

LLM-as-Joiner: Decoupling Alignment from Language Modeling in Label-synchronous ASR

Jaeyoung Lee^{1,2}, Masato Mimura¹, Ryo Magoshi², Tatsuya Kawahara²

¹ NTT, Inc., Japan

² Kyoto University, Japan

jae.lee@ntt.com, masato.mimura@ntt.com

Abstract

We present LLM-as-Joiner, an LLM-based ASR architecture that decouples speech-text alignment from language modeling. A speech encoder first converts an input of T acoustic frames into U token-level speech states, one per output token, following the Aligner-Encoder framework. A pretrained LLM then predicts the transcript from these U states, with speech injected at a chosen LLM layer; only upper LLM layers are adapted via LoRA. Compared to conventional decoder-only speech-as-prefix decoding, the LLM operates on a shorter context (U rather than $T + U$), reducing memory and compute. On LibriSpeech and multilingual Common Voice, the proposed LLM-as-Joiner outperforms size-matched baselines, and joint training with a lightweight recognizer yields additional gains in the lightweight head, suggesting implicit transfer of linguistic knowledge from the LLM to the speech encoder.

Index Terms: Speech recognition, Large language models (LLMs), Aligner-Encoders

1. Introduction

Modern end-to-end automatic speech recognition (ASR) is commonly approached with RNN-Transducer (RNN-T) models [1] or attention-based sequence-to-sequence frameworks [2, 3]. Progress in both lines has been driven by stronger speech encoders: the Conformer architecture [4] captures both local and long-range structure in speech, and large-scale self-supervised pretraining [5, 6, 7] further improves acoustic representations and also supports monotonic alignment from speech to text. Sequence-to-sequence approaches have further been popularized by Whisper-style systems [8], especially relevant in the LLM era as their decoders are architecturally close to language-modeling Transformers.

Decoder-only speech-as-prefix integration has become a common way to leverage strong linguistic priors from LLMs into ASR [9, 10, 11, 12]. While pretrained LLMs provide powerful text modeling, they are not pretrained with speech-text temporal alignment; consequently, decoder-only ASR must acquire this alignment during speech fine-tuning, often with additional alignment aids (e.g., compressors [10], boundary tokens [13], or forced alignment [12]). In ASR, alignment is monotonic and has conventionally been well addressed by speech encoders; asking a full LLM to infer alignment over long acoustic contexts could be inefficient. Moreover, speech-prefix decoding expands the LLM context and can dominate memory cost.

Recent label-synchronous ASR explores obtaining token-level structure without full-sequence decoder-side attention over encoder frames. Continuous Integrate-and-Fire (CIF) [14] offers an alternative monotonic alignment mechanism by predicting token boundaries and aggregating the T -frame sequence

into U token-level embeddings via integrate-and-fire, after which a decoder operates on these U states [14, 15]. Aligner-Encoders [16] push this further by simply applying frame-wise cross-entropy to encoder states at fixed label positions $1, \dots, U$ and decoding in $O(U)$ time, moving alignment into the encoder. Our method builds on Aligner-Encoders and leverages this U -length interface, particularly effective for LLM integration because it shortens the LLM context and removes the need for the LLM to learn speech-text alignment from scratch.

We propose *LLM-as-Joiner*: the speech encoder performs monotonic alignment and produces U label-synchronous states, while a pretrained LLM provides joiner-side language modeling and modality fusion. The LLM is integrated as-is, with no blank token and no added input/output embeddings, and its layers are partitioned into lower *predictor* and upper *joiner* roles, with speech conditioning injected at the partition; only the joiner-side layers are LoRA-tuned. Because the LLM runs on U positions rather than $T + U$ speech-prefix contexts, memory cost is reduced.

This work makes three main contributions: (1) We introduce a novel LLM-based ASR architecture that decouples monotonic alignment (learned by an Aligner-Encoder) from language modeling (provided by a pretrained LLM used as the joiner), where the LLM operates on U aligned label positions rather than a long speech-prefixed sequence. (2) We present a practical integration recipe that reuses the LLM with tokenizer/embeddings/LM head unchanged, injects speech at a chosen layer, and adapts only joiner-side layers via LoRA. (3) We show consistent WER improvements over size-matched baselines on LibriSpeech [17] and multilingual Common Voice [18] under a controlled setting without large pretrained speech encoders, and find that end-to-end training through the LLM joiner yields implicit knowledge transfer to the speech encoder.

