

LLAMA-OMNI: SEAMLESS SPEECH INTERACTION WITH LARGE LANGUAGE MODELS

**Qingkai Fang^{1,3}, Shoutao Guo^{1,3}, Yan Zhou^{1,3}, Zhengrui Ma^{1,3},
Shaolei Zhang^{1,3}, Yang Feng^{1,2,3*}**

¹ Key Laboratory of Intelligent Information Processing
Institute of Computing Technology, Chinese Academy of Sciences (ICT/CAS)

² Key Laboratory of AI Safety, Chinese Academy of Sciences

³ University of Chinese Academy of Sciences, Beijing, China
{fangqingkai21b, fengyang}@ict.ac.cn

<https://arxiv.org/abs/2409.06666>

Demo

LLaMA-Omni: Seamless Speech Interaction with Large Language Models

Llama-3.1-8B-Omni

Speech Input

将音频拖放到此处
- 或 -
点击上传

Parameters

Temperature	0
Top P	0.7
Max Output Tokens	512
Chunk Size	40

Examples

vicuna_1.wav vicuna_2.wav vicuna_3.wav vicuna_4.wav vicuna_5.wav helpful_base_1.wav helpful_base_2.wav helpful_base_3.wav helpful_base_4.wav helpful_base_5.wav

Send Clear

Text Output

Unit Output

Speech Output

通过 API 使用 · 使用 Gradio 构建

LLaMA-Omni integrates a pretrained speech encoder, a speech adaptor, an LLM, and a streaming speech decoder. It can simultaneously generate text and speech responses directly from speech instructions with extremely low latency.

To align the model with speech interaction scenarios, the authors construct a dataset named **InstructS2S-200K**, which includes 200K speech instructions and corresponding speech responses.

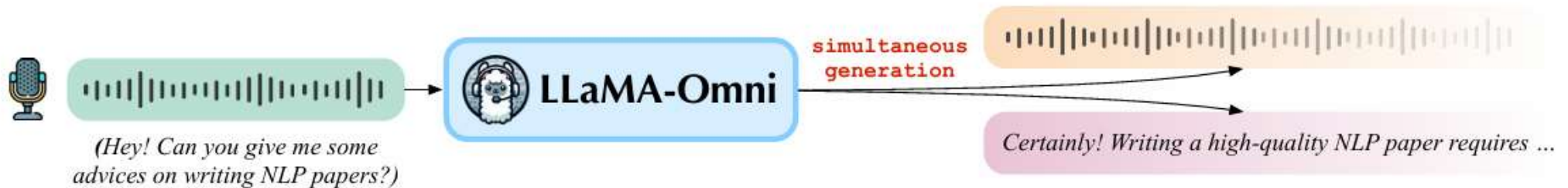
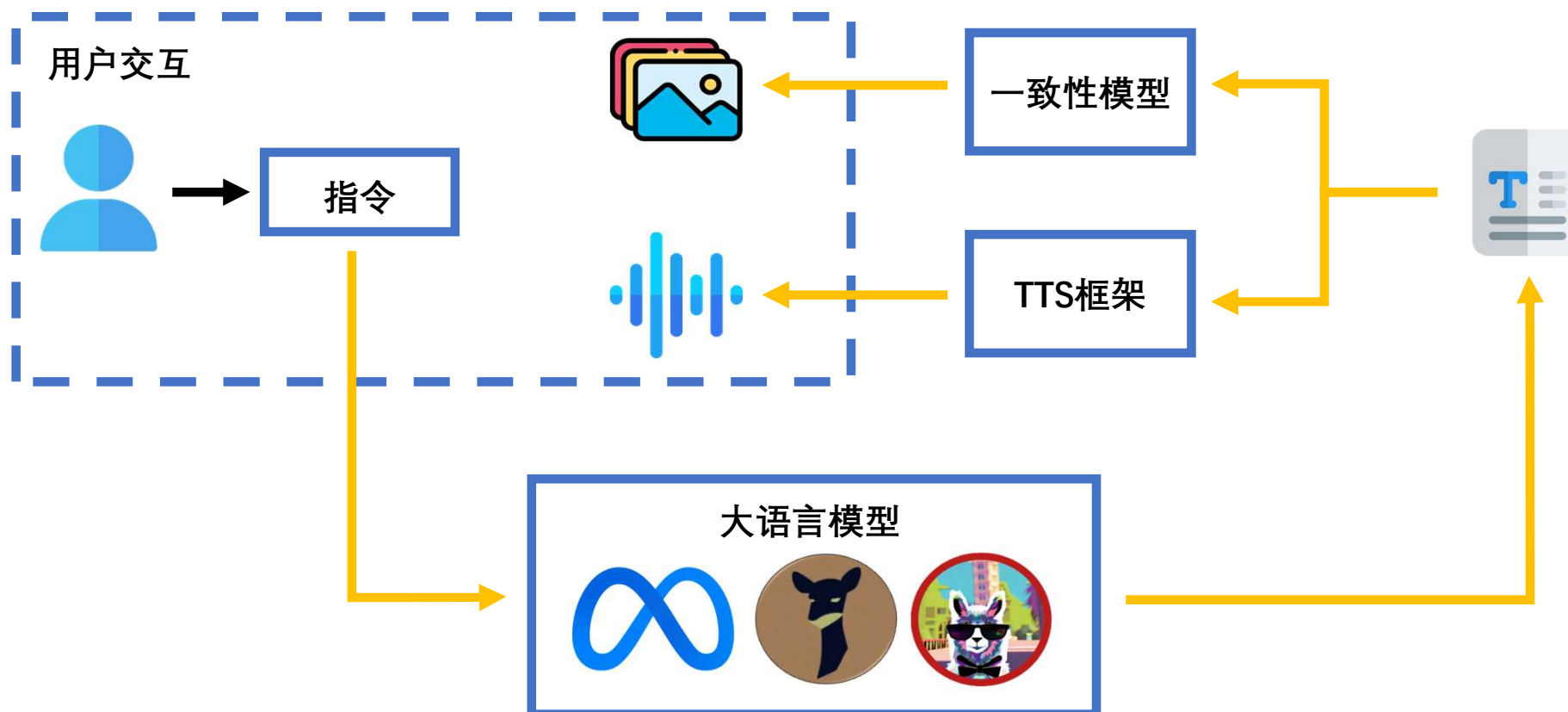


