Cross-Modal Embeddings in Music and Audio

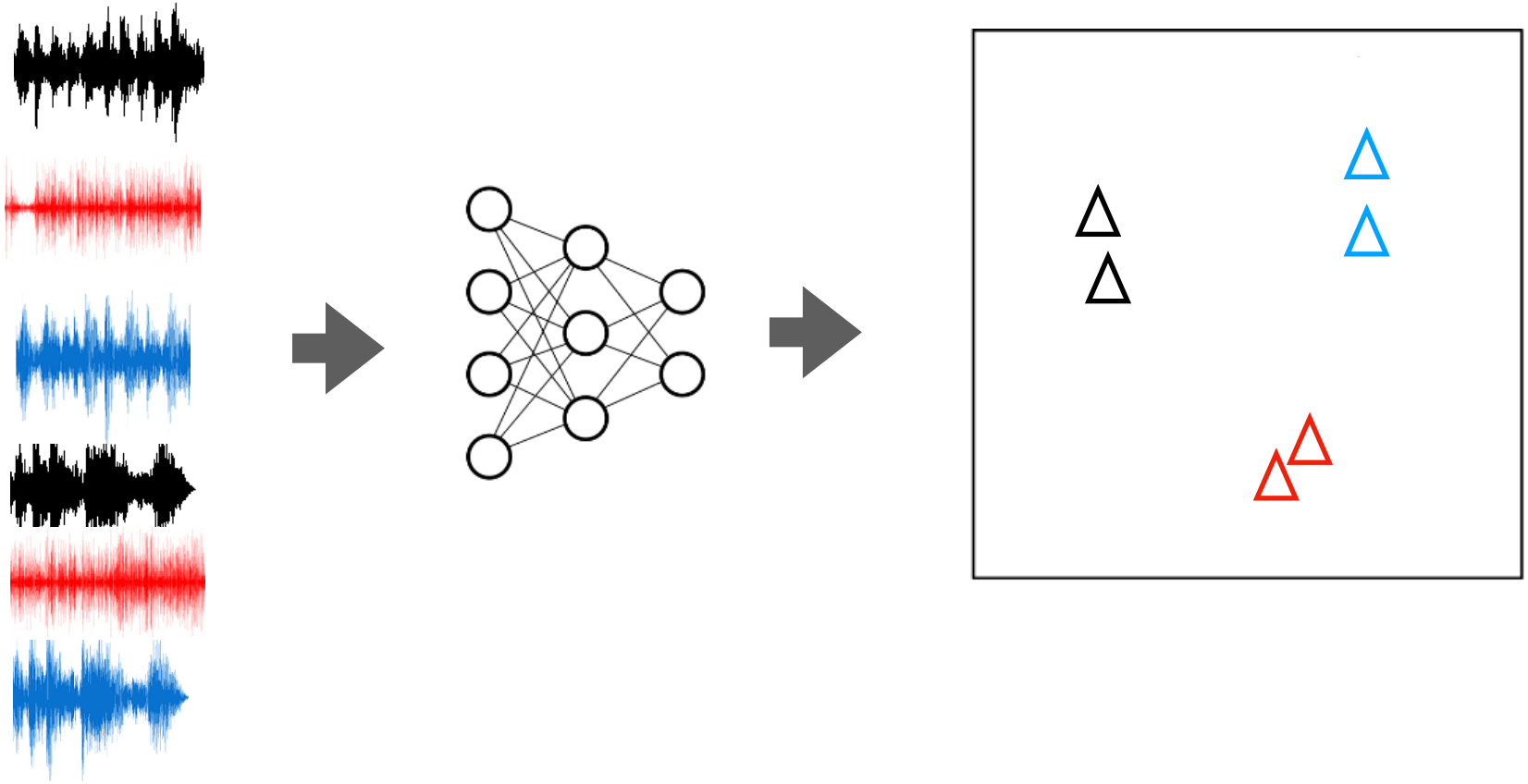
CS352 Spring 2025
May 28, 2025
Annie Chu

Things we'll cover

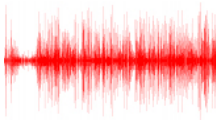
1. Embeddings Recap
2. What is multimodality?
3. Multimodal Embedders (we'll look primarily at CLAP)
    3.1. Training Methodology
    3.2. Evaluation tasks
    3.3. Case Study: Text2FX
4. Visualizing embeddings
5. Get started with CLAP

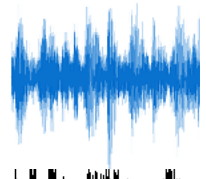
Recap: Audio Embeddings

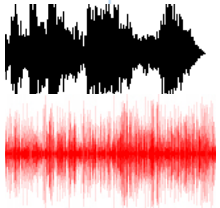
A (trained) embedding net

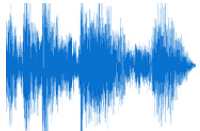


A (trained) embedding net

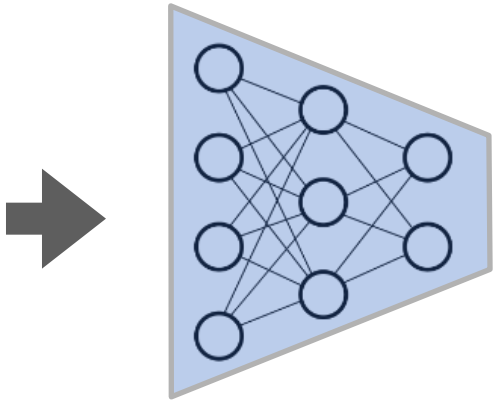




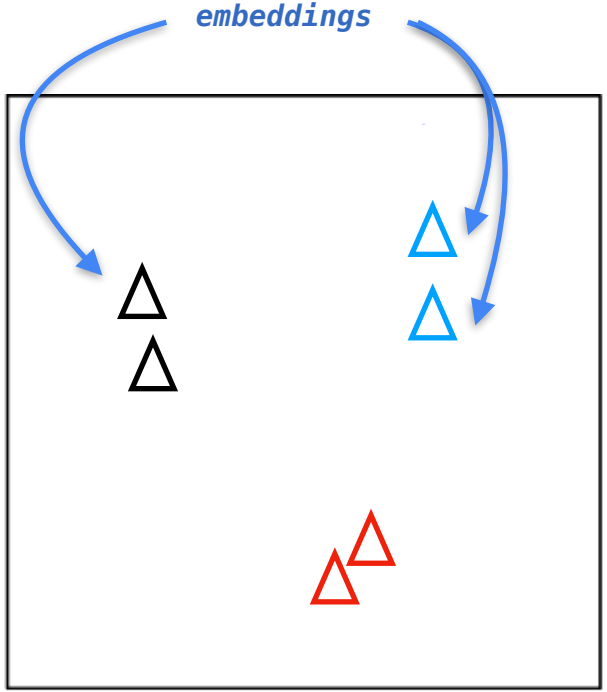




input audio representation



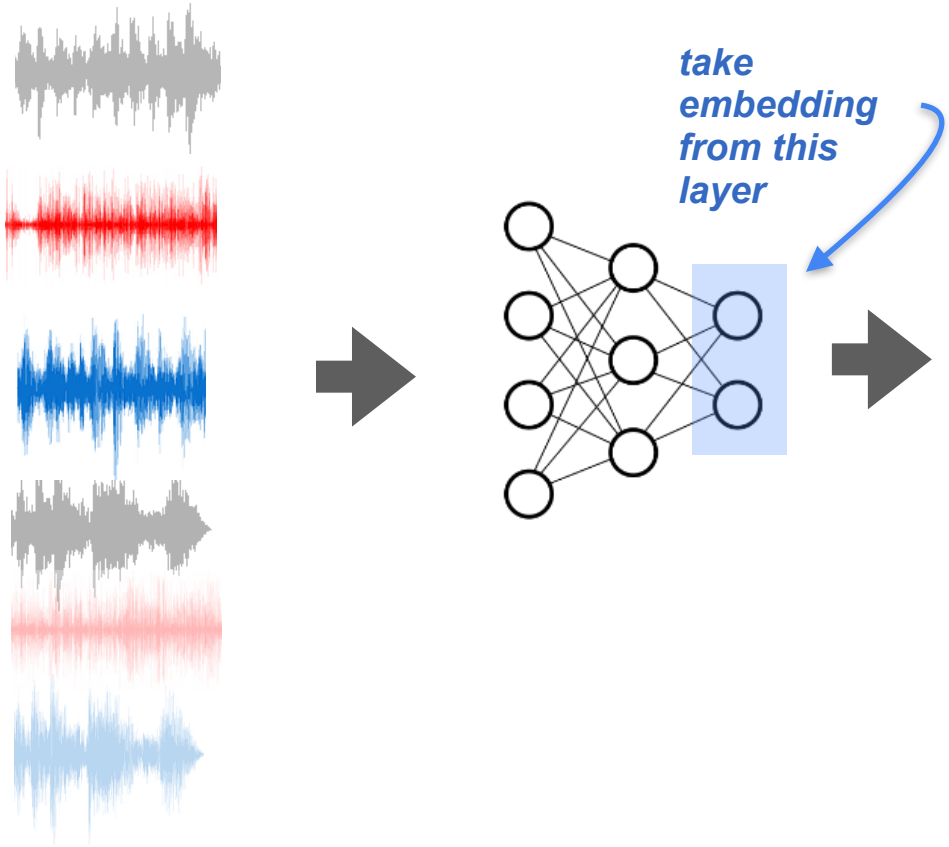
embedding NN
aka “encoder”



N–D Embedding Space

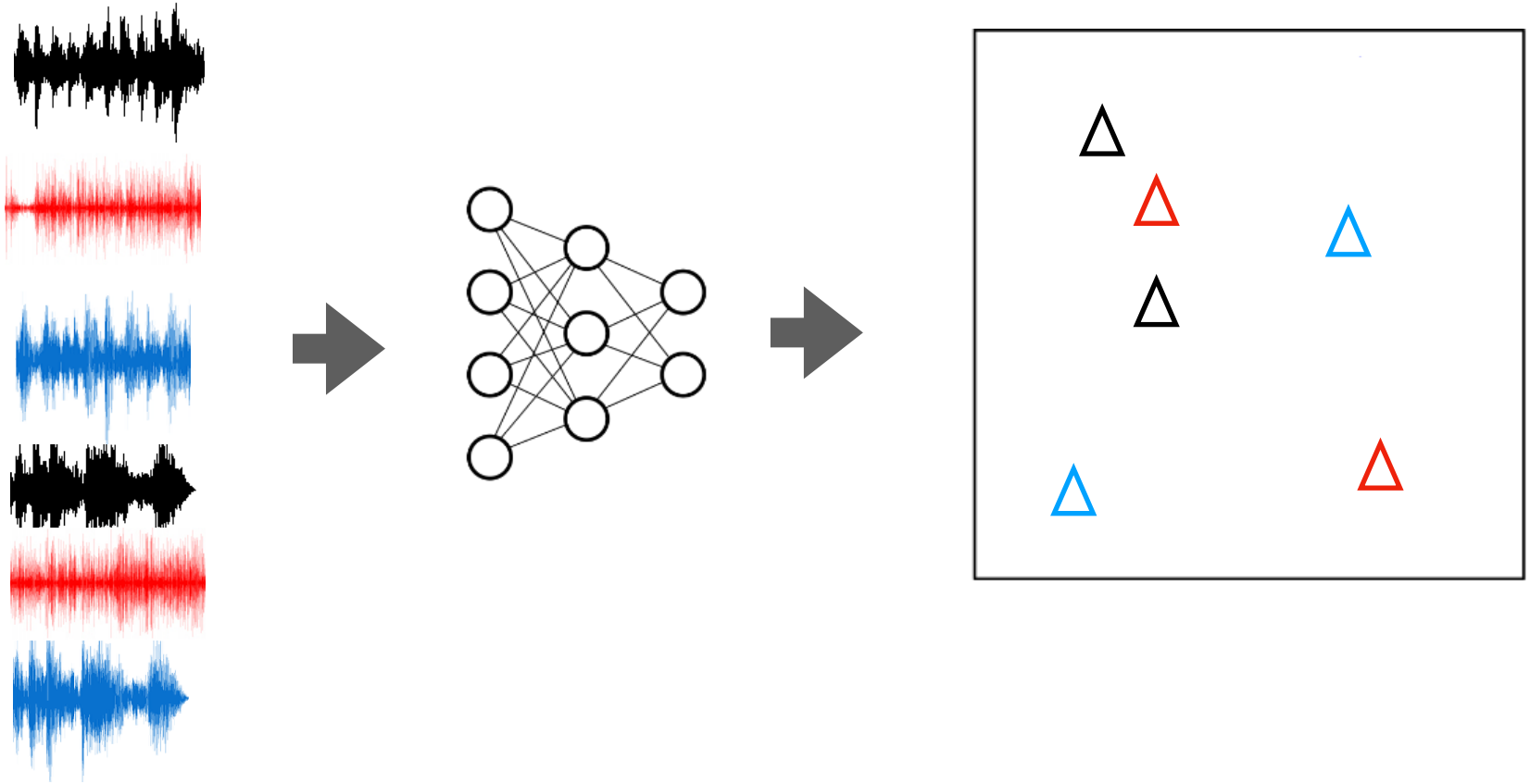
N = 2

A (trained) embedding net

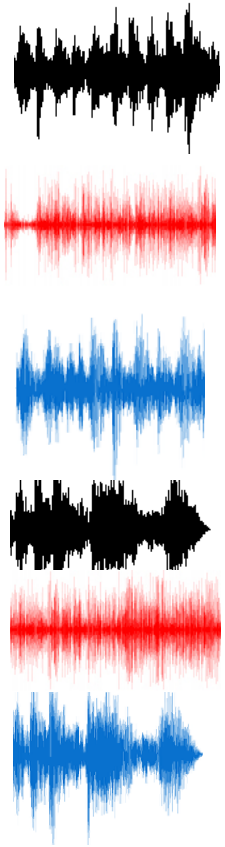


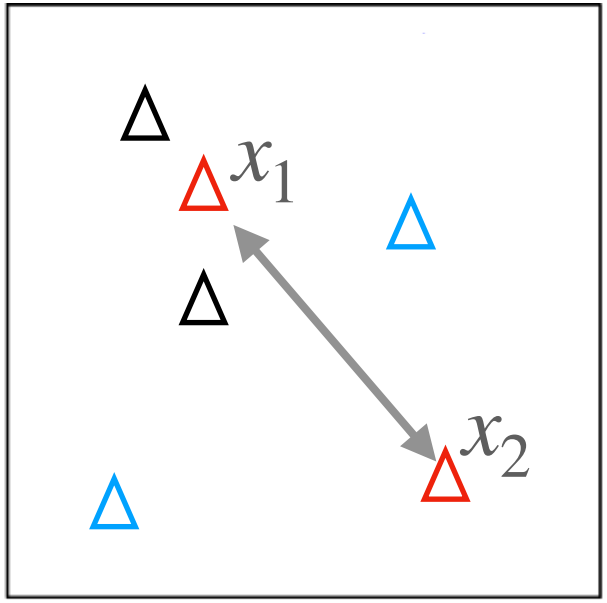
vector encodes semantic or structural information about the input

Training an embedding net



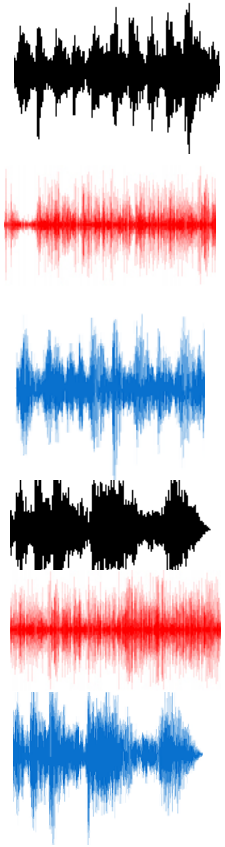
Training an embedding net

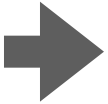


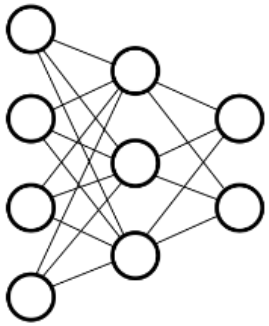


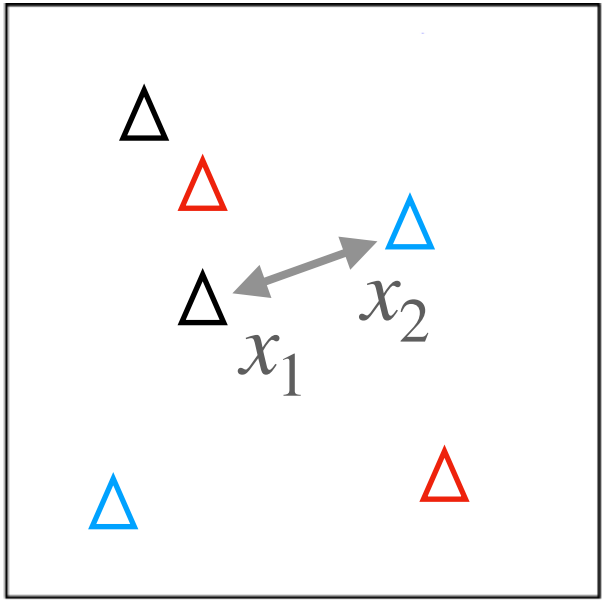
$$Loss \propto D_f(x_1, x_2)$$

Training an embedding net



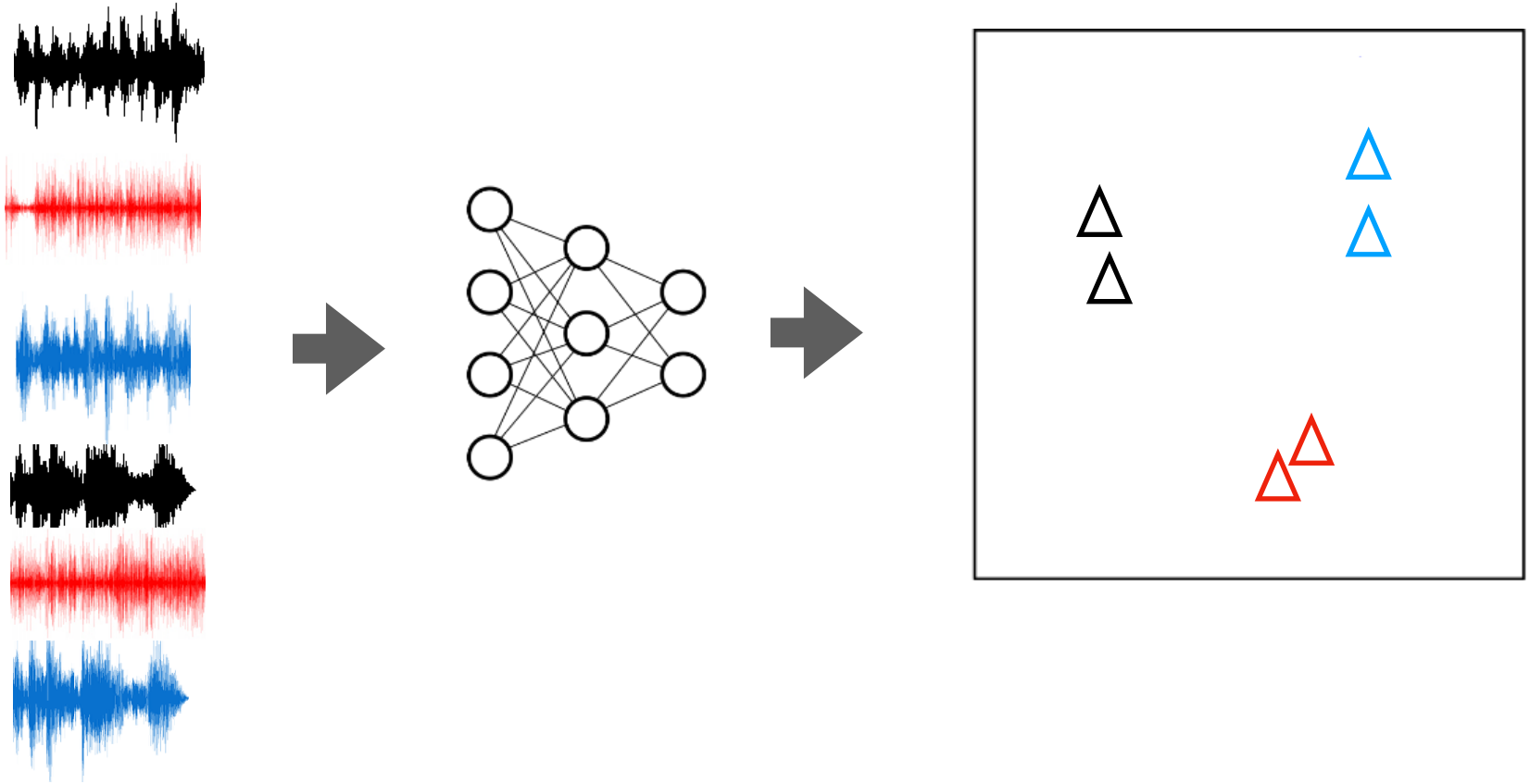






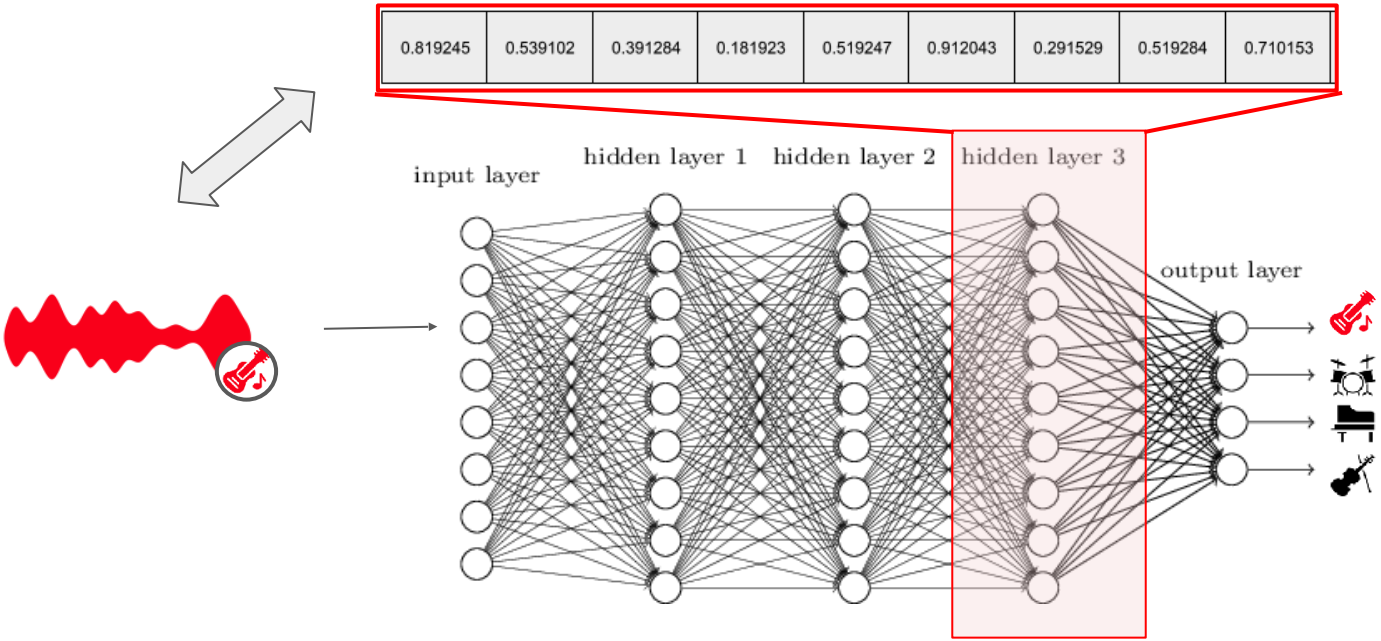
$$Loss \propto M - D_f(x_1, x_2)$$

Training an embedding net



Don’t have to train an embedder explicitly

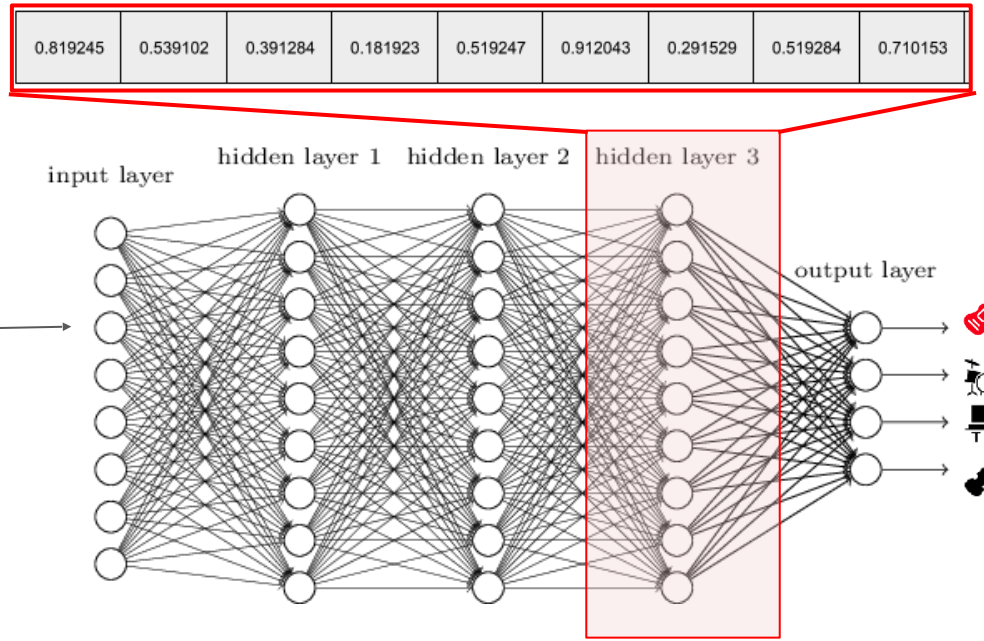
Embeddings extracted as general-purpose feature vector



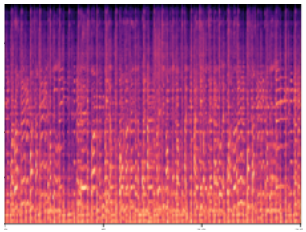
Any deepnet to “encode” audio in a relevant semantic space, even say instrument classifier

we’ll take embeddings from penultimate layer

Input Audio Representations — we have options



raw waveform input



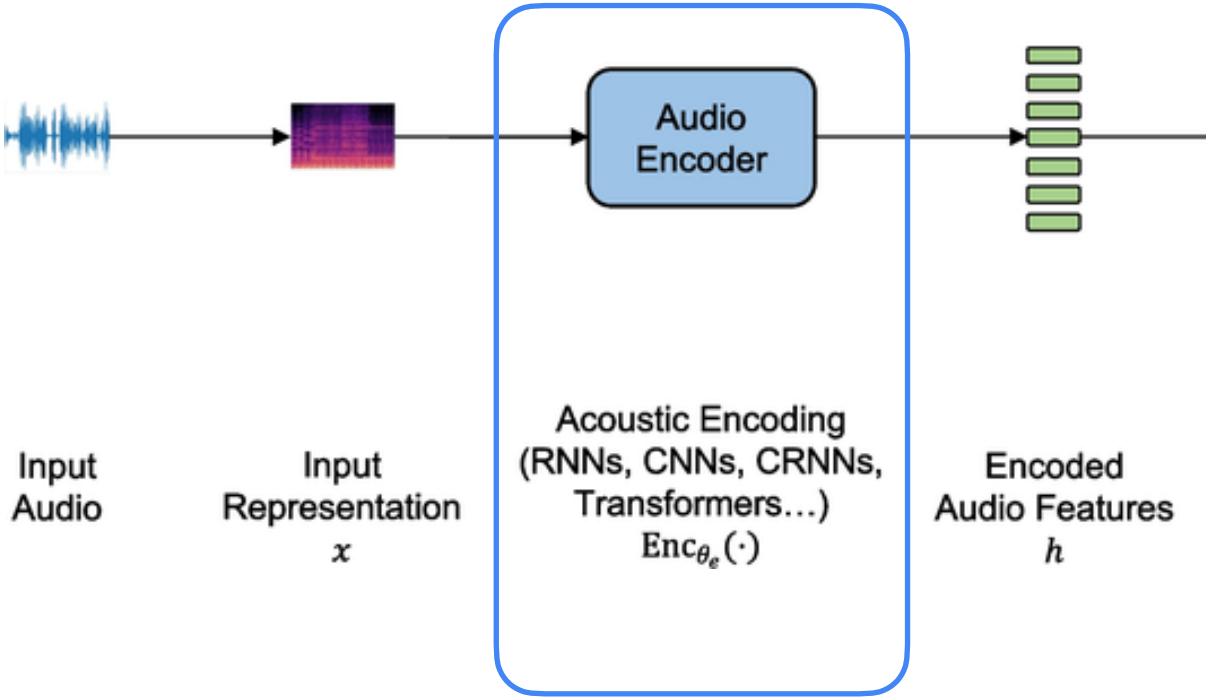
or spectrogram

Any deepnet to "encode" audio in a relevant semantic space, even say instrument classifier

we'll take embeddings from penultimate layer

Encoder Architectures— some options

Any deepnet to “encode” audio in a relevant semantic space

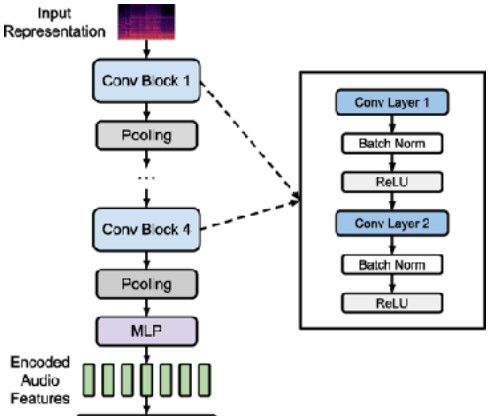


Encoder Architectures— some options

Any deepnet to “encode” audio in a relevant semantic space

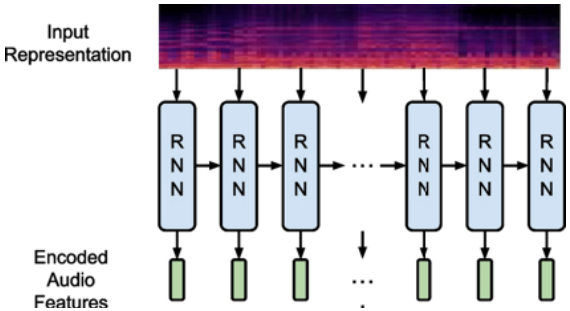
CNN

local patterns (timbre, onsets)



