TF-MLPNet: Tiny Real-Time Neural Speech Separation

Malek Itani[1], *Tuochao Chen*[1], *Shyamnath Gollakota*[1]

[1]University of Washington

{malek,tuochao,gshyam}@cs.washington.edu

Abstract

Speech separation on hearable devices can enable transformative augmented and enhanced hearing capabilities. However, state-of-the-art speech separation networks cannot run in real-time on tiny, low-power neural accelerators designed for hearables, due to their limited compute capabilities. We present *TF-MLPNet,* the first speech separation network capable of running in real-time on such low-power accelerators while outperforming existing streaming models for blind speech separation and target speech extraction. Our network operates in the time-frequency domain, processing frequency sequences with stacks of fully connected layers that alternate along the channel and frequency dimensions, and independently processing the time sequence at each frequency bin using convolutional layers. Results show that our mixed-precision quantization-aware trained (QAT) model can process 6 ms audio chunks in real-time on the GAP9 processor, achieving a 3.5-4x runtime reduction compared to prior speech separation models.

Speech separation, streaming, quantization

1. Introduction

Over the past decade, two key technological trends have emerged. First, deep learning has become central to speech separation algorithms [1, 2, 3, 4], which typically require large, energy intensive resources like GPUs. Second, there is increasing interest in incorporating speech separation into hearables, such as hearing aids, headphones, and earbuds, to develop advanced augmented and enhanced hearing applications that program acoustic scenes and address the cocktail party problem [5, 6, 7, 8]. However, these small, power-constrained devices have limited computing capabilities. To bridge this gap, several hardware efforts have focused on developing tiny, low-power neural network accelerators [9, 10]. These platforms are, however, significantly more constrained than both GPUs and general-purpose embedded CPUs like Raspberry Pi.

State-of-the-art speech separation networks cannot run in real-time on low-power neural accelerators for hearables. These networks operate in the time-frequency (TF) domain, modeling time and frequency components as sequences using recurrent networks [11, 12], attention mechanisms [13], or both [2]. At each time interval, the frequency sequence sub-network processes the entire sequence of frequencies, while the time sequence sub-network independently processes across time, at each of the frequency bins. Streaming applications like enhanced hearing however require neural networks to operate in real-time on small blocks ($\leq$10 ms), which imposes significant computational and algorithmic constraints.

We introduce *TF-MLPNet*, the first on-device real-time neural speech separation network for low-power hearables. We make two key observations: 1) Using recurrent models to process frequency sequences sequentially slows computation, while transformer and state-space architectures are incompatible with our target low-power accelerators; 2) Typically, time sequence modeling at each frequency bin are parallelized using batched processing. However, our target accelerators are designed for low-power inference and operate on one input at a time [9, 10], i.e., with a batch size of 1. This precludes batched processing, leading again to slow, sequential computation.

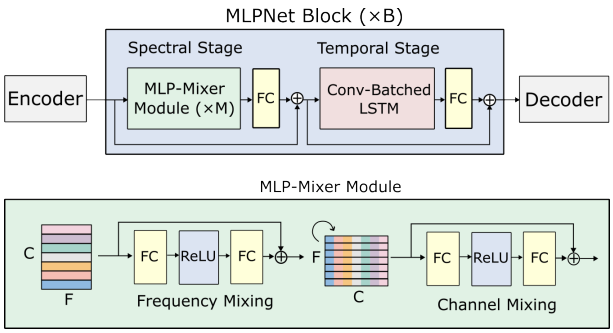

Figure 1: *The TF-MLPNet architecture has a conv-batched LSTM in the temporal stage, enabling parallel inference of batched LSTM inputs via convolutional layers, and a highly parallel all-MLP-Mixer module in the spectral stage, replacing the sequential bidirectional LSTM. (FC: fully connected layer)*

Inspired by all-MLP architectures [14, 15, 16], our model (Fig. 1) replaces the recurrent networks used for processing frequency bins, with stacks of fully connected layers applied alternately along the channel and frequency dimensions, enabling parallel processing across frequency bins for improved efficiency. We also introduce a hardware-software co-design method that parallelizes time sequence processing at each frequency bin. By using just convolutional layers with a batch size of 1, we effectively parallelize batched LSTMs, mimicking the batched inference of a recurrent network in a single time step. Finally, we also present a mixed-precision QAT strategy that balances performance and runtime. Since different modules in the network have varying sensitivity to quantization and its impact on runtime, we apply distinct quantization configurations to different sub-components in our network, optimizing performance while maintaining real-time constraints.

