LauraGPT: Listen, Attend, Understand, and Regenerate Audio with GPT

Anonymous ACL submission

Abstract

Generative Pre-trained Transformer (GPT) models have achieved remarkable performance on various natural language processing tasks, and have shown great potential as backbones for audio-and-text large language models (LLMs). Previous mainstream audio-and-text LLMs use discrete audio tokens to represent both input and output audio; however, they suffer from performance degradation on tasks such as automatic speech recognition, speech-to-text translation, and speech enhancement over models using continuous speech features. In this paper, we propose **LauraGPT**, a novel unified audio-and-text GPT-based LLM for audio recognition, understanding, and generation. LauraGPT is a versatile LLM that can process both audio and text inputs and generate outputs in either modalities. We propose a novel data representation that combines continuous and discrete features for audio: LauraGPT encodes input audio into continuous representations using an audio encoder and generates output audio from discrete codec codes. We propose a one-step codec vocoder to overcome the prediction challenge caused by the multimodal distribution of codec tokens. We finetune LauraGPT using supervised multi-task learning. Extensive experiments show that LauraGPT consistently achieves comparable to superior performance compared to strong baselines on a wide range of audio tasks related to content, semantics, paralinguistics, and audio-signal analysis, such as automatic speech recognition, speech-to-text translation, text-to-speech synthesis, speech enhancement, automated audio captioning, speech emotion recognition, and spoken language understanding.

1 Introduction

Large language models (LLMs) are neural networks that generate natural language texts based on a given context. LLMs can learn from massive amounts of text data and mimic human language to acquire human knowledge. LLMs such as GPT-4 (OpenAI, 2023), PaLM2 (Anil et al., 2023), LLaMA (Touvron et al., 2023) have demonstrated impressive capabilities across various domains, exhibiting zero-shot generalization without the need for task-specific fine-tuning. However, these models are primarily limited to processing text data.

Recent research aims to seamlessly integrate text and audio since they are two important modalities for human communication. These efforts include **Audio-to-Text LLMs** (Radford et al., 2022; Zhang et al., 2023b; Deshmukh et al., 2023; Arora et al., 2023; Tang et al., 2023; Chu et al., 2023), which can convert audio input into text and perform tasks such as automatic speech recognition (ASR) and spoken language understanding (SLU); **Text-to-Audio LLMs** (Yang et al., 2023a; Vyas et al., 2023; Kreuk et al., 2023; Liu et al., 2023b; Huang et al., 2023a; Wang et al., 2023a), which can convert text input into audio and perform tasks such as text-to-speech synthesis (TTS) and text-to-music synthesis. An emerging line of research focuses on develop more universal and comprehensive **Audio-and-Text LLMs** (Ao et al., 2022; Chen et al., 2021b; Zhang et al., 2023a; Wang et al., 2023b; Rubenstein et al., 2023; Huang et al., 2023b), which can support audio-and-text tasks, that is, process and generate both audio and text and perform tasks such as speech enhancement (SE) and speech-to-speech translation (S2ST), in addition to tasks supported by audio-to-text and text-to-audio LLMs. Audio-to-text and text-to-audio LLMs can be considered as subsets of audio-and-text LLMs.

Audio-and-Text LLMs can be categorized into two directions. One direction builds **a collaborative AI system** using LLMs as controllers to interface specialized audio models, such as ASR and TTS models, to support various audio-and-text tasks (Shen et al., 2023; Huang et al., 2023b). These methods have serious drawbacks, including high complexity, significant resource consumption, and unavoidable error accumulation problems. The other direction develops a **unified Audio-and-Text LLM** leveraging LLMs as the backbone to support audio-and-text tasks (Ao et al., 2022; Chen et al., 2021b; Wang et al., 2023b; Rubenstein et al., 2023). Decoder-only audio-and-text LLMs (Zhang et al., 2023a; Wang et al., 2023b; Rubenstein et al., 2023) are the dominant technique under this category. These models convert continuous audio into discrete tokens and integrate text and audio tokens into unified vocabulary. These models suffer from information loss from quantization of speech signals into discrete tokens, which leads to notable performance degradation on ASR compared to models using continuous speech features (Chen et al., 2023a; Chang et al., 2023; Yang et al., 2023c; Puvvada et al., 2023). In this paper, we focus on improving the second category of unified Audio-and-Text LLMs. Moreover, recent advances in audio generation from unified audio-and-text LLMs (Wang et al., 2023a,b) discretize speech into codec codes, then use an autoregressive language model (LM) to predict output tokens from the first quantizer and use a non-autoregressive model to predict tokens from the other quantizers individually. One limitation of this mechanism is that it needs many prediction steps (hence called **multi-step audio synthesis scheme**) to generate good quality speech. Another limitation is that predicting the indices of the other codec groups is challenging due to the multi-modal distribution nature of codec tokens (Jenrungrot et al., 2023).

To overcome the drawbacks of existing *unified audio-and-text LLMs*, we propose **LauraGPT**, a novel **unified Audio-and-Text LLM** based on the GPT framework for audio recognition, understanding, and generation. LauraGPT is a versatile LLM that can process both audio and text inputs and generate outputs in either modalities, with a single model. We propose **a novel data representation that combines continuous and discrete features for audio**: LauraGPT encodes input audio into continuous representations using an audio encoder and generates output audio from discrete codec codes. This data representation improves the performance of audio-input tasks and also facilitates joint autoregressive modeling of audio and text features for audio generation tasks.

We also propose **a one-step codec vocoder in LauraGPT to address the two limitations of the popular multi-step audio synthesis scheme**. Our one-step codec vocoder uses a transformer-based predictor to estimate the sum of all codec token groups instead of the individual indices, by minimizing the reconstruction losses. Our approach simplifies the audio generation process to a *single* feed-forward calculation and also overcomes the prediction challenge caused by the multi-modal distribution of codec tokens.

We fine-tune LauraGPT using **supervised multi-task learning on diverse audio tasks**, including tasks focusing on content, semantics, paralinguistics, and audio-signal analysis, such as ASR, speech-to-text translation (S2TT), TTS, SE, automated audio captioning (AAC), speech emotion recognition (SER), and SLU. **Comprehensive experiments show that, to the best of our knowledge, LauraGPT[1] consistently achieves comparable to superior performance compared to strong baselines on the largest and the most diverse set of audio recognition, understanding, and generation tasks among existing decoder-only unified audio-and-text LLMs focusing on these tasks** (Zhang et al., 2023a; Wang et al., 2023b; Rubenstein et al., 2023). The results are remarkable since existing general speech models either focus solely on speech recognition and understanding tasks but neglect speech generative tasks, or support speech generation but suffer from severe performance degradation on speech recognition and understanding tasks.

2 Related Work

Audio-to-Text LLMs Audio-to-Text LLMs can generate text from audio inputs. Whisper (Radford et al., 2022) and USM (Zhang et al., 2023b) can perform speech recognition and translation across multiple languages and domains. Pengi (Deshmukh et al., 2023) is an audio LM that formulates audio tasks as text-generation tasks. UniverSLU (Arora et al., 2023) is a universal SLU model that supports various speech classification and sequence generation tasks. SALMONN (Tang et al., 2023) and Qwen-Audio (Chu et al., 2023) integrate pretrained text LLMs with separate speech and audio encoders into a single multimodal model.

Text-to-Audio LLMs Text-to-Audio LLMs can convert text input into audio output and perform tasks such as TTS or text-to-music synthesis. Recently, two prominent categories of approaches have emerged for generating audio from text prompts. In the first category, continuous representations such as utterance-level em-

[1]Demos are available at `https://lauragpt.github.io`

beddings (Elizalde et al., 2022; Liu et al., 2023a; Huang et al., 2023a) and Mel-frequency spectrograms (Nachmani et al., 2023) are used as the targets. However, continuous representations present a challenge for unified modeling of text and audio within a single LM. In the second category, discrete codec tokens are employed as audio representations and generated by diffusion models (Yang et al., 2023b) or autoregressive LMs (Kreuk et al., 2023; Borsos et al., 2023; Copet et al., 2023; Wang et al., 2023a). Among models in the second category, in models such as AudioGen (Kreuk et al., 2023), AudioLM (Borsos et al., 2023), and MusicGen (Copet et al., 2023), multiple output heads are used after the LM to predict synchronized or delayed groups of codec tokens. However, this mechanism is only suitable for audio generation and may not be applicable to diverse audio-and-text tasks. Alternatively, in VALL-E (Wang et al., 2023a), the LM predicts output tokens of the first quantizer, while tokens of the remaining quantizers are predicted by a non-autoregressive model one by one. This mechanism requires numerous prediction procedures to generate acceptable speech quality. Moreover, the indices of the remaining codec groups are challenging to predict due to the multi-modal distribution nature of codec tokens (Jenrungrot et al., 2023).

Audio-and-Text LLMs Audio-and-Text LLMs can process and generate both audio and text, which can be categorized into two directions. One direction uses LLMs as controllers to interface specialized audio models, such as ASR and TTS models, to enable direct audio interaction with LLMs and support various audio-and-text tasks, such as HuggingGPT (Shen et al., 2023) and AudioGPT (Huang et al., 2023b). However, these models are complex, resource-intensive, and prone to error accumulation. The second direction uses LLMs as the backbone for a unified model that handles audio-and-text tasks (Ao et al., 2022; Chen et al., 2021b; Wang et al., 2023b; Rubenstein et al., 2023). SpeechT5 (Ao et al., 2022) and SpeechNet (Chen et al., 2021b) perform various speech tasks with an encoder-decoder model, but they require modal-specific pre-nets and post-nets to deal with different input&output modalities. VioLA (Wang et al., 2023b), AudioPaLM (Rubenstein et al., 2023), SpeechGPT (Zhang et al., 2023a), and SpeechGen (Wu et al., 2023) use decoder-only Transformers to model discrete audio tokens and text tokens as a shared vocabulary, but they suffer from information loss from quantization of audio signals into discrete tokens (Chen et al., 2023a; Chang et al., 2023; Yang et al., 2023c; Puvvada et al., 2023).

