LLMs and Speech: Integration vs. Combination

Robin Schmitt, Albert Zeyer, Mohammad Zeineldeen*, Ralf Schlüter, Hermann Ney*

Machine Learning and Human Language Technology Group, Faculty of Computer Science,
RWTH Aachen University, Aachen, Germany
AppTek GmbH, Aachen, Germany

{ schmitt,zeyer,zeineldeen,schlueter,ney }@ml.rwth-aachen.de

Abstract

In this work, we study how to best utilize pre-trained LLMs for automatic speech recognition. Specifically, we compare the tight integration of an acoustic model (AM) with the LLM ("speech LLM") to the traditional way of combining AM and LLM via shallow fusion. For tight integration, we provide ablations on the effect of different label units, fine-tuning strategies, LLM sizes and pre-training data, attention interfaces, encoder downsampling, text prompts, and length normalization. Additionally, we investigate joint recognition with a CTC model to mitigate hallucinations of speech LLMs and present effective optimizations for this joint recognition. For shallow fusion, we investigate the effect of fine-tuning the LLM on the transcriptions using different label units, and we compare rescoring AM hypotheses to single-pass recognition with label-wise or delayed fusion of AM and LLM scores. We train on Librispeech and Loquacious and evaluate our models on the HuggingFace ASR leaderboard.

Index Terms: speech recognition, large language models, shallow fusion

1. Introduction & Related Work

Traditionally, speech recognition systems consist of an acoustic model (AM) and a separate language model (LM), which allows to utilize large text-only corpora in addition to the typically smaller speech corpora. Multiple approaches for the combination of AMs and LMs have been proposed over the years, such as shallow fusion, deep fusion [1], cold fusion [2], internal language model correction [3], and sequence discriminative training [4]. In recent years, many techniques utilizing large language models (LLMs) for automatic speech recognition (ASR) have been proposed.

Delayed fusion [5] allows for efficient single-pass combination of AM and LLM scores and aleviates the problem of label unit mismatch. Transducer-LLaMA [6] is a transducer-based ASR system which uses a pre-trained LLM as the prediction network. The most common approach, commonly referred to as *speech LLM*, is to train an LLM to also accept acoustic information from an audio encoder as input. While earlier works discretized the encoder output and extended the LLM vocabulary to include audio tokens [7, 8], current approaches directly use the continuous encoder output. Different interfaces between encoder and LLM have been proposed. Most works [9–14] prepend the encoder output to the LLM input and use self-attention to attend over the combined sequence. The self-attention on the prefix part can be causal or bidirectional [15], commonly referred to as causal decoder-only and prefix LLM [16], respectively. Here, we refer to both as prefix LLMs (PLLM), since the term decoder-only might be misleading given that we use a non-trivial encoder. Prefix LMs can use interleaved audio-text inputs to enable streaming [17]. [18, 19] add gated cross-attention layers to the LLM decoder, effectively turning it into an attention-based encoder-decoder (AED) model. Typically, pre-trained text LLM are fine-tuned to include multiple modalities. However, more recently, [20, 21] trained multi-modal LLMs from scratch by including text, audio, image, and video data in the pre-training data.

While most works on speech LLMs investigate a large set of downstream tasks to effectively utilize the broad range of capabilities of the LLM, few focus specifically on ASR performance [10, 11, 14]. What is largely missing from existing literature is a consistent comparison between speech LLMs and classical ASR systems under comparable conditions to allow for a comparison of the different approaches to utilize LLMs for ASR, rather than evaluating the differences in training data exploited. To the best of our knowledge, [13] and [11] are the only exceptions. [13] compare their model to greedy decoding of their baseline CTC model but this is not fair due to the missing label context in CTC. [11] compare their model to their baseline AED model but do not state whether the training time is consistent between the two models. In other works, different training sets are used to train the baseline ASR model and the speech LLM. Furthermore, we are not aware of any work systematically comparing the tight integration of AM and LLM versus the traditional way of combining AM and LLM via shallow fusion. In this work, we pose the question whether using pre-trained text-based LLMs is beneficial for ASR performance and, if so, how to best utilize them. Our contributions are as follows:

- For speech LLMs, we conduct ablations on the effect of different label units, fine-tuning strategies, LLM sizes and pre-training data, attention interfaces, encoder downsampling, text prompts, beam sizes, and length normalization.
- We investigate joint CTC recognition to mitigate hallucinations of speech LLMs and present novel optimizations for this joint recognition.
- We compare speech LLMs to baseline ASR systems which were only trained on the paired ASR data.
- We compare tight integration of encoder and LLM to shallow fusion of CTC and LLM scores.

To our knowledge, this is the first investigation of such extent of using LLMs for ASR and the first work to systematically compare the different approaches to utilize LLMs for ASR under comparable conditions.

*These authors contributed equally.

2. Shallow Fusion Baseline

Acoustic Encoder. All our models utilize a Conformer-based encoder [22]. The input of 10ms log mel features is first processed by a convolutional front-end with a subsampling factor of 6. This sequence is further processed by a Conformer to produce the encoder output

$$h_1^T = \text{Encoder}(x_1^{T'}),$$

where $T = \lceil T'/6 \rceil$ and T' is the length of the input sequence.

Acoustic CTC output. On top of the encoder, we have a linear projection to the label vocabulary dimension including blank, and we apply CTC [23] (for training and optionally also for recognition):

$$p_{\text{CTC}}(y_t \mid h_t) = \text{softmax}(\text{Linear}(h_t))[y_t],$$

$$p_{\text{CTC}}(a_1^S \mid h_1^T) = \sum_{y_1^T \in \mathcal{B}^{-1}(a_1^S)} \prod_{t=1}^{T} p_{\text{CTC}}(y_t \mid h_t),$$

where $\mathcal{B}$ is the CTC collapsing function which removes blanks and repeated labels.

Language model. We use a standalone Transformer-based language model

$$p_{\text{LM}}(a_1^S) = \prod_{s=1}^{S} p_{\text{LM}}(a_s \mid a_0^{s-1})$$

where a_0 denotes the `BOS` token and a_S denotes the `EOS` token. We use the perplexity metric to evaluate our language models:

$$\text{PPL}(p_{\text{LM}}, a_1^S) = \exp\left(-\frac{1}{S}\sum_{s=1}^{S} \log p_{\text{LM}}(a_s \mid a_0^{s-1})\right).$$

2.1. Training

We train the acoustic model using the CTC loss [23]

$$L_{\text{CTC}} = -\log p_{\text{CTC}}(a_1^S \mid h_1^T).$$

The language model is trained separately using the standard cross-entropy loss on the text data

$$L_{\text{LM}} = -\log p_{\text{LM}}(a_1^S).$$

We optionally use a pre-trained LLM as starting point and fine-tune it on the transcription data.

2.2. Recognition

We use time-synchronous beam search [24] for recognition. In case of a vocabulary mismatch between the acoustic model and the language model, we use time-synchronous beam search with delayed fusion [5].

