A LIGHTWEIGHT DUAL-STAGE FRAMEWORK FOR PERSONALIZED SPEECH ENHANCEMENT BASED ON DEEPFILTERNET2

Thomas Serre, Mathieu Fontaine, Éric Benhaim, Geoffroy Dutour, Slim Essid

A LIGHTWEIGHT DUAL-STAGE FRAMEWORK FOR PERSONALIZED SPEECH ENHANCEMENT BASED ON DEEPFILTERNET2

Thomas Serre[⋆†]*, Mathieu Fontaine*[⋆]*, Éric Benhaim*[†]*, Geoffroy Dutour*[†]*, Slim Essid*[⋆]

[†] Orosound, Signal Processing Lab [⋆]LTCI, Télécom Paris, Institut Polytechnique de Paris

ABSTRACT

Isolating the desired speaker's voice amidst multiple speakers in a noisy acoustic context is a challenging task. Personalized speech enhancement (PSE) endeavours to achieve this by leveraging prior knowledge of the speaker's voice. Recent research efforts have yielded promising PSE models, albeit often accompanied by computationally intensive architectures, unsuitable for resource-constrained embedded devices. In this paper, we introduce a novel method to personalize a lightweight dual-stage Speech Enhancement (SE) model and implement it within DeepFilterNet2, a SE model renowned for its state-of-the-art performance. We seek an optimal integration of speaker information within the model, exploring different positions for the integration of the speaker embeddings within the dual-stage enhancement architecture. We also investigate a tailored training strategy when adapting DeepFilterNet2 to a PSE task. We show that our personalization method greatly improves the performances of DeepFilterNet2 while preserving minimal computational overhead.

Index Terms— Target speech extraction, speech enhancement, real-time.

1. INTRODUCTION

Personalized speech enhancement (PSE) aims to extract the voice of a speaker with prior information on their voice. Such systems become very interesting when it comes to making a call in a crowded environment, working from home or even enhancing the voice of hearing aid users.

Over the years, deep neural network (DNN) based speech enhancement frameworks have emerged and achieved superior results, even in particularly noisy environments. However, such architectures usually perform poorly in an environment where interfering voices overlap with the voice of interest. PSE is therefore well suited for this task, as it benefits from prior information on the target speaker. This information is usually extracted from a speaker-specific enrollment clip using an encoder model. Then, it is fed to a downstream speech enhancement model and used as a cue to identify the target voice in the noisy content. These two models can either be trained jointly [1, 2, 3, 4] or separately [5, 6, 7]. Frameworks like VoiceFilter [5], SpEx [1], SpeakerBeam [8] or pPercepNet [6] paved the way for the PSE task showing its superiority to standard SE, especially with interfering voices.

Recently, the multi stage approach has been introduced to PSE after showing great results in speech enhancement [9]. The task is thus no longer performed as one task, but it is divided into several sub tasks instead. Ju, Yukai, et al. showed the benefits of the multistage approach in PSE by introducing the TEA-PSE [10] composed of a sub network called the MAG-Net, that estimates the clean magnitude, and another, the COM-Net, that estimates the clean complex spectrogram.

Although the multistage approach has led to smaller and more effective architectures in standard SE, such as GaG-Net [11] or DeepFilterNet2 [12], recent dual-stage models in PSE tend to be much larger. Many studies have worked on lighter PSE models that can be used in real-time on embedded devices [13, 14, 15]. However, the approach of adapting a lightweight dual-stage SE framework to PSE has not been considered yet, despite being critical for PSE on embedded devices, where system complexity must be controlled.

In this work, we propose a method to adapt a dual stage state-of-the-art SE model to the PSE task. We apply this method to DeepFilterNet2, a lightweight dual-stage framework renowned for its performance and its compact architecture. We seek an optimal integration of speaker information within the model, by exploring different positions for the integration of the speaker embeddings within the dual-stage enhancement architecture. We demonstrate that our personalization approach significantly enhances the performance of DeepFilterNet2[1] while maintaining a negligible increase in computational resource utilization.