Figure 1: LLaMA-Omni can simultaneously generate text and speech responses based on the speech instruction, with extremely low response latency.

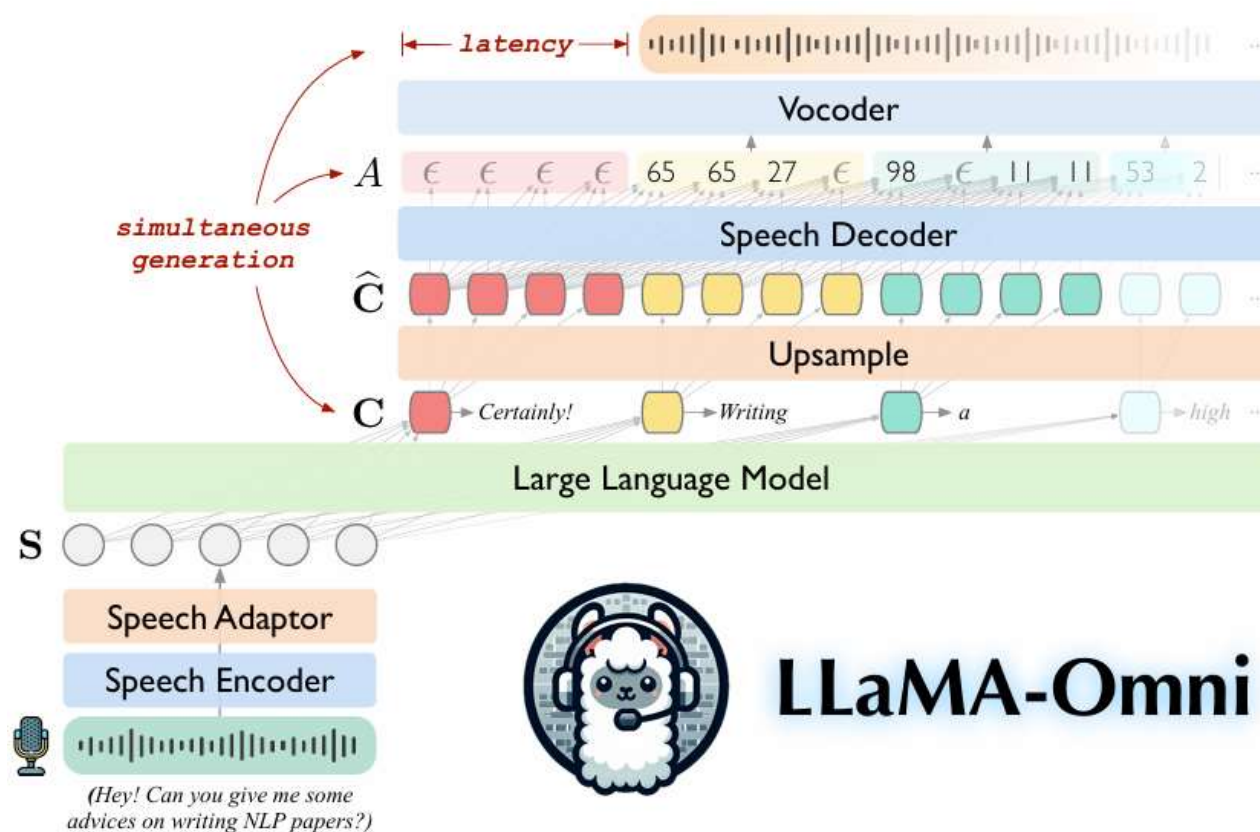
Cascaded System

The simplest way to enable speech interaction with LLM is through a cascaded system based on automatic speech recognition (ASR) and text-to-speech (TTS) models, where the ASR model transcribes the user's speech instruction into text, and the TTS model synthesizes the LLM's response into speech. However, the overall system tends to have higher latency.



LLaMA-Omni

LLaMA-Omni consists of a **speech encoder**, a **speech adaptor**, an **LLM**, and a **speech decoder**. We denote the user's speech instruction, text response, and speech response as X^S , Y^T , and Y^S respectively.



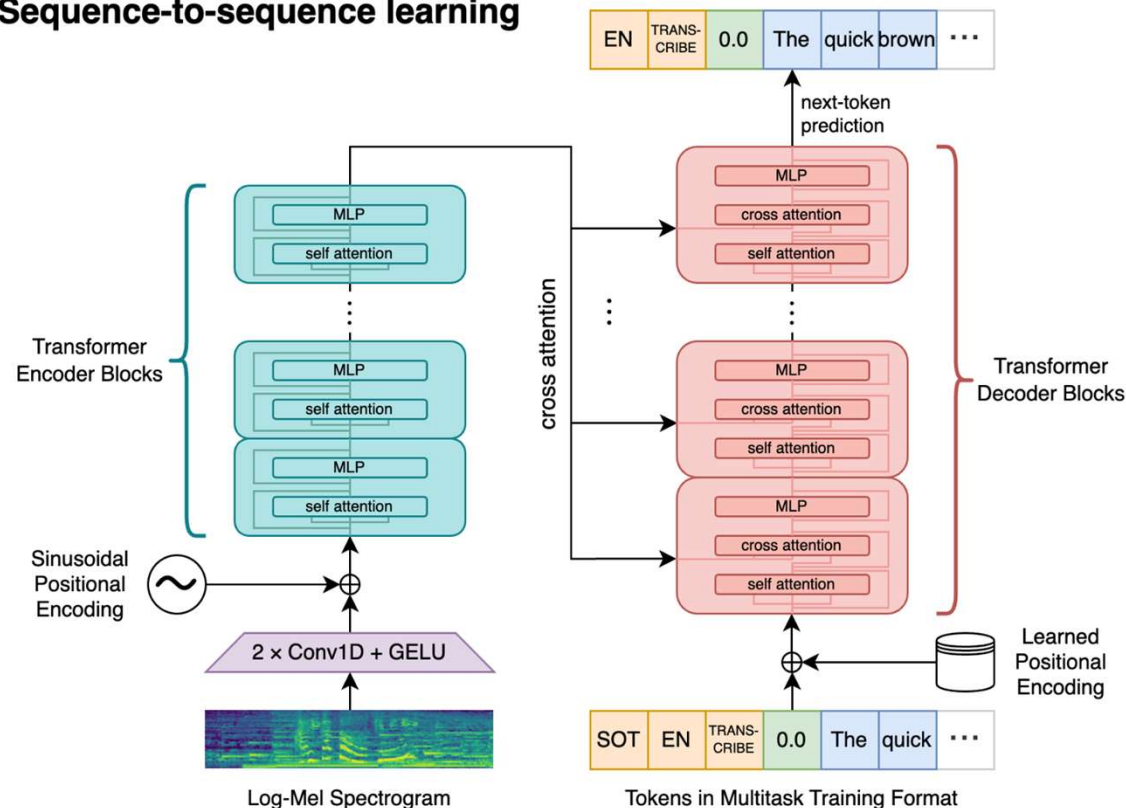
LLaMA-Omni

Speech Encoder

The authors use the encoder of **Whisper-large-v3** as the speech encoder ε . Whisper is a general-purpose speech recognition model trained on a large amount of audio data, and its encoder is capable of extracting meaningful representations from speech. They keep the speech encoder's parameters frozen throughout the entire training process.

$H = \varepsilon(X^S)$, where $H = [h_1, \dots, h_N]$ is the speech representation sequence of length N .