2. Related Work

2.1. Label-synchronous ASR and Aligner-encoders

AED ASR (e.g., LAS and Transformer variants) is label-synchronous in that decoding advances one step per output token [2, 3], where alignment is typically realized via decoder-side attention over the length- T encoder frame sequence. An alternative line of work constructs token-level acoustic representations with explicit monotonic structure. Continuous Integrate-and-Fire (CIF) predicts monotonic token boundaries and aggregates a frame sequence into U token-level embeddings [14].

Aligner-Encoders [16], illustrated in Figure 2, further simplify training of alignment by supervising a fixed set of encoder positions directly with a token-level cross-entropy objective. Given a transcript of length U , Aligner-Encoders apply cross-

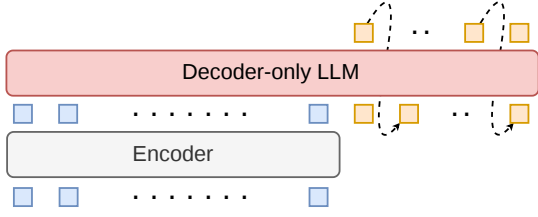


Figure 1: *Conventional Decoder-only LLM integration to ASR.*

entropy supervision to the encoder outputs at positions $1, \dots, U$ and ignore the remaining encoder states beyond U . The encoder’s self-attention is thereby encouraged to concentrate the relevant acoustic evidence for label position u ($1 \leq u \leq U$) into the corresponding encoder state, yielding an explicit U -length, label-indexed speech interface.

2.2. LLM-based ASR integration

Recent LLM-based ASR systems typically connect a speech encoder to a pretrained text LLM by *conditioning a decoder-only model on speech embeddings* (“speech-as-prefix”). As illustrated in Figure 1, the encoder produces a sequence of acoustic embeddings that are prepended to the text stream, and the decoder-only LLM autoregressively predicts text while attending over this speech prefix. Speech-LLaMA studies decoder-only speech-to-text and LLM integration, while CJST introduces a CTC-based compressor to shorten the speech prefix and trains speech/text jointly [10, 11]. A related line keeps the LLM largely unchanged and learns a lightweight adaptor between the encoder and LLM; SLAM-ASR shows that such simple compositions can be strong baselines [19]. In contrast, in our approach the LLM serves as the joiner and operates on U aligned label positions rather than a long speech-prefixed context.

2.3. LLMs within transducer-style factorisations

Several works aim to preserve or improve modularity of transducer models by isolating language-model-like components. HAT [20] characterizes and leverages an internal language model within a transducer formulation, and the Factorized Neural Transducer [21] separates blank and vocabulary prediction to facilitate language-model adaptation. More recently, Transducer-Llama integrates LLMs into streamable transducer-based ASR [22]. Our method differs in mechanism and objective: by adopting an Aligner-Encoder backbone, we avoid transducer lattice computation and blank-token integration, and instead apply an LLM as a joiner over a U -length interface.

3. Methodology

Figure 3 summarizes the proposed LLM-as-Joiner architecture, which decouples monotonic speech-text alignment from language modeling in a label-synchronous ASR framework. An Aligner-Encoder backbone learns the monotonic alignment and produces a sequence of U label-synchronous speech states consumed by downstream modules. We instantiate the predictor/joiner computation with a pretrained LLM by injecting the speech states at a split layer of the LLM via gated fusion; only the upper (joiner-side) layers are LoRA-tuned [23]. The LLM is reused as-is, without introducing a blank token or modifying its tokenizer, input embeddings, or LM head.

3.1. Aligner-Encoder

Given acoustic features $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_T)$ and a tokenized transcript $\mathbf{y} = (y_1, \dots, y_U)$ with $y_U = \langle \text{eos} \rangle$, we model

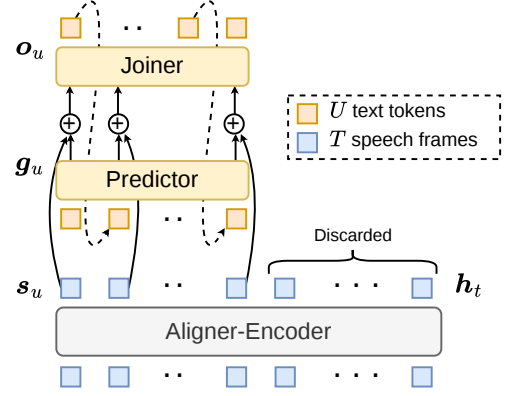


Figure 2: *Aligner-Encoder structure.*

$$p(\mathbf{y} | \mathbf{X}) = \prod_{u=1}^U p(y_u | y_{<u}, \mathbf{X}). \quad (1)$$