3 Methodology

Figure 1 depicts the architecture of the proposed LauraGPT. Section 3.1 describes the audio encoder, the text tokenizer, and the modified GPT LM for unified audio-and-text modeling. Section 3.2 elaborates the audio tokenizer. Section 3.3 introduces an efficient one-step codec vocoder for converting audio tokens into high-quality raw waveforms. Section 3.4 describes the multi-task fine-tuning and shows that LauraGPT provides an extensible framework for supporting more complex tasks.

3.1 Modified Language Model for Unifying Audio-and-Text Modeling

For audio inputs, different from other audio-and-text LLMs using discrete tokens to represent audio inputs, we extract the log-compressed Mel spectrogram features and convert them into *continuous representations* using a Conformer-based audio encoder. Text inputs and outputs are tokenized using the Qwen tokenizer (Bai et al., 2023), which inherits the tiktoken tokenizer (Jain, 2022) and incorporates additional augmentations for commonly used characters and words in different languages. The tokenized input text undergoes embedding matrix transformation to generate dense vectors. The audio representations and text embeddings have the same dimension D. The Conformer-based encoder is initialized with weights from a pre-trained ASR model (Gao et al., 2023). Since batch normalization can lead to endless loop decoding, we replace it with layer normalization in the Conformer-based encoder (details are in Appendix C.2).

To achieve audio generation capabilities, the audio outputs are discretized into tokens using an audio tokenizer (Section 3.2) to obtain *discrete representations* and the softmax output layer is augmented with the audio tokens. As a result, the weight matrix $\mathbf{W}$ in the output layer is of size $(N + M + L) \times D$ and is utilized to calculate the logits for audio and text tokens at each position, where N, M, and L denote the vocabulary sizes of text, audio, and task tokens, respectively. Task tokens are used to inform the model which task should be performed. Note that in order to control the sequence length, we perform the low frame rate (LFR) method (Gao et al., 2020) to downsample audio inputs to 60ms and only select the first codec group of the audio outputs.

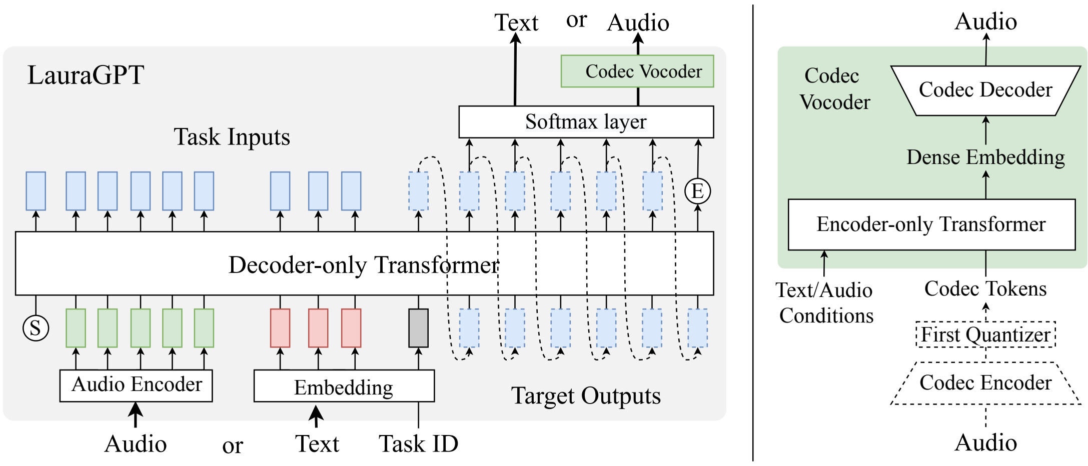

Figure 1: The overview of the proposed LauraGPT model. The right part provides an enlarged view of the one-step Codec Vocoder (Section 3.3) in LauraGPT. The dashed modules are only used in the training stage. Ⓢ and Ⓔ denote the "start of sequence" and "end of sequence" tokens. We omit the text tokenizer and detokenizer for simplicity.

Based on the aforementioned representations, the GPT backbone is trained to model various audio and text tasks by minimizing the cross-entropy loss:

$$\mathcal{L}_{LM} = -\frac{1}{T_v}\sum_{j=1}^{T_v} \log p_\theta\left(\mathbf{v}_j | \mathbf{u}_{1:T_u}, \mathbf{u}_{task}, \mathbf{v}_{1:j-1}\right) \tag{1}$$

where $\mathbf{u}$ denotes the input embeddings with a sequence length T_u and $\mathbf{v}$ represents the sequence of target tokens with a length T_v. To specify a task, a special task-related token $\mathbf{u}_{task}$ is inserted between the input embeddings and output tokens. Note that only the losses of outputs are taken into account, while losses on inputs and task token embeddings are masked out. After the final output layer, audio tokens are decoded to raw waveforms using a codec vocoder (Section 3.3). Since it is challenging to train an LLM from scratch with limited data and computational resources, we use the open-source GPT LLM, Qwen (Bai et al., 2023), as the backbone. Qwen is pre-trained on a diverse corpus covering various domains in English and Chinese and supports 8192 context length. Compared with other open-source GPT models with similar model sizes, Qwen models demonstrate impressive competitiveness, achieving better performance on widely used benchmarks, especially on Chinese tasks (Bai et al., 2023). Within LauraGPT, all parameters including the Qwen backbone are jointly optimized, except for the codec vocoder, which is trained independently and kept frozen during both training and inference stages of LauraGPT.

3.2 Audio Tokenizer

For audio generation, we utilize a codec model as the audio tokenizer to extract *discrete* representations. Our codec model shares a similar architecture as EnCodec (Défossez et al., 2022), which comprises convolutional recurrent encoder and decoder (Tagliasacchi et al., 2020) and a residual vector quantizer (RVQ) (Vasuki and Vanathi, 2006). We enhance the original EnCodec model with the following modifications: 1) Add reconstruction losses in the magnitude spectrum domain to improve the quality of middle- and high-frequency signals. 2) Stack five strided convolution blocks with strides of $[8, 5, 4, 2, 2]$ to address the challenge of long sequence lengths, resulting in a token rate of 25Hz for each token group. 3) Use 32 quantizers with structured dropout in the RVQ module, each with vocabulary size 1024. This revision improves speech quality with more quantizers while preserving most information in the shallow quantizers. The encoder and the *first RVQ quantizer* are used as the audio tokenizer, and **the outputs of the first quantizer are used as the audio tokens**. The choice of the first N RVQ quantizers to use is a tradeoff between performance and sequence length (hence efficiency). The remaining quantizers and the decoder are only used when training the codec model. Details of training and the pre-trained codec model are in (Du et al., 2023).

3.3 One-step Codec Vocoder for Audio Generation

We propose a one-step codec vocoder in LauraGPT to generate waveforms from the audio tokens, which are extracted from the *first* quantizer as described in Section 3.2. Our vocoder comprises two components: a transformer-based predictor and a codec decoder. The predictor is trained to estimate the summation of codec embeddings from the 32 RVQ quantizers by minimizing the L1 and L2 distances between the predicted embeddings $\hat{\mathbf{E}}$ and their corresponding ground truth $\mathbf{E}$:

$$\mathcal{L}_{pre} = \sum_{t,i}^{T,D_c} |\mathbf{E}_{t,i} - \hat{\mathbf{E}}_{t,i}|_1 + |\mathbf{E}_{t,i} - \hat{\mathbf{E}}_{t,i}|_2 \quad (2)$$

where T denotes the total number of frames and D_c denotes the dimension of the codec embeddings. After obtaining the estimated embeddings, the decoder of an pre-trained codec model is utilized to reconstruct the raw audio waveforms.

Alongside the predicted audio tokens from the LLM, text and audio inputs are used as conditions and fed to the predictor. For zero-shot TTS task, the text inputs serve as a condition as well as the prompt audio features. For SE task, the input noisy speech features are employed as conditions. Such text and audio conditionings allow the model to generate high-quality audio signals by leveraging the diverse information in prompt audios and noisy speeches, which is lacked in the discrete tokens (output from the first quantizer). Therefore, different from existing Text-to-Audio LLMs, **our approach simplifies the audio generation process to a single feed-forward calculation and overcomes the prediction challenge caused by the multi-modal distribution of codec tokens**.

3.4 Multi-task Finetuning

Basic Tasks We unify modeling of the following *basic tasks* in the single LauraGPT model and use these tasks for multi-task fine-tuning: Automatic Speech Recognition (**ASR**), Spoken Language Understanding (**SLU**), Speech-to-Text Translation (**S2TT**), Speech Emotion Recognition (**SER**), Automated Audio Captioning (**AAC**), Speech Enhancement (**SE**), and Text-to-speech Synthesis (**TTS**). Task definitions are in Appendix A.1.

Unified Task Expression LauraGPT operates based on a unified task expression: [input embeddings, task ID, output tokens]. With the same inputs, the desired outputs can differ across tasks. For instance, ASR and S2TT tasks require different outputs even for the same audio input. Task tokens are included in both input embedding and output weight matrices. The TTS task takes text embeddings as inputs, while the ASR, S2TT, SLU, SE, ACC, and SER tasks take audio encodings as inputs. The TTS and SE tasks use audio tokens as the target outputs, while the remaining tasks use text tokens as the target outputs.

