trained accordingly, LSTM networks have the ability to process sequences of arbitrary length. This makes them suitable for online inference scenarios.
- LSTM-based models have shown state-of-the-art performance even for complex tasks such as speaker verification [3].

Now, not all methods presented in this work do meet the lightweight criteria for the system. That being said, some of the more heavy-weight solutions offer better performance and accuracy, so it is still worth exploring those particular approaches, as they might prove useful in situations, where the resource limits aren't as strict.

The rest of the paper is organized in the following manner: section 2 describes the approach taken to adapting the baseline VAD model to be able to detect a target speaker's speech and section 3 provides an overview of the proposed personal VAD model architectures, explaining their parameters, pros, and cons. Section 4 describes how the training and testing datasets were obtained and which measures were taken to ensure the robustness of the final systems. Lastly, section 5 describes the experiments performed using the trained models and discusses potential performance improvements achieved by altering the method of issuing speaker verification scores to each audio frame.

2. Detecting a target speaker's speech

To build a system, that will be able to reliably detect a target speaker's speech, it is first necessary to decide upon the strategy used to implement such a system.

Many state-of-the-art speaker recognition and diarization systems [3, 4] would suffice for this task (especially in terms of accuracy), however, there are some drawbacks.

First of all, typical diarization systems are designed to establish boundaries between *all* present speakers. Therefore, a lot of effort has to go towards determining the number of speakers in the recording, including extracting their representations, clustering, and then segmenting the original recording based on the calculated boundaries. For one target speaker, all this is unnecessary, as one can simply have the representation of the target speaker available beforehand and the system does not need to distinguish between all the non-target speakers.

Second of all, most RNN-based VAD systems can easily be used as streaming models, producing a VAD probability on the arrival of each individual frame. On the other hand, speaker diarization systems often have to work in a window-based, segment-based manner, sometimes requiring the full utterance before beginning the diarization process.

This is why adapting a simple VAD model to detect speech frames of a target speaker can be useful for some scenarios. For example the previously mentioned on-device speech recognition, or general online speech processing situations, where there's only one target.

To perform such adaptation, the system can be trained to *draw its attention* towards the target speaker by providing their representation along with the acoustic features. This would be done most often in the form of a speaker embedding (i-vectors [5], d-vectors [3]) obtained previously during the enrollment process.

Speaker representations are often obtained from systems much more complex than a typical VAD, so effectively, the aim is to teach the VAD to ”distill” some knowledge from these abstract representations and identify the target speaker based on their acoustic ”profile”.

Similar approaches have previously been adopted also in the domains of speech recognition [6], speech extraction [7, 8] and then consequently also in diarization [9].

2.1 Speaker verification method overview

Even though the previously cited methods mostly use i-vectors as their speaker embedding type of choice, the Personal VAD paper proposes the use of d-vectors. The speaker verification system used to extract the d-vector embeddings was introduced in [3].

This text-independent speaker verification system is based on a three-layer LSTM neural network architecture and trained to produce one representative 256-dimensional speaker embedding for a sliding window of a fixed size, typically around 160 frames.

These embeddings are by default extracted from the hidden state activations of the last LSTM layer of the network – right after processing the last frame of the current window – and L2 normalized. Then, to produce a baseline speaker embedding, these window-level d-vectors are averaged over the whole enrollment utterance as shown in Fig. 1.

To measure the similarity between two embedding vectors, cosine similarity can be used:

$$similarity = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \cdot \|\mathbf{B}\|},$$

where $\mathbf{A}$ and $\mathbf{B}$ are the two embedding vectors respectively. This score can be interpreted as the speaker verification score for the two embedding vectors.

The reason, why d-vectors are used rather than i-vectors is that some of the Personal VAD architecture variants require continuous frame-level embedding extraction in order to issue a speaker verification score to each particular frame. This can be achieved by changing the original window-level inference of the d-vector extractor to continuously produce embedding vectors for each individual frame. In other words, the system's inference is changed from sequence-to-vector to sequence-to-sequence, while also refraining from limiting the sequences in terms of length. The system of course might not perform optimally, as it was originally trained for window-level inference, as shown in Fig. 1. However, the conducted experiments in section 5 indicate that this approach can be adapted without degrading the quality of the extracted scores too significantly. Section 5.2 then explores two other frame scoring methods, where the default window-level embedding inference is used.

The speaker verification system implementation used in this work is called "Resemblyzer"[1] and is freely available on Github as a community implementation of [3].