2. PROPOSED METHOD

2.1. Speaker Encoder: ECAPA-TDNN

The speaker encoder is used to encode the acoustic information of the target speaker into an embedding. Given an observation x_{obs}, the speaker encoder transforms it into a vector $x_e \in R^D$, where D is the dimension of the latent space of the embedding. This process is summarized as $x_e = E(x_{obs})$ where E corresponds to the speaker encoder, and x_e to the

[1] Audio demo available at: http://pdeepfilternet2.github.io/

embedding. In this work, we use the ECAPA-TDNN [16] as the speaker encoder. This framework achieves state-of-the-art results in speaker recognition and is widely used in PSE. Greatly inspired by the X-vector [17], it consists of Time Delay Neural Network (TDNN) layers followed by an attention pooling layer. In our case, the encoder is trained before the enhancement network training, and its weights are frozen so that they remain unchanged during the training of the enhancement model.

2.2. Enhancement Network: DeepFilterNet2

2.2.1. DeepFilterNet2

DeepFilterNet2 [12] is a very light framework that achieves state-of-the-art results in speech enhancement. Based on a U-Net type of architecture, this framework is composed of two separated blocks. The first block performs a coarse estimation using Equivalent Rectangular Bandwidth (ERB) features as input, and the second block performs a finer estimation using complex domain spectrograms and Deep Filtering (DF). The latter is applied to the output of the coarse estimation, thus instantiating the dual-stage framework. In this work, we mainly focus on the encoder, which can be summarised as follows:

$$X_{enc}(k) = F_{enc}(X_{erb}(k,b), X_{df}(k, f_{df})) \quad (1)$$

where X_{erb} is the ERB feature, X_{df} the complex feature and F_{enc} the encoder of DeepFilterNet2. Then, the first stage estimates only the speech envelope and the second stage performs Deep Filtering on the output of stage 1. The filter coefficients are generated from the lower part of the complex spectrogram and $f_{df} = 5kHz$.

2.2.2. Personalized model: pDeepFilterNet2

We propose pDeepFilterNet2, the personalized version of DeepFilterNet2 [12]. The personalization of this framework is done by introducing the embedding of the pre-trained ECAPA-TDNN into the speech enhancement model. We thus investigate several locations to add the embedding. As most of the feature extraction and analysis occurs within the encoder, we mainly focus on the encoder. More precisely, we propose two architectures: the unified encoder and the dual encoder (see Fig. 1).

Unified encoder. Our first intuition is to simply concatenate the embedding where the two branches of the encoder are joined, as shown on the top schematics of Fig. 1. This is a straightforward approach that allows the speaker information to be used by both branches while minimizing the computational cost. The encoder can thus be modelled as follows.

$$X_{enc}(k) = F_{enc}(X_{erb}(k,b), X_{df}(k, f_{df}), x_e) \quad (2)$$

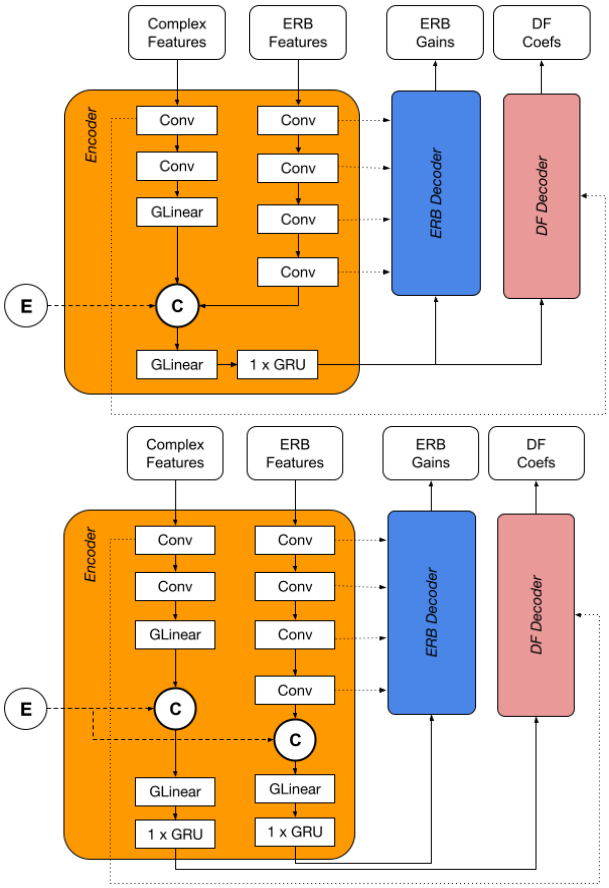

Fig. 1: Personalized DeepFilterNet2 with unified encoder (top) - Personalized DeepFilterNet2 with dual encoder (bottom) - **E** represents the embedding and **C** is the concatenation operation.

