Neural Audio Fingerprint for High-specific Audio Retrieval based on Contrastive Learning

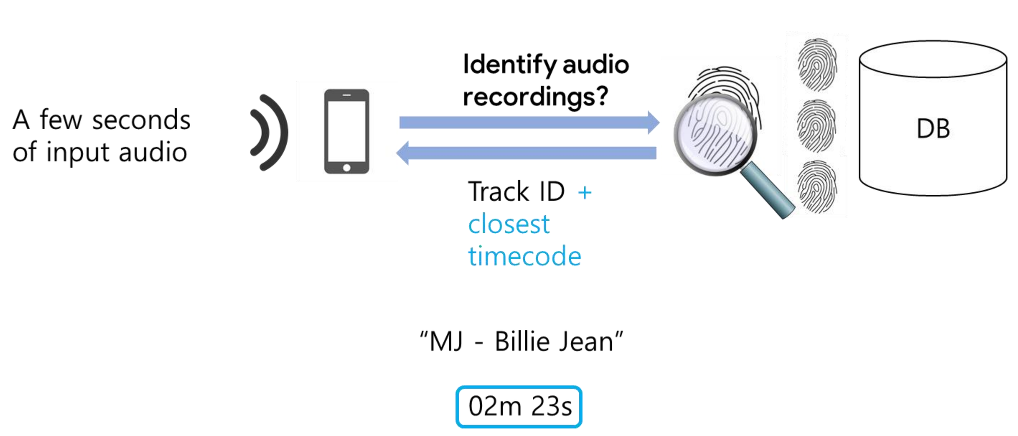
Sungkyun Chang[1] Donmoon Lee[1, 2] Jeongsoo Park[1] Hyungui Lim[1] Kyogu Lee[2] Karam Ko[3] Yoonchang Han[1]
Cochlear.ai, [2] Seoul National University, [3] SK Telecom

Abstract

Most of existing audio fingerprinting systems have limitations to be used for high-specific audio retrieval at scale. In this work, we generate a low-dimensional representation from a short unit segment of audio, and couple this fingerprint with a fast maximum inner-product search. To this end, we present a contrastive learning framework that derives from the segment-level search objective. Each update in training uses a batch consisting of a set of pseudo labels, randomly selected original samples, and their augmented replicas. These replicas can simulate the degrading effects on original audio signals by applying small time offsets and various types of distortions, such as background noise and room/microphone impulse responses. In the segment-level search task, where the conventional audio fingerprinting systems used to fail, our system using 10x smaller storage has shown promising results. Our code and dataset are available at https://mimbres.github.io/neural-audio-fp/.

Introduction

Goal: high-specific audio retrieval



Key aspects

- Segment-level audio search
  - allowing miss-match less than 250 ms
- Contrastive learning framework
  - simulating maximum inner-product search (MIPS) in mini-batch
  - explicitly sampling positive pairs to have original–replica relations
- Audio data augmentation chain
  - generating various types of acoustic distractors that helps training a robust audio fingerprint

Neural Audio Fingerprinter

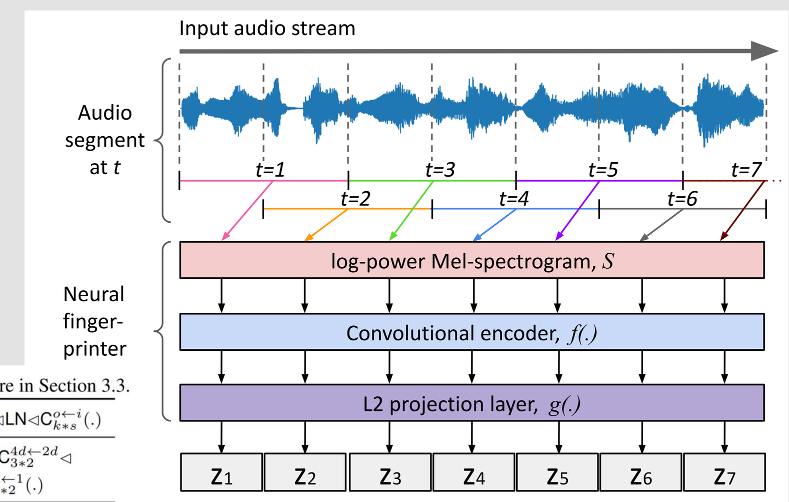


Table 1. Fingerprinter (FP) network structure in Section 3.3.