3. Integrated Models

Encoder and CTC output. We use the same encoder and CTC output as in the shallow fusion baseline.

Adapter. The adapter can optionally further downsample the encoder output, and then optionally performs a linear transformation to match the decoder input dimension. The downsampling is applied directly to the encoder output h_1^T. In the simple case, we concatenate consecutive frames [25], which is the same as applying a 1D convolution with kernel size and stride equal to the downsampling factor. We also use a variant of *CTC compression* [26–28] which dynamically compresses the encoder output by taking the probabilities of the auxiliary CTC model into account.

$$\begin{aligned}
\hat{y}_t &= \text{argmax}_y\, p_{\text{CTC}}(y \mid h_t), \\
m_t &= (\hat{y}_t = \hat{y}_{t-1}) && \text{(merge mask)} \\
&\quad \wedge (p_{\text{CTC}}(\hat{y}_t \mid h_t) \geq \tau) \\
&\quad \wedge (p_{\text{CTC}}(\hat{y}_{t-1} \mid h_{t-1}) \geq \tau), \\
i_t &= \begin{cases} 1 & \text{if } t = 1 \\ i_{t-1} & \text{if } m_t, t \geq 2 \\ i_{t-1} + 1 & \text{if } \neg m_t, t \geq 2 \end{cases} && (1)
\end{aligned}$$

where τ is a threshold hyperparameter. Then we can compress the encoder output by pooling over the frames with the same index i_t:

$$\text{compress}(h_1^T)_i = \text{mean}(\{h_t \mid i_t = i\}), \quad i \in 1, \ldots, i_T.$$

Decoder and attention interface variants. The probability of the output label sequence a_1^S given the encoder output h_1^T is defined as

$$p_{\text{Dec}}(a_1^S \mid h_1^T) = \prod_{s=1}^{S} p(a_s \mid a_0^{s-1}, h_1^T).$$

At each step s, the Transformer-based decoder [29] autoregressively predicts the next label a_s by attending over the encoder output h_1^T. We define $a_0 = \texttt{BOS}, a_S = \texttt{EOS}$ as the beginning- and end-of-sequence token, respectively, where `EOS` implicitly models the probability of the sequence length.

We investigate models with different attention interfaces for connecting the encoder and the decoder. *AED models* [29–31] use dedicated *cross-attention* modules, where the queries are derived from the decoder and the keys and values are derived from the encoder. *Prefix language models (LMs)* [12–14] *prepend* the encoder output to the decoder input and use self-attention to attend over the combined sequence, i.e., the queries, keys, and values are all derived from the combined sequence. *Merged attention models* [32] can be seen as a mixture of the previous two, where the queries are derived from the decoder, but the keys and values are derived from the combined sequence. Compared to prefix LLMs (PLLMs), merged attention reduces computation since the feed-forward layers only need to process the text part of the combined sequence, while still allowing to use a pre-trained LLM decoder without changing the architecture.

3.1. Training

We train our model using the standard label-wise cross-entropy (CE) criterion

$$L_{\text{Dec}} = -\log p_{\text{Dec}}(a_1^S \mid h_1^T).$$

We also add CTC [23, 33] as a loss

$$L_{\text{CTC}} = -\log p_{\text{CTC}}(a_1^S \mid h_1^T),$$

which is either just an auxiliary loss for training or also used for joint recognition.

We optionally use a pre-trained acoustic model as starting point for the encoder parameters, and a pre-trained LLM as starting point for the decoder parameters.

Table 1: *ASR dataset statistics. For transcriptions, we compare our 10K SPM ASR vocabularies to the 150K BPE vocabulary of the Qwen models and we report the percentage of unique labels in each vocabulary that are used in the transcripts.*

Dataset	# Label units [K]	# Frames [M]			Vocab. usage [%]	# Seqs. [M]
		Audio	Transcript	Σ		
Loquacious	150	1,500	265	1,765	26	9.5
	10		271	1771	100	
Librispeech	150	60	10	70	19	0.3
	10		10	70	100	

Table 2: *Details of the used LMs and their training data. Comparing the pre-trained Qwen models to our own model which is trained on the ASR transcription data. For training frames (tokens), we assume the reported numbers for Qwen account for a single epoch.*

LM	Size [B]	# Label units	# Training frames	# Layers	Model dim.
Qwen2	0.5	150K	12T	24	896
	1.5		7T	28	1,536
	7.0				3,584
Qwen3	0.6		36T		1,024
Ours (Loquacious)	0.4	10K	271M	32	1,024
Ours (Librispeech)			10M		

3.2. Recognition

For recognition, we use label-synchronous beam search on $p_{\text{Dec}}(a_1^S \mid h_1^T)$ and optionally log-linearly combine with the prefix scores of the CTC output $p_{\text{CTC}}(a_1^S \mid h_1^T)$ as in [33].

For the CTC output, we also optionally apply CTC compression using Equation (1) with threshold τ as

$$s_{\text{CTC}'}(y_i \mid h_1^T) = \max_{t|i_t=i} p_{\text{CTC}}(y_t{=}y_i \mid h_t), \quad i \in 1, \dots, i_T$$

$$p_{\text{CTC}'}(y_i \mid h_1^T) = \frac{s_{\text{CTC}'}(y_i \mid h_1^T)}{\sum_{y'} s_{\text{CTC}'}(y' \mid h_1^T)}.$$

For the CTC output, we also investigate top-k pruning of the CTC output distribution by taking the top-k labels over the CTC probabilities after taking the maximum over the time dimension:

$$s_{\max}(y \mid h_1^T) = \max_t p_{\text{CTC}}(y_t{=}y \mid h_t),$$

$$\text{TopK}(s_{\max}, k) \subset \mathcal{Y} \Leftrightarrow \begin{cases} |\operatorname{TopK}(\dots)| = k, \\ \min_{y \in \text{TopK}} s_{\max}(y) \geq \max_{y \notin \text{TopK}} s_{\max}(y) \end{cases}$$

$$s_{\text{topk}}(y_t \mid h_t) = \begin{cases} p_{\text{CTC}'}(y_t \mid h_t) & y \in \operatorname{TopK}(\dots) \\ 0 & \text{otherwise} \end{cases}$$

$$p_{\text{CTC}*}(y_t \mid h_t) = \frac{s_{\text{topk}}(y_t \mid h_t)}{\sum_{y'} s_{\text{topk}}(y' \mid h_t)}.$$

Intuitively, this only keeps labels that get a high CTC probability in at least one frame.

When not using the CTC output, by default, we use length normalization by dividing the step-wise log-probability $p_{\text{Dec}}(a_1^{\bar{S}} \mid h_1^T)$ by the intermediate sequence length $\bar{S}$. This choice is motivated by our experience with AED models, which tend to perform worse without length normalization.