Support More Complex Tasks With its modularized design, LauraGPT provides an extensible framework to support complex tasks. By breaking a task into sub-tasks among the basic tasks and cascading the raw inputs and model outputs of sub-tasks, LauraGPT can perform more complex tasks. For example, we demonstrate that LauraGPT is capable of performing the advanced speech-to-speech translation (S2ST) task by combining the S2TT and TTS tasks. Initially, a sequence is constructed to translate the speech content into the target language text using the S2TT task token: [audio encoding, <S2TT>]. Subsequently, the translated text is combined with the TTS task token to synthesize speech: [text embedding, <TTS>]. If maintaining the speaker identity is desired, the original inputs and content can be incorporated to perform *personalized TTS*. This can be achieved with an input sequence as [ASR recognized text embedding, S2TT translated text embedding, <TTS>, audio token of input speech], where ASR recognized text embedding is obtained using the ASR task: [audio encoding, <ASR>]. This approach treats the bilingual text as the complete input and allows the model to generate an output sequence of codec tokens while maintaining the same speaker identity. Audio samples of S2ST can be found on the demo site. More examples of complex tasks are in Appendix D.

4 Experimental Settings

Model Architecture The Conformer-based audio encoder consists of 32 conformer blocks. Each block consists of a feed-forward module with 1536 units, an attention module with 16 heads and a dimension of 512, a convolutional module including the pointwise and depthwise convolution layers, and a second feed-forward module with 1536 units. Sinusoidal positional encoding is applied on the audio inputs. For a trade-off between performance and training efficiency, we use Qwen-1.8B[2] as the backbone and LauraGPT has 2B parameters. Qwen-1.8B comprises 24 transformer layers with a hidden size 2048 and 16 attention heads. **Although Conformer and Qwen-1.8B are selected as the audio encoder and GPT backbone, they can be replaced by other encoders and GPT models**.

Training Setup In all experiments, we initialize the Qwen backbone and audio encoder with the pretrained checkpoints. We then optimize the model parameters through multi-task fine-tuning. The

[2] https://github.com/QwenLM/Qwen

training&test datasets and evaluation metrics are presented in Appendix A.2 and A.3. Appendix A.4 describes the three-stage training process to address the significant variation in data volume across different tasks, and details the inference process.

5 Results and Analysis

Section 5.1 presents the main results of performance comparison on the basic tasks from the state-of-the-art (SOTA) model, a **comparable** baseline, and our LauraGPT. Ablation studies in Section 5.2 demonstrate the advantages of using continuous representations for audio inputs in LauraGPT by comparing to a counterpart with both discrete inputs and outputs (denoted **Discrete IO**), the superiority of our one-step codec vocoder, and effectiveness of multi-task finetuning. Further analyses include comparison with related unified Audio-and-Text LLMs (Appendix B), more analysis of multi-task fine-tuning on SER task (Appendix C.1), comparing batch normalization with layer normalization in the audio encoder (Appendix C.2), and studying impact of initialization from pre-trained models (Appendix C.3).

5.1 Results on All Tasks

Table 1 shows the results from the SOTA model, a comparable baseline, and our LauraGPT[3], in that order, on a variety of speech recognition, understanding, and generation benchmarks. The SOTA model yields the best results on each test set based on our literature review. The baseline for each task is chosen to facilitate fair comparison with LauraGPT: they are comparable to LauraGPT in model architecture or training data and are also common competitive baselines in the literature. We cite the SOTA results to validate that LauraGPT consistently performs competitively on all the speech recognition, understanding, and generation tasks and the baselines are competitive. However, LauraGPT results cannot be fairly compared to the SOTA results. Specifically, QwenAudio achieves SOTA performance on most speech understanding tasks, but compared to LauraGPT, QwenAudio uses a much larger LLM ($\sim$7B VS. our 1.8B LLM), and uses the Whisper audio encoder trained on a large amount of ASR data while we use a Conformer encoder trained on much less data. Moreover, QwenAudio does not support speech generative tasks hence cannot handle SE and TTS tasks. Paraformer-large and UniverSLU achieve SOTA results on AISHELL-2 test-ios for Chinese ASR and on SLURP test for SLU; however, they only support single tasks and also train on more data than LauraGPT on the corresponding task. Appendix B shows that LauraGPT greatly outperforms Whisper Large V2 on Chinese ASR test sets while the gap on English ASR test sets are primarily attributed to the much smaller English data used for training LauraGPT. For TTS, the SOTA VALL-E Phone outperforms baseline VALL-E Token[4], suggesting the importance of text representation for TTS. Compared to both VALL-E models, LauraGPT achieves comparable speaker similarity (SECS) and speech quality (MOSNet). The degradation in content consistency (WER) from LauraGPT results from the generalization issue, since the training data is too limited for LauraGPT with 2B parameters. Overall, the results show that **LauraGPT consistently achieves comparable to superior performance than strong baselines on diverse speech tasks, demonstrating the general effectiveness of LauraGPT on speech recognition, understanding, and generative tasks**.

5.2 Analysis

Discrete VS. Continuous Representations for Audio Inputs Existing unified Audio-and-Text LLMs use discrete tokens to represent audio inputs. We analyze the efficacy of using continuous representations for audio inputs in LauraGPT by comparing to its counterpart **Discrete IO** on ASR, S2TT, and SE tasks, representing **audio-input recognition and understanding, and audio generation capacities**. In Discrete IO, both audio inputs and outputs are represented by flattened codec tokens from *the first four quantizers*[5], resulting in a token rate of 100Hz. In LauraGPT, audio inputs are represented by continuous acoustic features, which are also fed into our one-step vocoder as a condition to achieve high-quality outputs. Table 2 shows that **LauraGPT consistently outperforms Discrete IO with remarkable gains on all tasks**. **For ASR task**, the performance degrades drastically

[3]Our results are from single runs due to the stability of the models and limited computational resources.

[4]We re-implement two VALL-E models with 0.34B trainable parameters, both trained with the same data as LauraGPT. VALL-E Phone uses phonemes as the text input representation, while VALL-E Token uses WordPiece tokens from the text tokenizer.

[5]Using outputs of the first quantizer (as in LauraGPT) for audio tokenizer renders very poor performance for audio-input tasks with the Discrete IO models. Using more quantizers improves performance but reduces efficiency.

Table 1: Results from the **SOTA**, a *comparable* baseline, and our **LauraGPT**, in that order, on **speech recognition, understanding, and generation tasks**. The better results between the baseline and LauraGPT are in bold.

Task	Test Set	Metric	Model	Performance
ASR	AISHELL-1 test	CER ↓	Qwen-Audio (Chu et al., 2023)	1.3
			MMSpeech-large (Zhou et al., 2022)	1.9
			LauraGPT	**1.8**
	AISHELL-2 test-ios	CER ↓	Paraformer-large (Gao et al., 2023)	2.9
			MMSpeech-large (Zhou et al., 2022)	3.9
			LauraGPT	**3.2**
	LibriSpeech test-clean	WER ↓	Qwen-Audio (Chu et al., 2023)	2.0
			Whisper Large V2 (Radford et al., 2023)	**2.5**
			LauraGPT	4.4
	LibriSpeech test-other	WER ↓	Qwen-Audio (Chu et al., 2023)	4.2
			Whisper Large V2 (Radford et al., 2023)	**4.9**
			LauraGPT	7.7
SLU	SLURP test	Intent ACC ↑ | SLU-F1 ↑	UniverSLU (Arora et al., 2023)	90.5 | 80.5
			Wav2Vec 2.0 (Ravanelli et al., 2021)	85.3 | **74.6**
			LauraGPT	**87.9** | 73.5
S2TT	BSTC dev (Zh→EN)	BLEU ↑	-	-
			Cascade-System (Zhang et al., 2021)	**18.2**
			LauraGPT	17.8
	CoVOST2 test set (En→Zh)	BLEU ↑	Qwen-Audio (Chu et al., 2023)	41.5
			EncDec-Attn (Wang et al., 2020)	25.4
			LauraGPT	**38.5**
SER	MELD test	WA ↑ | UA ↑ | WF1 ↑	Qwen-Audio (Chu et al., 2023)	0.557 | - | -
			Vesper-12 (Chen et al., 2023b)	**0.535** | 0.268 | 0.480
			LauraGPT	0.507 | **0.312** | **0.492**
AAC	Clotho eval	SPICE ↑ | CIDEr ↑ | SPIDEr ↑	Qwen-Audio (Chu et al., 2023)	0.14 | 0.44 | 0.29
			Ensemble (Koizumi et al., 2020)	**0.09** | **0.32** | **0.21**
			LauraGPT	0.08 | 0.22 | 0.15
SE	Mixup of LibriSpeech test-clean, FSD50K and noise-92	WER ↓ | PESQ ↑ | STOI ↑	-	-
			CMGAN (Cao et al., 2022)	**12.29** | 2.95 | **91.0**
			LauraGPT	15.94 | **2.97** | 88.0
TTS	AISHELL-1	CER ↓ | SECS ↑ | MOSNet ↑	VALL-E Phone (Wang et al., 2023a)	4.75 | 0.91 | 3.22
			VALL-E Token (Wang et al., 2023a)	**6.52** | **0.91** | **3.19**
			LauraGPT	6.91 | 0.90 | 3.14
	LibriTTS	WER ↓ | SECS ↑ | MOSNet ↑	VALL-E Phone (Wang et al., 2023a)	4.30 | 0.92 | 3.28
			VALL-E Token (Wang et al., 2023a)	**6.57** | **0.93** | **3.28**
			LauraGPT	8.62 | 0.91 | 3.26

when replacing continuous features with discrete audio tokens. Although the performance degradation can be reduced by using more quantizers (more codec groups), e.g. 32 (Puvvada et al., 2023), more codec groups always cause higher token rates and longer sequence and in turn higher computational demands. **For S2TT task**, Discrete IO only yields BLEU scores of 5.1 and 5.0 on test sets, basically suggesting lack of translation capability. **For SE task**, using codec tokens as inputs cannot improve the quality and intelligibility of noisy speeches, suggesting lack of enhancement capability, probably because the distribution of noisy speech is too complicated to be accurately represented by four groups of discrete audio tokens.