where x_e corresponds to the embedding. However, this approach does not allow us to understand what branch benefits the most from the embedding. Therefore, we also propose a dual encoder version to better leverage the embeddings.

Dual encoder. On the bottom schematics of Fig. 1, we propose the dual encoder version of the personalized DeepFilterNet2, we call it pDeepFilterNet2^D. Splitting the encoder into two independent branches allows us to get a fair comparison between adding the embeddings in the ERB branch, or in the DF branch. That way, we can understand how the embedding is used and where its addition is the most effective. The dual encoder is modelled as follows.

$$X_{ERB}(k) = F_{ERB\_enc}(X_{erb}(k,b), x_e) \quad (3)$$

$$X_{DF}(k) = F_{DF\_enc}(X_{df}(k, f_{df}), x_e) \quad (4)$$

Thus, this architecture allows us to test three possibilities: adding the embedding in both branches, in the ERB branch only, and in the DF branch only. These three models are respectively named as follows: pDeepFilterNet$^D_{both}$, pDeepFilterNet$^D_{erb}$ and pDeepFilterNet$^D_{df}$.

2.2.3. Loss function

The training objective remains almost the same as the one used for the training of DeepFilterNet2. It consists of a spectral loss $\mathcal{L}_{spec}$, a multi-resolution loss $\mathcal{L}_{MR}$ and an over-suppression loss $\mathcal{L}_{OS}$ [18]. $\mathcal{L}_{MR}$ is computed for the 4 following windows {5,10,20,40} ms. The overall loss can thus be summarized as follows.

$$\mathcal{L} = \lambda_{spec}\mathcal{L}_{spec} + \lambda_{MR}\mathcal{L}_{MR} + \lambda_{OS}\mathcal{L}_{OS} \tag{5}$$

where λ_{spec}, λ_{MR} and λ_{OS} correspond to loss weights which are found during training.

3. EXPERIMENTS

3.1. Speaker Encoder training

In this work, we do not focus on the training of the speaker encoder. Therefore we simply use the pretrained ECAPA-TDNN[2] and its weights are frozen during the training of pDeepFilterNet2. The pretrained version was trained on VoxCeleb2 and achieved 0.96% EER on the VoxCeleb1 test set. The dimension of the embeddings is set to $D = 192$.

3.2. pDeepFilterNet training

3.2.1. Dataset

For training set generation, we used the DNS5 personalized dataset [19] for target speech and noise, and we used Mozilla Common Voice (MCV) [20] for interfering speech. The DNS5 personalized dataset contains 3230 speakers with a total duration of 750 hours. However, when looking at some samples of clean speech in the dataset, we found out that some recordings contained several speakers. This problem was mentioned in [21] and we used the same method to remove unwanted speakers. We reserved VCTK data for test set generation resulting in 2448 target speakers in the training set. We then completed our training with the MCV corpus for interfering speech. We used 261 hours with 7069 different voices. We generated 950 hours in total and the excerpts were split into three categories: *target+noise*, *target+interfering* and *voice+interfering+noise*. The distribution of those excerpts in the dataset is respectively 20%, 30% and 50%. The SNR and SIR are drawn from a Gaussian distribution in [-5; 35] dB and [-5, 25] dB respectively. Finally, we reserve around 50 hours of the generated dataset for validation.