4. Experiments

For training, we use Librispeech 960h [34] and the *large* split of the Loquacious [35] corpus, which contains 25K hours of speech data. We apply on-the-fly speed perturbation and SpecAugment [36]. We show statistics of the datasets in Table 1. We use RETURNN [37] with PyTorch [38] backend for training and recognition and we use ESPnet [39] to calculate the CTC prefix scores for shallow fusion. We will release all relevant code upon publication.

Table 3: *Details of our baseline ASR and prefix LLM models. The number of encoder parameters includes the CTC projection matrix of the auxiliary layers.*

Model	Pre-trained Decoder	# Label units [K]	# Params [M]			
			Encoder	Adapter	Decoder	Σ
Baseline AED	-	10	417	-	133	550
Prefix LLM	Qwen2 0.5B	150	612	2	367	786
					494	1,108
	Qwen2 1.5B				1,544	2,158
	Qwen2 7B				7,616	8,230

We train baseline AED, prefix LLM, and merged attention models from scratch. All baseline models are either trained for 20 epochs on Loquacious or for 100 epochs on Librispeech. Next, we train speech LLMs by using the baseline AED encoder together with a pre-trained LLM for the decoder as the starting point. By default, we use the 0.5B version of Qwen2 [40] as the pre-trained LLM. For fine-tuning, if not stated otherwise, we train for 1 epoch on Loquacious and for 10 epochs on Librispeech. The weights of the adapter are always initialized randomly. For the decoder weights, we experiment with full fine-tuning and low rank adaptation (LoRA) [41]. By default, we apply rank-stabilized LoRA [42] adapters with a rank of 320 on the query and value projection matrices of the LLM attention blocks, as is done in [13]. We also experiment with a 2-stage training scheme, where we first only train the newly initialized weights and then fine-tune the whole model jointly.

All our baseline ASR models use a 16-layer Conformer encoder and a 6-layer Transformer decoder with 10K sentencepiece (SPM) subword label units [43]. When training speech LLMs, we add two more randomly initialized Conformer layers on top of our encoder to better adapt to the label units of the LLM. For all models, we add an auxililary CTC loss on top of the encoder output which uses the same label units as the decoder.

For the LLM, we investigate using different sizes of Qwen2 and Qwen3 [44]. Furthermore, we compare using our own language model which is trained on the ASR transcriptions in case of Loquacious and on the external LM data in case of Librispeech. We compare details of the different LMs in Table 2 and of the different ASR models in Table 3.

To improve the perplexity of the pretrained LLMs on our tasks, we fine-tune the LLMs on the ASR transcription data and if available, we also utilize external text data. We train for 2 epochs for all tasks. We apply full fine-tuning for Qwen2 0.5B and Qwen2 1.5B LLMs and apply LoRA for the Qwen2 7B LLM to reduce computational costs. Specifically, for Qwen2 7B, we apply rank-stabilized LoRA with rank of 64 and alpha of 128 to all linear layers except the embedding layer and with a LoRA dropout of 0.1. We also fine-tune the LLMs using the ASR SPM vocab where the emebedding layer is intialized based on the pretrained LLM embeddings as proposed in [6]. For LR scheduling, we use a warmup phase of 5% of total training epochs followed by a cosine decay to a minimum learning rate of 1e-7. For the full fine-tuning trainings, we use a peak learning rate of 1e-5 while for the LoRA trainings, we use a peak learning rate of 5e-6.

The training transcripts for Loquacious and Librispeech are fully uppercased. We lowercase them to reduce the vocabulary

Table 4: *Comparison of **pretrained vs. from scratch** on Loquacious. The AED model is trained from scratch for 20 epochs, while the prefix LLM is initialized with the baseline AED encoder and the Qwen2 0.5B decoder and fine-tuned for 2 epoch using LoRA for the decoder.*

Model	Pre-trained decoder	WER [%]			
		Standalone		Joint CTC	
		dev	test	dev	test
AED	-	5.84	6.47	5.40	5.96
Prefix LLM	Qwen2 0.5B	6.00	6.99	5.79	6.38

Table 5: *Effect of **different number of epochs** when training prefix LLMs (PLLM). All models are initialized with the baseline AED encoder and the Qwen2 0.5B decoder. We fine-tune for either 1 or 2 epochs on Loquacious using LoRA for the decoder.*

# Epochs	WER [%]			
	PLLM		PLLM +CTC	
	dev	test	dev	test
1	6.43	7.03	5.99	6.55
2	6.00	6.99	5.79	6.38

Table 6: *Variants of log-linear combination of CTC with standalone LM on Loquacious. PPL is computed on the word-level. The optimal search settings for each combination are used, which can differ between the different combinations, and is restricted by GPU memory.*

CTC	LM		dev		test	
Label unit	Name	Label unit	PPL	WER	PPL	WER
150K BPE	-	-	-	6.64	-	7.26
	Qwen2 0.5B	150K BPE	59	5.73	60	6.25
10K SPM	-	-	-	5.94	-	6.51
	Our LM	10K SPM	72	5.37	73	5.94
	Qwen2 0.5B	150K BPE	59	5.29	60	5.84
		10K SPM	60	5.30	60	5.82
	Qwen2 1.5B	150K BPE	53	5.22	53	5.78
		10K SPM	53	5.24	53	5.80
	Qwen2 7B	150K BPE	45	5.18	45	5.73

mismatch between the ASR transcription data and the LLM pre-training data, which we found to improve performance.

4.1. From Scratch vs. Fine-tuning

We compare the performance of our baseline AED model with a fine-tuned prefix LLM in Table 4. Even though the prefix LLM is initialized with the AED encoder, its performance after 2 epochs of fine-tuning is significantly worse (5.79% vs. 5.40% on dev). This suggests that it is not trivial to leverage the pre-trained LLM decoder for ASR without longer training.

4.2. Number of Epochs

We compare the effect of fine-tuning for different number of epochs in Table 5. As expected, fine-tuning for more epochs leads to better performance (5.99% vs. 5.79% on dev). However, because of computational constraints, we focused on having more ablations with fewer epochs rather than fewer ablations with more epochs.

4.3. Shallow Fusion Recognition

We compare log-linear combination of different CTC models and separate LMs in Table 6. We see that the best performance is achieved by using the largest LLM, Qwen2 7B, as a pure LM in combination with CTC. We study the effect of different beam sizes in Table 7. Already with a beam size of 8, we are close to the best performance achieved with a beam size of 64.