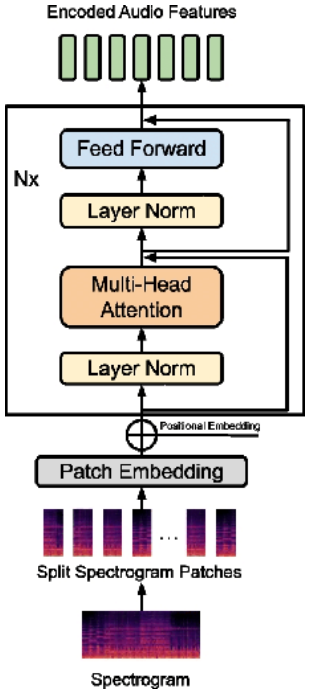
RNN

sequence dynamics (time dependencies)



transformer

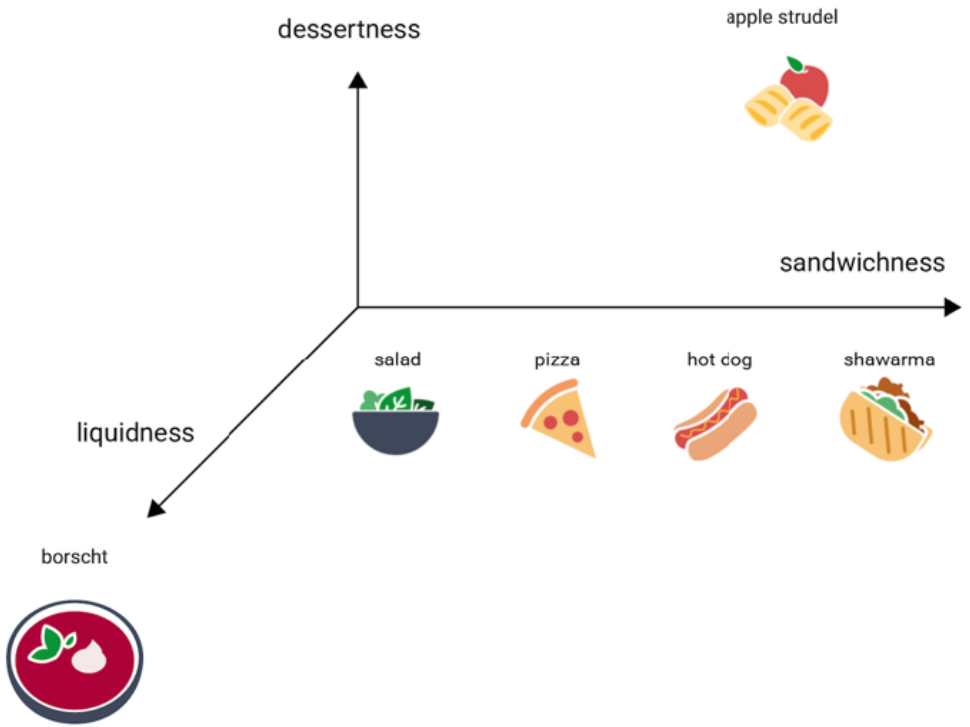
global attention over time



https://www.researchgate.net/publication/364288810_Automated_audio_captioning_an_overview_of_recent_progress_and_new_challenges?_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6Il9kaXJlY3QiLCJwYWdlIjoiX2RpcmVjdCJ9fQ

The point of embedding spaces

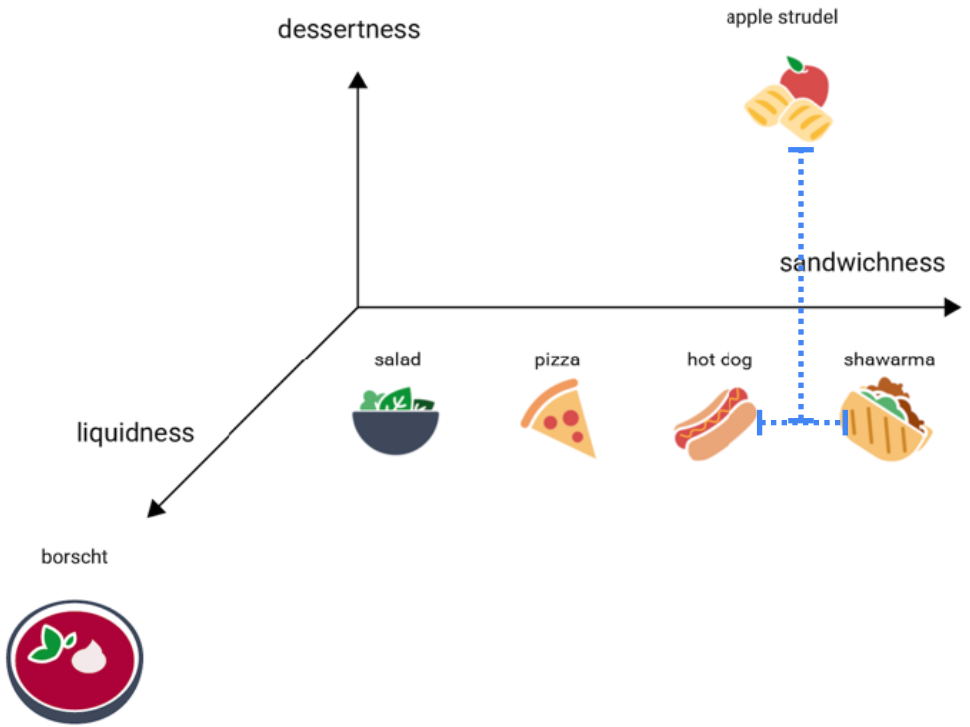
- Low-dimensional representations of high-dimensional data that captures **semantic relationships between items**
- similar items closer together in the embedding space and dissimilar items further apart



source: google

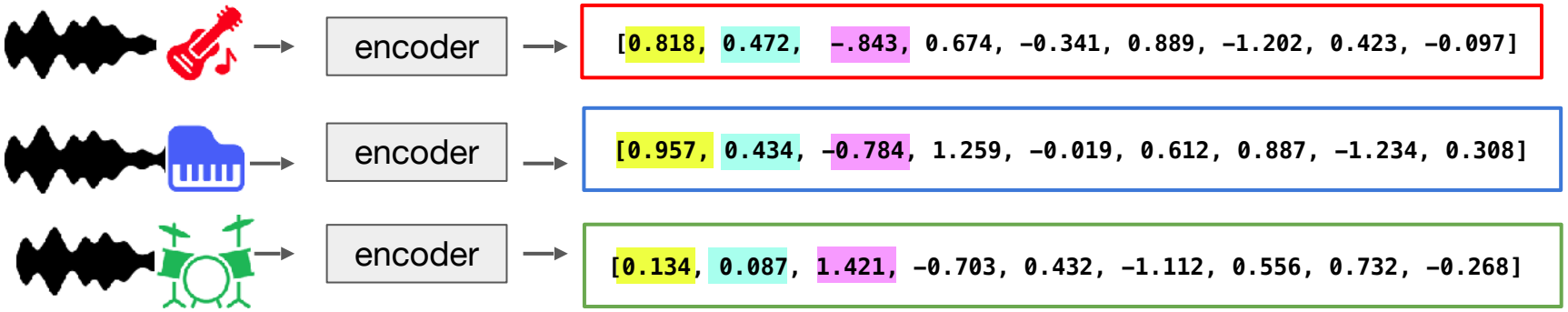
The point of embedding spaces

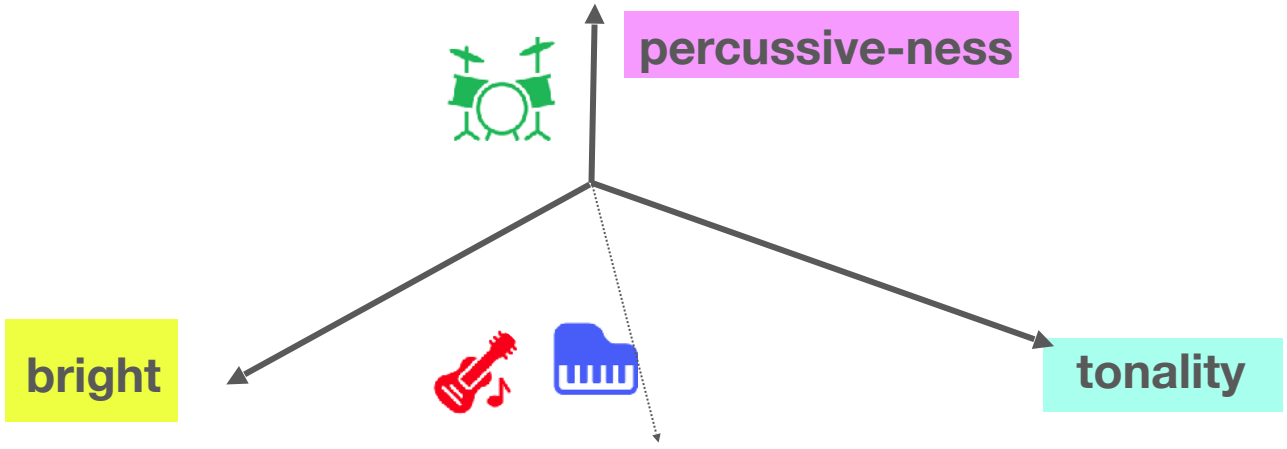
- Low-dimensional representations of high-dimensional data that captures **semantic relationships between items**
- similar items closer together in the embedding space and dissimilar items further apart



source: google

In the audio space, it might look like this





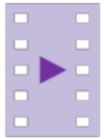
And we can do this for other modalities

What do we mean by modality?

audio
environmental sound, speech, music

image
pixels, shapes, colors

Mode
distinct source of information



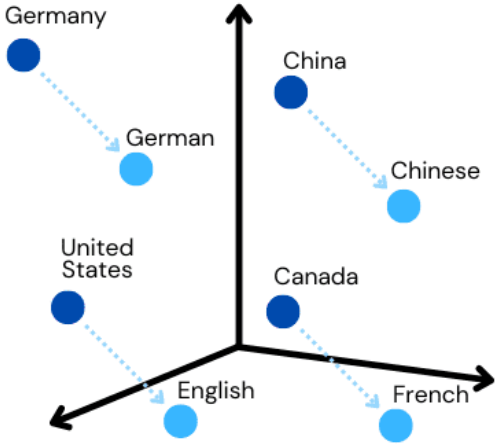
video
visual frames + temporal motion

text
words, language

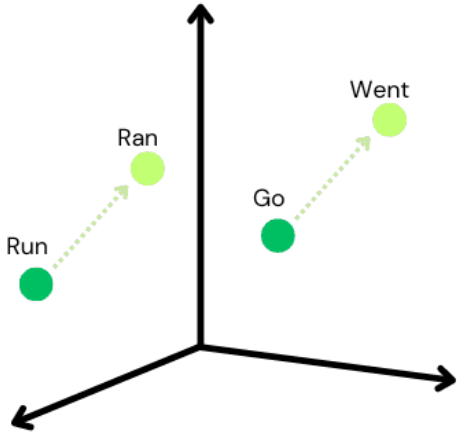
sensors
GPS, accelerometer, etc

Multimodal systems combine two or more modalities to form **richer representations**

Text Embedding Space



Country - Language



Verb Tense

vector offsets = consistent vector transformations reusable across space

(“embedding math”)

vectors capture semantic relationships

image source

So what if we add combine multiple modalities?

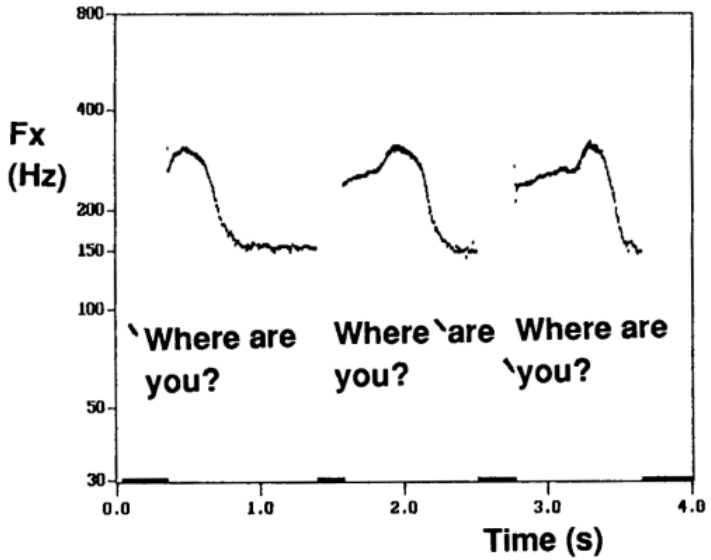
Why do we care about multimodality?

Faithfulness to natural human interaction

- Broader human experience is multimodal

Complementary information

- Different modalities can compensate information the other is missing



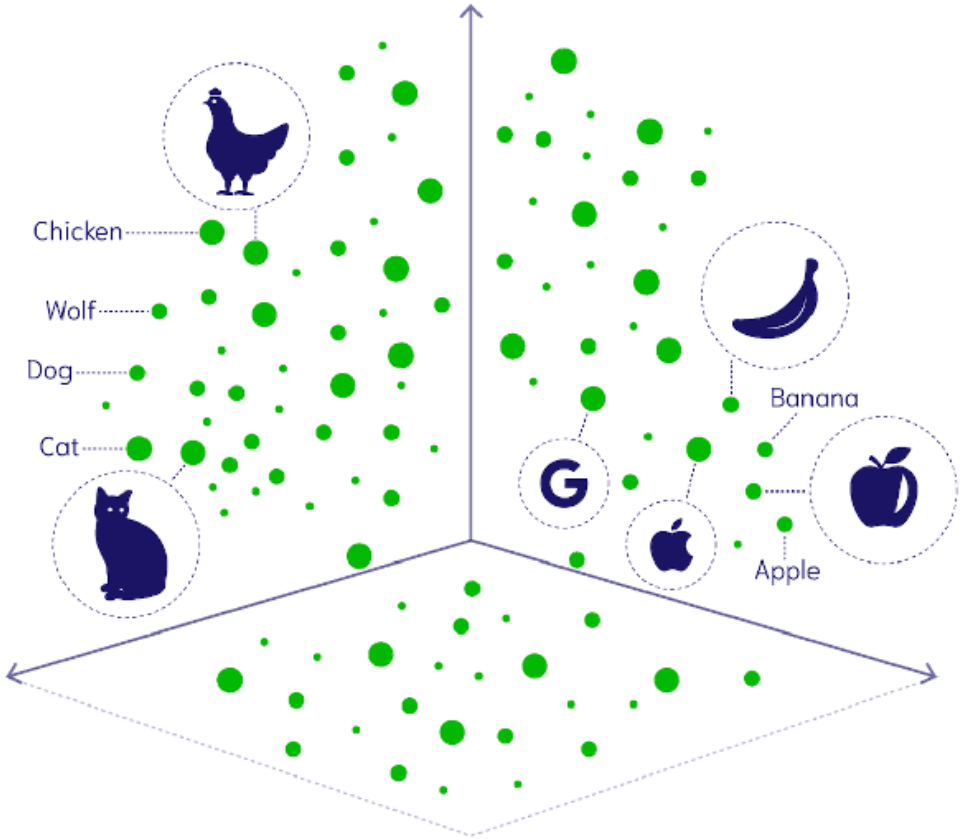
prosody example:

text modality can capture what a person is saying, but not how they're saying it

Multimodal Embeddings

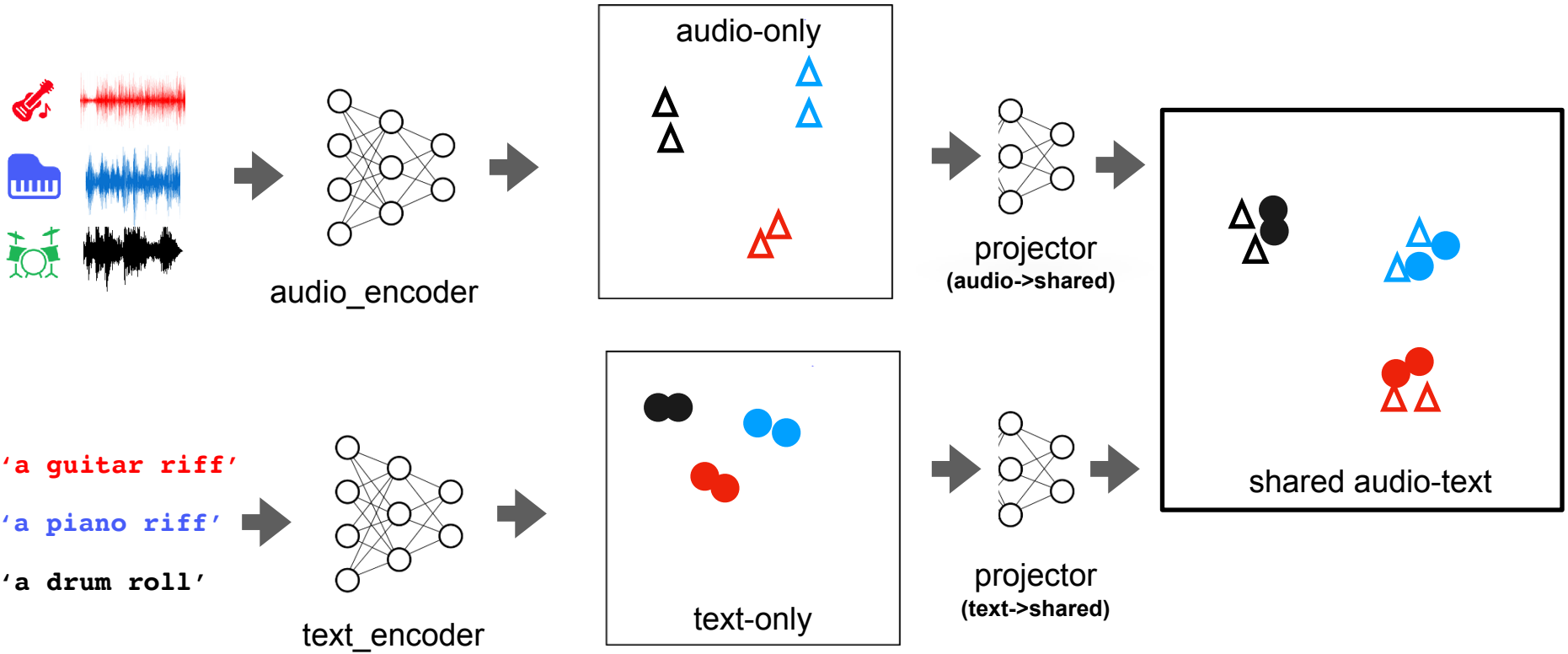
Different modalities, same embedding space

Shared semantic vector space for across multiple modalities (e.g.,audio, text, video, symbolic)



example of shared text-image embedding space

Multimodal Embedder Overview



Let’s look at some audio+X multimodal models

Audio+Text

CLAP

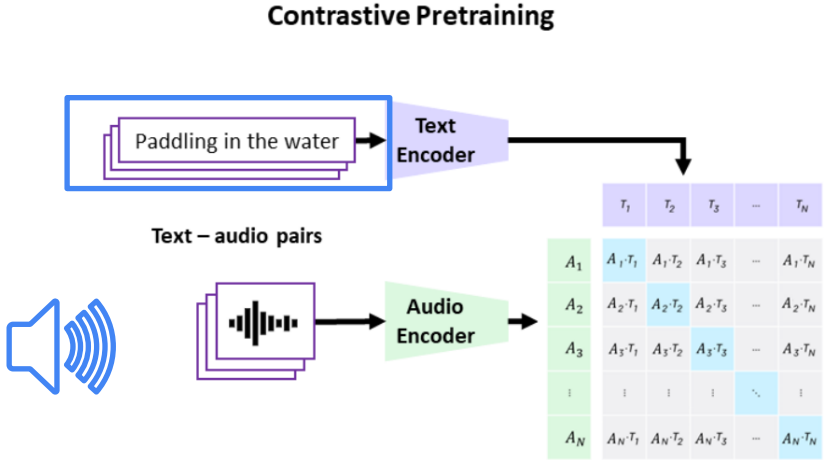
Contrastive Language-Audio Pretraining

CLAP LEARNING AUDIO CONCEPTS FROM NATURAL LANGUAGE SUPERVISION

Benjamin Elizalde, Soham Deshmukh, Mahmoud Al Ismail, Huaming Wang

Microsoft

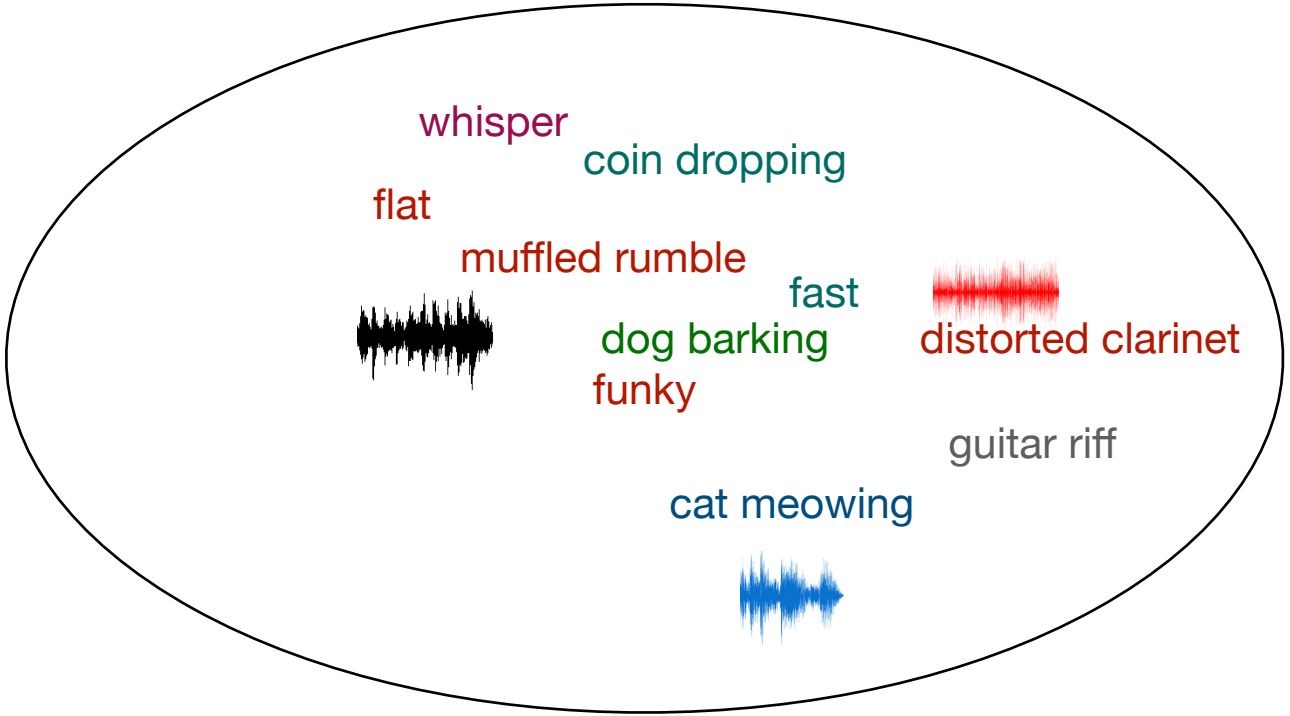
{benjaminm, sdeshmukh, malismail, huawang}@microsoft.com



B. Elizalde, S. Deshmukh, M. Al Ismail, and H. Wang, "Clap: Learning audio concepts from natural language supervision," in ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2023, pp. 1–5.

Audio is FreeSound 16770__dobroide__20060307paddling.wav

Shared Text-Audio embedding space



✅ a joint embedding space aligning audio concepts with corresponding text ✅

Audio+Visual

Audio-Visual coherence

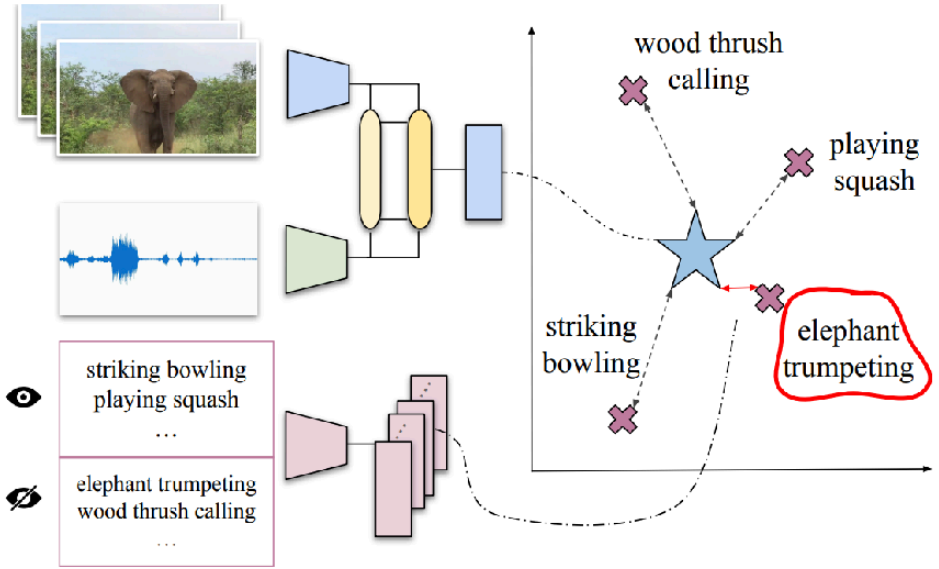
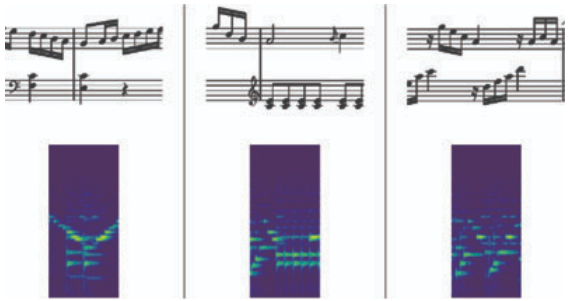


Figure 1. Our audio-visual (generalised) ZSL framework aligns an audio-visual embedding with the corresponding textual label embedding via cross-modal attention. It can classify videos from previously unseen classes (*e.g. elephant trumpeting*) by predicting the class (red) whose textual label embedding (purple cross) is closest to the audio-visual embedding (blue star).

Audio+Symbolic



Learning Audio–Sheet Music Correspondences for Cross-Modal Retrieval and Piece Identification
https://transactions.ismir.net/articles/10.5334/tismir.12?ref=https%3A%2F%2Fgithubhelp.com

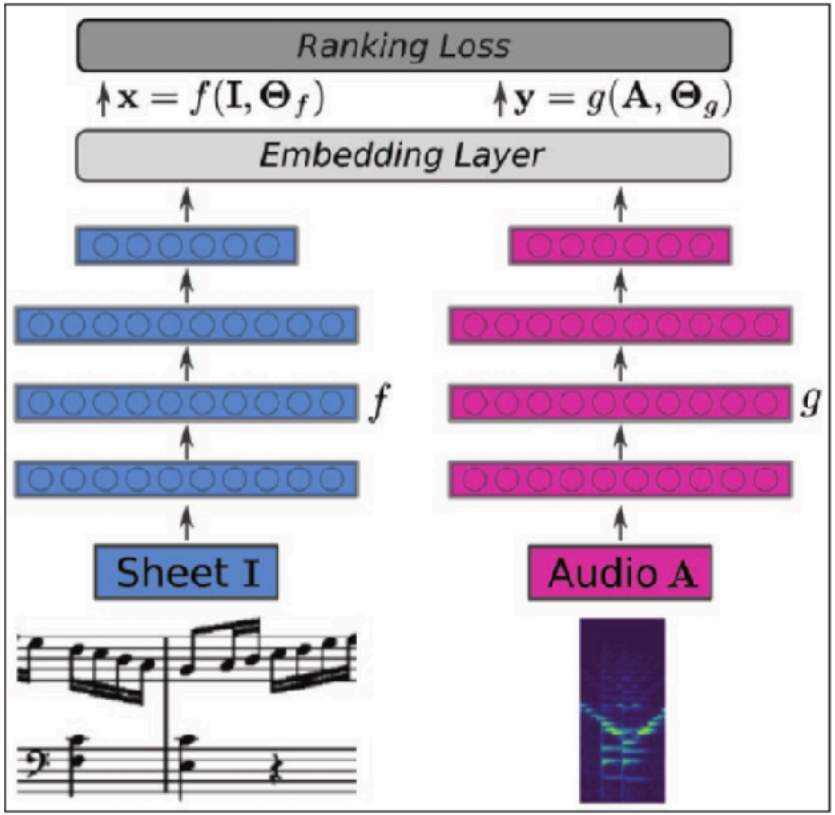
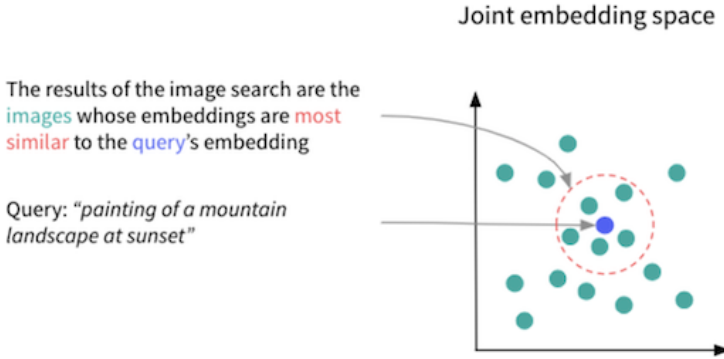


Figure 5: Architecture of correspondence learning network. The network is trained to optimize the similarity (in embedding space) between corresponding audio and sheet image snippets by minimizing a pair-wise ranking loss.

What are multimodal embeddings useful for?

- **Retrieval** (audio <> text // “music that sounds like this description”)
- **Automatic tagging/captioning** (audio → text)
- **Multimodal generation** (text → audio)
- **Bridging audio with symbolic representations** (e.g., MIDI, sheet music)



example of retrieval (image)

once modalities are embedded in the same space, we can map between them in flexible, meaningful ways

How do we train multimodal models?

First we need data

Training Data

We need (a lot) of paired data

example text-audio datasets:

AudioCaps (Kim et al., 2019)
(audio <> text_captions)
source: AudioSet (Youtube)

Clotho (Drossos et al,, 2019)
(audio_event <> text_captions)
source: Freesound

Audealize (Seetharaman & Pardo, 2017)
(audio_FX_parameters <> text_descriptors)
source: Crowdsourced

[Audio Classification] rumble | vehicle | speech | car | outside

[Video Captioning] A bus passing by with some people walking by in the afternoon.

[Audio Captioning] A muffled rumble with man and woman talking in the background while a siren blares in the distance.

