Knowledge boosting during low-latency inference

Vidya Srinivas,[1] *Malek Itani,*[1] *Tuochao Chen,*[1] *Emre Sefik Eskimez,*[2] *Takuya Yoshioka,*[3] *Shyamnath Gollakota*[1]

[1]University of Washington, [2]Microsoft, [3]AssemblyAI

{vysri,malek,tuochao}@cs.washington.edu, sefik.eskimez@microsoft.com, takuya.yoshioka@ieee.org, gshyam@cs.washington.edu

Abstract

Models for low-latency, streaming applications could benefit from the knowledge capacity of larger models, but edge devices cannot run these models due to resource constraints. A possible solution is to transfer hints during inference from a large model running remotely to a small model running on-device. However, this incurs a communication delay that breaks real-time requirements and does not guarantee that both models will operate on the same data at the same time. We propose knowledge boosting, a novel technique that allows a large model to operate on time-delayed input during inference, while still boosting small model performance. Using a streaming neural network that processes 8 ms chunks, we evaluate different speech separation and enhancement tasks with communication delays of up to six chunks or 48 ms. Our results show larger gains where the performance gap between the small and large models is wide, demonstrating a promising method for large-small model collaboration for low-latency applications.

Code, dataset, and audio samples available at https://knowledgeboosting.cs.washington.edu/

Index Terms: Model collaboration, source separation

1. Introduction

Advancements in deep learning, hardware, and algorithms have enabled models to run on diverse devices, from wearables to GPU clusters. While some small models can run on-device, large models require remote servers or the cloud. Resource-constrained applications can greatly benefit from the knowledge capacity of larger models, but cannot easily utilize these models during inference. We pose the following question: Can a remote large model boost the performance of an on-device small model during low-latency inference? An affirmative answer would benefit real-time applications across various domains such as robotics, self-driving vehicles, and audio and video processing.

In this paper, we explore this question in the context of hearables and augmented audio applications, shown in Fig. 1, targeting real-time speech manipulation tasks such as target speech extraction [1, 2], speech enhancement [3, 4], and blind source separation [5]. Such applications have low latency requirements that demand real-time, streaming processing of small chunks of audio ($\leq$ 10 ms) and must operate on-device with limited computational resources. This requires models with minimal parameters and computational footprint [6, 7, 8]. Given these constraints, we investigate if a large model running, for instance, on a nearby smartphone, can boost the inference-time performance of a small model running on a wearable.

A key challenge is that communication latency between two devices can exceed real-time processing requirements. Data wirelessly transmitted to a remote device introduces a delay between current inputs of the on-device small model and the remote large model. For example, according to the Bluetooth 5.0 standard [9], the minimum delay is 15 ms roundtrip, but this can increase depending on wireless network congestion and interference. As a result, low-latency audio Bluetooth chips (e.g., Qualcomm aptX) guarantee only 40 ms latency [10]. Thus, the remote large model must operate on time-delayed input without access to the current audio chunks.

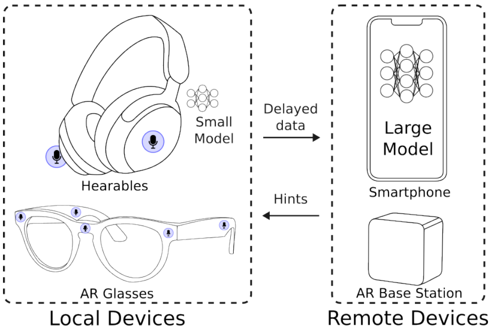

Figure 1: *Example use cases for knowledge boosting. During inference, small models running locally can benefit from the knowledge capacity of large models running remotely. In these examples, the communication latency between the local and remote devices can exceed the real-time processing requirements.*

In this paper, we introduce *knowledge boosting*, a novel technique in which delayed hints are provided by a large model to a small model during low-latency inference. Knowledge boosting enables a small model to accept hints after a time delay from a larger model, boosting its performance. Our key insight is that delayed large model information, when aligned with relevant history, can still enhance the current small model output. Further, through joint training, the large model can learn to provide useful hints that can improve real-time performance.