Table 7: *Effect of beam size for first-pass time-synchronous CTC + LM recognition with Qwen2 0.5B as pure LM fine-tuned with the ASR SPM10k vocabulary on Loquacious transcriptions. WERs [%] on Loquacious.*

Beam Size	Dev WER	Test WER
1	5.75	6.40
2	5.48	6.05
4	5.36	5.91
8	5.32	5.87
12	5.31	5.84
16	5.30	5.84
32	5.30	5.83
64	5.30	5.82

Table 8: *Effect of **different attention interfaces when training from scratch** on Librispeech. All models are trained for 100 epochs and use the same encoder architecture. Results with time-synchronous joint CTC decoding.*

Model	WER [%]			
	dev-clean	dev-other	test-clean	test-other
AED	1.87	4.21	2.04	4.37
Prefix LM	1.96	4.28	2.11	4.46
Merged attention	2.00	4.20	2.12	4.56

Table 9: *Effect of **different attention interfaces when fine-tuning**. All models are initialized with the baseline AED encoder and the Qwen2 0.5B decoder. We fine-tune for 10 epochs on Librispeech and for 1 epoch on Loquacious. Results with label-synchronous joint CTC decoding.*

Model	WER [%]					
	Librispeech				Loquacious	
	dev-clean	dev-other	test-clean	test-other	dev	test
Prefix LLM	2.22	4.81	2.44	5.15	5.99	6.55
Merged attention	2.30	5.15	2.44	5.48	6.10	6.62

4.4. Attention Interfaces

We compare training models with different attention interfaces from scratch in Table 8. We observe similar performance across different attention interfaces, with the AED model performing slightly better than the prefix LLM and the merged attention model.

In Table 9, we compare the WERs of the different attention interfaces when fine-tuning from the baseline AED encoder and Qwen2 0.5B. In our initial experiments, we were not able to successfully fine-tune an AED model with a pre-trained LLM, which is why we only report results for the prefix LLM and merged attention models. For Librispeech, the performance of the prefix LLM is significantly better than the merged attention model (5.15% vs. 4.81% on dev-other), while for Loquacious, the prefix LLM performs slightly better (6.10% vs. 5.99% on dev).

4.5. Tight Integration vs. Shallow Fusion

We compare tight integration of acoustic encoder and language model in the prefix LLM with shallow fusion of CTC and a stan-

Table 10: *Effect of **tight integration vs. shallow fusion** of acoustic and language model components. Comparing label-synchronous beam search of standalone prefix LLM (PLLM) vs. CTC + LLM shallow fusion. PLLM encoder/ CTC are initialized with the baseline AED encoder and PLLM decoder/ standalone LLM with the Qwen2 0.5B decoder. We fine-tune for 2 epochs on Loquacious.*

Integration	WER [%]	
	dev	test
Tight (PLLM)	6.00	6.99
Shallow Fusion (CTC + LLM)	5.93	6.55

Table 11: *Effect of **bidirectional attention on the decoder prefix** of prefix LLMs (PLLM). All models are initialized with the baseline AED encoder and the Qwen2 0.5B decoder. We fine-tune for 1 epoch on Loquacious using full fine-tuning for the decoder.*

Prefix attention	WER [%]			
	PLLM		PLLM +CTC	
	dev	test	dev	test
Causal	6.07	6.78	5.72	6.27
Bidirectional	6.11	6.64	5.75	6.37

Table 12: *Effect of **different auxiliary losses** for the prefix LLM on Loquacious dev. All models use bidirectional self-attention on the prefix. All models are initialized with the baseline AED encoder. For models with decoder, the decoder is initialized with Qwen2 0.5B. We fine-tune for 1 epoch on Loquacious using full fine-tuning for the decoder.*

Losses			WER [%]				
CE	CTC		Greedy CTC		Prefix LLM		
	Enc.	Dec.	Enc.	Dec.	Standalone	+Enc. CTC	+Dec. CTC
	✓	✓	6.68	6.43	-	-	-
✓	✓		6.59	-	6.11	5.75	-
✓	✓	✓	6.74	6.43	6.72	5.85	5.90

dalone LLM in Table 10. We observe that tight integration outperforms shallow fusion slightly on Loquacious dev (6.00% vs. 5.93%) and significantly on Loquacious test (6.99% vs. 6.55%).

4.6. Effect of Bidirectional Attention in Decoder Prefix

We compare bidirectional to causal self-attention for the decoder prefix of PLLMs in Table 11. We observe that using bidirectional attention has a slight benefit for the standalone performance on Loquacious test (6.78% vs. 6.64%), while the performance with joint CTC decoding is similar.

4.7. Effect of Auxiliary Losses

Similar to using CTC auxiliary losses on the encoder output, we can also apply a CTC loss on the prefix part of the output of the decoder. We compare using different combinations of cross-entropy and CTC losses on the encoder and decoder in Table 12. We observe that greedy decoding of CTC on the decoder output performs better than greedy decoding of CTC on the encoder output (6.74% vs. 6.43%), which is expected since the decoder effectively adds 24 layers on top of the encoder. However, the performance of the standalone PLLM and also PLLM+CTC is worse compared to only using the CTC loss on the encoder output.

Table 13: *Effect of different **decoder fine-tuning strategies** for prefix LLMs (PLLM). All models are initialized with the baseline AED encoder and the Qwen2 0.5B decoder. We fine-tune for 1 epoch on Loquacious.*

Decoder FT	# Trainable params. [B]	WER [%]			
		PLLM		PLLM +CTC	
		dev	test	dev	test
Full	1.1	5.96	6.78	5.72	6.27
LoRA	0.6	6.43	7.03	5.99	6.55

Table 14: *Effect of **different training schedules** for the prefix LLM (PLLM). All models are initialized with the baseline AED encoder and the Qwen2 0.5B decoder. We fine-tune for 1 epoch on Loquacious using LoRA for the decoder. One-stage refers to directly fine-tuning all parameters from the start, while two-stage refers to first training only the newly initialized parameters and then fine-tuning all weights jointly. In both cases, we use LoRA for the decoder fine-tuning.*

Training schedule	WER [%]			
	PLLM		PLLM +CTC	
	dev	test	dev	test
One-stage	6.43	7.03	5.99	6.55
Two-stage	6.76	7.69	5.75	6.27

Table 15: *Effect of **joint CTC recognition** on substitutions (S), deletions (D), insertions (I), and total errors (Σ) for a prefix LLM. All models are initialized with the baseline AED encoder and the Qwen2 0.5B decoder. We fine-tune for 1 epoch on Loquacious using LoRA for the decoder.*

Joint CTC recog.	WER [%]							
	dev				test			
	S	D	I	Σ	S	D	I	Σ
	3.89	1.16	1.39	6.43	4.21	1.24	1.58	7.03
✓	3.87	1.07	1.06	5.99	4.19	1.16	1.20	6.55

put.

4.8. Fine-Tuning Settings

We show the effect of different fine-tuning variants for the decoder in Table 13. When not using joint CTC recognition, full fine-tuning of the decoder performs significantly better than using LoRA (6.43% vs. 5.96% on dev). However, with joint recognition, the LoRA model performs only slightly worse than the fully fine-tuned model (5.99% vs. 5.72% WER on dev) while using only about half of the trainable parameters.