Comparison on Audio Synthesis Schemes VALL-E (Wang et al., 2023a) introduces a commonly used scheme formulating audio synthesis as a *classification problem*: A neural network is shared to predict the codec tokens in the following group with the previous ones as inputs and synthesizing target audio requires multiple steps or iterations to achieve a reasonable speech quality. In contrast, our one-step codec vocoder formulates audio synthesis as a *regression problem*. As described in Section 3.3, our one-step codec vocoder simplifies audio synthesis into a single feed-forward calculation and overcomes the pre-

Table 2: Comparison of Discrete IO models and LauraGPT on ASR, S2TT, and SE tasks for analysis of discrete VS. continuous representations for audio inputs. The best results on each test set are in bold.

Task	Dataset	Metric	Discrete IO	LauraGPT
ASR	AISHELL-1 test	CER ↓	7.1	**1.8**
	AISHELL-2 test-ios	CER ↓	8.6	**3.2**
	LibriSpeech test-clean	WER ↓	9.1	**4.4**
	LibriSpeech test-other	WER ↓	24.0	**7.7**
S2TT	BSTC dev (Zh→EN)	BLEU ↑	5.1	**17.8**
	CoVOST2 test set (En→Zh)	BLEU ↑	5.0	**38.5**
SE	Mixup of LibriSpeech test-clean, FSD50K and noise-92	PESQ ↑	1.96	**2.97**
		STOI ↑	64.0	**88.0**
		WER ↓	53.97	**15.94**

Table 3: Comparison of our one-step audio synthesis scheme and the multi-step scheme on the SE task.

Scheme	PESQ ↑	STOI(%) ↑	CER ↓	WER ↓
Multi-step	2.55	**88.0**	10.52	19.32
One-step	**2.97**	**88.0**	**9.05**	**15.94**

diction challenge caused by the multimodal distribution of codec tokens. Table 3 shows that **our one-step codec vocoder greatly outperforms the multi-step scheme in terms of content consistency (CER, WER) and speech quality (PESQ), while obtaining the same intelligibility (STOI)**.

Effectiveness of Multi-task Finetuning The multi-task fine-tuned LauraGPT (Section 3.4) could be advantageous over individual single-task models: (1) Multi-task learning could exploit synergy between tasks and reduce over-fitting, in turn yield high performance on diverse tasks and achieve better performance than single-task training. (2) Multi-task learning could learn a single model capable of supporting a wide range of tasks, hence practical deployment is greatly simplified through unified model implementation and API.

We investigate whether the multi-task trained LauraGPT could achieve better performance than single-task training for tasks with limited training data. Among the basic tasks (Table 5), AAC, SLU, and SER tasks all have limited training data. We initialize the Qwen backbone and the audio encoder the same as LauraGPT before conducting multi-task training, then train the single-task model only using the task-specific training data. The results are shown in Table 4.

For the AAC task, we find that the multi-task trained LauraGPT outperforms the single-task model on SPICE, CIDEr and SPIDEr on the Clotho evaluation set. **For the SLU task**, on the SLURP test set, LauraGPT greatly outperforms the single-task model on intent accuracy by **+2.9** absolute and on SLU-F1 by **+22.5** absolute. **For the SER task**, on the MELD test set, LauraGPT substantially outperforms the single-task model in terms of UA and the primary WF1 metrics, while the WA result is slightly worse. More analyses in Appendix C.1 show that multi-task learning dramatically improves accuracies of the minority classes. **In summary, these results verify that multi-task learning for LauraGPT consistently achieves better performance than single-task training for tasks with limited training data.**

Table 4: Comparison of single-task finetuning and multi-task finetuning on the AAC, SLU, and SER tasks.

Task	Dataset	Metric	Single	Multi
AAC	Clotho eval	SPICE ↑	0.07	**0.08**
		CIDEr ↑	0.16	**0.22**
		SPIDEr ↑	0.11	**0.15**
SLU	SLURP test	Intent ACC ↑	85.0	**87.9**
		SLU-F1 ↑	51.0	**73.5**
SER	MELD test	WA ↑	**0.508**	0.507
		UA ↑	0.221	**0.312**
		WF1 ↑	0.426	**0.492**

6 Conclusion

We propose LauraGPT that can handle both audio and text inputs and outputs and perform audio recognition, understanding, and generation. We propose combining continuous and discrete features for audio and a one-step codec vocoder, and employ multi-task learning. Experiments demonstrate that LauraGPT achieves comparable to superior performance compared to strong baselines on a wide range of speech tasks on content, semantics, paralinguistics, and audio-signal analysis.

Limitations

In this work, in order to support a wide range of audio recognition, understanding, and generation tasks, we choose to train all parameters in LauraGPT during supervised multi-task finetuning, including the Qwen backbone, except for the codec vocoder. This strategy results in substantial computations for training. In future work, we plan to investigate parameter-efficient fine-tuning to reduce computation demands. Also, due to the limited computation resources, our comparisons between the multi-task trained LauraGPT and single-task models are focused on the low-resource tasks, that is, AAC, SLU, and SER tasks. We find that multi-task learning for LauraGPT consistently achieves better performance than single-task training for tasks with limited training data. Next, we plan to complete comparisons of LauraGPT and single-task models on all tasks, including relatively rich-resource tasks such as ASR. These studies will promote understandings on where tasks could benefit from each other, including tasks with even conflicting objectives. We also plan to conduct deeper analysis on the potential risk of catastrophic forgetting of the original text capabilities of the pre-trained text LLM, due to multi-task learning of speech tasks. Note that exploration of parameter-efficient fine-tuning may also help preserve the original text capabilities of the pre-trained text LLMs.

LauraGPT relies on discrete audio tokens for speech generative tasks. Our research shows that the performance of this paradigm strongly depends on the quality of the audio tokenizer. We plan to systematically analyze the impact of various audio tokenizers on diverse audio generative tasks. We plan to develop new audio tokenizers that are more suitable for unified Auio-and-Text LLMs and provide desirable representations for generative tasks.

There are great emerging interests in fundamental speech models that are similar to those in the field of NLP. This is a tremendously valuable research direction. Our work achieves important milestone for this research question, as we explore and provide promising answers to the following question: *How to design more efficient and scalable unified GPT-style Audio-and-Text LLMs than existing approaches that can leverage large-scale labeled data and achieve highly competitive performance on a diverse set of speech tasks, including speech recognition, understanding and generation, using a single model?* Note that previous general speech models either focus solely on speech recognition and understanding tasks but neglect speech generative tasks, or support speech generation but suffer from severe performance degradation on speech recognition and understanding tasks.

Inspired by the recent advances of LLMs in NLP, we envision that the fundamental speech models should have the following capabilities:

- In-context learning ability like GPT-3, which can learn from few-shot examples and adapt to new tasks, such as predicting the age of the speaker from a speech sample.
- Instruction-following ability like InstructGPT and ChatGPT, which can perform the appropriate speech-related task given a natural language instruction, such as synthesizing a speech with a specific emotion or style.
- General audio modeling abilities, i.e., speech, non-speech audio, and music, such as music generation.

Our work demonstrates that the current LauraGPT has made solid progress and reached one important milestone toward a speech foundation model. From LauraGPT to the next-generation speech foundation model we envision, most remaining efforts are in more task data collection and more self-supervised and/or supervised pre-training and supervised fine-tuning. There is no need to modify the model architecture.

References

Adaeze Adigwe, Noé Tits, Kevin El Haddad, Sarah Ostadabbas, and Thierry Dutoit. 2018. The emotional voices database: Towards controlling the emotion dimension in voice generation systems. *arXiv preprint arXiv:1806.09514*.

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernández Ábrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan A. Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vladimir Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, and et al. 2023. Palm 2 technical report. *CoRR*, abs/2305.10403.

Junyi Ao, Rui Wang, Long Zhou, Chengyi Wang, Shuo Ren, Yu Wu, Shujie Liu, Tom Ko, Qing Li, Yu Zhang, Zhihua Wei, Yao Qian, Jinyu Li, and Furu Wei. 2022. Speecht5: Unified-modal encoder-decoder pre-training for spoken language processing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 5723–5738. Association for Computational Linguistics.

Siddhant Arora, Hayato Futami, Jee-weon Jung, Yifan Peng, Roshan S. Sharma, Yosuke Kashiwagi, Emiru Tsunoo, and Shinji Watanabe. 2023. Universlu: Universal spoken language understanding for diverse classification and sequence generation tasks with a single network. *CoRR*, abs/2310.02973.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and et. al. 2023. Qwen technical report. *arxiv preprint, 2309.16609*.

Emanuele Bastianelli, Andrea Vanzo, Pawel Swietojanski, and Verena Rieser. 2020. Slurp: A spoken language understanding resource package. *arXiv preprint arXiv:2011.13205*.

Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matthew Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, and Neil Zeghidour. 2023. Audiolm: A language modeling approach to audio generation. *IEEE ACM Trans. Audio Speech Lang. Process.*, 31:2523–2533.

Hui Bu, Jiayu Du, Xingyu Na, Bengu Wu, and Hao Zheng. 2017. Aishell-1: An open-source mandarin speech corpus and a speech recognition baseline. In *2017 20th conference of the oriental chapter of the international coordinating committee on speech databases and speech I/O systems and assessment (O-COCOSDA)*, pages 1–5. IEEE.

Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N Chang, Sungbok Lee, and Shrikanth S Narayanan. 2008. Iemocap: Interactive emotional dyadic motion capture database. *Language resources and evaluation*, 42:335–359.

Houwei Cao, David G Cooper, Michael K Keutmann, Ruben C Gur, Ani Nenkova, and Ragini Verma. 2014. Crema-d: Crowd-sourced emotional multimodal actors dataset. *IEEE transactions on affective computing*, 5(4):377–390.

Ruizhe Cao, Sherif Abdulatif, and Bin Yang. 2022. CMGAN: Conformer-based Metric GAN for Speech Enhancement. In *Proc. Interspeech 2022*, pages 936–940.