Sequence-to-sequence learning



To enable the LLM to comprehend the input speech, the authors incorporate a trainable speech adaptor A that maps the speech representations into the embedding space of the LLM. The speech adaptor first downsamples the speech representations H to reduce the sequence length. Every k consecutive frames are concatenated along the feature dimension:

$$H' = [h'_1, \dots, h'_{\lfloor N/k \rfloor}], \text{ where } h'_i = [h_{k \times (i-1) + 1} \oplus h_{k \times (i-1) + 2} \oplus \dots \oplus h_{k \times i}]$$

Next, H' is passed through a 2-layer perceptron with ReLU activation between the linear layers, resulting in the final speech representations S .

$$S = A(H) = \text{Linear}(\text{ReLU}(\text{Linear}(\text{DownSample}(H))))$$

Large Language Model

The authors use **Llama-3.1-8B-Instruct** as the LLM M , which is currently the state-of-the-art open-source LLM. The prompt template $P(\cdot)$ is shown in Figure 3. The speech representation sequence S is filled into the position corresponding to **<speech>**, and then the entire sequence $P(S)$ is input into the LLM. The LLM generates the text response $Y^T = [y_1^T, \dots, y_M^T]$ directly based on the speech instruction.

$$L_{LLM} = - \sum_{i=1}^M \log P(y_i^T | P(S), Y_{<i}^T)$$

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>
```

```
You are a helpful language and speech assistant. You are able to understand the speech content that the user provides, and assist the user with a variety of tasks using natural language.<|eot_id|>
```