AudioCaps: Generating Captions for Audio in the Wild

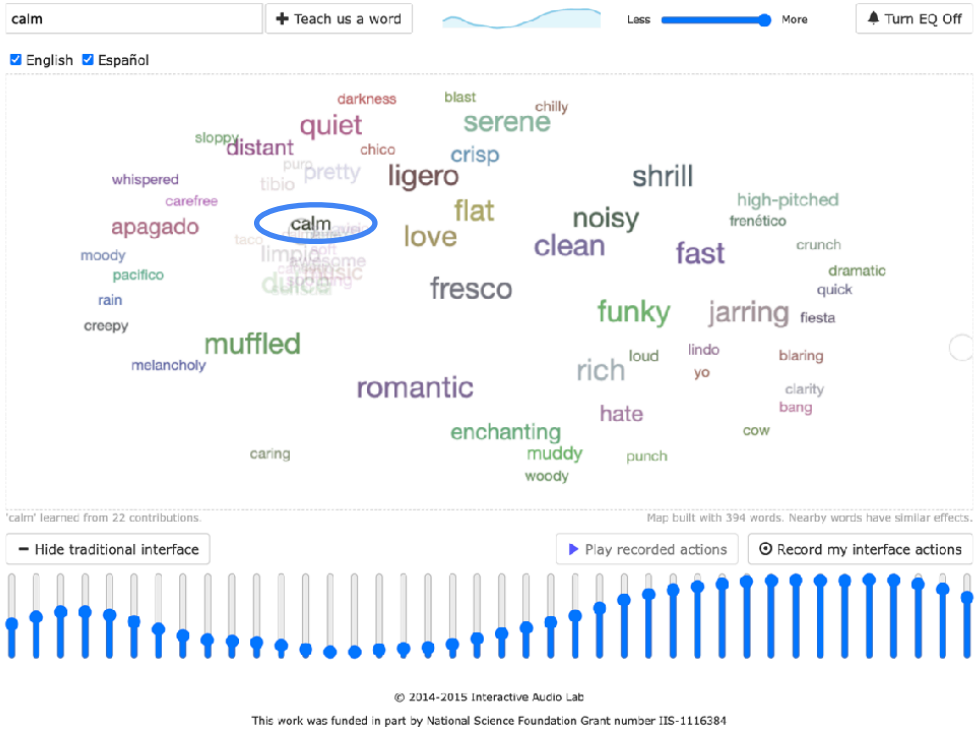
file_name	caption_1	caption_2	caption_3	caption_4
Distorted AM Radio noise.wav	A muddled noise of broken channel of the TV	A television blares the rhythm of a static TV.	Loud television static dips in and out of focus	The loud buzz of static constantly changes pitch and volume.

clotho example

Audealize

(Seetharaman & Pardo, 2017)

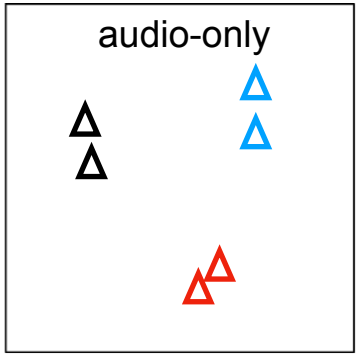
“calm” <> FX parameters (40-band EQ)



Next we need to embed our data
(in their respective modalities)

Choose encoder

audio_encoder

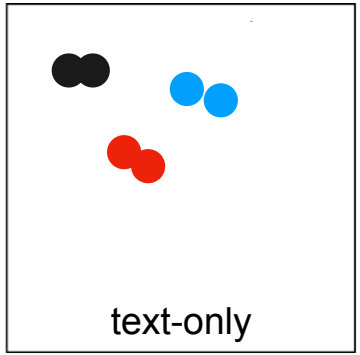


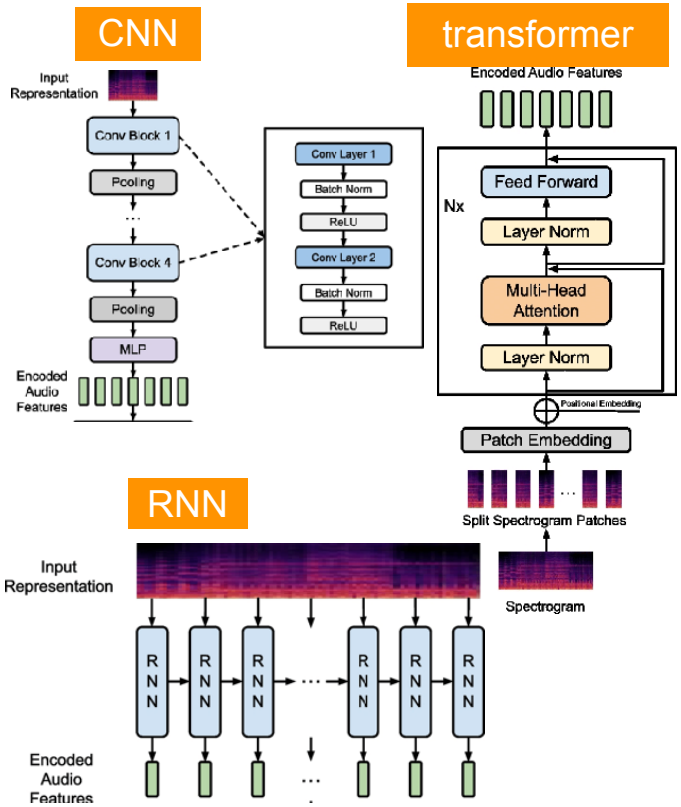
'a guitar riff'

'a piano riff'

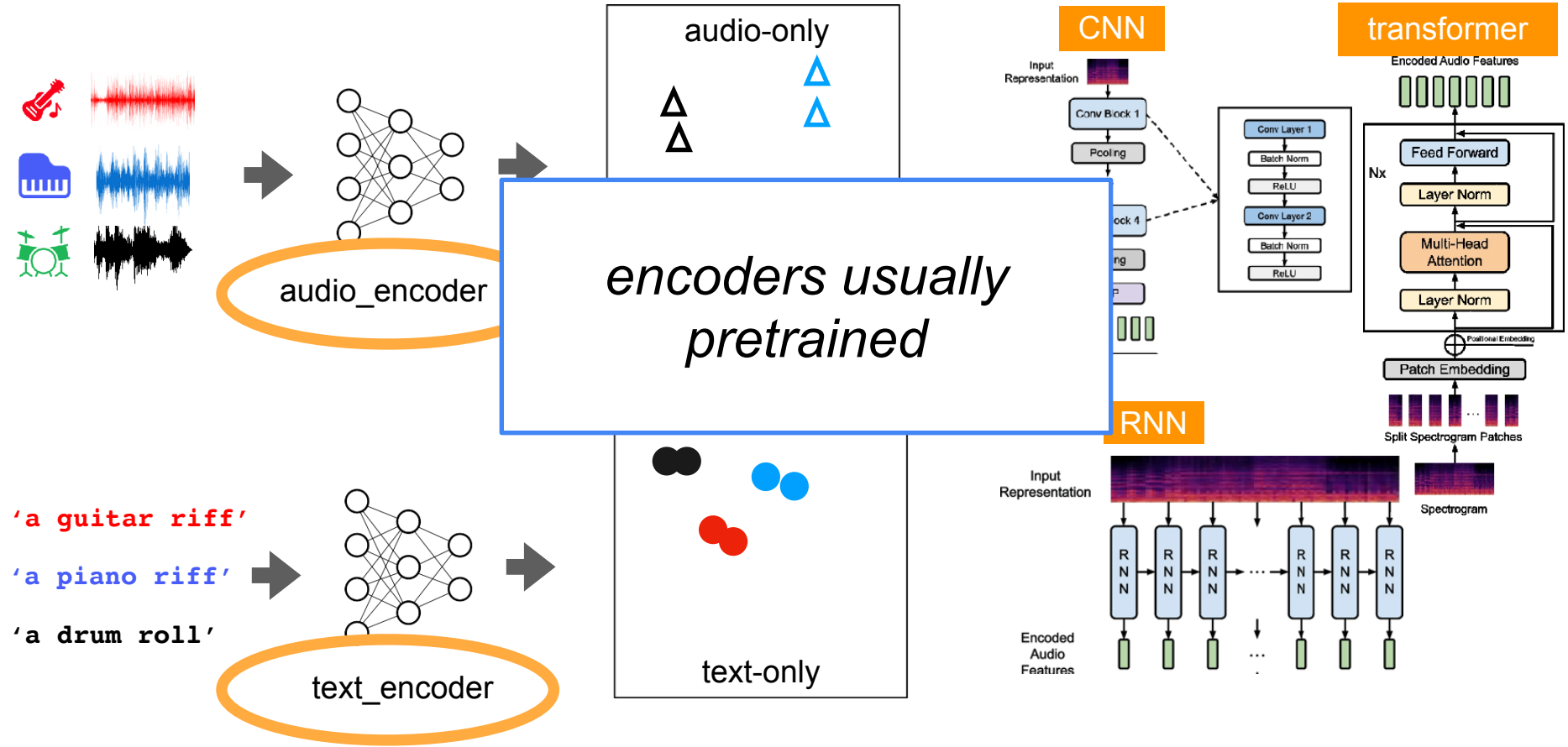
'a drum roll'

text_encoder



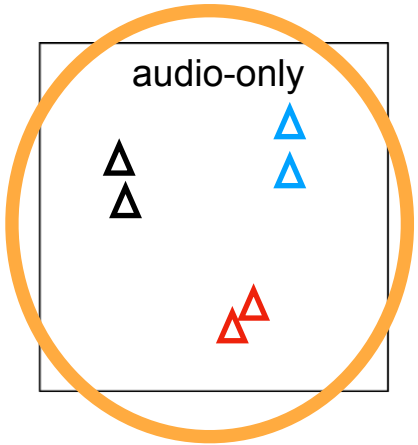


Choose encoder



Get embeddings

audio_encoder

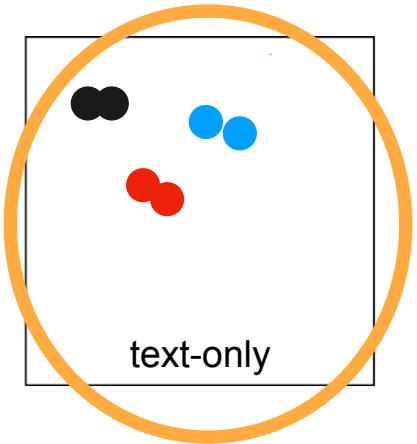


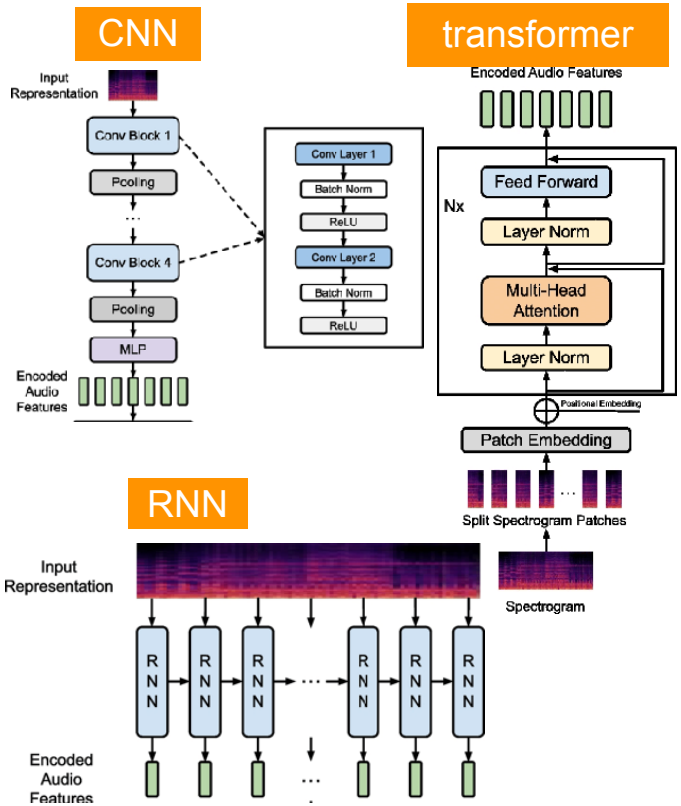
'a guitar riff'

'a piano riff'

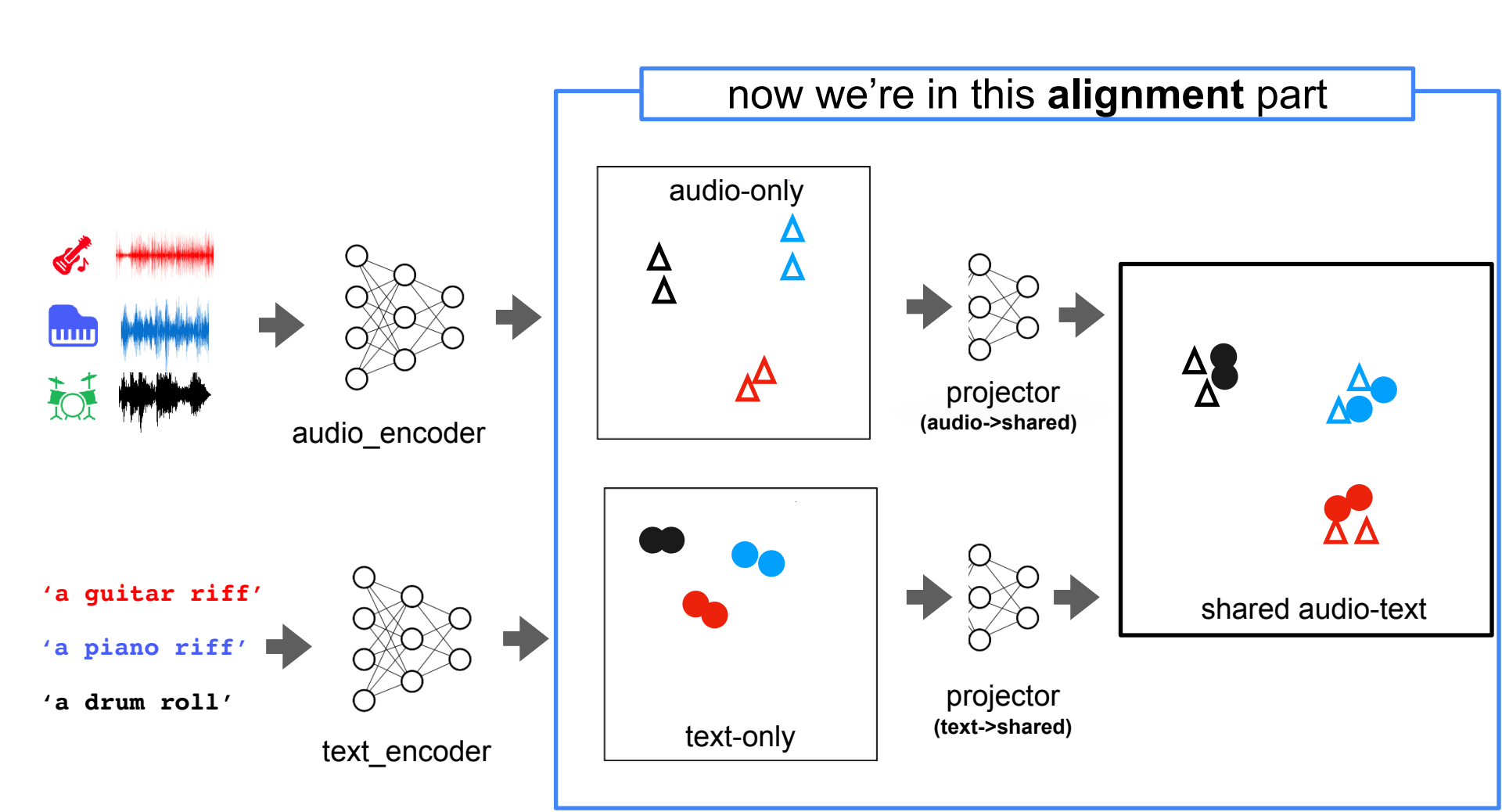
'a drum roll'

text_encoder





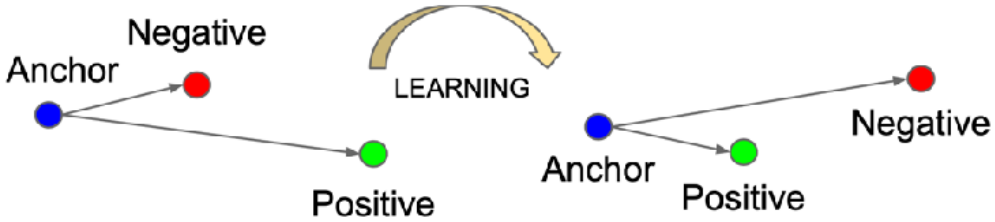
Now we need to ALIGN the single-modality embeddings spaces into a cross-modal embedding space


now we're in this **alignment** part
audio_encoder
'a guitar riff'
'a piano riff'
'a drum roll'
text_encoder
audio-only
text-only
projector
(audio->shared)
projector
(text->shared)
shared audio-text

How? We need to set a training objective aka what do we want the model to learn.

Training Objective

Contrastive Loss



Pull similar pairs together,
push dissimilar pairs apart

Let's take a look at contrastive learning in the audio-only domain

https://www.v7labs.com/blog/contrastive-learning-guide

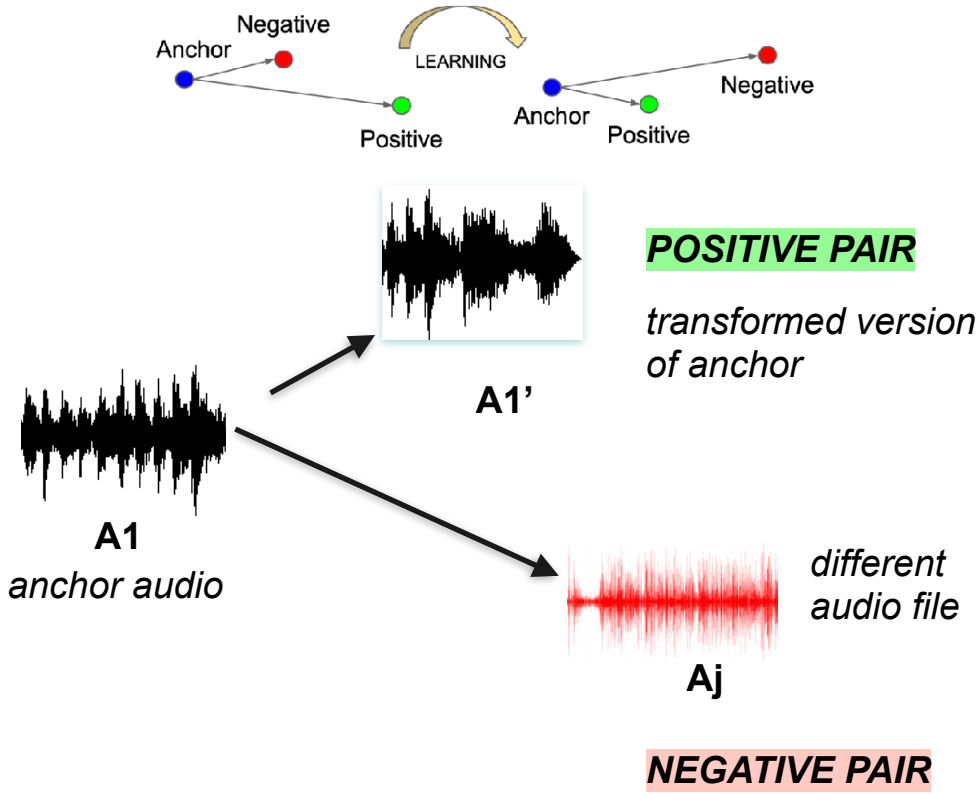
CLMR (Contrastive Learning of Musical Representations)

J. Spijkervet and J.A. Burgoyne, "Contrastive Learning of Musical Representations", in Proc. of the 22nd Int. Society for Music Information Retrieval Conf., Online, 2021

Goal: learn useful representations from musical audio (e.g., genre, instrumentation, dynamics) without paired labels

Data: (30s music clips + human-annotated tags)

- MagnaTagaTune (6.6k songs)
- MillionSongDataset (240k songs)



CLMR (Contrastive Learning of Musical Representations)

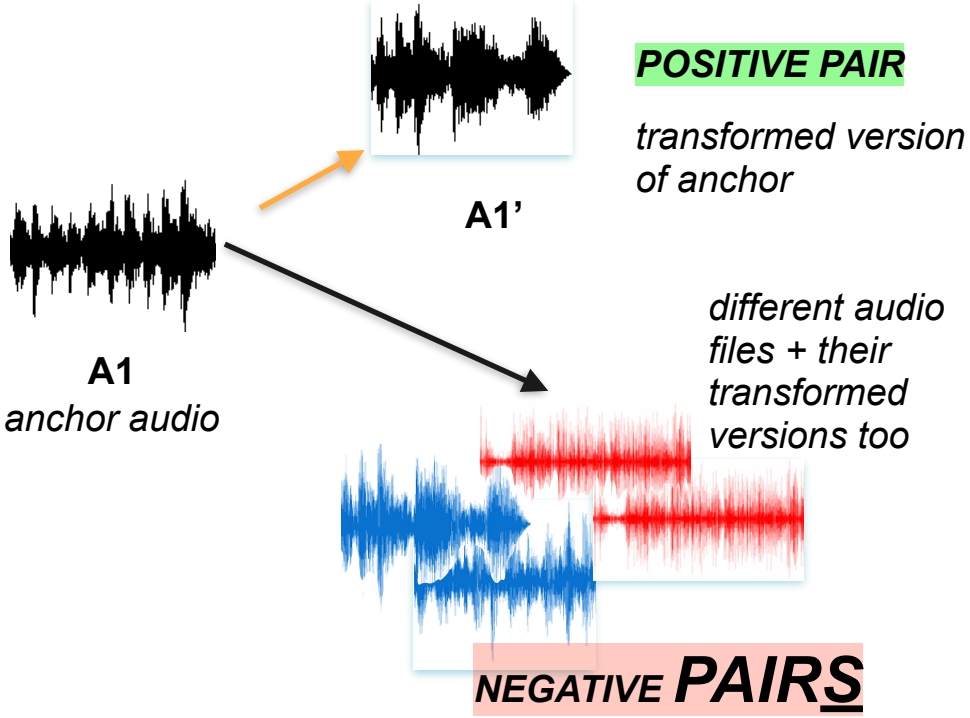
J. Spijkervet and J.A. Burgoyne, "Contrastive Learning of Musical Representations", in Proc. of the 22nd Int. Society for Music Information Retrieval Conf., Online, 2021

1 positive similar pair: via **data augmentations**

Multiple negative pairs: different audio files

Self-supervised contrastive training

Transform
Filter
Reverb
Polarity
Noise
Pitch
Gain
Delay
Crop



CLMR (Contrastive Learning of Musical Representations)

J. Spijkervet and J.A. Burgoyne, "Contrastive Learning of Musical Representations", in Proc. of the 22nd Int. Society for Music Information Retrieval Conf., Online, 2021

contrastive learning
all positive pairs "attract"
all negative pairs "repel"

encoding audio

making augmented
positive pairs

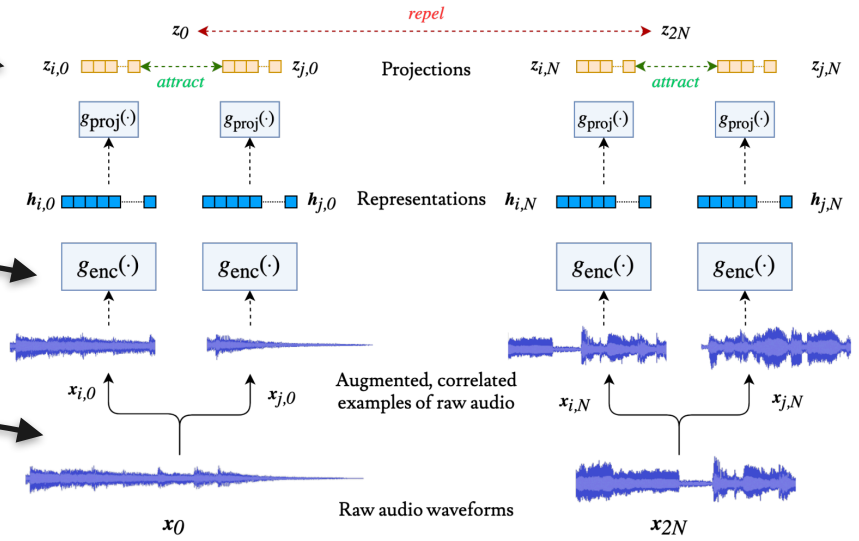


Figure 2: The complete framework operating on raw audio, in which the contrastive learning objective is directly formulated in the latent space of correlated, augmented examples of pairs of raw audio waveforms of music.

Evaluated via classification tasks

Now let's see it in action for audio <> text with CLAP

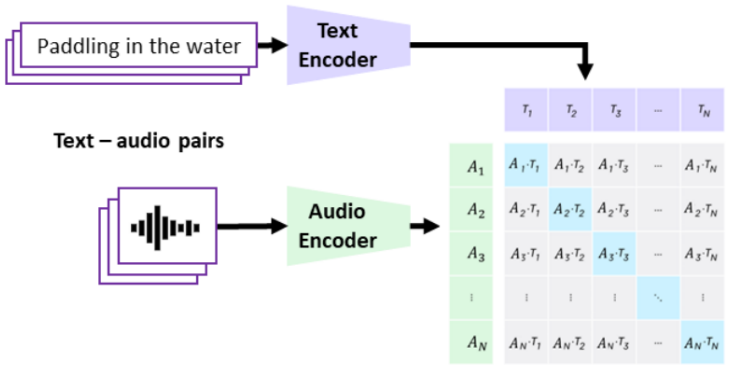
CLAP LEARNING AUDIO CONCEPTS FROM NATURAL LANGUAGE SUPERVISION

Benjamin Elizalde, Soham Deshmukh, Mahmoud Al Ismail, Huaming Wang

Microsoft

{benjaminm, sdeshmukh, malismail, huawang}@microsoft.com

Contrastive Pretraining



B. Elizalde, S. Deshmukh, M. Al Ismail, and H. Wang, "Clap: Learning audio concepts from natural language supervision," in ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2023, pp. 1–5.

CLAP Dataset

number of pairs was 119k instead of 128k. The training datasets for the 4.6M collection are: WavCaps [6], AudioSet [2], FSD50K [12], Clotho [13], AudioCaps [14], MACS [15], WavText5k [5], SoundDesc [16], NSynth [17], FMA [18], Mosi [19], Meld [20], Iemocap [21], Mosei [22], MSP-Podcast [23], CochlScene [24], LJspeech [25], EpicKitchen [26], Kinectics700 [27], findsounds.com. Details on GitHub.

4.6M audio-text pairs
mostly general sound, some speech, some music

human-annotated

`Wind and a race car make noise, with a man speaking and the sound of accelerating and tire squealing.`

WavCaps example
(pulled from AudioSet)

`A woman is rapping while a medium engine runs and a cat meows.`

AudioSet Example

inferred captions
(likely from metadata)

genre:
electronic

`This is an [electronic] song`

FMA

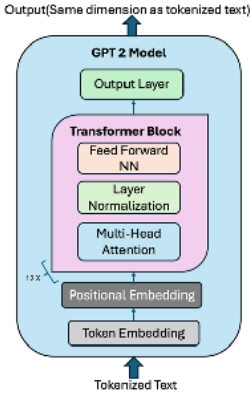
filename:
bass_synthetic
_033-047-050

`This is the sound of [synthetic bass]`

NSynth

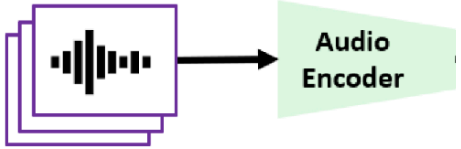
CLAP's Encoders

GPT2
(transformer)



fine-tuned only

Text – audio pairs



HTSAT
(transformer)

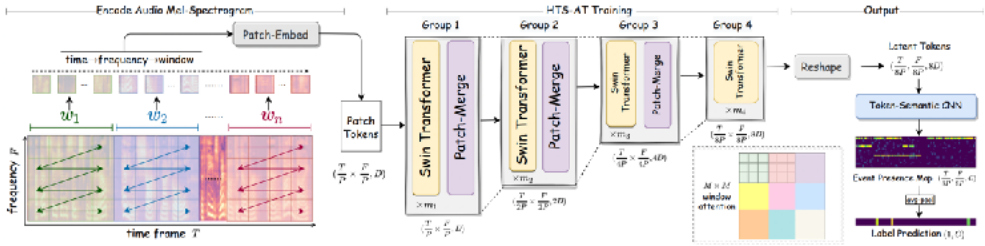


Fig. 1: The model architecture of HTS-AT.

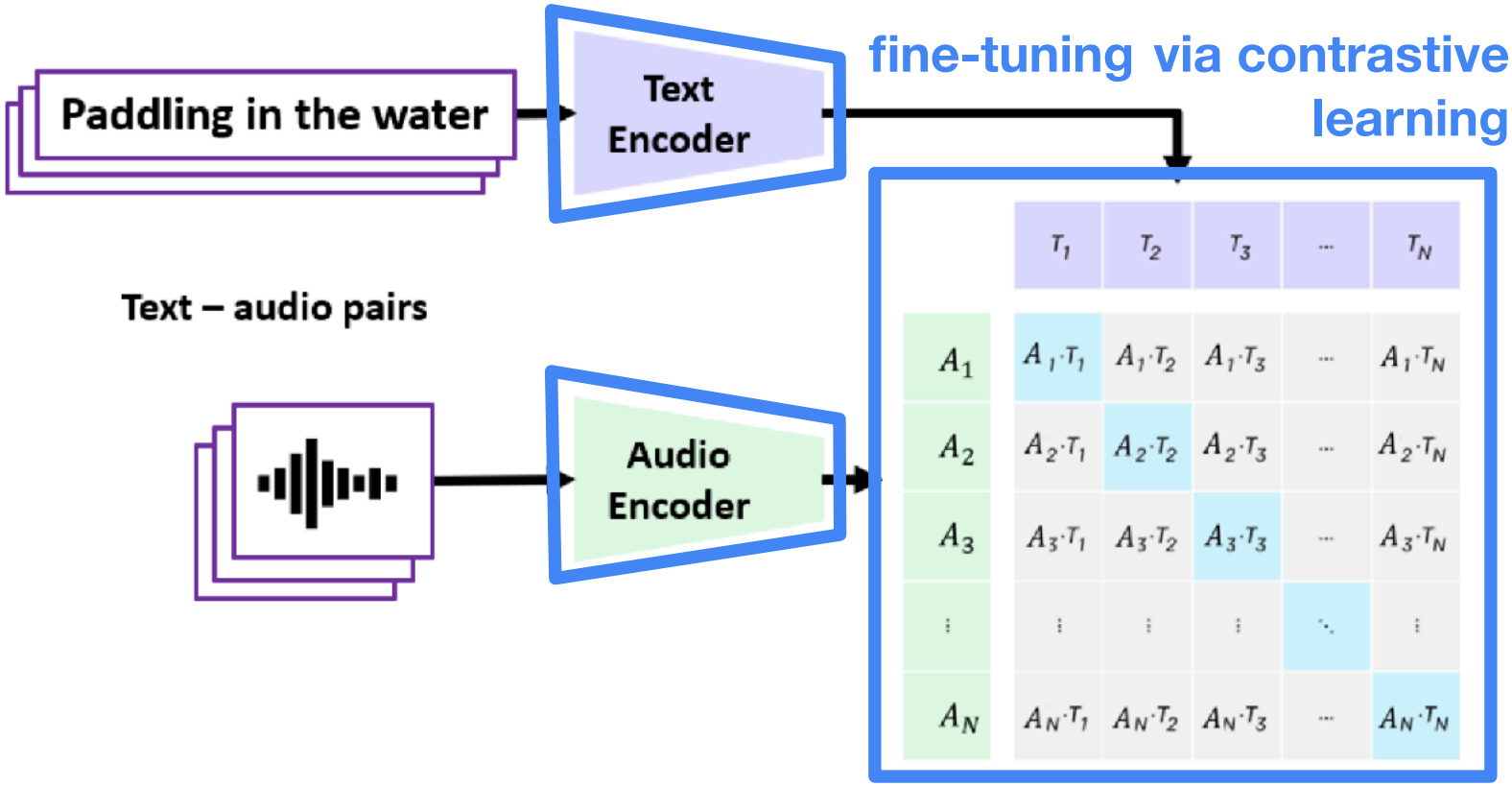
https://arxiv.org/pdf/2202.00874

trained, then fine-tuned

trained on 22 audio tasks (e.g, classification, retrieval, captioning)

Image from Elizalde, et al. arXiv:2206.04769v1

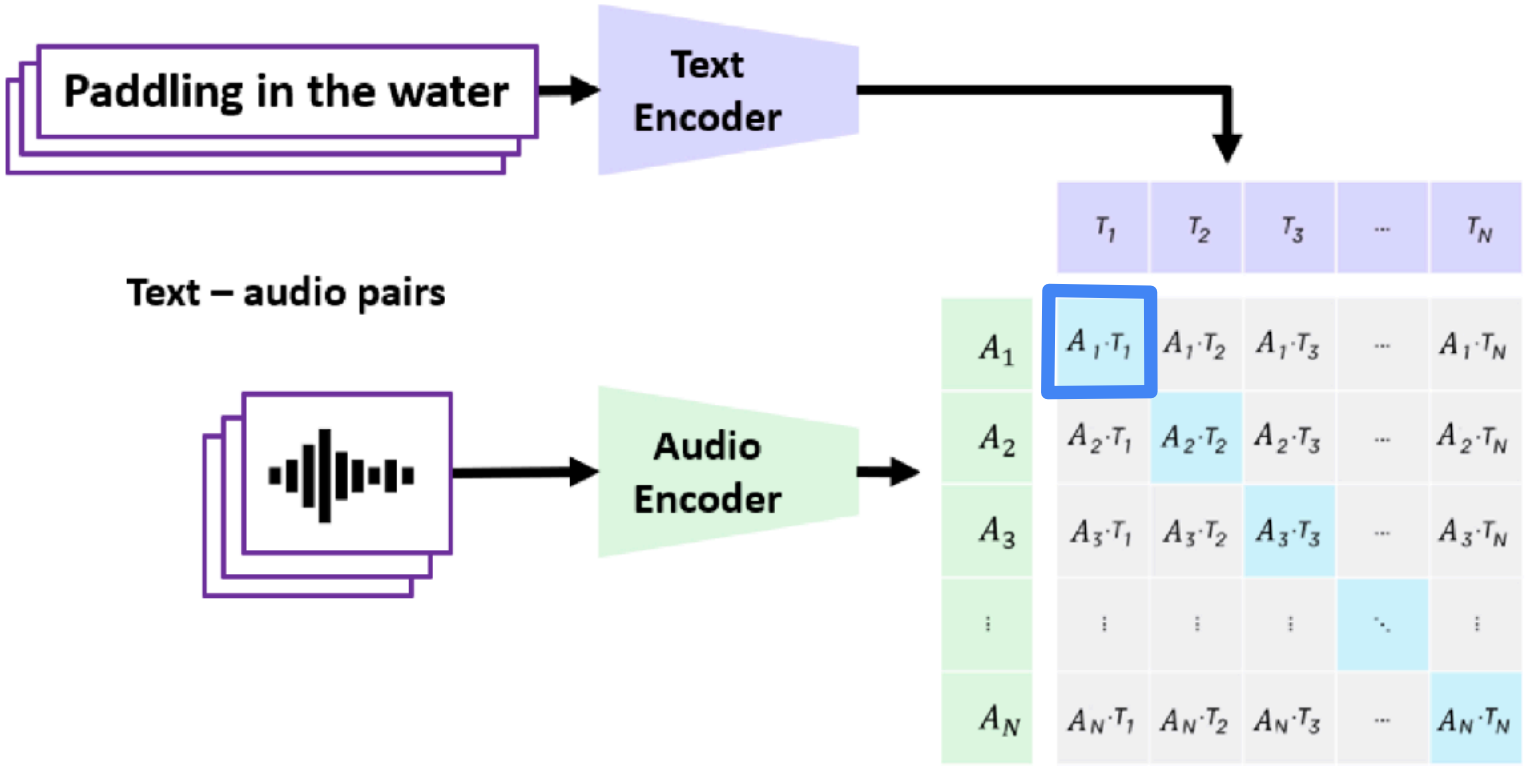
CLAP training—> fine-tuning encoders together