Aligner-Encoders [16] learn monotonic alignment within the encoder by applying token-level supervision only at fixed label positions $u \in \{1, \dots, U\}$. Because supervision is applied directly to encoder states at positions $1, \dots, U$, alignment is realized within the encoder via self-attention, rather than through decoder cross-attention over $\mathbf{H}$. The encoder maps frames to hidden states

$$\mathbf{H} = (\mathbf{h}_1, \dots, \mathbf{h}_T) = \text{Enc}(\mathbf{X}), \quad \mathbf{h}_t \in \mathbb{R}^d, \quad (2)$$

and we take the first U states as the label-synchronous speech interface $\mathbf{S} = (\mathbf{s}_1, \dots, \mathbf{s}_U)$ with $\mathbf{s}_u = \mathbf{h}_u$ (Figure 2).

To predict y_u , we use a predictor-joiner factorization:

$$\mathbf{g}_u = \text{Pred}(y_{<u}), \quad (3)$$

$$\mathbf{o}_u = \text{Join}(\mathbf{s}_u, \mathbf{g}_u), \quad (4)$$

$$p(y_u | y_{<u}, \mathbf{X}) = \text{softmax}(\mathbf{o}_u). \quad (5)$$

Training uses teacher forcing, and at inference the predictor conditions on previously emitted tokens. We compute $\mathbf{H}$ once and decode left-to-right, stopping when $\langle \text{eos} \rangle$ is emitted or when u reaches a preset maximum length. The token-level negative log-likelihood is

$$\mathcal{L} = - \sum_{u=1}^U \log p(y_u | y_{<u}, \mathbf{X}), \quad (6)$$

and decoding requires $O(U)$ joiner evaluations.

3.2. LLM-as-Joiner

We instantiate the predictor-joiner factorization (Section 3.1) with a pretrained decoder-only LLM by splitting it at layer ℓ : lower blocks act as the predictor and upper blocks as the joiner, with speech injected at the split. This split-and-inject design is motivated by analyses showing modality fusion emerges in upper decoder layers in AED [24, 25]. It also lets us condition the LLM on speech while reusing its pretrained input embeddings and LM head; using the LLM as the predictor would instead require an additional joiner-side output head. We treat the split layer ℓ as a hyperparameter; empirically, early to mid injection yields similar accuracy, while late injection degrades (Section 4.2). Given the token history, the embedding layer and lower blocks produce text states $\mathbf{G} = (\mathbf{g}_1, \dots, \mathbf{g}_U)$ under a causal mask; we fuse $\mathbf{G}$ with label-synchronous speech

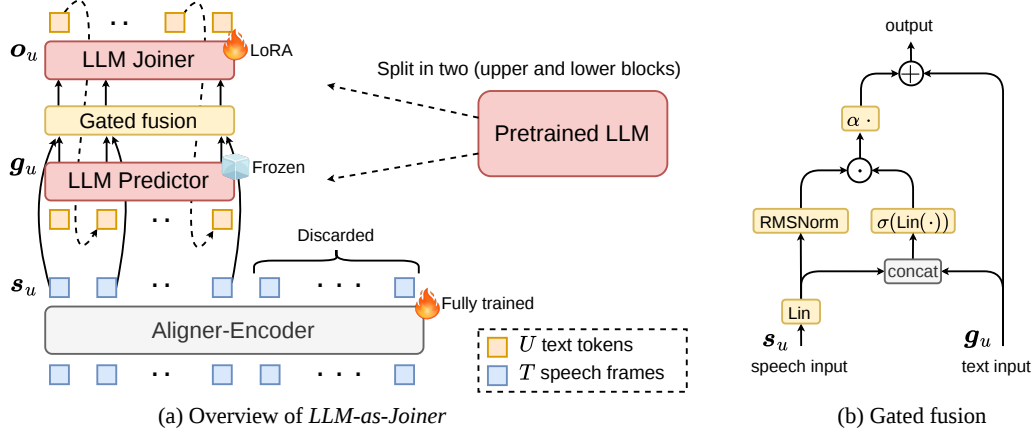


Figure 3: *LLM-as-Joiner*. We keep the autoregressive Aligner-Encoder objective (cross-entropy on label positions $1, \dots, U$), but replace the predictor/joiner with lower/upper blocks of a single pretrained LLM and inject speech at the split via position-wise gated fusion. The encoder processes T frames, while the LLM runs on U states (not $T + U$).