```
<|start_header_id|>user<|end_header_id|>
```

```
<speech>
```

```
Please answer the questions in the user's input speech.<|eot_id|>
```

```
<|start_header_id|>assistant<|end_header_id|>
```

Figure 3: Prompt template of LLaMA-Omni.

Speech Decoder

For the speech response Y^S , the authors first use the pretrained **HuBERT** model to extract representations of the speech, and then convert these representations into discrete cluster indices using a K-means model. Consecutive identical indices are merged into a single unit, resulting in $Y^U = [y_1^U, \dots, y_L^U]$, $y_i^U \in \{0, 1, \dots, K - 1\}$, $\forall 1 \ll i \ll L$, where K is the number of clusters, and L is the length of discrete unit sequence. The discrete units can be converted into waveform with an additional unit-based vocoder V .

To generate speech and text simultaneously, the speech decoder D takes the output hidden states corresponding to the text from the LLM.

$$C = [c_1, \dots, c_M], \text{ where } c_i = M(P(S), Y_{<i}^T)$$

Then upsample them into a chunk by a factor of λ .

$$\hat{C} = [\hat{c}_1, \dots, \hat{c}_{\lambda \cdot M}], \text{ where } \hat{c}_i = c_{\lfloor i/\lambda \rfloor}$$

Next \hat{C} is fed into the speech decoder D , resulting in $O = [o_1, \dots, o_{\lambda \cdot M}]$. Use **connectionist temporal classification (CTC)** to align O with Y^U . CTC extends the output space with a special blank token ϵ :

$$P(a_i | O) = \text{softmax}(W o_i + b)[a_i], \forall a_i \in \{0, 1, \dots, K - 1, \epsilon\}$$

where $W \in R^{(K+1) \times d}$ and $b \in R^{K+1}$ are weights and biases of the linear layer, and the sequence $A = [a_1, \dots, a_{\lambda \cdot M}]$ is known as the *alignment*. CTC introduces a collapsing function $\beta(A)$, which merges all consecutive repeated tokens in A and eliminates all blank token ϵ . $\beta([1, 1, 2, \epsilon, \epsilon, 2, 3]) = [1, 2, 2, 3]$

$$L_{CTC} = -\log P(Y^U | O) = -\log \sum_{A \in \beta^{-1}(Y^U)} P(A | O) = -\log \sum_{A \in \beta^{-1}(Y^U)} \prod_{i=1}^{\lambda \cdot M} P(a_i | O)$$

Where $\beta^{-1}(Y^U)$ denotes all possible alignments of length $\lambda \cdot M$ that can be collapsed to Y^U .

Training

In the first stage, train the model to generate text responses directly from the speech instructions. The speech encoder is frozen, and the speech adaptor and the LLM are trained using the objective L_{LLM} .

In the second stage, train the model to generate speech responses. The speech encoder, speech adaptor, and LLM are all frozen, and only the speech decoder is trained using the objective L_{CTC} .



Once the LLM generates a text response prefix $Y_{\leq i}^T$, the corresponding upsampled hidden states $\hat{C}_{\leq \lambda \cdot i}$ can be fed into the speech decoder to generate a partial alignment $A_{\leq \lambda \cdot i}$, which in turn yields the discrete units corresponding to the generated text prefix.

To enable streaming synthesis of speech waveforms, when the number of generated units reaches a pre-defined chunk size Ω , input this unit segment into the vocoder to synthesize a speech segment.

Algorithm 1: Inference Process

Input: speech instruction X^S .
Output: text outputs Y^T , units outputs Y^U , waveform outputs Y^S .
Model: speech encoder \mathcal{E} , speech adaptor \mathcal{A} , LLM \mathcal{M} , speech decoder \mathcal{D} , vocoder \mathcal{V} .
Require: Minimum chunk size for units Ω .
Initialization: $i = 1, j = 0, Y^T = \square, Y^U = \square, Y^S = \square, \hat{C} = \square$.
while $y_{i-1}^T \neq \langle \text{EOS} \rangle$ **do**
 $\mathbf{S} \leftarrow \mathcal{A}(\mathcal{E}(X^S));$
 $\mathbf{c}_i \leftarrow \mathcal{M}(\mathcal{P}(\mathbf{S}), Y_{<i}^T);$
 $y_i^T \leftarrow \arg \max_{y_i^T} P(y_i^T | \mathcal{P}(\mathbf{S}), Y_{<i}^T);$
 $Y^T \leftarrow Y^T + y_i^T;$
 $\hat{C} \leftarrow \hat{C} + \text{UpSample}(\mathbf{c}_i);$
 $\mathbf{O} \leftarrow \mathcal{D}(\hat{C});$
 $A^* \leftarrow \arg \max_A P(A | \mathbf{O});$
 $Y^U \leftarrow \beta(A^*);$
 if $|Y^U| - j \geq \Omega$ **then**
 $y^S \leftarrow \mathcal{V}(Y_{j+1}^U);$
 $Y^S \leftarrow Y^S + y^S;$
 $j \leftarrow |Y^U|;$
 end
 $i \leftarrow i + 1;$
end
if $j < |Y^U|$ **then**