Furthermore, we compare different training schedules in Table 14. We observe that first training only the newly initialized weights and then fine-tuning all trainable parameters jointly has a slight edge compared to directly fine-tuning all parameters from the start.

4.9. Joint CTC Recognition & Hallucinations

We investigate the effect of joint CTC recognition in detail in Table 15. With standalone recognition, the main source of errors are insertions. More specifically, we observe heavy *oscillations* [45], i.e. cases where most of the model output is correct, but with the addition of a repeating n-gram. Often a single or very few sequences account for the majority of the insertions. Joint CTC recognition removes all such oscillations. Furthermore, we never observe such behavior in standalone recognition

Table 16: *Effect of **combining multiple prefix LLM scores** on Loquacious dev. By forwarding the label context with and without the encoder output through the decoder, we can use the prefix LLM both as an acoustic model and as a language model and combine the scores. All models are initialized with the baseline AED encoder and the Qwen2 LLM of the respective size as the decoder. We fine-tune for 1 epoch on Loquacious using LoRA for the decoder. Results using label-synchronous beam search with different combinations of prefix LLM (PLLM), CTC, and LLM scores.*

LLM size [B]	WER [%]			
	PLLM	PLLM +CTC	LLM +CTC	PLLM +CTC + LLM
7	5.70	5.73	5.93	5.41
1.5	5.98	5.88	5.90	5.64

Table 17: *Effect of **different label units** for prefix LLMs (PLLM) on Loquacious dev. All models are initialized with the baseline AED encoder and the Qwen2 0.5B decoder. We fine-tune for 1 epoch on Loquacious using full fine-tuning for the decoder. Comparing greedy CTC, PLLM beam search, and joint CTC+PLLM decoding.*

# Label units	WER [%]		
	CTC	PLLM	PLLM +CTC
150K BPE	6.87	6.01	5.77
10K SPM	5.99	5.79	5.51

of fully fine-tuned models.

4.10. Combining Multiple Prefix LLM Scores

PLLMs can be used in two separate modes. If we feed the output of the encoder into the decoder, they act as an acoustic model. On the other hand, if we don't feed the encoder output into the decoder, they act as a standalone language model. Furthermore, we can also use them in a combined way, i.e. by forwarding the label context with and without the encoder output through the decoder and then combining the scores. We investigate the effect of using PLLMs in such ways in Table 16. Interestingly, for the 7B model, we observe that the standalone PLLM performs on-par or better than PLLM+CTC and LLM+CTC combinations, but that combining all three scores leads to a significant improvement (5.70% vs. 5.41% WER on dev).

4.11. Effect of Label Units

We compare the effect of using different label units in Table 17. To initialize the embeddings of the base LLM with the SPM vocabulary, we use the same method as described in [6]. Interestingly, we observe that using a smaller SPM vocabulary of 10K units leads to better performance than using the original Qwen vocabulary of 150K units (5.77% vs. 5.51%). Especially the auxiliary CTC model performs much better with the smaller SPM vocabulary (6.87% vs. 5.99%). We hypothesize that this is because the encoder was pre-trained using the smaller SPM vocabulary, and that longer training would be needed to adapt the encoder to the different Qwen vocabulary.

4.12. Downsampling

We compare CTC compression to static downsampling for the adapter in Table 18. As expected, memory usage, training time, and recognition time are reduced with higher downsampling factors. Furthermore, the performance of the standalone PLLM degrades with higher downsampling factors, while the performance with joint CTC recognition remains relatively stable.

Table 18: *Effect of **different downsampling strategies in the adapter** of prefix LLMs (PLLM) on Loquacious dev. All models are initialized with the baseline AED encoder and the Qwen2 0.5B decoder. We fine-tune for 1 epoch on Loquacious using LoRA for the decoder. For CTC compression, we use a threshold of 0.9 and reduce frames using their mean. Memory usage, RTFs, and WERs [%] on Loquacious dev. Memory usage and RTF are shown for joint CTC recognition.*

Downsampling		Max. GPU mem. [GB]		Time		WER [%]	
Method	Factor	Train	Recog	Train [h]	Rec. RTF [10^{-3}]	PLLM	PLLM +CTC
Concat frames	1	89.1	9.4	32.1	8.4	6.18	5.99
	2	90.2	9.3	29.7	8.2	6.43	5.99
	3	81.4	9.3	28.7	8.4	6.26	6.02
	5	79.3	9.3	28.2	8.3	6.48	6.03
	10	78.4	9.3	28.3	8.1	7.09	6.04
	20	77.8	9.2	28.1	8.0	8.67	6.11
CTC comp.	Dynamic (avg. 3.5)	84.7	9.4	29.3	8.3	6.23	5.95

Table 19: *Effect of **LLM size** for prefix LLMs (PLLM). All models are initialized with the baseline AED encoder and the corresponding Qwen2 model for the decoder. We fine-tune for 1 epoch on Loquacious using LoRA for the decoder.*

Pre-trained LLM size [B]	WER [%]			
	PLLM		PLLM +CTC	
	dev	test	dev	test
0.5	6.43	7.03	5.99	6.55
1.5	5.98	6.75	5.88	6.39
7.0	5.70	6.46	5.73	6.32

Table 20: *Effect of using **Qwen2 vs. Qwen3** as base model for prefix LLMs (PLLM). For these models, the biggest difference is the amount of pre-training data - 12T tokens for Qwen2 0.5B vs. 36T tokens for Qwen3 0.6B. All models are initialized with the baseline AED encoder. We fine-tune for 1 epoch on Loquacious using LoRA for the decoder.*

Base Model	WER [%]			
	PLLM		PLLM +CTC	
	dev	test	dev	test
Qwen2 0.5B	6.43	7.03	5.99	6.55
Qwen3 0.6B	6.18	6.76	5.96	6.46

4.13. LLM Size

We compare using different LLM sizes for the decoder of PLLMs in Table 19. We observe that larger LLMs lead to better performance (5.70% vs 6.43% WER on dev), especially for standalone PLLM decoding. When using the 7B model, standalone PLLM decoding is on-par with joint CTC+PLLM decoding.