We evaluate this architecture on two tasks 1) two-speaker blind speech separation (BSS) and 2) target speech extraction (TSE). We compare TF-MLPNet with multiple variants of the causal TF-GridNet model [2] for the BSS task and with pDCCRN [17], and TinyDenoiser [18] for the TSE task. Our results show that TF-MLPNet achieves state-of-the-art real-time on-device performance. Furthermore, our real-time mixed-precision quantized model results in a performance drop of only 0.6 dB, compared to a fully floating-point network.

2. Related work

Blind speech separation and target speaker extraction. Prior neural architectures [19, 2, 20, 21] use components like convolutional [19], LSTM [4], transformer [3], and state-space [22]

layers. However, these models prioritize speech quality over real-time, on-device, or low-power constraints. Furthermore, transformers and state-space models have runtimes and memory demands that exceed our target hardware's capabilities.

Low-latency speech processing. For augmented hearing, minimizing input-to-output latency is crucial but can reduce performance due to the limited information available for predicting output [23]. Prior work has explored architectures for low-latency speech tasks [24, 25, 26, 6, 27, 28], but these are evaluated on devices with much higher clock frequencies, power budgets, and memory footprints than our target hardware. The most relevant, FSPEN [11], enhances speech in real-time using gated recurrent units (GRUs). Our results show that replacing its bidirectional GRU with an all-MLP layer improves efficiency on our target hardware platform.

TinyML. In the audio domain, prior work has applied TinyML methods to classification tasks like keyword spotting [29], speaker verification [30], and sound event detection [31], as well as regression tasks like speech enhancement and denoising [32, 18, 33, 34, 35, 36, 37]. These networks however are not designed for speech separation. They also do not process the time and frequency components as individual sequences as has become an essential component in state-of-the-art speech separation models. [38] designs a quantized audio separation network but the proposed network is neither causal, real-time nor can run on low-power accelerators.

3. Methods

3.1. System Requirements and Runtime Decomposition

Real-time on-device enhanced hearing applications impose strict constraints on model size, runtime, and power consumption. For example, if our model receives 6 ms long audio chunks, running inference on these chunks should take less than 6 ms for real-time operation. Further, non-volatile storage is limited, e.g. given the size of GAP9 eMRAM, the model size can be at most 1.5 MB to avoid using additional memory components. Finally, to ensure $>$ 6 hours of continuous use on a 675 hearing aid battery, power consumption must stay below 100 mW.

We start with a causal dual-path model, TF-GridNet, that operates on the TF-domain [2]. Given our target hardware, we remove the self-attention module but keep the recurrent modules. We quantize all nodes in TF-Gridnet into INT8, with the exception of Layer Normalization and the batched time-domain LSTM, which are quantized to FLOAT16 since these layers lack INT8 out-of-the-box support. Fig. 2 profiles the runtime on GAP9 running at 370 MHz. This model requires 23.5 ms to process one 6 ms chunk. The major contributors to the runtime are 1) the frequency-domain bidirectional LSTMs and 2) the time-domain batched unidirectional LSTMs.

3.2. TF-MLPNet

TF-MLPNet is a tiny real-time network for speech separation. Our network has two main components: a conv-batched LSTM, that enables parallel inference of a batch of inputs to an LSTM using convolutional layers, and a highly parallel all-MLP-Mixer module that replaces the sequential bidirectional LSTM.