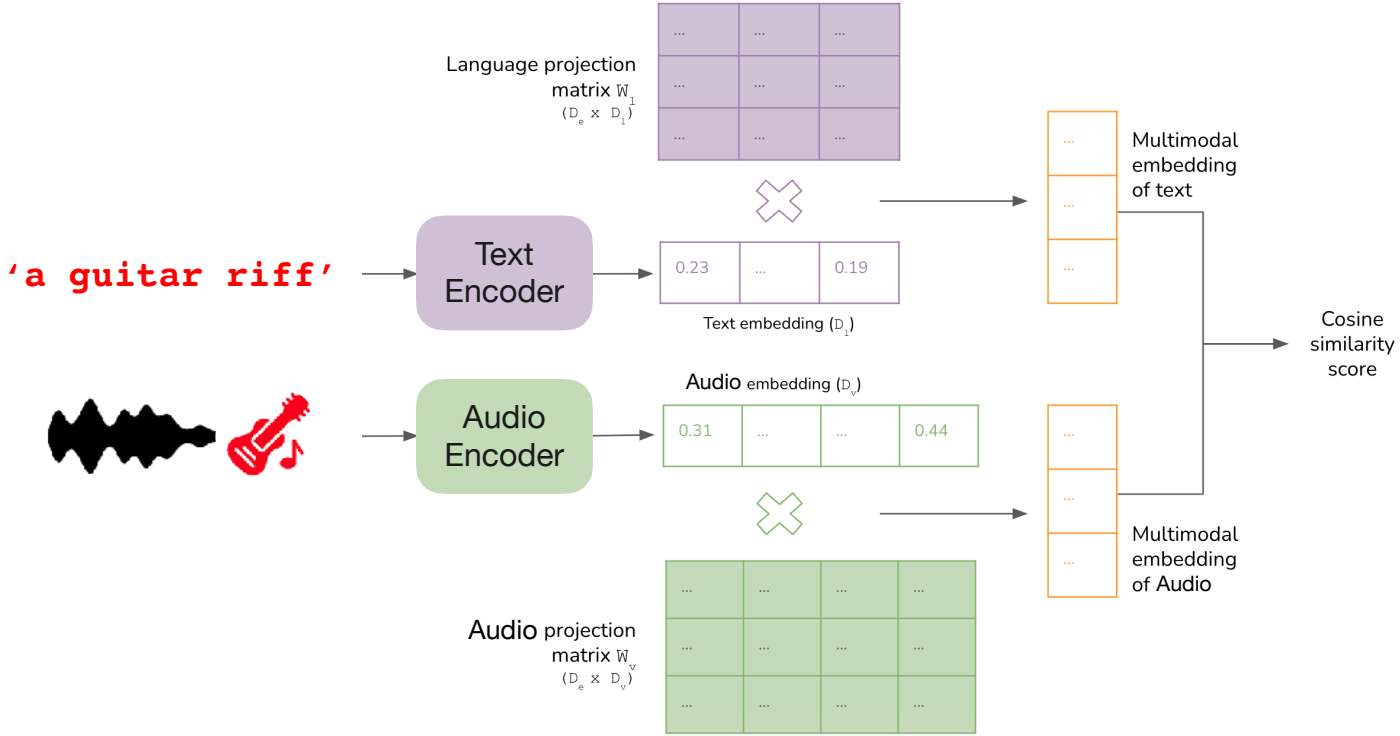
Audio is FreeSound 16770__dobroide__20060307paddling.wav

Image from Elizalde, et al. arXiv:2206.04769v1

Single Pair Example (A1, T1)



Single Pair Example (A1, T1)



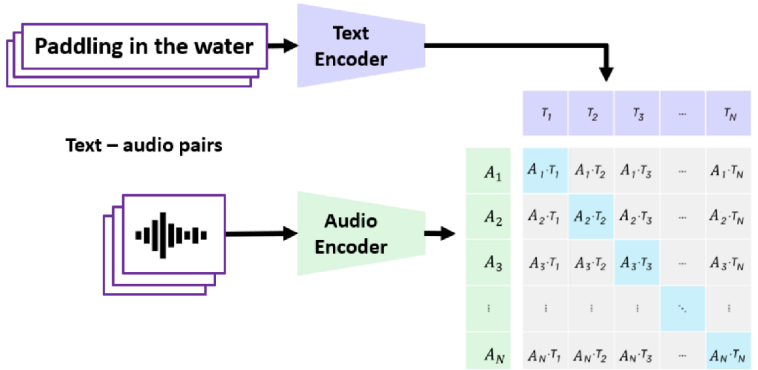
https://huyenchip.com/2023/10/10/multimodal.html

What if we do this across a batch of paired examples?

Contrastive Loss in Full Batch→ InfoNCE

For a batch of N matching text-audio pairs:

- Each **(audio, text)** pair is a **positive**.
- The remaining **N-1** text or audio items in the batch are treated as **negatives**.



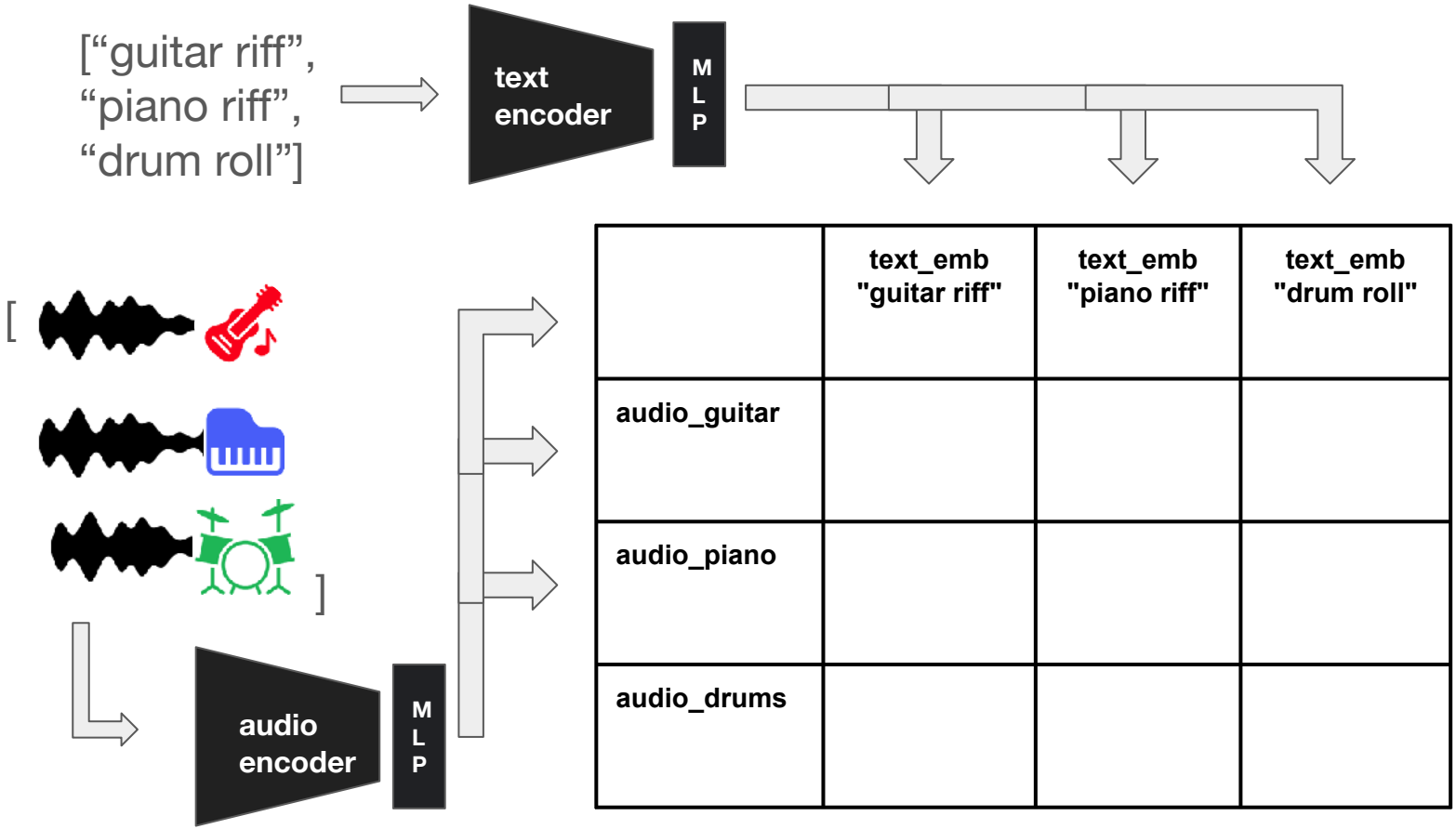
$$\mathcal{L}_i^{\text{audio}\to\text{text}} = -\log \frac{\exp(\text{sim}(f(a_i), g(t_i))/\tau)}{\sum_{j=1}^{N} \exp(\text{sim}(f(a_i), g(t_j))/\tau)}$$

InfoNCE loss

(for one audio -> text example)

We want the model to **maximize the similarity** between the matched pair and **minimize it** between unmatched ones — all within the same batch.

from 0

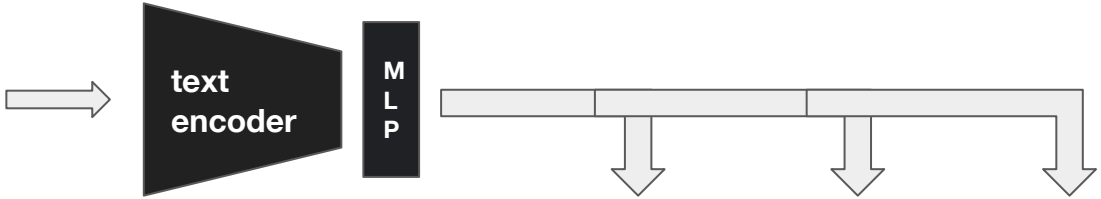


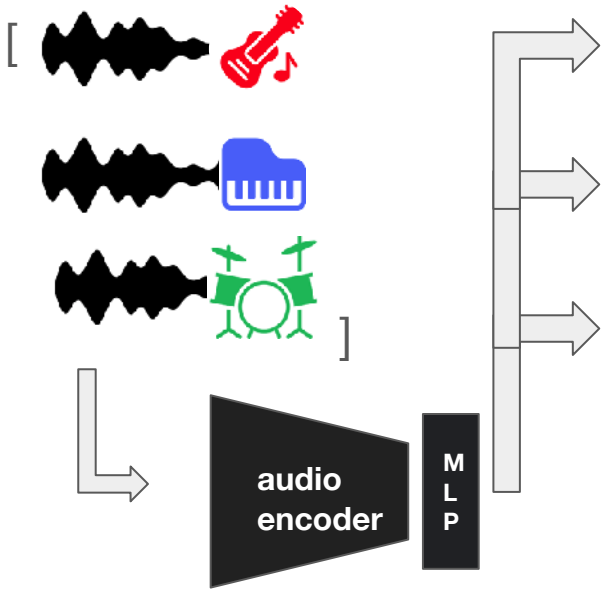
	text_emb "guitar riff"	text_emb "piano riff"	text_emb "drum roll"
audio_guitar			
audio_piano			
audio_drums			

from 0

random vals

["guitar riff",
"piano riff",
"drum roll"]

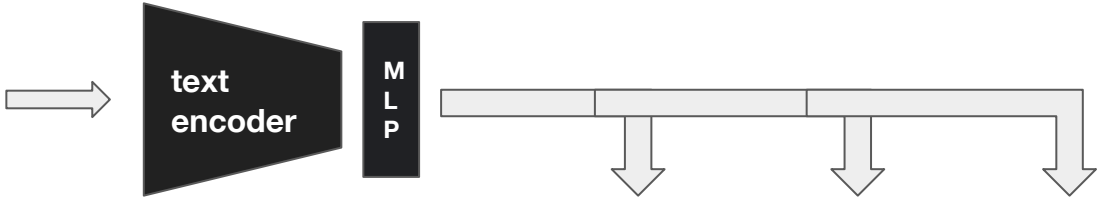


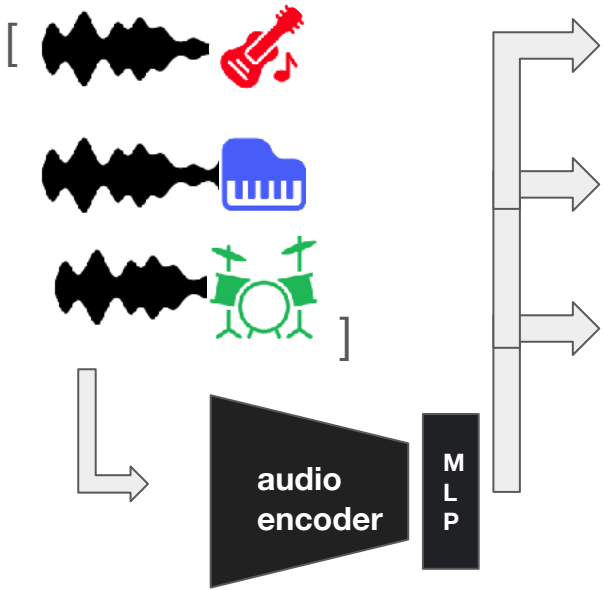


	text_emb "guitar riff"	text_emb "piano riff"	text_emb "drum roll"
audio_guitar	0.2	0.6	0.1
audio_piano	0.5	0.6	0.6
audio_drums	0.11	0.33	0.22

TARGET

[“guitar riff”,
“piano riff”,
“drum roll”]



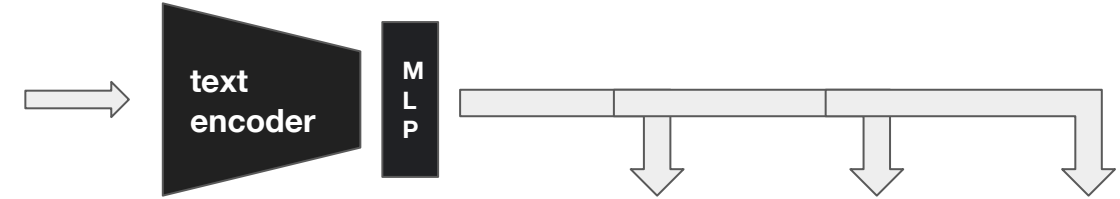


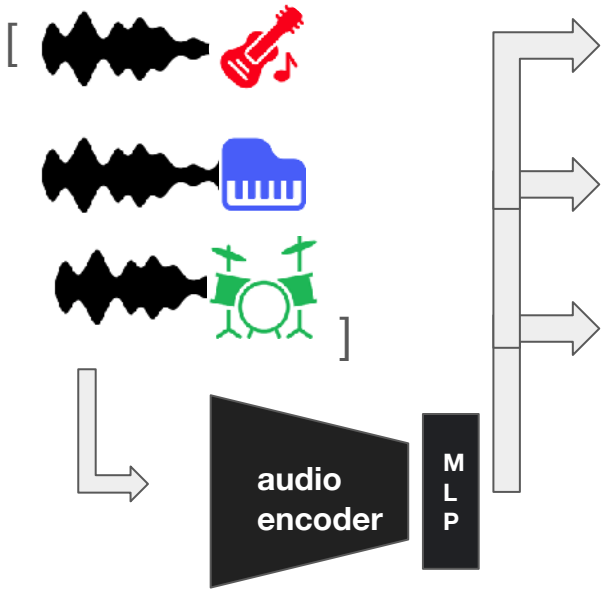
	text_emb "guitar riff"	text_emb "piano riff"	text_emb "drum roll"
audio_guitar	**1**	0	0
audio_piano	0	**1**	0
audio_drums	0	0	**1**

TARGET

Let’s just take one row

[“guitar riff”,
“piano riff”,
“drum roll”]



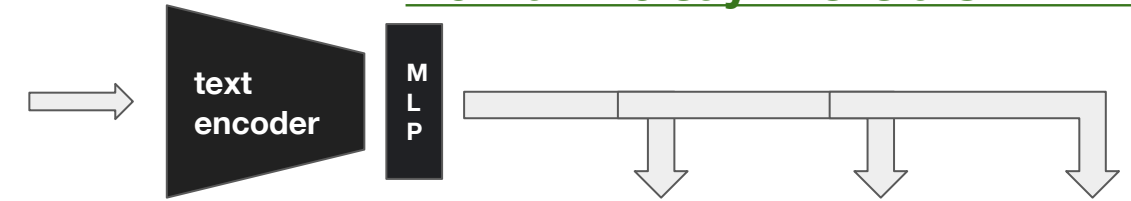


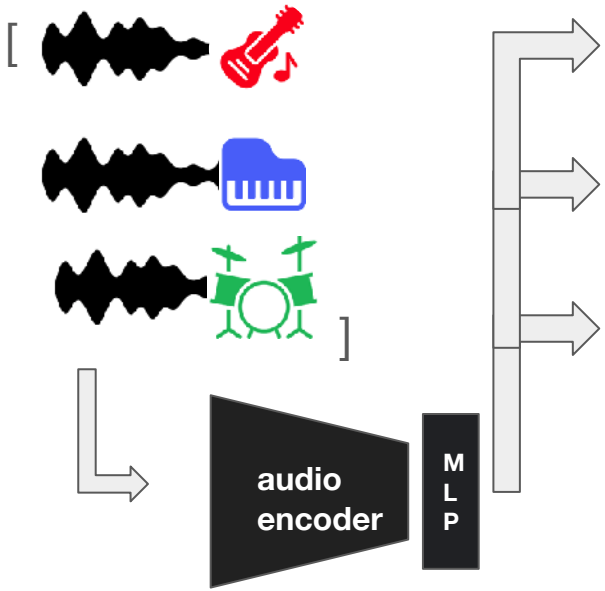
	text_emb "guitar riff"	text_emb "piano riff"	text_emb "drum roll"
audio_guitar	1	0	0
audio_piano	0	1	0
audio_drums	0	0	1

TARGET

we want to say this is a SIMILAR pair

["guitar riff",
"piano riff",
"drum roll"]

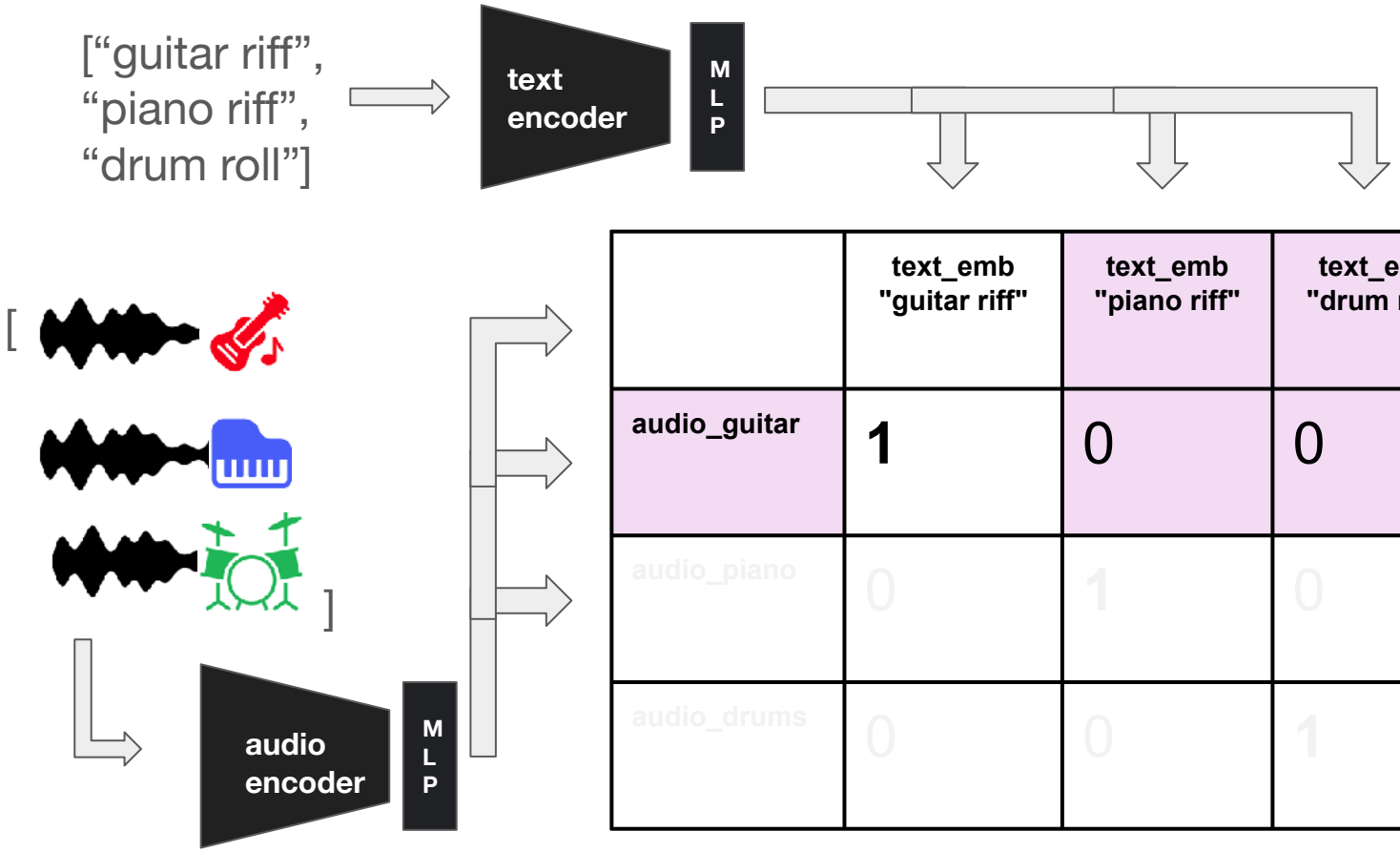




	text_emb "guitar riff"	text_emb "piano riff"	text_emb "drum roll"
audio_guitar	1	0	0
audio_piano	0	1	0
audio_drums	0	0	1

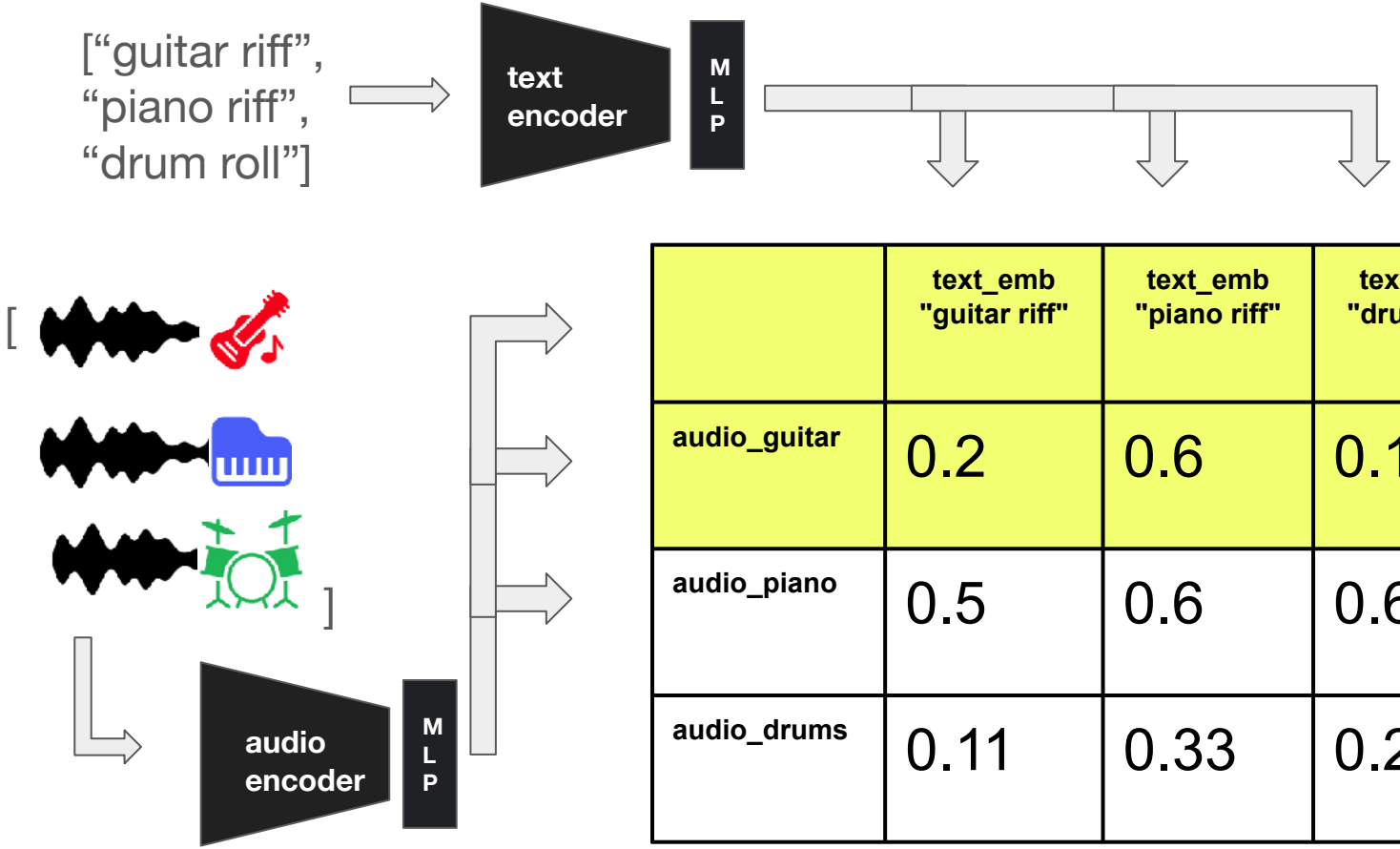
TARGET

we can also say these are DISSIMILAR pairs



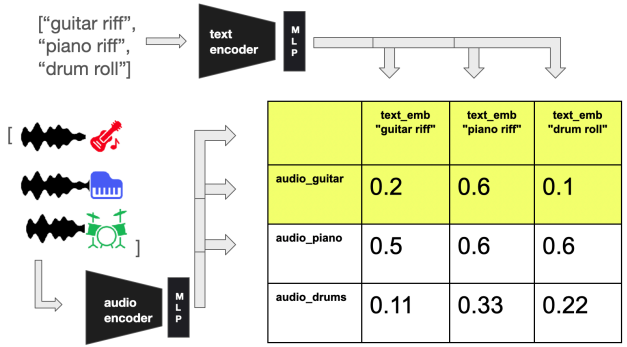
	text_emb "guitar riff"	text_emb "piano riff"	text_emb "drum roll"
audio_guitar	1	0	0
audio_piano	0	1	0
audio_drums	0	0	1

Let's calculate the InfoNCE loss at step 0 for this row (audio_guitar —> all_texts)



	text_emb "guitar riff"	text_emb "piano riff"	text_emb "drum roll"
audio_guitar	0.2	0.6	0.1
audio_piano	0.5	0.6	0.6
audio_drums	0.11	0.33	0.22

So let's calculate the InfoNCE at step 0 looking at audio = audio_guitar



	text_emb "guitar riff"	text_emb "piano riff"	text_emb "drum roll"
audio_guitar	0.2	0.6	0.1
audio_piano	0.5	0.6	0.6
audio_drums	0.11	0.33	0.22

a_i = audio_guitar
t_i = "guitar riff"

Similarity scores
sim(audio_guitar, "guitar riff") = **0.20** ← pos
sim(audio_guitar, "piano riff") = **0.60** ← neg_1
sim(audio_guitar, "drum roll") = **0.10** ← neg_2

say we set temperature T to 0.1

$$L_i^{audio \to text} = -\log\left(\frac{\exp(sim(f(a_i), g(t_i))/\tau)}{\sum_{j=1}^{N}\exp(sim(f(a_i), g(t_j))/\tau)}\right)$$

The numerator is the model's score for the correct match. We raise it to the power of 1/τ to exaggerate differences – smaller τ makes the softmax sharper

$$L_i^{audio \to text} = -\log\left(\frac{\exp(0.2/0.1)}{\exp(0.2/0.1) + \exp(0.6/0.1) + \exp(0.1/0.1)}\right)$$

This is the total score across all candidates. Since "piano riff" had a high similarity (0.6), it dominates the denominator

$$L_i^{audio \to text} = -\log\left(\frac{e^2}{e^2 + e^6 + e^1}\right) = -\log\left(\frac{7.39}{7.39 + 403.43 + 2.72}\right)$$

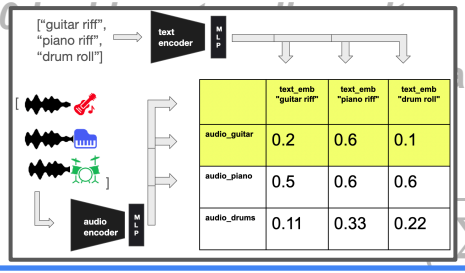
The model only assigns ~1.8% of the probability mass to the correct caption. BAD – the model isn't confident in the right answer.

$$L_i^{audio \to text} = -\log\left(\frac{7.39}{413.54}\right) = -\log(0.0179) \approx 4.02$$

*Because the model gave **higher similarity to the wrong caption (0.60)** than the correct one (0.20), the loss is **high (4.02)**.*

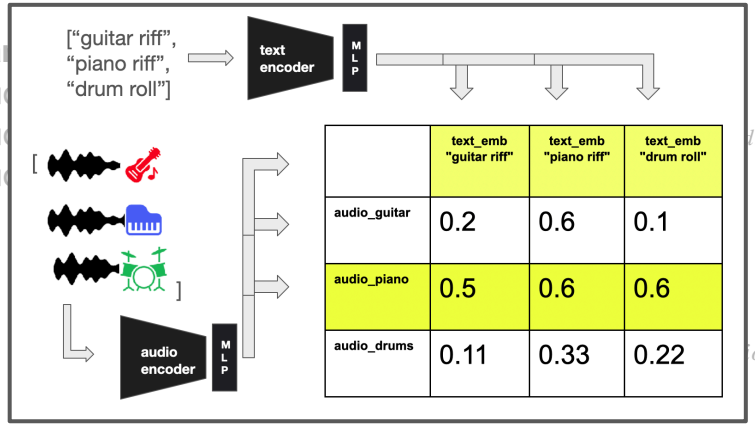
So let's calculate the InfoNCE at step

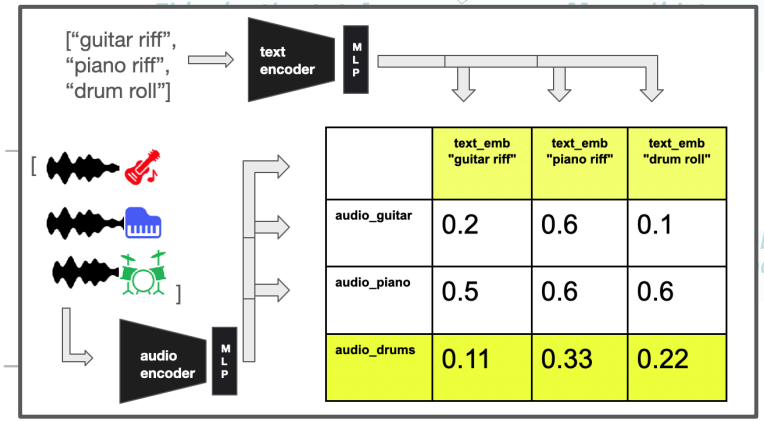
	text_emb "guitar riff"	text_emb "piano riff"	text_emb "drum roll"
audio_guitar	0.2	0.6	0.1
audio_piano	0.5	0.6	0.6
audio_drums	0.11	0.33	0.22



then we'll calculate this same loss other audios in the batch

a_i = audio_guitar
t_i = "guitar riff"