 $y^S \leftarrow \mathcal{V}(Y_{j+1}^U);$
 $Y^S \leftarrow Y^S + y^S;$
end

Data Construction

Data format: <speech instruction, text response, speech response>. However, most publicly available instruction data is in text form.

Step 1: Instruction Rewriting. Rewrite the text instructions according to the following rules: (1) Add appropriate filler words (such as “hey”, “so”, “uh”, “um”, etc.) to the instructions to simulate natural speech patterns. (2) Convert non-text symbols in the instructions (such as numbers) into their corresponding spoken forms to ensure synthesis by TTS. (3) Modify the instructions to be relatively brief without excessive verbiage.

Prompt for Instruction Rewriting (Model: Llama-3-70B-Instruct)

Below is an instruction data containing the user’s instruction. I would like to generate a speech version of this instruction for training a large language model that supports speech input. Therefore, please rewrite my instruction data according to the following requirements:

1. Modify the instruction to simulate human speech, adding fillers as appropriate (but not too many ‘you know’, ‘like’, etc.).
2. The question should not contain content that cannot be synthesized by the TTS model. Numbers should be written in English words rather than Arabic numerals.
3. The question should be relatively brief without excessive verbiage.

[instruction]: **{instruction}**

Please output in JSON format as follows: {“question”: {question}}.

Data Construction

Data format: <speech instruction, text response, speech response>. However, existing responses from text instructions are not suitable for direct use as speech instructions responses.

Step 2: Response Generation. Generate responses for speech instructions according to the following rules: (1) The response should not contain content that cannot be synthesized by the TTS model, such as parentheses, ordered lists, etc. (2) The response should be very concise and to the point, avoiding lengthy explanations.

Prompt for Response Generation (Model: Llama-3-70B-Instruct)

Below is the transcribed text of a user's speech query. Please provide a response to this question, which will be converted to speech using TTS. Please follow these requirements for your response:

1. Your response should not contain content that cannot be synthesized by the TTS model, such as parentheses, ordered lists, etc. Numbers should be written in English words rather than Arabic numerals.
2. Your response should be very concise and to the point, avoiding lengthy explanations.

[instruction]: **{instruction}**

Please output in JSON format as follows: {"response": {response}}.

Data Construction

Data format: <speech instruction, text response, speech response>.

Step 3: Speech Synthesis. For the instructions, use [CosyVoice-300M-SFT](#) model to synthesize the speech naturally. For the responses, use the [VITS](#) model trained on the LJSpeech dataset to synthesize the speech.

For the basic text instructions, the authors collect around 50K instructions from the [Alpaca](#) dataset and 150K instructions from the [UltraChat](#) dataset. Note that UltraChat is a large-scale multi-turn conversation dataset, but only select the first 150K entries and use only the first-round instruction.

Ultimately obtain 200K speech instructions data, referred to as **InstructS2S-200K**.

Datasets For the training data, we use the **InstructS2S-200K** dataset mentioned in Section 3, which includes 200K speech instruction data. To extract discrete units corresponding to the target speech, we use a pre-trained K-means quantizer⁹, which has learned 1000 clusters from the HuBERT features. The pretrained HiFi-GAN vocoder (Kong et al., 2020; Polyak et al., 2021) is used to synthesize discrete units into waveform. For the evaluation data, we select two subsets from Alpaca-Eval¹⁰ (Li et al., 2023): *helpful_base* and *vicuna*, as their questions are more suitable for speech interaction scenarios. We remove questions related to math and code, resulting in a total of 199 instructions. To obtain the speech version, we use the CosyVoice-300M-SFT model to synthesize the instructions into speech. We refer to this test set as **InstructS2S-Eval** in the following sections.