We evaluate our approach with very small models (around 40k parameters) that can fit on-device for wearables and large models (around 500k parameters) that can fit on-device on smartphones [6]. We test the performance of our models on three binaural audio tasks, namely blind source separation (SS), speech enhancement (SE), and target speech extraction (TSE), using a streaming version of TF-GridNet [11]. We also analyze our technique with ablation studies, through training configurations, compression ratios, and delays. Our results demonstrate that, at a delay of 48 ms (or six audio chunks), knowledge boosting improves the scale-invariant signal-to-distortion-ratio (SI-SDR) for SE, SS, and TSE by 0.23, 2.31, and 3.53 dB, respectively, over the corresponding vanilla models with a similar number of parameters as our on-device models. Our results demonstrate that the improvement from knowledge boosting de-

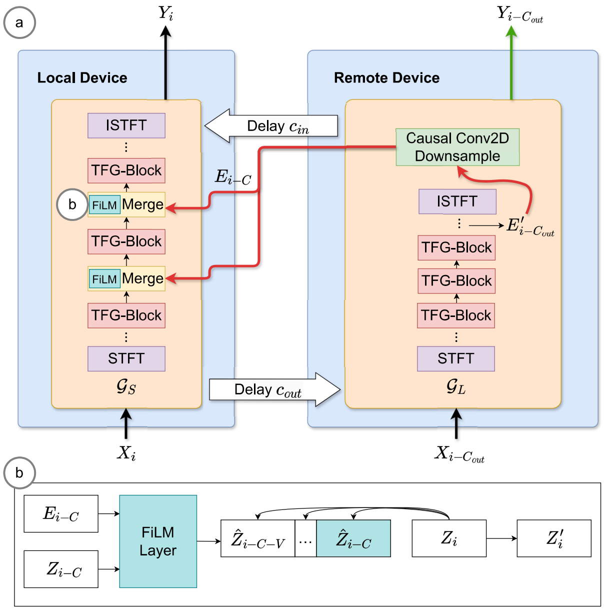

Figure 2: *Our system architecture. The green arrow is present only during large model pre-training. The red arrows are present during knowledge boosting. The black arrows are present both during pre-training and knowledge boosting.*

pends on the performance gap between small and large models, with large gains observed for TSE, and smaller gains for SE.

2. Related Work

Model partitioning: This involves taking one large model and determining an optimal partitioning point such that one part of the model runs on a smaller device, and the other part runs on the cloud or a larger device. Prior works [12, 13] have proposed model partitioning to adapt models to resource-constrained environments, executing only the necessary computation on-device and offloading the rest of the computation to the cloud. In this setting, both models have access to the same input at the same time. In contrast, our work aims to utilize a representation of the knowledge contained in a large model and use it to assist a small model after a delay. More importantly, because of communication and inference delay, the large model does not have access to the current small model input and has to work on previously received input samples.

Retrieval during inference and speculative decoding: In retrieval-augmented knowledge distillation [14], a student and teacher model are trained jointly to minimize divergence between their probability distributions. Embeddings from the teacher model are frozen into a database. The student model uses its output to look up the related embedding from the teacher database to assist during inference. Another technique, speculative decoding, provides a prompt to a draft student model and then uses a larger teacher model for verification [15, 16, 17]. The student model provides a set of proposals, or distributions for the target task. These proposals are then confirmed or denied by the large model. While the idea of using teacher knowledge to enhance student performance is similar to our work, these techniques do not take into account communication delay between the small and large model, the need for low latency inference, or model viability on small devices.

Knowledge distillation: This technique, first proposed in [18] transfers the knowledge of a large model to a small model during training for classification tasks. Variants of the original knowledge distillation proposal have been proposed for regression-based tasks [19], and more specifically for speech and audio tasks [20, 21, 22]. In contrast to these works, which distill knowledge during training, knowledge boosting transfers representations between a small and large model during inference time for boosting the small model's performance.

