2025 edition

Deep Learning for Music Analysis and Generation

Language & Music

(music captioning, text-audio joint embeddings, audio language models)

Yi-Hsuan Yang Ph.D.

yhyangtw@ntu.edu.tw

Reference: ISMIR 2024 Tutorial

https://mulab-mir.github.io/music-language-tutorial/

Connecting Music Audio and Natural Language

Seung Heon Doh, Ilaria Manco, Zachary Novack, JongWook Kim and Ke Chen

Timeline	Content
09:00 - 09:05	Chapter 1. Introduction / Motivation [SeungHeon]
09:05 - 09:40	Chapter 2. Overview of Language Models [Jong Wook]
09:40 - 10:30	Chapter 3. Music Description [Ilaria]
10:30 - 11:00	Coffee Break
11:00 - 11:40	Chapter 4. Text-to-Music Retrieval For Music Discovery [SeungHeon]
11:40 - 12:20	Chapter 5. Text-to-Music Generation for New Sound [Zachary & Ke]
12:20 - 12:30	Chapter 6. Conclusion

25th International Society for Music Information Retrieval (ISMIR) Conference, Tutorials

Outline

- **Basics**
- Music description (music captioning)
- Music retrieval (text-audio joint embedding)
- Audio language models

Limits of Supervised Music Classification Models

https://mulab-mir.github.io/music-language-tutorial/intro.html

- Fixed vocabulary
  - cannot handle unseen vocabulary/classes (out-of-vocabulary; OOV)

- Labels are represented through one-hot or multi-hot encoding
  - model cannot capture relationships between different labels
  - Examples
    - Song A: [1 0 0 0 0]
    - Song B: [0 1 0 0 0]
    - Song C: [0 0 1 0 0]

Limits of Supervised Music Classification Models

https://mulab-mir.github.io/music-language-tutorial/introduction/advantange.html

Music and Language

https://mulab-mir.github.io/music-language-tutorial/intro.html

Music Generation

https://mulab-mir.github.io/music-language-tutorial/intro.html

Language Representation: Text as Sequence of Tokens

https://mulab-mir.github.io/music-language-tutorial/lm/framework.html

Conditioning in Language Models

https://mulab-mir.github.io/music-language-tutorial/lm/framework.html

- Adaptive modulation
- Channel concatenation
- Prefix conditioning
- Cross attention

Advances in Language Models

https://mulab-mir.github.io/music-language-tutorial/lm/advances.html

Limitation	so it's difficult to ...	but we can ...	Solution
It's a generative model	use for discriminative tasks	use probes or fine-tune on each task	Transfer learning
It generates nonsense	use the model	use more data / bigger Transformers	Scaling
Now it's is too huge and slow	generate quickly and cheaply	distill a small model from a large one	Distillation
It doesn't behave as desired	use as a helpful assistant	fine-tune for human preference	Post-Training / RLHF
Its context is limited	work with larger search spaces	query external sources for more context	RAG / Tool Use / Function Calling
Multimodal data is huge	hard to use as condition	use an encoder, e.g. CLIP and Whisper	Multimodal Encoders
Outputs have to be discrete	sample continuous data	use a VQ-VAE and/or diffusion	Multimodal Decoders
Constant compute per token	make it think deeper when needed	make the model think step-by-step	Chain-of-Thought Reasoning

Outline

- Basics
- **Music description (music captioning)**
- Music retrieval (text-audio joint embedding)
- Audio language models

Music Description: Why?

https://mulab-mir.github.io/music-language-tutorial/lm/framework.html

- Annotate large collections of music for easier search, navigation and organization
- Generate human-readable summaries for people with hearing loss
- Automatically caption music sections in videos and films
- Produce educational resources
- Personalized music recommendation using natural language queries
- **Synthetic data generation** to support training of other models (e.g., text-to-music generation)

Types of Music Description Tasks

https://mulab-mir.github.io/music-language-tutorial/lm/framework.html

Music Description

https://mulab-mir.github.io/music-language-tutorial/intro.html

Task 1: Music Captioning

https://mulab-mir.github.io/music-language-tutorial/description/tasks.html

$$P(Y|a) = \prod_{t=1}^{L} P(y_t|y_1, y_2, \ldots, y_{t-1}, a).$$

- Basically a **conditional text generation** problem (not generating audio)
- Predict the next text token given the audio *a* and the preceding text

Fine-grained Captions

- Identify the music's temporal changes at key transition points and their musical functions
- Generate detailed descriptions for each music segment

Ref: Wu et al, “FUTGA: Towards fine-grained music understanding through temporally-enhanced generative augmentation,” NLP4MusA 2024

Task 2: Music Question Answering (QA)

https://mulab-mir.github.io/music-language-tutorial/description/tasks.html

- Again a **conditional text generation** problem
- Need a dataset of (music, question, answer) tuples

Multimodal Audio-Language Models: LLM for Musical Audio

- Also known as audio language models (ALMs)

Model	Audio encoder	LLM
MusiLingo [9]	MERT [40]	Vicuna 7B [41]
MuLLaMa [8]	MERT [40]	LLaMA-2 7B [42]
M2UGen [12]	MERT [40]	LLaMA-2 7B [42]
SALMONN [11]	BEATS [43] & $\text{Whisper}_{\text{large-v2}}$ [44]	Vicuna 7B [41]
Qwen-Audio [13]	$\text{Whisper}_{\text{large-v2}}$ [44]	Qwen 7B [45]

Table 2. Overview of models we evaluate in our study.