Model Configuration We use the encoder of Whisper-large-v3 as the speech encoder, and use Llama-3.1-8B-Instruct as the LLM. The speech adapter performs a $5\times$ downsampling on the speech representations. The speech decoder consists of 2 Transformer layers with the same architecture as LLaMA, with a hidden dimension of 4096, 32 attention heads, and a feed-forward network dimension of 11008, which contains 425M parameters. The upsample factor λ is set to 25. For the minimum unit chunk size Ω input to the vocoder, we set $\Omega = +\infty$ in the main experiment, meaning we wait for the entire unit sequence to be generated before inputting it to the vocoder for speech synthesis. In subsequent experiments, we will analyze how adjusting the value of Ω can control response latency, as well as the trade-off between latency and speech quality.

Training LLaMA-Omni follows a two-stage training process. In the first stage, we train the speech adapter and the LLM with a batch size of 32 for 3 epochs. We use a cosine learning rate scheduler with the first 3% of steps for warmup, and the peak learning rate is set to $2e-5$. In the second stage, we train the speech decoder, using the same batch size, number of steps, and learning rate scheduler as the first stage, but with the peak learning rate set to $2e-4$. The entire training process takes approximately 65 hours on 4 NVIDIA L40 GPUs.

Two tasks: speech-to-text instruction-following (S2TIF) and speech-to-speech instruction-following (S2SIF).

ChatGPT Score To evaluate the model's ability to follow speech instructions, we use GPT-4o (OpenAI, 2024) to score the model's responses. For the S2TIF task, scoring is based on the transcribed text of the speech instructions and the model's text response. For the S2SIF task, we first transcribe the model's speech responses into text using the Whisper-large-v3 model, and then score it in the same manner as the S2TIF task. GPT-4o gives scores on two aspects: *content* and *style*. The **content score** evaluates whether the model's response adequately addresses the user's instruction, while the **style score** assesses whether the model's response style is suitable for speech interaction scenarios. The detailed prompt can be found in Appendix A.

Speech-Text Alignment To evaluate the alignment between text responses and speech responses, we use the Whisper-large-v3 model to transcribe the speech responses into text, and then calculate the Word Error Rate (WER) and Character Error Rate (CER) between the transcribed text and the text response. We refer to these metrics as **ASR-WER** and **ASR-CER**, respectively.

Speech Quality To evaluate the quality of the generated speech, we utilize a Mean Opinion Score (MOS) prediction model called UTMOS¹¹ (Saeki et al., 2022), which is capable of predicting the MOS score of the speech to assess its naturalness. We refer to this metric as the **UTMOS** score.

Response Latency The **latency** is a key metric for speech interaction models, referring to the time interval between the input of a speech instruction and the start of the speech response, which has a significant impact on user experience. Additionally, we calculate the number of words already generated in the text response when the speech response begins, referred to as the **#lagging word**.

Prompt for ChatGPT Scoring (Model: GPT-4o)

I need your help to evaluate the performance of several models in the speech interaction scenario. The models will receive a speech input from the user, which they need to understand and respond to with a speech output. Your task is to rate the model's responses based on the provided user input transcription [Instruction] and the model's output transcription [Response]. Please evaluate the response from two perspectives: content and style, and provide a score for each on a scale of 1 to 5.

Content (1-5 points):

1 point: The response is largely irrelevant, incorrect, or fails to address the user's query. It may be off-topic or provide incorrect information.

2 points: The response is somewhat relevant but lacks accuracy or completeness. It may only partially answer the user's question or include extraneous information.

