Unmixer

An interface for extracting and remixing loops

Jordan B. L. Smith
Yuta Kawasaki
Masataka Goto

AIST

Suppose you are a remix artist...

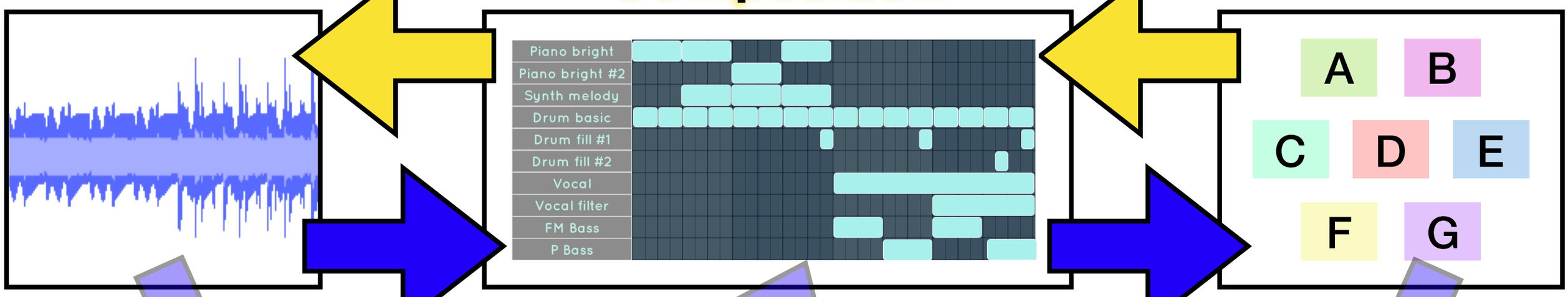
Remix artists have the STEREO MIX...

...but they want the ISOLATED LOOPS.

Song

← Composition ←

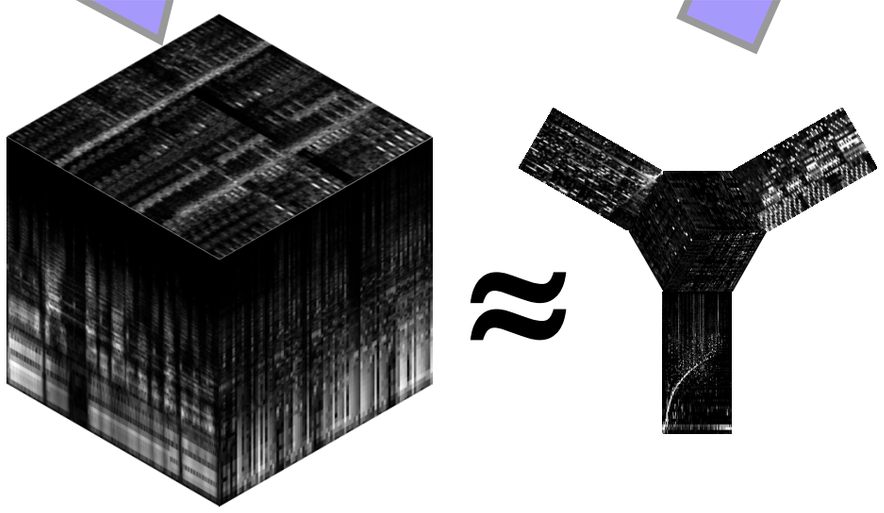
Ingredients



→ De-composition →

Goal of Unmixer: extract loops from mixed audio

Algorithm



Interface

Unmixer Interface

1. User uploads audio file

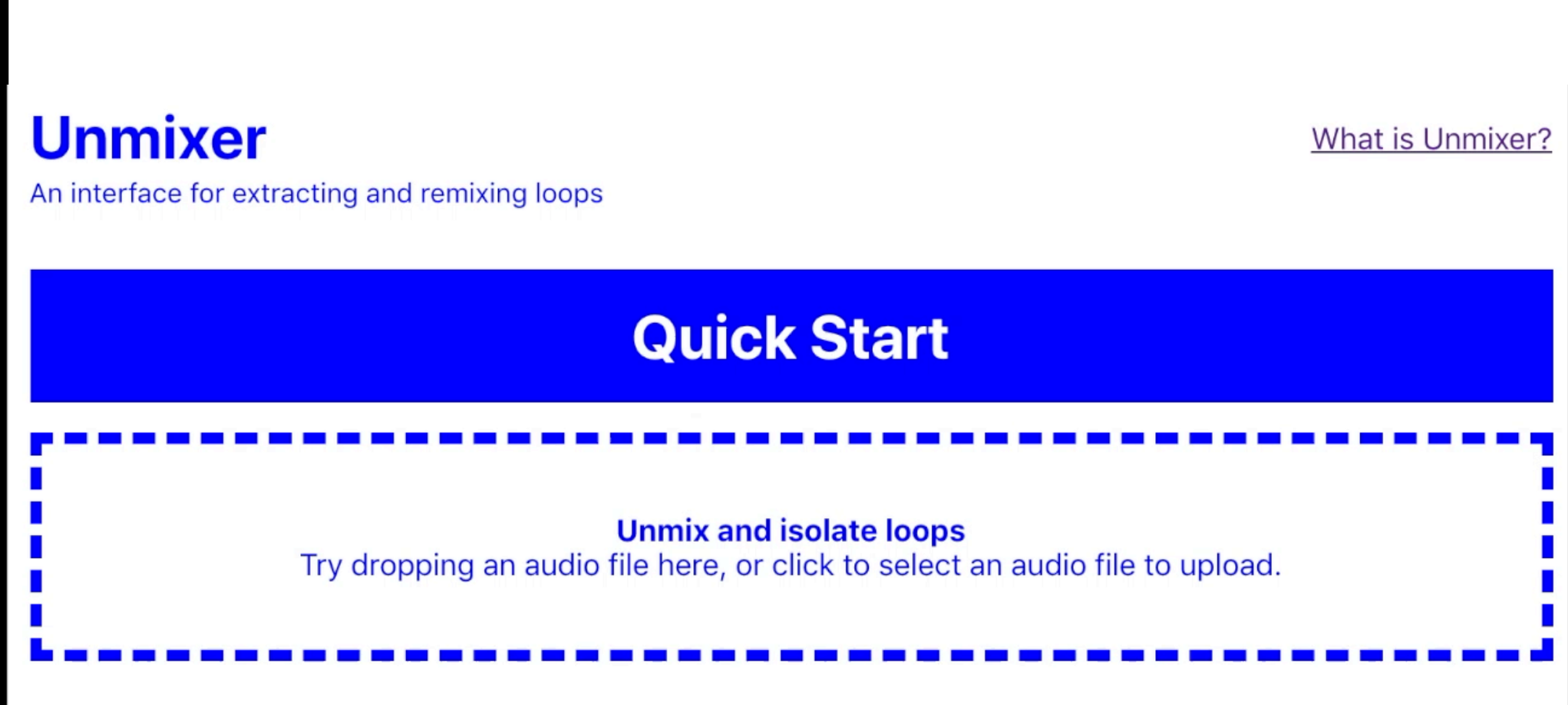
2. Wait for results (several minutes)

3. Click tiles to add or remove them from the mix

4. Add more songs, test out mashups

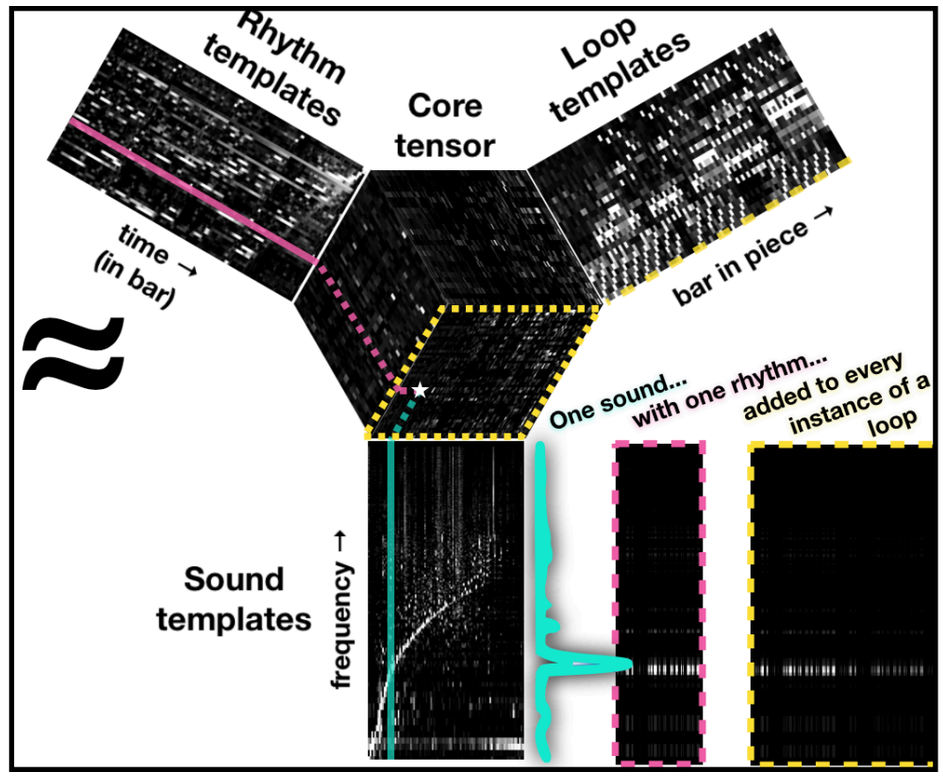
5. Adjust tempo

6. Download loops to remix in your favourite DAW



How does it work?

Very fancy non-negative Tucker decomposition!



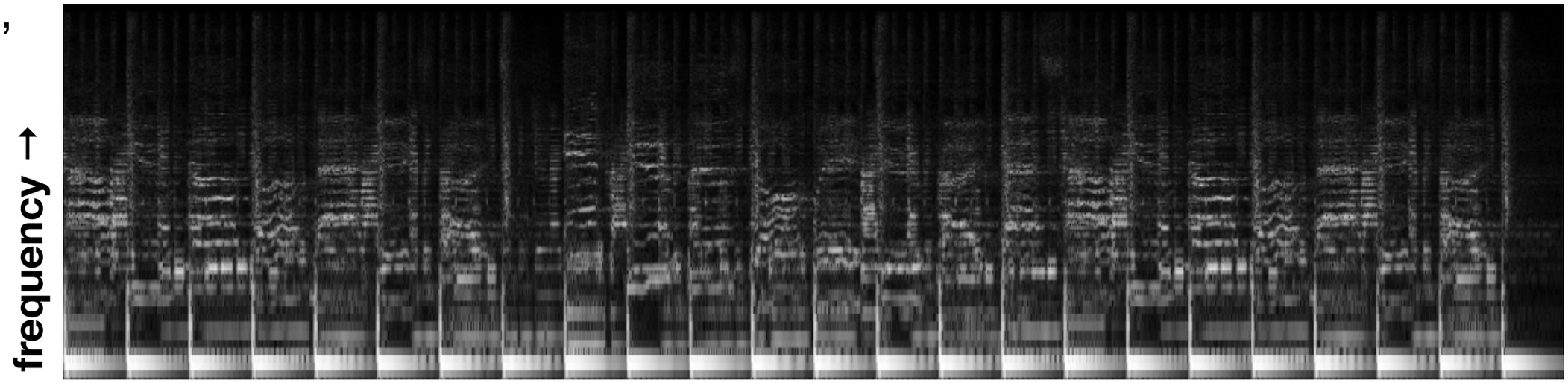
But first... normal NMF

Regular NMF

- ***Non-negative matrix factorization*** (NMF) is a technique for compressing matrices.
- It can be used for source separation:
  - First, model spectrogram (a matrix) as a set of *frequency templates,* each paired with an *activation pattern.*
  - Second: figure out how to group the templates into sources. **[This part is hard!]**
  - Lastly, reconstruct signal one template (or group of templates) at a time.