3. Knowledge Boosting

3.1. Problem formulation

Knowledge boosting utilizes two models — a small and a large model. The small model receives chunks of binaural audio chunks $X_1, \ldots, X_i$, where $X_i \in \mathbb{R}^{2\times\tau f_s}$, τ is the chunk duration in seconds, and f_s is the sampling rate. After receiving a chunk X_i, the local device sends it to the remote device, where it is received after a communication delay of c_{out} seconds, or C_{out} chunks. The remote device processes this chunk with a neural network $\mathcal{G}_\mathcal{L}$, computes an embedding E_i, and transmits it back to the local device. The embedding reaches the local device after a communication delay of c_{in} seconds, or C_{in} chunks. As a result, the embedding computed from the chunk X_i arrives back at the local device after a delay, $c = c_{in} + c_{out}$. During this time, the local device would have received $C = \lfloor \frac{c_{in}+c_{out}}{\tau} \rfloor$ additional chunks. Given our low-latency streaming requirements, the small model, through $\mathcal{G}_S$, must produce an output chunk Y_i while only using the information available to it by the time of input chunk, X_i, namely, the input chunks $X_1, \ldots, X_i$ and the embeddings from the large model, $E_1, \ldots, E_{i-C}$.

3.2. System architecture

We design our network architecture using the multi-channel causal TF-GridNet implementation [23]. Specifically, we use this network for both the small and the large models, $\mathcal{G}_S$ and $\mathcal{G}_\mathcal{L}$. Each network takes a time-domain binaural audio signal $x \in \mathbb{R}^{2\times t}$ of length t samples and uses a short-time Fourier transform (STFT) to convert it to a time-frequency representation $S \in \mathbb{C}^{2\times F\times T}$, where F is the number of frequency bins and T is the number of time frames. Then, a D-dimensional latent representation $Z \in \mathbb{R}^{D\times F\times T}$ is generated and processed with a sequence of TF-GridNet blocks, where the output of the j-th TF-GridNet block is Z^j. The output of the last TF-GridNet block is then mapped to K time-frequency domain channels, $\hat{S} \in \mathbb{C}^{K\times F\times T}$, and converted back to the time domain using an inverse STFT. Further details can be found in [11].

The large model generates embeddings that are received by the small model during training and inference. The embeddings are generated from the intermediate representation, $\hat{S}$, right before the inverse STFT in the TF-GridNet model. Specifically, we concatenate the real and imaginary components along the channel dimension, K, to generate the embedding, $E' \in \mathbb{R}^{2K\times F\times T}$. This embedding is passed through a compression module, which takes an input E' and outputs $E \in \mathbb{R}^{2K/P\times F\times T}$ where P is the compression ratio. The compression module is implemented as a single casual convolution layer with a kernel size of 3. The outputs of the compression module are then passed to the small model.

For a given chunk X_i, the small model computes an input representation Z_i^0 which it passes through a sequence of TF-GridNet blocks. The small model incorporates the compressed embeddings coming from the large model via the merge modules, shown in Fig. 2b. These modules are located in between consecutive TF-GridNet blocks. The j-th merge module takes two inputs— the latent representation, Z^j, from the output of the j-th TF-GridNet block and a time-delayed embedding, E_{i-C} from the large model. We use a context length V in our merge modules. Specifically, given the large model embedding, E_{i-C}, and the latent representation, Z^j_{i-C}, we compute the contextual representation, $\hat{Z}^j_{i-C}$, using a FiLM layer [24]. We then compute the merged output, Z'^j_i, using multi-head cross attention between $[\hat{Z}^j_{i-C-V}, \cdots, \hat{Z}^j_{i-C}]$ and Z^j_i. Z'^j_i is then provided as input to the $(j+1)$-th TF-GridNet block. During inference, to minimize computational complexity, we cache the last $C+V$ contextual representations computed previously.

3.3. Training procedure

We first pre-train the large model on the target task, which yields a reasonable initial set of weights. To train for knowledge boosting, we first process an input audio sequence with the large model to obtain the embeddings $E_1, \cdots, E_N$, where N is the total length of the audio sequence. Then, we simulate the communication latency by time-shifting the embedding sequence to the right by C, feeding in zeros to the first time frames where no embedding is available, and passing this sequence to the small model along with the original chunks $X_1, \ldots, X_N$. Both models are jointly trained, and the small model's output is used to compute the loss function for training. During backpropagation, we update the parameters of both the large and small models.

