DEEPSHARKLAB · 정승원

Open-MusicLM
실험 및 결과 분석

C293242 정승원

Text-to-Music

MusicLM 구조를 오픈소스 구현으로 직접 실험

FMA Large

106,574 tracks · 93GB · 30초 MP3

Fine-tuning

semantic → coarse → fine 순차 학습

MusicLM & Open-MusicLM 개요

MusicLM

텍스트 설명을 음악으로 생성하는 Google text-to-music 모델

Open-MusicLM

MusicLM의 핵심 아이디어를 공개 모델 기반으로 재현한 구현체

- waveform을 직접 생성하지 않고 token sequence를 예측
- 여러 단계의 token을 만든 뒤 마지막에 오디오로 복원
- 공개 구현 덕분에 직접 전처리·학습·평가 가능

기존 MusicLM과 Open-MusicLM의 차이

🎵 MusicLM (Google, 2023)

- Text → Music 생성
- 3단계 계층적 autoregressive 구조
  Semantic → Coarse → Fine
- MuLan 임베딩으로 텍스트 조건부 생성
- 1B+ 파라미터 규모

⚠️ 코드 미공개

→

🔓 Open MusicLM (zhvng, 2023)

MuLan	→	**CLAP**
SoundStream	→	**Encodec**
w2v-BERT	→	**MERT**

- 오픈소스 파인튜닝
- Semantic / Coarse / Fine 3단계 유지
- FMA, MusicCaps 등 데이터셋 지원
- 총 ~870M 파라미터 (Large 기준)

핵심 구조는 유지하되, 실험 가능한 공개 모델 조합으로 대체

Open-MusicLM의 3단계 생성 구조

Text Prompt

CLAP 텍스트 인코더
(512-d 임베딩)

CLAP RVQ

텍스트 임베딩
양자화 (RVQ)

Semantic Transformer

MERT 토큰
자기회귀 생성
283M params

Coarse Transformer

Encodec coarse
코드북 예측
289M params

Fine Transformer

잔여 코드북
세부 표현
297M params

Encodec Decoder → 최종 음악 파형 출력 (24kHz)

실험 목표와 전체 흐름

FMA Large 데이터

106,574곡
93 GB MP3

CLAP RVQ 학습

1,000 steps
clap.rvq.990.pt

K-means (Hubert)

kmeans.joblib
Semantic 토큰

데이터 전처리 v2

pretrained RVQ
fma_preprocessed_v2

Semantic 학습

20,000 steps
best checkpoint

Coarse 학습

7,000 steps
best checkpoint

Fine 학습

10,500 steps
best checkpoint

모델 크기 Semantic 283M · Coarse 289M · Fine 297M → 합계 ~ 870M params

FMA Large 데이터셋

🎵
106,574
총 트랙 수

💿
93 GB
MP3 용량

🎸
161
장르 수

⏱️
30초
트랙 길이

데이터셋 버전 비교

버전	트랙 수	용량
Small	8,000	7.2 GB
Medium	25,000	22 GB
Large ✓	**106,574**	**93 GB**

왜 FMA Large를 선택했나?

- 오픈 라이선스 (CC) — 학습 적합
- 장르 다양성: Rock, Pop, Electronic 등 161개
- 메타데이터 제공 (장르, 아티스트, 트랙 정보)
- Open MusicLM 공식 학습 데이터셋

전처리가 필요한 이유

왜 전처리를 하나?