3.2.2. Training setup

The training procedure is very similar to the one of DeepFilterNet2. The model was trained with the Adam optimizer with the same learning rate and weight decay as in [12]. We also used batch scheduling from 8 to 128 to speed up the convergence. We perform early stopping on the validation loss with a patience of 15 epochs. The loss factors are set like in the original paper: $\lambda_{spec} = 1e3$ and $\lambda_{MR} = 5e2$, and $\lambda_{OS} = 5e2$. We also keep the same computational parameters by choosing 20-ms windows, an overlap of 50%, and a lookahead of 2 frames resulting in a 40-ms lookahead. Finally, we also take 32 ERB bands, $f_{df} = 5$ kHz, and $N = 5$ for the Deep Filtering order.

[2]https://github.com/TaoRuijie/ECAPA-TDNN

4. RESULTS

We evaluate our models on two test sets. The first one is synthetically generated and the second one is the DNS5 blind test set. To generate our test set, we use the VCTK corpus for both target speech and interfering speech, and DNS5 challenge for the additive noise. We equally generate *target+noise*, *target+interfering* and *voice+interfering+noise* excerpts. The SNR and the SIR are uniformly drawn in [-5, 35] dB and [-5, 25] dB respectively. In total, we generate 3600 files of 5 s.

4.1. Synthetic test set

The results on the synthetic test set are featured in table 1. We thus compare the four personalization methods mentioned in 2.2.2 to the non-personalized version of DeepFilterNet2 and we also compare the personalized models together. We use the same metrics as the one used in [12] which are the PESQ, the STOI and the composite metric.

Comparison with DeepFilterNet2. We observe that all personalized models perform better than DeepFilterNet2. This improvement can especially be seen with the PESQ and the CSIG metrics. This underlines the added value of the embedding to recover the target voice. This can also be observed on Fig. 2 where we compare DeepFilterNet2 and pDeepFilterNet2 using the PESQ on subsets of the test set. We see that on the *pn* subset, both models achieve the same PESQ. However, on *ps* and *psn* subsets, the personalized model performs much better. This result was expected, and showcases the benefit of the embedding in the presence of an interfering speaker.

Embedding integration. We also study the position of the embedding integration. Among all personalized models, the unified encoder model is the best one. This suggests that concatenating the speaker embedding with the features of the two branches leads to a better use of the embedding in the network. However, even though pDeepFilterNet2$^{D}_{both}$ contains the embedding information in both branches, it does not lead to the same performance. This underlines the importance of the concatenation of the three features together compared to independently. Comparing pDeepFilterNet2$^{D}_{erb}$ and pDeepFilterNet2$^{D}_{df}$ allows one to understand the weight of each branch in the personalization process. We thus observe that concatenating the embedding in the DF branch only or in

Table 1: Results on generated test set. For evaluation scores, higher is better. **Bold** means best results.

Model	PESQ	STOI	CSIG	CBAK	COVL	Params (M)	MACs (G)	RTF
Noisy	1.81	0.75	2.99	2.45	2.37	-	-	-
DeepFilterNet2	2.10	0.75	3.11	2.66	2.58	**2.31**	**0.33**	**0.03**
pDeepFilterNet2	**2.36**	0.78	**3.66**	**2.90**	**3.01**	**2.31**	**0.33**	**0.03**
pDeepFilterNet2$^{D}_{both}$	2.31	0.78	3.60	2.83	2.95	2.71	0.40	**0.03**
pDeepFilterNet2$^{D}_{erb}$	2.21	0.77	3.51	2.76	2.85	2.71	0.40	**0.03**
pDeepFilterNet2$^{D}_{df}$	2.32	**0.79**	3.58	2.85	2.94	2.71	0.40	**0.03**

Table 2: PDNSMOS P.835 results on the DNS5 blind test set. Higher is better. **Bold** means better than noisy. Underline means best results.

Model	Track 1: Headset			Track 2: Speakerphone		
	SIG	BAK	OVRL	SIG	BAK	OVRL
Noisy	4.15	2.37	2.71	4.05	2.16	2.50
Proposed	3.69	**3.51**	**3.04**	3.63	**3.38**	**2.94**
TEA-PSE 3.0	4.11	4.05	3.65	3.99	3.95	3.49

both branches leads to almost identical results, while concatenating it in the ERB branch only leads to lower results. We can guess that the embedding allow the DF branch to focus on removing everything but the target voice. In fact, isolating two voices is a difficult task which may explain why the DF branch has more impact than the ERB branch as it is used for fine estimation.