Ref: Weck et al, “MuChoMusic: Evaluating music understanding in multimodal audio-language models,” ISMIR 2024

LLM for Symbolic Music

Music Theory Exercise

Default: "You will see JSON-formatted instruction data followed by questions. Your responses should only indicate the selected option (using uppercase letters), without providing any analysis."

CoT: "You will see a JSON-formatted instruction data followed by questions. Your responses should include an analysis step by step. The returned JSON format is as follows: {"reason": "**Let's think step by step**", "answer": "A"} "

ICL: **CoT** + "Here is an example of a question and its answer:
Read the following questions from the four options (A, B, C and D) given in each question. Choose the best option. Which of the following is the name of the note in the example?",

"L:1/4 M:4/4 K:Cb, D,4 |]",
Options: {"A": "B-flat", "B": "D", "C": "B", "D": "D-flat"},
Answer: "D".

Tasks
Music theory exercise
Motif extraction (ME)
Musical form extraction (MFE)
Chord-conditioned generation (CCG)
Melody harmonization (MH)
Musical-form-and-motif-conditioned generation

Ref: Zhou et al, "Can LLMs "reason" in music? An evaluation of LLMs' capability of music understanding and generation," ISMIR 2024

Task 3: Conversational Music Description

https://mulab-mir.github.io/music-language-tutorial/description/tasks.html

- Dialogue-based instead one-off
- Responses are expected to be based on the entire dialogue history

Music Description: How?

https://mulab-mir.github.io/music-language-tutorial/description/models.html

- Encoder-decoder
- Multimodal LLM (ALM)

Approach 1: Encoder-Decoder

- Audio encoder
  - mel spectrogram as the input
  - Whisper-like model architecture
- Text decoder: GPT

Ref: Doh et al, "LP-MusicCaps: LLM-based pseudo music captioning," ISMIR 2023

LP-MusicCaps: LLM-Based Pseudo Music Captioning

https://github.com/seungheondoh/lp-music-caps

Ref: Doh et al, “LP-MusicCaps: LLM-based pseudo music captioning,” ISMIR 2023

LP-MusicCaps: LLM-Based Pseudo Music Captioning

	Dataset	# item	Duration (h)	C/A	Avg. Token
	General Audio Domain				
	AudioCaps [30]	51k	144.9	1	9.0±N/A
	LAION-Audio [22]	630k	4325.4	1-2	N/A
	WavCaps [18]	403k	7568.9	1	7.8±N/A
	Music Domain				
small, high-quality data	MusicCaps [12]	6k	15.3	1	48.9±17.3
	MuLaMCap* [19]	393k	1091.0	12	N/A
	LP-MusicCaps-MC	6k	15.3	4	44.9±21.3
	LP-MusicCaps-MTT	22k	180.3	4	24.8±13.6
large, varied-quality data	**LP-MusicCaps-MSD**	514k	4283.1	4	37.3±26.8

Table 3. Comparison of audio-caption pair datasets. C/A stands for the number of caption per audio. *Although we include MuLaMCap in the table for comparison, it is not publicly accessible.

Ref: Doh et al, "LP-MusicCaps: LLM-based pseudo music captioning," ISMIR 2023

LP-MusicCaps: LLM-Based Pseudo Music Captioning

https://github.com/seungheondoh/lp-music-caps

- LP-MusicCaps generates a caption every 10 seconds
- You can use another LLM to summarize them into a song-level description

Quick Start: Audio to Caption

```
cd lpmc/music_captioning
wget https://huggingface.co/seungheondoh/lp-music-caps/resolve/main/transfer.pth -O exp/transfer/lp_mus
python captioning.py --audio_path ../../dataset/samples/orchestra.wav
```

```
{'text': "This is a symphonic orchestra playing  a piece that's riveting, thrilling and exciting.
The peace would be suitable in a movie when something grand and impressive happens.
There are clarinets, tubas, trumpets and french horns being played. The brass instruments help create t
'time': '0:00-10:00'}

{'text': 'This is a classical music piece from a movie soundtrack.
There is a clarinet playing the main melody while a brass section and a flute are playing the melody.
The rhythmic background is provided by the acoustic drums. The atmosphere is epic and victorious.
This piece could be used in the soundtrack of a historical drama movie during the scenes of an army mar
'time': '10:00-20:00'}

{'text': 'This is a live performance of a classical music piece. There is a harp playing the melody whi
The atmosphere is epic. This piece could be used in the soundtrack of a historical drama movie during t
'time': '20:00-30:00'}
```

Approach 2: Multimodal LLM

- Adapted LLMs
  - adapt text-only LLMs so that they become multimodal
- Natively multimodal LLMs