states $\mathcal{S}$ to form $\mathbf{F} = (\mathbf{f}_1, \dots, \mathbf{f}_U)$ (Section 3.3), which is then processed by the upper blocks and the native LM head to yield logits $\mathbf{o}_u$. We keep the LLM tokenizer, input embeddings, and LM head unchanged (no blank token and no added embeddings). All pretrained LLM weights are frozen; we train only LoRA adapters above ℓ plus fusion parameters and the speech encoder. Because the LLM is evaluated only on U positions, its memory scales with U rather than $T + U$.

3.3. Split-layer gated fusion

Speech conditioning is injected at the split layer via a position-wise gated residual (see Figure 3 (b)). We project each label-synchronous speech state to the LLM hidden dimension,

$$\tilde{\mathbf{s}}_u = \text{Lin}(\mathbf{s}_u), \quad (7)$$

where $\text{Lin}(\cdot)$ is a linear projection with bias. Let $\mathbf{g}_u$ denote the text hidden state produced by the lower LLM blocks at position u . We form the fused state

$$\mathbf{f}_u = \mathbf{g}_u + \alpha \cdot \sigma(\mathbf{z}_u) \cdot \text{RMSNorm}(\tilde{\mathbf{s}}_u), \quad (8)$$

where $\sigma(\cdot)$ is the sigmoid function and α is a trainable scalar. We explore three variants of how the gate logit $\mathbf{z}_u$ is computed:

$$\mathbf{z}_u = \begin{cases} \text{Lin}(\tilde{\mathbf{s}}_u) & (\text{speech-only}), \\ \text{Lin}(\mathbf{g}_u) & (\text{text-only}), \\ \text{Lin}([\tilde{\mathbf{s}}_u; \mathbf{g}_u]) & (\text{both}). \end{cases} \quad (9)$$

The resulting $\mathbf{F} = (\mathbf{f}_1, \dots, \mathbf{f}_U)$ replaces the hidden state at the split and is consumed by the upper LLM blocks, which realize the joiner computation.

3.4. Lightweight standalone recognizer

To obtain a deployable model without the LLM at inference, we attach a lightweight non-LLM head to the same Aligner-Encoder and train it jointly with the LLM-as-Joiner head. The lightweight head consumes the same label-synchronous speech states $\mathcal{S}$ and defines its own predictor and joiner: the predictor is a 1-layer LSTM [26] over the token history and the joiner is a small feed-forward network. The lightweight head predicts over a corpus-matched subset of LLM-tokenizer tokens to keep tokenization consistent on both training and test sets. At inference we can drop the LLM and decode using only the

lightweight head, while joint training encourages the shared encoder to learn linguistic structure induced by the LLM pathway without an explicit distillation loss.

3.5. Training objective

We train a single shared speech encoder with two autoregressive heads: an LLM-as-Joiner head and a lightweight head. Both heads use the token-level negative log-likelihood in Eq. 6; we denote these losses as $\mathcal{L}_{\text{llm}}$ and $\mathcal{L}_{\text{lite}}$ when Eq. 6 is evaluated under the LLM and lightweight distributions, respectively. We additionally apply an inter-CTC loss $\mathcal{L}_{\text{ctc}}$ [27] to encoder outputs at an intermediate layer. The overall objective is a weighted sum:

$$\mathcal{L} = \lambda_{\text{llm}} \mathcal{L}_{\text{llm}} + \lambda_{\text{lite}} \mathcal{L}_{\text{lite}} + \lambda_{\text{ctc}} \mathcal{L}_{\text{ctc}},$$

where we set $\lambda_{\text{llm}} = 1.0$, $\lambda_{\text{lite}} = 0.5$, and $\lambda_{\text{ctc}} = 0.1$.

4. Experimental evaluations

We evaluate on LibriSpeech and Common Voice 16.1 using a five-language subset (*cv-5langs*: de/en/es/fr/it) [17, 18]. For *cv-5langs*, we train a single multilingual model on the union of all five languages and report per-language WER on the test splits; for LibriSpeech we report WER on test-clean and test-other.