Regular NMF

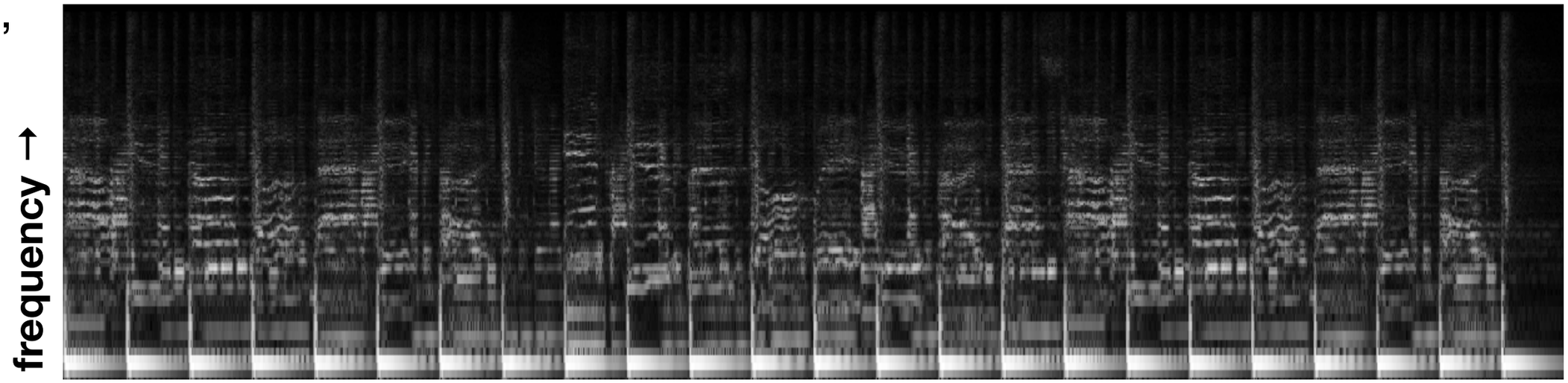
Song: “Doin’ it Right” by Daft Punk



≈

Regular NMF

Song: “Doin’ it Right” by Daft Punk

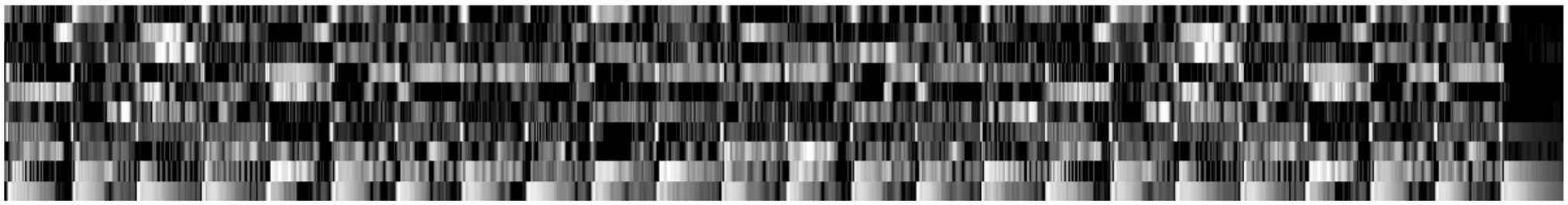


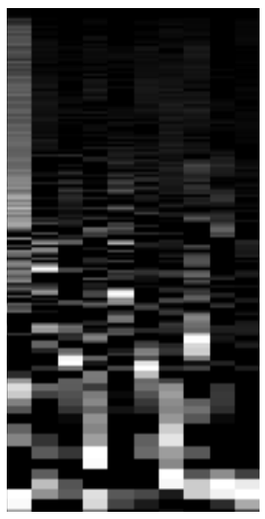
≈

activation templates

×

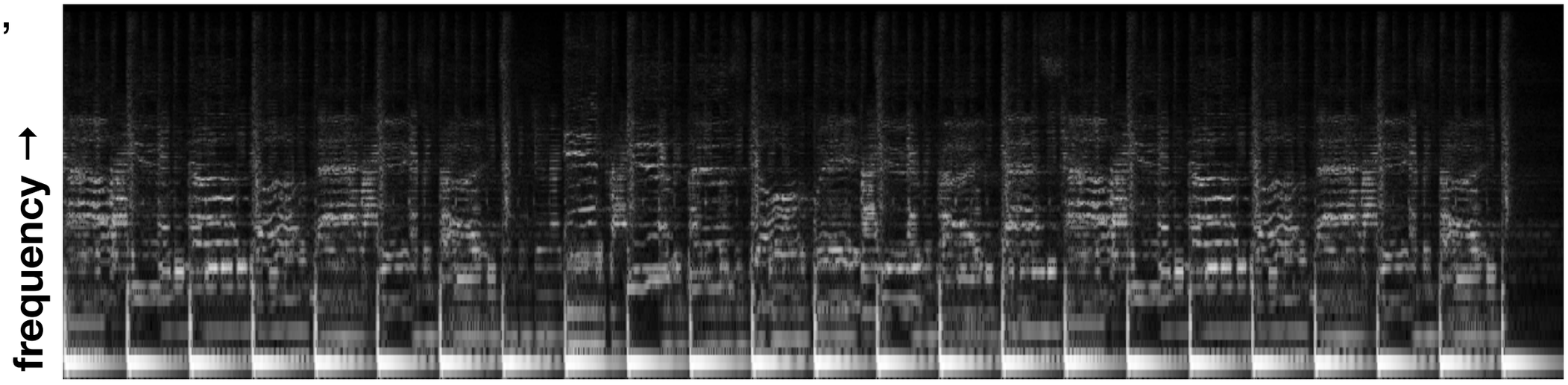
frequency templates



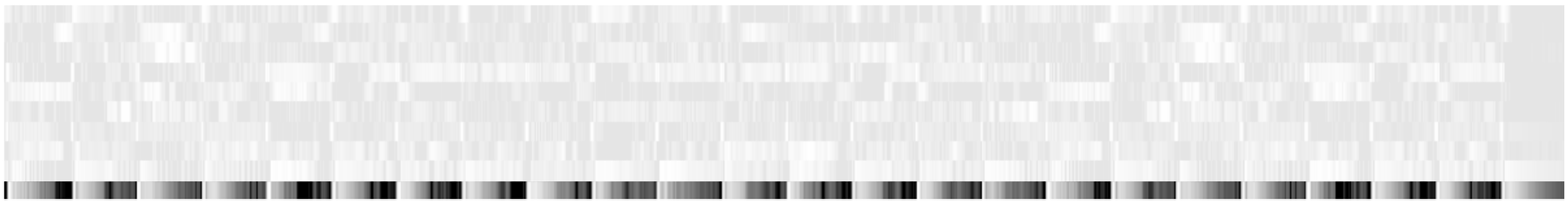


Regular NMF

Song: “Doin’ it Right” by Daft Punk



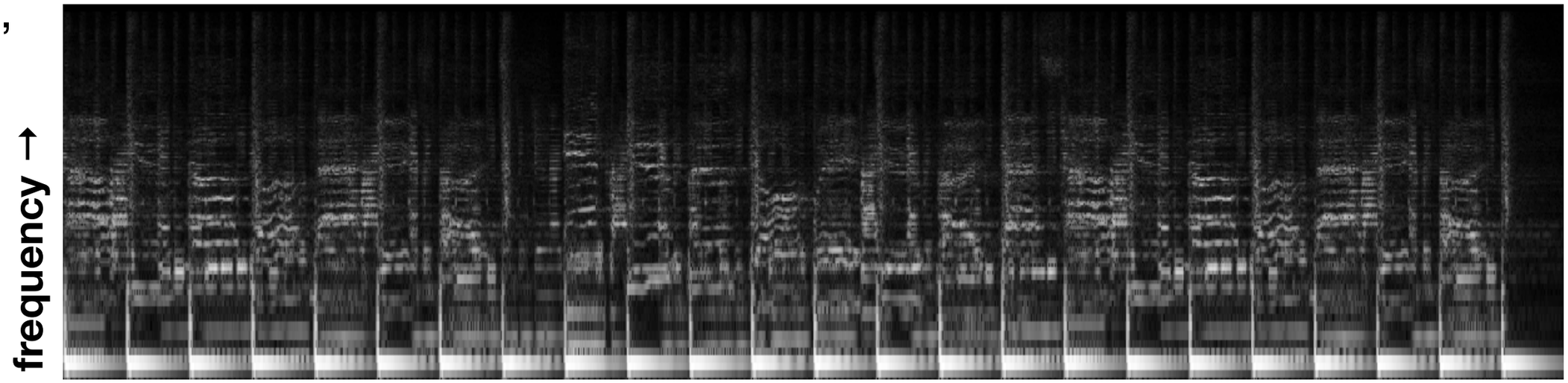
≈



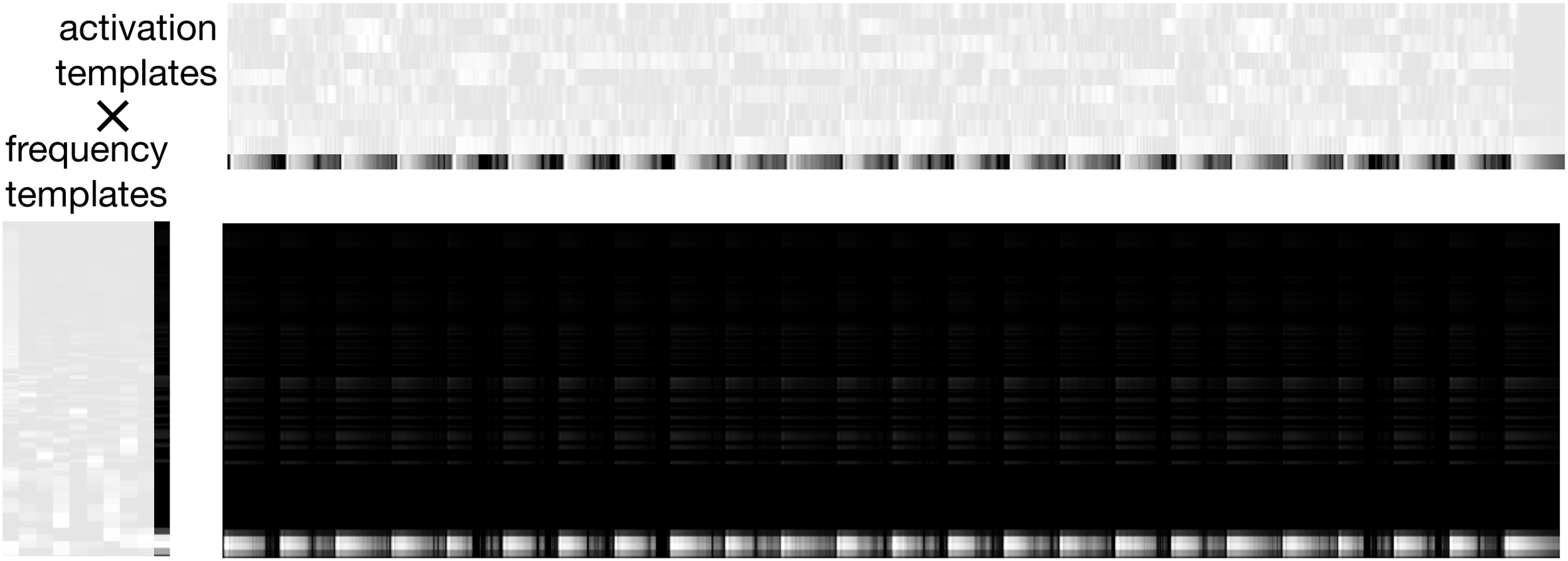
×

Regular NMF

Song: “Doin’ it Right” by Daft Punk

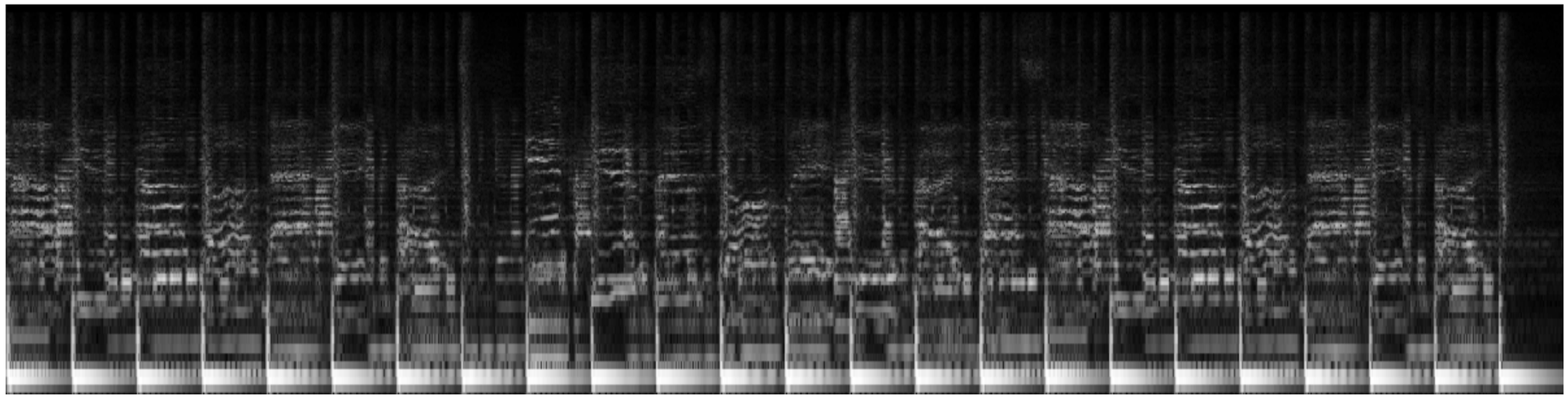


≈



Regular NMF

Song: “Doin’ it Right” by Daft Punk

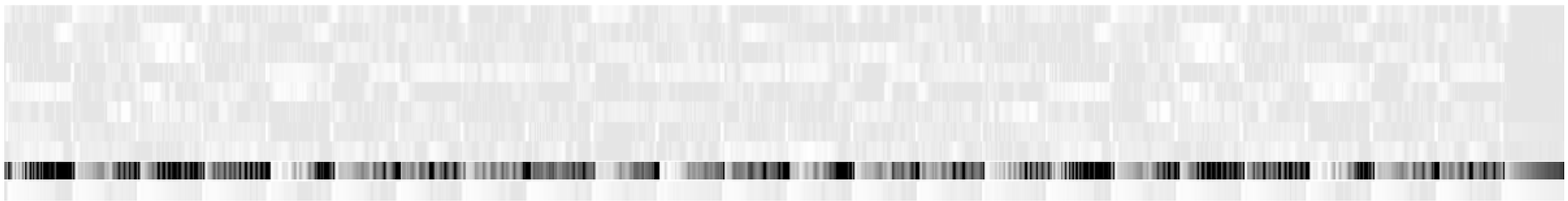


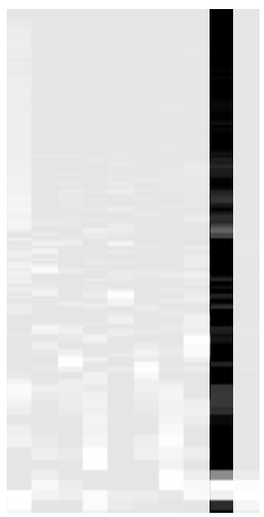
≈

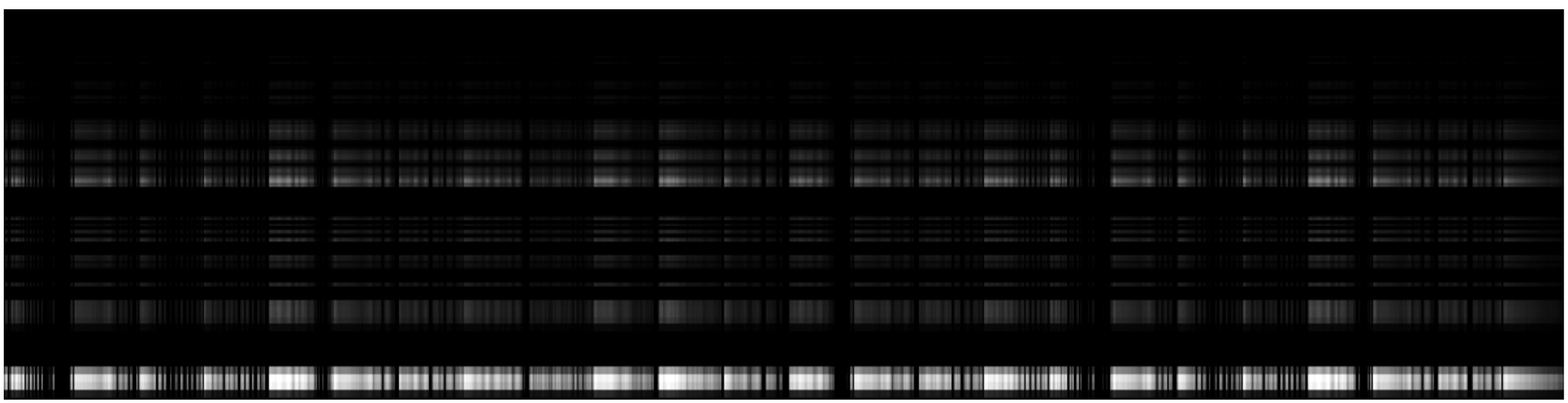
activation templates

×

frequency templates

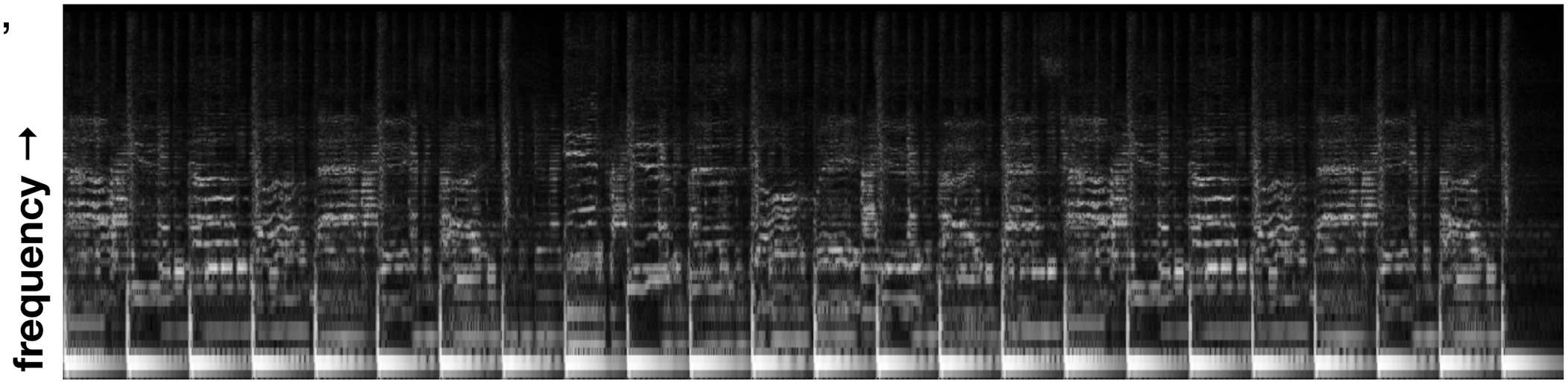






Regular NMF

Song: “Doin’ it Right” by Daft Punk

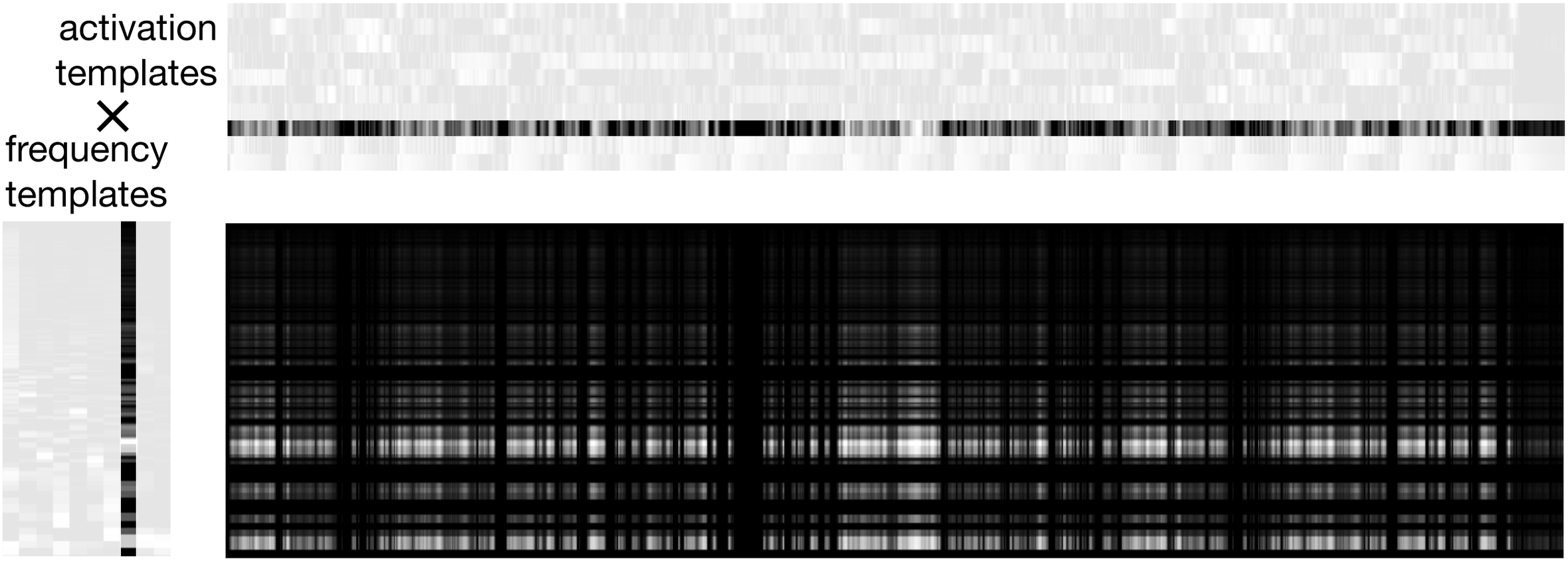


≈

activation templates

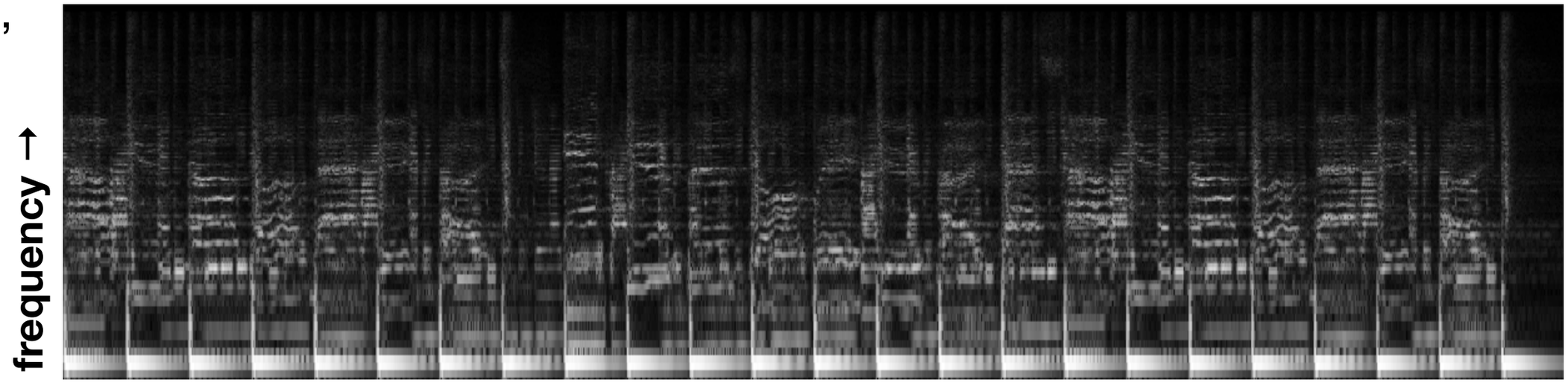