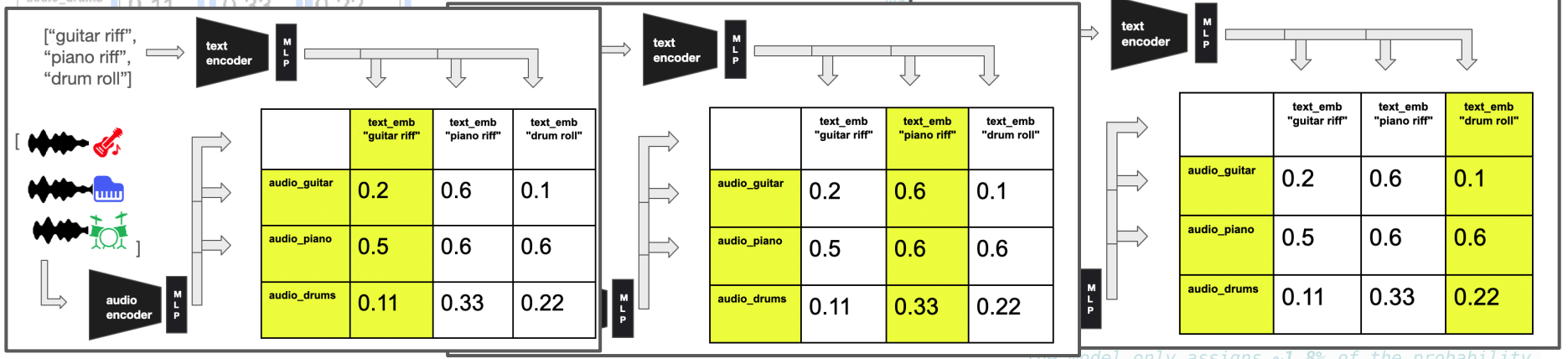




*Because the model gave **higher similarity to the wrong caption (0.60)** than the correct one (0.20), the loss is **high (4.02)**.*

So let's calculate the InfoNCE

and THEN all of them for the flip side of text -> audio



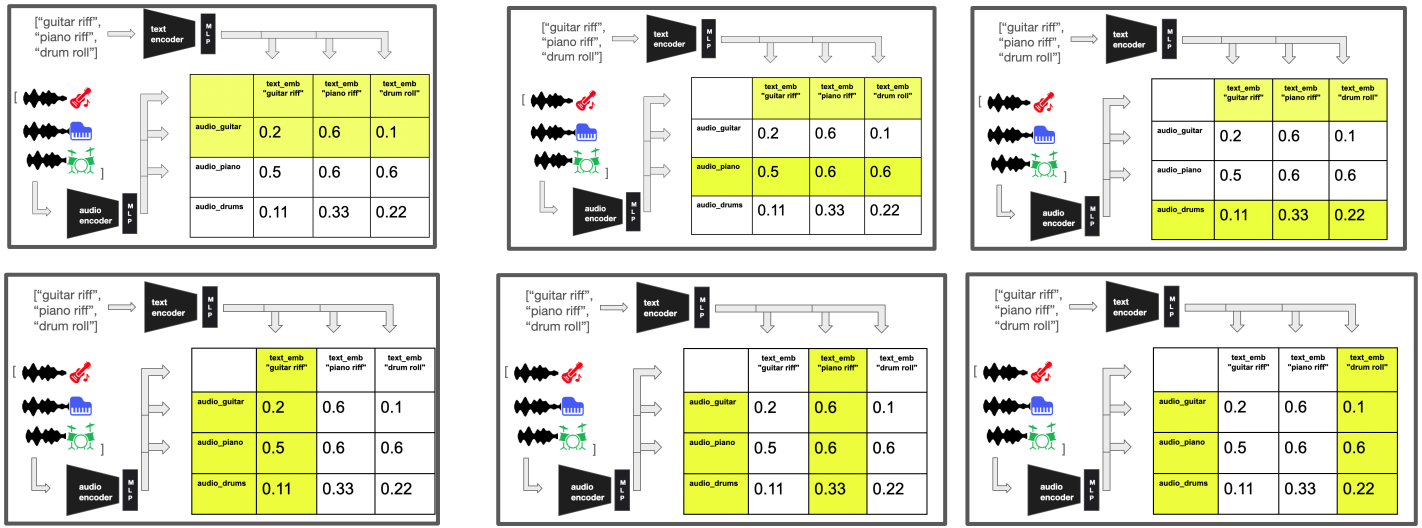
The model only assigns ~1.8% of the probability mass to the correct caption. BAD – the model isn't confident in the right answer.

$$L_i^{audio->text} = -\log\left(\frac{7.39}{413.54}\right) = -\log(0.0179) \approx 4.02$$

*Because the model gave **higher similarity to the wrong caption (0.60)** than the correct one (0.20), the loss is **high (4.02)**.*

$$L_{batch}^{audio \to text} = \sum_{i=1}^{N} L_i^{audio \to text}$$

$$L_{batch}^{text \to audio} = \sum_{i=1}^{N} L_i^{text \to audio}$$



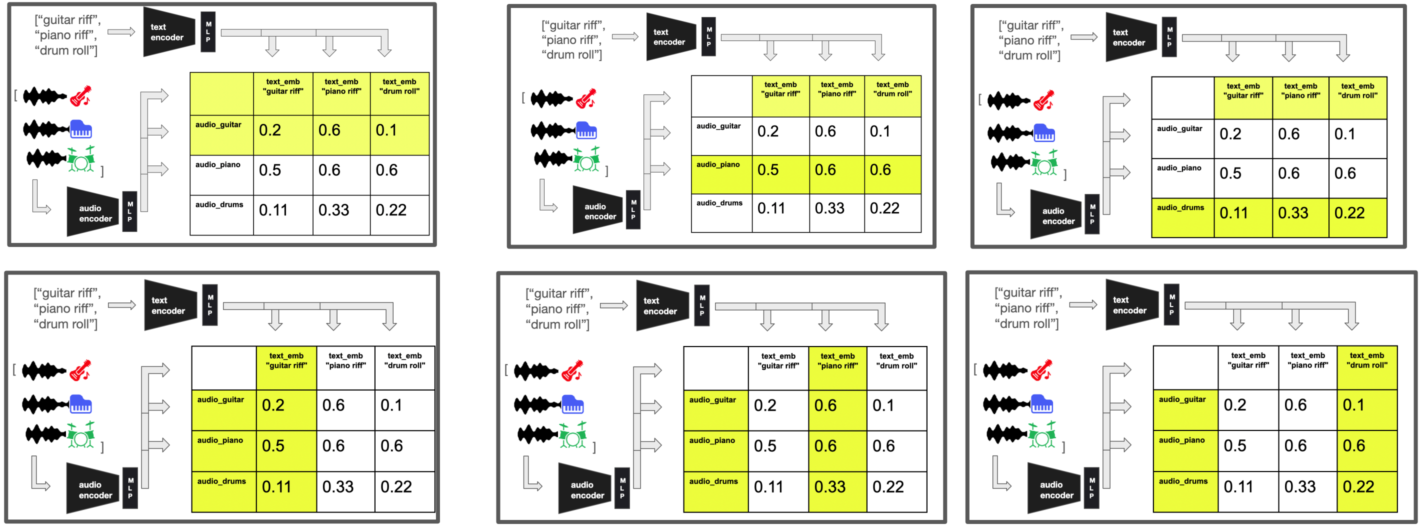
and then sum and take the average of all those losses

full InfoNCE loss at step 0

$$L_{batch} = \frac{1}{2}\left(\sum_{i=1}^{N} \mathscr{L}_i^{audio \to text} + \sum_{i=1}^{N} L_i^{text \to audio}\right)$$

$$L_{batch}^{audio\to text} = \sum_{i=1}^{N} L_i^{audio\to text}$$

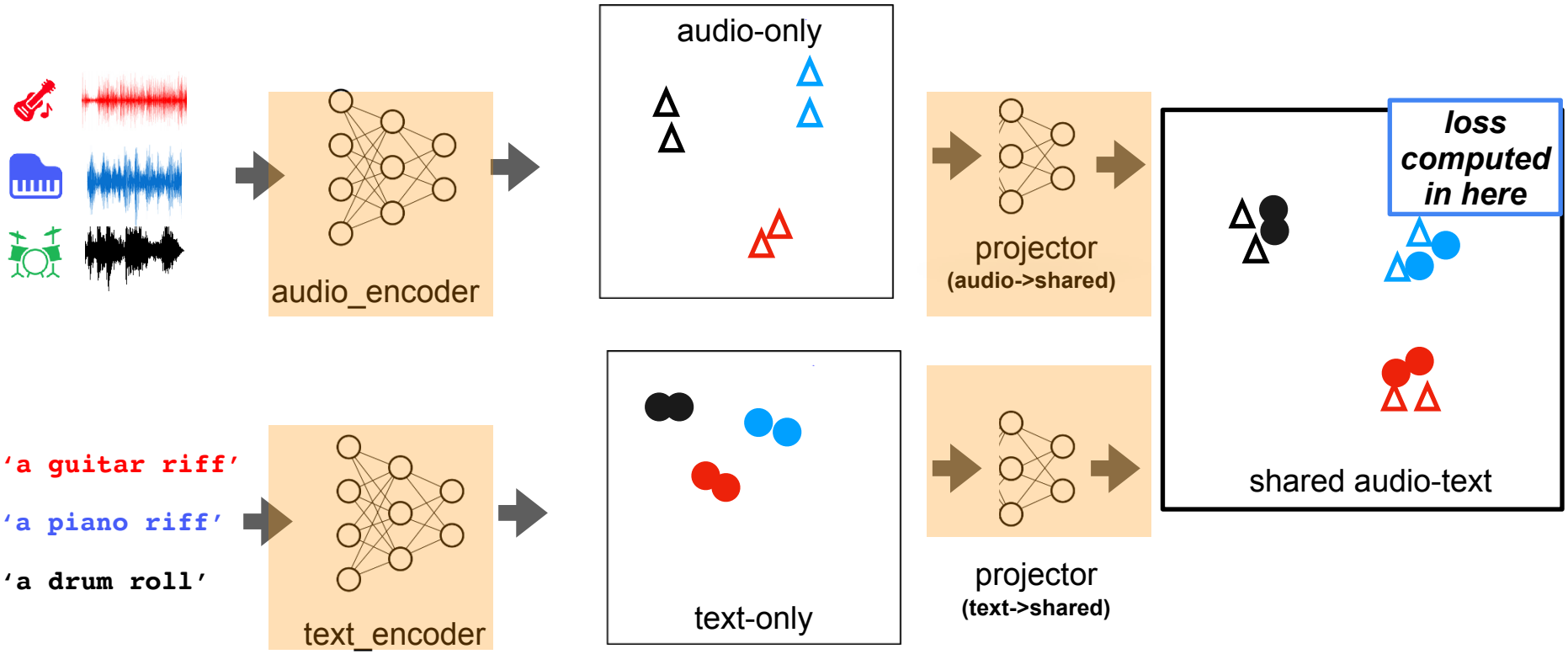
$$L_{batch}^{text\to audio} = \sum_{i=1}^{N} L_i^{text\to audio}$$



then we'll backpropagate this loss and update our weights and biases

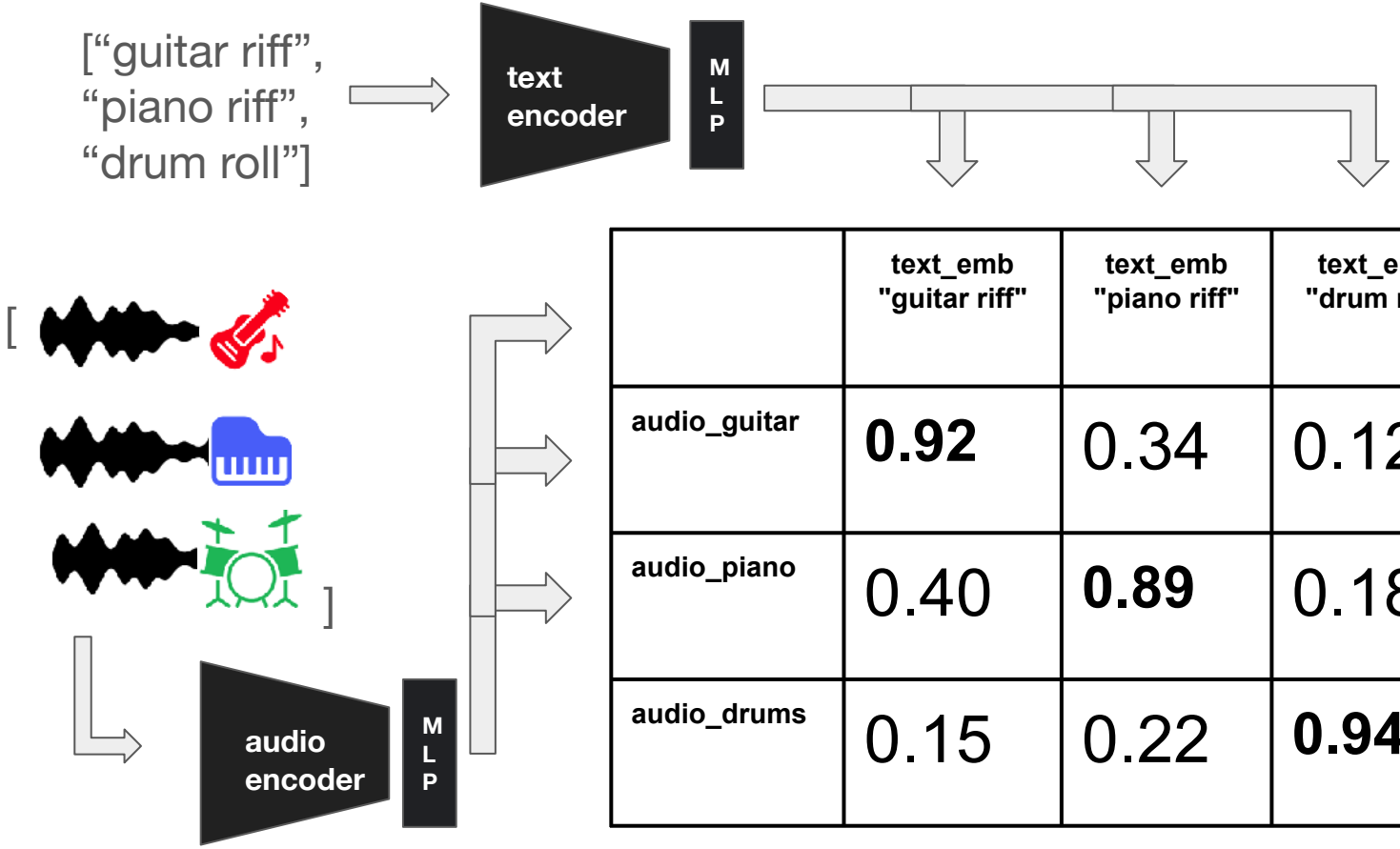
$$L_{batch} = \frac{1}{2}\left(\sum_{i=1}^{N} \mathscr{L}_i^{audio\to text} + \sum_{i=1}^{N} L_i^{text\to audio}\right)$$

Backpropagate to where? our trainable elements



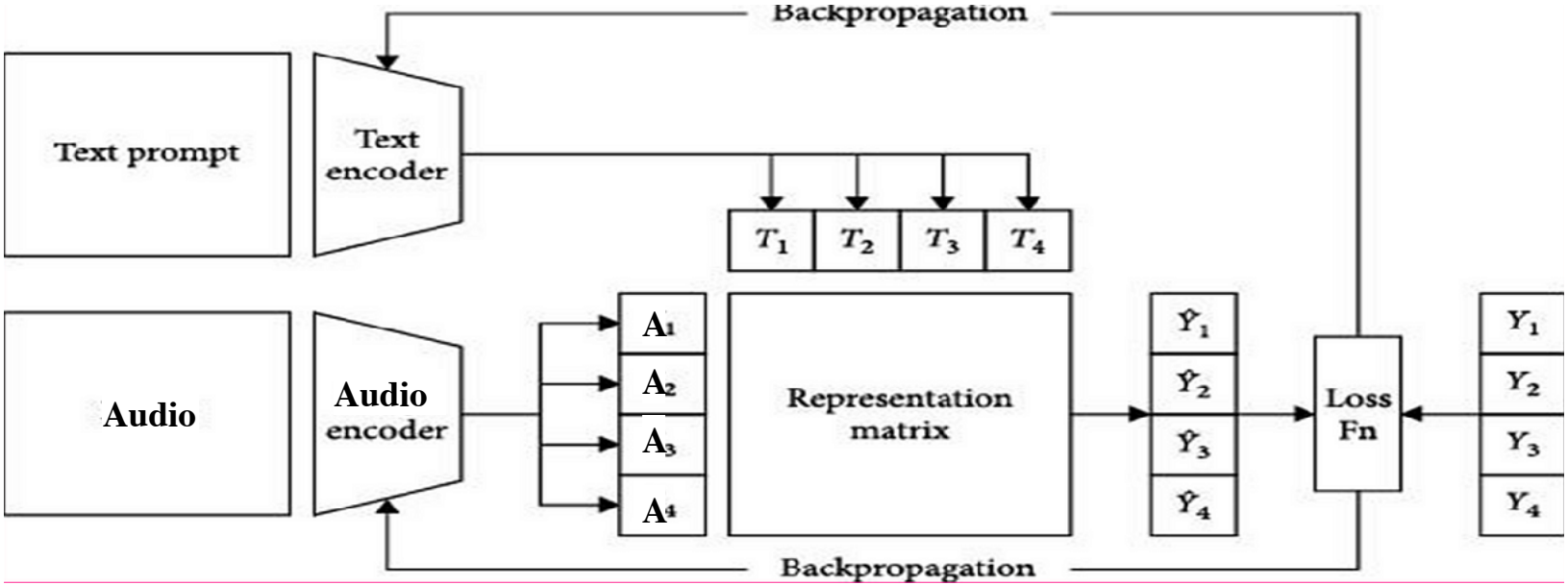
After xyz epochs of training….

AFTER SOME TRAINING (w InfoNCE loss)



	text_emb "guitar riff"	text_emb "piano riff"	text_emb "drum roll"
audio_guitar	**0.92**	0.34	0.12
audio_piano	0.40	**0.89**	0.18
audio_drums	0.15	0.22	**0.94**

Full batch training overview, N=4



Now we have something like this

Content

dog barking

cat meowing

guitar riff

whisper

muffled rumble

coin dropping

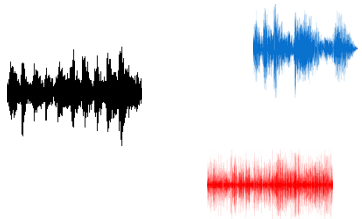
car rumbling

funky

distorted clarinet

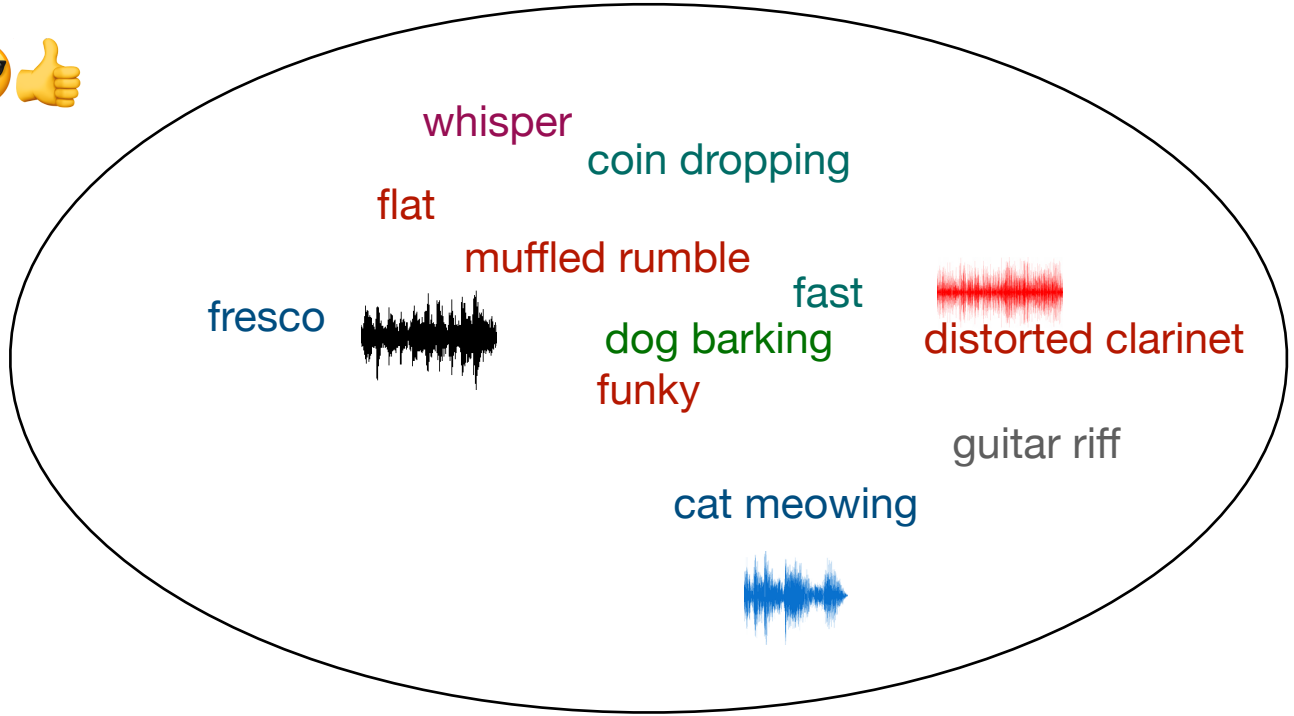
fresco

Sounds



Shared Text-Audio embedding space

nice 😎👍



✅ a joint embedding space aligning audio concepts with corresponding text ✅

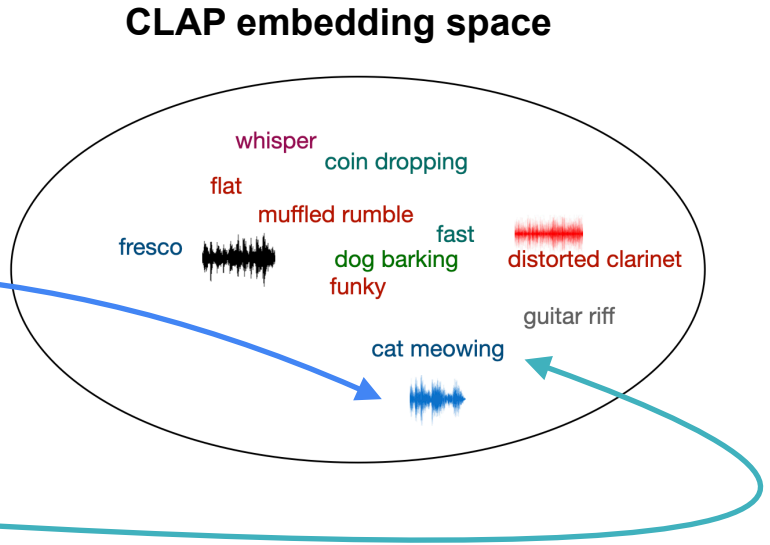
How do we evaluate multimodal embedding models?

Things we want to check

Cross-Modal Alignment (text <> audio)

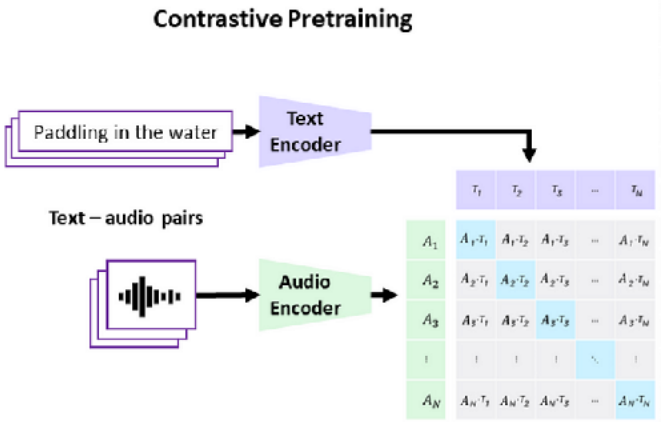
How well do the model's representations of different modalities (e.g., audio and text) align semantically?

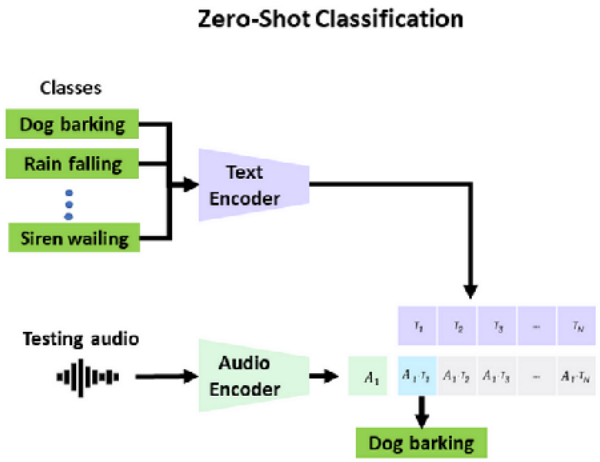
Does the **sound of cat meow** have a high similarity score with the text **“cat meowing”**?

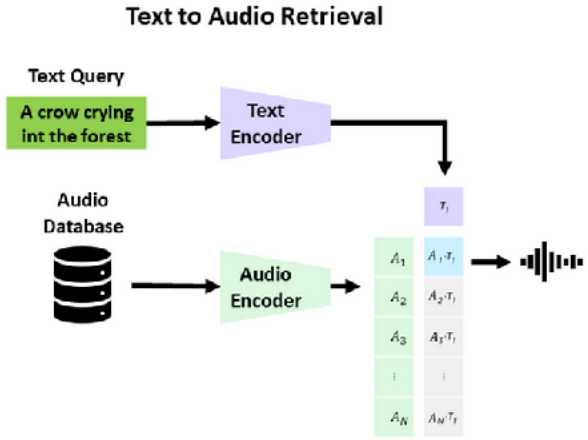


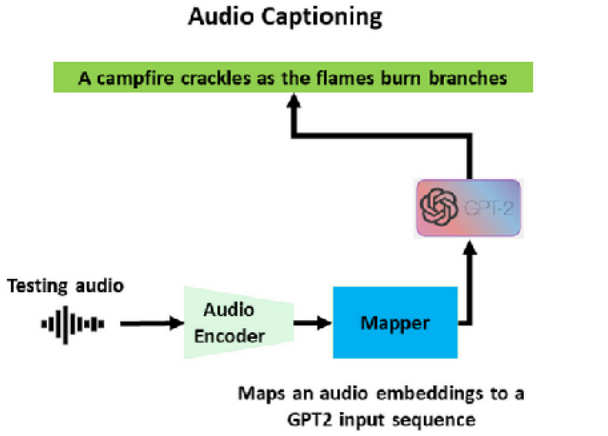
we can test this on core downstream tasks

CLAP core downstream tasks









Downstream Tasks

Task Type	Description	Common Metrics
Cross-Modal Retrieval	Match one modality (e.g., audio, image) to another (e.g., text) — e.g., "find the caption for this sound"	Recall@K, Precision@K, Median Rank, mAP
Classification	Predict labels (e.g., “guitar”, “clapping”) using single-modality embeddings with optional fine-tuning	Top-1 Accuracy, F1-score, Precision, Recall
Captioning / Generation	Generate descriptive text from audio, image, or video	BLEU, ROUGE, CIDEr, METEOR, SPICE
Auditory QA (VQA/ AQA)	Answer multiple choice questions based on auditory input and associated text	QA Accuracy, Exact Match, VQA Score
Zero-Shot Learning	Perform tasks with no labeled examples — often via alignment in shared embedding space	Accuracy, F1-score, Recall@K (task-dependent)
Human Evaluation	Collect subjective ratings of match quality, fluency, or semantic correctness	Relevance, Fluency, Preference Scores, Likert Ratings

QA benchmarks

e.g. MuChoMusic

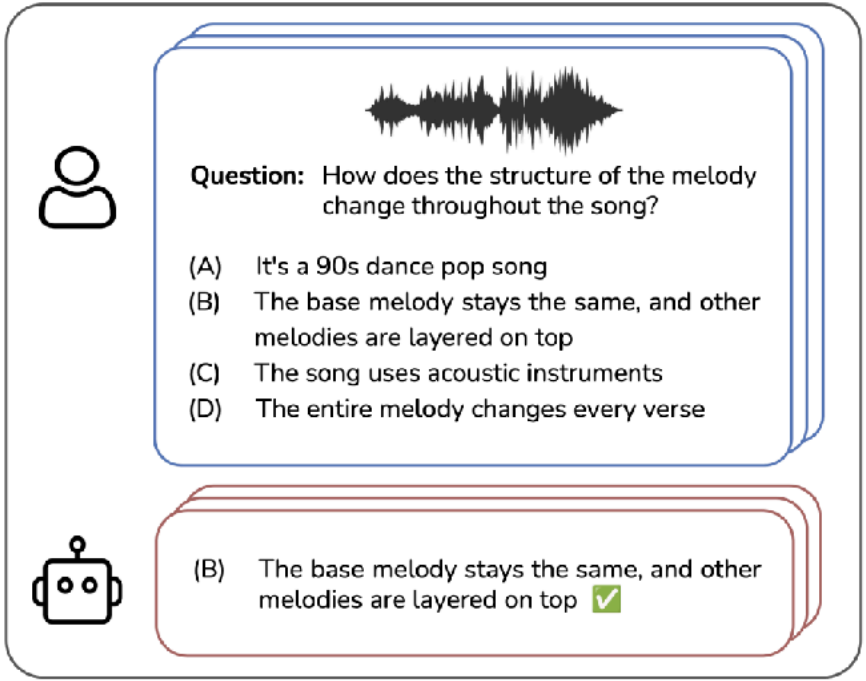


Figure 1. Multiple-choice questions in MuChoMusic have four answer options of different levels of difficulty.

B. Weck, I. Manco, E. Benetos, E. Quinton, G. Fazekas, and D. Bogdanov, "MuchoMusic: Evaluating music understanding in multimodal audio-language models," arXiv preprint arXiv:2408.01337, 2024.

Cross-Modal Retrieval

Task: Given audio, retrieve the matching text, or vice versa

rank-agnostic metrics

quality $\text{Precision@K} = \frac{\text{Number of relevant items in top K}}{K}$

ex. Precision@5: 60% of top 5 retrieved results are relevant

coverage $\text{Recall@K} = \frac{\text{Number of relevant items in top K}}{\text{Total number of relevant items in dataset for the query}}$

ex. Recall@1 = 70% means correct text is top result 70% of the time

both $\text{F1@K} = 2 \times \frac{\text{Precision@K} \times \text{Recall@K}}{\text{Precision@K} + \text{Recall@K}}$

Cross-Modal Retrieval: Precision@K

Task: Given text query, retrieve matching audio clips

Text Query

`"a flock of birds chirping in the morning"`

Rank	Retrieved Audio Label	Relevant (bird-related)?
1	`Birds chirping in forest`	YES
2	`City traffic and sirens`	no
3	`Seagulls near the ocean`	yes
4	`Children playing at a park`	no
5	`Songbirds in early morning`	yes

Out of the results it retrieved, how many were actually relevant?

Relevant items in Top 5: 3
Total retrieved (K): 5

$$\text{Precision@K} = \frac{\text{Number of relevant items in top K}}{K}$$

$$\text{Precision@5} = \frac{3}{5} = 0.6 \text{ or } 60\%$$

Cross-Modal Retrieval: Recall@K

Task: Given text query, retrieve matching audio clips

Text Query

`"a flock of birds chirping in the morning"`

Rank	Retrieved Audio Label	Relevant (bird-related)?
1	Birds chirping in forest	YES
2	City traffic and sirens	no
3	Seagulls near the ocean	yes
4	Children playing at a park	no
5	Songbirds in early morning	yes

Out of all the relevant items in the dataset, how many did it manage to retrieve?