[1] https://github.com/resemble-ai/Resemblyzer

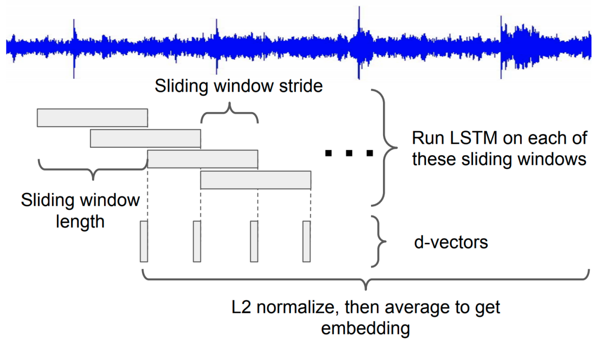
Sliding window stride
Run LSTM on each of these sliding windows
Sliding window length
d-vectors
L2 normalize, then average to get embedding

Figure 1. Window-level d-vector extraction. Diagram taken from [3].

3. System architecture

The Personal VAD system has two main components. The first and main part is the LSTM-based VAD model, which will be trained to classify the input speech frames into the three previously mentioned categories: non-speech (`ns`), target speaker speech (`tss`), and non-target speaker speech (`ntss`).

The second part is a speaker verification system trained to produce speaker embeddings for the input audio. Those will be used either to train the resulting system to recognize the target speaker given the acoustic features and their embedding vector, or to provide a method of assigning a speaker verification score to each individual frame.

In the following sections 3.1, 3.2, 3.3 and 3.4, four different Personal VAD architectures are described, each using a different set of input features. All four architectures share the same neural network basis: a 2-layer LSTM network with 64 cells, followed by one fully connected layer of 64 neurons. The input and output layer dimensions vary depending on the particular architecture variant. All model variants also use the same acoustic features: 40-dimensional log mel-filterbank features of 25ms width and 10ms step.

3.1 System 1: Score combination (SC)

The baseline architecture for the personal VAD task consists of a classical VAD model, which takes the 40-dimensional acoustic features features as input and produces speech probability p_t^s for each incoming frame $\mathbf{x}_t$ at each time step: $p_t^s = f_{VAD}(\mathbf{x}_t)$. Each individual frame is also passed through the speaker verification model and a d-vector embedding $\mathbf{e}_t$ is obtained for that frame. This d-vector is then compared with the target speaker embedding $\mathbf{e}_{\texttt{target}}$ using cosine similarity, thus obtaining a speaker verification score for each frame: $s_t = \cos(\mathbf{e}_t, \mathbf{e}_{\texttt{target}})$.

The obtained score is then combined with the speech probability p_s to produce unnormalized personal VAD probabilities z_t^k for each class k:

$$z_t^k = \begin{cases} s_t \cdot p_t^s & \text{for } k = \texttt{tss}, \\ (1-s_t) \cdot p_t^s & \text{for } k = \texttt{ntss}, \\ 1 - p_t^s & \text{for } k = \texttt{ns}, \end{cases} \tag{1}$$

It should be noted that due to some specific implementation decisions made in the d-vector extractor that was used in the experiments, the resulting d-vectors are limited to positive values in all dimensions. Therefore, the resulting cosine similarity scores will always lie in the $< 0, 1 >$ interval.

It is apparent, that this approach represents a very naive solution to the problem. However, to keep consistency with the experiments presented in the original paper, experiments with the SC system were conducted nonetheless.

It would most likely be better to simply treat this system as a twofold classification problem. First, the input frames would be classified as speech or non-speech. Afterward, the speech frames would be classified as either `tss` or `ntss` based on the cosine similarity score value, for example using a simple logistic regression classifier. The next architecture presented in Section 3.2 is intuitively supposed to perform the same task, just without the necessity of training another classifier.

3.2 System 2: Score conditioned training (ST)

The ST architecture expands on the SC by combining the acoustic features $\mathbf{x}_t$ with the speaker verification score s_t into one 41-dimensional feature vector $\hat{\mathbf{x}}_t = [\mathbf{x}_t, s_t]$. The system is then trained using these features to directly produce Personal VAD class probabilities – non-speech, non-target speaker speech and target-speaker speech: $\mathbf{z}_t = [z_t^{\texttt{ns}}, z_t^{\texttt{ntss}}, z_t^{\texttt{tss}}]$. This system is expected to perform better than the SC, as it learns to infer the output probabilities from the input rather than using a set-in-stone transformation function as in 3.1.