In TF-MLPNet, we first apply a short-time Fourier transform (STFT) to convert a time-domain audio signal $x \in \mathbf{R}^{1\times t}$ into a time-frequency (TF) representation $X' \in \mathbf{C}^{1\times F\times T}$ over T frames and F frequency bins. The real and imaginary components are concatenated along the channel dimension to get $X \in \mathbf{R}^{2\times F\times T}$. As shown in Fig. 1(a), a causal 3×3 2D convolution encoder extracts a C-dimensional latent representation $\hat{X} \in \mathbf{R}^{C\times F'\times T'}$. $\hat{X}$ is processed by B MLPNet blocks, each containing an MLP-Mixer and a conv-batched LSTM module, producing $\tilde{X} \in \mathbf{R}^{1\times F'\times T'}$. A causal 3×3 2D transposed convolution decoder then recovers the TF representations $Y \in \mathbf{R}^{2S\times F\times T}$ for each of the S speakers ($S = 2$ for BSS and $S = 1$ for TSE). This produces two channels per speaker — the first half corresponds to the real components of the TF representation, while the second half corresponds to the imaginary components. Accordingly, we reinterpret Y as $Y' \in \mathbf{C}^{S\times F\times T}$ use inverse STFT and overlap-add operations to reconstruct the time-domain output signal $y \in \mathbf{R}^{1\times t}$ for each speaker.

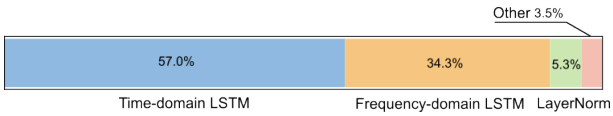

Figure 2: *Runtime profile of an existing dual-path model.*

3.2.1. MLP-Mixer Module

At the spectral stage of existing dual-path models [2], the key contributor to the inference time is the bidirectional frequency-domain LSTM. Due to the nature of RNNs, each frequency bin in the input must be processed sequentially. One way to reduce the runtime is to compress the number of frequencies with strided convolutions before and after frequency-domain processing [8]. However, this was shown to reduce the algorithm's performance. An alternative approach is to replace the recurrent architecture with one that can better utilize the parallel processing capabilities of the neural accelerator. While this can be achieved using transformers or linear RNNs, today's low-power hardware accelerators do not support these complex operations.

We instead replace the bidirectional frequency-domain LSTM with an MLP-Mixer [14], that applies a sequence of multilayer perceptions alternately along the frequency and channel dimensions (Fig. 1). This MLPMixer module is repeated M times successively in every MLPNet Block. To further reduce inference time, we replace the GELU nonlinearities with simpler ReLU nonlinearities and omit the layer normalization.

3.2.2. Conv-Batched LSTM

Existing dual-path models [2] process frequency components independently in the time domain. During training, GPUs handle this efficiently by treating each frequency bins as a separate batch. However, inference on low-power accelerators is challenging since many lack support for batched processing.

During streaming inference, the batched LSTM receives $\hat{X} \in \mathbf{R}^{\mathbf{C}\times\mathbf{F'}\times\mathbf{1}}$ and processes a single LSTM step on F' independent sequences. A naive solution is sequential processing per frequency bin to maintain a batch size of 1, but this underutilizes parallel processing and greatly increases inference time.

Instead of a standard LSTM kernel, we decompose the LSTM into its building blocks. We use 1D convolutions (kernel size = 1) for the linear gates, treating the frequency dimension as the sequence dimension. This allows parallel processing of all frequencies without special hardware support. Note that since we process only a single time frame at a time during streaming inference, batched inference for frequency-domain processing is not required, and so the batch size, i.e., the number of independent time frames, is just 1.

Table 1: *Main results. SISDR, PESQ and DNS-MOS are reported for* FP32 *BSS models. For runtime, we assume the networks are fully* INT8 *quantized, except in the case where they have Layer Normalization, which is quantized to* FP16. *Values in red do not meet our system requirements.*

Name	Runtime (ms)	SISDR (dB)	PESQ	DNS -MOS	Param (K)
Mixture	–	0.00	1.24	2.48	–
TF-GridNet [2]	16.4	14.78	2.39	3.28	173
TFG-LN	15.1	14.08	2.29	3.12	173
TFG-LN+2F	8.8	13.78	2.27	3.14	198
TFG-LN+4F	5.6	13.51	**2.25**	3.09	222
TFG-LN+6F	4.5	13.16	2.17	3.07	247
TFG-LN+GRU	12.4	14.47	2.35	3.22	147
TFG-LN+2F+GRU	7.4	12.87	2.13	3.00	172
TFG-LN+4F+GRU	4.9	12.66	2.10	2.96	197
TF-MLPNet	3.6	**14.12**	2.23	**3.20**	493
TF-MLPNet+2F	2.8	13.06	2.09	3.04	215

3.3. Mixed-Precision Quantization

We use Quantization-Aware Training (QAT) [39] to simulate quantization errors during training and reduce performance degradation. We start with a floating-point model and fine-tune it using the FQSE quantization framework [33]. Our weight quantization is symmetric and per-channel, while activation quantization is asymmetric and per-tensor.