$\mathsf{SC}^{o\leftarrow i}_{k*s}(.) := \mathsf{ReLU}\triangleleft\mathsf{LN}\triangleleft\mathsf{C}^{o\leftarrow i}_{k'*s'}\triangleleft\mathsf{ReLU}\triangleleft\mathsf{LN}\triangleleft\mathsf{C}^{o\leftarrow i}_{k*s}(.)$
$f(.) := \mathsf{SC}^{h\leftarrow h}_{3*2}\triangleleft\mathsf{SC}^{h\leftarrow 4d}_{3*2}\triangleleft\mathsf{SC}^{4d\leftarrow 4d}_{3*2}\triangleleft\mathsf{SC}^{4d\leftarrow 2d}_{3*2}\triangleleft\mathsf{SC}^{2d\leftarrow 2d}_{3*2}\triangleleft\mathsf{SC}^{2d\leftarrow d}_{3*2}\triangleleft\mathsf{SC}^{d\leftarrow d}_{3*2}\triangleleft\mathsf{SC}^{d\leftarrow 1}_{3*2}(.)$
$g(.) := \mathsf{L2}\triangleleft\mathsf{Concat}\triangleleft\mathsf{C}^{1\leftarrow u}_{1*1}\triangleleft\mathsf{ELU}\triangleleft\mathsf{C}^{u\leftarrow v}_{1*1}\triangleleft\mathsf{Split}^{h/d}(.)$
$\mathsf{FP} := g\triangleleft f(\mathsf{input} := s_t)$

- We employ $g\circ f : S \mapsto \mathcal{Z}\,(.)$ as a segment-wise fingerprinter. It can generate fingerprint z_t that can represent a unit segment of 1 s audio x_t at the time step t.
- $f(.)$ is a base encoder with separable convolution (SC) that computes internal representation.
- $g(.)$ is a multi-head linear projection layer with L2 normalization.

Training $g\circ f(.)$ with contrastive loss can be viewed as a common form of self-supervised learning (SSL). We maintain the self-supervised $g(.)$ up to the final target task.

Contrastive Learning Framework

Due to the L2 normalization layer of $g(.)$, we can use the inner-product $z_a^T z_b$ as a measure of similarity (z_a, z_b). Searching the most similar point (*) of database $V = \{v_i\}$ for a given query q in $\mathbb{Z}^d$ space can be formulated as maximum inner-product search (MIPS):

$$v_i^{\{*\}} := argmax_i(q^T v_i)$$

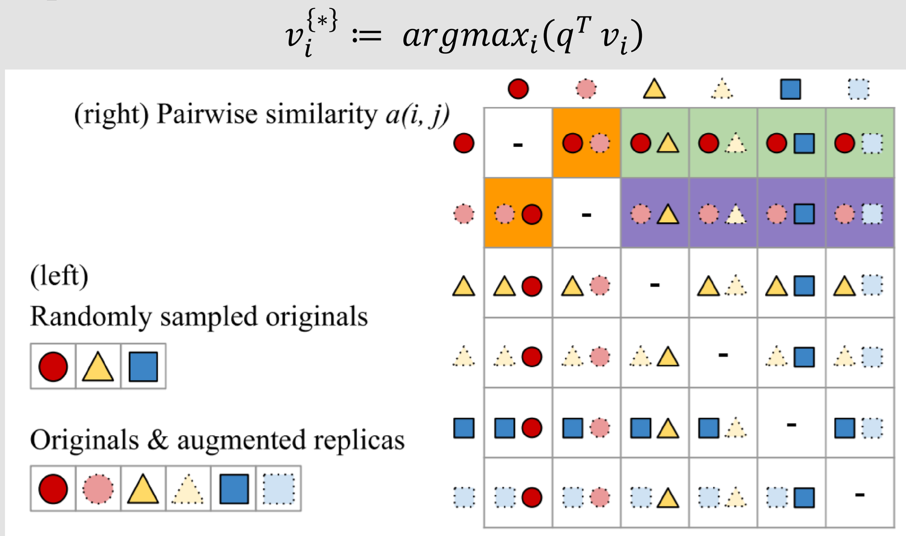


Fig. 2. Illustration of the contrastive prediction task in Section 2.1. (left) Batch size $N = 6$. We prepare $N/2$ pairs of original/replica. The same shapes with solid/dashed lines represent the positive pair of original/replica, respectively. (right) Each element in the matrix represents pairwise similarity. In each row, a prediction task can be defined as classifying a positive pair (one of the orange squares) against the negative pairs (green or purple squares) in the same row.

$$\ell(i,j) = -\log\frac{\exp(a_{i,j}/\tau)}{\sum_{k=1}^{N}\mathbb{1}(k \neq i)\exp(a_{i,j}/\tau)}, \quad (1)$$

$\mathbb{1}(.) \in \{0, 1\}$ is an indicator function that returns 1 iff $(.)$ is true, and $\tau > 0$ denotes the temperature parameter for softmax.