⚡ 학습 속도

매 스텝마다 MP3 디코딩은 너무 느림.
미리 토큰화해두면 I/O 부담 대폭 감소

🔢 토큰 통일

CLAP RVQ 코드 + MERT Semantic 토큰을
미리 추출해 일관된 형식으로 저장

💾 재사용성

한번 전처리하면 다양한 실험에
반복 재활용 가능

전처리 파이프라인 (4단계)

1 **CLAP RVQ 학습**

clap.rvq.990.pt · 1,000 steps

2 **Hubert K-means**

kmeans.joblib · semantic 토큰화 기반

3a **v1 전처리**

자체 RVQ 사용 — codebook 불일치 문제 발생

3b **v2 전처리 ✓**

pretrained RVQ 사용 — 최종 사용 버전

전처리 v1: 직접 학습한 RVQ 사용

RVQ v1

FMA 기반 직접 학습 · 약 990 step

전처리 완료

token 데이터 생성은 정상 완료

문제 발생

semantic loss가 약 6.932에서 고정

```
echo "n" | nohup python ./scripts/preprocess_data.py \
    --model_config ./configs/model/musiclm_small.json \
    --training_config ./configs/training/train_fma_preprocess.json \
    --rvq_path ./results/clap_rvq/clap.rvq.990.pt \
    --kmeans_path ./results/hubert_kmeans/kmeans.joblib \
    > ./logs/preprocess.log 2>&1 &
```

원인: 전처리에 사용한 RVQ와 pretrained checkpoint의 RVQ가 서로 다름

RVQ Codebook 불일치 문제

Pretrained RVQ

code 120 = pretrained가 학습한 의미

Custom RVQ

code 120 = 전혀 다른 의미일 수 있음

loss ≈ 6.932 ≈ ln(1024)

1024개 code 중 거의 랜덤으로 예측하는 수준과 비슷

Fine-tuning에서는 데이터뿐 아니라 token 체계 자체가 pretrained 기준과 일치해야 함

전처리 v2: Pretrained RVQ 사용

- 직접 학습한 RVQ 대신 checkpoint와 함께 제공된 RVQ 사용
- clap.rvq.950_no_fusion.pt
- kmeans_10s_no_fusion.joblib
- musiclm_large_small_context.json
- 최종 전처리 결과: fma_preprocessed_v2

```
echo "n" | nohup python ./scripts/preprocess_data.py \
    --model_config ./results/pretrained/musiclm_large_small_context.json \
    --training_config ./configs/training/train_fma_preprocess.json \
    --rvq_path ./results/pretrained/clap.rvq.950_no_fusion.pt \
    --kmeans_path ./results/pretrained/kmeans_10s_no_fusion.joblib \
    > ./logs/preprocess_v2.log 2>&1 &
```

v2 데이터가 semantic / coarse / fine fine-tuning에 사용된 최종 학습 데이터

Semantic / Coarse / Fine 파인튜닝

Semantic

텍스트 조건 + CLAP code → semantic token sequence

```
python ./scripts/train_semantic_stage.py \
    --results_folder ./results/semantic \ # where to save results and checkpoints
    --model_config ./configs/model/musiclm_small.json \
    --training_config ./configs/training/train_musiclm_fma.json \
    --rvq_path PATH_TO_RVQ_CHECKPOINT \ # path to previously trained rvq
    --kmeans_path PATH_TO_KMEANS_CHECKPOINT # path to previously trained kmeans
```

Coarse

semantic token → EnCodec coarse code

```
python ./scripts/train_coarse_stage.py \
    --results_folder ./results/coarse \ # where to save results and checkpoints
    --model_config ./configs/model/musiclm_small.json \
    --training_config ./configs/training/train_musiclm_fma.json \
    --rvq_path PATH_TO_RVQ_CHECKPOINT \ # path to previously trained rvq
    --kmeans_path PATH_TO_KMEANS_CHECKPOINT # path to previously trained kmeans
```