Our fully INT8-quantized network with QAT produces a noticeable performance degradation. To bridge the gap between quantized and floating-point models, we designed a mixed-precision QAT approach to balance performance and runtime.

We quantize the first input convolution and the last deconvolution layers to BFLOAT16 instead of INT8 to preserve the high-precision information from the input in the first layer and to reconstruct high-quality audio with the last deconvolution. Additionally, our experiments with batched LSTMs revealed that convolution layers dominate the runtime and that quantization errors accumulate temporally. So, we use a mixed-precision LSTM, where we quantize convolution layers to INT8 for efficiency while keeping activation, addition, multiplication operations, and cell states in BFLOAT16 to minimize quantization noise and improve performance.

To ensure real-time operation while minimizing performance loss, we use INT16 activations for the MLP modules at odd-numbered MLPNet blocks and INT8 activations for those at even-numbered blocks. The MLP module weights are always quantized in INT8. Finally, we incorporate the SDR-aware knowledge distillation loss function [38] into the QAT process.

4. Experiments and results

Datasets. We train our model using mixtures from LibriSpeech [40] and evaluate it on the LibriSpeech test set [40] and VCTK [41]. Each 5-second, 16 kHz speech mixture is created by sampling two different speaker utterances from the same corpus split. Utterances longer than 5 seconds are cropped, while shorter ones are padded with silence. For TSE, one speaker is chosen as the target, and the speaker d-vector [42] embedding is computed from a different utterance by that target speaker. The interfering speech is scaled to achieve an input SNR uniformly distributed in [-10,10] dB. Training speech files come from `train-clean-360`, validation from `dev-clean`, and testing from `test-clean`. The training set is generated on-the-fly, while validation and test sets contain 2k and 1k mixtures, respectively.

Table 2: *SI-SDRi (dB) for blind source separation (BSS) as a function of percentage of training set used for training models.*

	Training Dataset Percentage (%)						
Model	**1**	**2**	**5**	**10**	**25**	**50**	**100**
TFG-LN+4F	4.52	7.83	11.44	12.99	13.34	13.26	13.51
TF-MLPNet	4.18	6.08	9.65	12.99	13.85	14.04	14.12

Evaluation setup. We compare with multiple model variants. For TF-GridNet, we use the causal implementation from [6] without self-attention and with hyperparameters $B = 6$, $D = 32$ and $H = 32$. We remove the LayerNormalization modules to obtain the model TFG-LN. We further introduced frequency compression on the frequency-domain processing component used in [8], and we refer to the resulting model with a frequency compression rate α as TFG-LN+αF. We also considered a variant where we replaced the LSTM with a GRU, referred to as TFG-LN+αF+GRU. In addition to the hyperparameters enumerated above, the TF-MLPNet used in our experiments has an MLP-Mixer with $M = 2$ MLP-Mixer repetitions. Additionally, when we apply frequency compression to our TF-MLPNet architecture, we refer to it as TF-MLPNet+αF.

Following [23], we use a 10 ms output window and a 6 ms hop size, resulting in a 10 ms algorithmic latency. For BSS, the decoder outputs two channels (one per speaker), while TSE uses a single channel. In TSE, the model is conditioned on d-vector embeddings via a FiLM layer after the encoder. We convert PyTorch models into optimized kernels for GAP9 using GAPFlow.

Loss function and training hyper-parameters. We train BSS models using Permutation Invariant Training with negative SI-SDR as the loss function. For TSE, we use a combined loss: $L_{SI-SDR} + L_{PESQ}$, where L_{SI-SDR} is negative SI-SDR, and L_{PESQ} is calculated using `torch-pesq`. Each epoch processes 20k mixtures before validation. Models are trained for 400 epochs using a three-stage schedule: (1) linearly increasing the learning rate from 1e-4 to 1e-3 over 10 epochs, (2) maintaining 1e-3 for 200 epochs, and (3) halving it every 30 epochs for the remaining 190. Model parameters are optimized using AdamW and we use a gradient clipping of 0.1. Model performance is evaluated using the the epoch with the lowest validation loss.