Xuankai Chang, Brian Yan, Yuya Fujita, Takashi Maekaku, and Shinji Watanabe. 2023. Exploration of efficient end-to-end ASR using discretized input from self-supervised learning. *CoRR*, abs/2305.18108.

Guoguo Chen, Shuzhou Chai, Guanbo Wang, Jiayu Du, Wei-Qiang Zhang, Chao Weng, Dan Su, Daniel Povey, Jan Trmal, Junbo Zhang, et al. 2021a. Gigaspeech: An evolving, multi-domain asr corpus with 10,000 hours of transcribed audio. *arXiv preprint arXiv:2106.06909*.

Qian Chen, Wen Wang, Qinglin Zhang, Siqi Zheng, Shiliang Zhang, Chong Deng, Yukun Ma, Hai Yu, Jiaqing Liu, and Chong Zhang. 2023a. Loss masking is not needed in decoder-only transformer for discrete-token based ASR. *CoRR*, abs/2311.04534.

Weidong Chen, Xiaofen Xing, Peihao Chen, and Xiangmin Xu. 2023b. Vesper: A compact and effective pretrained model for speech emotion recognition. *CoRR*, abs/2307.10757.

Yi-Chen Chen, Po-Han Chi, Shu-Wen Yang, Kai-Wei Chang, Jheng-Hao Lin, Sung-Feng Huang, Da-Rong Liu, Chi-Liang Liu, Cheng-Kuang Lee, and Hung-yi Lee. 2021b. Speechnet: A universal modularized model for speech processing tasks. *CoRR*, abs/2105.03070.

Yunfei Chu, Jin Xu, Xiaohuan Zhou, Qian Yang, Shiliang Zhang, Zhijie Yan, Chang Zhou, and Jingren Zhou. 2023. Qwen-audio: Advancing universal audio understanding via unified large-scale audio-language models. *CoRR*, abs/2311.07919.

Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. 2023. Simple and controllable music generation. *CoRR*, abs/2306.05284.

Soham Deshmukh, Benjamin Elizalde, Rita Singh, and Huaming Wang. 2023. Pengi: An audio language model for audio tasks. *CoRR*, abs/2305.11834.

Konstantinos Drossos, Samuel Lipping, and Tuomas Virtanen. 2020. Clotho: An audio captioning dataset. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 736–740. IEEE.

Jiayu Du, Xingyu Na, Xuechen Liu, and Hui Bu. 2018. Aishell-2: Transforming mandarin asr research into industrial scale. *arXiv preprint arXiv:1808.10583*.

Zhihao Du, Shiliang Zhang, Kai Hu, and Siqi Zheng. 2023. Funcodec: A fundamental, reproducible and integrable open-source toolkit for neural speech codec.

Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. 2022. High fidelity neural audio compression. *arXiv:2210.13438*.

Benjamin Elizalde, Soham Deshmukh, Mahmoud Al Ismail, and Huaming Wang. 2022. CLAP: learning audio concepts from natural language supervision. *CoRR*, abs/2206.04769.

Eduardo Fonseca, Xavier Favory, Jordi Pons, Frederic Font, and Xavier Serra. 2022. FSD50K: an open dataset of human-labeled sound events. *IEEE ACM Trans. Audio Speech Lang. Process.*, 30:829–852.

Zhifu Gao, Zerui Li, Jiaming Wang, Haoneng Luo, Xian Shi, Mengzhe Chen, Yabin Li, Lingyun Zuo, Zhihao Du, Zhangyu Xiao, and Shiliang Zhang. 2023. Funasr: A fundamental end-to-end speech recognition toolkit. In *INTERSPEECH*.

Zhifu Gao, Shiliang Zhang, Ming Lei, and Ian McLoughlin. 2020. San-m: Memory equipped self-attention for end-to-end speech recognition. *arXiv preprint arXiv:2006.01713*.

Zhifu Gao, Shiliang Zhang, Ian McLoughlin, and Zhijie Yan. 2022. Paraformer: Fast and accurate parallel transformer for non-autoregressive end-to-end speech recognition. In *INTERSPEECH*, pages 2063–2067. ISCA.

Rongjie Huang, Jiawei Huang, Dongchao Yang, Yi Ren, Luping Liu, Mingze Li, Zhenhui Ye, Jinglin Liu, Xiang Yin, and Zhou Zhao. 2023a. Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 13916–13932. PMLR.

Rongjie Huang, Mingze Li, Dongchao Yang, Jiatong Shi, Xuankai Chang, Zhenhui Ye, Yuning Wu, Zhiqing Hong, Jiawei Huang, Jinglin Liu, Yi Ren, Zhou Zhao, and Shinji Watanabe. 2023b. Audiogpt: Understanding and generating speech, music, sound, and talking head. *CoRR*, abs/2304.12995.

Philip Jackson and SJUoSG Haq. 2014. Surrey audio-visual expressed emotion (savee) database. *University of Surrey: Guildford, UK*.

Shantanu Jain. 2022. tiktoken: A fast BPE tokeniser for use with OpenAI's models.

Teerapat Jenrungrot, Michael Chinen, W. Bastiaan Kleijn, and et al. 2023. LMCodec: A low bitrate speech codec with causal transformer models. In *ICASSP*.

Chris Dongjoo Kim, Byeongchang Kim, Hyunmin Lee, and Gunhee Kim. 2019. Audiocaps: Generating captions for audios in the wild. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 119–132.

Tom Ko, Vijayaditya Peddinti, Daniel Povey, Michael L Seltzer, and Sanjeev Khudanpur. 2017. A study on data augmentation of reverberant speech for robust speech recognition. In *ICASSP*, pages 5220–5224.

Tom Kocmi, Rachel Bawden, Ondřej Bojar, Anton Dvorkovich, Christian Federmann, Mark Fishel, Thamme Gowda, Yvette Graham, Roman Grundkiewicz, Barry Haddow, et al. 2022. Findings of the 2022 conference on machine translation (wmt22). In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 1–45.

Yuma Koizumi, Daiki Takeuchi, Yasunori Ohishi, Noboru Harada, and Kunio Kashino. 2020. The ntt dcase2020 challenge task 6 system: Automated audio captioning with keywords and sentence length estimation. *arXiv preprint arXiv:2007.00225*.

Felix Kreuk, Gabriel Synnaeve, Adam Polyak, Uriel Singer, Alexandre Défossez, Jade Copet, Devi Parikh, Yaniv Taigman, and Yossi Adi. 2023. Audiogen: Textually guided audio generation. In *ICLR*. OpenReview.net.

Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo P. Mandic, Wenwu Wang, and Mark D. Plumbley. 2023a. Audioldm: Text-to-audio generation with latent diffusion models. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pages 21450–21474. PMLR.

Haohe Liu, Qiao Tian, Yi Yuan, Xubo Liu, Xinhao Mei, Qiuqiang Kong, Yuping Wang, Wenwu Wang, Yuxuan Wang, and Mark D. Plumbley. 2023b. Audioldm 2: Learning holistic audio generation with self-supervised pretraining. *CoRR*, abs/2308.05734.

Steven R Livingstone and Frank A Russo. 2018. The ryerson audio-visual database of emotional speech and song (ravdess): A dynamic, multimodal set of facial and vocal expressions in north american english. *PloS one*, 13(5):e0196391.

Xinhao Mei, Chutong Meng, Haohe Liu, Qiuqiang Kong, Tom Ko, Chengqi Zhao, Mark D Plumbley, Yuexian Zou, and Wenwu Wang. 2023. Wavcaps: A chatgpt-assisted weakly-labelled audio captioning dataset for audio-language multimodal research. *arXiv preprint arXiv:2303.17395*.

Eliya Nachmani, Alon Levkovitch, Julian Salazar, Chulayuth Asawaroengchai, Soroosh Mariooryad, R. J. Skerry-Ryan, and Michelle Tadmor Ramanovich. 2023. Lms with a voice: Spoken language modeling beyond speech tokens. *CoRR*, abs/2305.15255.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE.

Douglas B. Paul and Janet M. Baker. 1992. The design for the wall street journal-based CSR corpus. In *ICSLP*, pages 899–902. ISCA.

M Kathleen Pichora-Fuller and Kate Dupuis. 2020. Toronto emotional speech set (tess). *Scholars Portal Dataverse*, 1:2020.

Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, Gautam Naik, Erik Cambria, and Rada Mihalcea. 2018. Meld: A multimodal multi-party dataset for emotion recognition in conversations. *arXiv preprint arXiv:1810.02508*.

Krishna C. Puvvada, Nithin Rao Koluguri, Kunal Dhawan, Jagadeesh Balam, and Boris Ginsburg. 2023. Discrete audio representation as an alternative to mel-spectrograms for speaker and speech recognition. *CoRR*, abs/2309.10922.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust speech recognition via large-scale weak supervision. *CoRR*, abs/2212.04356.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning*, pages 28492–28518. PMLR.

Mirco Ravanelli, Titouan Parcollet, Peter Plantinga, Aku Rouhe, Samuele Cornell, Loren Lugosch, Cem Subakan, Nauman Dawalatabad, Abdelwahab Heba, Jianyuan Zhong, Ju-Chieh Chou, Sung-Lin Yeh, Szu-Wei Fu, Chien-Feng Liao, Elena Rastorgueva, François Grondin, William Aris, Hwidong Na, Yan Gao, Renato De Mori, and Yoshua Bengio. 2021. Speechbrain: A general-purpose speech toolkit. *CoRR*, abs/2106.04624.