All Aligner-based systems use the same Conformer-L speech encoder (17 layers, 8 heads, hidden dim 2048) on 80-dim log-mel filterbank features, with inter-CTC loss at 12th layer, trained from scratch on the target dataset (except *Whisper-small*); thus our results are not directly comparable to LLM-ASR work using externally pretrained speech encoders.

For all LLM-based settings we use Llama-3.2-3B (base) [28] and inject speech at split layer $\ell \in \{0, 7, 14\}$ (of 28 layers; default $\ell = 7$), where $\ell = 0$ injects speech directly with the token embeddings (no lower LLM blocks). LoRA (rank 16) is applied only to attention modules above the injection point (Section 3).

For the lightweight head, we use a subset vocabulary (4,359 for LibriSpeech and 6,351 for *cv-5langs*) formed from the subset of tokens produced by the LLM tokenizer on the training transcripts. We use the Transformer learning rate schedule with the peak at 10^{-3} , 20k warmup steps, and subsequent decay, with an effective batch size of 70 minutes of audio per update. All systems are decoded with beam search (beam size 6) without external LM, rescoring, or test-time adaptation.

Table 1: LibriSpeech and Common Voice WERs (%). Except for Whisper-small, all systems use the same Conformer-L speech encoder trained from scratch. Real time factor (RTF) values are measured on a single RTX 6000 Ada GPU.

Model		Model Size	RTF	LibriSpeech	Common Voice 16.1					
				test-clean/other	de	en	es	fr	it	avg.
Aligner (Lite; from-scratch baseline)		118M	0.02	3.9 / 6.5	11.3	19.9	11.0	16.2	13.3	14.4
LLM Decoder-only baseline		3.3B	0.90	3.7 / 7.1		(did not converge)				
Whisper-small		241M	0.18	2.8 / 6.5	10.7	17.0	10.0	16.2	12.4	13.3
LLM-as-Joiner	Lite head	118M	0.02	3.8 / 6.4	9.4	17.9	9.5	14.6	11.2	12.5
	LLM head	3.3B	0.65	3.2 / 5.6	8.9	15.8	8.7	13.9	10.0	11.5

We train a single LLM-as-Joiner model with a shared Conformer encoder and two autoregressive heads: an LLM joiner head (*LLM head*) and a lightweight non-LLM head (*Lite head*); these differ only in the inference head. As a baseline, *Aligner (Lite; from-scratch)* uses the same lightweight head and vocabulary but is trained without the LLM pathway. We train Aligner-based systems for 100 epochs on LibriSpeech and 30 epochs on *cv-5langs*. Unless otherwise specified below, results are reported using model averaging over the best checkpoints on the validation set.

4.1. Baseline settings

We include two baselines in Table 1. *LLM Decoder-only* is a speech-as-prefix baseline (Figure 1) on LibriSpeech trained in two stages. (i) We first train the Conformer encoder only with a CTC objective for 50 epochs. (ii) We then attach the frozen encoder to a decoder-only LLM in a speech-as-prefix ASR setup and train a small speech adaptor together with LoRA-tuning of the LLM for 10 epochs, reporting parameter-averaged results over the best five checkpoints. We use two-stage training because it empirically outperformed one-stage training. The adaptor performs frame stacking with stack size 5 (similarly to [19]), followed by a trainable linear projection to 2048, a ReLU, and a second linear projection to the LLM embedding dimension. The LLM is adapted via LoRA with rank 16. To isolate architectural effects, this baseline uses the same from-scratch Conformer encoder and the same pretrained LLM as LLM-as-Joiner, and likewise does not use external pretrained speech encoders.

Whisper-small refers to multilingual Whisper-small [8], which we fully fine-tune for 10 epochs on each dataset (LibriSpeech and *cv-5langs*) and report parameter-averaged results over the best five checkpoints.

4.2. Results

Table 1 reports WER and real time factor (RTF) values on LibriSpeech and *cv-5langs*. On LibriSpeech, the proposed LLM-as-Joiner (LLM head) achieves 3.2/5.6 WER (test-clean/other), substantially improving over the *LLM Decoder-only* baseline (3.7/7.1), verified to be statistically significant at the 1% level. The decoder-only baseline did not converge on *cv-5langs*.