Ref: Doh et al, "LP-MusicCaps: LLM-based pseudo music captioning," ISMIR 2023

Datasets for Music Descprition

https://mulab-mir.github.io/music-language-tutorial/description/datasets.html

- Among the datasets containing music captions, only three provide fully human-written descriptions
  - MusicCaps
  - Song Describer Dataset
  - YouTube8M-MusicTextClips

Dataset 1: The MusicCaps Dataset

https://www.kaggle.com/datasets/googleai/musiccaps

- Published in 2023 along with Google's MusicLM paper (the first text-to-music paper)
- **5.5k** music-text pairs, with rich text descriptions from **human experts**
  - Music clips are from **AudioSet** (the same as the "**AudioCaps**" dataset)
  - For each 10-second music clip, MusicCaps provides: (1) a free-text caption consisting of four sentences on average, describing the music and (2) a list of music aspects, describing genre, mood, tempo, singer voices, instrumentation, dissonances, rhythm, etc.
  - There is a *genre-balanced* split with 1k examples

Ref: Agostinelli et al, "MusicLM: Generating music from text," arXiv 2023

Dataset 1: The MusicCaps Dataset

Ref: Agostinelli et al, “MusicLM: Generating music from text,” arXiv 2023

Dataset 2: The Song Describer Dataset (SDD)

https://zenodo.org/records/10072001

- Crowdsourced (still human-written, but not experts)
- 1.1k natural language descriptions of 706 CC-licensed audio from Jamendo

Table 1: An overview of the SDD compared to other music-caption datasets, MusicCaps (MC) [2] and YT8M-MusicTextClips (MTC) [25]. * denotes the validated subset.

Dataset	Annotators	Text			Audio			
		Caption #	Avg length	Vocab size	Audio #	Length	Source	Public
MC	10	5,521	54.8 ± 18.7	6,144	5,521	10 sec	[11]	✗
MTC	-	4,169	16.9 ± 4.4	2,599	4,169	10 sec	[1]	✗
SDD*	114	746	18.2 ± 7.6	1,942	547	2 min	[4]	✓
SDD	142	1106	21.7 ± 12.4	2,859	706	2 min	[4]	✓

Ref: Manco et al, "The Song Describer Dataset: a corpus of audio captions for music-and-language evaluation," NLP4MusA 2023

Evaluation Metrics

https://mulab-mir.github.io/music-language-tutorial/description/evaluation.html

- Basically a text generation problem
- Match-based metrics
  - BLEU
  - METEOR
  - ROUGE
  - CIDEr
  - SPICE
  - BERT-Score

Sample Code

http://mulab-mir.github.io/music-language-tutorial/description/code.html

- MusicFM for music understanding (audio encoder)

```
musicfm_embeds = load_dataset("mulab-mir/lp-music-caps-magnatagatune-3k-musicfm-embedding"
```

- GPT-2 for generating captions (text decoder)

```
self.a2t_projection = nn.Sequential(
    nn.Linear(self.audio_embedding_dim, self.text_embedding_dim),
    nn.ReLU(),
    nn.Linear(self.text_embedding_dim, self.text_embedding_dim)
```

- A **mapping module** to project audio embeddings extracted through MusicFM to the input space of our text decoder. These are then passed to GPT2 as **a prefix**

```
def freeze_backbone_model(self):
    for param in self.text_model.parameters():
        param.requires_grad = False
    self.text_model.eval()
```

Using Music Captioning Models to Create Data to Train Text-to-Music Generation Models

- **Music captioning** (conditional text generation): *music* → *text*
- **Text-to-music generation** (conditional audio generation): *text* → *music*
- Both require {text, audio} data pairs
- Given an audio-only collection of training data, we can apply music captioning to create training data pairs

Using Music Captioning Models to Create Data to Train Text-to-Music Generation Models

https://github.com/fundwotsai2001/Text-to-music-dataset-preparation

- "we […] preprocess the data following this pipeline"
  - Resample the audio to 44.1 kHz and segmented into fixed-length clips
  - Employ the sound event detection capabilities of PANNs to exclude any audio samples containing vocals
  - Generate captions for each audio clip using **Qwen2-Audio-7B-Instruct** model

```
python slice_2_47s.py --input_folder ./mtg-full --output_folder ./mtg-full-47s

python ./panns_inference/filter_vocal.py --audio_folder ./mtg-full-47s --json_path ./filtered_vocal_all

python Qwen2audio_captioning.py --input_json ./filtered_vocal_all.json --output_json ./filtered_vocal_a
```

Ref: Tsai et al, "MuseControlLite: Multifunctional music generation with lightweight conditioners," ICML 2025

Outline

- Basics
- Music description (music captioning)
- **Music retrieval (text-audio joint embedding)**
- Audio language models

Music Retrieval: Why?

https://mulab-mir.github.io/music-language-tutorial/retrieval/intro.html

- The task of finding a set of music pieces that match a given query
- The output is music, but it's about "retrieving" existing pieces rather than "generating" new ones
- Need to connect text and music, though

Early Stage Retrieval Methods

https://mulab-mir.github.io/music-language-tutorial/retrieval/intro.html

- Supervised classification models
- Use tags to define an embedding space to perform search