QAT hyper-parameters. We start with the trained floating-point parameters and fine-tune it for 100 epochs. Since QAT is time-consuming, each epoch processes 4k mixtures. Training starts with a 1e-3 learning rate using a `ReduceLROnPlateau` scheduler (patience = 5, factor = 0.5). Early stopping is triggered if validation loss does not improve for 20 epochs.

Results. Table 1 compares floating-point performance of different models on the BSS task using SI-SDR, PESQ, and DNS-MOS OVRL [43]. We measure quantized runtime for processing a 6 ms audio chunk on GAP9 at 370 MHz.

The original TF-GridNet achieves the best floating-point performance (14.78 dB) but requires 25.8 ms runtime at FP16, reduced to 16.4 ms with INT8 quantization and FP16 layer normalization. Removing LayerNorm (TFG-LN) and applying our Conv-Batched LSTM method cuts FP16 runtime to 22.3 ms, with full INT8 quantization reducing it to 15.1 ms—still too slow for real-time use. Frequency compression meets real-time requirements, bringing INT8 runtime down to 5.6 ms ($\times 4$

Table 3: *Quantization results. We measure the SISDRi, model size, runtime and power consumption for different QAT strategies. MixLSTM refers to quantizing convolution layers to* INT8 *while keeping others in* BFLOAT16*. FPConv refers to quantizing the input convolution and output deconvolution in* BFLOAT16 *instead of* INT8*. KD refers to using SDR-aware knowledge distillation as the training loss. MixMLP refers to quantizing only half the MLP modules into INT16, while in FullMLP, we quantize all of them to INT16. The reported model size includes the memory needed to store the quantized model parameters, but not the quantization constants.*

QAT config TF-MLPNet	SISDRi (dB)	Size (kB)	Runtime (ms)	Power (mW)
FP32	14.12	1926	–	–
INT8	10.21	481	3.6	54.9
+MixLSTM	11.22	481	4.0	58.1
+FPConv	12.57	483	4.2	60.7
+KD	13.07	483	4.2	60.7
+MixMLP	13.52	483	5.6	80.1
+FullMLP	13.65	483	6.5	–

compression) and 4.5 ms ($\times$6 compression), but at a 0.5-1 dB floating-point SI-SDR performance drop. Replacing LSTM with GRU also speeds up INT8 runtime but affects performance.

Our TF-MLPNet model, which replaces frequency-domain LSTMs with MLP-Mixer, achieves a drastic runtime reduction to 3.6 ms with INT8 quantization. This improvement is due to its use of simple, parallelizable MLPs that efficiently leverage the neural accelerator. TF-MLPNet offers the lowest runtime while maintaining a floating-point performance at 14.12 dB. A paired t-test was conducted between our TF-MLPNet and other baselines which meets the real-time requirements for each metric, showing a significant difference with $p < 0.05$.

We analyze how training dataset size affects TF-MLPNet performance compared to TF-LN+4F, the best real-time baseline model variant. In Table 2, both models are trained for the same number of epochs, but with varying proportions of speakers seen during training. While TF-MLPNet underperforms TF-LN+4F with limited data, it surpasses it as more speakers are included in training. This indicates that TF-MLPNet scales better with increased data, which matches the observation in [14].

Quantization impacts model performance, so we experimented with different quantization settings and QAT techniques to assess their effects on performance and runtime. We evaluate the following configurations: (1) FP32: original float-pointing model, (2) INT8: fully quantized INT8 model, (3) MixLSTM: Mixed-precision LSTM (see §3.3), with other modules quantized to INT8, (4) MixLSTM+FPConv: mixed-precision LSTM, BFLOAT16 Conv/Deconv, other modules quantized to INT8, (5) MixLSTM+FPConv+MixMLP: mixed-precision LSTM, BFLOAT16 Conv/Deconv, mixed-precision MLP (see §3.3), others in INT8, and (6) MixLSTM+FPConv+FULLMLP: mixed-precision LSTM, BFLOAT16 Conv/Deconv, fully INT16-quantized MLP, others in INT8. We trained with SI-SDR loss and also explored SDR-aware knowledge distillation ("+KD"). As shown in Table 3, full INT8 quantization with standard QAT led to around 4 dB drop in SI-SDRi. However, incorporating KD loss and mixed-precision quantization—including mixed-precision LSTM, BFLOAT16 Conv, and mixed-precision MLP—recovered SI-SDRi to 13.52 dB while still achieving real-time performance.