Fine

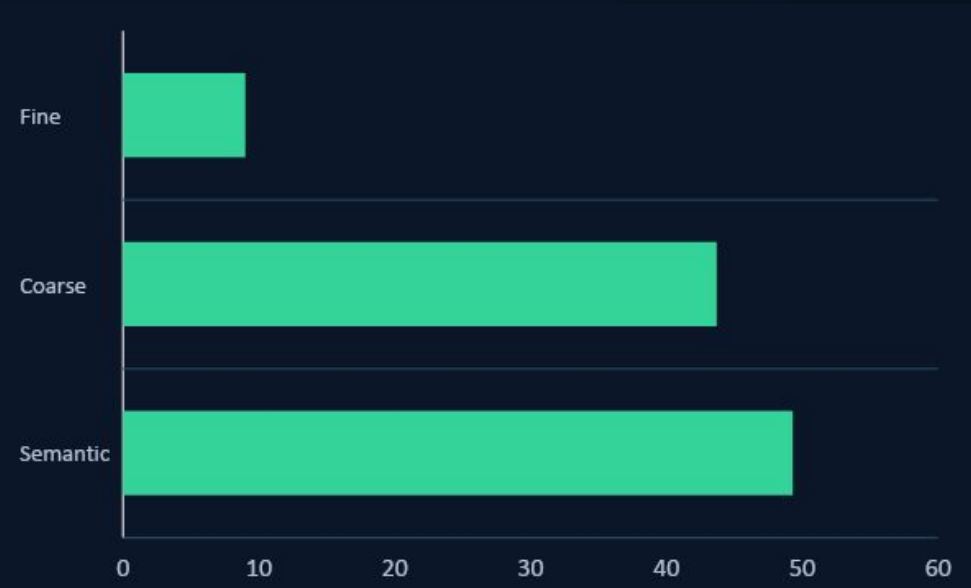
남은 codebook을 채워 음색과 질감 보완

```
python ./scripts/train_fine_stage.py \
    --results_folder ./results/fine \ # where to save results and checkpoints
    --model_config ./configs/model/musiclm_small.json \
    --training_config ./configs/training/train_musiclm_fma.json \
    --rvq_path PATH_TO_RVQ_CHECKPOINT \ # path to previously trained rvq
    --kmeans_path PATH_TO_KMEANS_CHECKPOINT # path to previously trained kmeans
```

OPEN-MUSICLM / FMA LARGE

학습 결과와 Checkpoint

Model	Step	Valid loss	Valid acc.
Semantic	20,000	2.224	49.3%
Coarse	7,000	2.676	43.7%
Fine	10,500	4.757	9.0%