- Multi-hot limits
- OOV

Music Retrieval: How

https://mulab-mir.github.io/music-language-tutorial/retrieval/models.html

- Learn a **text-audio joint embedding space**

What's the Benefit of Joint Embedding?

https://mulab-mir.github.io/music-language-tutorial/retrieval/models.html

- The key advantage of joint embedding is that it allows us to leverage the **embedding space of pre-trained language models as supervision**, rather than being limited to a fixed vocabulary

Music Retrieval: How

https://mulab-mir.github.io/music-language-tutorial/retrieval/models.html

- The key is the embedding space
- Application can be two directions
  - Text-to-music retrieval
  - Music-to-text retrieval

Evaluation Metrics

- Results are sorted, and we may consider up to top-K ranked items
- Information retrieval metrics
  - Mean average precision (MAP@K)
  - Mean reciprocal rank (MRR)

MuLan: Text-Music Joint Embedding Learning

- MuLan takes the form of a two-tower, joint audio-text embedding model trained using **44 million music recordings** (370K hours) and weakly-associated, free-form text annotations
  - **Audio encoder**: Resnet-50, or audio spectrogram Transformer
  - **Text encoder**: BERT

- However, **not** open source…

Ref: Huang et al, "MuLan: A joint embedding of music audio and natural language," ISMIR 2022

MuLan: Text-Music Joint Embedding Learning

- MuLan takes the form of a two-tower, joint audio-text embedding model trained using **44 million music recordings** (370K hours) and weakly-associated, free-form text annotations
  - Data cleansing
    - Fine-tune a pre-trained BERT with a binary classification task on a small curated set of 700 sentences, which are manually labeled to be music-descriptive or not
    - However, found that training is surprisingly robust to annotation noise, achieving similar performance using *unfiltered* training text

Table 1. Text annotation examples.

Type	Examples
Short-form (SF)	tags like genre, mood, instrument, artist name, song title, album name
Long-form (LF)	'Hip-hop features rap with an electronic backing.' 'The melody is so nostalgic and unforgettable.'
Playlist (PL)	'Feel-good mandopop indie', 'Latin workout' 'Salsa for broken hearts', 'Piano for study'

Ref: Huang et al, "MuLan: A joint embedding of music audio and natural language," ISMIR 2022

Microsoft's CLAP: Text-Audio Joint Embedding Learning

Ref: Elizalde et al, "CLAP: Learning audio concepts from natural language supervision," ICASSP 2023

Microsoft's CLAP: Text-Audio Joint Embedding Learning

- 128,010 audio/text pairs from 4 datasets (FSD50k, ClothoV2, AudioCaps, MACS)
  - Each audio clip is randomly truncated to 5 secs
- Audio encoder: **PANNs-CNN14**
  - 80.8 million parameters & an embedding size of 2048; pretrained on AudioSet
- Text encoder: **BERT**
  - 110 million parameters
- Use 8-24 V100 GPUs (16GB VRAM)
- **Batch size** is important (as large as 256)

Ref: Elizalde et al, "CLAP: Learning audio concepts from natural language supervision," ICASSP 2023

MILA+UCSD+LAION's CLAP: Text-Audio Joint Embedding Learning

Dataset	Pairs	Audio Durations (hrs)
Clotho [15]	5,929	37.00
SoundDescs [16]	32,979	1060.40
AudioCaps [17]	52,904	144.94
LAION-Audio-630K	633,526	4325.39

Model	AudioCaps (mAP@10)	
	A→T	T→A
PANN+CLIP Trans.	4.7	11.7
PANN+BERT	34.3	44.3
PANN+RoBERTa	37.5	45.3
HTSAT+CLIP Trans.	2.4	6.0
HTSAT+BERT	43.7	49.2
HTSAT+RoBERTa	**45.7**	**51.3**

Ref: Wu et al, "Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation," ICASSP 2023

MILA+UCSD+LAION's CLAP: Text-Audio Joint Embedding Learning

- More than 633,526 audio/text pairs
- Use 10-second audio segments
- Tokenize the text with a maximum token length of 77
- Very large **batch size**
  - Train the model using a batch size of 768 on AudioCaps+Clotho dataset, 2,304 on training dataset containing LAION-Audio-630K, and 4,608 on training dataset containing AudioSet

Ref: Wu et al, "Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation," ICASSP 2023

CLAP

https://github.com/microsoft/CLAP

CLAP weights are downloaded automatically (choose between versions *2022*, *2023*, and *clapcap*), but are also available at: Zenodo or HuggingFace

clapcap is the audio captioning model that uses the 2023 encoders.

https://github.com/LAION-AI/CLAP

- For general audio less than 10-sec: 630k-audioset-best.pt or 630k-best.pt
- For general audio with variable-length: 630k-audioset-fusion-best.pt or 630k-fusion-best.pt
- For music: music_audioset_epoch_15_esc_90.14.pt
- For music and speech: music_speech_epoch_15_esc_89.25.pt
- For speech, music and general audio: music_speech_audioset_epoch_15_esc_89.98.pt