3.3 System 3: Embedding conditioned training (ET)

The ET architecture represents the desired solution of the personal voice activity detection problem, as it does not require a speaker verification system at runtime, making it a very lightweight solution. This architecture combines the target speaker's enrollment embedding $\mathbf{e}_{\texttt{target}}$ with the acoustic features into one 296-dimensional feature vector: $\hat{\mathbf{x}}_t = [\mathbf{x}_t, \mathbf{e}_{\texttt{target}}]$.

This system is expected to learn the relationship between the input acoustic features and the target embedding, distilling this knowledge for classification purposes.

3.4 System 4: Score and embedding conditioned training (SET)

The last Personal VAD architecture combines the characteristics of the previous two systems. The system input consists of the acoustic features, the target speaker embedding as well as the speaker verification score for the current frame, giving us a 297-dimensional input feature vector: $\hat{\mathbf{x}}_t = [\mathbf{x}_t, s_t, \mathbf{e}_{\texttt{target}}]$. Even though that it is expected that this architecture will provide the best results of the four, it still requires a running speaker verification model at runtime, so that the frame scoring can be performed.

3.5 Loss functions

Because personal VAD represents a ternary classification problem, it is possible to train the model by minimizing the following cross-entropy loss, (also known as the *softmax loss*):

$$L_{\text{ce}}(y, \mathbf{z}) = -\log \frac{\exp(z^y)}{\sum_k \exp(z^k)}, \tag{2}$$

where $\mathbf{z}$ is the vector of pre-softmax network outputs for each class, y denotes the index of the target label, z^y denotes the system's output for the target class and z^k denotes the system's output for the k-th class.

The Personal VAD paper [1] also proposes the use of a new loss function, the *weighted pairwise loss*, which allows issuing different weights to each class pair. In doing so, confusion errors between certain classes can have a lesser impact on the system's performance:

$$L_{\text{wpl}}(y, \mathbf{z}) = -\mathbb{E}_{k \neq y} \left[w_{<k,y>} \cdot \log \frac{\exp(z^y)}{\exp(z^y) + \exp(z^k)} \right], \tag{3}$$

where $w_{<k,y>}$ is the weight between classes k and y. By setting the weight of `<ns,ntss>` to a smaller value than `<tss,ntss>` or `<ns,tss>`, the system should focus more on distinguishing the target speaker's speech from the other two classes, more so than preoccupying itself with `<ns,ntss>` confusion errors.

4. Creating the dataset

In order to train the system, it was necessary to find a dataset that would:

1. contain enrollment utterances for each speaker to extract their embedding vectors,
2. consist of utterances, where speakers would be taking turns.

For this purpose, the publicly available LibriSpeech [10] dataset was used and modified to contain speaker turns.

The LibriSpeech dataset is a corpus of roughly 980 hours of recorded read English speech and contains speech utterances of 2484 different speakers, with on average 25 – 30 minutes of speech per speaker in the main three subcorpuses.

4.1 Generating training and testing data

To simulate speaker turns, n utterances are randomly chosen, each coming from a different speaker. The parameter n comes from a uniform distribution:

$$n \sim \text{Uniform}(1,\ 3).$$

These randomly chosen utterances are then concatenated, as are their ground truth annotations obtained from [11] originally extracted using the Montreal Forced Aligner [12]. Additionally, one of the speakers present in the resulting utterance is chosen and marked as the target.

Using this method, approximately 140 thousand concatenated utterances from 2338 different speakers were generated for the training dataset. All these utterances come from the 100-hour, 360-hour, and 500-hour LibriSpeech subsets respectively and no utterance was used twice. The testing set was obtained using the `dev` and `test` LibriSpeech subsets, totaling at around 5500 concatenated utterances from 146 different speakers.

The training and testing sets are completely disjoint both in terms of used utterances and in terms of speakers. Both sets were augmented to four times their original size using the approach described in section 4.2.

4.2 Data augmentation

One of the requirements for a quality voice activity detection system is its ability to perform well even in acoustically challenging environments, including different reverberation conditions and high levels of background noise. Therefore it is desirable to augment the training data to match these potential conditions so that the model can learn to account for them.