From ground truth metadata, say we know there are 10 total bird-related audio clips

Relevant items retrieved: 3
Total relevant items in dataset: 10
Total retrieved (K): 5

$$\text{Recall@K} = \frac{\text{Number of relevant items in top K}}{\text{Total number of relevant items in dataset for the query}}$$

$$\text{Recall@5} = \frac{3}{10} = 0.3 \text{ or } 30\%$$

*Only 3 of the 10 possible bird-related audio clips were retrieved in the top 5. So while **Precision@5 was 60%, Recall@5 is only 30%** — showing that although our top results were reasonably accurate, the system **missed many other relevant clips** in the dataset.*

Cross-Modal Retrieval: F1@K

Task: Given text query, retrieve matching audio clips

Text Query

`"a flock of birds chirping in the morning"`

Rank	Retrieved Audio Label	Relevant (bird-related)?
1	`Birds chirping in forest`	YES
2	`City traffic and sirens`	no
3	`Seagulls near the ocean`	yes
4	`Children playing at a park`	no
5	`Songbirds in early morning`	yes

How balanced was the system's accuracy and coverage?

$$\text{F1@K} = 2 \cdot \frac{\text{Precision@K} \cdot \text{Recall@K}}{\text{Precision@K} + \text{Recall@K}}$$

$$\text{F1@5} = 2 \cdot \frac{0.6 \cdot 0.3}{0.6 + 0.3} = \frac{0.36}{0.9} = 0.4$$

Cross-Modal Retrieval

Task: Given audio, retrieve the matching text, or vice versa

rank-based metrics

Mean reciprocal rank

$$\mathrm{MRR} = \frac{1}{Q}\sum_{q=1}^{Q}\frac{1}{\text{rank of first relevant item for query } q}$$

what rank was the first relevant item?

avg precision (per query)

$$\mathrm{AP} = \frac{1}{R}\sum_{k=1}^{N}\mathrm{Precision@k}\cdot 1[\text{item at } k \text{ is relevant}]$$

Mean AP

$$\mathrm{mAP} = \frac{1}{Q}\sum_{q=1}^{Q}\mathrm{AP}_q$$

R: Total number of relevant items for the query
N: Total number of returned items (can be all or top-K)
1[·]: 1 if item is relevant, 0 otherwise

do relevant items appear fairly early in the ranked list?

Cross-Modal Retrieval: MRR

Task: Given text query, retrieve matching audio clips

Text Query

`"a flock of birds chirping in the morning"`

Rank	Retrieved Audio Label	Relevant (bird-related)?
1	`Birds chirping in forest`	YES
2	`City traffic and sirens`	no
3	`Seagulls near the ocean`	yes
4	`Children playing at a park`	no
5	`Songbirds in early morning`	yes

How soon was the first relevant item retrieved?

First relevant rank: 1

$$\text{MRR} = \frac{1}{1} = 1.0$$

(If first relevant was at rank 3, MRR = 1/3 ≈ 0.3)

Cross-Modal Retrieval: Average Precision

Task: Given text query, retrieve matching audio clips

Text Query

`"a flock of birds chirping in the morning"`

Rank	Retrieved Audio Label	Relevant (bird-related)?
1	Birds chirping in forest	YES
2	City traffic and sirens	no
3	Seagulls near the ocean	yes
4	Children playing at a park	no
5	Songbirds in early morning	yes

Where did the relevant results appear in the ranking?

Relevant ranks: 1,3,5

Calculate precision at every relevant position
P@1 = 1/1 = 1
P@3 = 2/3 = 0.67
P@5 = 3/5 = 0.6

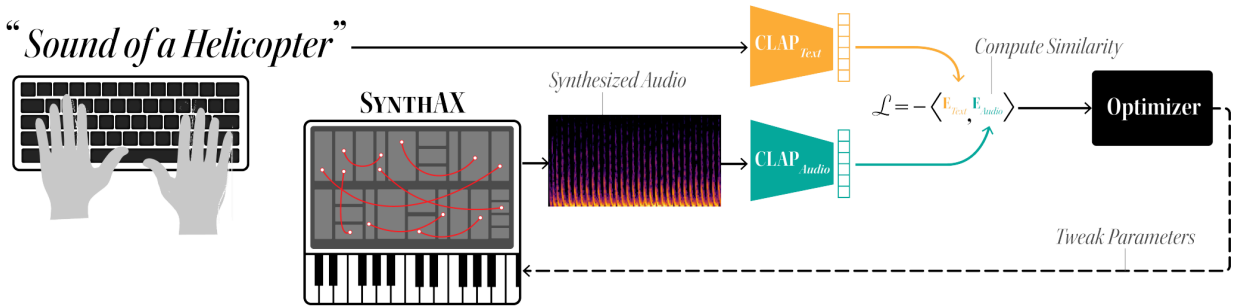
$$\text{AP} = \frac{1.0 + 0.667 + 0.6}{3} \approx 0.756$$

if we had multiple queries, we'd do the same for them then take avg for mAP

How else have people used CLAP? Fun applications

Text-to-Audio Generation: Controlling a Synthesizer

CLAP's embedding space as loss space



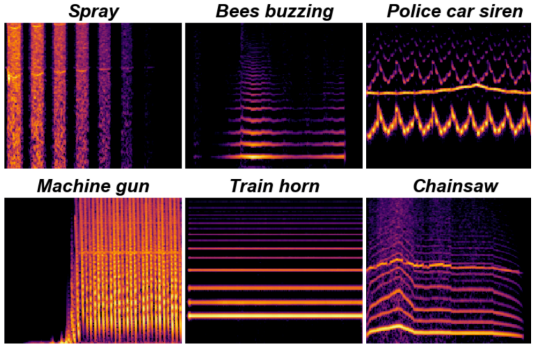


Figure 1. CTAG leverages a virtual modular synthesizer to generate sounds capturing the semantics of user-provided text prompts in a sketch-like way, rather than being acoustically literal. Spectrograms of auditory outputs corresponding to six text prompts showcase the range of sounds this approach can yield, accompanied by a fully interpretable and controllable parameter space.

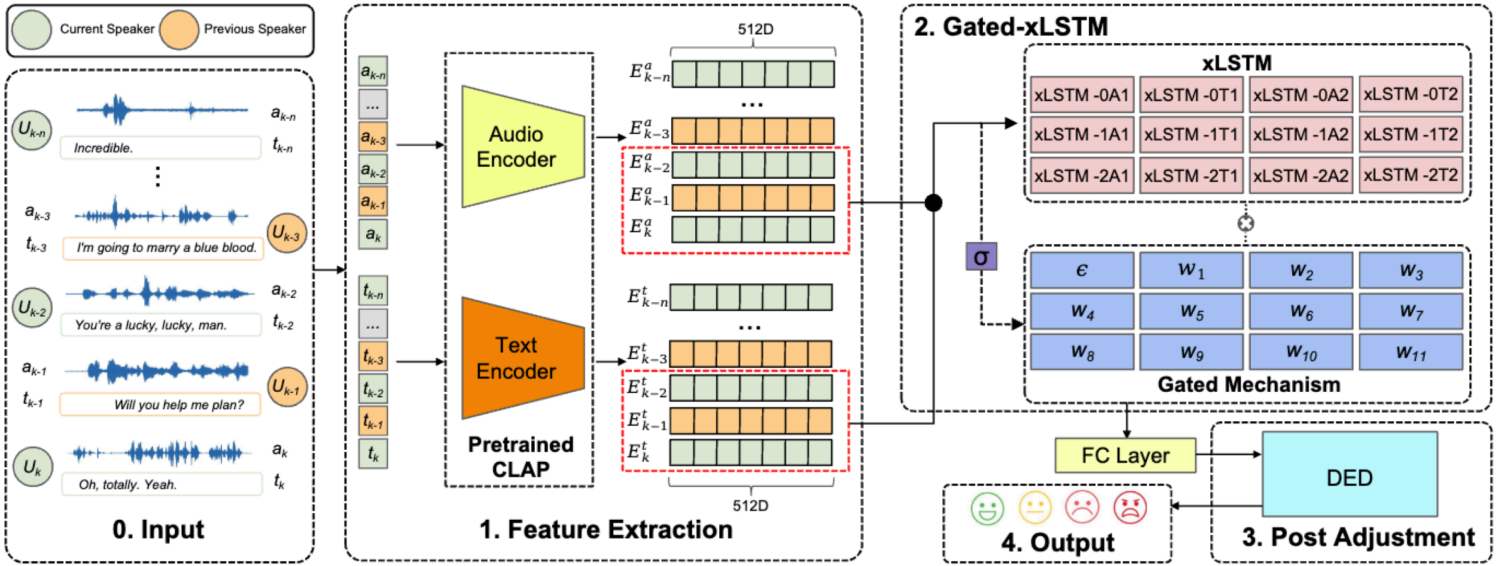
M. Cherep, N. Singh, and J. Shand, "Creative Text-to-Audio Generation via Synthesizer Programming," arXiv preprint arXiv:2406.00294, 2024. [Online]. Available: https://arxiv.org/abs/2406.00294

EXAMPLEs: https://ctag.media.mit.edu/

How else have people used CLAP? Fun applications

Speech emotion recognition

CLAP embeddings as input feature vector



Y. Li, Q. Sun, S. M. Krishna Murthy, E. Alturki, and B. W. Schuller, "GatedxLSTM: A multimodal affective computing approach for emotion recognition in conversations," arXiv preprint arXiv:2503.20919, 2025. [Online]. Available: https://api.semanticscholar.org/CorpusID:277349399

Let’s look at using CLAP for audio production

Can we take an audio source and make it sound “crunchy” or “warm”?

Text2FX

Harnessing CLAP Embeddings for Text-Guided Audio Effects

Annie Chu, Patrick O'Reilly, Julia Barnett, Bryan Pardo




Text2FX: Harnessing CLAP Embeddings for Text-Guided Audio Effects

Annie Chu
Northwestern University

Patrick O'Reilly
Northwestern University

Julia Barnett
Northwestern University

Bryan Pardo
Northwestern University

***Abstract*—This work introduces Text2FX, a method that leverages CLAP embeddings and differentiable digital signal processing to control audio effects, such as equalization and reverberation, using open-vocabulary natural language prompts (e.g., "make this sound in-your-face and bold"). Text2FX operates without retraining any models, relying instead on single-instance optimization within the existing embedding space, thus enabling a flexible, scalable approach to open-vocabulary sound transformations through interpretable and disentangled FX manipulation. We show that CLAP encodes valuable information for controlling audio effects and propose two optimization approaches using CLAP to map text to audio effect parameters. While we demonstrate with CLAP, this approach is applicable to any shared text-audio embedding space. Similarly, while we demonstrate with equalization and reverberation, any differentiable audio effect may be controlled. We conduct a listener study with diverse text prompts and source audio to evaluate the quality and alignment of these methods with human perception. Demos and code are available at anniejchu.github.io/text2fx**

***Index Terms*—intelligent audio production, audio effects, multimodal embeddings, DDSP**

I. Introduction

Audio effects (e.g., equalization, reverberation, compression) are essential tools in modern audio production. From mainstream pop to podcasts to film scores, audio effects (FX) are integral in shaping the final sound. However, their complex and often unintuitive controls (e.g., decay, cutoff frequency) can be extremely challenging for non-experts and time-consuming for professionals. For instance, despite its seemingly straightforward description, transforming a simple drum recording into the 'crunchy hyperpop' drum sound of Charli XCX may require a complex process involving the careful adjustment of over 20 distinct effect parameters across multiple FX, such as distortion, saturation, equalization, and compression.

Semantic audio production research aims to bridge the gap between *high-level concepts* (e.g., 'old time telephone') and *signal-level effect parameters* (e.g., controls of a parametric equalizer) [1]. Pre-deep-learning efforts, such as Sabin et al. [2] and Audealize [3], used crowdsourcing to map natural language terms to specific effect parameters, such as equalization (EQ) or reverberation (Reverb). While effective, these methods produced closed-vocabulary mappings limited to single FX, unable to generalize beyond new words or phrases. This work also resulted in word-parameter setting datasets for single FX, such as SocialFX [4] (EQ, Reverb, compression) and SAFE [5] (four open-source plugins). Most recently, Balasubramaniam et al. [6] explored text-driven audio manipulation by training a deep model on the EQ subset of Audealize [3]. However, as their approach focuses on text-to-audio generation rather than directly mapping text to effect parameters, it functions as a black box, limiting users' ability to shape the final result. Like earlier work, it is limited by the closed vocabulary of single-word descriptors from training. We seek to overcome these limitations by exploring method that enables open-vocabulary text prompts to control any set of differentiable effects without retraining for new words or FX.

This work was supported by NSF Award Number 2222369.

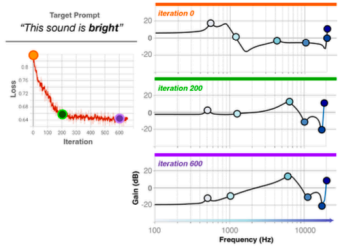


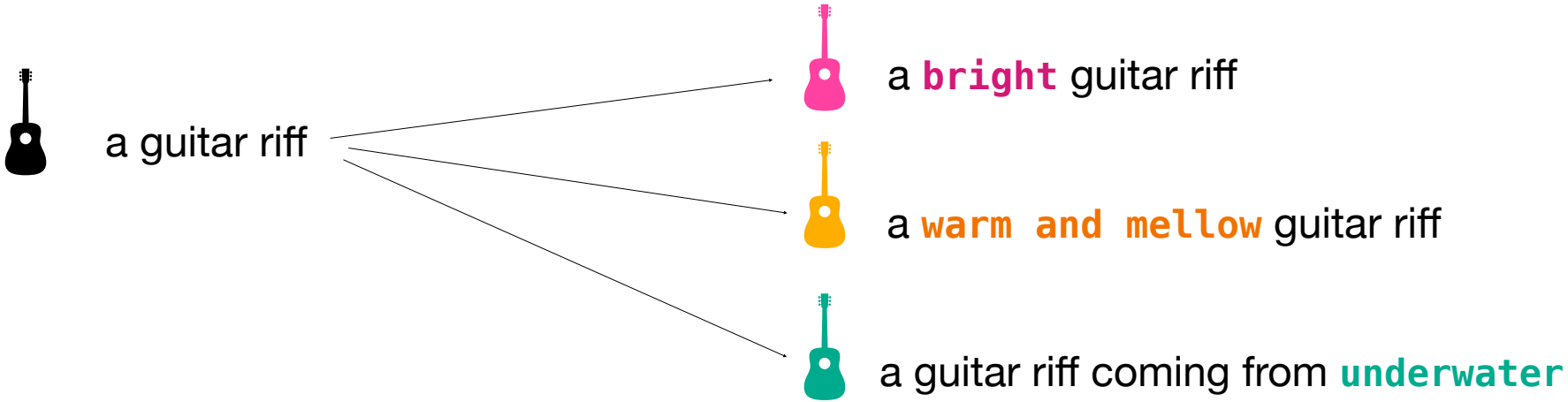
Fig. 1. **Text2FX Example.** A previous study [3] found listeners associate 'bright' with boosting high frequencies (> 2 kHz) and cutting low ones (< 2 kHz). Optimizing the audio in a shared text-audio embedding space (CLAP) towards the embedding for text 'bright' achieves this;. **Left:** Optimization loss curve. **Right:** Estimated settings for a 6-band parametric EQ.

Recent large multimodal embedding models like CLAP [7] have made great strides in bridging natural language with audio. Trained on a diverse, extensive dataset of paired audio-text captions, CLAP features a joint embedding space aligning audio with corresponding textual descriptions. Though successfully applied to zero-shot classification and audio captioning [7], as well as text-to-audio generation [8], CLAP's ability to encode qualitative notions of audio FX—such as what constitutes a 'bright' sound— remains unexplored.

Differentiable digital signal processing (DDSP) [9, 10] allows traditional DSP parameters (e.g., filter coefficients, gain controls, and synthesis parameters) to be learned through gradient-based optimization. DDSP has been successfully applied in tasks including speech synthesis [11], synthesizer-based sound generation [8], style transfer for audio FX [12], and mastering [13], but has not been applied to text-driven audio FX.

In this paper, we explore whether CLAP embeddings contain actionable knowledge for natural language-based control of audio FX. To leverage this knowledge, we introduce Text2FX, a method that uses CLAP's learned representations to manipulate audio FX through cross-modal optimization. Integrating CLAP with DDSP, Text2FX performs single-instance optimization within the audio FX parameter space, aligning the audio embedding with that of a given text description. Given an audio recording, a prompt (e.g., 'shrill and sharp'), and an FX chain (i.e., sequence of audio FX like EQ → Reverb), Text2FX generates both the "effected" audio along with the interpretable, adjustable FX parameters applied to achieve the

Sound Semantics: How do we describe sound?



How do we make something sound `bright`?

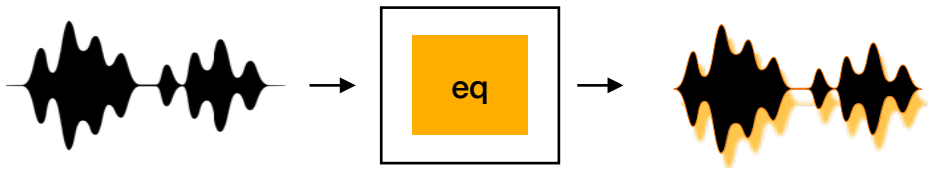
Audio Effects (FX)

Audio FX are digital signal processing (DSP) based tools used to modify sound by transforming the audio signals

Common Examples of Audio FX

Some common types of audio FX

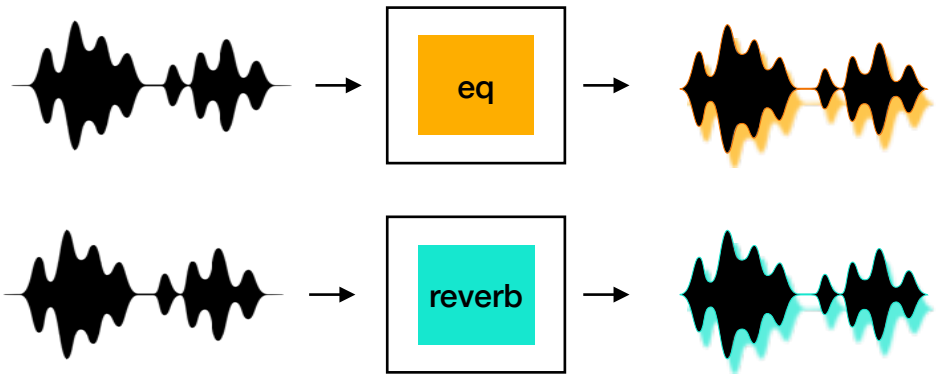
- **EQ (Equalization)** – Adjusts frequencies to balance tone



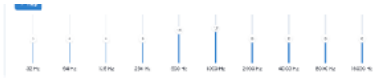
- **Reverb** – Creates a sense of space and depth.

Some common types of audio FX

- **EQ (Equalization)** – Adjusts frequencies to balance tone
- **Reverb** – Creates a sense of space and depth.



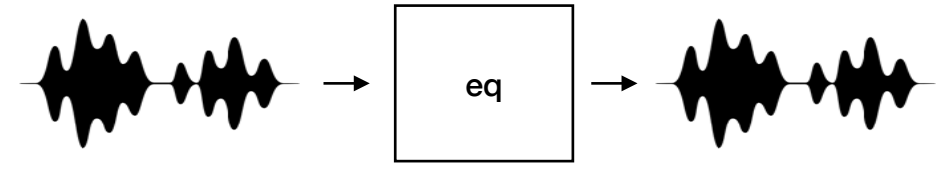
Some common types of audio FX

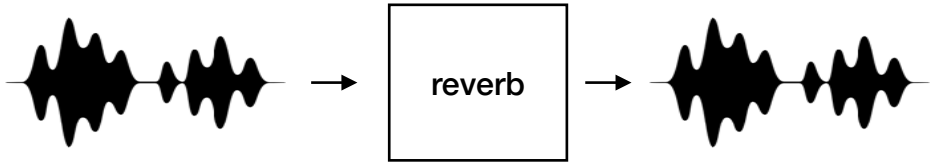


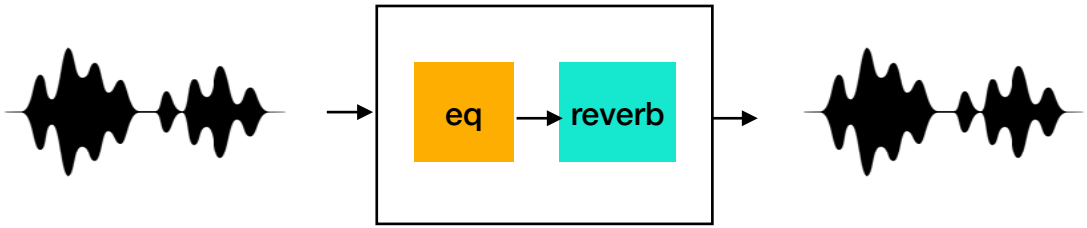
- **EQ (Equalization)** – Adjusts frequencies to balance tone

- **Reverb** – Creates a sense of space and depth.

and we can chain them!



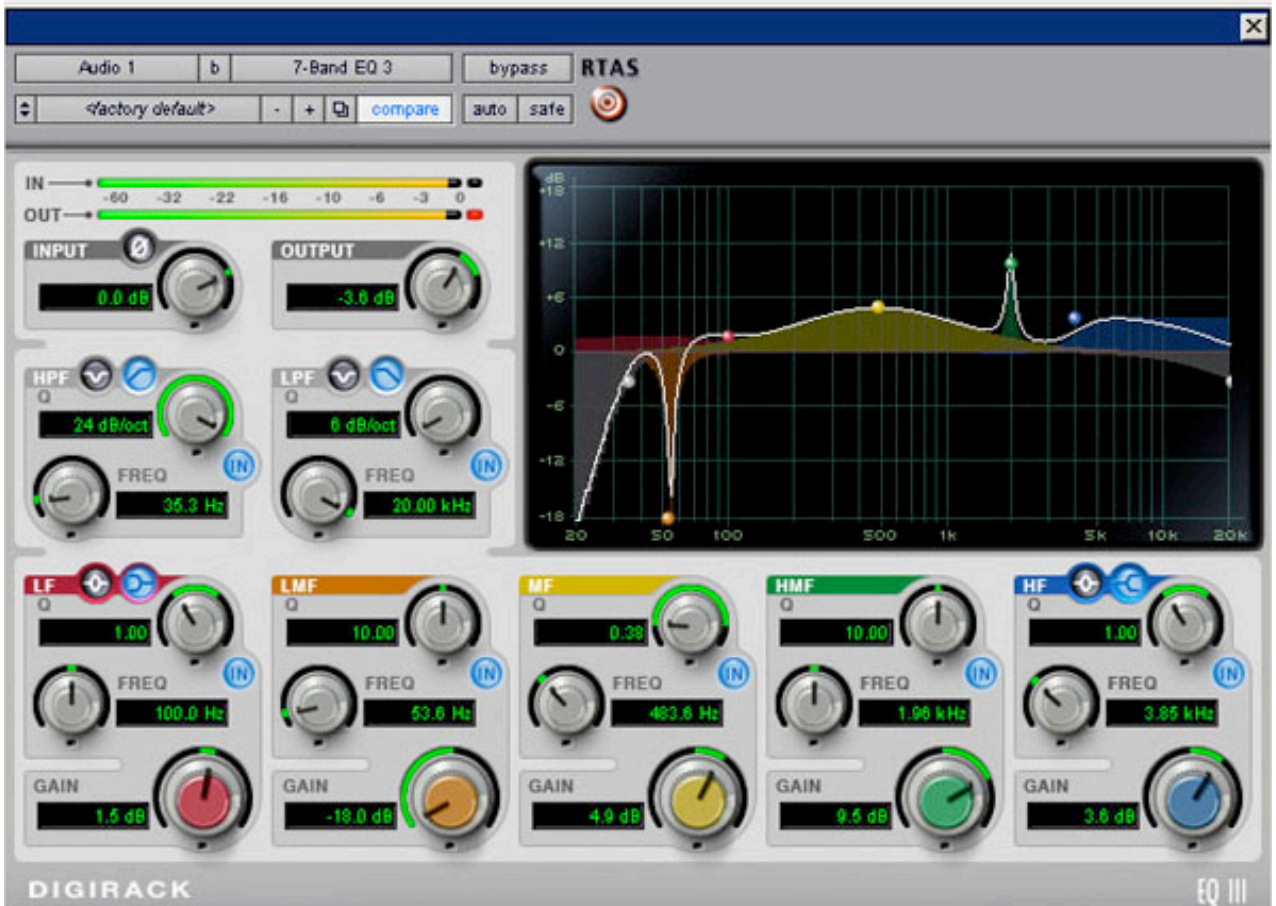




this is an FX chain of length 2

back

All FX have DSP-based controls like these

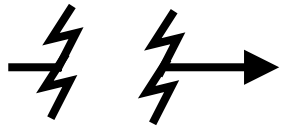


Confusing and unintuitive

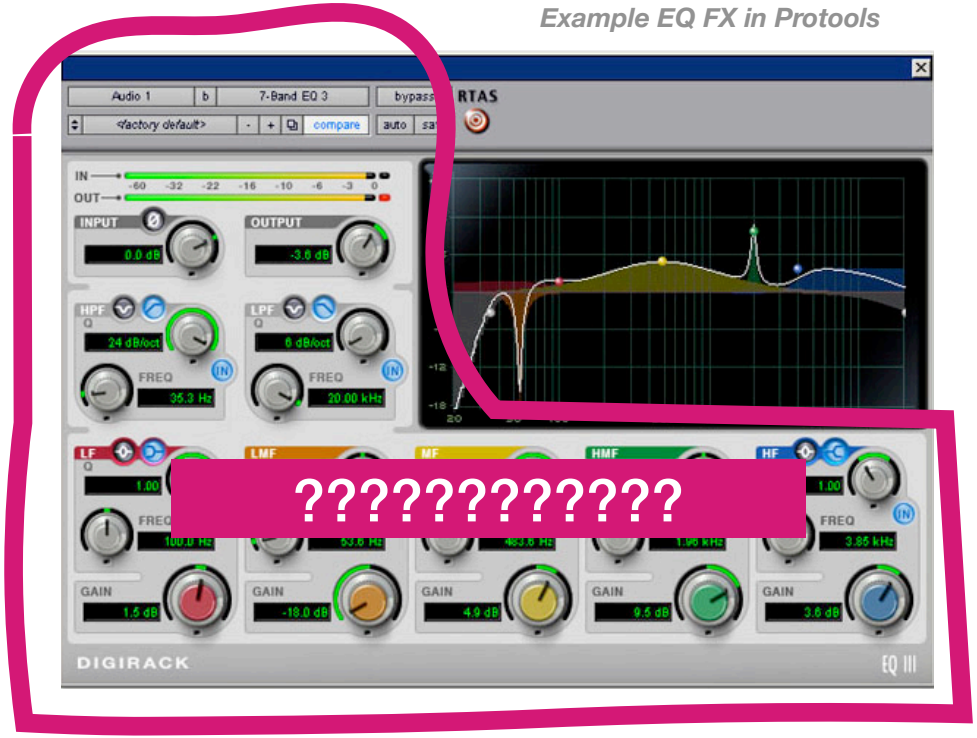
Especially to novice users

Key Problem

‘old timey radio’



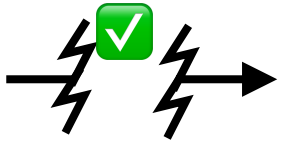
Example EQ FX in Protools

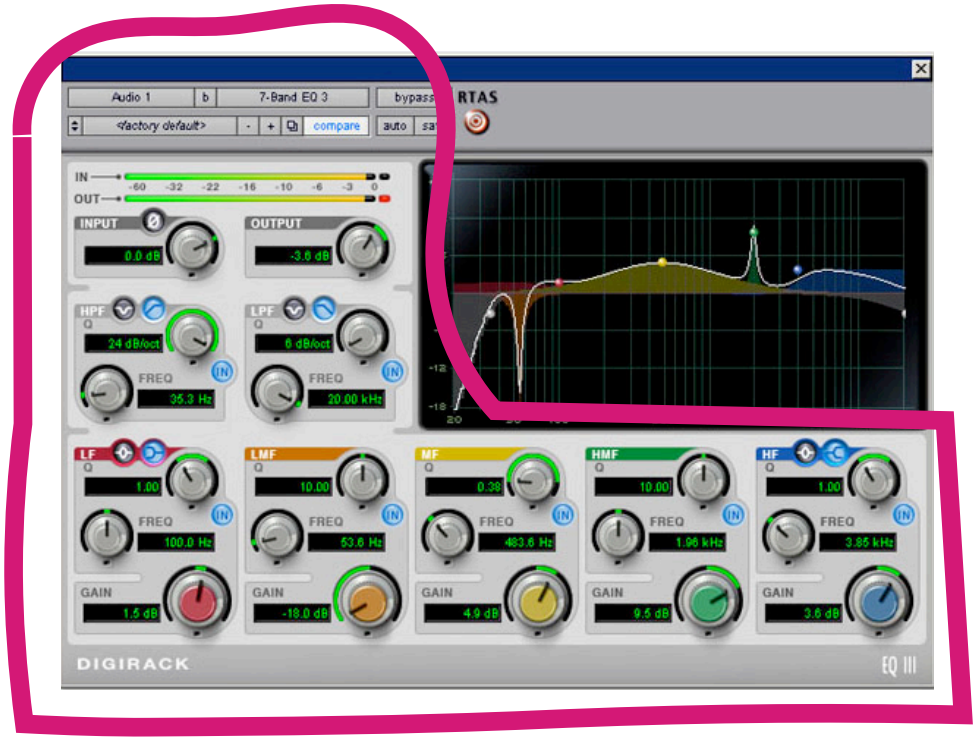


disconnect between ***intuition*** and ***implementation***

Enter Text2FX

`‘old timey radio’`





Text2FX: A Semantic Audio Production Tool

Can we use **CLAP** to connect **any high-level semantic text descriptor** (e.g., ‘bright’) to **low-level signal processing parameters** (e.g., EQ controls)?

Text2FX

A system that maps

ANY high-level concept <> ANY set of FX knobs

(e.g., ‘warm’, ‘dark and roomy’) (e.g., EQ, Reverb, Compression)

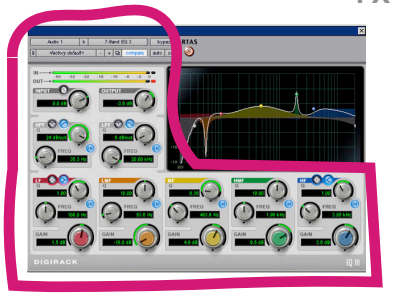
‘warm’ ✅

FX chain: EQ —> Reverb —> Compression

‘in-your-face and bold’ ✅

FX chain: EQ —> Compression

‘old timey radio’ ✅



FX chain: EQ-only

Text2FX

e.g. EQ → Delay → Reverb → Dynamic Range Compression

another example of an FX chain, length = 4

e.g. EQ → Reverb

with specific FX chain

inputs

1

'clean' input audio

2

Make this sound...

like it's on a 1920s radio

target text descriptor

Text2FX

outputs

1

'effected' audio

2

this model added...