4.14. Amount of Pre-Training Data

We compare using Qwen2 vs. Qwen3 base models as the pre-trained LLM for PLLMs in Table 20. For these models, the biggest difference is the amount of pre-training data - 12T to-

Table 21: *Effect of using **length normalization** for recognition with prefix LLMs (PLLM) on Loquacious dev. All models are initialized with the baseline AED encoder and the Qwen2 0.5B decoder. We fine-tune for 1 epoch on Loquacious using full fine-tuning for the decoder. Results with standalone PLLM.*

Beam Size	WER [%]	
	w/ length norm.	w/o length norm.
1	6.14	
4	6.07	6.06
16	6.24	6.12
32	7.14	6.16

Table 22: *Effect of **beam size** for recognition with prefix LLMs (PLLM) on Loquacious dev. All models are initialized with the baseline AED encoder and the Qwen2 0.5B decoder. We fine-tune for 1 epoch on Loquacious using full fine-tuning for the decoder. All results are with length normalization.*

Beam Size	WER [%]	
	PLLM	PLLM +CTC
1	6.14	5.87
4	6.07	5.72
16	6.24	5.70
32	7.14	5.69

Table 23: *Effect of **optimizations for joint CTC recognition** with prefix LLMs. All models are initialized with the baseline AED encoder and the Qwen2 0.5B decoder. We fine-tune for 1 epoch on Loquacious using LoRA for the decoder. WER, real time factor (RTF) and maximum GPU memory usage when decoding Loquacious dev using a single NVIDIA H100 GPU.*

Top-k pruning	Compression threshold	Max. GPU mem. [GB]	RTF [10^{-3}]	WER [%]
No optimization				
None	None	68.8	69.0	6.02
Only top-k pruning				
30,000	None	18.9	16.2	6.02
10,000		10.7	11.5	6.01
100		7.2	10.9	6.05
Only CTC compression				
None	0.95	63.1	29.7	6.02
	0.90	63.1	29.3	6.03
	0.85	57.3	28.9	6.04
	0.80	47.8	28.6	6.03
Both optimizations				
10,000	0.90	10.3	6.7	6.02
100		7.3	6.6	6.06

kens for Qwen2 0.5B vs. 36T tokens for Qwen3 0.6B. As a standalone PLLM, the Qwen3-based model performs significantly better than the Qwen2-based model (6.43% vs. 6.18% on dev). For PLLM+CTC decoding, the difference is less pronounced, but the Qwen3-based model still performs slightly better (6.55% vs. 6.46% on test).

4.15. Prefix LLM Recognition Settings

We investigate the effect of length normalization and beam size on standalone recognition with PLLMs in Table 21. To our surprise, we observe that, for higher beam sizes, length normalization leads to worse results. This is in contradiction to our experience with AED models, where length normalization is crucial for good performance with higher beam sizes. Unfortunately, using length normalization is turned on by default in all our experiments.

In Table 22 we compare the effect of beam size for joint CTC+PLLM recognition with PLLMs. Here, we observe that performance stagnates after a beam size of 4.

We compare different options for optimizing the joint CTC recognition in Table 23. Both top-k pruning and CTC compression reduce the maximum GPU memory usage and the real time factor (RTF) for recognition. With top-k pruning alone we can reduce the maximum GPU memory usage from 68.8GB to 7.2GB and the RTF from $69.0 \cdot 10^{-3}$ to $10.9 \cdot 10^{-3}$, while almost maintaining the same WER (6.05% vs. 6.02%) compared to not using any optimizations. Additionally applying CTC compression further reduces the RTF to $6.6 \cdot 10^{-3}$.

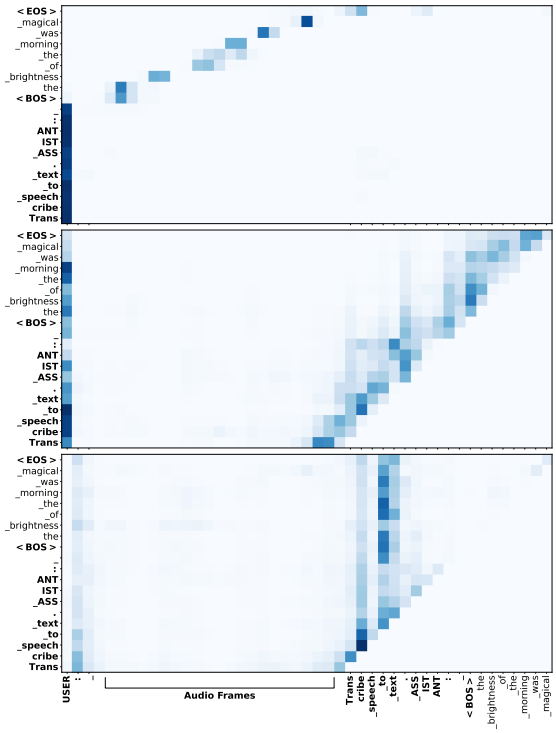

Figure 1: *Self-attention weights of prefix LLM. Model is initialized with the baseline AED encoder and the Qwen2 0.5B decoder and fine-tuned for 1 epoch on Loquacious. From bottom to top: layer 24 head 7, layer 5 head 8, layer 13 head 13. The X-axis shows the key/value positions for corresponding inputs, and the Y-axis shows the query positions for the decoder output (omitting part of the sequence for better visibility).*

4.16. Visualization of Attention Weights

The self-attention weights of a selected prefix LLM based on Qwen2 0.5B are shown in Figure 1. We manually went through the attention weights of all heads and layers and identified three heads with interesting attention patterns. The selected heads show the attention to the encoder output for the transcription labels. We identify different patterns of attention weights. The topmost plot (layer 13; head 13) shows a head that attends to the encoder output in a monotonic way, similar to the attention patterns of AED models. The next plot (layer 5; head 8) shows an n-gram pattern, where the attention is focused on a small window before the current query position, which has also been observed for attention heads in text-based Transformer LMs [46].

Table 24: *Effect of **text prompt** for prefix LLMs (PLLM) on Loquacious dev. All models are initialized with the baseline AED encoder and the Qwen2 0.5B decoder. We fine-tune for 1 epoch on Loquacious using LoRA for the decoder.*

Prompt	WER [%]	
	PLLM	PLLM +CTC
USER: ⟨audio⟩ Transcribe this speech to text. ASSISTANT: ⟨BOS⟩	6.42	5.99
⟨audio⟩⟨BOS⟩	6.51	6.01

Table 25: *Results on the **HuggingFace ASR leaderboard**. We use the official text normalization on the model output and reference transcript for evaluation. Our models are trained on Loquacious. The baseline AED model is trained from scratch for 20 epochs, while the prefix LLM is initialized with the baseline AED encoder and the Qwen2 0.5B decoder and fine-tuned for 1 epoch on Loquacious using LoRA for the decoder. Comparing to other models from the literature and leaderboard.*

System	WER [%]								
	AMI	Earnings22	Giga speech	LBS Clean	LBS Other	SPGI Speech	Ted lium	Vox populi	Avg.
Others									
Canary-Qwen-2.5B (Qwen3 1.7B)	10.19	10.45	9.43	1.61	3.10	1.90	2.71	5.66	5.63
Qwen3-ASR-0.6B (Qwen3 0.6B)	11.66	11.06	9.14	2.13	4.45	3.03	2.85	7.07	6.42
Ours									
CTC + PLLM (Qwen2 0.5B)	19.39	29.87	11.39	1.91	3.72	4.47	4.05	6.45	11.21
CTC + AED	20.88	34.59	13.16	1.84	3.75	5.53	4.28	6.19	12.49

The last plot (layer 24; head 7) shows a head which is focused on the prompt.