Table 4 shows that TF-MLPNet's performance gains generalize to out-of-distribution datasets. We created 5-second audio mixtures using VCTK data, following the same process as before. BSS models trained on LibriSpeech mixtures were then evaluated on the VCTK mixtures. TF-MLPNet outperformed baseline models, showing generalization across datasets.

Table 4: *BSS results on 2-speaker mixtures from VCTK. All models were only trained on data from LibriSpeech. SISDR, PESQ and DNS-MOS are reported for* FP32 *BSS models. For runtime, we assume the networks are fully INT8 quantized.*

Name	SISDR (dB)	PESQ	Runtime (ms)
Mixture	-0.01	1.55	–
TFG-LN+4F	12.36	**1.97**	5.6
TFG-LN+6F	11.51	1.94	4.5
TFG-LN+4F+GRU	11.68	1.89	4.9
TF-MLPNet	**12.63**	1.95	3.6
TF-MLPNet+2F	11.59	1.85	2.8

Table 5: *Target Speech extraction (TSE) results. SISDR, PESQ and DNS-MOS are reported for* FP32 *TSE models. For runtime, we assume the networks are fully INT8 quantized. Values in red do not meet our runtime or memory requirements.*

Name	SISDR (dB)	PESQ	DNS -MOS	Param (K)	Runtime (ms)
Mixture	0.05	1.26	2.50	–	–
pDCCRN [17]	10.71	2.15	3.05	3218	–
TFG-LN+2F	12.22	2.32	3.23	213	8.9
TFG-LN+4F	11.90	2.28	3.18	238	5.7
TFG-LN+6F	11.28	2.24	3.15	263	4.6
TFG-LN+2F+GRU	12.20	2.32	3.18	188	7.5
TFG-LN+4F+GRU	12.00	2.30	3.15	213	5.0
TinyDenoiser [18]	8.15	1.70	2.65	1056	0.464
TF-MLPNet	**12.37**	**2.37**	**3.32**	509	3.6
TF-MLPNet+2F	11.77	2.18	3.12	231	2.9

Finally, we evaluated TF-MLPNet on the TSE task, comparing it against baseline models and prior work, including pDCCRN [17] and TinyDenoiser [18]. For a fair comparison, all models used the same STFT configuration and a FiLM layer right after the encoder for target speaker conditioning. pDCCRN had the number of convolution filters set to $[16, 32, 64, 128, 256, 256]$, a 5×2 kernel with a 2×1 stride, and an LSTM hidden size of 256. For TinyDenoiser, aside from the STFT configuration, we use the same hyperparameters described in [18]. TF-MLPNet outperformed all models across metrics while maintaining real-time performance, demonstrating its effectiveness across both speech separation tasks.

5. Conclusion

We introduce TF-MLPNet, the first real-time speech separation network for low-power hearables, outperforming existing streaming architectures. While we use hardware consistent with prior research [44], exploring our methods on platforms like Qualcomm's S7 series, Analog Devices' MAX78002, and Syntiant's NDP120 offers interesting future directions. Further work includes enabling other audio tasks such as target sound extraction, distance-based multi-channel source separation [8], and directional hearing [45] on constrained hardware.

6. References

[1] K. Zmolikova, M. Delcroix, T. Ochiai, K. Kinoshita, J. Černocký, and D. Yu, "Neural target speech extraction: An overview," *IEEE Signal Processing Magazine*, 2023.

[2] Z.-Q. Wang, S. Cornell, S. Choi, Y. Lee, B.-Y. Kim, and S. Watanabe, "Tf-gridnet: Making time-frequency domain models great again for monaural speaker separation," in *ICASSP*, 2023.