LLM-as-Joiner (Lite head), the lightweight Aligner head trained jointly with the LLM pathway, consistently outperforms the size-matched Aligner baseline (Lite, from-scratch). The gain is statistically insignificant on LibriSpeech (3.9/6.5 $\rightarrow$ 3.8/6.4), but significant on *cv-5langs* (e.g., de: 11.3 $\rightarrow$ 9.4; it: 13.3 $\rightarrow$ 11.2), suggesting implicit knowledge transfer from the pretrained multilingual LLM into the shared speech encoder despite the absence of an explicit distillation loss.

Notably, the lightweight head also outperforms Whisper-small fine-tuned on *cv-5langs* (e.g., de: 10.7 $\rightarrow$ 9.4; it: 12.4 $\rightarrow$ 11.2, statistically significant) except in English, while using roughly half the parameters (118M vs. 241M) and achieving

Table 2: Ablation on the LLM split layer (LibriSpeech WER %; clean/other).

Split layer	Lite head	LLM head
14th	4.1 / 6.6	4.0 / 5.8
7th	3.8 / 6.4	3.2 / 5.6
0th	3.8 / 6.5	3.2 / 5.6

Table 3: Ablation on the gating input (LibriSpeech WER %; clean/other) with split layer 7th.

Gating variant	Lite head	LLM head
Speech-only	3.9 / 6.5	3.4 / 5.8
Text-only	4.2 / 6.6	3.2 / 5.7
Both ($[h; g]$)	3.8 / 6.4	3.2 / 5.6

substantially lower latency (RTF 0.02 vs. 0.18, $\sim 9\times$ faster).

LLM-as-Joiner is also shown to improve decoding efficiency: the *LLM head* runs at RTF 0.65, compared to 0.90 for the LLM decoder-only baseline. This speedup follows from the shorter LLM context: in LLM-as-Joiner the LLM processes only U label positions, whereas decoder-only speech-as-prefix must attend over a speech prefix in addition to the generated tokens, yielding a longer effective context ($T + U$).

Tables 2 and 3 analyze the split layer and gating design. For the split layer, accuracy is similar for $\ell=0$ and $\ell=7$, indicating robustness to injection depth in this range. However, very late injection ($\ell=14$) degrades performance, implying effective fusion requires exposing enough LLM layers to speech. For gating, the LLM head is relatively insensitive to whether the gate conditions on speech: *text-only* matches *both* on test-clean (3.2) and is close on test-other (5.7 vs. 5.6). The lightweight head seems to benefit more from speech-conditioned gating: *speech-only* and *both* outperform *text-only*, indicating that conditioning the gate on acoustic information is important for transferring useful information from LLM to the shared speech encoder.

5. Conclusion

We presented *LLM-as-Joiner*, a label-synchronous ASR architecture that separates monotonic alignment from language modeling by letting an Aligner-Encoder produce a U -length token-level speech interface and applying a pretrained decoder-only LLM as the joiner. Across LibriSpeech and multilingual Common Voice, LLM-as-Joiner improves WER over size-matched baselines, and the jointly trained lightweight head benefits from implicit knowledge transfer from the LLM pathway, yielding strong multilingual gains. LLM-as-Joiner also improves decoding efficiency by avoiding a long speech-prefixed LLM context.

6. Generative AI Use Disclosure

Generative AI tools were employed to aid in code development and manuscript editing. The authors thoroughly reviewed and validated all generated content.