3 points: The response is relevant and mostly accurate, but it may lack conciseness or include unnecessary details that don't contribute to the main point.

4 points: The response is relevant, accurate, and concise, providing a clear answer to the user's question without unnecessary elaboration.

5 points: The response is exceptionally relevant, accurate, and to the point. It directly addresses the user's query in a highly effective and efficient manner, providing exactly the information needed.

Style (1-5 points):

1 point: The response is poorly suited for speech interaction, possibly including structured elements like lists or being overly complex, disjointed, or difficult to understand.

2 points: The response is somewhat suitable but may be too long, too short, or awkwardly phrased, making it less effective in a speech interaction context.

3 points: The response is generally suitable for speech interaction, but it may have minor issues with length, clarity, or fluency that detract slightly from the overall effectiveness.

4 points: The response is well-suited for speech interaction, with appropriate length, clear language, and a natural flow. It is easy to understand when spoken aloud.

5 points: The response is perfectly suited for speech interaction. It is the ideal length, highly clear, and flows naturally, making it easy to follow and understand when spoken.

Below are the transcription of user's instruction and models' response:

[Instruction]: {**instruction**}

[Response]: {**response**}

After evaluating, please output the scores in JSON format: {"content": content score, "style": style score}. You don't need to provide any explanations.

SpeechGPT SpeechGPT (Zhang et al., 2023) is a speech-language model that supports both speech input and output. We use the chain-of-modality prompting adopted in the original paper for decoding, which sequentially outputs the text instruction, text response, and speech response based on the speech instruction.

SALMONN (+TTS) SALMONN (Tang et al., 2024) is a LLM capable of accepting speech and audio inputs and responding with text, enabling it to perform the S2TIF task. For the S2SIF task, we add a VITS TTS model after SALMOON to generate speech responses in a cascaded manner.

Qwen2-Audio (+TTS) Qwen2-Audio (Chu et al., 2024) is a powerful general-purpose audio understanding model capable of performing various audio-related tasks, including the S2TIF task. We also build a cascaded system with Qwen2-Audio and VITS to complete the S2SIF task.

Results

Table 1: ChatGPT scores for S2TIF and S2SIF tasks on the InstructS2S-Eval benchmark, along with the alignment scores between speech and text responses. Here we set $\Omega = +\infty$ for the S2SIF task.

Model	S2TIF		S2SIF		Alignment	
	Content \uparrow	Style \uparrow	Content \uparrow	Style \uparrow	ASR-WER \downarrow	ASR-CER \downarrow
SpeechGPT	2.59	3.15	1.58	1.81	47.62	37.26
SALMONN (+TTS)	2.57	2.79	2.46	2.84	21.77	20.25
Qwen2-Audio (+TTS)	2.73	2.64	2.32	2.58	55.72	53.09
LLaMA-Omni	3.23	3.81	2.69	3.12	11.61	7.59

Table 2: Latency, speech-text alignment and speech quality under different unit chunk sizes.

Chunk Size Ω	Latency (ms)	#Lagging Word	ASR-WER \downarrow	ASR-CER \downarrow	UTMOS \uparrow
10	226.13	1.82	10.44	6.94	3.2304
20	256.28	2.83	10.51	6.98	3.4748
40	326.63	4.68	10.99	7.24	3.6688
60	391.96	6.47	11.17	7.27	3.7549
80	467.34	8.29	11.40	7.45	3.7858
100	527.64	10.02	11.48	7.45	3.8242
$+\infty$	1924.62	41.40	11.61	7.59	3.9296

Table 3: Average decoding time (s) of different models on S2TIF and S2SIF tasks.

Model	S2TIF	S2SIF
SpeechGPT	4.28	25.60
SALMONN	4.78	/
Qwen2-Audio	8.42	/
LLaMA-Omni	1.49	1.92

Case Study

Table 4: Responses from different models for the instruction: “How do I wrap a present neatly?”

Model	Response