[3] C. Subakan, M. Ravanelli, S. Cornell, M. Bronzi, and J. Zhong, "Attention is all you need in speech separation," in *ICASSP*, 2021.

[4] Y. Luo, Z. Chen, and T. Yoshioka, "Dual-path RNN: efficient long sequence modeling for time-domain single-channel speech separation," in *ICASSP*. IEEE, 2020.

[5] B. Veluri, M. Itani, J. Chan, T. Yoshioka, and S. Gollakota, "Semantic hearing: Programming acoustic scenes with binaural hearables," in *ACM UIST*, 2023.

[6] B. Veluri, M. Itani, T. Chen, T. Yoshioka, and S. Gollakota, "Look once to hear: Target speech hearing with noisy examples," in *ACM CHI*, 2024.

[7] S. Cornell, Z.-Q. Wang, Y. Masuyama, S. Watanabe, M. Pariente, and N. Ono, "Multi-channel target speaker extraction with refinement: The wavlab submission to the second clarity enhancement challenge," in *arXiv*, 2023.

[8] T. Chen, M. Itani, S. Eskimez, T. Yoshioka, and S. Gollakota, "Hearable devices with sound bubbles," *Nature Electronics*, 2024.

[9] "NDP120 – Syntiant," https://www.syntiant.com/ndp120.

[10] "GAP9 processor — GreenWaves Technologies," https://greenwaves-technologies.com/gap9$_processor/$.

[11] L. Yang, W. Liu, R. Meng, G. Lee, S. Baek, and H.-G. Moon, "Fspen: an ultra-lightweight network for real time speech enahncment," in *ICASSP*, 2024.

[12] R. Chao, W.-H. Cheng, M. La Quatra, S. M. Siniscalchi, C.-H. H. Yang, S.-W. Fu, and Y. Tsao, "An investigation of incorporating mamba for speech enhancement," *arXiv*, 2024.

[13] Y.-X. Lu, Y. Ai, and Z.-H. Ling, "Mp-senet: A speech enhancement model with parallel denoising of magnitude and phase spectra," *arXiv*, 2023.

[14] I. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, M. Lucic, and A. Dosovitskiy, "Mlp-mixer: An all-mlp architecture for vision," in *Neurips*, 2021.

[15] F. Mai, J. Zuluaga-Gomez, T. Parcollet, and P. Motlicek, "Hyperconformer: Multi-head hypermixer for efficient speech recognition," in *Interspeech*, 2023.

[16] T. Parcollet, R. van Dalen, S. Zhang, and S. Bhattacharya, "Summarymixing: A linear-complexity alternative to self-attention for speech recognition and understanding," in *Interspeech*, 2024.

[17] S. E. Eskimez, T. Yoshioka, H. Wang, X. Wang, Z. Chen, and X. Huang, "Personalized speech enhancement: new models and comprehensive evaluation," in *IEEE ICASSP*, 2022.

[18] M. Rusci, M. Fariselli, M. Croome, F. Paci, and E. Flamand, "Accelerating rnn-based speech enhancement on a multi-core mcu with mixed fp16-int8 post-training quantization," in *arXiv*, 2022.

[19] Y. Luo and N. Mesgarani, "Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 2019.

[20] S. Zhao and B. Ma, "Mossformer: Pushing the performance limit of monaural speech separation using gated single-head transformer with convolution-augmented joint self-attentions," in *ICASSP*, 2023.

[21] U.-H. Shin, S. Lee, T. Kim, and H.-M. Park, "Separate and reconstruct: Asymmetric encoder-decoder for speech separation," in *arXiv*, 2024.

[22] K. Li and G. Chen, "Spmamba: State-space model is all you need in speech separation," in *arXiv*, 2024.

[23] Z.-Q. Wang, G. Wichern, S. Watanabe, and J. Le Roux, "Stft-domain neural speech enhancement with very low algorithmic latency," *Trans. on Audio, Speech, and Language Processing*, 2022.

[24] J.-M. Valin, U. Isik, N. Phansalkar, R. Giri, K. Helwani, and A. Krishnaswamy, "A perceptually-motivated approach for low-complexity, real-time enhancement of fullband speech," in *arXiv*, 2020.