Variants of CLAP

- M2D-CLAP

Ref: Niizumi et al, "M2D-CLAP: Masked modeling duo meets clap for learning general-purpose audio-language representation," INTERSPEECH 2024

- FLAP

Ref: Yeh et al, "FLAP: Fast language-audio pre-training," ASRU 2023

MusicCoCa

https://github.com/magenta/magenta-realtime

- Updated and open-sourced version of Google's MuLan model
  - Use contrastive captioner (CoCa)
  - Two embedding towers, mapping each modality to a shared 768-dim space
  - **Audio embedding tower**: a 12-layer Vision Transformer (ViT)
    - Input: log-mel spectrogram of a 10s slice of 16kHz audio
    - 128 mel channels and temporal length 992; split into patches of size 16×16
  - **Text embedding tower**: a 12-layer Transformer
    - operates on tokenized text with a maximum sequence length of 128 tokens

Ref: Caillon et al, "Live music models," arXiv 2025

Datasets

Dataset	Hours	Type	Source
Clotho	152	Caption	Drossos et al.
AudioCaps	109	Caption	Kim et al.
MACS	100	Caption	Martín-Morató & Mesaros
WavText5Ks	25	Caption	Deshmukh et al. (2022)
BBC sound effects	481	Caption	https://sound-effects.bbcrewind.co.uk/
Audiostock	43	Caption	https://audiostock.net/se
Filter AudioSet	2084	Label	Gemmeke et al.
ESC-50	3	Label	Piczak
FSD50K	108	Label	https://annotator.freesound.org/fsd/
Sonniss Game Effects	20	Label	https://sonniss.com/gameaudiogdc/
WeSoundEffects	11	Label	https://wesoundeffects.com/
Epidemic Sound	220	Label	https://www.epidemicsound.com/
UrbanSound8K	8	Label	Salamon et al.
LibriTTS	300	Language-free	Zen et al. (2019)
LP-MusicCaps-MSD	4283	Caption	Doh et al. (2023)
LP-MusicCaps-MC	15	Caption	Doh et al. (2023)

Ref: “HarmonyLM: Advancing unified large-scale language modeling for audio and music generation,” arXiv 2024

Text-Audio Joint Embeddings for Other Tasks

- Not only for music retrieval tasks
- Can also be used as an **input condition** in text-to-music generation models
- Can also be used as an **evaluation metric** (for text/audio alignment)

	$\text{CLAP}_{score} \uparrow$
Training data (upper bound)	-
Autoencoded training data	-
Stable Audio w/ $\text{CLAP}_{\text{ours}}$	<u>0.44</u>
Stable Audio w/ $\text{CLAP}_{\text{LAION}}$	0.43
Stable Audio w/ T5	0.41
AudioLDM2-music	0.30
AudioLDM2-large	0.30
AudioLDM2-48kHz	0.22
MusicGen-small	0.33
MusicGen-large	0.36
MusicGen-large-stereo	0.32
Stable Audio	**0.46**

Sample Code

https://mulab-mir.github.io/music-language-tutorial/retrieval/code.html

```python
class JointEmbeddingModel(torch.nn.Module):
    def __init__(self, joint_dim=128, temperature=0.07):
        super().__init__()
        self.joint_dim = joint_dim
        self.temperature = temperature
        # Add projection part
        self.init_temperature = torch.tensor([np.log(1/temperature)])
        self.logit_scale = nn.Parameter(self.init_temperature, requir
        self.text_embedding_dim = 768 # roberta dim
        self.audio_embedding_dim = 1024 # music Fm dim
        self.audio_projection = nn.Sequential(
            nn.Linear(self.audio_embedding_dim, self.joint_dim, bias=
            nn.ReLU(),
            nn.Linear(self.joint_dim, self.joint_dim, bias=False)
        )
        self.text_projection = nn.Sequential(
            nn.Linear(self.text_embedding_dim, self.joint_dim, bias=False),
            nn.ReLU(),
            nn.Linear(self.joint_dim, self.joint_dim, bias=False)
        )
```

```python
def simple_contrastive_loss(self, z1, z2):
    z1 = nn.functional.normalize(z1, dim=1)
    z2 = nn.functional.normalize(z2, dim=1)
    temperature = torch.clamp(self.logit_scale.exp(), max=100)
    logits = torch.einsum('nc,mc->nm', [z1, z2]) * temperature.
    N = logits.shape[0]  # batch size per GPU
    labels = torch.arange(N, dtype=torch.long, device=self.devi
    return torch.nn.functional.cross_entropy(logits, labels)

def forward(self, batch):
    z_audio = self.audio_forward(batch['h_audio'].to(self.devic
    z_text = self.text_forward(batch['h_text'].to(self.device))
    loss_a2t = self.simple_contrastive_loss(z_audio, z_text)
    loss_t2a = self.simple_contrastive_loss(z_text, z_audio)
    loss = (loss_a2t + loss_t2a) / 2
    return loss
```

```python
def audio_forward(self, h_audio):
    z_audio = self.audio_projection(h_audio)
    return z_audio

def text_forward(self, h_text):
    z_text = self.text_projection(h_text)
    return z_text
```

Difference Between Description and Retrieval

- Audio encoder + text encoder + project layers
- *Cascade* vs *parallel*

Conversational Music Retrieval

https://mulab-mir.github.io/music-language-tutorial/intro.html

Outline

- Basics
- Music description (music captioning)
- Music retrieval (text-audio joint embedding)
- **Audio language models**

Audio Language Models (ALMs)

- ALMs or LALMs (large audio langauge models)
- Adapted LLMs
  - adapt text-only LLMs so that they become multimodal by augmenting them with additional modelling components
- Reference: visual language models (https://lilianweng.github.io/posts/2022-06-09-vlm/)

- Questions
  - How do they represent audio?
  - Are they expensive to train?