Qwen2-Audio	Wrapping a present neatly can be a bit tricky, but with some practice and patience, you can master the art of wrapping presents beautifully. Here are some steps to follow:\n\n1. Start with a clean, flat surface: Find a clean, flat surface where you can work on wrapping your present. A large table or countertop works well.\n\n2. Gather your supplies: You will need wrapping paper, gift wrap scissors, a ruler or measuring tape, and any other decorations you want to use (such as ribbon, bows, or stickers).\n\n3. Measure the gift: Use the ruler or measuring tape to measure the width and length of the gift. Make sure to add a little extra space around the edges for wrapping.\n\n4. Cut the wrapping paper: Cut the wrapping paper to the size you measured. If you have multiple pieces of wrapping paper, layer them together so that they fit neatly around the gift.\n\n5. Fold the paper: Begin by folding one of the sides of the wrapping paper over the gift, crease it well, and then fold the other side over the top. This creates a simple, yet attractive fold.\n\n6. Add decorations: If you like, you can add decorations such as ribbon, bows, or stickers to the gift. Simply attach them to the top of the gift using glue or double-sided tape.\n\n7. Wrap the gift: Starting at the bottom of the gift, place a piece of wrapping paper over it. Bring the left side of the paper up over the gift, tuck it in tightly, and press it down. Repeat this process on the right side of the gift, bringing the right side of the paper up over the gift, tucking it in tightly, and pressing it down.\n\n8. Finish the edges: Once you have wrapped the gift, trim off any excess paper along the edges using the gift wrap scissors. You can also add additional touches such as a bow or decorative label to complete the look.\n\nRemember, the key to wrapping a present neatly is to take your time and pay attention to detail. With practice, you'll be able to create beautiful wrapping paper presentations every time!
SALMOON	To wrap a present neatly, you can start by laying out a clean and flat surface to work on. Then, place the present on the surface and wrap it in a piece of wrapping paper that is slightly larger than the present. Fold the paper in half and then in half again, so that the top and bottom flaps meet in the middle. Then, fold the left and right flaps in towards the center, so that they meet at the top and bottom of the present. Finally, seal the present by pressing the flaps together and using some tape to secure them.
SpeechGPT	Wrapping a present neatly requires a few simple steps. First, you should fold the paper in half and then unfold it. Then, you should place the present in the center of the paper and fold the paper around it. Finally, you should tie the ends of the paper with a ribbon or string.
LLaMA-Omni	To wrap a present neatly, start with a flat surface, place the gift in the center, and fold one edge of the wrapping paper over the gift, crease it, and tape it down. Repeat this process with the remaining edges, making sure to align them evenly and smooth out any air bubbles.

The response given by LLaMA-Omni is more **detailed** and **helpful** while maintaining a **concise style**, outperforming previous models in speech interaction scenarios

Award

452	我们爱学习	基于物联网的智慧图书馆系统	上海大学
453	我看你真是饿了队	“内镜先锋”--基于深度学习的内镜镜视频帧智滤系统	山东师范大学
454	AnVive-麻醉深度智能控制系统	AnVive-麻醉深度智能控制系统	西安交通大学
455	酸甜苦辣咸	基于LLM与LangChain结合的本地表格自动填写系统	吉首大学
456	拉文克劳队	人工智能驱动的膀胱镜智能诊断系统	对外经济贸易大学
457	MedAI先锋队	智慧医疗：基于ChatGLM和LangChain的本地化医疗辅助系统	陕西科技大学
458	智绘传承	智绘传承——经典影像修复，让历史走进现实	南京理工大学
459	CodeACT	CodeACT: 高效微调框架及AI辅助编程系统	南京航空航天大学
460	小虎队	智慧司法办案平台	河南财经政法大学
461	医切有我们	医下知道-大模型赋能下的智慧医疗平台	贵州大学
462	星空探索者	星澜：天文领域大模型	贵州大学
463	痧韵流芳队	智能中医刮痧机器人诊疗系统	北京工业大学
464	NLP火炬队	“律智”--司法交互平台	贵州大学
465	智网创想家	智网先锋-大模型赋能的网络高阶自智技术先行者	北京邮电大学
466	退退退	“巡采运充”-智能果园机器人	中原工学院
467	OpenNeuro元擎科技	OpenNeuro——神经解码交互大模型生态及应用	哈尔滨工业大学

“华为杯”第六届中国研究生人工智能创新大赛全国三等奖初评结果公示

指导单位：教育部学位管理与研究生教育司

主办单位：中国学位与研究生教育学会、中国科协青少年科技中心

承办单位：哈尔滨工程大学

赞助单位：华为技术有限公司

关于第十三届“天宫杯”研究生创新实验竞赛安排的通知

作者：访问量：3560 发布时间：2024-09-19

各学院、各位研究生：

为进一步提高研究生创新实践能力，培养研究生创新精神和创新意识，营造浓厚的研究生创新氛围，学校决定举办第十三届“天宫杯”研究生创新实验竞赛。相关事宜通知如下：

一、竞赛内容

以研究生创新实验为主题，通过现场比赛的方式进行竞赛，鼓励研究生进行创新设计、创新研究和创新制造。

二、参赛对象

本次竞赛参赛对象为南京航空航天大学在校研究生，专业不限，人数不限。

三、参赛要求

本次竞赛以项目的形式进行申报，以个人或团队的名义申请，以学院为单位组织，鼓励跨学科、跨学院组成竞赛队。竞赛评审的主要依据是参赛作品的创新性、实践性和可展示性。

2023年立项的“研究生创新实验竞赛”培育项目中，**中期考核结果为“继续资助”**的项目必须提交参赛申请，中期考核结果为“结题”的项目可以提交参赛申请，其中已获省部级及以上奖项的，须参赛展示，但不参与特一二三等奖的评比，另设杰出成果奖。

其他已获校级及以上奖项的作品，本次竞赛不接受申报。如不据实申报，一经查出，取消其参赛资格及所获奖项。

Thanks