REVERB

EQ

DRC OFF

FX parameters

which the user can then adapt and tweak

Single-Instance Optimization via CLAP Tuning

*inspired by **TagBox** (Manilow et al., 2021)*

Given an **input sound** and **target descriptor**, find me the best audio FX parameters (FXparams) via

(1) **Randomly pick FXparams**

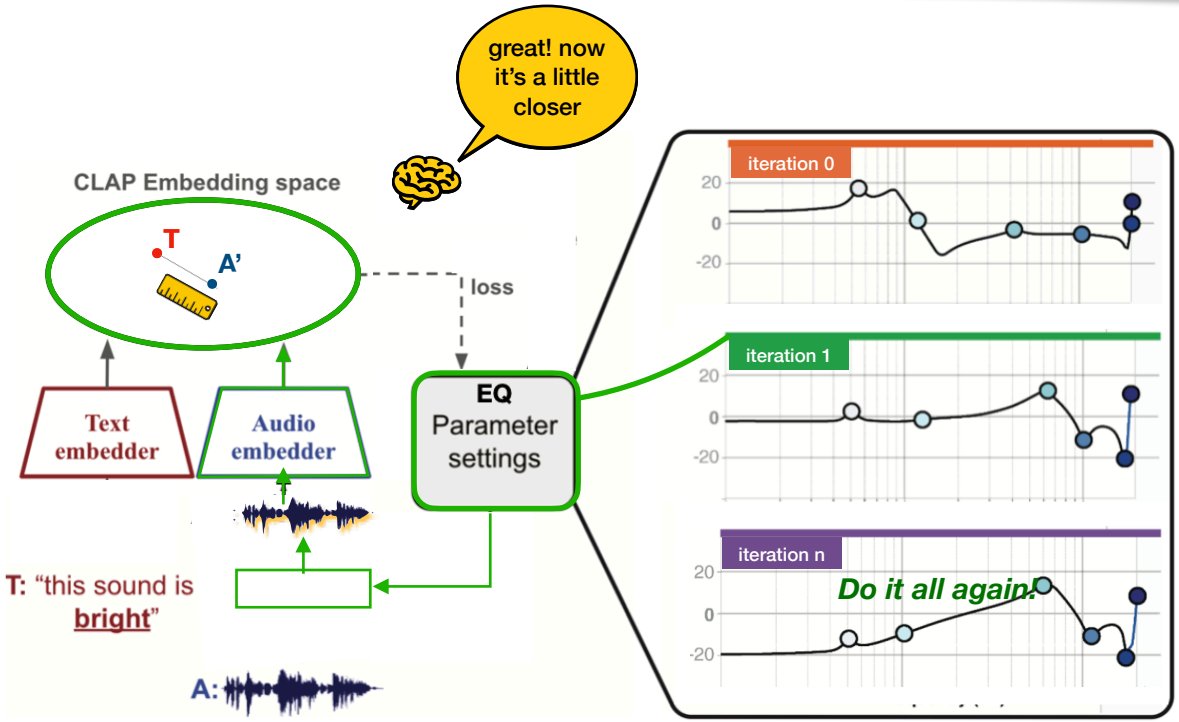
(2) **Apply to input sound to get "effected" sound (A')**

(3) **Generate CLAP embeddings for A' and T**

(4) **Measure distance between A' and T (loss)**

(5) **Backpropagate loss, adjust EQ parameters to minimize distance**

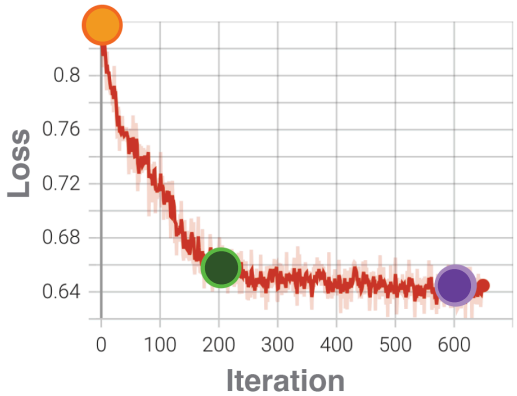
Again!

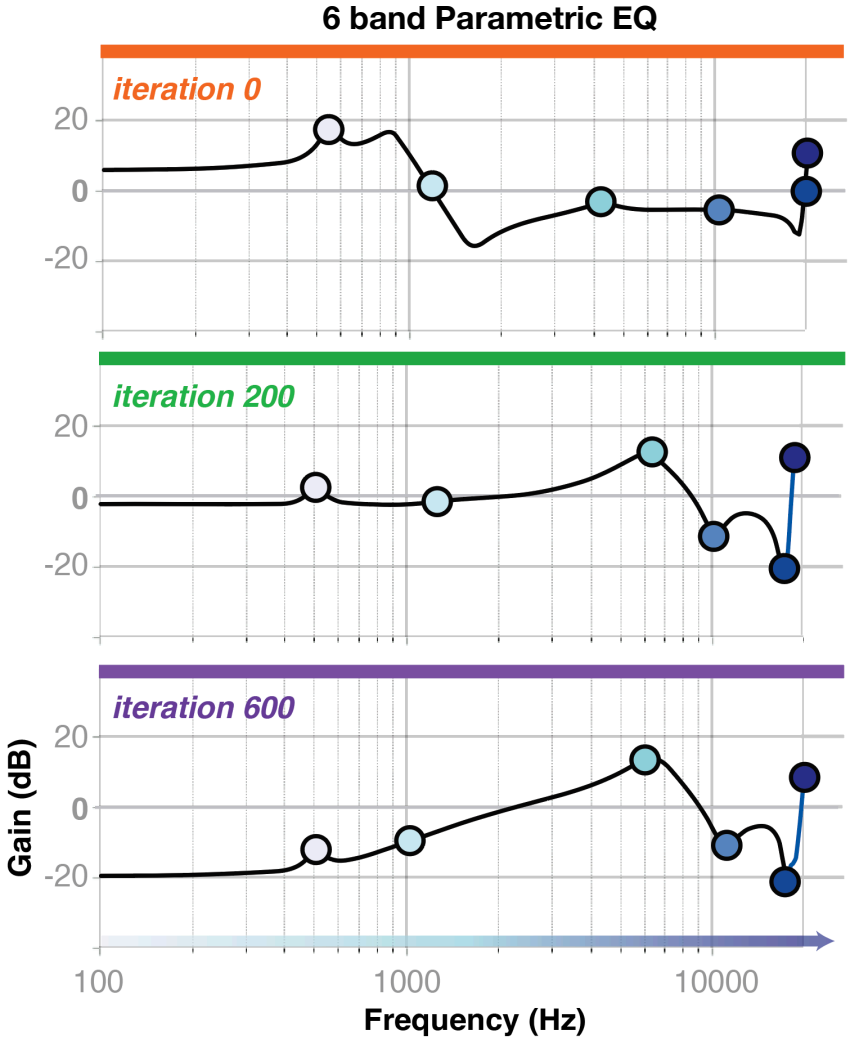


great! done optimizing ✅

Target Prompt

*"This sound is **bright**"*





Text2FX Optimization Example

Cool artifacts of this optimization algorithm

Single-Instance Optimization of FXparams

Given an **input sound** and **target descriptor**, find me the best audio FX parameters (FXparams) via

(1) Randomly pick FXparams

(2) Apply to input sound to get modified sound (A')

(3) Run A' and T into CLAP

(4) Use CLAP to quantify how close A' is to T

(5) Use that to then adapt EQ parameters

Again!

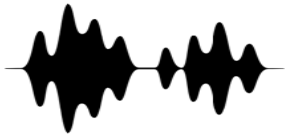
- No training of a neural network
- Bypasses requirement of needing a large dataset
- Avoids generation of unwanted audio artifacts (only modifies FX parameters, not audio itself)

Listening Examples

Example

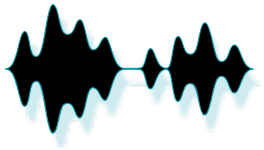
Target: underwater

FX chain: Reverb



input audio

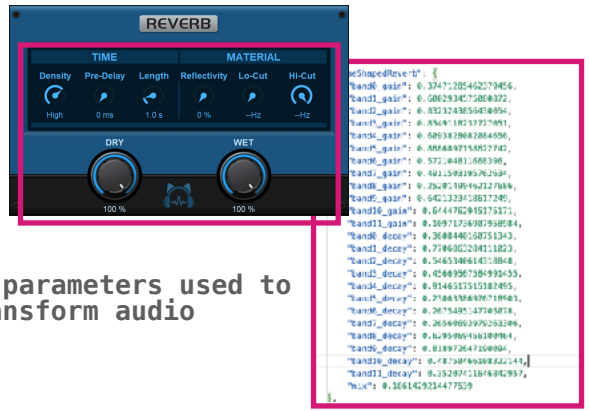
Text2FX



transformed audio

"underwater"

target text

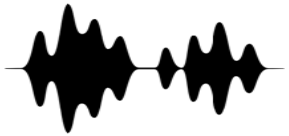


FX parameters used to transform audio

Example

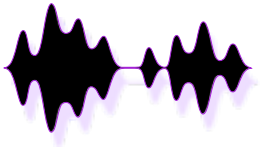
Target: dramatic

FX chain: EQ → Reverb



input audio

Text2FX



transformed audio

“dramatic”

target text

FX parameters used to transform audio

TEXT2FX

What's the best way to steer embeddings in the CLAP space?

Two different approaches of accounting for initial audio content

1. **Text2FX-cosine**
   see what CLAP knows

2. **Text2FX-directional**
   provide some context of the initial audio's texture

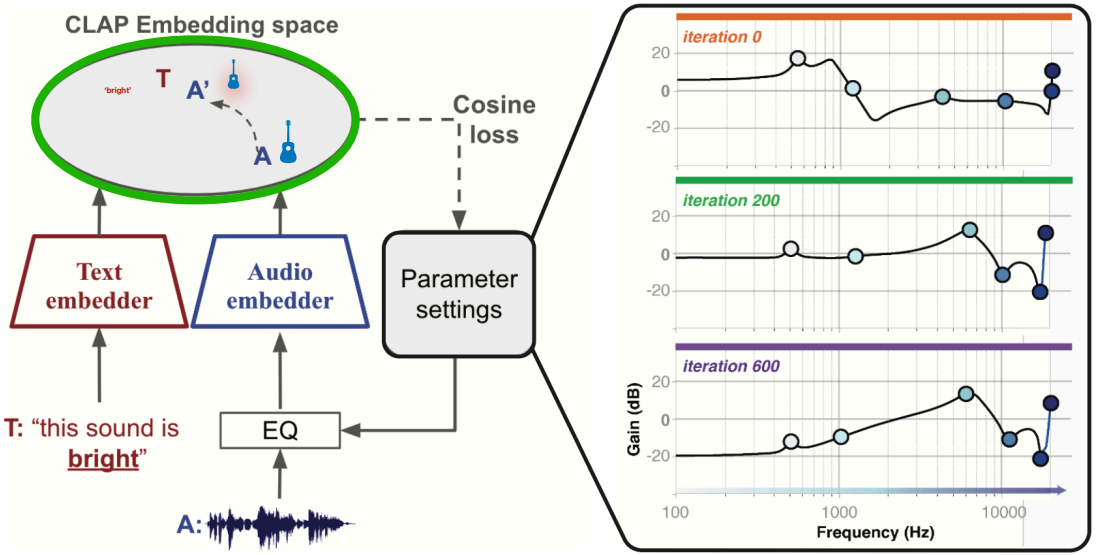
Text2FX-cosine

Most basic approach (no extra adaptation to account for content)

Cosine-loss

minimize of cosine distance between a single audio-text embedding pair

T — fixed, target text prompt
A' — modified, "effected" audio
A — fixed, input audio



Text2FX-cosine

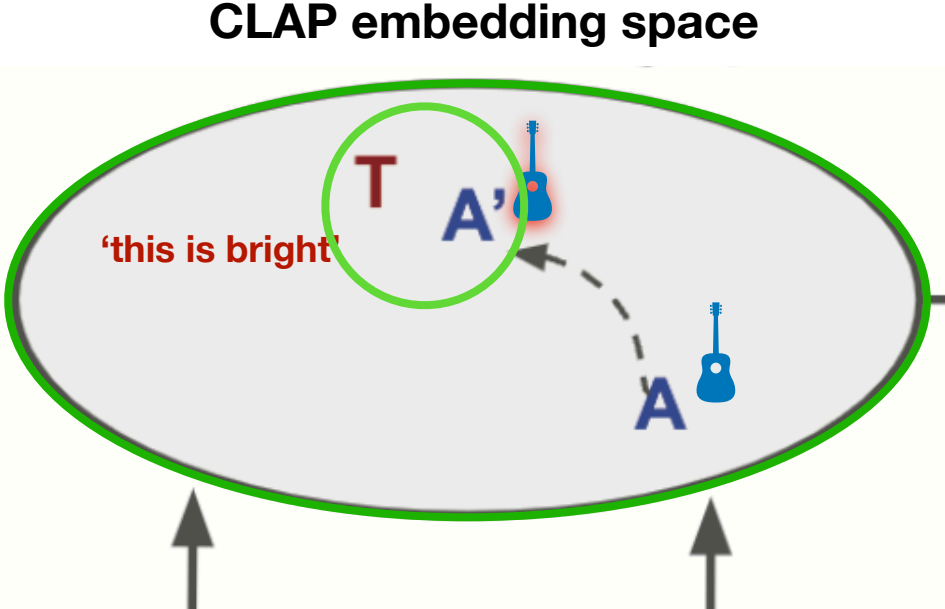
Most basic approach (no extra adaptation to account for content)

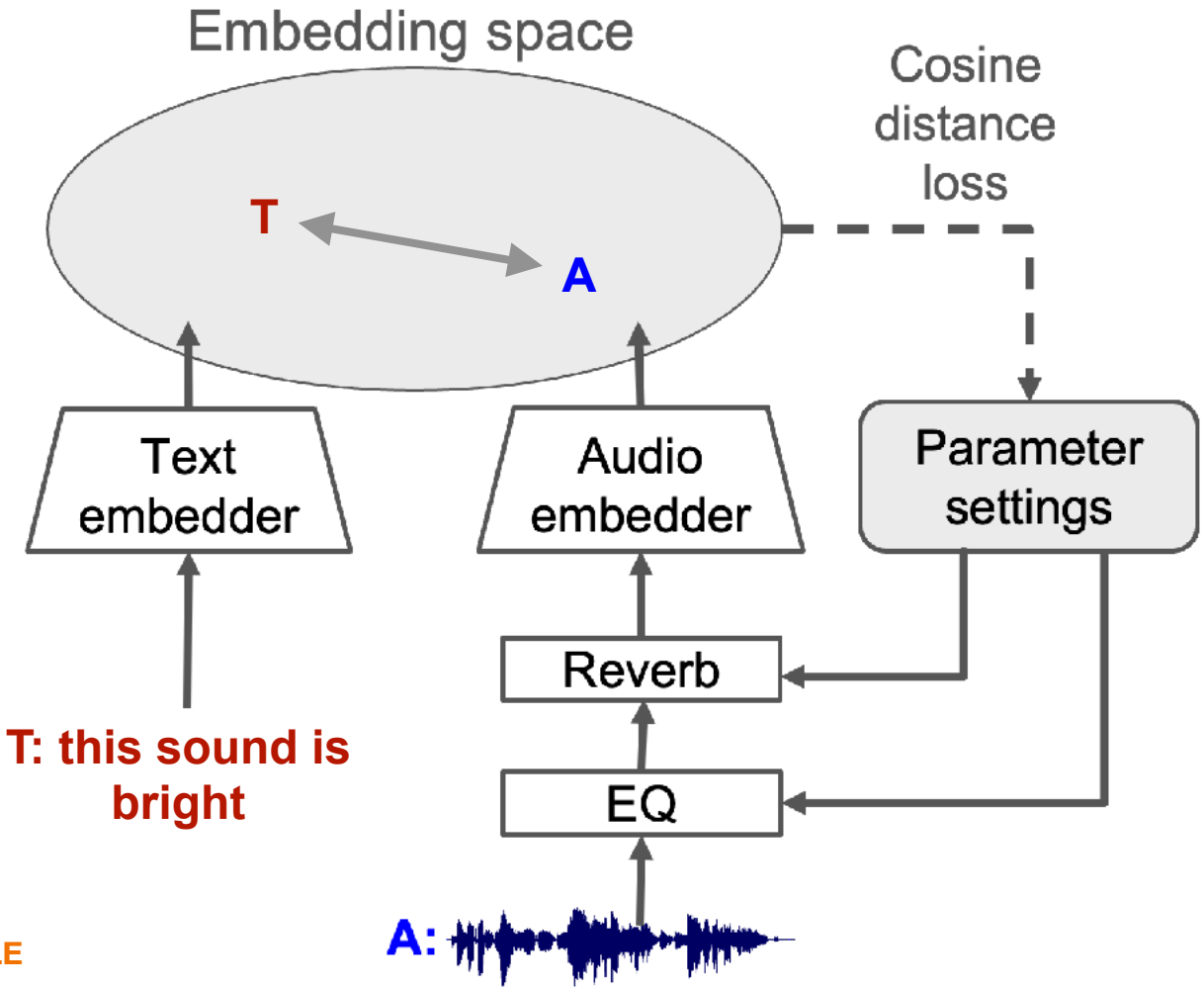
Cosine-loss

minimize of cosine distance between a single audio-text embedding pair

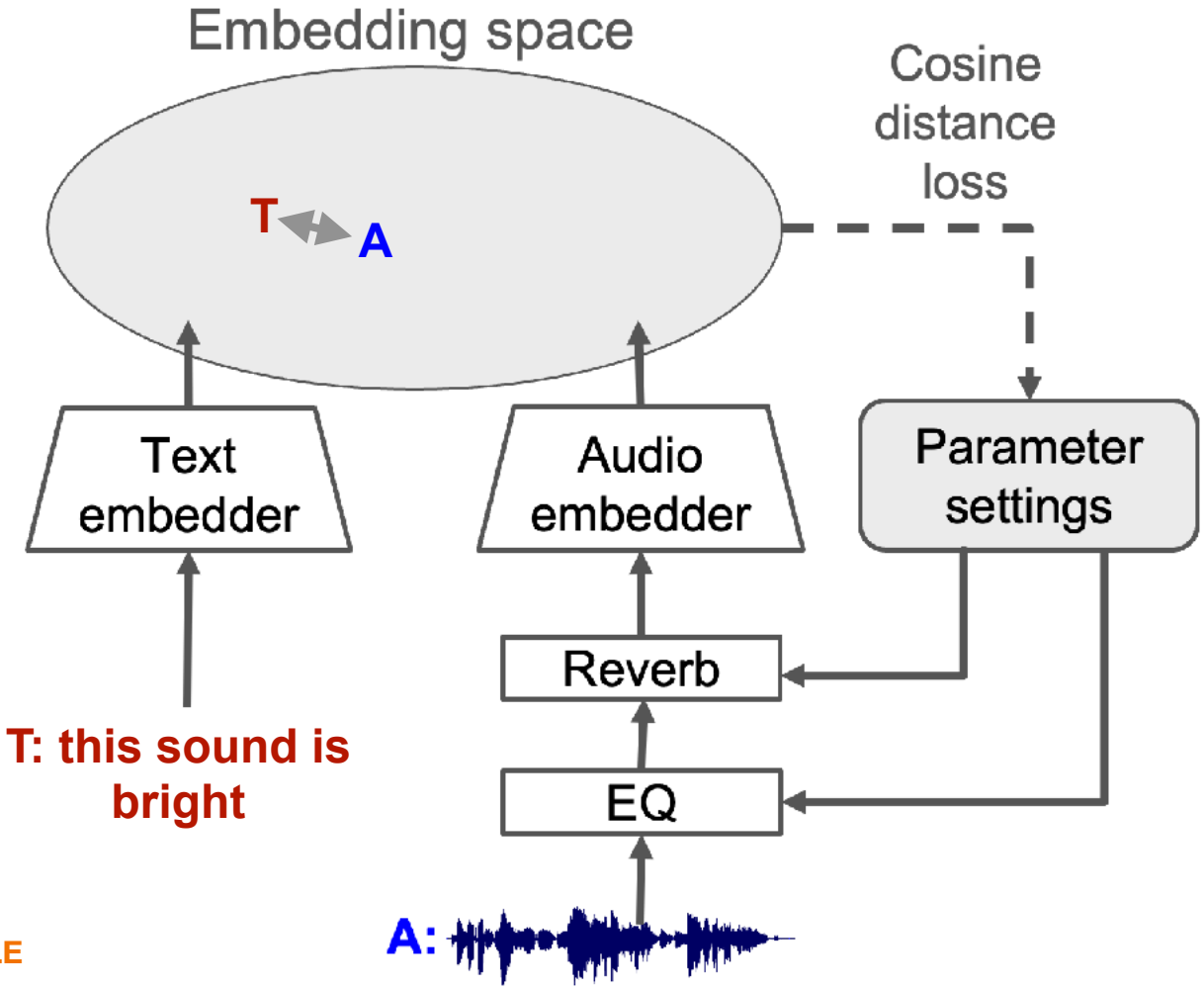
T — fixed, target text prompt
A’ — modified, “effected” audio
A — fixed, input audio

Modify the audio (via optimized FX params) such that it gets closer to the text itself





Text2FX-cos EXAMPLE



Text2FX-cos EXAMPLE

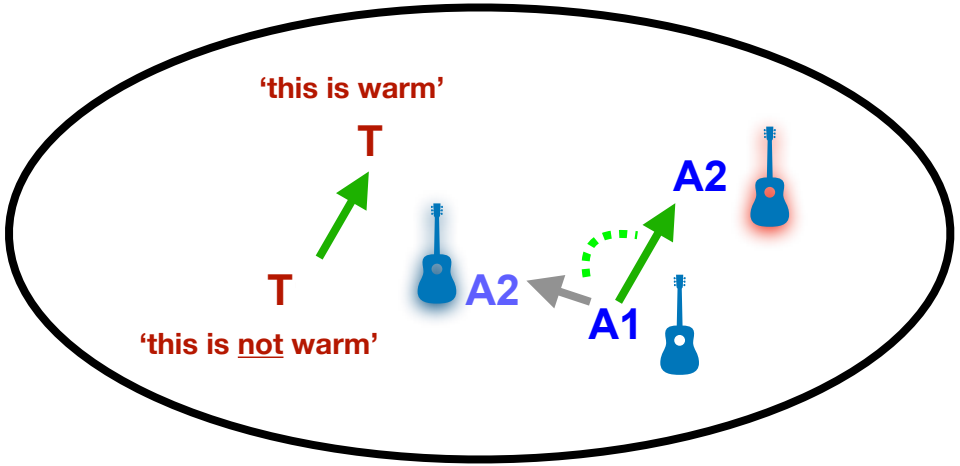
Text2FX-directional

Give context, use two embedding pairs

Directional Loss

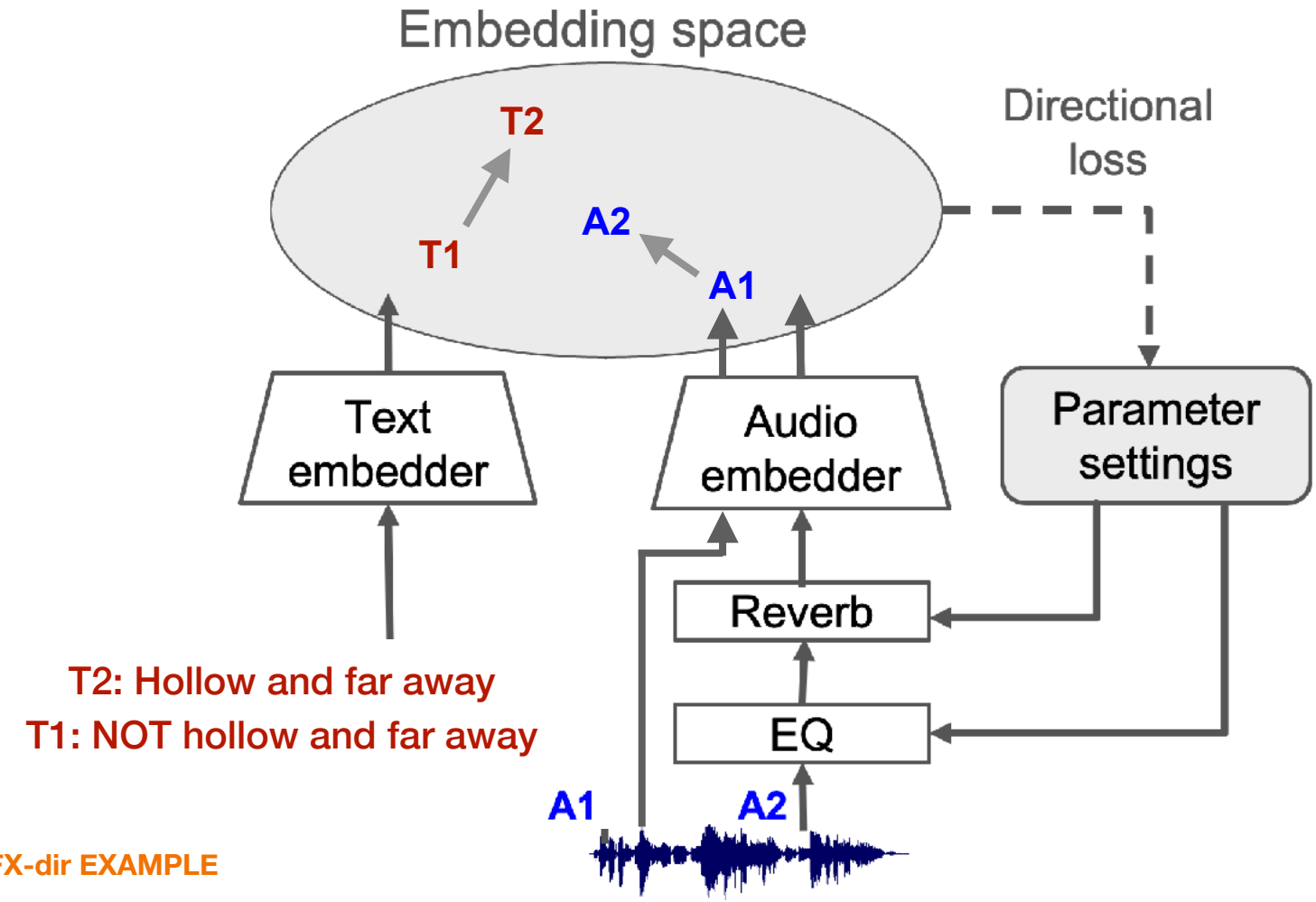
From *DiffusionCLIP (Kim et al., 2022)*

Use an **extra contrasting text prompt as an anchor** to guide the optimization of modified audio

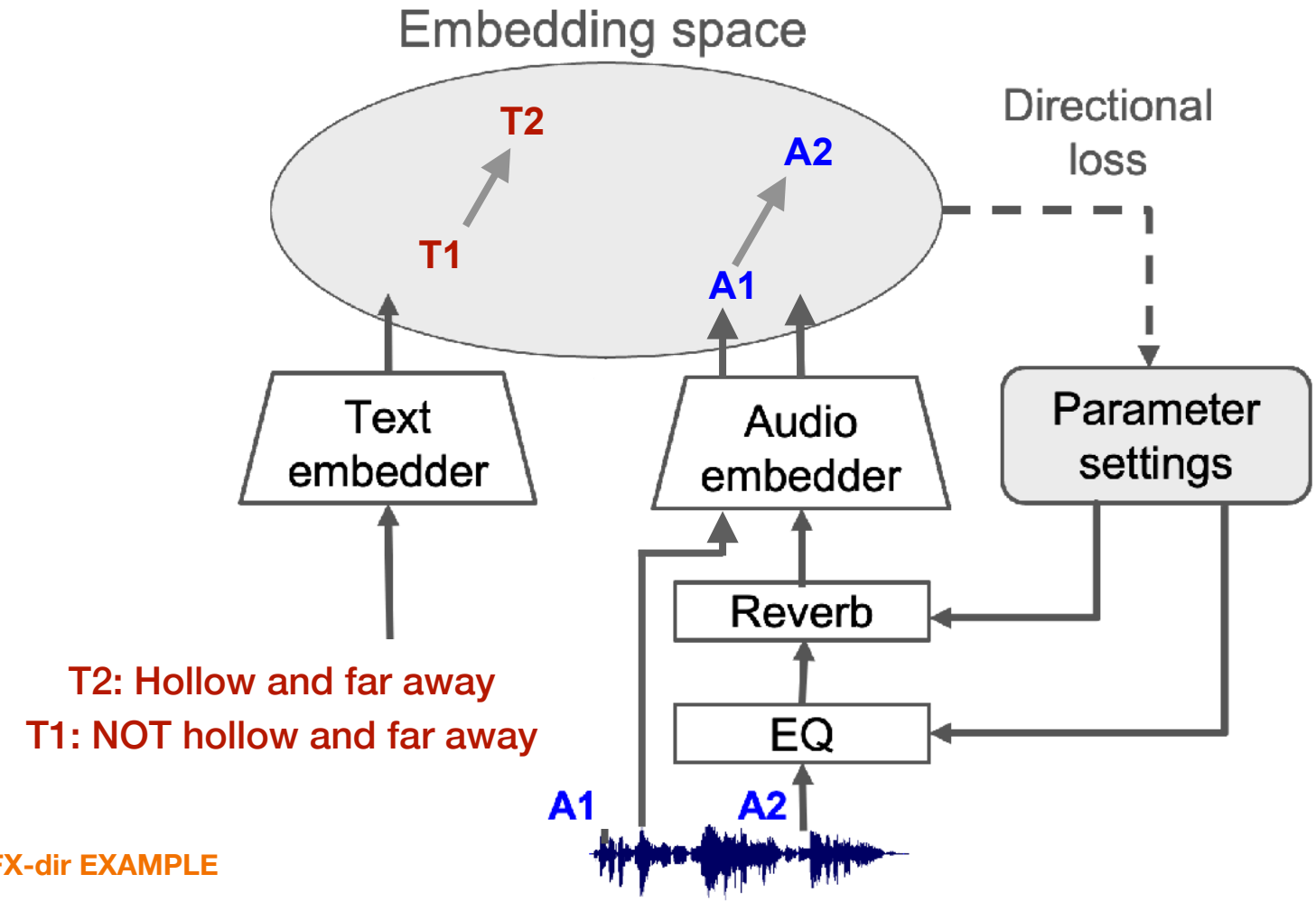


CLAP embedding space

T1 — fixed, extra anchor text prompt
T2 — fixed, target text prompt
A1 — fixed, input audio
A2 — modified, "effected" audio



Text2FX-dir EXAMPLE



Text2FX-dir EXAMPLE

Listening Study

They both seem to work…
Which one works better?