4. Experiments and Results

Datasets. To generate binaural audio mixtures, we first sampled rooms from one of four binaural room impulse response datasets – CIPIC [25], RRBRIR [26], ASH-Listening-Set [27], and CATTRIR [28] – with probability 0.35, 0.05, 0.45 and 0.15. We then sampled speech utterances from the LibriSpeech [29] dataset and convolved each of them with a binaural room impulse response from the sampled room. We summed up these binaural speech signals to obtain a binaural speech mixture. For the SS and TSE tasks, the resulting mixture was created from two binaural speech utterances. For SE, we only used a single speech utterance. Instead of using the room impulse response on a single-channel noise signal, which would only simulate a noise source in a single direction, we used a binaural noise signal recorded in the WHAM! [30] dataset as our ambient binaural noise. We scaled the noise so that the resultant average signal-to-noise ratio across both microphone channels is uniformly distributed between $[-6, 6]$ dB. For each task, we generated 100,000 mixtures for training, 5,000 for testing, and 5,000 for validation. Speech utterances were sampled from LibriSpeech's `train-clean-360`, `test-clean` and `dev-clean`, respectively. Noise samples were sampled from WHAM!'s `tr`, `tt`, and `cv`, respectively. All mixtures were 5 s long and the sampling rate was 16 kHz. There was no overlap in the identity of the speakers and the noise sources between the train, test, and validation splits.

Evaluation Setup. We compared the performance of a small model augmented with delayed hints from a large model during inference time with a vanilla model of a similar parameter size. Following the naming conventions in [11], D is the embedding dimension for each TF unit, B is the number of TF-GridNet blocks, I is the kernel size for unfold and Deconv1D, J is the stride size for unfold and Deconv1D, H is the number of hidden units in the LSTM layers and L is the number of heads in self-attention. In all our experiments, we set $I = J = 1$. We computed the time-frequency representations using uncentered 12 ms STFT windows with a hop size of 8 ms. We considered three baseline models per task. We trained a baseline small model with $D = 16, L = 4, B = 3, H = 16$, and no attention. We also trained a baseline medium model with $D = 26, L = 4, B = 3, H = 18$, and no attention. Finally, we trained a baseline large model with $D = 64, L = 8, B = 3, H = 64$ with attention. We limit self-attention to the last 50 chunks. After establishing these baselines, we trained knowledge boosting with a joint model training method. Specifically, we jointly trained a large model, with the same hyperparameters as our baseline large model, to boost the performance of a small model with the same hyperparameters as our baseline small model. In all experiments, we initialized the large model with the baseline large model weights, while the small model was not pre-trained. We use a context length of $V = 49$ for cross-attention modules.

Table 1: *Main results with delay of 48 ms ($C = 6$). The prefixes "Mx", and "KB" refer to the input mixture and knowledge boosting, respectively. The prefixes ”S”, ”M”, and ”L” refer to the small, medium, and large model baselines, respectively."Param." specifies the number of model parameters and "MACs" the number of multiply-accumulate operations. The MACs reported for KB configurations are for the small model only, as the small model is the only part that must run on a local device. For TSE, the speaker embedding parameters are excluded since they only need to be computed once and cached; this computation can occur on the remote device.*

Name	SI-SDR (dB)	PESQ	STOI	Param. (K)	MACs (M)
Mx-SS	0.00	1.28	0.70	-	-
M-SS	9.72	2.05	0.85	37.38	3.70
KB-SS	**12.03**	**2.23**	**0.87**	36.54	2.68
S-SS	8.65	1.93	0.84	23.96	2.40
L-SS	13.92	2.59	0.89	518.77	37.98
Mx-SE	0.01	1.12	0.67	-	-
M-SE	9.34	1.75	0.82	36.44	3.61
KB-SE	**9.57**	**1.77**	**0.83**	35.70	2.60
S-SE	9.01	1.71	0.82	23.38	2.35
L-SE	11.28	2.12	0.87	516.46	37.76
Mx-TSE	1.05	1.36	0.72	-	-
M-TSE	4.00	1.45	0.75	36.44	6.19
KB-TSE	**7.53**	**1.92**	**0.82**	35.70	4.19
S-TSE	3.95	1.42	0.75	23.38	3.94
L-TSE	8.52	2.21	0.84	516.46	44.12