semantic과 coarse는 안정적, fine은 세부 음질 단계에서 추가 개선 필요

생성 결과 비교 방법

같은 prompt 입력

pretrained 모델과 finetuned 모델에 동일 prompt 사용

8개 샘플 생성

각 prompt마다 후보 8개 생성

Best sample 선택

CLAP cosine similarity가 가장 높은 샘플 선택

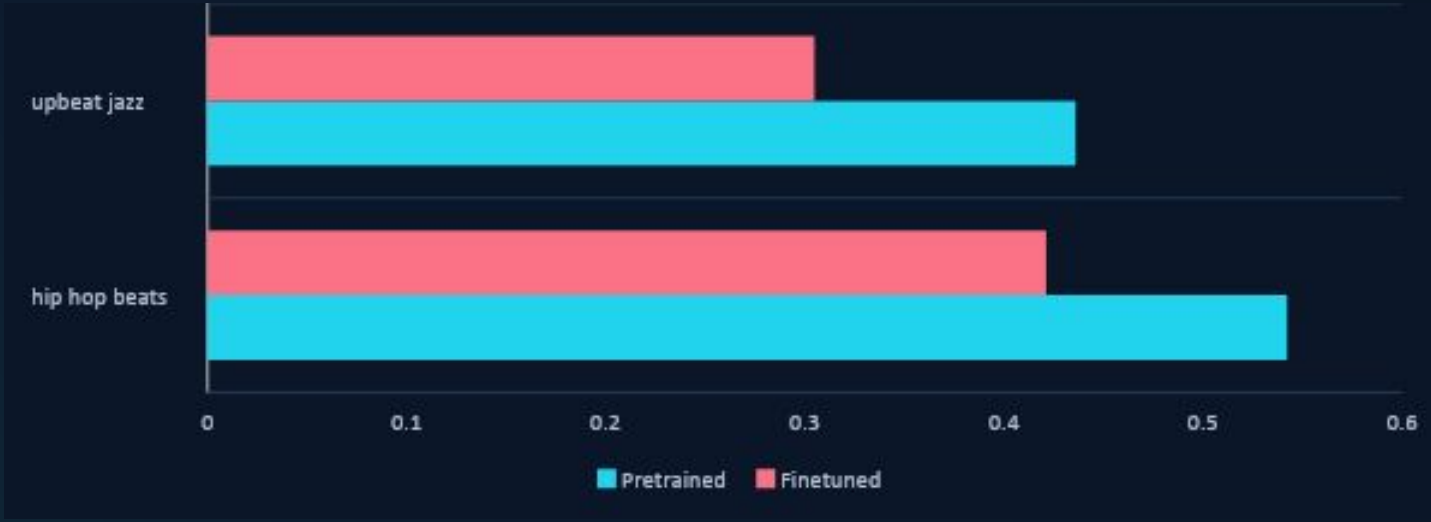
결과 비교

prompt alignment 관점에서 pretrained vs finetuned 비교

```
python scripts/infer_top_match.py \
    "프롬프트" \
    --num_samples 8 \
    --num_top_matches 1 \
    --semantic_path ./results/pretrained/semantic.transformer.14000.pt \
    --coarse_path ./results/pretrained/coarse.transformer.18000.pt \
    --fine_path ./results/pretrained/fine.transformer.24000.pt \
    --rvq_path ./results/pretrained/clap.rvq.950_no_fusion.pt \
    --kmeans_path ./results/pretrained/kmeans_10s_no_fusion.joblib \
    --model_config ./results/pretrained/musiclm_large_small_context.json \
    --duration 10
```

```
python scripts/infer_top_match.py \
    "프롬프트" \
    --num_samples 8 \
    --num_top_matches 1 \
    --semantic_path ./results/finetune_v2/semantic_v4/semantic.transformer.20000.pt \
    --coarse_path ./results/finetune_v2/coarse_v2/coarse.transformer.7000.pt \
    --fine_path ./results/finetune_v2/fine/fine.transformer.10500.pt \
    --rvq_path ./results/pretrained/clap.rvq.950_no_fusion.pt \
    --kmeans_path ./results/pretrained/kmeans_10s_no_fusion.joblib \
    --model_config ./results/pretrained/musiclm_large_small_context.json \
    --duration 10
```

Pretrained vs Finetuned 유사도 결과



7개 중 6개

prompt에서 finetuned similarity가 pretrained보다 낮음

예외: cinematic orchestral rock electric guitar 계열 prompt는 finetuned가 아주 조금 높음

1. FMA 데이터셋 특성

음악 파일은 많지만 자세한 자연어 설명이 약함

2. 학습 데이터 상이

기존의 범용 prompt 이해 능력이 일부 약해질 수 있음

3. Loss와 평가 지표 차이

token prediction loss 감소가 CLAP similarity 개선을 보장하지 않음

생성 음악 청취 비교

Prompt: power rock drum

Pretrained 생성 음악

Finetuned 생성 음악

CLAP similarity

텍스트 prompt와 생성 음악 embedding의 cosine similarity

```
python scripts/infer_top_match.py \
    "power rock drum" \
    --num_samples 8 \
    --num_top_matches 1 \
    --semantic_path ./results/finetune_v2/semantic_v4/semantic.transformer.20000.pt \
    --coarse_path ./results/finetune_v2/coarse_v2/coarse.transformer.7000.pt \
    --fine_path ./results/finetune_v2/fine/fine.transformer.10500.pt \
    --rvq_path ./results/pretrained/clap.rvq.950_no_fusion.pt \
    --kmeans_path ./results/pretrained/kmeans_10s_no_fusion.joblib \
    --model_config ./results/pretrained/musiclm_large_small_context.json \
    --duration 10
```

Thank You