To augment the data, the MUSAN [13] corpus was used along with a set of room impulse responses from [14]. The augmentation strategy used in this work is similar to [15]. Each clean concatenated utterance is replicated three times using:

1. **Reverb** – a room impulse response is chosen randomly and applied to the clean utterance.
2. **Noise** – random background noises are added to the clean track at one-second intervals, at volumes ranging from 0 to 15 dB SNR.
3. **Music** – an instrumental music piece is chosen randomly from MUSAN and added to the clean utterance at volumes ranging from 5 to 15 dB SNR.

The augmentation strategy is the same for both the training and the testing data, enlarging both datasets to four times their original size.

5. Experiments

All four architectures were implemented using the PyTorch[2] toolkit and trained on the augmented dataset described in section 4.1. For data augmentation, the Kaldi speech recognition toolkit [16] was used. During training, the Adam optimizer was used with a scheduled learning rate, ranging from 1×10^{-3} to 1×10^{-5}. The computational resources necessary for training the models were kindly provided by MetaCentrum[3].

To evaluate the system performance, the *average precision* (AP) metric was used for each individual class. AP corresponds to the area under the precision-recall curve for different classification thresholds. Afterwards, *mean average precision* (mAP) was computed over all classes, adopting the *micro-mean*[4] approach to account for class imbalance.

5.1 Comparing the architectures

After training, each individual system was evaluated using the previously unseen testing dataset described in 4. All evaluation experiments were run both for clean speech only and for the whole augmented validation dataset. The results can be seen in Table 1.

It is apparent that the ST, ET, and SET architectures outperform the SC baseline system in all aspects except the AP for the `ns` class. That being said, the most important statistic for the personal VAD task is the AP score for the `tss` class, as, during inference, both the non-speech and non-target speaker speech frames will be discarded.

The SET architecture has outperformed both the ST and the ET architectures in this aspect, achieving an AP score of 0.960 for clean and 0.934 for augmented `tss`.

The ET architecture, although it does not use frame scoring for classification and relies solely on the target speaker embedding, reached an AP score of 0.938

[2] https://pytorch.org/
[3] https://metavo.metacentrum.cz/
[4] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.average_precision_score.html

Architecture	Loss	Clean				Augmented			
		`ns`	`ntss`	`tss`	**mAP**	`ns`	`ntss`	`tss`	**mAP**
SC – baseline	-	0.948	0.845	0.864	0.825	0.915	0.775	0.811	0.796
ST	CE	0.932	0.913	0.913	0.914	0.889	0.855	0.857	0.857
ET	CE	0.936	0.946	0.938	0.942	0.897	0.925	0.916	0.918
SET	CE	0.929	0.961	0.960	0.957	0.889	0.937	0.934	0.929
ET	WPL	0.931	0.952	0.946	0.947	0.893	0.931	0.924	0.923

Table 1. Average precision score comparison of different personal VAD architectures for clean and augmented speech. Class labels: `ns` for non-speech, `ntss` for non-target speaker speech and `tss` for target speaker speech. CE is the cross-entropy loss, WPL denotes the weighted pairwise loss.

for clean and 0.916 for augmented `tss`, even outperforming the ST model in this comparison. This shows, that frame scoring is not necessary to achieve good classification results and that this lightweight model represents a viable option for a personal VAD system.

The ST architecture ended up falling behind both the ET and the SET architectures, reaching only 0.914 mAP for clean and 0.857 mAP for the augmented speech set. One of the reasons may be the poor adaptation of the used speaker verification system for streaming frame-level inference. This aspect is further explored in section 5.2.

The ET architecture was also trained using the *weighted pairwise loss*, with weight values for `<tss, ns>` and `<tss, ntss>` pairs set to 1.0 and the weight for `<ns, ntss>` set to 0.5. Other `<ns, ntss>` weight values were explored, but the value of 0.5 ended up showing best performance, improving the AP score for `tss` by 0.8% absolute. This indicates that the weighted pairwise loss indeed has a positive (albeit rather marginal) effect on the system's ability to detect `tss`. The results can be seen in the last row of Table 1.

5.2 About frame scoring...

In the previous section, all experiments featuring the SC, ST, and SET architectures used the frame-level speaker verification scoring approach suggested by [1]. However, as it was previously established, the speaker verification system used to extract d-vectors from the input features was trained to produce one embedding vector based on a sliding window of 160 frames with an arbitrary overlap between the individual windows.

Just to reiterate, the baseline scoring approach used in the previous experiments required the d-vector extractor to be adapted for continuous frame-level inference, producing one embedding vector for each individual frame.