Loss functions and training hyperparameters. For the TSE and SE tasks, we optimized the network parameters to maximize the average scale-invariant signal-to-distortion ratio (SI-SDR) [31] across the two microphone channels. For SS, we output two speaker channels per microphone, and used permutation invariant training to maximize the average SI-SDR across speakers and microphones. In all three tasks, since we were not particularly concerned about binaural cues, we treated the left and right channels independently when computing the optimal scale factor for SI-SDR. We used this same scale factor to rescale when computing PESQ and STOI. For all experiments, in each epoch, we iterated over the entire training and validation sets, and halved the learning rate if the average scale-invariant

Table 2: *Knowledge boosting ablation experiments on SS for different delay configurations (C) with no compression. The prefixes "KB", and "FKB" refer to knowledge boosting and knowledge boosting with a frozen large model, respectively.*

Name	C	SI-SDR (dB)	PESQ	STOI
FKB-SS	0	**13.50**	**2.47**	**0.89**
KB-SS	0	13.11	2.45	**0.89**
FKB-SS	1	10.73	2.04	0.87
KB-SS	1	**11.88**	**2.23**	**0.88**
FKB-SS	3	9.77	1.89	0.84
KB-SS	3	**11.21**	**2.07**	**0.86**
FKB-SS	6	9.27	1.93	0.84
KB-SS	6	**10.73**	**2.04**	**0.86**

Table 3: *Knowledge boosting ablation experiments on SS for different delay values with different compression factors (P).*

Name	C	P	SI-SDR (dB)	PESQ	STOI	MACs (M)
KB-SS	0	2	13.17	2.46	0.89	2.655
KB-SS	0	4	12.75	2.43	0.89	2.643
KB-SS	1	2	11.98	2.25	0.88	2.655
KB-SS	1	4	11.13	2.08	0.87	2.643
KB-SS	6	2	10.59	2.00	0.86	2.655
KB-SS	6	4	10.28	1.99	0.86	2.643

signal-to-noise ratio (SI-SNR) over the validation set does not decrease after four iterations. For all training runs, we used a batch size of 8 and gradient clipping with the norm set to 1. Our baseline models were trained for 100 epochs with an initial learning rate of 2e-3. Our main experiments were trained until convergence of the loss function with an initial learning rate of 1e-3. All of our ablation experiments were trained for 20 epochs. We used the Adam optimizer. In our evaluations, we used the best-performing weights on the validation set.

Main results. We tested the viability for knowledge boosting at a delay of 48 ms, or $C = 6$, and trained with a joint configuration, as specified above, for SS, SE, and TSE. We compared the results of a small model trained with knowledge boosting and a vanilla medium model of similar parameter size in Table 1. Overall, we observed that knowledge boosting tends to improve the performance of a small model over a vanilla model of a similar parameter size without knowledge boosting. We do so with a notable reduction in MACs. At delay $C = 6$, we achieved a relative improvement of 0.23, 2.31, and 3.53 dB for SE, SS, and TSE, respectively, over the vanilla medium models for each task. A paired t-test was conducted for each task, showing a significant difference with $p < 0.05$. As compared to the vanilla medium models for SE, SS, and TSE, respectively, we achieved a 1.01 M, 1.02 M, and 2.00 M reduction in multiply-accumulates (MACs) using knowledge boosting. This performance improvement despite MAC reduction was due to the fact that we cut down the length of computationally expensive units, such as LSTMs, reduced the dimensionality of embeddings used in TF-GridNet, and replaced these "missing" parameters with merge modules and delayed, but valuable hints from the large model.

Ablation studies. We performed ablation studies to evaluate the effects of large model weight freezing, compression, and different delays on the SS tasks, and evaluate the performance at different delays on the SE and TSE tasks. Specifically, for the SS task, we trained the large model and small model jointly, with a compression ratio $P = 1$ in Table 2, and swept delays $C = 0, 1, 3, 6$ chunks, corresponding to $0, 8, 24$, and 48 ms, respectively. In Table 2, we also investigated the effects of freezing the large model during training on overall performance, using $P = 1$ and sweeping $C = 0, 1, 3, 6$ chunks. In Table 3, we tested compression ratios $P = 1, 2, 4$ for $C = 0, 1, 6$ chunks. Finally, we also swept $C = 0, 1, 3, 6$ with $P = 1$ across SE and TSE in Table 4.