The Eq(1) can replace MIPS from the property: computing the top-k predictions in the softmax function is equivalent to the MIPS.

Experiments

- Dataset (from FMA)
  - Train 10K songs (30s cut each)
  - Test-DB: unseen 100K songs (full)
- Test-Query (synthesized)
  - 2K seq. generated with unseen augmentation source
- Augmentation source dataset (from AudioSet & others)
  - BG mix: 6.6 h environmental noise (subway, metro, pub, café,...) with "no music"
  - MIC and room/space IR
- Top-1 hit rate (%) = $100 \times \frac{(n\ of\ hits\ @Top\text{-}1)}{(n\ of\ hits\ @Top\text{-}1) + (n\ of\ miss\ @Top\text{-}1)}$

Table 3. Top-1 hit rate (%) of large-scale (total of 100K songs) segment-level search. *d* denotes the dimension of fingerprint embedding. *exact match* means that our system finds the exact index. *near match* means a mismatch within ± 1 index or ± 500 ms.

Method	d	match	Query length in seconds: 1 s	2 s	3 s	5 s	6 s	10s
Now-playing (replicated)	128	exact	-	44.3	60.1	73.6	81.0	86.1
		near	-	46.8	63.5	75.2	81.6	86.3
Now-playing (modified for 1 s unit)	64	exact	25.8	58.5	69.3	78.5	81.4	87.7
		near	30.9	61.3	71.2	79.5	82.2	88.3
	128	exact	26.3	58.2	69.5	78.4	81.4	87.8
		near	30.9	61.1	71.8	79.8	83.0	89.2
This work (N=640)	64	exact	54.6	78.9	85.4	90.4	92.0	94.9
		near	61.3	81.7	86.7	90.9	92.7	95.1
	128	exact	**62.2**	**83.2**	**87.4**	**92.0**	**93.3**	**95.6**
		near	68.3	84.9	88.7	92.7	94.1	95.8
This work (N=320)	128	exact	61.0	82.2	87.1	91.8	93.1	95.2
		near	67.1	84.1	88.1	92.5	93.9	95.5
This work (N=120)	128	exact	55.9	78.8	84.9	90.9	92.2	95.3
		near	62.3	80.9	86.3	91.5	92.8	95.5
This work (no aug.)	128	exact	0.0	0.0	0.0	0.0	0.0	0.0
		near	0.0	0.0	0.0	0.0	0.0	0.0

Table 4. Effect of fingerprint dimension d in 1 s segment search.

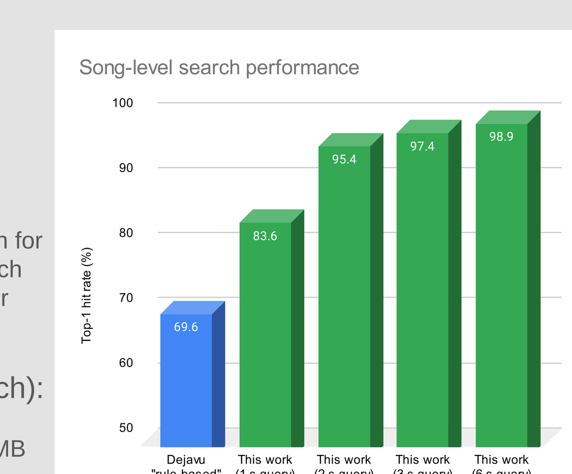
Embedding dimension	d=16	d=32	d=64	d=128
Top-1 hit rate@1 s (%)	11.6	40.2	54.6	62.2

- Now-playing < Now-playing mod.
- In every cases, our model outperformed over Now-playing. Contrastive learning framework > semi-hard triplet embedding.
- Augmentation was critical in training.
- The larger the batch size, the better the performance in all experiments.
- The longer the query sequence, the better the performance.
- Increasing embedding dimension d improves performance.

VS. Dejavu (rule-based)

In the small (10K-30s) DB test:
- This work (6 s query) = 98.9 % exact match for segment-level search
- Dejavu (6 s query) = 69.6% exact match for song-level search

Memory usage (w/ in-memory search):
- Dejavu = 400 MB
- This work (d=64, after compression) < 40 MB
- This work uses 10x less memory!