I decided to deviate from the original paper's direction in this experiment and tried altering the manner in which the input frames are scored. Both methods I tried required me to refrain from the original pure streaming frame-level inference of the personal VAD in the sense that the speaker verification scores are now calculated from window-level embedding vectors, as depicted in Fig. 1. The length of the sliding window is 160 frames and the step between the individual windows is 40 frames.

Now the two scoring approaches are the following:

1. Extract window-level d-vectors. One embedding vector is obtained for each sliding window position. Compute the scores for the extracted embeddings and span the scores across the utterance. The result is a constant score value for each 40 frames in between the adjacent embedding vectors.
2. Extract the d-vectors in the same manner, but instead of leaving the score value constant for each 40 frames, linearly interpolate the adjacent score values within the 40 frame step window. This is to simulate the "movement" in the speaker verification scores if the scores were computed for each frame individually.

An illustration of how these two scoring methods look in comparison to the baseline approach can be seen in Fig. 2.

Both scoring method alterations led to significant performance improvements for both clean and noisy speech as seen in Table 2, where the three scoring variants are compared. The ST architecture is used for the comparison. With `tss` AP scores of 0.979 for clean and 0.953 for augmented speech, the *partially-constant* scoring method outperforms every other presented architecture and even surpasses the 0.970/0.938 (clean/augmented) AP scores for `tss` of the most powerful SET architecture presented in [1].

The reason for this improvement is that the d-vector extractor indeed does benefit from working with limited context windows, resetting its LSTM state every time a new d-vector is extracted. This in turn results in d-vectors, whose discriminative properties are much better than those of the d-vectors extracted by the baseline frame-level approach.

Scoring method	Clean				Augmented			
	`ns`	`ntss`	`tss`	**mAP**	`ns`	`ntss`	`tss`	**mAP**
Frame-level, baseline	0.932	0.913	0.913	0.914	0.889	0.855	0.857	0.857
Window-level, partially constant	0.931	0.977	0.979	0.972	0.888	0.950	0.953	0.942
Window-level, linearly interpolated	0.927	0.972	0.972	0.967	0.889	0.944	0.947	0.937

Table 2. Performance comparison of different frame scoring methods, each used in conjunction with the ST architecture.

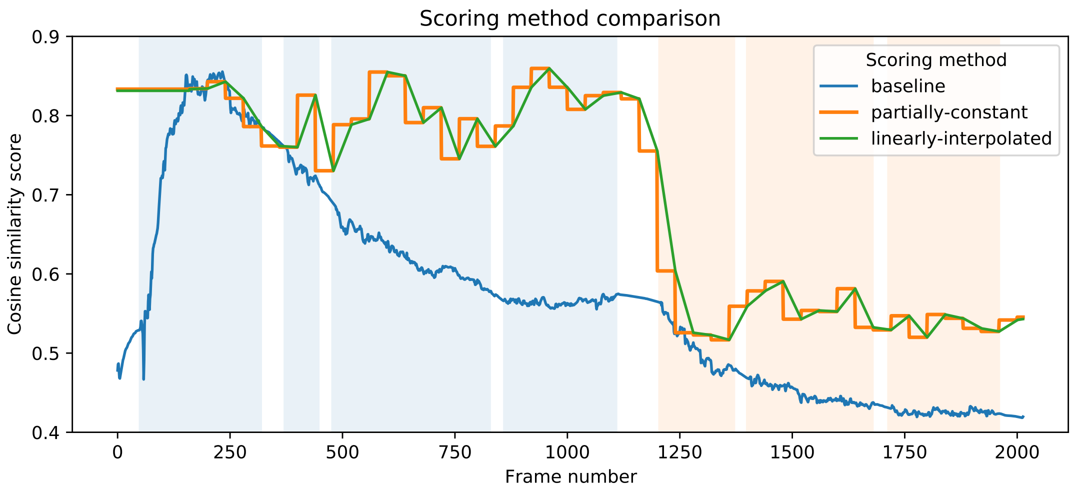

Figure 2. An exemplary comparison of the three scoring methods. Blue and orange areas denote `tss` and `ntss` segments, respectively.

Both these scoring alterations require even more computational resources than the baseline method, as each frame is processed up to four times to obtain window-level embeddings. This process however can easily be parallelized and for applications, where resources are not as limited, it can be a useful way of making the target speaker speech detection more robust.