×

frequency templates



Regular NMF

Song: “Doin’ it Right” by Daft Punk

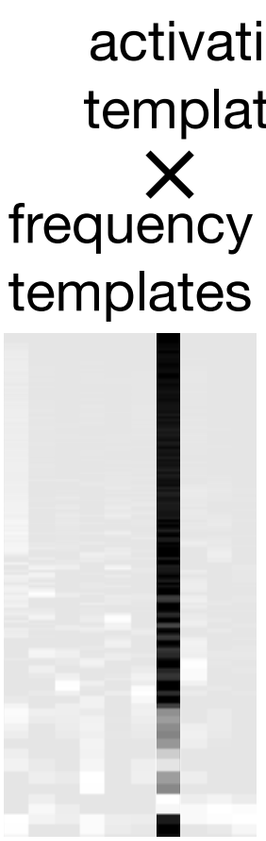


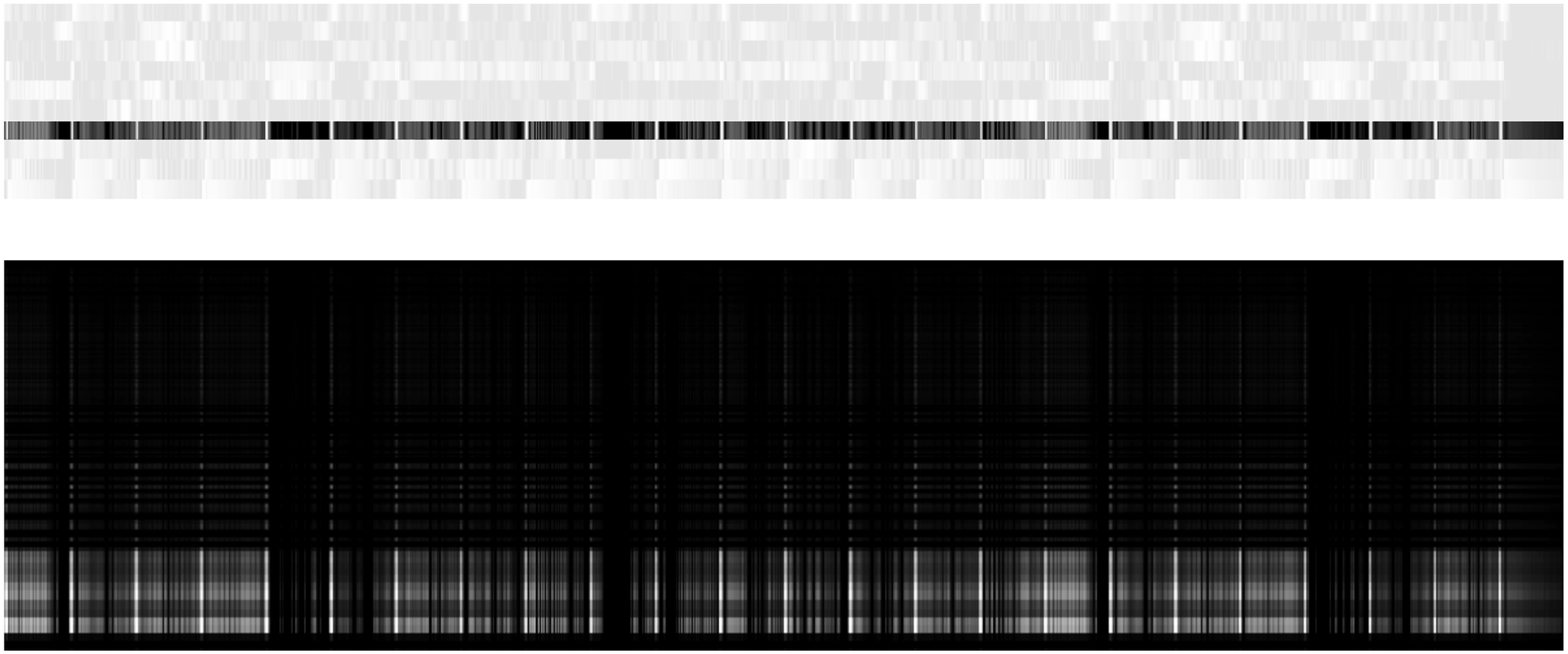
≈

activation templates

×

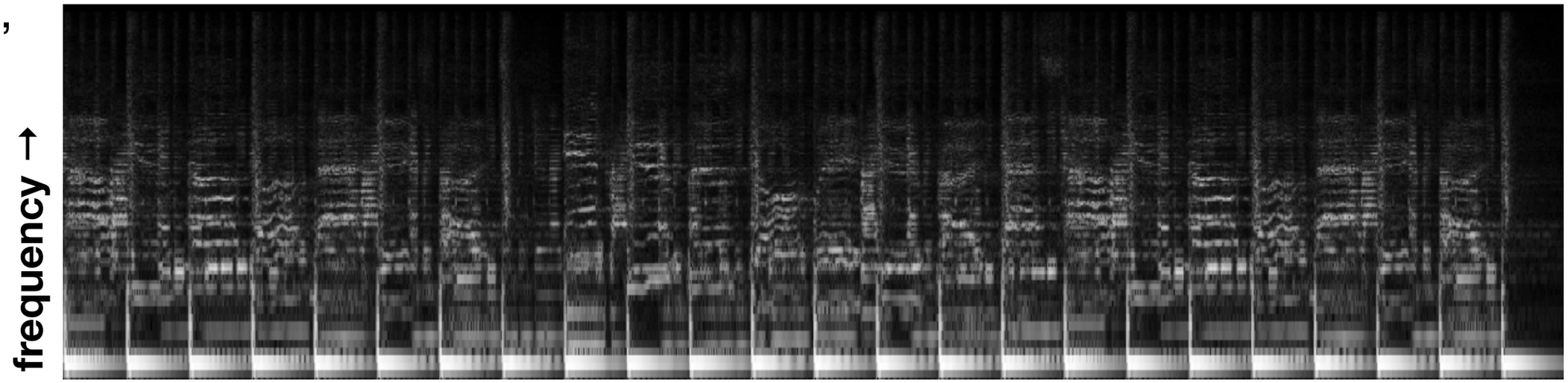
frequency templates





Regular NMF

Song: “Doin’ it Right” by Daft Punk

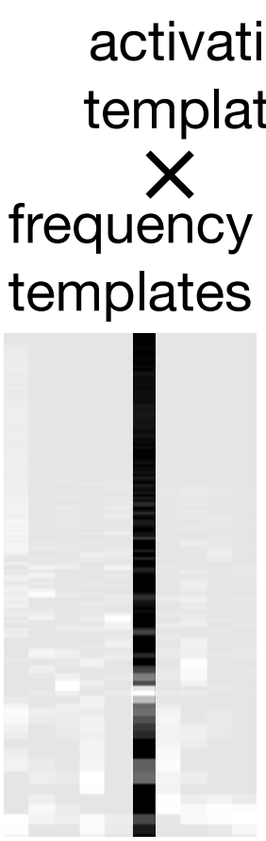


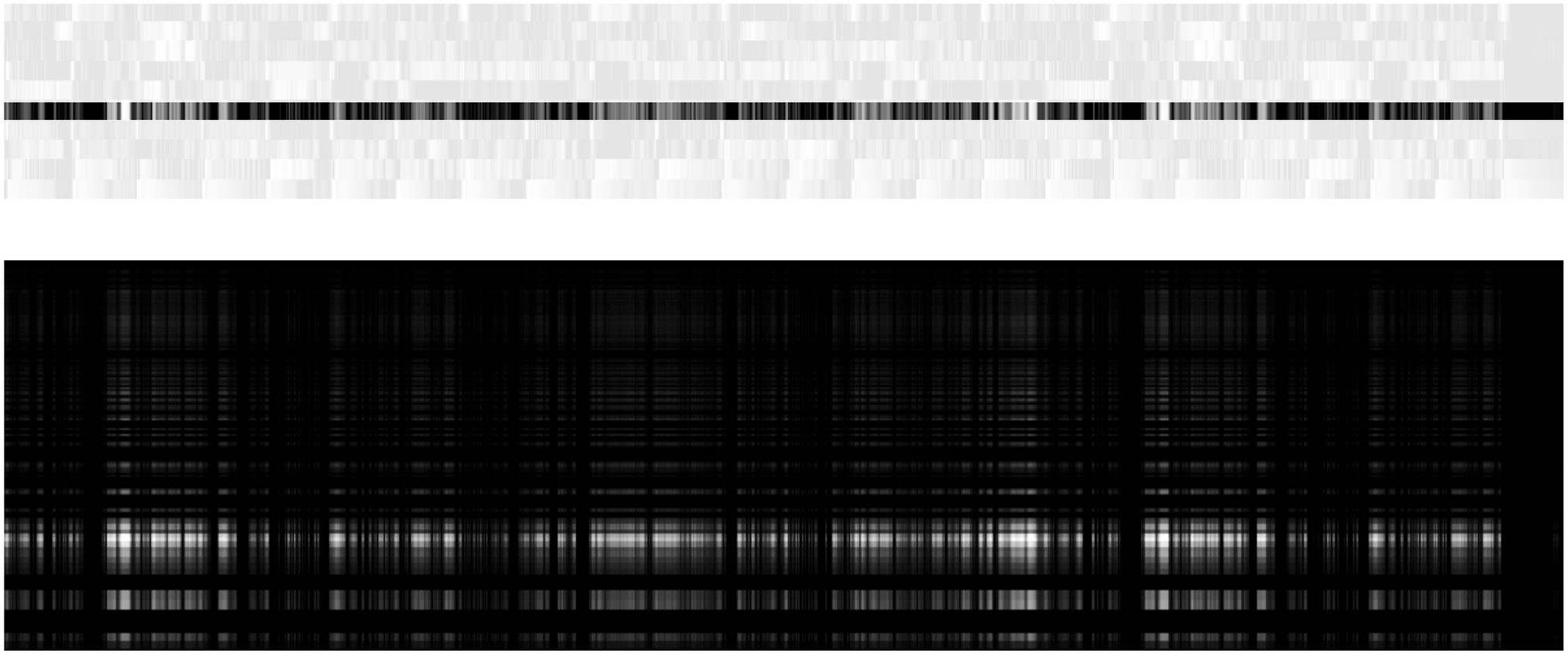
≈

activation templates

×

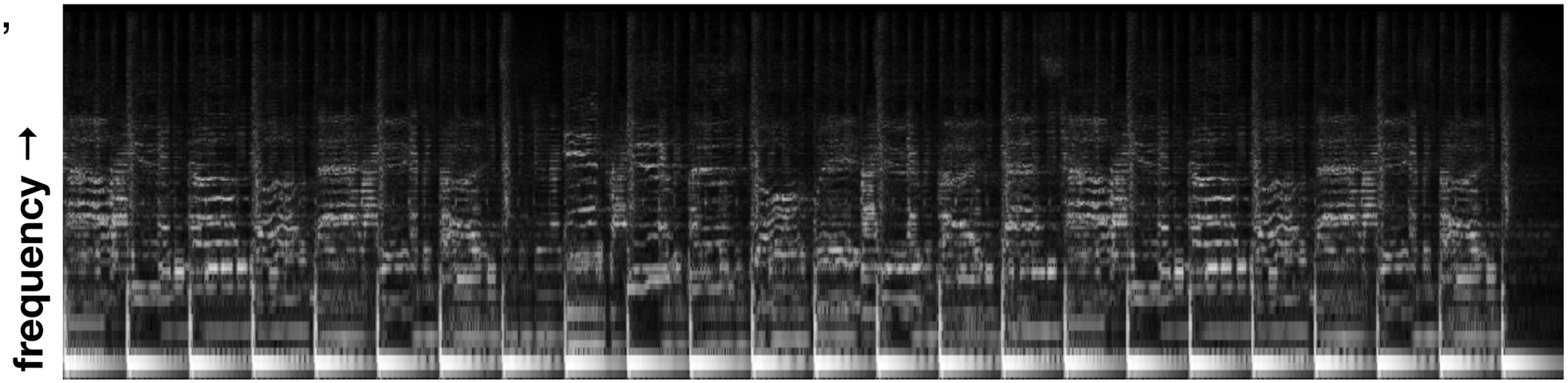
frequency templates





Regular NMF

Song: “Doin’ it Right” by Daft Punk

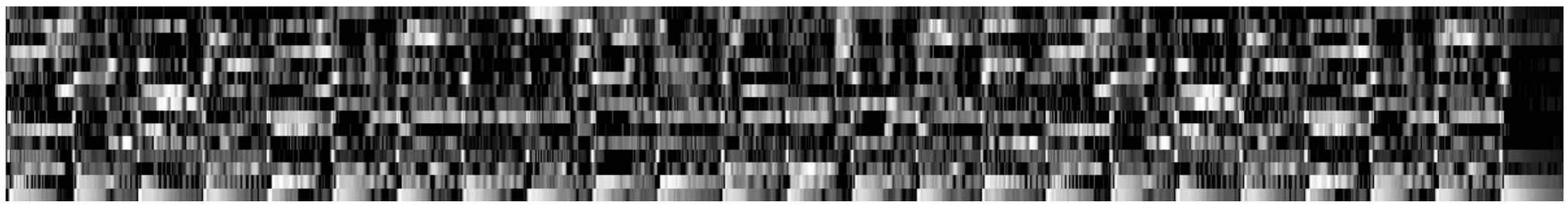


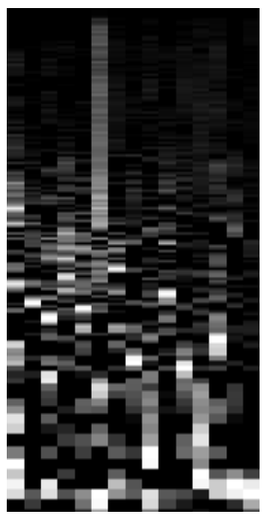
≈

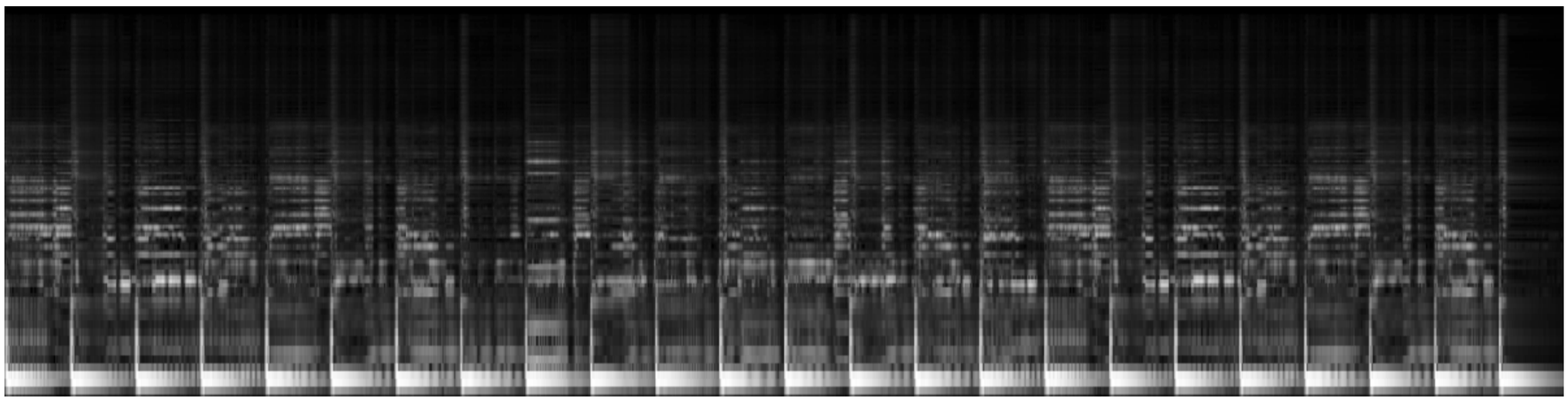
activation templates

×

frequency templates

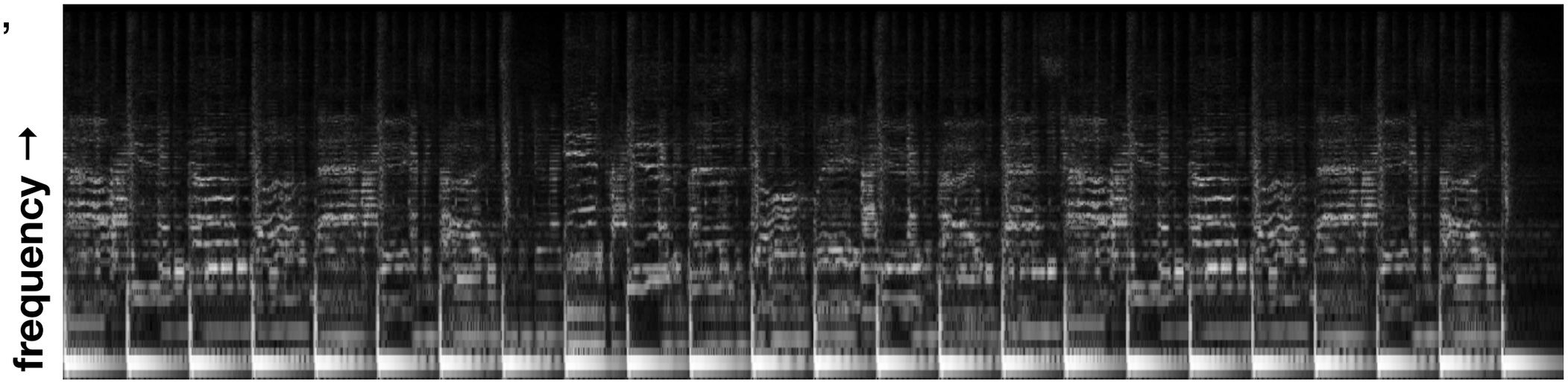




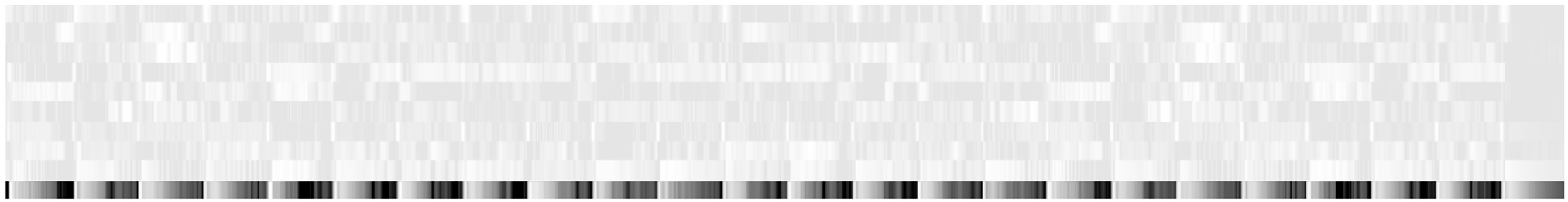


Regular NMF

Song: “Doin’ it Right” by Daft Punk



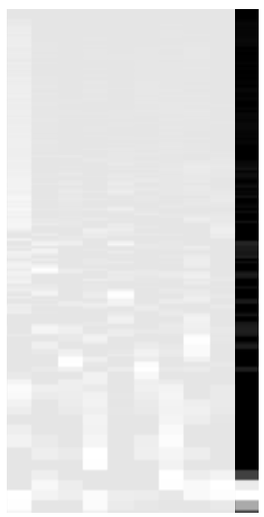
≈

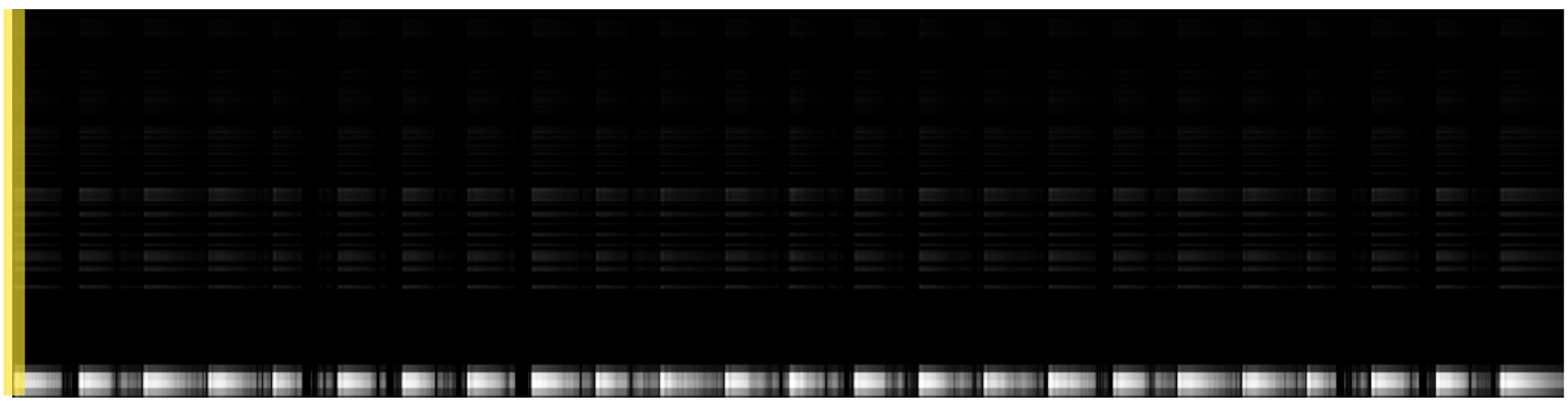


activation templates

×

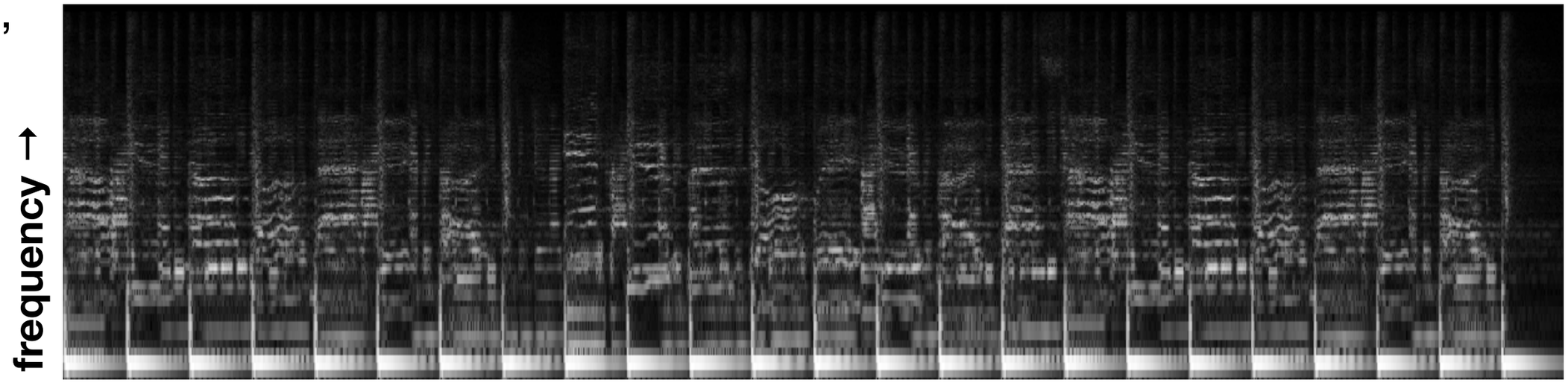