Paul K. Rubenstein, Chulayuth Asawaroengchai, Duc Dung Nguyen, Ankur Bapna, Zalán Borsos, Félix de Chaumont Quitry, Peter Chen, Dalia El Badawy, Wei Han, Eugene Kharitonov, Hannah Muckenhirn, Dirk Padfield, James Qin, Danny Rozenberg, Tara N. Sainath, Johan Schalkwyk, Matthew Sharifi, Michelle Tadmor Ramanovich, Marco Tagliasacchi, Alexandru Tudor, Mihajlo Velimirovic, Damien Vincent, Jiahui Yu, Yongqiang Wang, Vicky Zayats, Neil Zeghidour, Yu Zhang, Zhishuai Zhang, Lukas Zilka, and Christian Havnø Frank. 2023. Audiopalm: A large language model that can speak and listen. *CoRR*, abs/2306.12925.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving AI tasks with chatgpt and its friends in huggingface. *CoRR*, abs/2303.17580.

Marco Tagliasacchi, Yunpeng Li, Karolis Misiunas, and Dominik Roblek. 2020. Seanet: A multi-modal speech enhancement network. In *INTERSPEECH*, pages 1126–1130.

Changli Tang, Wenyi Yu, Guangzhi Sun, Xianzhao Chen, Tian Tan, Wei Li, Lu Lu, Zejun Ma, and Chao Zhang. 2023. SALMONN: towards generic hearing abilities for large language models. *CoRR*, abs/2310.13289.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

A Vasuki and PT Vanathi. 2006. A review of vector quantization techniques. *IEEE Potentials*, 25(4):39–47.

Apoorv Vyas, Bowen Shi, Matthew Le, Andros Tjandra, Yi-Chiao Wu, Baishan Guo, Jiemin Zhang, Xinyue Zhang, Robert Adkins, William Ngan, Jeff Wang, Ivan Cruz, Bapi Akula, Akinniyi Akinyemi, Brian Ellis, Rashel Moritz, Yael Yungster, Alice Rakotoarison, Liang Tan, Chris Summers, Carleigh Wood, Joshua Lane, Mary Williamson, and Wei-Ning Hsu. 2023. Audiobox: Unified audio generation with natural language prompts. *CoRR*, abs/2312.15821.

Changhan Wang, Anne Wu, and Juan Pino. 2020. Covost 2 and massively multilingual speech-to-text translation. *arXiv preprint arXiv:2007.10310*.

Chengyi Wang, Sanyuan Chen, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, Lei He, Sheng Zhao, and Furu Wei. 2023a. Neural codec language models are zero-shot text to speech synthesizers. *CoRR*, abs/2301.02111.

Tianrui Wang, Long Zhou, Ziqiang Zhang, Yu Wu, Shujie Liu, Yashesh Gaur, Zhuo Chen, Jinyu Li, and Furu Wei. 2023b. Viola: Unified codec language models for speech recognition, synthesis, and translation. *CoRR*, abs/2305.16107.

Xiangpeng Wei, Heng Yu, Yue Hu, Rongxiang Weng, Weihua Luo, and Rong Jin. 2022. Learning to generalize to more: Continuous semantic augmentation for neural machine translation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, ACL 2022*.

Haibin Wu, Kai-Wei Chang, Yuan-Kuei Wu, and Hung-yi Lee. 2023. Speechgen: Unlocking the generative power of speech language models with prompts. *CoRR*, abs/2306.02207.

Dongchao Yang, Jinchuan Tian, Xu Tan, Rongjie Huang, Songxiang Liu, Xuankai Chang, Jiatong Shi, Sheng Zhao, Jiang Bian, Xixin Wu, Zhou Zhao, Shinji Watanabe, and Helen Meng. 2023a. Uniaudio: An audio foundation model toward universal audio generation. *CoRR*, abs/2310.00704.

Dongchao Yang, Jianwei Yu, Helin Wang, Wen Wang, Chao Weng, Yuexian Zou, and Dong Yu. 2023b. Diffsound: Discrete diffusion model for text-to-sound generation. *IEEE ACM Trans. Audio Speech Lang. Process.*, 31:1720–1733.

Yifan Yang, Feiyu Shen, Chenpeng Du, Ziyang Ma, Kai Yu, Daniel Povey, and Xie Chen. 2023c. Towards universal speech discrete tokens: A case study for ASR and TTS. *CoRR*, abs/2309.07377.

Binbin Zhang, Hang Lv, Pengcheng Guo, Qijie Shao, Chao Yang, Lei Xie, Xin Xu, Hui Bu, Xiaoyu Chen, Chenchen Zeng, et al. 2022. Wenetspeech: A 10000+ hours multi-domain mandarin corpus for speech recognition. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6182–6186. IEEE.

Dong Zhang, Shimin Li, Xin Zhang, Jun Zhan, Pengyu Wang, Yaqian Zhou, and Xipeng Qiu. 2023a. Speechgpt: Empowering large language models with intrinsic cross-modal conversational abilities. *CoRR*, abs/2305.11000.

Ruiqing Zhang, Xiyang Wang, Chuanqiang Zhang, Zhongjun He, Hua Wu, Zhi Li, Haifeng Wang, Ying Chen, and Qinfei Li. 2021. Bstc: A large-scale chinese-english speech translation dataset. *arXiv preprint arXiv:2104.03575*.

Yu Zhang, Wei Han, James Qin, Yongqiang Wang, Ankur Bapna, Zhehuai Chen, Nanxin Chen, Bo Li, Vera Axelrod, Gary Wang, Zhong Meng, Ke Hu, Andrew Rosenberg, Rohit Prabhavalkar, Daniel S. Park, Parisa Haghani, Jason Riesa, Ginger Perng, Hagen Soltau, Trevor Strohman, Bhuvana Ramabhadran, Tara N. Sainath, Pedro J. Moreno, Chung-Cheng Chiu, Johan Schalkwyk, Françoise Beaufays, and Yonghui Wu. 2023b. Google USM: scaling automatic speech recognition beyond 100 languages. *CoRR*, abs/2303.01037.

Siqi Zheng, Luyao Cheng, Yafeng Chen, Hui Wang, and Qian Chen. 2023. 3d-speaker: A large-scale multi-device, multi-distance, and multi-dialect corpus for speech representation disentanglement. *arxiv preprint, 2306.15354*.

Xiaohuan Zhou, Jiaming Wang, Zeyu Cui, Shiliang Zhang, Zhijie Yan, Jingren Zhou, and Chang Zhou. 2022. Mmspeech: Multi-modal multi-task encoder-decoder pre-training for speech recognition. *arXiv preprint arXiv:2212.00500*.

Appendices

A Experimental Details

A.1 Basic Tasks

The following tasks are used in supervised multi-task learning of LauraGPT and also in evaluations:

Automatic speech recognition (ASR) is a vital task in the speech processing community. It focuses on transcribing speech into textual content.

Spoken language understanding (SLU) is a task of directly deriving high-level semantic meaning from audio input. It aims to identify the user's intent and the relevant entity slots that fill the intent. An intent is usually composed of a scenario type and an action type, while slots and fillers are key-value pairs that specify the details of the intent.

Speech-to-text translation (S2TT) is similar to machine translation, but it directly translates the source language speech into the target language text.

Speech emotion recognition (SER) categorizes the emotions in speech input. Compared to textual emotion recognition, speech signals convey additional information, including tone and speaking rate, which enhances emotion recognition.

Automated audio captioning (AAC) aims to generate a natural language sentence that describes the content of an audio clip.

Speech enhancement (SE) is an audio-to-audio task that aims to improve speech quality through noise suppression and dereverberation. In order to incorporate this task into a unified modeling framework, we reformulate the task as a classification problem using codec tokens.

Text-to-speech synthesis (TTS) can be considered as the inverse process of ASR, where it generates speech that matches the given text.

A.2 Training Datasets

To ensure reproducibility, all training data and test data for LauraGPT are publicly available datasets, with licenses of Apache 2.0, CC BY 4.0, CC0, non-commercial research and education use, etc. The training data for the basic tasks listed in Section 3.4 and defined in Appendix A.1 are prepared as follows.

For the ASR task, we utilize open-source Chinese datasets such as AISHELL-1 (Bu et al., 2017), AISHELL-2 (Du et al., 2018), WenetSpeech (Zhang et al., 2022), as well as open-source English datasets including LibriSpeech (Panayotov et al., 2015) and GigaSpeech (Chen et al., 2021a).

For the S2TT task, we employ the commonly used BSTC (Zhang et al., 2021) and CoVOST 2 (Wang et al., 2020) datasets. Due to the limited data volumes of BSTC and CoVoST 2, we further augment the training set by translating AISHELL-1 and AISHELL-2 datasets into English and translating LibriSpeech dataset into Chinese using a publicly available text translation model (Wei et al., 2022). Consequently, we obtain approximately 2,000 hours of supplementary data for Chinese-to-English and English-to-Chinese S2TT tasks. As a supplement of training data for S2TT, we also add the ParaCrawl v9 dataset (Kocmi et al., 2022), which consists of 14M parallel text sentences for Zh→En (Chinese-to-English) and En→Zh (English-to-Chinese) translations.

For the SER task, we collect corpora including MELD (Poria et al., 2018), IEMOCAP (Busso et al., 2008), RAVDESS (Livingstone and Russo, 2018), TESS (Pichora-Fuller and Dupuis, 2020), Crema-D (Cao et al., 2014), Emov-DB (Adigwe et al., 2018), and SAVEE (Jackson and Haq, 2014). These corpora are recorded in multi-modal formats, comprising audio or visual data. No other corpora are used for the SER task.

For the SLU task, we use the multi-domain Spoken Language Understanding Resource Package (SLURP) dataset (Bastianelli et al., 2020), which covers 18 scenarios.

For the AAC task, we use AudioCaps (Kim et al., 2019), WavCaps (Mei et al., 2023), and Clotho (Drossos et al., 2020) datasets.