6. Conclusions

In this work, I implemented and evaluated four different LSTM-based real-time target speaker voice activity detection models proposed by [1], each using a different set of input features to draw the system's attention to the speech of a target speaker. The architectures were trained on an augmented version of a 960-hour set of concatenated utterances, generated from the LibriSpeech dataset to simulate speaker turns.

The most powerful SET architecture achieved an average precision (AP) score of 0.960 and 0.934 for clean and augmented target speaker speech respectively. The most lightweight ET architecture trained with the custom *weighted pairwise loss* also shows good results, achieving 0.946 and 0.924 AP for clean and augmented target speaker speech respectively, indicating potential usefulness for scenarios, where computational resources are limited.

Additionally, I propose two alterations to the baseline frame scoring method, both outperforming the original method, the better of the two raising the AP score for target speaker speech by 6.6% absolute for clean and by 9.6% absolute for augmented speech.

Acknowledgements

I would like to thank my supervisor, Ing. Ján Švec for his support, advice, and valuable remarks while working on this project.

Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA LM2018140) provided within the program Projects of Large Research, Development and Innovations Infrastructures.

References

[1] Shaojin Ding, Quan Wang, Shuo-Yiin Chang, Li Wan, and Ignacio Lopez Moreno. Personal VAD: Speaker-Conditioned Voice Activity Detection. In *Proc. Odyssey 2020 The Speaker and Language Recognition Workshop*, pages 433–439, 2020.

[2] 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. In *2013 IEEE*

International Conference on Acoustics, Speech and Signal Processing, pages 483–487, 2013.

[3] Li Wan, Quan Wang, Alan Papir, and Ignacio Lopez Moreno. Generalized end-to-end loss for speaker verification, 2020.

[4] Quan Wang, Carlton Downey, Li Wan, Philip Andrew Mansfield, and Ignacio Lopz Moreno. Speaker diarization with lstm. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5239–5243. IEEE, 2018.

[5] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798, 2011.

[6] Naoyuki Kanda, Shota Horiguchi, Ryoichi Takashima, Yusuke Fujita, Kenji Nagamatsu, and Shinji Watanabe. Auxiliary interference speaker loss for target-speaker speech recognition. *CoRR*, abs/1906.10876, 2019.

[7] Quan Wang, Hannah Muckenhirn, Kevin Wilson, Prashant Sridhar, Zelin Wu, John R. Hershey, Rif A. Saurous, Ron J. Weiss, Ye Jia, and Ignacio Lopez Moreno. VoiceFilter: Targeted Voice Separation by Speaker-Conditioned Spectrogram Masking. In *Proc. Interspeech 2019*, pages 2728–2732, 2019.

[8] M. Delcroix, K. Zmolikova, K. Kinoshita, A. Ogawa, and T. Nakatani. Single channel target speaker extraction and recognition with speaker beam. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5554–5558, 2018.

[9] Ivan Medennikov, Maxim Korenevsky, Tatiana Prisyach, Yuri Khokhlov, Mariya Korenevskaya, Ivan Sorokin, Tatiana Timofeeva, Anton Mitrofanov, Andrei Andrusenko, Ivan Podluzhny, and et al. Target-speaker voice activity detection: A novel approach for multi-speaker diarization in a dinner party scenario. *Interspeech 2020*, Oct 2020.

[10] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: An asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, 04 2015.

[11] Loren Lugosch, Mirco Ravanelli, Patrick Ignoto, Vikrant Singh Tomar, and Yoshua Bengio. Speech Model Pre-Training for End-to-End Spoken Language Understanding. In *Proc. Interspeech 2019*, pages 814–818, 2019.

[12] Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. Montreal forced aligner: Trainable text-speech alignment using kaldi. In *Proc. Interspeech 2017*, pages 498–502, 2017.

[13] David Snyder, Guoguo Chen, and Daniel Povey. MUSAN: A music, speech, and noise corpus. *CoRR*, abs/1510.08484, 2015.

[14] Tom Ko, Vijayaditya Peddinti, Daniel Povey, Michael L. Seltzer, and Sanjeev Khudanpur. A study on data augmentation of reverberant speech for robust speech recognition. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*, pages 5220–5224. IEEE, 2017.

[15] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. X-vectors: Robust DNN embeddings for speaker recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018*, pages 5329–5333. IEEE, 2018.

[16] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011. IEEE Catalog No.: CFP11SRW-USB.