frequency templates





Regular NMF

Song: “Doin’ it Right” by Daft Punk

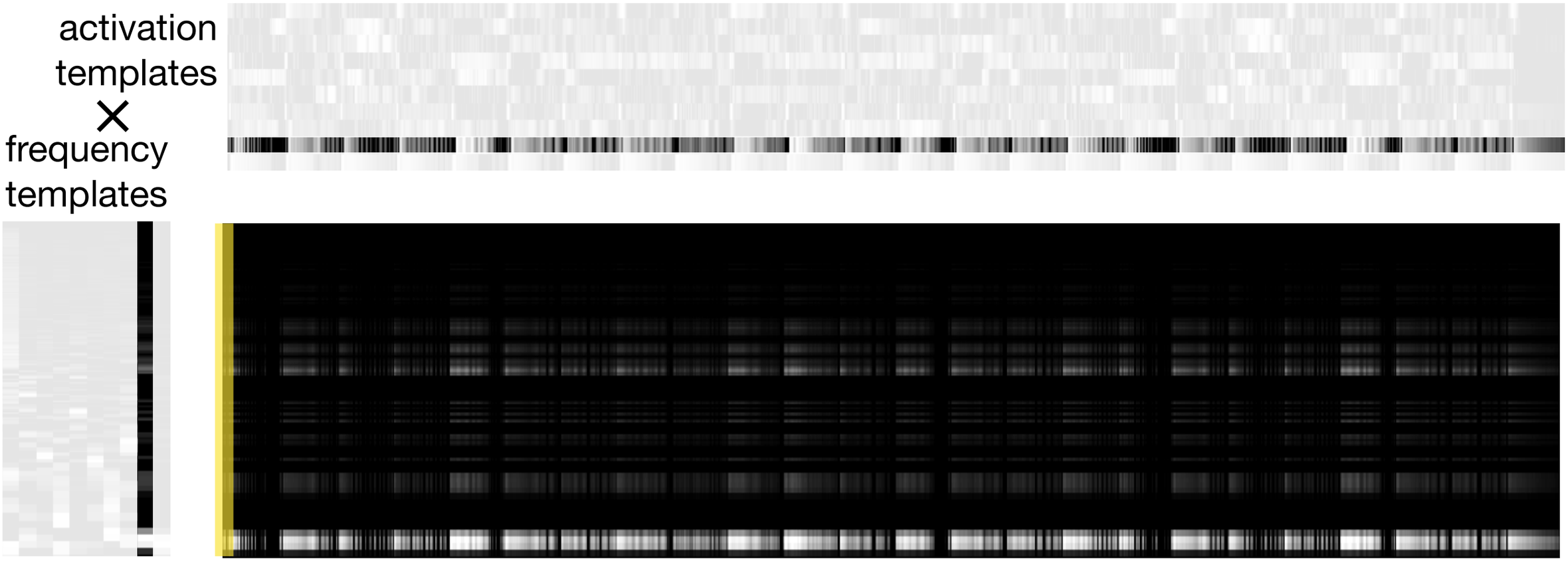


≈

activation templates

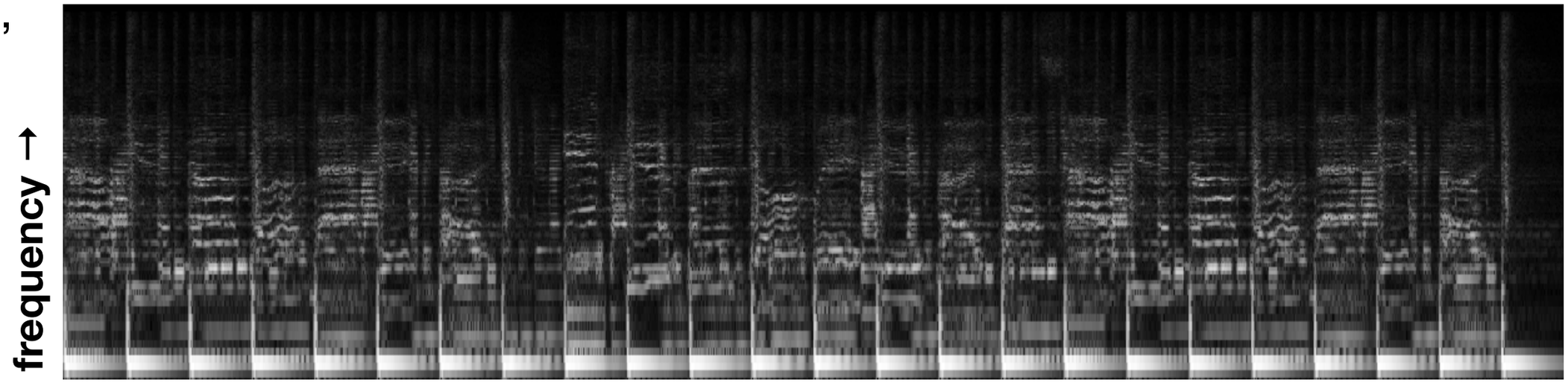
×

frequency templates



Regular NMF

Song: “Doin’ it Right” by Daft Punk

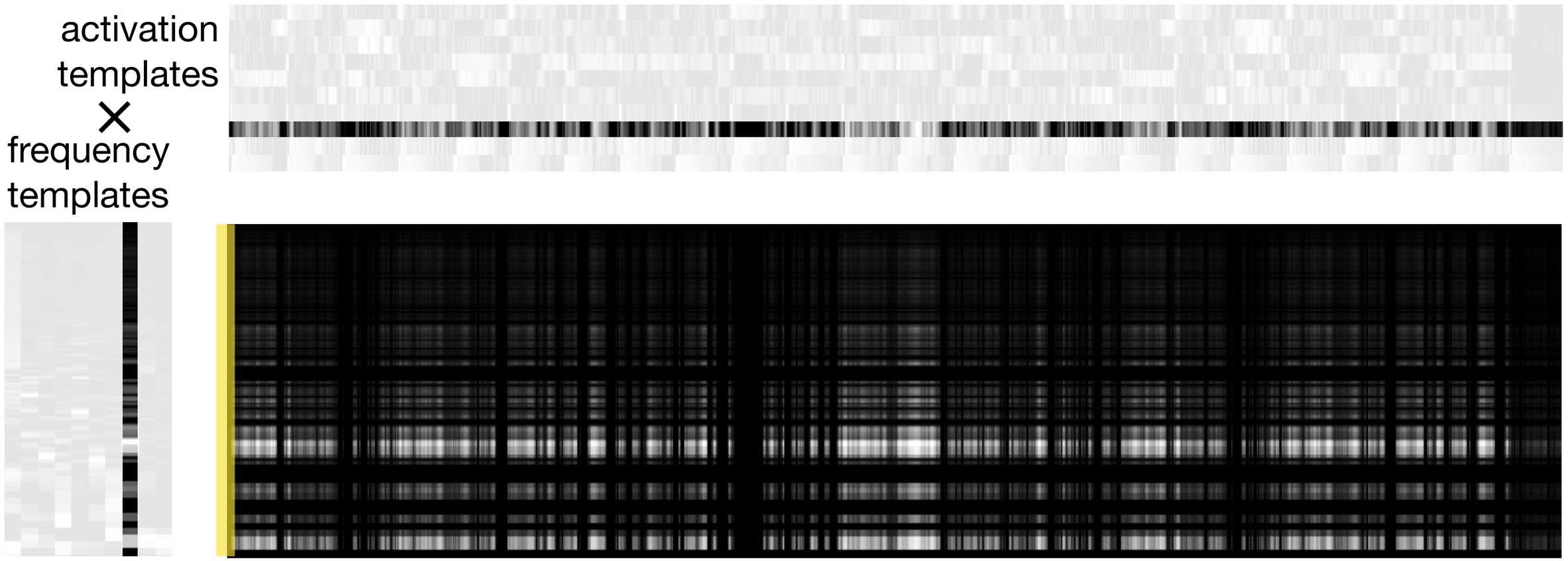


≈

activation templates

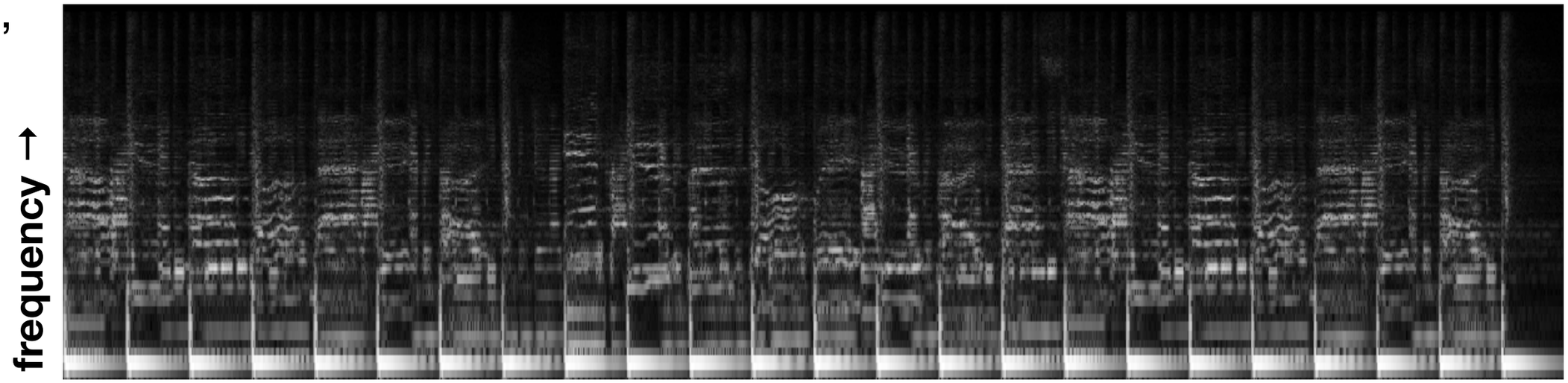
×

frequency templates



Regular NMF

Song: “Doin’ it Right” by Daft Punk

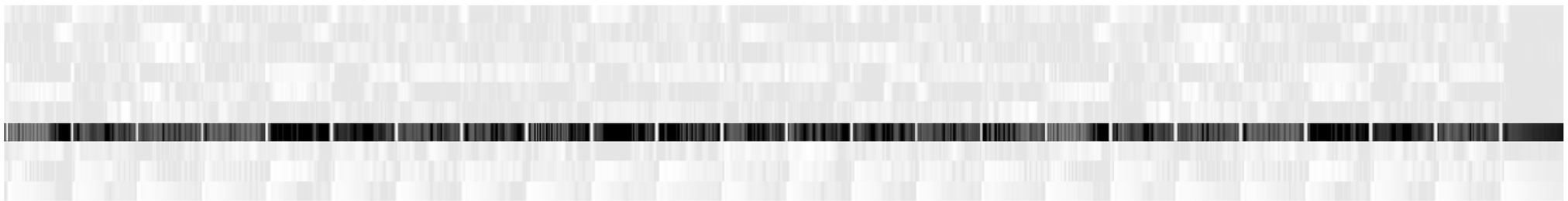


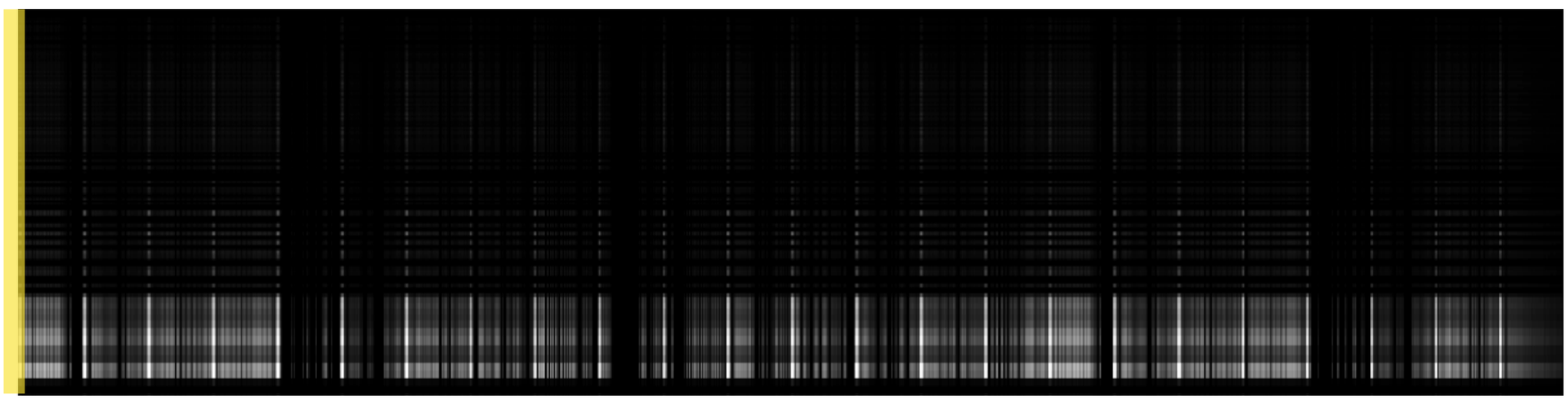
≈

activation templates

×

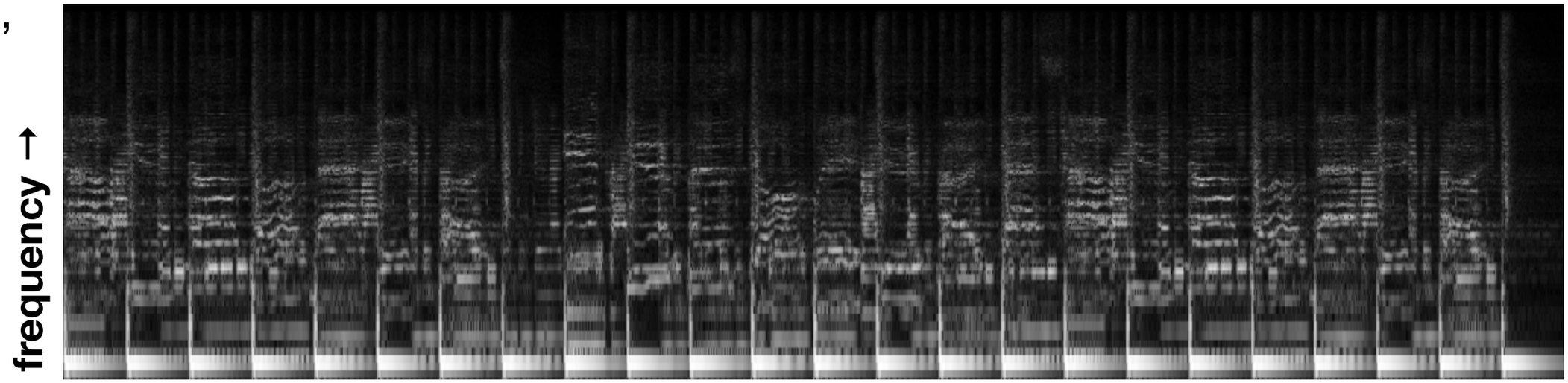
frequency templates



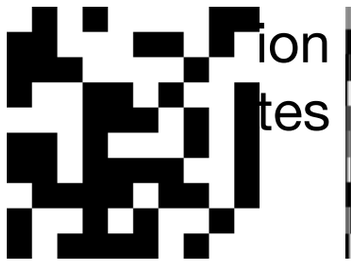


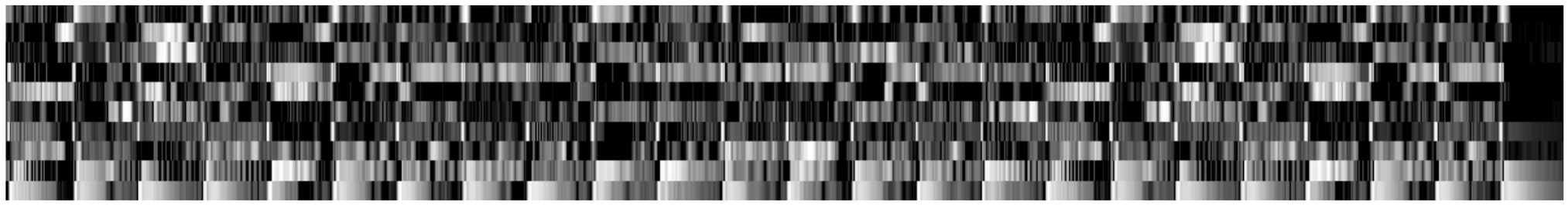
Regular NMF

Song: “Doin’ it Right” by Daft Punk



≈



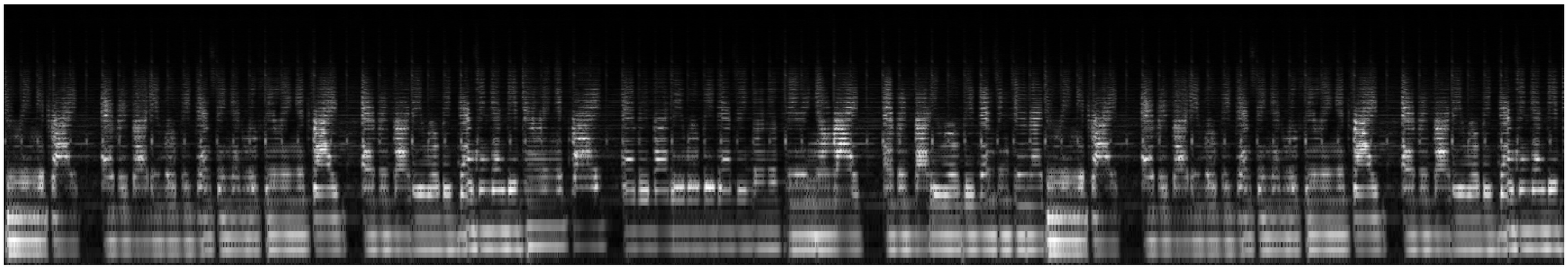


- Problem: how to group templates that belong to the same source?
  - **Unmixer** model: uses periodic repetition
- Note: templates are all paired 1-to-1.
  - **Unmixer** model: what if we allowed inter-mixing?