For the SE task, pairs of noisy and clean speech are required for training. The clean utterances are extracted from the AISHELL-1, AISHELL-2, LibriSpeech, and WSJ datasets (Paul and Baker, 1992), while the noisy counterparts are generated by mixing the clean speech with noises from the FSD-50K dataset (Fonseca et al., 2022) at random signal-to-noise rates (SNR) ranging from 2 to 15. Besides the additional noises, we also simulate convolutional noises by convolving the clean speech data with room impulse responses (Ko et al., 2017). As a result, we obtain approximately 6000 hours of paired data for the SE task.

For the TTS task, we use the open-source LibriTTS and 3D-speaker datasets (Zheng et al., 2023). Further details of the training data for all tasks can be found in Table 5.

Note that for all the training and test datasets, our use of the data is consistent with their intended use. We use all data sets in the same ways as prior research works, hence we did not check whether the data that was used contains any information that names or uniquely identifies individual people or offensive content.

A.3 Evaluation Datasets and Metrics

Table 6 presents the evaluation datasets and evaluation metrics for various tasks. The metrics used in our experiments are described below:

- **CER** stands for Character Error Rate, a commonly used metric to evaluate the recognition performance of Chinese and English utterances. We also utilize CER to assess the content consistency in TTS task.
- **WER** stands for Word Error Rate, which considers entire words rather than individual characters. In our experiments, we use WER to evaluate ASR recognition performance, content consistency in TTS, and speech intelligibility in SE.
- **SECS**, which stands for Speaker Encoder Cosine Similarity, utilizes speaker embeddings extracted from a pre-trained speaker verification model [6] for both prompt and synthesized speech. The cosine similarity between the two embeddings is then employed to measure the speaker similarity between the prompt speech and the synthesized speech. Furthermore, the naturalness of the synthesized speech is evaluated using **MOSNet**, a non-intrusive score derived from a pre-trained neural network [7].
- **BLEU** represent the Bilingual Evaluation Understudy metric. BLEU is commonly used to assess the quality of machine-generated text by comparing it to reference translations. In our experiments, we use BLEU to evaluate S2TT.
- **PESQ** represents Perceptual Evaluation of Speech Quality, while **STOI** stands for Short-time Objective Intelligibility. Both metrics are widely used to assess speech enhancement. PESQ ranges from -0.5 to 4.5, whereas STOI is in the range of $[0, 1]$.
- **SPICE**, **CIDEr** and **SPIDEr** are metrics borrowed from the image captioning task and employed for AAC evaluation. SPICE stands for Semantic Propositional Image Caption Evaluation, CIDEr denotes Consensus-based Image Description Evaluation, and SPIDEr represents the average of SPICE and CIDEr.
- **WA**, **UA** and **WF1** stands for weighted accuracy, unweighted accuracy and the weighted F1 score. WA corresponds to the overall accuracy, UA corresponds to the average class-wise accuracy, and WF1 corresponds to the average class-wise F1 score.
- **ACC** measures the accuracy of predicting the intent. **SLU-F1** is a metric that balances Word-F1 and Char-F1, computed as the sum of the confusion matrices.

A.4 Details of Training and Inference

In all experiments, we optimize the model parameters through the following steps: (1) We initialize the Qwen backbone and the audio encoder with the pre-trained checkpoints. (2) We then perform multi-task finetuning.

Due to the significant variation in data volume across different tasks, the training process is conducted in three stages. In the first training stage, the model is fine-tuned on all tasks using the complete training data as shown in Table 5. The AdamW optimizer is utilized with a peak learning rate of 5×10^{-4} and 10K warmup steps. In the second stage, we further fine-tune the model on tasks that have small-scale datasets, including TTS, SE, AAC, SER, and SLU tasks. The AdamW optimizer is utilized with a peak learning rate of 2×10^{-4} and 10K warmup steps. In the third training stage, we fine-tune the model on all tasks using the complete training data again. The peak learning rate of the AdamW optimizer for the third stage is reduced by half as 1×10^{-4}, while the warmup step remains at 10K.

For the codec vocoder, we train the predictor on the training data of the TTS and SE tasks. We use the Adam optimizer with a peak learning rate of 0.001 and 25K warmup steps. The decoder of the codec vocoder is initialized with the pre-trained checkpoints[8] and kept frozen during the multi-task finetuning of LauraGPT.

As stated in Section 3, during the training stage, the input is converted into input embeddings by the audio encoder if the input is audio, or converted by the embedding matrix W if the input is text, while the output is converted into output embeddings by the same embedding matrix W for teacher-forcing. Meanwhile, this matrix W is also used to convert

[6] Code is available at https://huggingface.co/microsoft/wavlm-base-plus-sv

[7] Code is available at https://github.com/lochenchou/MOSNet

[8] https://funcodec.github.io

Table 5: Statistics of the training data for basic tasks in Section 3.4. Corpus$^{\times N}$ means that the training samples in this corpus are copied N times during training.

Task	Training Data	# Samples
ASR	AISHELL-1, AISHELL-2, WenetSpeech, LibriSpeech, GigaSpeech	24.2 M
SLU	SLURP$^{\times 10}$	1.2 M
S2TT	BSTC$^{\times 5}$, CoVOST 2$^{\times 2}$, AISHELL-1, AISHELL-2, LibriSpeech	2.2 M
SER	MELD$^{\times 10}$, IEMOCAP$^{\times 10}$, RAVDESS$^{\times 10}$, TESS$^{\times 10}$ Crema-D$^{\times 10}$, Emov-DB$^{\times 10}$, SAVEE$^{\times 10}$	0.3 M
AAC	Clotho$^{\times 10}$, AudioCaps$^{\times 10}$, WavCaps$^{\times 5}$	1.3 M
SE	AISHELL-1$^{\times 3}$, AISHELL-2$^{\times 3}$, LibriSpeech$^{\times 3}$, WSJ$^{\times 2}$, FSD-50K$^{\times 2}$, RIR	5.3 M
TTS	LibriTTS$^{\times 2}$, 3D-Speaker$^{\times 2}$, AISHELL-1$^{\times 2}$, AISHELL-2$^{\times 2}$, LibriSpeech$^{\times 2}$	5.0 M

Table 6: Evaluation datasets and metrics for different tasks. $\uparrow$ indicates that higher values of the metric are desirable, while $\downarrow$ implies the opposite.

Task	Evaluation Datasets	Evaluation Metrics
ASR	AISHELL-1 test, AISHELL-2 test-ios, Librispeech test-clean & test-other	CER $\downarrow$, WER $\downarrow$
SLU	SLURP test	ACC $\uparrow$, SLU-F1 $\uparrow$
S2TT	BSTC dev, En$\rightarrow$Zh subset of CoVOST2	BLEU $\uparrow$
SER	MELD test	WA $\uparrow$, UA $\uparrow$, WF1 $\uparrow$
AAC	Clotho eval	SPICE $\uparrow$, CIDEr $\uparrow$, SPIDEr $\uparrow$
SE	LibriSpeech test-clean, FSD50K, noise-92	PESQ $\uparrow$, STOI $\uparrow$, WER $\downarrow$
TTS	AISHELL-1 test, LibriTTS test-clean	CER $\downarrow$, WER $\downarrow$, SECS $\uparrow$, MOS $\uparrow$

the task-ID token into an embedding. Then, these embeddings are composed into an embedding sequence as [input embeddings, task-ID embedding, output embeddings], which is taken as the input of Qwen LLM. To train the model, a masked cross-entropy loss function is applied, as shown in Eq. 1. As described in Section 3, in addition to masking out the losses on inputs, the cross-entropy loss at the position of the task token is also masked out.

During the inference stage, the input is converted into input embeddings as done during the training stage. Then the corresponding task-ID embedding is added at the end of the input embedding sequence. Next, the Qwen LLM generates output tokens in an autoregressive manner until the "end of sequence" token is generated. Finally, for text-format output, the Qwen tokenizer is employed to convert tokens into final output, while for audio-format output, the codec vocoder is employed to convert tokens into raw waveforms.

A.5 Details of the SER Evaluation

During the training stage, emotion labels within different training corpora are unified into the following nine classes: anger, disgust, neutral, like, sadness, surprise, happiness, joy, and fear. At the test stage, we map the "like" and "happiness" emotion classes into the "joy" class to match the MELD test set. LauraGPT uses an autoregressive structure to generate emotion labels. Out-of-domain outputs are considered as classification errors, making the task harder. Both WavLM Base model and WavLM Large model utilize the weighted sum of multiple layers with learnable parameters as speech features, which are fed into a downstream network for classification.

B Comparison with Related Unified Audio-and-Text Models

Table 7 compares our LauraGPT against the most related works, which, similar to LauraGPT, are all multi-task unified audio-and-text models. Due to the drastic differences in experimental settings, datasets used and lack of open source codebase and checkpoints, it is difficult to conduct a fair comparison between LauraGPT and these most related multi-task unified audio-and-text models. Despite all these difficulties, below we provide the most relevant results for comparing LauraGPT and these related models.

Whisper (Radford et al., 2022) is solely studied on the ASR task in the original paper, hence we compare LauraGPT to Whisper only on the ASR task. As shown in Table 8, on the Chinese test sets AISHELL-1 test and AISHELL-2 test-ios, LauraGPT greatly outperforms Whisper by **-3.9** and **-2.3** absolute on CER with much smaller training data. On the English test sets Librispeech test-clean and test-other, LauraGPT performs worse than Whisper Large V2 as Whisper Large V2 uses much more English training data than LauraGPT.

SpeechT5 (Ao et al., 2022) is evaluated on ASR, TTS, S2TT, voice conversion (VC), SE, and speaker identification (SID). Since the training data of tasks other than ASR for SpeechT5 differs remarkably from those for LauraGPT, we compare LauraGPT against SpeechT5 only on ASR. For SpeechT5, the model is first pre-trained with large-scale unlabeled speech and text data. Then, it is finetuned on the Librispeech-960 corpus via the hybrid cross-entropy and CTC loss. As claimed in their paper, SpeechT5 achieves a WER of 7.3% on the Librispeech test-other subset without CTC and LM. Under a fair comparison, our LauraGPT achieves a comparable WER of 7.7%. **Note that different from SpeechT5, LauraGPT is directly trained on multi-task labeled datasets without benefiting from any self-supervised pre-training**.