Table 4: *Knowledge boosting ablation experiments on TSE and SE for different delay configurations (C) with no compression.*

Name	C	SI-SDR (dB)	PESQ	STOI
KB-SE	0	11.24	0.87	2.11
KB-SE	1	10.64	0.85	1.95
KB-SE	3	9.88	0.84	1.82
KB-SE	6	9.57	0.83	1.77
KB-TSE	0	9.34	0.84	2.31
KB-TSE	1	8.20	0.83	2.05
KB-TSE	3	7.65	0.82	1.86
KB-TSE	6	7.70	0.82	1.89

Our results show performance degradation with larger C values. We found that freezing the large model during training led to lower performance than a jointly-trained large model. We also found that compression of the information sent from the large model to the small model slightly drops performance across C values. For $C = 6$, this drop was only 0.14 dB (p $>$ 0.3) and 0.45 dB (p $>$ 0.001) at compression factors, $P = 2$ and $P = 4$, respectively, compared to no compression.

5. Discussion and Limitations

Our work has a few limitations that present opportunities for future research. The device running the small model has to continuously stream audio to the remote device. However, it has been shown that wearable devices can do this over Bluetooth Low Energy (BLE) and still run continuously for 40 hours with only a coin cell battery [4]. The compression module impacts throughput requirements for transmitting embeddings from the remote to the small device. For the TSE task, uncompressed embeddings require a data rate of 1.55 Mbps, while a compression factor of 2 reduces it to 776 kbps. Notably, BLE supports a maximum rate of 2 Mbps [9]; and, recent work demonstrated concurrent streaming of compressed audio from seven microphones over BLE [32]. Future work could explore more specialized compression techniques like neural vocoders [33] to further reduce throughput requirements.

We trained different pairs of small and large models for each communication delay. While it is possible to train a single pair of small and large models to accept variably delayed input, we defer this to future work along with exploring alternate merge methods. Finally, we explored delays of up to 48 ms in the context of Bluetooth streaming from a wearable to a smartphone or a home base station. One could have larger models running on the cloud, but transmission delays would be higher. However, such models could have a much larger capacity, potentially leading to applications for more complicated tasks; we leave this exploration to future research.

6. Conclusion

We proposed knowledge boosting, a novel technique for improving the performance of small models at inference time through delayed hints from a large model. Our results show that knowledge boosting is a promising approach, worthy of further exploration, for large-small model collaboration during low-latency streaming applications.

7. Acknowledgments

The UW researchers are funded by the Moore Inventor Fellow award #10617, UW CoMotion fund, and the NSF.

8. 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*, vol. 40, no. 3, p. 8–29, May 2023.

[2] B. Veluri, M. Itani, T. Chen, T. Yoshioka, and S. Gollakota, "Look once to hear: Target speech hearing with noisy examples," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2024.

[3] H. Dubey, A. Aazami, V. Gopal, B. Naderi, S. Braun, R. Cutler, A. Ju, M. Zohourian, M. Tang, H. Gamper, M. Golestaneh, and R. Aichner, "ICASSP 2023 Deep Noise Suppression Challenge," 2023.

[4] I. Chatterjee, M. Kim, V. Jayaram, S. Gollakota, I. Kemelmacher, S. Patel, and S. M. Seitz, "Clearbuds: wireless binaural earbuds for learning-based speech enhancement," ser. ACM MobiSys '22.

[5] D. Wang and J. Chen, "Supervised speech separation based on deep learning: An overview," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 10, pp. 1702–1726, 2018.

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

[7] B. Veluri, J. Chan, M. Itani, T. Chen, T. Yoshioka, and S. Gollakota, "Real-Time Target Sound Extraction," in *Proc. ICASSP*, 2023, pp. 1–5.

[8] 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 *Proc. WASPAA*, 2023, pp. 1–5.

[9] *Bluetooth Core Specification v5.0*, 2016.

[10] "Qualcomm aptX audio is designed to improve Bluetooth sound quality." [Online]. Available: https://www.qualcomm.com/products/features/aptx

[11] Z.-Q. Wang, S. Cornell, S. Choi, Y. Lee, B.-Y. Kim, and S. Watanabe, "TF-GridNet: Integrating Full- and Sub-Band Modeling for Speech Separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 3221–3236, 2023.

[12] E. Li, Z. Zhou, and X. Chen, "Edge Intelligence: On-Demand Deep Learning Model Co-Inference with Device-Edge Synergy," in *Proc. MECOMM*, ser. MECOMM'18. Association for Computing Machinery, 2018, p. 31–36.