Unmixer Algorithm

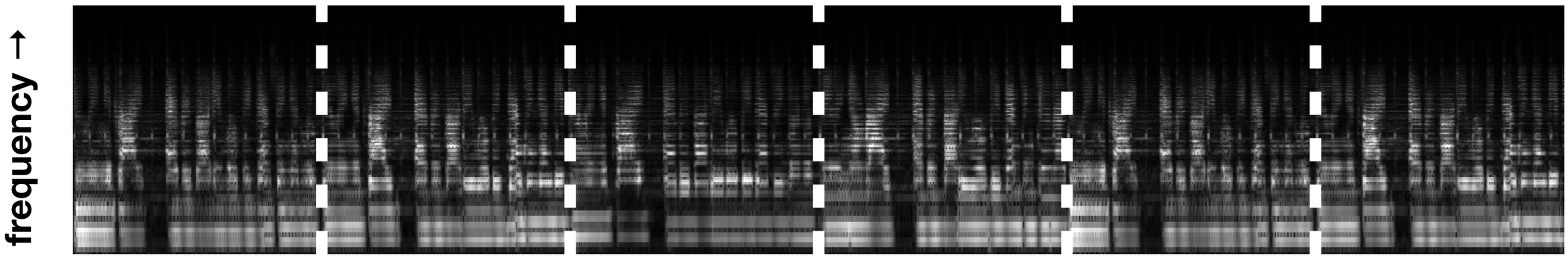
Algorithm

1. Compute spectrum



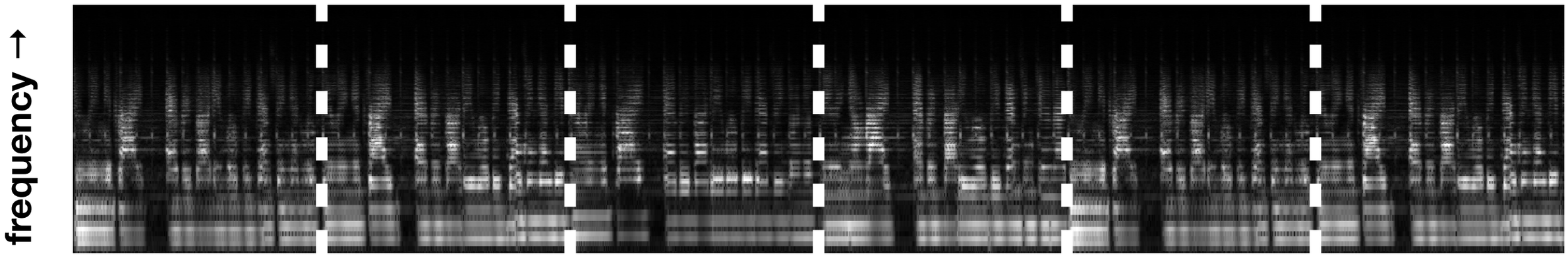
Algorithm

1. **Compute spectrum**
2. **Estimate downbeats (madmom)**



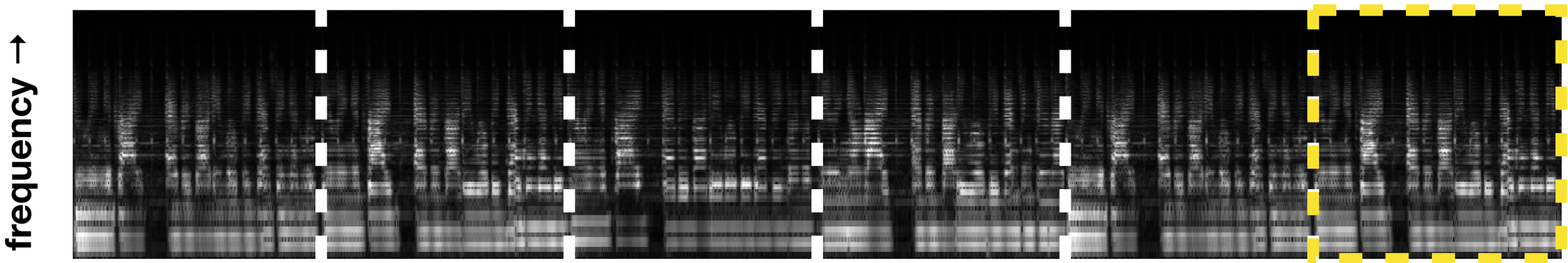
Algorithm

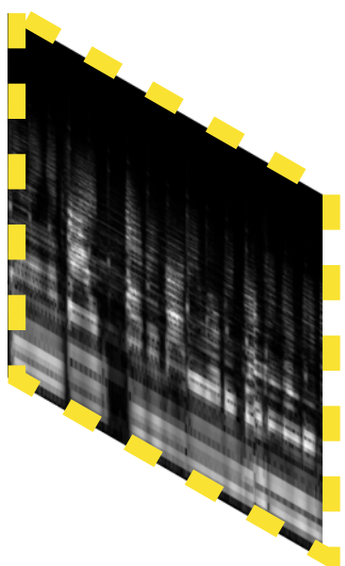
1. **Compute spectrum**
2. **Estimate downbeats (madmom)**
3. **Stack into cube**



Algorithm

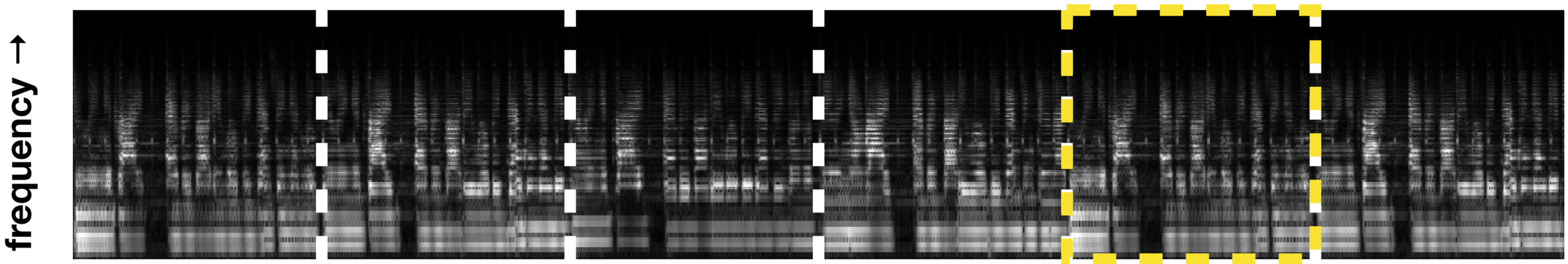
1. **Compute spectrum**
2. **Estimate downbeats (madmom)**
3. **Stack into cube**

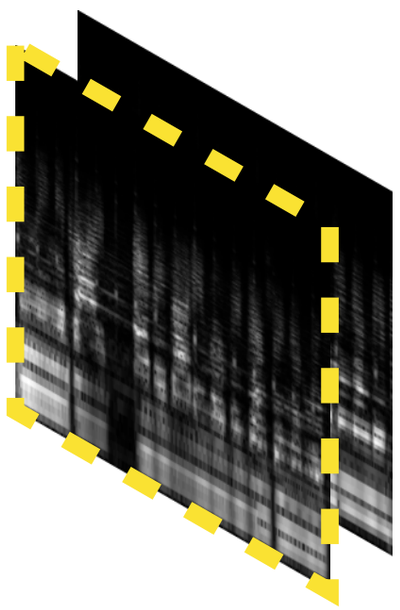




Algorithm

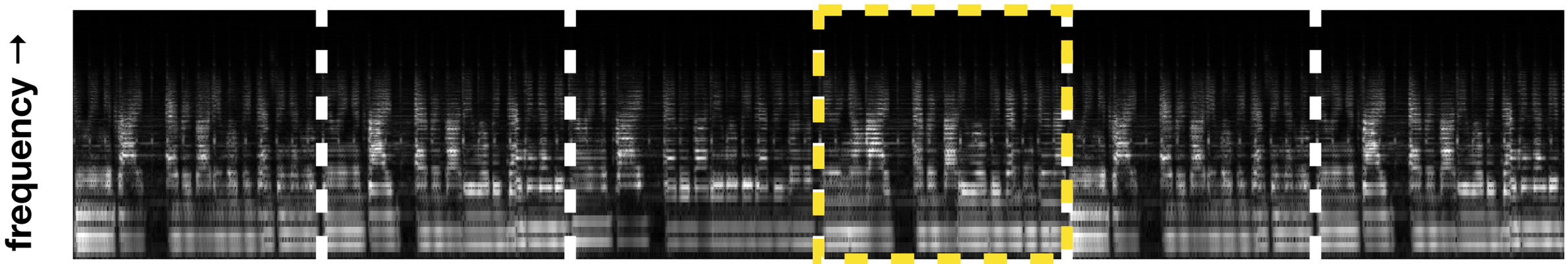
1. **Compute spectrum**
2. **Estimate downbeats (madmom)**
3. **Stack into cube**

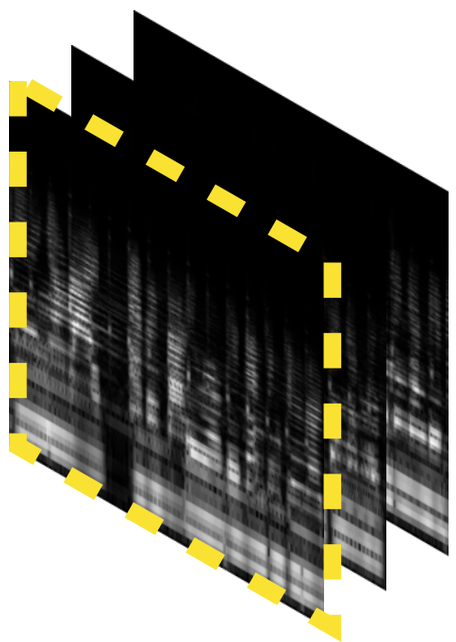




Algorithm

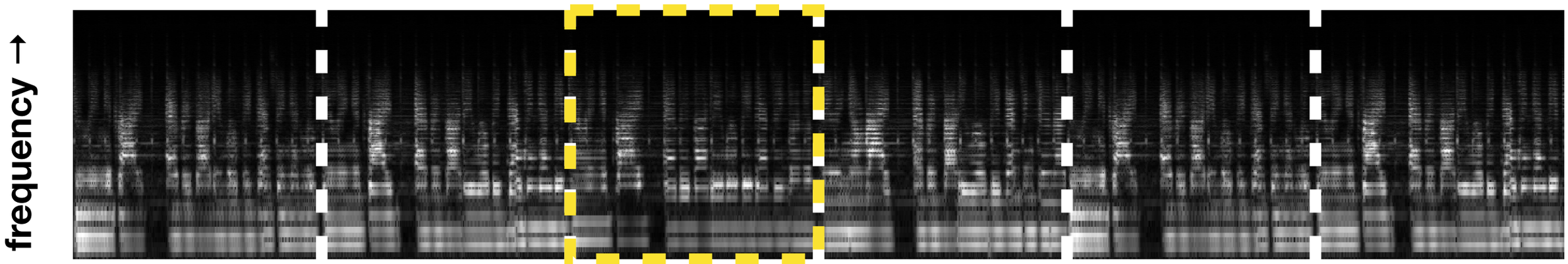
1. **Compute spectrum**
2. **Estimate downbeats (madmom)**
3. **Stack into cube**

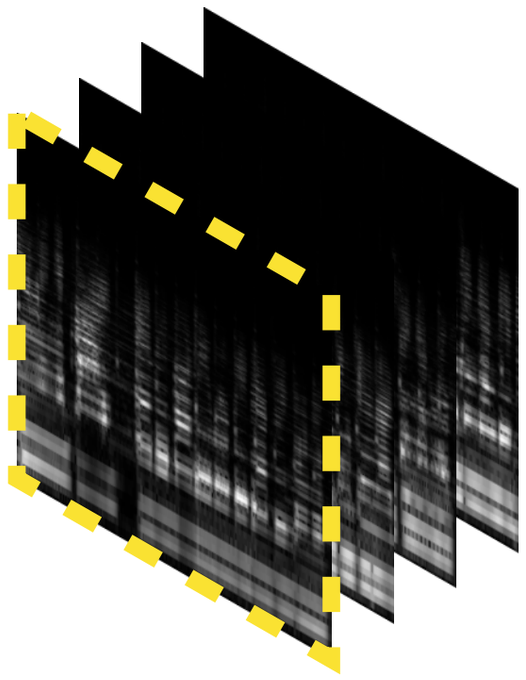




Algorithm

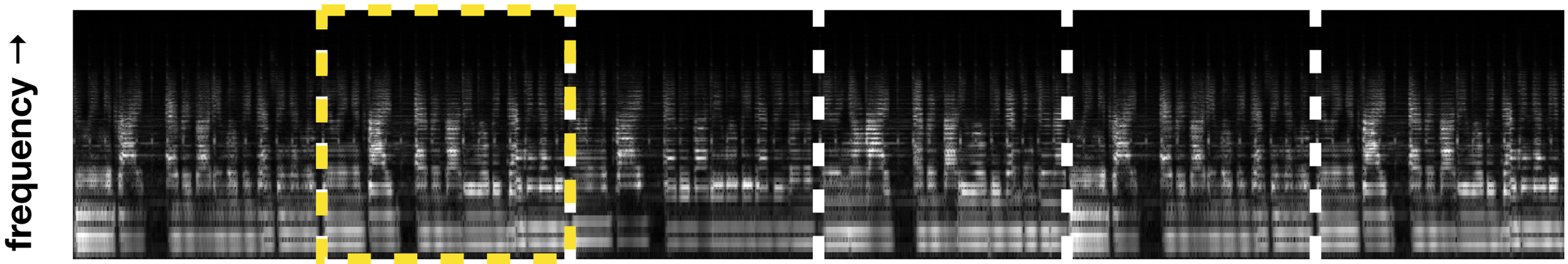
1. **Compute spectrum**
2. **Estimate downbeats (madmom)**
3. **Stack into cube**

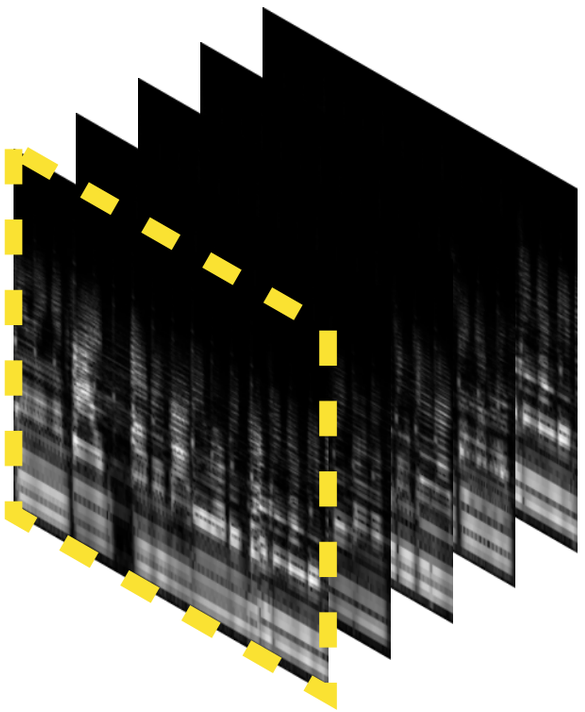




Algorithm

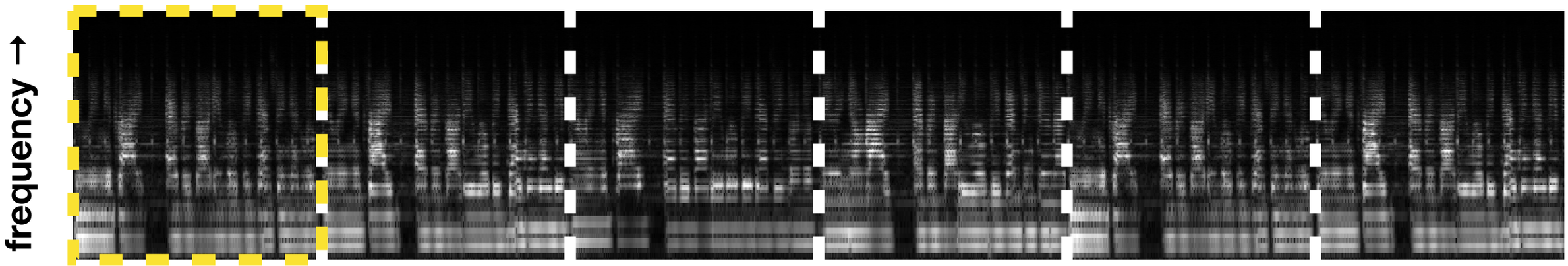
1. **Compute spectrum**
2. **Estimate downbeats (madmom)**
3. **Stack into cube**

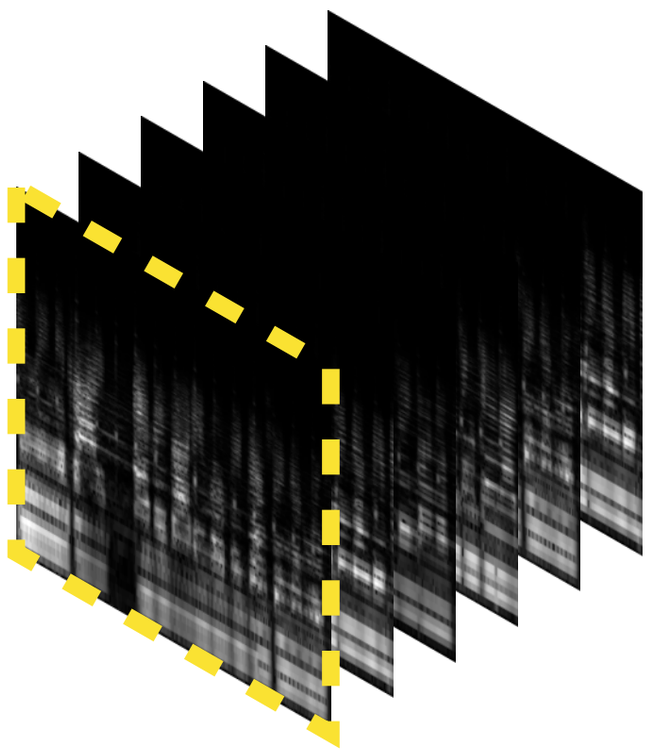




Algorithm

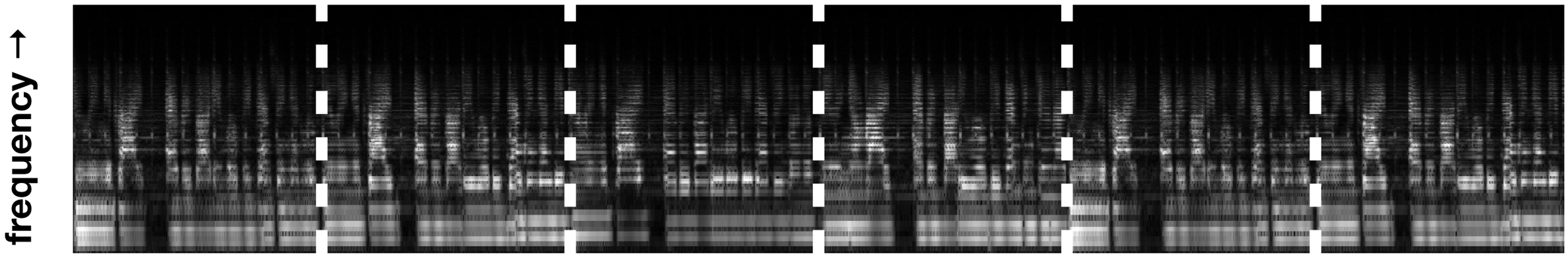
1. **Compute spectrum**
2. **Estimate downbeats (madmom)**
3. **Stack into cube**