Example LALMs

- LLark
- SALMONN
- Qwen2-Audio
- Audio Flamingo 3

LLark: Overview

Ref: Gardner et al, “LLARK: a multimodal instruction-following language model for music,” ICML 2024

LLark: Core Tasks

- **Understanding**: identify a single global property of a piece of music, such as tempo, key, genre, and instrumentation
- **Captioning**: summarize the content of a piece of audio in language
- **Reasoning**: tasks that require either (1) combining knowledge of multiple aspects of a track or (2) reasoning about how aspects of this track combine to external knowledge about the world
  - This can include reasoning about how instruments and playing techniques demonstrate the Baroque composition style, or identifying what aspects of a track make it appropriate for certain settings (e.g. dinner party, studying, dance club)

LLark: Data

- Develop an end-to-end procedure for transforming diverse, noisy, unaligned music data into a **unified** instruction-following format in three task categories, augmenting the data with musical annotations.
- Use a set of only publicly-available, open source music datasets
- 164,000 distinct tracks, 1.2M instruction pairs

Dataset	Tracks	Task Families
MusicCaps (Agostinelli et al., 2023)	2,663	💬
YouTube8M-MusicTextClips (McKee et al., 2023)	4,169	💬
MusicNet (Thickstun et al., 2017)	323	💬 𝄞 🧠
FMA (Defferrard et al., 2017)	84,353	𝄞 🧠
MTG-Jamendo (Bogdanov et al., 2019)	55,609	𝄞 🧠
MagnaTagATune (Law et al., 2009)	16,761	𝄞 🧠

LLark: Data Pipeline

Ref: Gardner et al, "LLARK: a multimodal instruction-following language model for music," ICML 2024

Instruction Finetuning

- You start with a pretrained LLM (trained with next token prediction on huge corpora)
- Then **fine-tune** it on datasets consisting of **(instruction → response)** pairs
- These instructions can be natural language prompts (e.g., "Summarize this article") and the target is a human-written or curated response
- Examples:
  - General **chatbots**
  - Domain-expert assistants
  - Task-specialized models (summarizers, translators, classifiers)
  - Multimodal **Q&A** or captioning models
  - Agent-style models that can follow tool-related instructions

Ref: Zhang et al, "Instruction tuning for large language models: a survey," arXiv 2025

Metadata Augmentation

1. Metadata Augmentation: Many music datasets lack important musical information that is useful for music understanding, and can be estimated directly from the audio. In this step, we extract a set of features from the raw audio files using pretrained models.

We extract four features: tempo (in beats per minute, or BPM), global key and mode (e.g. 'F# minor'), timestamped chords, and beat grid (timestamped markers of where downbeats occur, along with numeric indicators of the beat "number", e.g. 1, 2, 3, 4 for a song in 4/4 time). For all features, we use open-source estimators via Böck et al. (2016).

Ref: Gardner et al, "LLARK: a multimodal instruction-following language model for music," ICML 2024

Instruction-Tuning Generation via Language Model

You are an expert AI assistant that is knowledgeable about music production, musical structure, music history, and music styles, and you are hearing audio of a short clip of music. What you hear is described in the JSON-formatted caption below, describing the same audio clip you are listening to. Answer all questions as if you are hearing the audio clip. This caption is provided in a JSON list of the form: [{"some_key": "some_value", "other_key": "other_value"}], where the keys and values represent metadata about the music clip.

The JSON may contain the following fields:

'album.information': optional user-provided information about the album.
'album.tags': optional user-provided tags associated with the track album.
'artist.tags': optional user-provided tags associated with the track artist.
'track.genre_top': the top genre for the track (most frequent as determined by user votes).
'track.genres_all': all genre labels for the track.
'track.information': optional user-provided information about the track.
'track.language_code': the language of the track.
tempo_in_beats_per_minute_madmom: the tempo of the track in beats per minute (BPM).
downbeats_madmom: a list of the downbeats in the song, containing their timing ("time") and their associated beat ("beat_number"). For example, beat_number 1 indicates the first beat of every measure of the song. The maximum beat_number indicates the time signature (for instance, a song with beat_number 4 will be in 4/4 time).
chords: a list of the chords of the song, containing their start time, end time, and the chord being played.
key: the key of the song.

Design a conversation between you and a person asking about this music. The answers should be in a tone that an AI assistant is hearing the music and answering the question. Ask diverse questions and give corresponding answers.
Ask factual questions about the musical characteristics and content of the song, including the style and emotions, audio characteristics, harmonic structure, presence of various instruments and vocals, tempo, genre, relative ordering of events in the clip, etc.