60 text prompts allocated across 3 FX chains

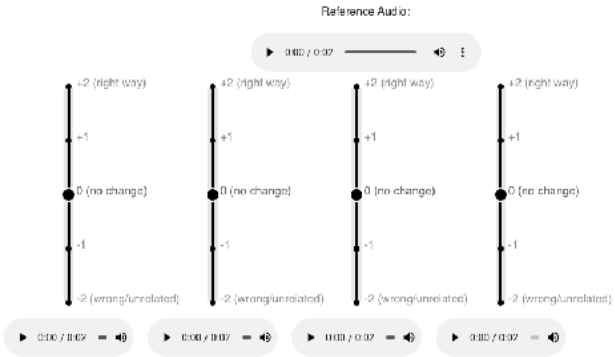
	Single Words		Multiwords	
	Concrete	*Abstract*	*Combination*	*Imagery*
EQ	tinny, muffled, light, deep, crisp, bright, mellow	ethereal, eerie, grand	soft yet vibrant, in-your-face and bold, shrill and sharp, quiet and gentle, cool and smooth	coming through an old telephone, coming from a speaker under a blanket, booming like a thunderstorm, delivered with a softer feel, like a hazy surreal dream
Reverb	boomy, spacious, dry, cavernous, echoey, underwater, dry, reverberant	empty, long, bold	booming and vast, clear but distant, cozy and enveloping, heavy and dramatic, hollow and far-away	coming from a cathedral, coming from a long hallway, coming from a small and intimate sound booth, like an explosion in a canyon, accompanied by a faint atmospheric haze in the background
EQ → Reverb	metallic, harsh cold, blaring, bassy, grainy, breezy	dramatic fluffy, powerful	barren and detached, warm and full-bodied, vibrant and powerful, resonant and harmonious, high and tinny	coming from a small cavern with a muffled echo, coming from underwater in a swimming pool, coming from a broken speaker in an empty warehouse, like a shrill Victorian ghost, like a distant radio broadcast with a warm lingering presence

What audio samples? 30 Reference audio files (15 speech, 15 music)

4-way evaluation of each prompt/audio combo

- **Text2FX-cosine:** FXparams optimized via cosine loss
- **Text2FX-directional:** FXparams optimized via directional loss
- **Random:** Randomly assigned FXparams
- **noFX:** The original reference audio without any FX

mimics what a novice audio producer might do

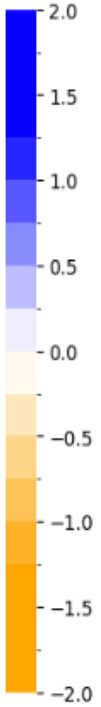


The rating scale

+2: The audio changed in the right direction
(i.e., definitely more warm than reference)

0: No noticeable change compared to the reference (neutral)

-2: The audio changed in the wrong or unrelated direction
(i.e., changed, but definitely not more warm than reference)



Specifically we'll compare for Text2FX (variants & aggregates) vs Random

- **Text2FX-cosine:** FXparams optimized via cosine loss
- **Text2FX-directional:** FXparams optimized via directional loss

- **Text2FX-Best:** When the better performing variant succeeds
- **Text2FX-Both:** When both variants succeed

PERCENTAGE (%) OF EVALUATIONS RESULTING IN POSITIVE SCORES

Model	**EQ**	**Reverb**	**EQ → Reverb**
Text2FX-cosine			
Text2FX-directional			
Random			
Text2FX-Best			
Text2FX-Both			

Percentage (%) of Evaluations Resulting in Positive Scores

Model	EQ	Reverb	EQ → Reverb
Text2FX-cosine	48.26	51.61	47.24
Text2FX-directional	45.49	53.23	50.61
Random	22.22	49.03	30.37

↑ **HIGHER IS BETTER**

Text2FX beats **Random**

PERCENTAGE (%) OF EVALUATIONS RESULTING IN POSITIVE SCORES

Model	EQ	Reverb	EQ → Reverb
Text2FX-cosine	48.26	51.61	47.24
Text2FX-directional	45.49	53.23	50.61
Random	22.22	49.03	30.37
Text2FX-Best	67.01	74.19	68.10
Text2FX-Both	26.74	30.65	29.75

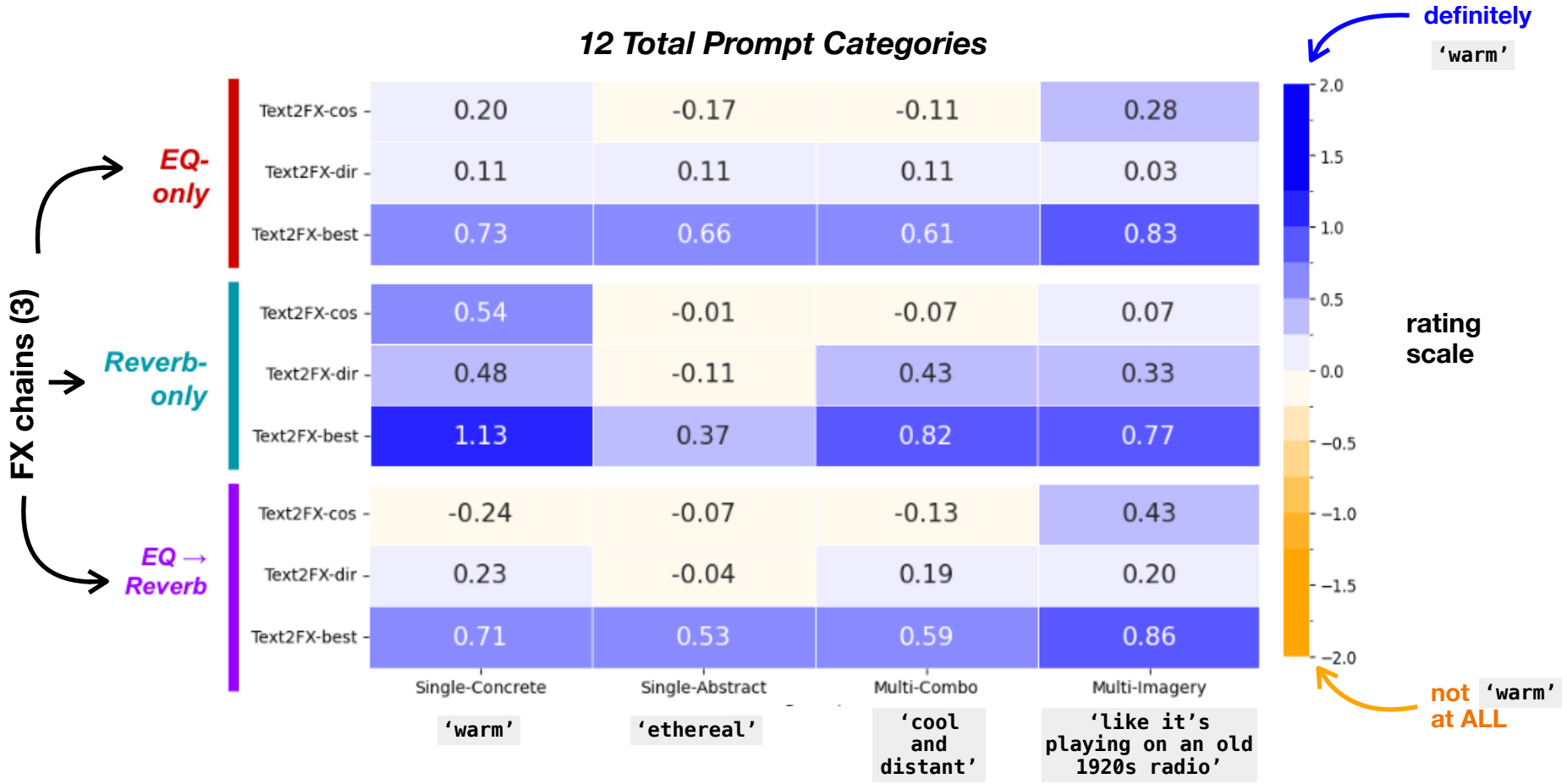
Text2FX beats **Random**

PERCENTAGE (%) OF EVALUATIONS RESULTING IN POSITIVE SCORES

Model	EQ	Reverb	EQ → Reverb
Text2FX-cosine	48.26	51.61	47.24
Text2FX-directional	45.49	53.23	50.61
Random	22.22	49.03	30.37
Text2FX-Best	67.01	74.19	68.10
Text2FX-Both	26.74	30.65	29.75

Text2FX-Best drastically beats **Random**

Breakdown of Listener Rating scores



Taking the system at its best — Text2FX-Best, are there particular strengths?

Prompt Type

Single-Concrete

e.g., ‘bright’

‘spacious’

‘harsh’

Multi-Imagery

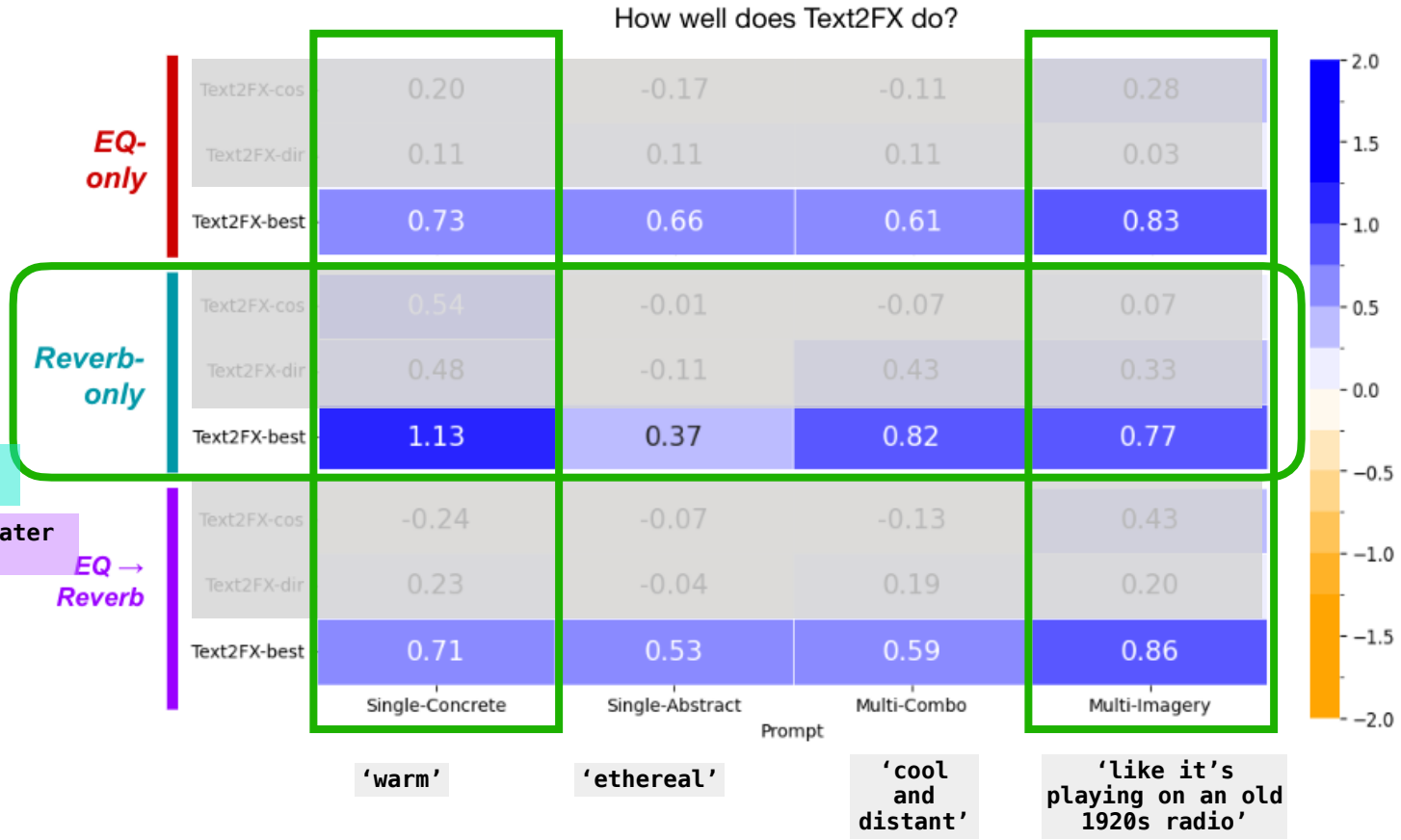
e.g. ‘coming through an old telephone’

‘like an explosion in a canyon’

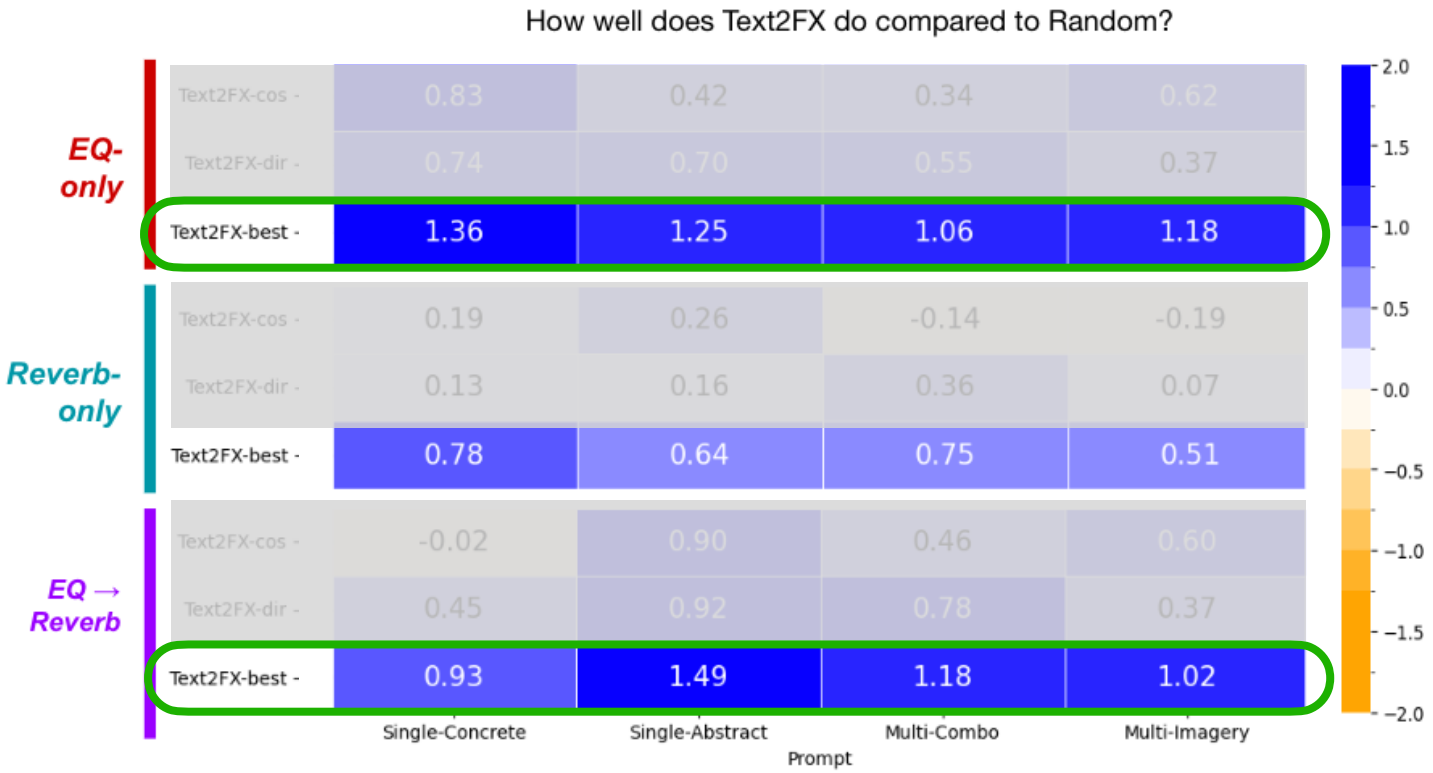
‘coming from underwater in a swimming pool’

FX chain

Reverb-only



Pronounced Advantage for *EQ-only* and *EQ → Reverb* FX Chains



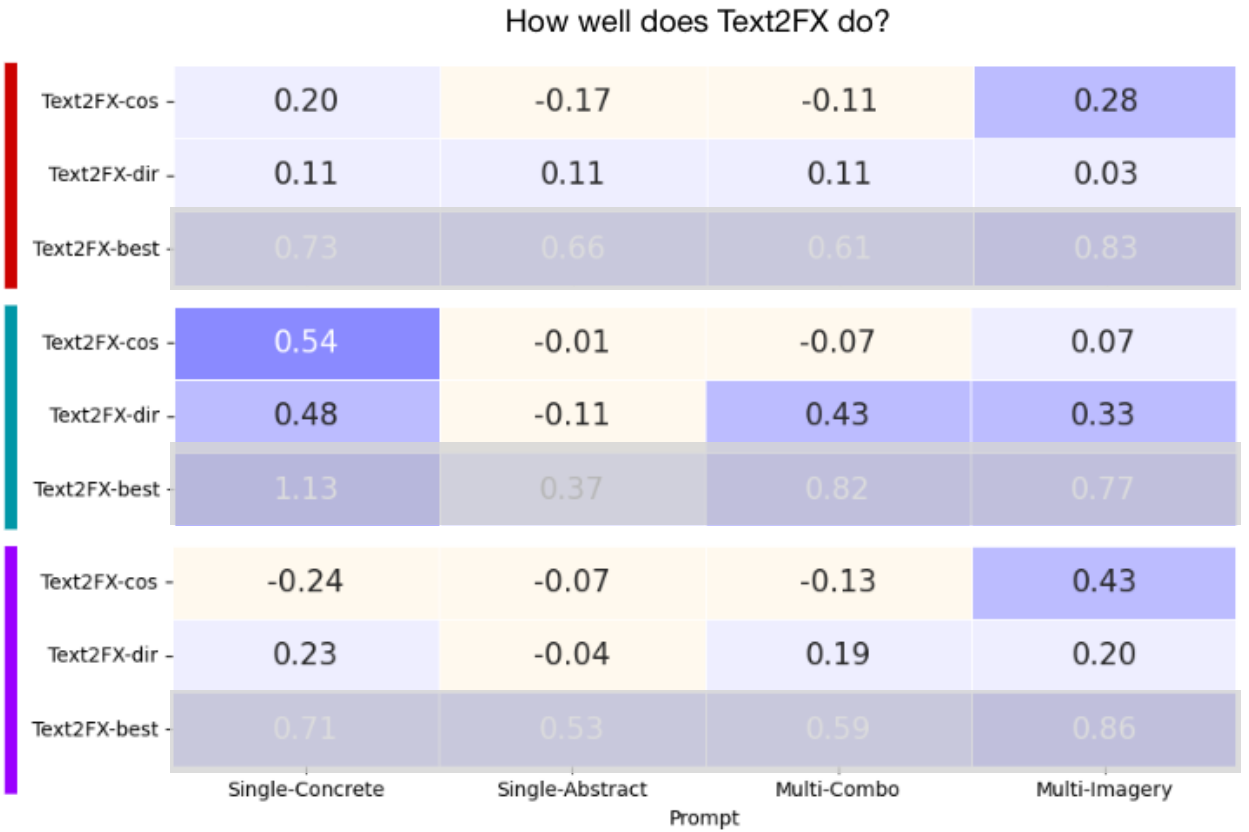
Comparing Text2FX-cos vs. Text2FX-dir

☆ **Text2FX-dir** provides more **reliable performance**

☆ **Text2FX-cos** produces more **polarizing transformations**

☆ **Text2FX-dir** may **generalize better**

- *prompt type*
- *longer FX chain*



Back to Embeddings

Something else we might wanna do is *visualize* the embedding space itself

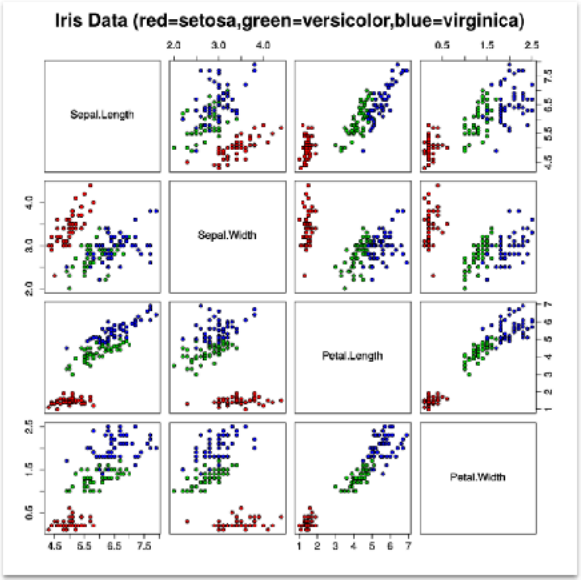
We can't visualize 512 dimension vectors, but reduce to 3D through some techniques ("the curse of dimensionality")

**for any embeddings, not just cross-modal*

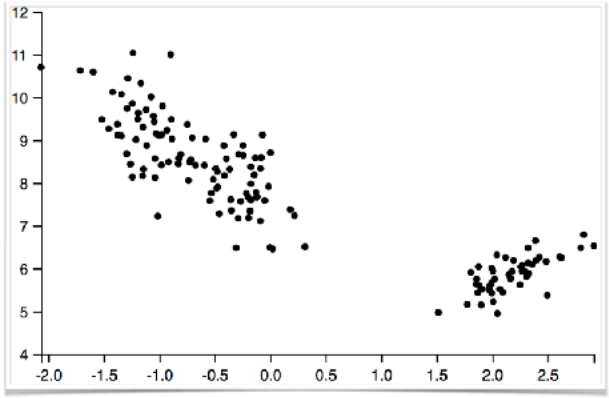
Dimensionality reduction

Goal: Map high dimensional data onto lower-dimensional data in a manner that preserves *distances/similarities*

Original Data (4 dims)



Projection with PCA (2 dims)



Objective: projection should "preserve" relative distances

slide from Bryon Wallace Northeastern

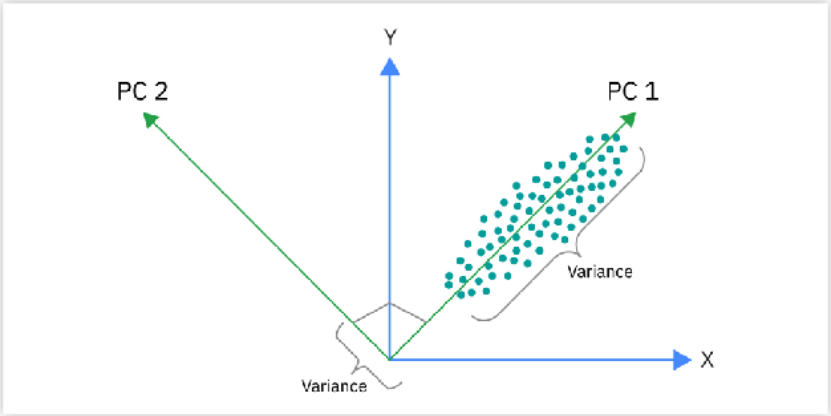
PCA: Principal Component Analysis

TLDR; Linear technique that works to maximize global variance

PCA finds **new axes (called principal components)** along which the **data varies the most**. These axes are linear combinations of the original features.

What it does: Finds the directions with the most variation in data.

How? It uses eigenvectors and eigenvalues of the covariance matrix to find those directions



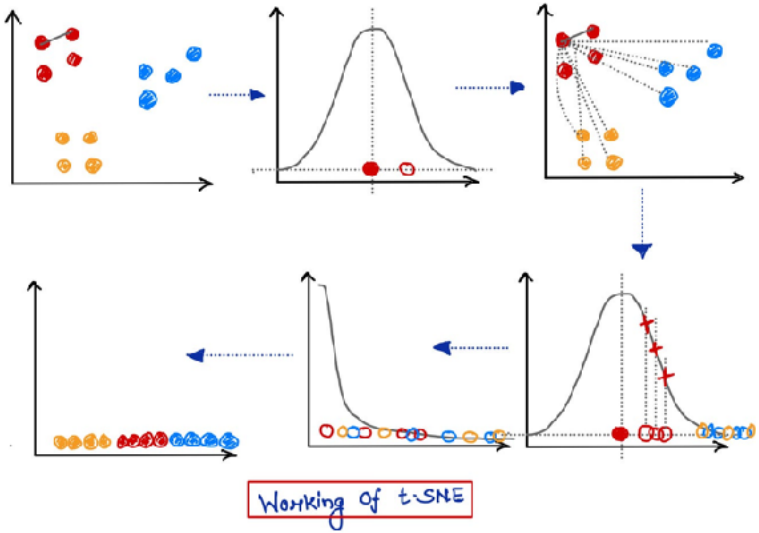
source: ibm

t-Distributed Stochastic Neighbor Embedding (t-SNE)

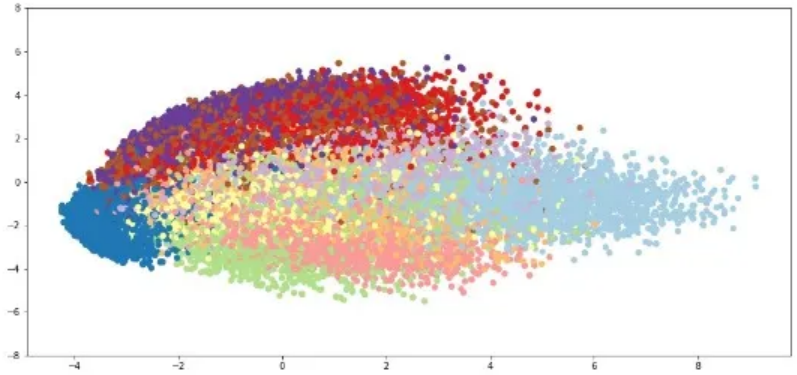
TLDR; Non-linear technique that preserves local structure by modeling pairwise similarities.

t-SNE arranges data in a way that keeps **similar items close together in the low-dimensional space**, making clusters easy to see.

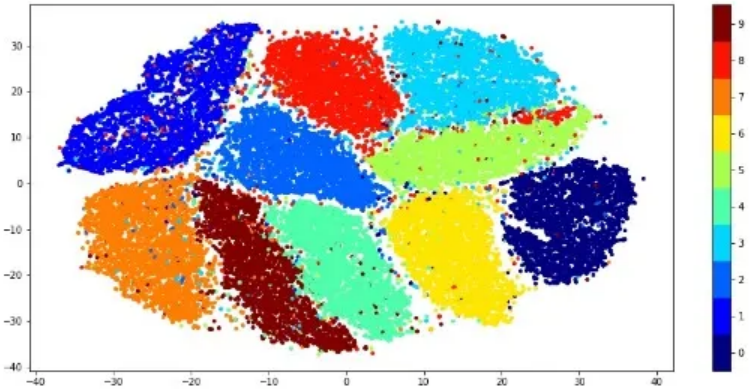
*Computes **pairwise similarities using Gaussian distributions** in high dimensions and **Student t-distributions** in low dimensions, then minimizes distance between distributions via KL divergence*



MNIST - PCA



MNIST - TSNE

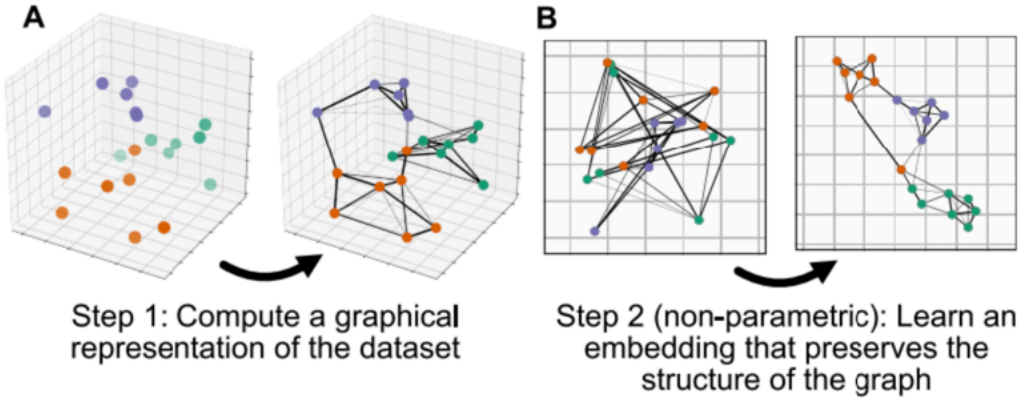


Uniform Manifold Approximation and Projection (UMAP)

TLDR; Non-linear technique that preserves both local + some global structure, scalable for large datasets.

UMAP constructs a **weighted graph** of the data's local structure and then optimizes a low-dimensional layout that preserves those relationships and the overall shape

Build a nearest-neighbor graph of the high-D data model local relationships, then learning a low-D embedding by minimizing a CE loss to aligns the graph structure with a similar graph in the lower-D space



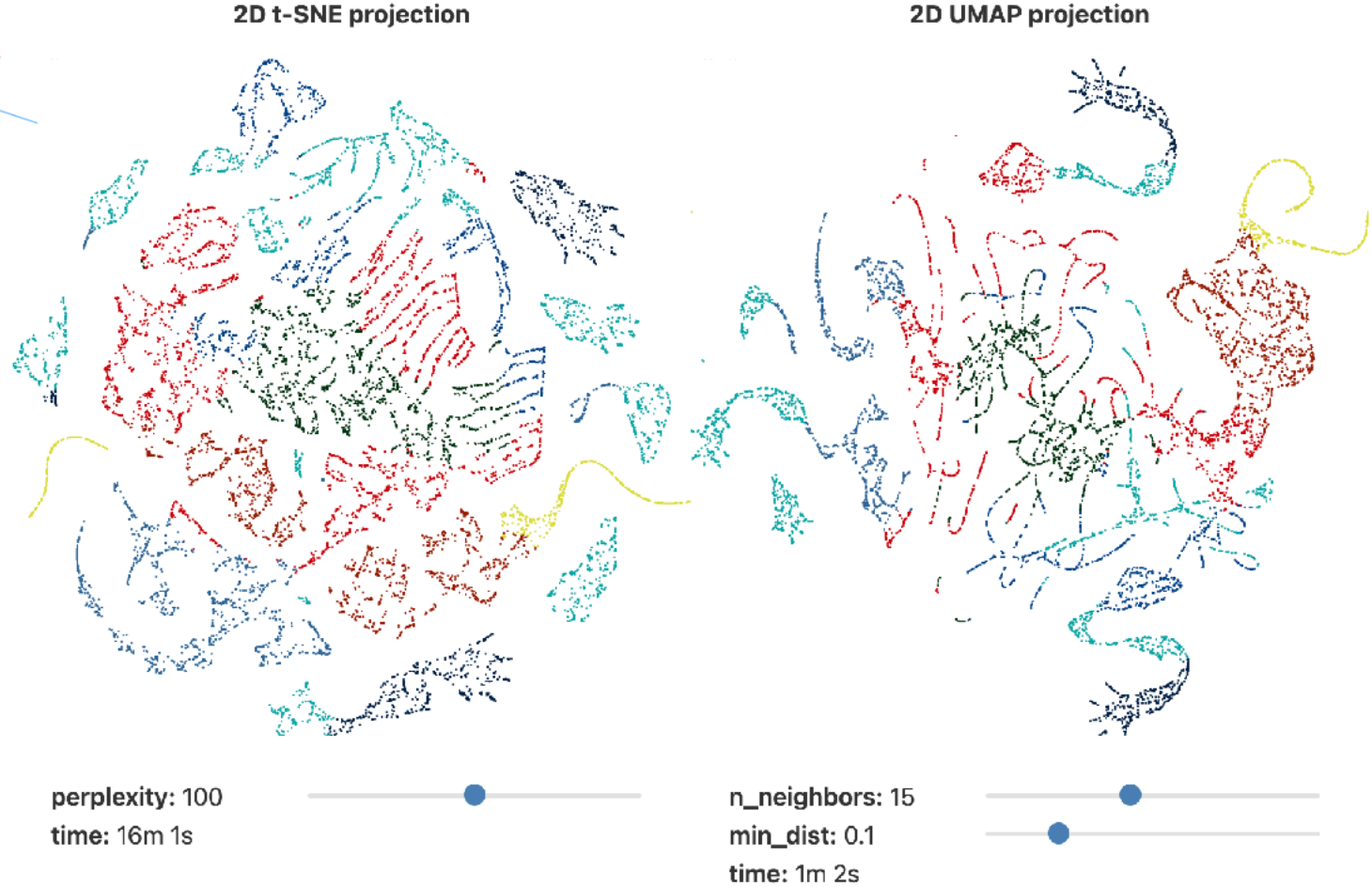


Figure 6: *A comparison between UMAP and t-SNE projections of a 3D woolly mammoth skeleton (50,000 points) into 2 dimensions, with various settings for parameters. Notice how much more global structure is preserved with UMAP, particularly with larger values of* `n_neighbors`.

Summary of PCA vs t-SNE vs UMAP

Feature	PCA	t-SNE	UMAP
Type	Linear	Non-linear	Non-linear
Goal	Maximize global variance	Preserve local structure	Preserve local and global structure
Preserves	Global variance patterns	Local clusters and pairwise similarity	Local neighborhoods and global layout
Interpretability	High	Low	Medium
Speed	Fast	Slow	Fast
Deterministic	Yes	No	Mostly (some stochastic elements)
Best For	Feature compression, initial dimensionality reduction	Visualizing cluster structure in compact space for small-medium datasets	Exploring both local clusters and broader relationships
Limitations	Cannot capture non-linear patterns	Distorts global structure, sensitive to parameters, slow for large datasets	Requires tuning, and still involves some randomness
Example use case	Reduce dimensionality of audio-text embedding space for downstream model input (feature engineering)	Visualize clusters of similar sound-text pairs (e.g., emotion categories, spoken keywords)	Understand large-scale relationships in joint audio-visual-text embeddings

When each might be good to use

Goal	Best Technique
Quick overview, compression, noise filtering	**PCA**
Visualizing clusters (e.g. categories)	**t-SNE**
Maintaining shape + cluster structure	**UMAP**
Handling very large or complex datasets	**UMAP**
Feature engineering for ML models	**PCA** (or UMAP but requires more digging)

Key Parameters

Key Parameter	PCA	t-SNE	UMAP
n_components (dimensionality)	Number of dimensions to retain aka what data to keep (important)	Output dimensionality mainly for visualization (doesn't impact the relationship structure in original space)	Output dimensionality mainly for visualization (doesn't impact the relationship structure in original space)
Neighborhood Size	Not applicable	perplexity – how many neighbors each point considers	n_neighbors – balances local vs. global structure
Cluster Spread	Not applicable	Not directly tunable	min_dist – controls spacing between points in embedding
random_state	Optional, for reproducibility	Affects layout stability	Controls reproducibility