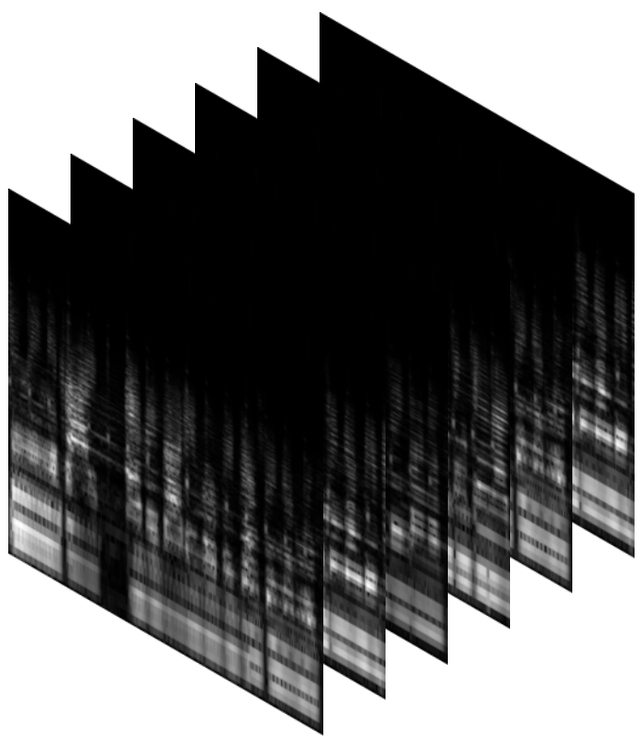




Algorithm

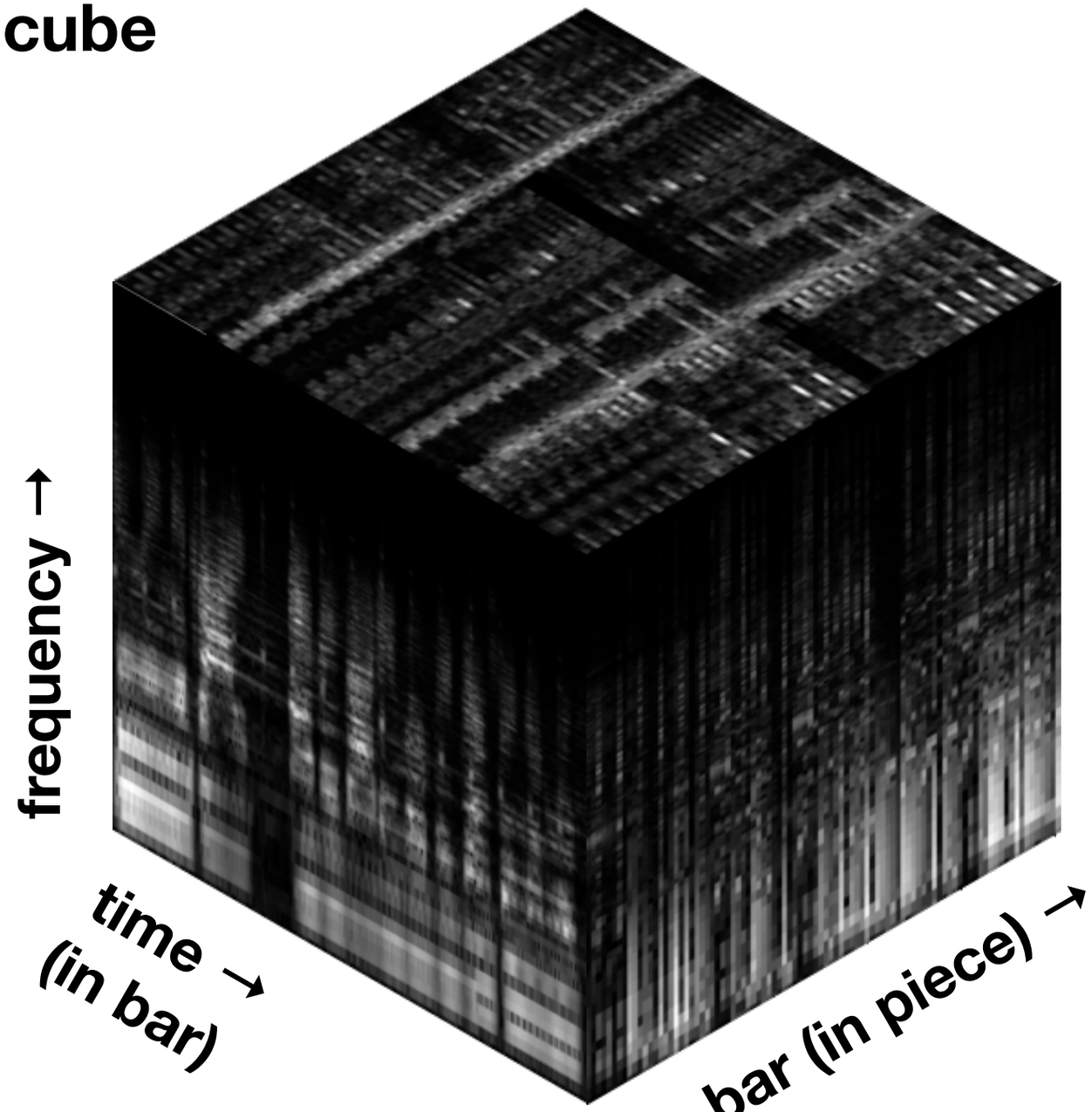
1. **Compute spectrum**
2. **Estimate downbeats (madmom)**
3. **Stack into cube**





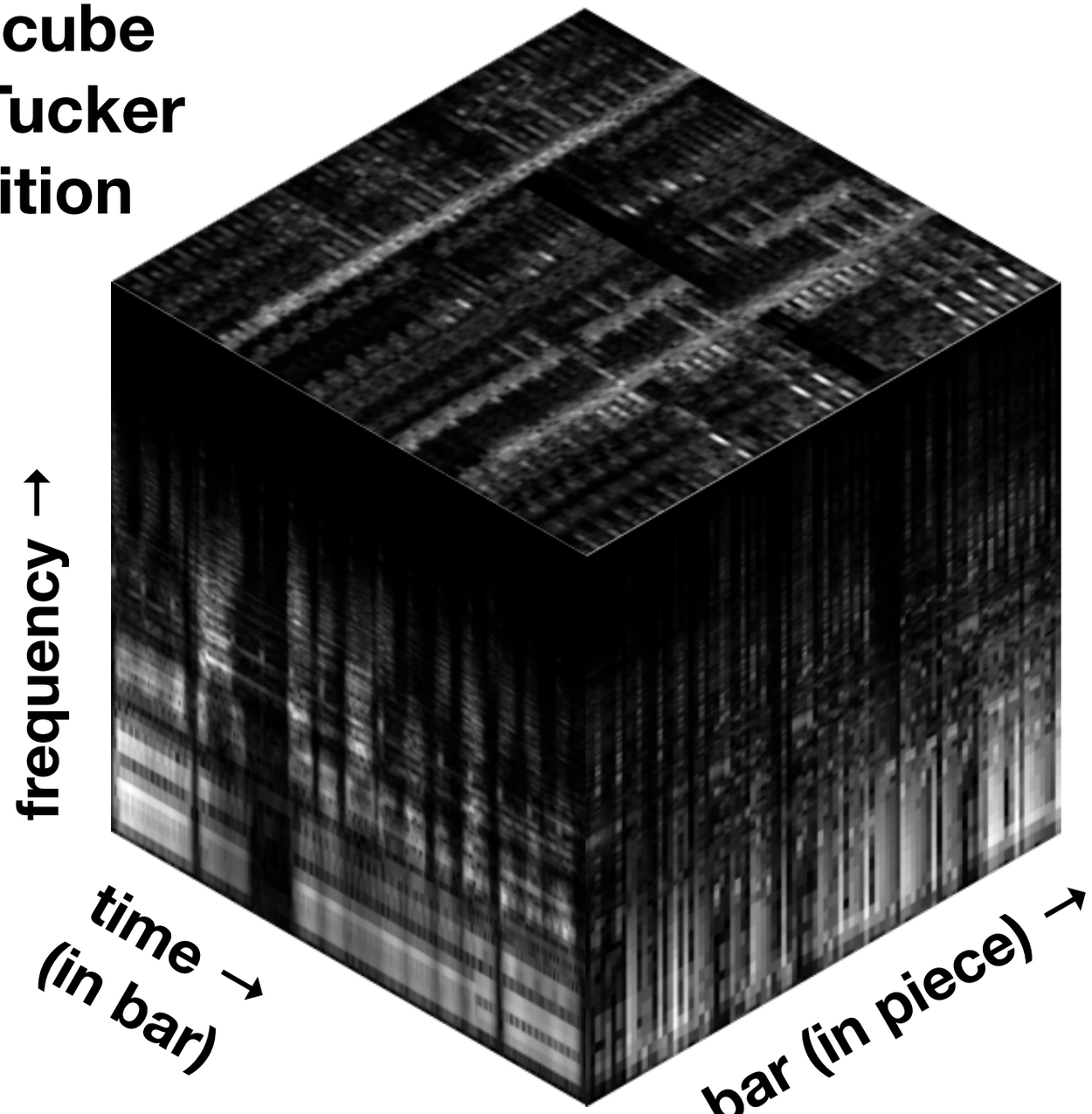
Algorithm

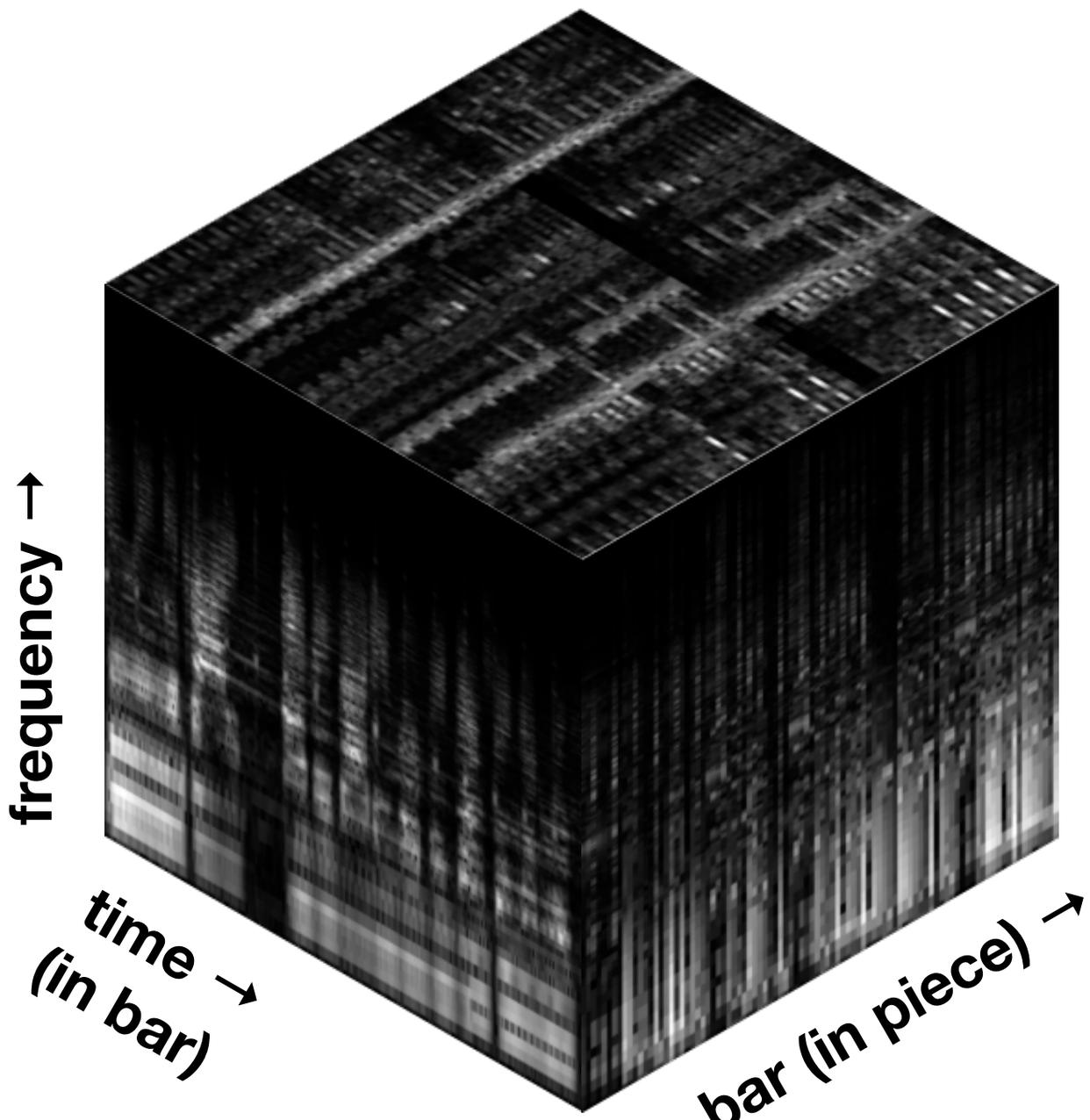
1. **Compute spectrum**
2. **Estimate downbeats (madmom)**
3. **Stack into cube**



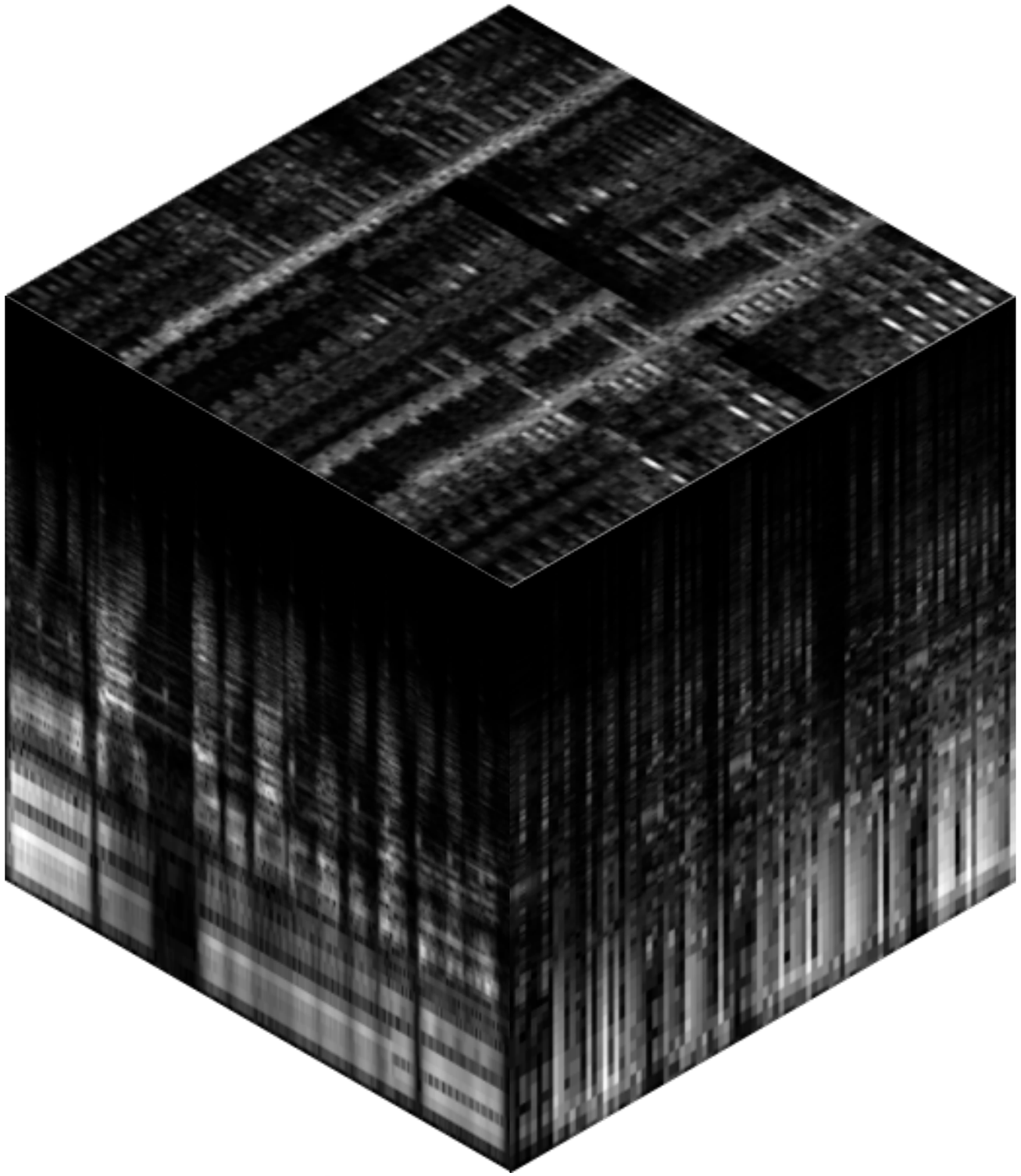
Algorithm

1. **Compute spectrum**
2. **Estimate downbeats (madmom)**
3. **Stack into cube**
4. **Compute Tucker decomposition**

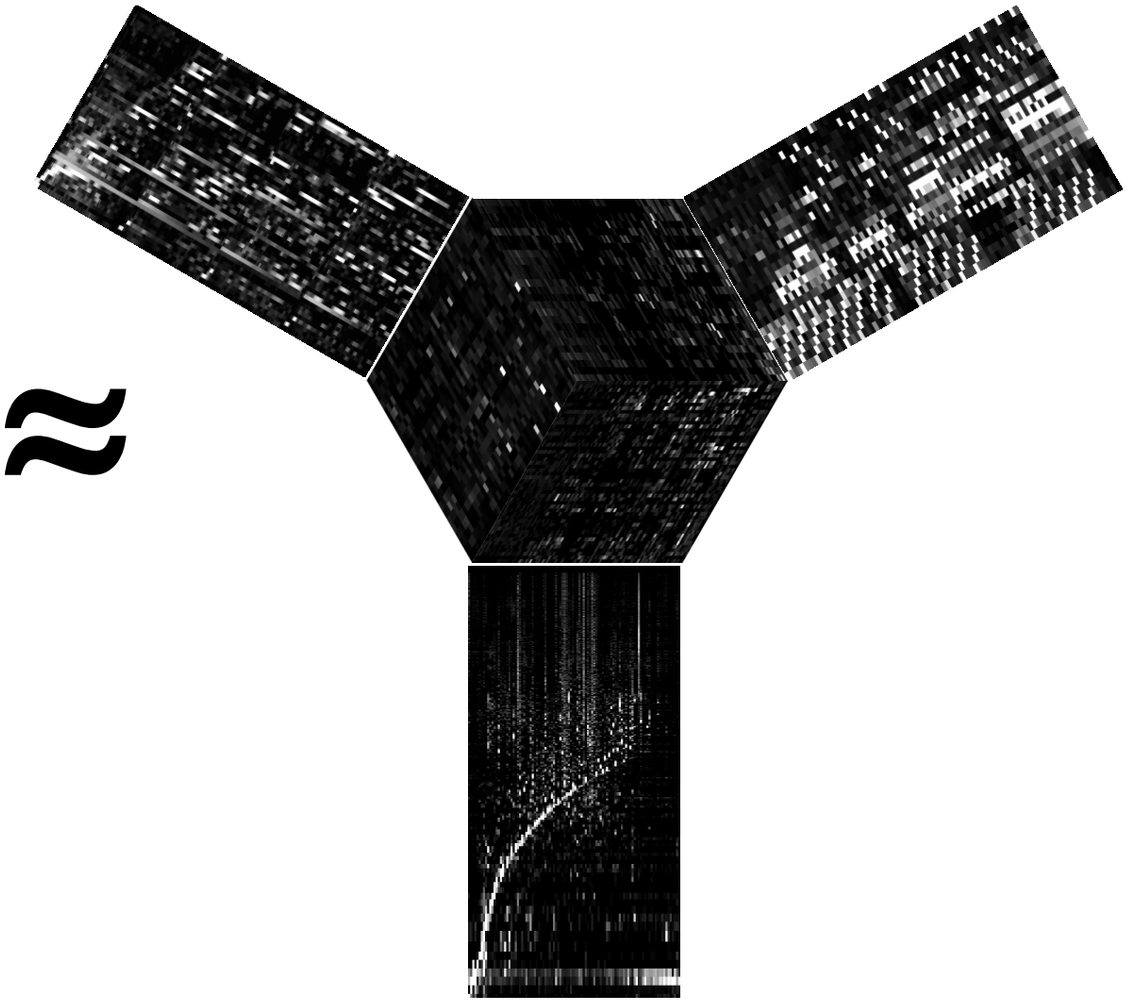



frequency →
time →
(in bar)
bar (in piece) →

Tucker decomposition:



≈

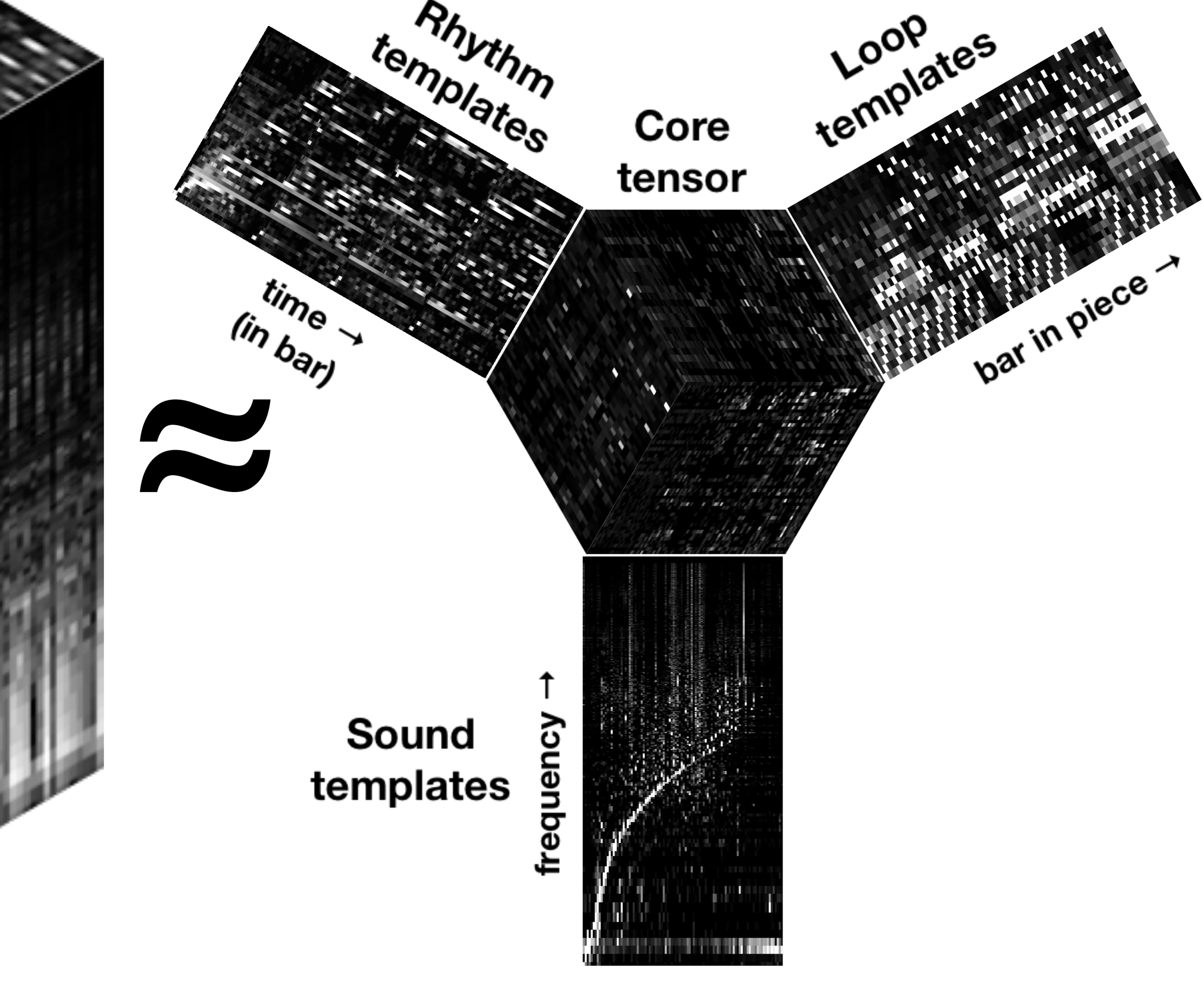


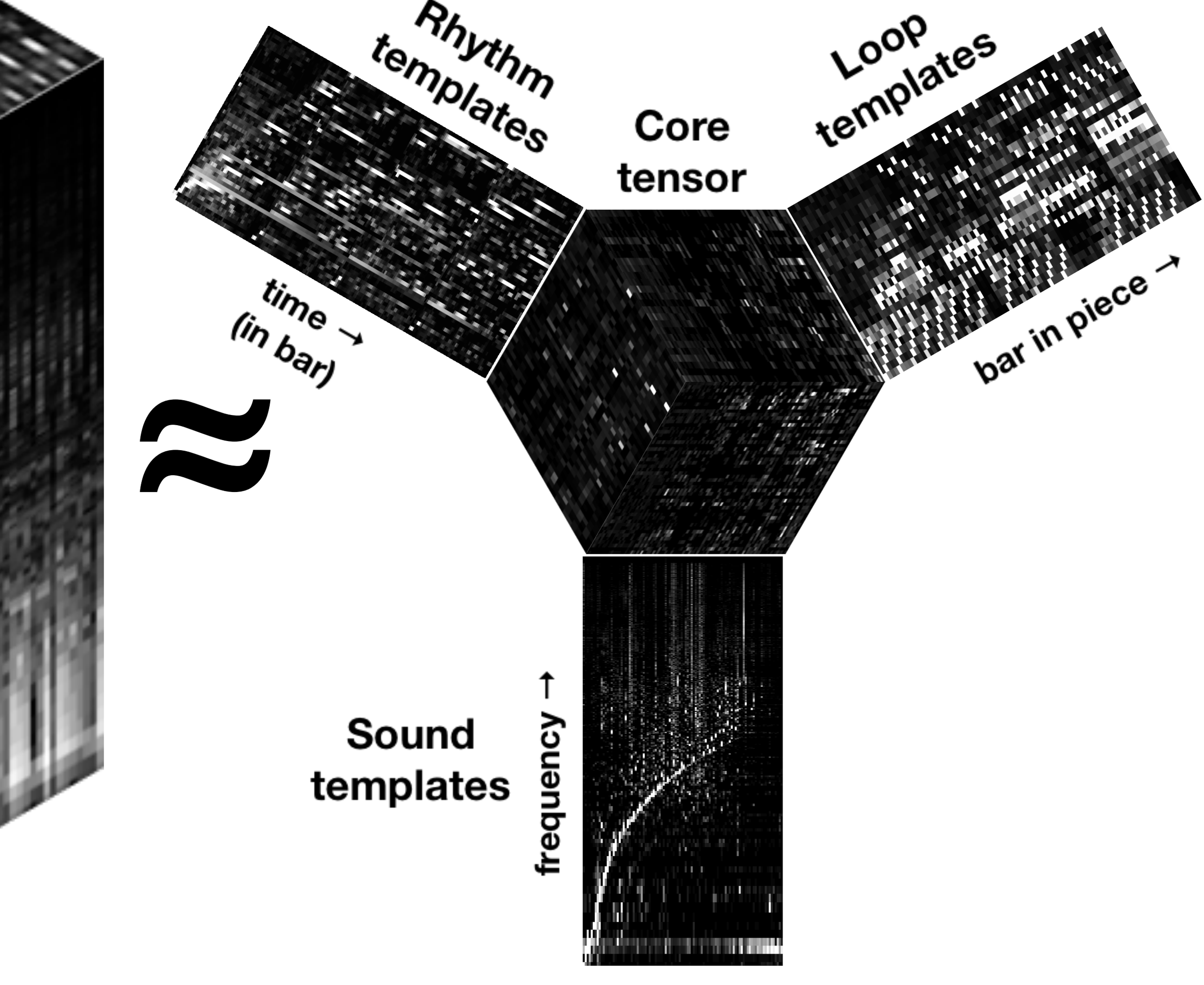
Spectral cube

~20M elements

Core tensor and 3 templates

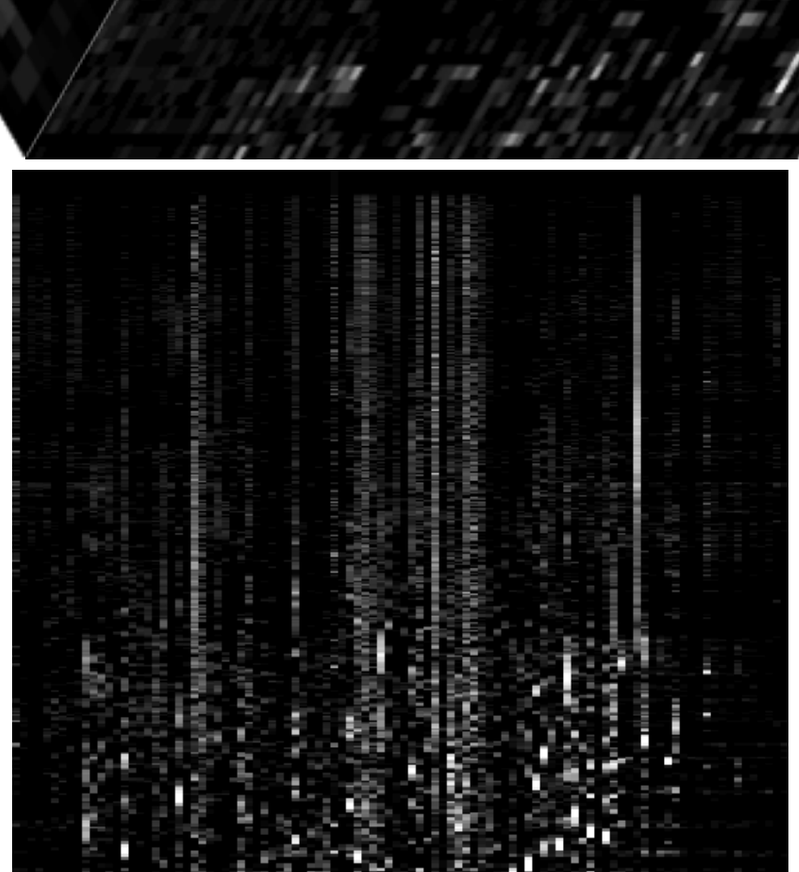
~0.2M elements


Rhythm templates
Core tensor
Loop templates
time → (in bar)
bar in piece →
Sound templates
frequency →


Rhythm templates
Core tensor
Loop templates
time → (in bar)
bar in piece →
Sound templates
frequency →

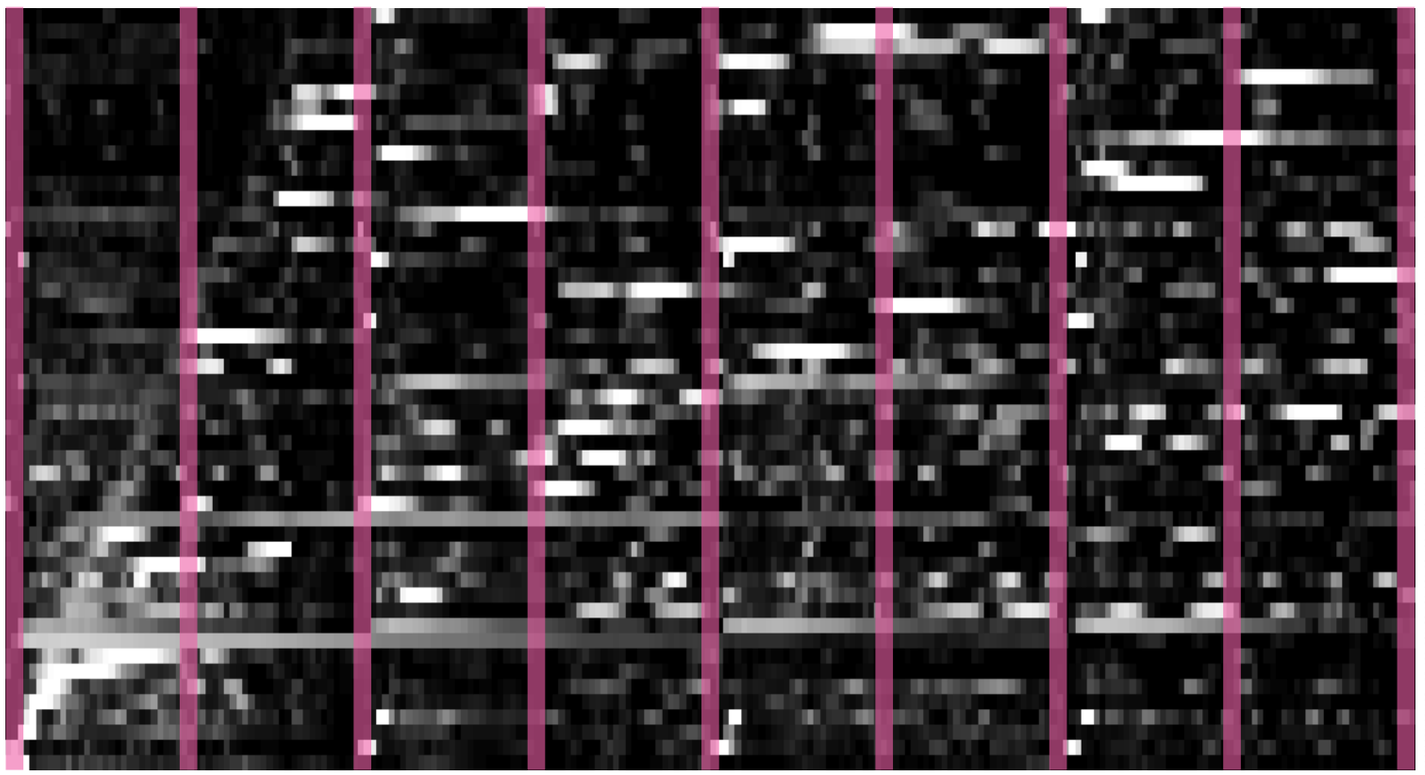
Sound templates

frequency →



Rhythm templates

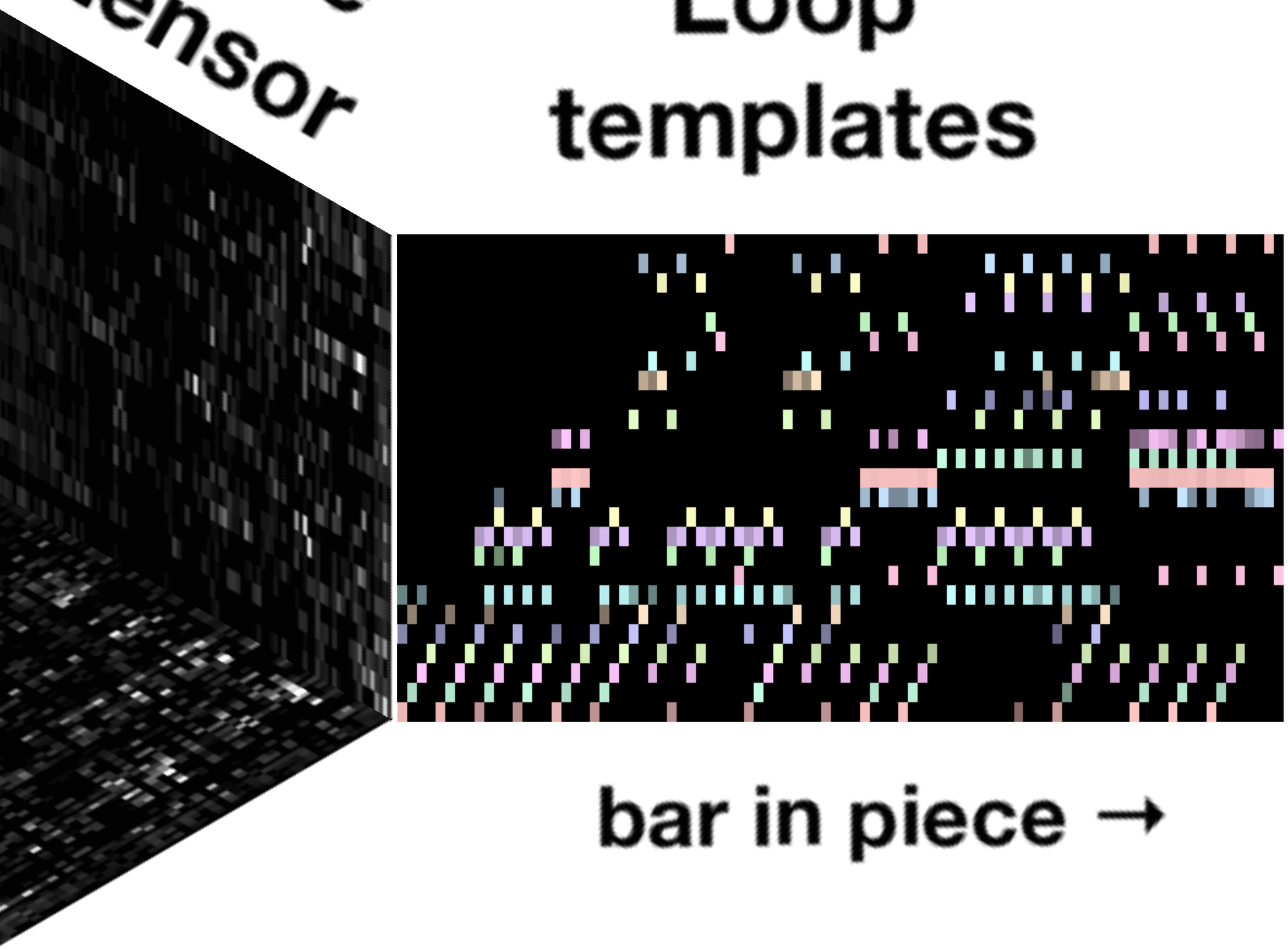
Core tensor



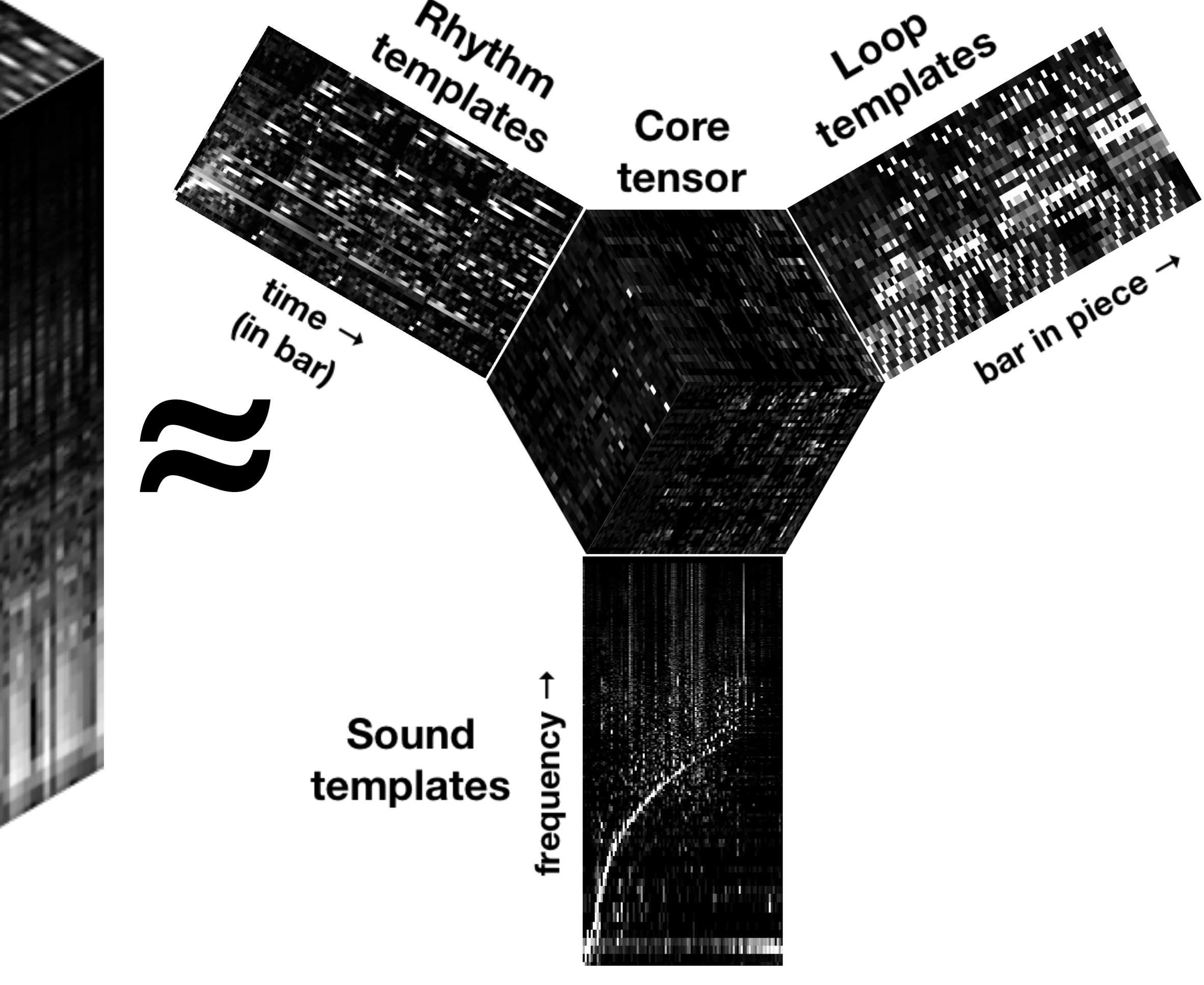
time →
(in bar)