Computational complexity. We finally focus on the impact of personalization on the computational complexity of DeepFilterNet2. We compare the number of parameters, the multiply-accumulate operations (MAC) and the real-time factor (RTF) computed on a Intel® Core™ i7-10870H CPU @ 2.20GHz × 16. We can see that simply concatenating the embedding within the unified encoder leads to minimal impact. Indeed, the MAC, the RTF and the number of parameters remain unchanged. Nonetheless, the three other personalization methods change the complexity of the framework. This was expected as deunifiying the encoder leads to an additional GRU that increases the complexity. In the end, even though the dual encoder architectures are slightly larger than the unified encoder architecture, the complexity is still very low compared to most of the recent PSE models.

4.2. Blind test set

We compare our best model on the generated test set, pDeepFilterNet2, to a larger model, the TEA-PSE 3.0 [22]. We compare them on the DNS5 blind test set using the DNSMOS [23], and we also compare their computational complexity. We can first see that for both tracks, pDeepFilterNet2 is not as good as the TEA-PSE 3.0. Yet, the gap between both models is less important on Track 2 which corresponds to speakerphone excerpts. This could be explained by the nature of our training set as it is closer to the speakerphone track. Notably, the TEA-PSE 3.0 has 22.24 million trainable parameters and the number of MAC is 19.66G. Our model is therefore ten times smaller and the number of MAC fifty times lower than TEA-PSE 3.0. Such a light model could be implemented on low-resource devices which is not the case for bigger models like the TEA-PSE 3.0.

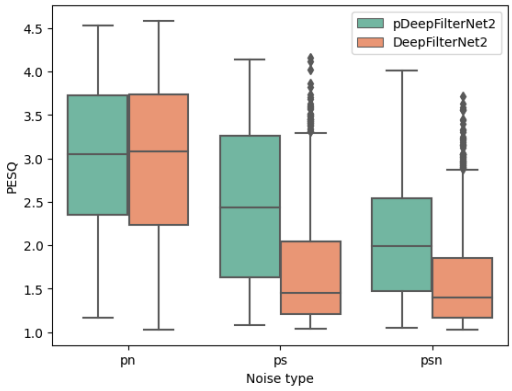

Fig. 2: Box plot featuring the PESQ for DeepFilterNet2 and pDeepFilterNet2 (unified encoder version) for different noise types: primary speaker + noise (pn), primary speaker + secondary speaker (ps), primary speaker + secondary speaker + noise (psn).

5. CONCLUSION & FUTURE WORK

In this work, we proposed a robust method to personalize a lightweight state-of-the-art SE model. We explored the different possibilities to integrate the embedding into this dual-stage framework and we determined the best implementation. We showed that our personalization strategy greatly improves the performances in the presence of interfering speakers while preserving the low computational complexity of DeepFilterNet2 making it a great candidate for real-time PSE on embedded devices. Future work will look into improving the performances of our model to make it competitive with state-of-the-art SE metrics while further optimizing its complexity.

6. REFERENCES

[1] C. Xu, W. Rao, E. S. Chng, and H. Li, "Spex: Multi-scale time domain speaker extraction network," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 28, pp. 1370–1384, 2020.

[2] M. Ge, C. Xu, L. Wang, E. S. Chng, J. Dang, and H. Li, "Spex+: A complete time domain speaker extraction network," *Proc. Interspeech*, 2020.

[3] ——, "Multi-stage speaker extraction with utterance and frame-level reference signals," in *Proc. ICASSP*, 2021, pp. 6109–6113.

[4] X. Ji, M. Yu, C. Zhang, D. Su, T. Yu, X. Liu *et al.*, "Speaker-aware target speaker enhancement by jointly learning with speaker embedding extraction," in *Proc. ICASSP*. IEEE, 2020, pp. 7294–7298.

[5] Q. Wang, H. Muckenhirn, K. Wilson, P. Sridhar, Z. Wu, J. Hershey *et al.*, "Voicefilter: Targeted voice separation by speaker-conditioned spectrogram masking," *Proc. Interspeech*, 2018.