4.17. Effect of Prompt Length

[47] observed that LLMs can learn to utilize meaningless filler tokens such as a sequence of dots to improve their performance. Since the prompt is always the same in our case, it does not provide any useful information for the model and can also be considered as a sequence of meaningless filler tokens. Motivated by the attention pattern of the last head in Figure 1, which is focused on the prompt, we test whether our PLLMs benefit from the long prompt by comparing our baseline prompt to a prompt consisting only of a BOS token in Table 24. For standalone PLLM decoding, the model benefits slightly from the long prompt (6.42% vs. 6.51% WER on dev), while for joint CTC+PLLM decoding, the performance is on par with the shorter prompt.

4.18. Final Results

We compare our best results to other models from the literature on the HuggingFace ASR leaderboard in Table 25. We use the official text normalization[1] on the model output and reference transcript for evaluation. We observe that, while our prefix LLM-based model performs worse than our AED baseline on Loquacious and Librispeech (Tables 4, 8 and 9), it outperforms the AED baseline on the HuggingFace ASR leaderboard. We attribute this to the fact that most of the corpora in the HuggingFace ASR leaderboard are out-of-domain for our baseline ASR models, while the LLM-based models can leverage the pre-training knowledge to generalize better to these domains. We note that the other models on the leaderboard are trained on magnitudes more ASR data than our models, which makes the comparison difficult. Still, except for the AMI and Earnings corpora, the performance of our prefix LLM lies in a similar range as the other models.

[1] https://github.com/huggingface/open_asr_leaderboard/blob/main/normalizer/normalizer.py

5. Conclusions & Outlook

In this work, we have investigated different utilizations of large language models (LLMs) for automatic speech recognition (ASR). We compared tight integration of acoustic encoder and LLMs, *speech LLMs* (SLLMs), to shallow fusion of acoustic and language model scores. Our findings are as follows:

- Prefix LLMs (PLLMs) outperform the merged attention interface in our experiments when starting from a pre-trained LLM decoder while performing similarly to the merged attention interface when training from scratch. The reason for this could be that the prefix LLM interface is more similar to how the LLM was pre-trained, and that longer fine-tuning is required for the merged attention interface to adapt to the new mode.
- Joint CTC+SSLM decoding can significantly improve the performance of standalone SLLM decoding, especially for smaller LLMs. We presented two optimizations for joint CTC+SLLM decoding without which the computational cost of joint decoding becomes prohibitive, especially for larger LLMs. When moving to larger LLMs, training longer, or using full fine-tuning instead of LoRA for the decoder, the performance of standalone SLLM decoding and joint decoding converges.
- PLLMs can be used as both acoustic and language model, depending on whether the encoder output is used or not. We find that combining these scores together with CTC scores yields significant improvements over CTC+PLLM decoding with negligible additional computational cost.
- Using a smaller vocab, which was fitted on the transcription data, improves performance compared to using the original vocab of the LLM.
- Using the same data conditions and model sizes, shallow fusion of LLM and CTC scores yield better performance than tightly integrating acoustic encoder and LLM into a single model. This is in opposition to the current trend of using large monolithic models for ASR. However, it is not clear whether our findings would also hold when using larger LLMs and more training data.
- In our experiments on Loquacious and Librispeech, the performance of our prefix LLM still lags behind our AED baseline which was solely trained on the ASR data. However, on the HuggingFace ASR leaderboard, our prefix LLM outperforms our AED baseline. From this, we conclude that LLMs are especially beneficial for improving the generalization of ASR models to different domains. However, for completeness, we also note the issue of test set leakage of public corpora into the pre-training data of LLMs, which could also contribute to the better performance of LLM-based models.

For future work, we want to investigate the effect of moving from text-based LLMs to native multi-modal models which were pre-trained on both text and audio data. Furthermore, we want to compare the ASR convergence of pre-trained LLMs to training models from scratch.

6. Generative AI Use Disclosure

We use LLMs to improve the formulations and grammar of the paper.

7. References

[1] C. Gulcehre, O. Firat, K. Xu, K. Cho, L. Barrault, H.-C. Lin, F. Bougares, H. Schwenk, and Y. Bengio, "On using monolingual corpora in neural machine translation," Preprint arXiv:1503.03535, 2015.

[2] A. Sriram, H. Jun, S. Satheesh, and A. Coates, "Cold Fusion: Training Seq2Seq Models Together with Language Models," in *Interspeech*, 2018, pp. 387–391.

[3] M. Zeineldeen, A. Glushko, W. Michel, A. Zeyer, R. Schlüter, and H. Ney, "Investigating methods to improve language model integration for attention-based encoder-decoder ASR models," in *Interspeech*, Aug. 2021, pp. 2856–2860. [Online]. Available: http://https://arxiv.org/abs/2104.05544

[4] K. Veselý, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks." in *Interspeech*, 2013, pp. 2345–2349.

[5] T. Hori, M. Kocour, A. Haider, E. McDermott, and X. Zhuang, "Delayed fusion: Integrating large language models into first-pass decoding in end-to-end speech recognition," in *IEEE ICASSP*, 2025, pp. 1–5.

[6] 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," in *IEEE ICASSP*, 2025, pp. 1–5.

[7] P. K. Rubenstein, C. Asawaroengchai, D. D. Nguyen, A. Bapna, Z. Borsos, F. de Chaumont Quitry, P. Chen, D. E. Badawy, W. Han, E. Kharitonov, H. Muckenhirn, D. Padfield, J. Qin, D. Rozenberg, T. Sainath, J. Schalkwyk, M. Sharifi, M. T. Ramanovich, M. Tagliasacchi, A. Tudor, M. Velimirović, D. Vincent, J. Yu, Y. Wang, V. Zayats, N. Zeghidour, Y. Zhang, Z. Zhang, L. Zilka, and C. Frank, "Audiopalm: A large language model that can speak and listen," 2023. [Online]. Available: https://arxiv.org/abs/2306.12925

[8] D. Zhang, S. Li, X. Zhang, J. Zhan, P. Wang, Y. Zhou, and X. Qiu, "Speechgpt: Empowering large language models with intrinsic cross-modal conversational abilities," 2023. [Online]. Available: https://arxiv.org/abs/2305.11000

[9] C. Tang, W. Yu, G. Sun, X. Chen, T. Tan, W. Li, L. Lu, Z. MA, and C. Zhang, "SALMONN: Towards generic hearing abilities for large language models," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=14rn7HpKVk

[10] Y. Bai, J. Chen, J. Chen, W. Chen, Z. Chen, C. Ding, L. Dong *et al.*, "Seed-asr: Understanding diverse speech and contexts with llm-based speech recognition," 2024. [Online]. Available: https://arxiv.org/abs/2407.04675

[11] K.-T. Xu, F.-L. Xie, X. Tang, and Y. Hu, "Fireredasr: Open-source industrial-grade mandarin speech recognition models from encoder-decoder to llm integration," 2025. [Online]. Available: https://arxiv.org/abs/2501.14350