Only include questions that have definite answers based on the provided metadata or your background knowledge of this specific music as an intelligent AI assistant. Write as many question as you can using the provided inputs. Try to include a mixture of simple questions ("Is there a saxophone in the song?" "Are there vocals in the clip?" "What is the approximate tempo of the clip in beats per minute (BPM)?")) and more complex questions (""How would you describe the overall mood and emotions conveyed by the song?"). Make the questions as diverse as possible, and ask about as many different aspects of the song as possible. Do not mention the name of the artist in the response.

Again, do not ask about uncertain details. Provide detailed answers when answering complex questions. For example, give detailed examples or reasoning steps to make the content more convincing and well-organized. Explain any musical concepts that would be unfamiliar to a non-musician. You can include multiple paragraphs if necessary. Make sure that the generated questions contain questions

Instruction Data Filtering

- The language model sometimes did not follow the prompt
- Filtering the QA pairs was important to improve the data qualit

Table 9. Keywords and phrases used to filter questions and answers after instruction data generation. Any query-response pairs wher the query or response contained a disallowed phrase from the respective list was excluded.

Query Keywords	Response Keywords
”what is the composer”, ”who is the composer”, ”tell me about the composer”, ”name of the composer”, ”who is the artist”, ”tell me about the artist”, ”what tags are associated with the artist”, ”what are the tags associated with the artist”, ”is there any information available about the album”, ”about the album”, ”name of the artist”, ”what is the name”, ”what is the movement”, ”what is the specific movement”, ”what is the title”, ”which movement is”, ”what is the length of this clip”, ”duration”, ”pack”,	”metadata”, ”is not provided”, ”based on the provided metadata”, ”based on the provided beat”, ”based on the provided chord”, ”based on the provided information”, ”based on the provided annotations”, ”no specific mood”, ”there is no mention of”, ”there is no specific mention of any”, ”As an AI assistant, I am unable to”, ”As an AI assistant, I do not”, ”it is difficult to determine”, ”it is not possible to determine”, ”no information is available about the album”, ”cannot determine”, ”violin 1”, ”violin 2”, ”violin 3”, ”viola 1”, ”viola 2”, ”viola 3”, ”pack”

Ref: Gardner et al, “LLARK: a multimodal instruction-following language model for music,” ICML 2024

LLark: Model

- **12B** parameters, with 3 modules
- **Language model**: *Llama2-7b-chat* (which can already do chatting)
- **Audio encoder**: *Jukebox-5B*
  - process 25-second audio clips
  - output of the 36th layer of the Jukebox encoder
  - 4800-dim **vectors** at originally a frequency of 345Hz, mean-pooled to 10Hz
- **Projection**: a linear projection layer (following LLaVA)
  - adding fewer than 0.1% additional parameters relative to the base models

LLark: Training Details

- The model is trained on 4 NVIDIA A100 GPUs with 80GB VRAM
- The model is trained for 100k steps with a global batch size of 32
- Training takes approximately 54 hours

LLark: Summary

- An early and representative ALM designed specifically for music
- Propose a pipeline to generate data for instruction finetuning

- Provide code (https://github.com/spotify-research/llark) but ***NOT*** the model checkpoint …

SALMONN

https://github.com/bytedance/SALMONN

- **Open** source code, model checkpoints and data
- **Not just for music** (also for speech and general sounds)

- Language model: *VicunaLLM-13b* (which can already do chatting, too)
- Dual audio encoders
  - *Whisper-Large-v2* (50 hz frame rate)
  - *BEATs* (50 hz frame rate)
- Module to bridge text and audio: "**Q-former**"
- Only train the Q-Former and LoRA (33M in total)

Ref: Tang et al, "SALMONN: Towards generic hearing abilities for large language models," ICLR 2024

SALMONN: Training Data

Task	Data Source	#Hours	#Samples
ASR	LibriSpeech + GigaSpeech	960 + 220	280K + 200K
En2Zh	CoVoST2-En2Zh (Wang et al., 2021)	430	290K
AAC	AudioCaps + Clotho	130 + 24	48K + 4K
PR	LibriSpeech	960	280K
ER	IEMOCAP Session 1-4 (Busso et al., 2008)	5	4K
MC	MusicCaps (Agostinelli et al., 2023)	14	3K
OSR	LibriMix (Cosentino et al., 2020)	260	64K
SV	VoxCeleb1 (Nagrani et al., 2019)	1200	520K
GR	LibriSpeech	100	28K
SQA	LibriSpeech	960	280K
AQA	WavCaps + AudioCaps	760 + 130	270K + 48K
MQA	MillionSong[1] + MusicNet (Thickstun et al., 2017)	400 + 3	48K + 0.3K
Total	–	~4400	~2.3M

Ref: Tang et al, “SALMONN: Towards generic hearing abilities for large language models,” ICLR 2024