[25] H. Sato, T. Moriya, M. Mimura, S. Horiguchi, T. Ochiai, T. Ashihara, A. Ando, K. Shinayama, and M. Delcroix, "Speakerbeam-ss: Real-time target speaker extraction with lightweight conv-tasnet and state space modeling," 2024.

[26] H. Schröter, A. N. Escalante-B., T. Rosenkranz, and A. Maier, "Deepfilternet2: Towards real-time speech enhancement on embedded devices for full-band audio," in *arXiv*, 2022.

[27] B. Veluri, J. Chan, M. Itani, T. Chen, T. Yoshioka, and S. Gollakota, "Real-time target sound extraction," in *ICASSP*, 2023.

[28] M. A. Akeroyd, W. Bailey, J. Barker, T. J. Cox, J. F. Culling, S. Graetzer, G. Naylor, Z. Podwińska, and Z. Tu, "The 2nd clarity enhancement challenge for hearing aid speech intelligibility enhancement: Overview and outcomes," in *ICASSP*, 2023.

[29] Y. Zhang, N. Suda, L. Lai, and V. Chandra, "Hello edge: Keyword spotting on microcontrollers," in *arXiv*, 2018.

[30] M. Pavan, G. Mombelli, F. Sinacori, and M. Roveri, "Tinysv: Speaker verification in tinyml with on-device learning," in *Proceedings of the 4th International Conference on AI-ML Systems*, New York, NY, USA, 2025, Association for Computing Machinery.

[31] Y. Wu, X. Quan, M. R. Izadi, and C.-C. J. Huang, ""it os okay to be uncommon": Quantizing sound event detection networks on hardware accelerators with uncommon sub-byte support," in *ICASSP*, 2024, pp. 281–285.

[32] I. Fedorov, M. Stamenovic, C. Jensen, L.-C. Yang, A. Mandell, Y. Gan, M. Mattina, and P. N. Whatmough, "Tinylstms: Efficient neural speech enhancement for hearing aids," *Interspeech*, 2020.

[33] E. Cohen, H. V. Habi, and A. Netzer, "Towards fully quantized neural networks for speech enhancement," in *Interspeech*, 2023, pp. 181–185.

[34] H.-S. Choi, S. Park, J. H. Lee, H. Heo, D. Jeon, and K. Lee, "Real-time denoising and dereverberation wtih tiny recurrent u-net," in *ICASSP*, 2021.

[35] N. L. Westhausen and B. T. Meyer, "Low bit rate binaural link for improved ultra low-latency low-complexity multichannel speech enhancement in hearing aids," in *WASPAA*, 2023.

[36] R. D. Nathoo, M. Kegler, and M. Stamenovic, "Two-step knowledge distillation for tiny speech enhancement," in *ICASSP*, 2024.

[37] X. Chen, G. Liu, J. Shi, J. Xu, and B. Xu, "Distilled binary neural network for monaural speech separation," in *IJCNN*, 2018.

[38] E. Cohen, H. V. Habi, R. Peretz, and A. Netzer, "Fully quantized neural networks for audio source separation," *IEEE Open Journal of Signal Processing*, vol. 5, pp. 926–933, 2024.

[39] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," in *Low-Power Computer Vision*. 2022.

[40] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An asr corpus based on public domain audio books," in *ICASSP*, 2015.

[41] C. Veaux, J. Yamagishi, and K. MacDonald, "Cstr vctk corpus: English multi-speaker corpus for speech synthesis," *University of Edinburgh. The Centre for Speech Technology Research*, 2017.

[42] Q. Wang, C. Downey, L. Wan, P. A. Mansfield, and I. L. Moreno, "Speaker diarization with lstm," in *ICASSP*. IEEE, 2018.

[43] C. K. A. Reddy, V. Gopal, and R. Cutler, "Dnsmos p.835: A non-intrusive perceptual objective speech quality metric to evaluate noise suppressors," in *ICASSP*, 2022, pp. 886–890.

[44] M. Itani, T. Chen, A. Raghavan, G. Kohlberg, and S. Gollakota, "Wireless hearables with programmable speech ai accelerators," in *arXiv*, 2025.

[45] A. Wang, M. Kim, H. Zhang, and S. Gollakota, "Hybrid neural networks for on-device directional hearing," *AAAI*, 2022.