7. References

- [1] A. Graves, “Sequence transduction with recurrent neural networks,” 2012.
- [2] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, “Listen, attend and spell: A neural network for large vocabulary conversational speech recognition,” in *Proc. ICASSP 2016*, 2016.
- [3] L. Dong, S. Xu, and B. Xu, “Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition,” in *Proc. ICASSP 2018*, 2018, pp. 5884–5888.
- [4] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, “Conformer: Convolution-augmented transformer for speech recognition,” in *Proc. INTERSPEECH 2020*, 2020, pp. 5036–5040.
- [5] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, “wav2vec 2.0: A framework for self-supervised learning of speech representations,” in *Advances in Neural Information Processing Systems*, 2020.
- [6] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhota, R. Salakhutdinov, and A. Mohamed, “Hubert: Self-supervised speech representation learning by masked prediction of hidden units,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3451–3460, 2021.
- [7] S. Chen, C. Wang, Z. Chen, Y. Wu, S. Liu, Z. Chen, J. Li, N. Kanda, T. Yoshioka, X. Xiao, J. Wu, L. Zhou, S. Ren, Y. Qian, Y. Qian, J. Wu, M. Zeng, X. Yu, and F. Wei, “Wavlm: Large-scale self-supervised pre-training for full stack speech processing,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 6, pp. 1505–1518, 2022.
- [8] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, “Robust speech recognition via large-scale weak supervision,” in *Proc. ICML 2023*, 2023.
- [9] Y. Fathullah, C. Wu, E. Lakomkin, J. Jia, Y. Shangguan, K. Li, J. Guo, W. Xiong, J. Mahadeokar, O. Kalinli, C. Fuegen, and M. Seltzer, “Prompting large language models with speech recognition abilities,” in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024, pp. 13 351–13 355.
- [10] J. Wu, Y. Gaur, Z. Chen, L. Zhou, Y. Zhu, T. Wang, J. Li, S. Liu, B. Ren, L. Liu, and Y. Wu, “On decoder-only architecture for speech-to-text and large language model integration,” in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop (ASRU) 2023*, Taipei, Taiwan, Dec. 2023, pp. 1–8.
- [11] W. Zhou, J. Jia, L. Sari, J. Mahadeokar, and O. Kalinli, “CJST: CTC compressor based joint speech and text training for decoder-only ASR,” 2024.
- [12] J. Jia, G. Keren, W. Zhou, E. Lakomkin, X. Zhang, C. Wu, F. Seide, J. Mahadeokar, and O. Kalinli, “Efficient streaming LLM for speech recognition,” 2024.
- [13] P. Chen, S. Sun, C. Shan, Q. Yang, and L. Xie, “Streaming decoder-only automatic speech recognition with discrete speech units: A pilot study,” in *Proc. INTERSPEECH 2024*, Kos Island, Greece, Sep. 2024, pp. 4468–4472.
- [14] L. Dong and B. Xu, “CIF: Continuous integrate-and-fire for end-to-end speech recognition,” 2019.
- [15] Z. Gao, S. Zhang, I. McLoughlin, and Z. Yan, “Paraformer: Fast and accurate parallel transformer for non-autoregressive end-to-end speech recognition,” 2022.
- [16] A. Stooke, R. Prabhavalkar, K. C. Sim, and P. Moreno Mengibar, “Aligner-encoders: Self-attention transformers can be self-transducers,” in *Advances in Neural Information Processing Systems*, 2024.
- [17] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “LibriSpeech: An ASR corpus based on public domain audio books,” in *Proc. ICASSP 2015*, 2015, pp. 5206–5210.
- [18] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber, “Common voice: A massively-multilingual speech corpus,” in *Proceedings of the Twelfth Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, 2020, pp. 4218–4222.
- [19] Z. Ma, G. Yang, Y. Yang, Z. Gao, J. Wang, Z. Du, F. Yu, Q. Chen, S. Zheng, S. Zhang, and X. Chen, “An Embarrassingly Simple Approach for LLM with Strong ASR Capacity,” 2024.
- [20] E. Variani, D. Rybach, C. Allauzen, and M. Riley, “Hybrid autoregressive transducer (HAT),” in *Proc. ICASSP 2020*, 2020, pp. 6139–6143.
- [21] X. Chen, Z. Meng, S. Parthasarathy, and J. Li, “Factorized neural transducer for efficient language model adaptation,” in *Proc. ICASSP 2022*, 2022.
- [22] K. Deng, J. Guo, Y. Ma, N. Moritz, P. C. Woodland, O. Kalinli, and M. Seltzer, “Transducer-llama: Integrating LLMs into streamable transducer-based speech recognition,” 2024.
- [23] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “LoRA: Low-rank adaptation of large language models,” in *International Conference on Learning Representations (ICLR)*, 2022. [Online]. Available: <https://openreview.net/forum?id=nZeVKeeFYf9>
- [24] D. Prabhu and P. Jyothi, “Blockdecoder: Boosting asr decoders with context and merger modules,” in *Proceedings of NeurIPS*, 2025.
- [25] C. Chen, R. Li, Y. Hu, S. M. Siniscalchi, P. Chen, E. Chng, and C. H. Yang, “It’s never too late: Fusing acoustic information into large language models for automatic speech recognition,” in *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7–11, 2024*, 2024.
- [26] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [27] J. Lee and S. Watanabe, “Intermediate loss regularization for ctc-based speech recognition,” in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 6224–6228.
- [28] A. Grattafiori *et al.*, “The llama 3 herd of models,” 2024. [Online]. Available: <https://arxiv.org/abs/2407.21783>