[12] Microsoft, A. Abouelenin, A. Ashfaq, A. Atkinson, H. Awadalla *et al.*, "Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras," 2025. [Online]. Available: https://arxiv.org/abs/2503.01743

[13] G. Saon, A. Dekel, A. Brooks, T. Nagano, A. Daniels, A. Satt, A. Mittal, B. Kingsbury, D. Haws, E. Morais, G. Kurata, H. Aronowitz, I. Ibrahim, J. Kuo, K. Soule, L. Lastras, M. Suzuki, R. Hoory, S. Thomas, S. Novitasari, T. Fukuda, V. Sunder, X. Cui, and Z. Kons, "Granite-speech: open-source speech-aware LLMs with strong English ASR capabilities," 2025. [Online]. Available: https://arxiv.org/abs/2505.08699

[14] X. Shi, X. Wang, Z. Guo, Y. Wang, P. Zhang, X. Zhang *et al.*, "Qwen3-ASR technical report," 2026. [Online]. Available: https://arxiv.org/abs/2601.21337

[15] A. Gupta, G. Saon, and B. Kingsbury, "Exploring the limits of decoder-only models trained on public speech recognition corpora," in *Interspeech*, 2024, pp. 252–256.

[16] T. Wang, A. Roberts, D. Hesslow, T. L. Scao, H. W. Chung, I. Beltagy, J. Launay, and C. Raffel, "What language model architecture and pretraining objective works best for zero-shot generalization?" in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 17–23 Jul 2022, pp. 22 964–22 984. [Online]. Available: https://proceedings.mlr.press/v162/wang22u.html

[17] 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," in *IEEE ICASSP*. IEEE, 2025, pp. 1–5.

[18] Z. Kong, A. Goel, R. Badlani, W. Ping, R. Valle, and B. Catanzaro, "Audio flamingo: a novel audio language model with few-shot learning and dialogue abilities," in *Proceedings of the 41st International Conference on Machine Learning*, ser. ICML. JMLR.org, 2024.

[19] W. Yu, C. Tang, G. Sun, X. Chen, T. Tan, W. Li, L. Lu, Z. Ma, and C. Zhang, "Connecting speech encoder and large language model for asr," 2023. [Online]. Available: https://arxiv.org/abs/2309.13963

[20] G. Team, R. Anil, S. Borgeaud, J.-B. Alayrac *et al.*, "Gemini: A family of highly capable multimodal models," 2025. [Online]. Available: https://arxiv.org/abs/2312.11805

[21] J. Xu, Z. Guo, H. Hu, Y. Chu, X. Wang, J. He *et al.*, "Qwen3-omni technical report," 2025. [Online]. Available: https://arxiv.org/abs/2509.17765

[22] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," in *Interspeech*, 2020.

[23] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.

[24] R. Prabhavalkar, T. Hori, T. N. Sainath, R. Schlüter, and S. Watanabe, "End-to-end speech recognition: A survey," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 32, pp. 325–351, 2023.

[25] 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. [Online]. Available: https://arxiv.org/abs/2402.08846

[26] M. Gaido, M. Cettolo, M. Negri, and M. Turchi, "CTC-based compression for direct speech translation," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, P. Merlo, J. Tiedemann, and R. Tsarfaty, Eds. Online: Association for Computational Linguistics, Apr. 2021, pp. 690–696. [Online]. Available: https://aclanthology.org/2021.eacl-main.57/

[27] 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 *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2023, pp. 1–8.

[28] W. Zhou, J. Jia, L. Sari, J. Mahadeokar, and O. Kalinli, "Cjst: Ctc compressor based joint speech and text training for decoder-only asr," in *IEEE ICASSP*, 2025, pp. 1–5.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017, pp. 6000–6010.

[30] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," Preprint arXiv:1506.07503, 2015. [Online]. Available: http://arxiv.org/abs/1506.07503

[31] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *IEEE ICASSP*, 2016.

[32] B. Zhang, P. Suganthan, G. Liu, I. Philippov, S. Dua, B. Hora, K. Black, G. Martins, O. Sanseviero, S. Pathak, C. Hardin, F. Visin, J. Zhang, K. Kenealy, Q. Yin, X. Song, O. Lacombe, A. Joulin, T. Warkentin, and A. Roberts, "T5gemma 2: Seeing, reading, and understanding longer," 2025. [Online]. Available: https://arxiv.org/abs/2512.14856

[33] T. Hori, S. Watanabe, Y. Zhang, and W. Chan, "Advances in joint CTC-attention based end-to-end speech recognition with a deep CNN encoder and RNN-LM," in *Interspeech*, 2017.

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

[35] T. Parcollet, Y. Tseng, S. Zhang, and R. C. van Dalen, "Loquacious Set: 25,000 hours of transcribed and diverse English speech recognition data for research and commercial use," in *Interspeech 2025*, 2025, pp. 4053–4057.

[36] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Interspeech*, 2019, pp. 2613–2617.

[37] A. Zeyer, T. Alkhouli, and H. Ney, "RETURNN as a generic flexible neural toolkit with application to translation and speech recognition," in *Annual Meeting of the Assoc. for Computational Linguistics*, Melbourne, Australia, Jul. 2018.

[38] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

[39] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. E. Y. Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, "ESPnet: End-to-end speech processing toolkit," 2018. [Online]. Available: https://arxiv.org/abs/1804.00015

[40] A. Yang, B. Yang, B. Hui, B. Zheng, B. Yu *et al.*, "Qwen2 technical report," 2024. [Online]. Available: https://arxiv.org/abs/2407.10671

[41] E. J. Hu, yelong 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*, 2022. [Online]. Available: https://openreview.net/forum?id=nZeVKeeFYf9

[42] D. Kalajdzievski, "A rank stabilization scaling factor for fine-tuning with lora," 2023. [Online]. Available: https://arxiv.org/abs/2312.03732

[43] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, E. Blanco and W. Lu, Eds. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 66–71. [Online]. Available: https://aclanthology.org/D18-2012/

[44] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui *et al.*, "Qwen3 technical report," 2025. [Online]. Available: https://arxiv.org/abs/2505.09388

[45] R. Frieske and B. E. Shi, "Hallucinations in neural automatic speech recognition: Identifying errors and hallucinatory models," 2024. [Online]. Available: https://arxiv.org/abs/2401.01572

[46] K. Irie, A. Zeyer, R. Schlüter, and H. Ney, "Language Modeling with Deep Transformers," in *Interspeech 2019*, 2019, pp. 3905–3909.

[47] J. Pfau, W. Merrill, and S. R. Bowman, "Let's think dot by dot: Hidden computation in transformer language models," in *First Conference on Language Modeling*, 2024. [Online]. Available: https://openreview.net/forum?id=NikbrdtYvG