[13] L. Zeng, X. Chen, Z. Zhou, L. Yang, and J. Zhang, "CoEdge: Cooperative DNN Inference With Adaptive Workload Partitioning Over Heterogeneous Edge Devices," *IEEE/ACM Transactions on Networking*, vol. 29, no. 2, pp. 595–608, 2021.

[14] J. Zhang, A. Muhamed, A. Anantharaman, G. Wang, C. Chen, K. Zhong, Q. Cui, Y. Xu, B. Zeng, T. Chilimbi, and Y. Chen, "ReAugKD: Retrieval-augmented knowledge distillation for pre-trained language models," in *Proc. ACL*, 2023.

[15] Y. Leviathan, M. Kalman, and Y. Matias, "Fast inference from transformers via speculative decoding," in *Proc. ICML*. JMLR.org, 2023.

[16] X. Liu, L. Hu, P. Bailis, I. Stoica, Z. Deng, A. Cheung, and H. Zhang, "Online Speculative Decoding," 2023.

[17] M. Yan, S. Agarwal, and S. Venkataraman, "Decoding Speculative Decoding," 2024.

[18] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[19] M. Takamoto, Y. Morishita, and H. Imaoka, "An Efficient Method of Training Small models for Regression Problems with Knowledge Distillation," in *Proc. MIPR*, 2020, pp. 67–72.

[20] X. Chen, G. Liu, J. Shi, J. Xu, and B. Xu, "Distilled Binary Neural Network for Monaural Speech Separation," in *Proc. IJCNN*, 2018, pp. 1–8.

[21] X. Hao, S. Wen, X. Su, Y. Liu, G. Gao, and X. Li, "Sub-band knowledge distillation framework for speech enhancement," *arXiv preprint arXiv:2005.14435*, 2020.

[22] R. D. Nathoo, M. Kegler, and M. Stamenovic, "Two-Step Knowledge Distillation for Tiny Speech Enhancement," 2023.

[23] S. Cornell, Z. Wang, Y. Masuyama, S. Watanabe, M. Pariente, N. Ono, and S. Squartini, "Multi-Channel Speaker Extraction with Adversarial Training: The Wavlab submission to the clarity ICASSP 2023 grand challenge," in *Proc. ICASSP*, 2023, pp. 1–2.

[24] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville, "FiLM: visual reasoning with a general conditioning layer," in *Proc. AAAI*. AAAI Press, 2018.

[25] V. Algazi, R. Duda, D. Thompson, and C. Avendano, "The CIPIC HRTF database," pp. 99–102, 2001.

[26] IoSR-Surrey, "IoSR-surrey/realroombrirs: Binaural impulse responses captured in real rooms." https://github.com/IoSR-Surrey/RealRoomBRIRs, 2016.

[27] S. Pearce, "Shanonpearce/ash-listening-set: A dataset of filters for headphone correction and binaural synthesis of spatial audio systems on headphones," 2022. [Online]. Available: https://github.com/ShanonPearce/ASH-Listening-Set/tree/main

[28] IoSR-Surrey, "Simulated Room Impulse Responses." https://iosr.uk/software/index.php, 2023.

[29] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. ICASSP*, 2015, pp. 5206–5210.

[30] M. Maciejewski, G. Wichern, E. McQuinn, and J. L. Roux, "WHAMR!: Noisy and Reverberant Single-Channel Speech Separation," in *Proc. ICASSP*, 2020, pp. 696–700.

[31] J. L. Roux, S. Wisdom, H. Erdogan, and J. R. Hershey, "SDR – Half-baked or Well Done?" in *Proc. ICASSP*, 2019, pp. 626–630.

[32] M. Itani, T. Chen, T. Yoshioka, and S. Gollakota, "Creating speech zones with self-distributing acoustic swarms," *Nature Communications*, vol. 14, 09 2023.

[33] H. Wang, M. Yu, H. Zhang, C. Zhang, Z. Xu, M. Yang, Y. Zhang, and D. Yu, "Unifying Robustness and Fidelity: A Comprehensive Study of Pretrained Generative Methods for Speech Enhancement in Adverse Conditions," 2023.