SALMONN: Model Architecture

Q-Former

- *LLark* (only for music): use a single linear projection layer
- *SALMONN* (general LALM): use Q-Former, a more complicated method to bridge text and audio
- **Q-Former**
  - From the BLIP-2 paper (ICML 2023)
  - Use **learnable queries** + self/cross **attention layers**
  - SALMONN adapts it so that it can deal with variable-length audio of up to 30 seconds

Qwen2-Audio

https://github.com/QwenLM/Qwen2-Audio

- Language model: *Qwen-7B*
- Audio encoder: *Whisperlarge-v3*
- Total parameters: 8.2B
- DPO (direct preference optimization) is used to learn from human preference data

Ref: Chu et al, "Qwen2-Audio technical report," arXiv 2024

Audio Flamingo 3

https://github.com/NVIDIA/audio-flamingo

Ref: Goel et al, "Audio Flamingo 3: Advancing audio intelligence with fully open large audio language models," arXiv 2025

Audio Flamingo 3

- Language model: *Qwen-2.5-7B*
- Audio encoder*: AF-Whisper* (adapted from *Whisper large-v3*)
- Module to bridge text and audio: and “audio adaptor”

Ref: Goel et al, “Audio Flamingo 3: Advancing audio intelligence with fully open large audio language models,” arXiv 2025

Audio Flamingo 3

Models	Audio Understanding			Voice		Multi-turn Chat		Long Audio (>30 secs)			Open-Source		
	Sound	Music	Speech	In	Out*	Single A	Multiple A	Speech	Sound	Music	Model	Data	Code
LTU	✓	✓	×	×	×	×	×	×	×	×	✓	✓	✓
LTU-AS	✓	✓	✓	×	×	×	×	×	×	×	✓	✓	✓
GAMA	✓	✓	×	×	×	×	×	×	×	×	✓	✓	✓
SALMONN	✓	✓	✓	×	×	×	×	×	×	×	✓	✓	✓
MuLLaMa	×	✓	×	×	×	×	×	×	×	×	✓	✓	✓
Phi-4-mm	✓	✓	✓	×	×	×	×	✓	✓	✓	✓	×	×
Qwen-Audio	✓	✓	✓	✓	✓	✓	×	×	×	×	✓	×	×
Qwen2-Audio	✓	✓	✓	✓	✓	✓	×	×	×	×	✓	×	×
Qwen2.5-Omni	✓	✓	✓	✓	✓	✓	×	✓	✓	✓	✓	×	×
GPT-4o Audio	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	×	×
Gemini 2.0 / 2.5	✓	✓	✓	✓	✓	✓	×	✓	✓	✓	×	×	×
Audio Flamingo	✓	✓	×	×	×	✓	×	×	×	×	✓	✓	✓
Audio Flamingo 2	✓	✓	×	×	×	×	×	×	✓	✓	✓	✓	✓
Audio Flamingo 3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Ref: Goel et al, “Audio Flamingo 3: Advancing audio intelligence with fully open large audio language models,” arXiv 2025

LALM: Comparisons

	MU-LLaMA	MusiLingo	LLARK	M2UGen
LLM	LLaMA-2 7B	Vicuna-7B	Llama2-7b	Llama2-7b
audio encoder	MERT	MERT	Jukebox	MERT
audio token frequency(Hz)	fixed embedding	0.17	10	fixed embedding
max train duration(s)	29	30	25	10
tune modules	adapter	adapter	projection, LLM	adapters, LLM(lora)

	Qwen2-Audio	Kimi-Audio	ours
LLM	Qwen-7B	Qwen2.5-7B	Qwen3-8B
audio encoder	whisper	whisper	whisper
audio token frequency(Hz)	25	12.5	10
max train duration(s)	30(likely)	30(likely)	390
tune modules	unknown	all	all

Ref: Jiang et al, “Advancing the foundation model for music understanding,” arXiv 2025

Dimension	LLark	SALMONN	Qwen2-audio	Audio Flamingo3
Music-specific or not	Yes	No	No	No
Total model size	12B parameters arxiv.org	~13B parameters arxiv.org	8.2B parameters arxiv.org	7B parameters arxiv.org
Audio encoder	Jukebox-5B	Whisper speech encoder + BEATs audio encoder	Whisper-large-v3	AF-Whisper (based on Whisper large-v3)
Pre-trained LLM model	Llama 2 (7B-chat)	Vicuna (13B)	Qwen-7B	Qwen-2.5-7B
Projection techniques to bridge audio and text	Simple linear projection	Window-level Q-Former	Multi-modal projector (simple linear)	Audio adaptor layers
Audio input length	25 seconds	30 seconds	30 seconds	10 minutes
Use open source data for fine-tuning or not	Yes	Yes	Yes	Yes
Shared checkpoint or not	No	Yes	Yes	Yes

(table generated by Grok)

MixAssist: An Example of Building A Customized LALM

Ref: Clemens & Marasović, "MixAssist: An audio-language dataset for co-creative ai assistance in music mixing," COLM 2025

Audio Language Models: Summary

- Prepare your instruction finetuing data
- Train your model
  - Language model
  - Audio encoder (e.g., MERT, BEATs)
  - Modules to connect audio and language

- It’s basically fine-tuning, so won’t be terribly compute-heavy
- Good quality data is critical