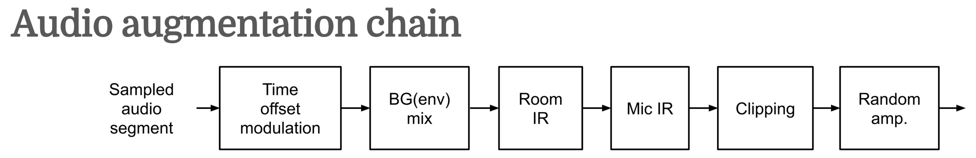


Conclusion

- The proposed contrastive learning framework could simulate the search task by explicitly sampling original-replica (clean-augmented) pairs as positive pairs.
- In the segment-level search task our model performed better than the model with triplet embeddings.
- Our model, using 10x less memory and shorter query length (< 3 s) than an existing rule-based algorithm, outperformed in song-level search.
- The superior performance of our model in the task is not due to any single design choice, but a combination of design choices.
- This study implies that the audio fingerprinting ta sk would inherently have self-supervised learnin g potentials.

Implementation Notes

Audio augmentation chain



Time-offset modulation

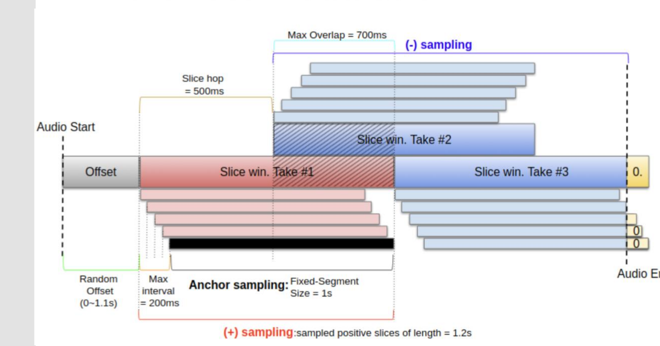
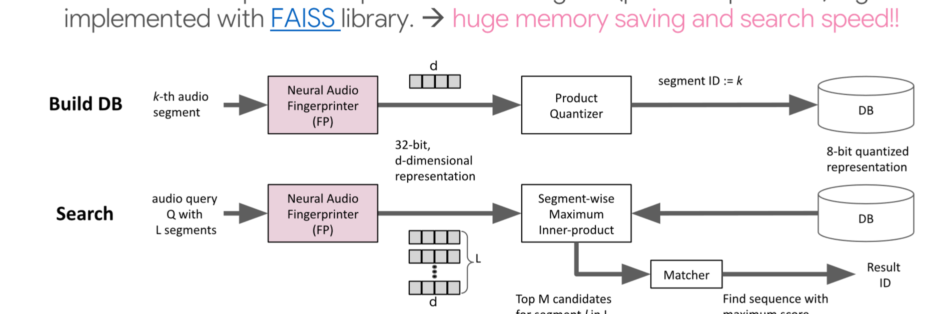


Table 2. Shared configurations for experiments

Parameter	Value
Sampling rate	8,000 Hz
STFT window function	*Hann*
STFT window length and hop	1024, 256
STFT spectrogram size $F \times T$	$512 \times T (T = 32)$
log-power Mel-spectrogram size $F\prime \times T$	$256 \times T (T = 32)$
Dynamic range	80 dB
Frequency {min, max}	{300, 4,000} Hz
Fingerprint {window length, hop}	$\{1s, 0.5s\}$ or $\{2s, 1s\}$
Fingerprint dimension d	64 or 128
Network parameters $\{h, u, v\}$	$\{1024, 32, h/d\}$
Batch size N	120 or 320 or 640

Build DB & Search with sequence matcher

We build DB using only original audio sources. The resulting fingerprint vector is stored as a compressed representation using PQ (product quantizer) algorithm implemented with FAISS library. → huge memory saving and search speed!!



References & Acknowledgements

- B. Gfeller et al., "Now playing: Continuous low-power music recognition," in NeurIPS 2017 Workshop on Machine Learning on the Phone and other Consumer Devices, 2017.
- Chen, Ting, et al. "A simple framework for contrastive learning of visual representations." International conference on machine learning(ICML). 2020.
- https://github.com/worldveil/dejavu

We would like to thank the TensorFlow Research Cloud (TFRC) program that gave us access to Google Cloud TPUs.

Sungkyun Chang (1ronyar@gmail.com)