VioLA (Wang et al., 2023b) is evaluated on ASR, S2TT, TTS and S2ST tasks. Considering the substantial differences in training data on tasks between VioLA and LauraGPT and lack of open-sourced VioLA codebase and models, it is difficult to fairly compare LauraGPT with VioLA. Among the tasks, direct comparison on ASR is also challenging since VioLA only conducts speech-to-phoneme recognition and reports Phoneme Error Rate (PER) rather than recognizing words/characters and reporting WER/CER as conducted by LauraGPT. According to their paper, ViOLA underperforms their in-house Attention-based Encoder-Decoder (AED) model (which we also have no access to) with relative 19.96% phoneme error rate (PER) degradation from 9.47% to 11.36% on Mandarin WenetSpeech dev set. Since higher PER always corresponds to much higher WER as a word comprises multiple phonemes, it would be safe to hypothesize that the relative degradation on WER from VioLA over AED is even greater. In contrast, compared with the Paraformer baseline, our LauraGPT achieves comparable CER on the Mandarin AISHELL-2 test-ios set and outperforms it on the English Librispeech test-other set, i.e., overall LauraGPT performs comparably to Paraformer. Note that Paraformer is a non-autoregressive AED model performing comparably to conventional auto-regressive AED model (Gao et al., 2022). Therefore, **through this chain of comparisons, we are confident to conclude that LauraGPT notably outperforms VioLA on ASR task**.

AudioPaLM (Rubenstein et al., 2023) is evaluated on ASR, S2TT and TTS tasks. Since the training and evaluation datasets for AudioPaLM and LauraGPT are disjoint, their performance results cannot be directly compared. In addition, the pre-trained model of AudioPaLM has not been released. Therefore, empirically comparing LauraGPT to AudioPaLM will require great effort and is not conducted in this work.

C More Analyses of Critical Design Choices

C.1 Effectiveness of Multi-task Finetuning on the SER task

Table 4 shows that for the SER task, on the MELD test set, the multi-task trained LauraGPT substantially outperforms the single-task model in terms of UA and WF1 metrics, while the WA result is slightly worse.

To further analyze the results of the SER task, we conduct a statistical analysis of the number of samples for each emotion class in both training and test sets of the MELD dataset, as well as their corresponding test accuracy. The results are shown in Table 10. Compared to the single-task model, the multi-task trained LauraGPT results in degradation in accuracy for classes with a larger number of

Table 7: Comparisons with the most related multi-task unified audio-and-text models. The table shows the tasks that each model is trained and evaluated on.

	SpeechT5	Whisper	VioLA	AudioPaLM	LauraGPT(Ours)
Date	2021.10	2022.12	2023.5	2023.6	2023.9
Organization	Microsoft	OpenAI	Microsoft	Google	Ours
Model Size	0.14B	1.5B	0.25B	8B	2.0B
Pair Data (hrs)	0.96K	680K	79K	48K	60K
Unsup. Pretrain	N/A	N/A	N/A	PaLM-2	Qwen-1.8B
Audio Input	Continuous	Continuous	Discrete	Discrete	Continuous
Audio Output	N/A	N/A	Discrete	Discrete	Discrete
Languages	EN	99	EN/CN	113	EN/CN
ASR	✓	✓	✓	✓	✓
S2TT	✓	✓	✓	✓	✓
TTS	✓	✗	✓	✓	✓
SE	✓	✗	✗	✗	✓
AAC	✗	✗	✗	✗	✓
SER	✗	✗	✗	✗	✓
SLU	✗	✗	✗	✗	✓

Table 8: Comparison of different models on the ASR task in terms of CER(%) $\downarrow$ for Chinese and WER(%) $\downarrow$ for English. Data size denotes the number of hours.

Model	Model Size	Data Size	AISHELL-1 test	AISHELL-2 test-ios	Librispeech test-clean	Librispeech test-other
Paraformer (CN)	0.2 B	60K	2.0	2.9	-	-
Paraformer (EN)	0.2 B	20K	-	-	3.5	8.2
Whisper Large V2	1.5 B	680K	5.7	5.5	2.5	4.9
LauraGPT (Ours)	1.8 B	22K	1.8	3.2	4.4	7.7

Table 9: Comparison of batch normalization (BN) and layer normalization (LN) on the SE task in terms of Loop Ratio (%), PESQ and STOI(%). $\uparrow$ indicates that higher values are desired, while $\downarrow$ implies the opposite.

Norm	Loop Ratio $\downarrow$	PESQ $\uparrow$	STOI $\uparrow$
BN	86.00	1.27	22.0
LN	4.60	2.97	88.0

training samples, while greatly improving the accuracy on classes with fewer training samples. This explains why WA decreases slightly from multi-task training while UA and WF1 show remarkable improvements. Note that **WF1 is the primary metric on the MELD dataset due to sample imbalance across different emotion classes (Chen et al., 2023b)**. That is, on the primary metric WF1, the multi-task trained LauraGPT greatly outperforms the single-task model. Furthermore, the accuracy of the *disgust* and *fear* classes from the single-task model is 0, which aligns with the fact that these two classes have the fewest training samples in the MELD dataset. Multi-task training not only remarkably improves the performance of emotion classes with low accuracy (*joy*, *sadness*, *surprise*), but also greatly improves the performance of classes that cannot be predicted with single-task training (*disgust*, *fear*).

C.2 Batch normalization versus layer normalization in audio encoder

In the original design, batch normalization is applied after the convolution module in the Conformer-based audio encoder. However, we discover that this choice leads to endless looping decoding due to inaccurate estimations of mean and variance, particularly for tasks with long sequence lengths. When the issue of endless looping decoding occurs, the model generates several fixed tokens repeatedly and cannot stop the generation until achieving a pre-defined maximum length. To address this issue, we replace batch normalization with layer normalization, which is more robust to various mini-batch sizes. We validate this design

Table 10: Accuracy on different emotion classes in the SER task from single-task finetuning and multi-task finetuning.

Model	anger	disgust	neutral	joy	sadness	surprise	fear
#Training Samples	1109	271	4710	1743	683	1205	268
#Testing Samples	345	68	1256	402	208	281	50
Single-task	0.396	0.000	0.875	0.119	0.029	0.128	0.000
LauraGPT	0.333	0.103	0.708	0.381	0.236	0.381	0.040

by focusing on the SE task, which generally has the longest sequence among all the included tasks. The results are shown in Table 9. BN means batch normalization while LN means layer normalization. To evaluate the occurring probability of endless loop decoding, we define the metric, "loop ratio", which represents the fraction of endless decoded cases among all test cases. The results indicate that batch normalization causes a significantly high loop ratio at the inference stage, leading to unacceptable PESQ and STOI scores. In contrast, **by replacing batch normalization with layer normalization, we observe a considerable reduction in the loop ratio to a very low level, thereby greatly improving the speech enhancement performance**. It should be noted that although the loop ratio of layer normalization is restricted, further research is still desired to explore more general normalization methods suitable for all audio-and-text tasks.

C.3 Impact of initialization from pre-trained models

In LauraGPT, both the GPT backbone and audio encoder are initialized with the weights of pre-trained checkpoints. We investigate how the initialization affects the performance of LauraGPT. The experimental results for the ASR, S2TT and SE tasks are presented in Table 11. From the results, we observe that the initialization has a significant impact on the performance of ASR and S2TT tasks, while its influence on the SE task is relatively limited. This suggests that the prior knowledge learned by the GPT backbone is crucial for text generation tasks, but less important for audio generation tasks. Consequently, we hypothesize that **a reasonable approach to enhance the quality of generated audios could be to pre-train the GPT backbone not only with text sequences but also with audio token sequences**.

D Supporting More Complex Tasks

As stated in Section 3.4, with its modular and flexible design, LauraGPT provides an extensible framework to support complex tasks. By breaking a task into sub-tasks among the basic tasks used in training and cascading the raw inputs and model outputs of sub-tasks, LauraGPT can perform more complex tasks than the basic tasks.

Similar to the speech-to-speech translation (S2ST) example, LauraGPT can perform more complex tasks by chaining together basic tasks as described above. Here are a few examples of other complex tasks that LauraGPT can support rather than doing them one by one.

Rich transcription We can extend LauraGPT to simultaneously transcribe audio into content, speaker information (speaker identification, etc), paralinguistic information (emotion, etc.) and high-level semantic information (intent, slots, etc.) by including different task IDs at the generation process. This approach could avoid error accumulation in a pipelined approach and is more efficient than performing these tasks individually.

Noise-robust ASR We can implement noise-robust ASR by chaining tasks and creating the following input sequence: [noisy speech embedding, <SE>, embedding of the enhanced speech, <ASR>]. Since SE and ASR are jointly trained for LauraGPT, LauraGPT could effectively exploit embeddings of the original noisy speech and enhanced speech for noise-robust ASR.

Table 11: Impact of initialization on the ASR, S2TT and SE tasks.

Task	Dataset	Metric	w/o init	LauraGPT
ASR	AISHELL-1 test	CER ↓	4.3	**1.8**
	AISHELL-2 test-ios	CER ↓	6.0	**3.2**
	LibriSpeech test-clean	WER ↓	8.3	**4.4**
	LibriSpeech test-other	WER ↓	17.6	**7.7**
S2TT	BSTC dev (Zh→En)	BLEU ↑	8.4	**17.8**
	CoVOST2 test set (En→Zh)	BLEU ↑	12.2	**38.5**
SE	Mixup of LibriSpeech test-clean, FSD50K and noise-92	PESQ ↑	2.88	**2.97**
		STOI ↑	85.3	**88.0**
		Loop Ratio ↓	6.00	**4.60**