[6] R. Giri, S. Venkataramani, J.-M. Valin, U. Isik, and A. Krishnaswamy, "Personalized percepnet: Real-time, low-complexity target voice separation and enhancement," *Proc. Interspeech*, 2021.

[7] S. E. Eskimez, T. Yoshioka, H. Wang, X. Wang, Z. Chen, and X. Huang, "Personalized speech enhancement: New models and comprehensive evaluation," in *Proc. ICASSP*, 2022, pp. 356–360.

[8] K. Žmolíková, M. Delcroix, K. Kinoshita, T. Ochiai, T. Nakatani, L. Burget *et al.*, "Speakerbeam: Speaker aware neural network for target speaker extraction in speech mixtures," *IEEE Jour. of Selected Topics in Signal Processing*, vol. 13, no. 4, pp. 800–814, 2019.

[9] A. Li, W. Liu, X. Luo, C. Zheng, and X. Li, "Icassp 2021 deep noise suppression challenge: Decoupling magnitude and phase optimization with a two-stage deep network," in *Proc. ICASSP*, 2021, pp. 6628–6632.

[10] Y. Ju, W. Rao, X. Yan, Y. Fu, S. Lv, L. Cheng *et al.*, "Tea-pse: Tencent-ethereal-audio-lab personalized speech enhancement system for icassp 2022 dns challenge," in *Proc. ICASSP*. IEEE, 2022, pp. 9291–9295.

[11] A. Li, C. Zheng, L. Zhang, and X. Li, "Glance and gaze: A collaborative learning framework for single-channel speech enhancement," *Applied Acoustics*, vol. 187, p. 108499, 2022.

[12] H. Schröter, A. Maier, A. Escalante-B, and T. Rosenkranz, "Deepfilternet2: Towards real-time speech enhancement on embedded devices for full-band audio," in *Proc. IWAENC*, 2022, pp. 1–5.

[13] M. Thakker, S. E. Eskimez, T. Yoshioka, and H. Wang, "Fast real-time personalized speech enhancement: End-to-end enhancement network (e3net) and knowledge distillation," *Proc. Interspeech*, 2022.

[14] Q. Wang, I. L. Moreno, M. Saglam, K. Wilson, A. Chiao, R. Liu *et al.*, "Voicefilter-lite: Streaming targeted voice separation for on-device speech recognition," *Proc. Interspeech*, 2020.

[15] M. Delcroix, K. Zmolikova, T. Ochiai, K. Kinoshita, S. Araki, and T. Nakatani, "Compact network for speakerbeam target speaker extraction," in *Proc. ICASSP*, 2019, pp. 6965–6969.

[16] B. Desplanques, J. Thienpondt, and K. Demuynck, "Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification," *Proc. Interspeech*, 2020.

[17] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *Proc. ICASSP*, 2018, pp. 5329–5333.

[18] S. E. Eskimez, T. Yoshioka, H. Wang, X. Wang, Z. Chen, and X. Huang, "Personalized speech enhancement: New models and comprehensive evaluation," in *Proc. ICASSP*, 2022, pp. 356–360.

[19] H. Dubey, A. Aazami, V. Gopal, B. Naderi, S. Braun, R. Cutler *et al.*, "Icassp 2023 deep noise suppression challenge," *Proc. ICASSP*, 2023.

[20] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer *et al.*, "Common voice: A massively-multilingual speech corpus," *LREC*, 2019.

[21] Z. Wang, R. Giri, D. Shah, J.-M. Valin, M. M. Goodwin, and P. Smaragdis, "A framework for unified real-time personalized and non-personalized speech enhancement," in *Proc. ICASSP*, 2023, pp. 1–5.

[22] Y. Ju, J. Chen, S. Zhang, S. He, W. Rao, W. Zhu *et al.*, "Tea-pse 3.0: Tencent-ethereal-audio-lab personalized speech enhancement system for icassp 2023 dns-challenge," in *Proc. ICASSP*, 2023, pp. 1–2.

[23] C. K. A. Reddy, V. Gopal, and R. Cutler, "Dnsmos: A non-intrusive perceptual objective speech quality metric to evaluate noise suppressors," *Proc. ICASSP*, 2021.