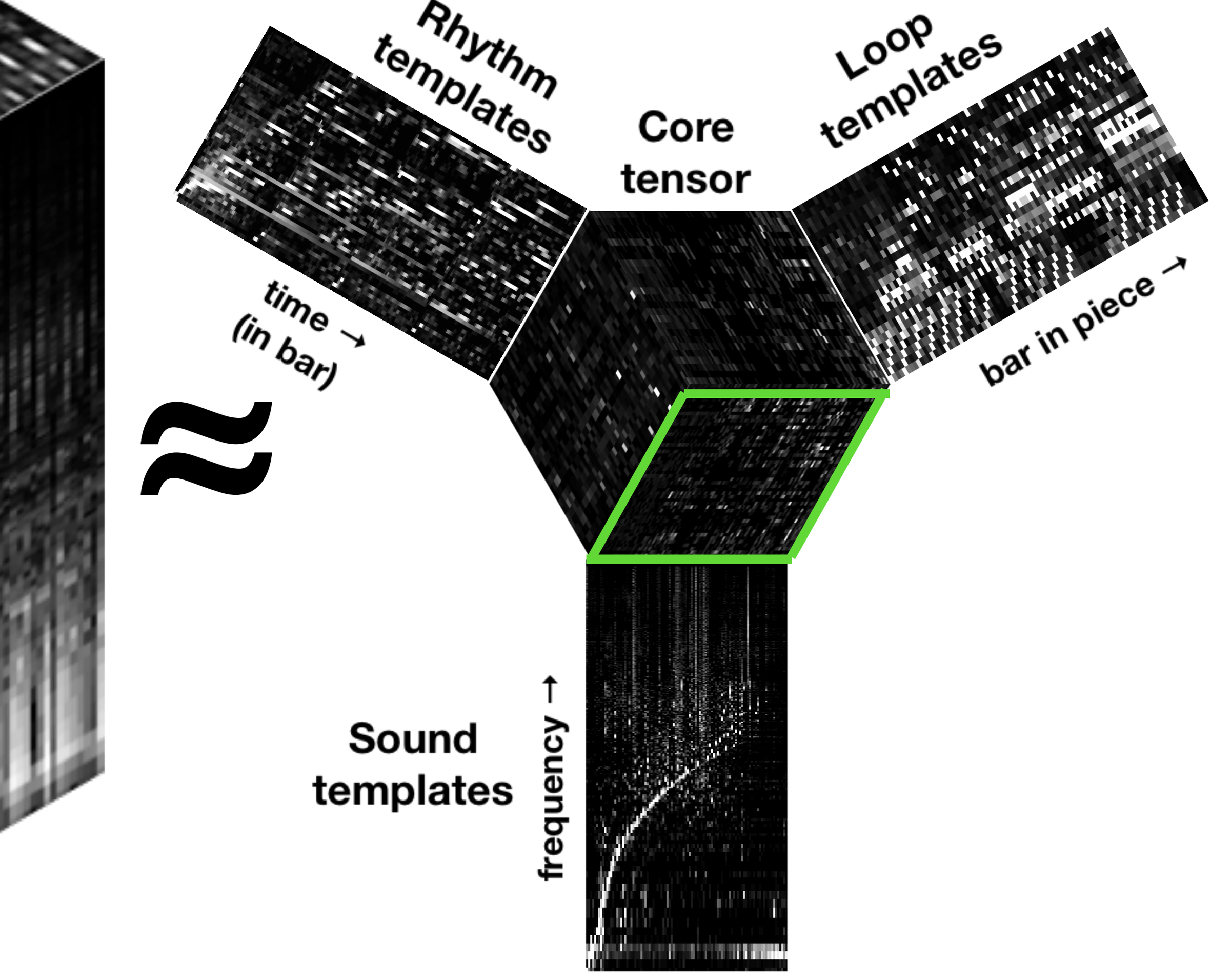
Core tensor

Loop templates

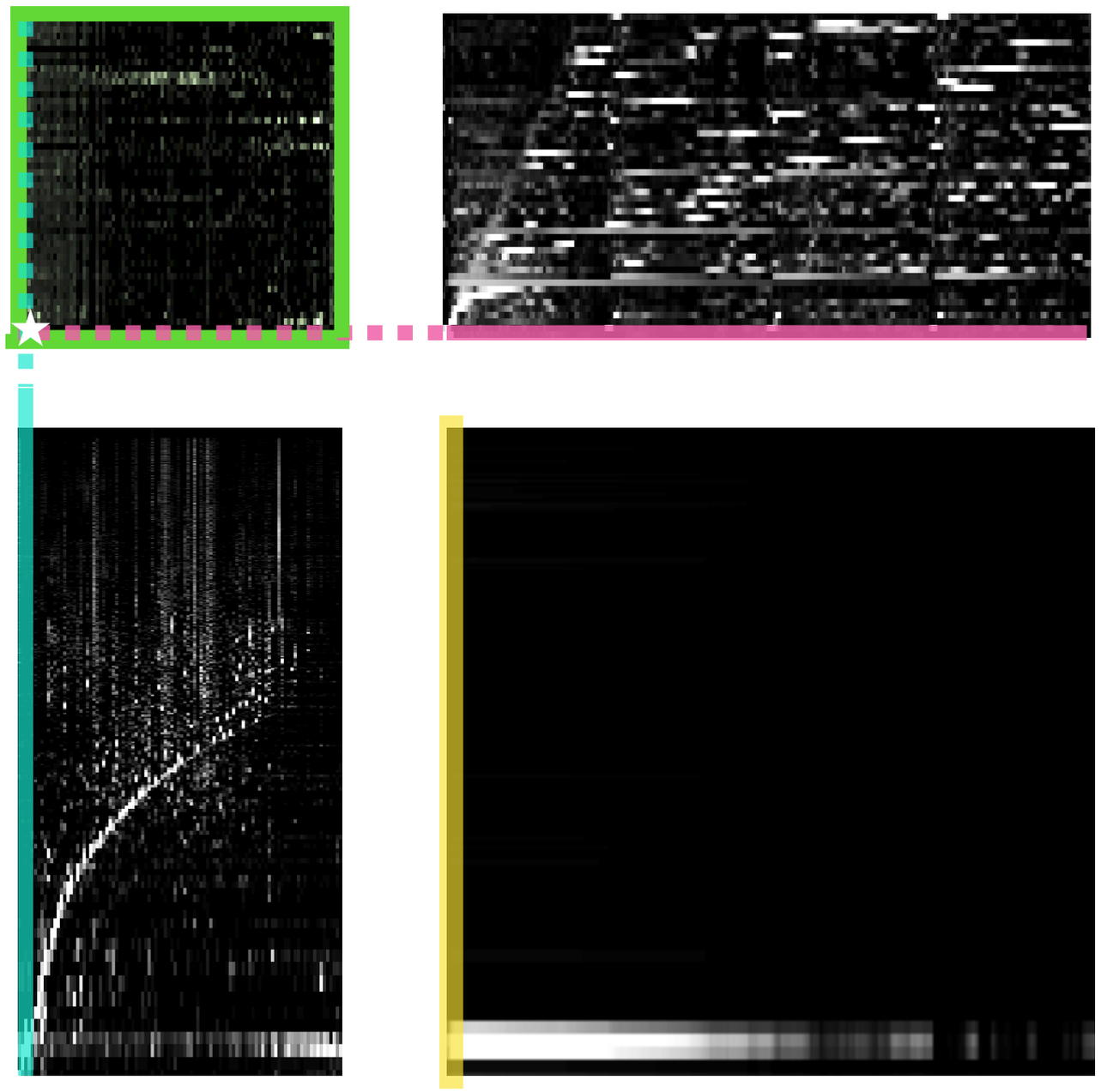


bar in piece →

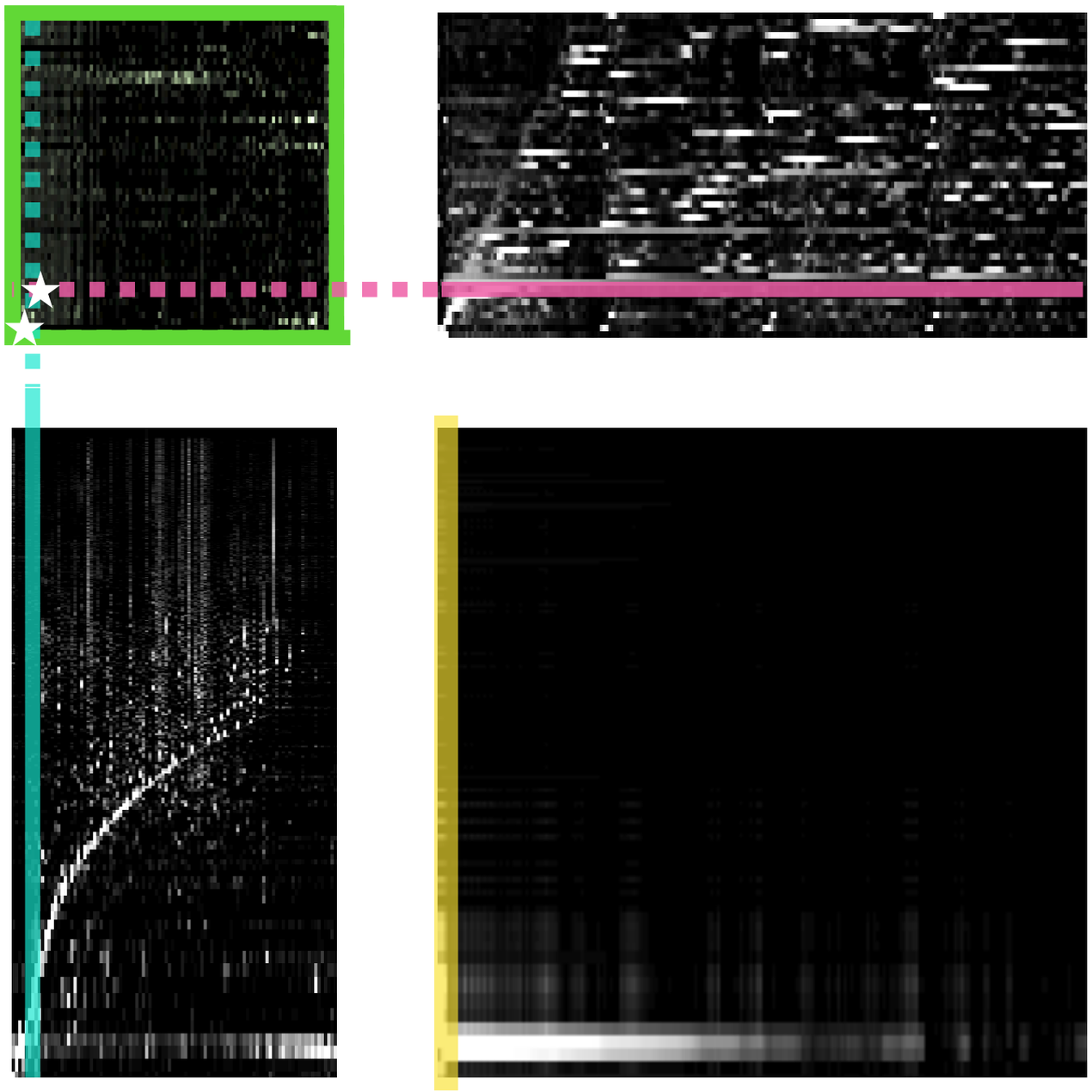

Rhythm templates
Core tensor
Loop templates
time → (in bar)
bar in piece →
Sound templates
frequency →


Rhythm templates
Core tensor
Loop templates
time → (in bar)
bar in piece →
Sound templates
frequency →

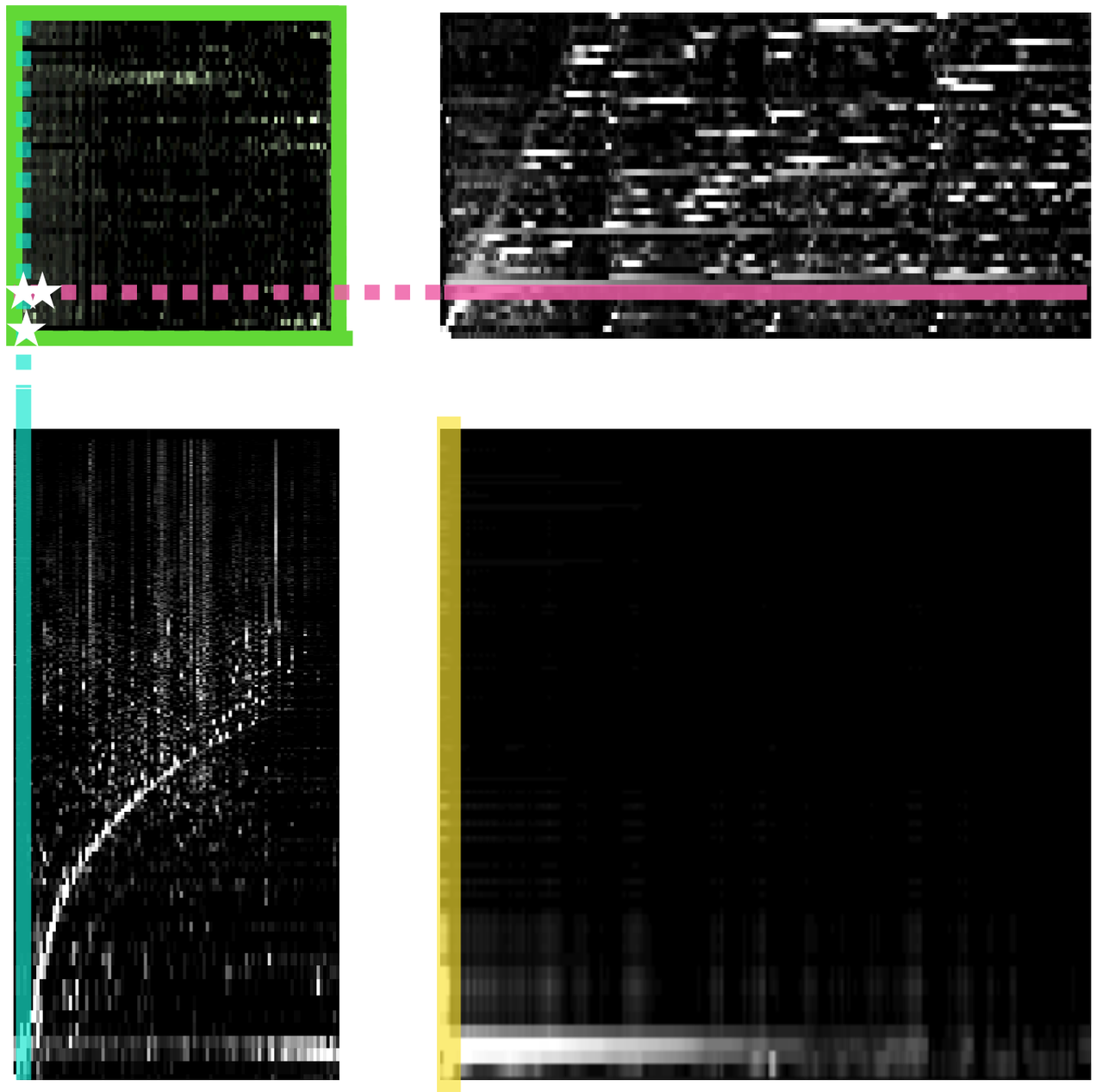
Loop 1 based on 1 component



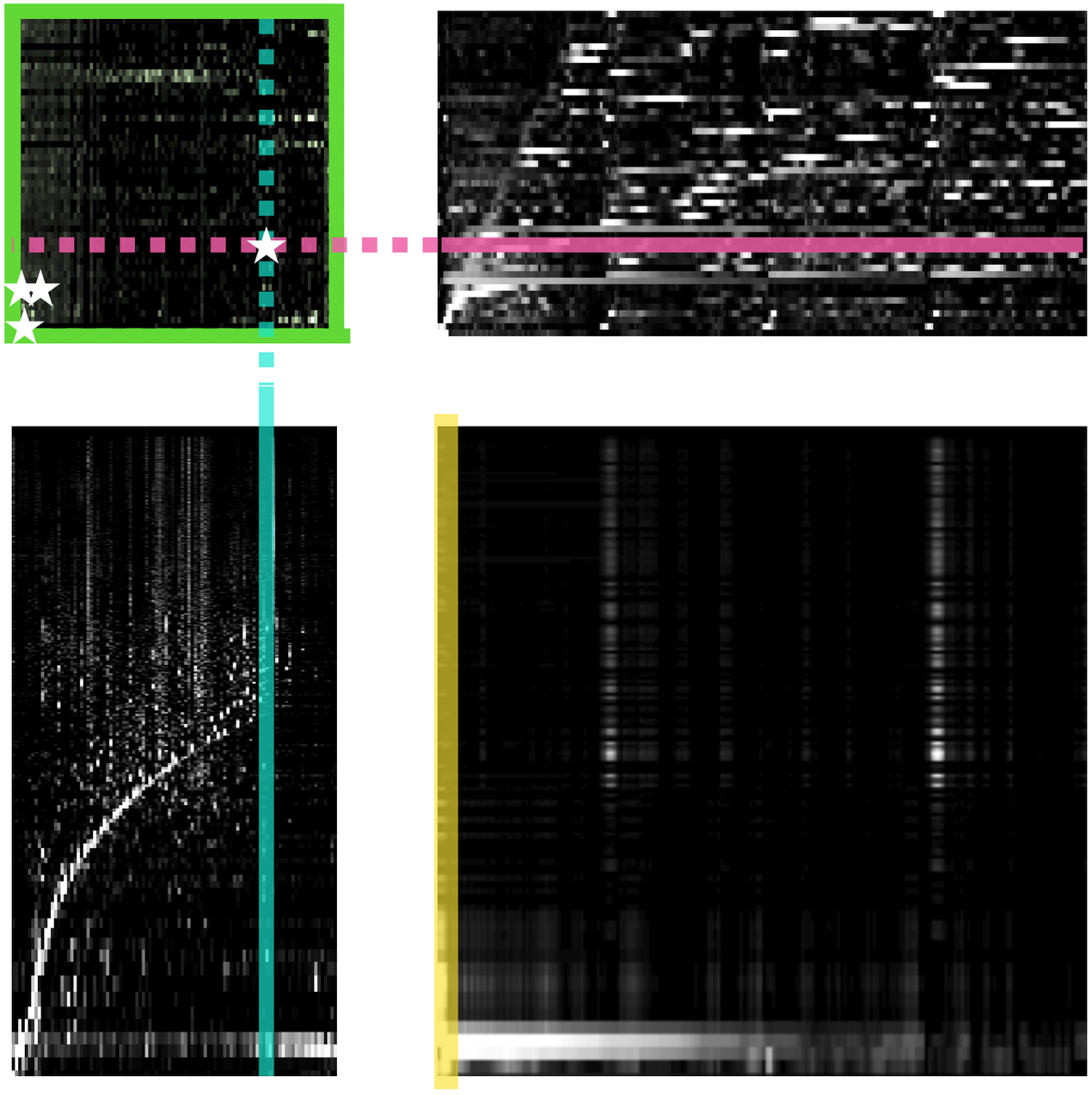
Loop 1 based on 2 components



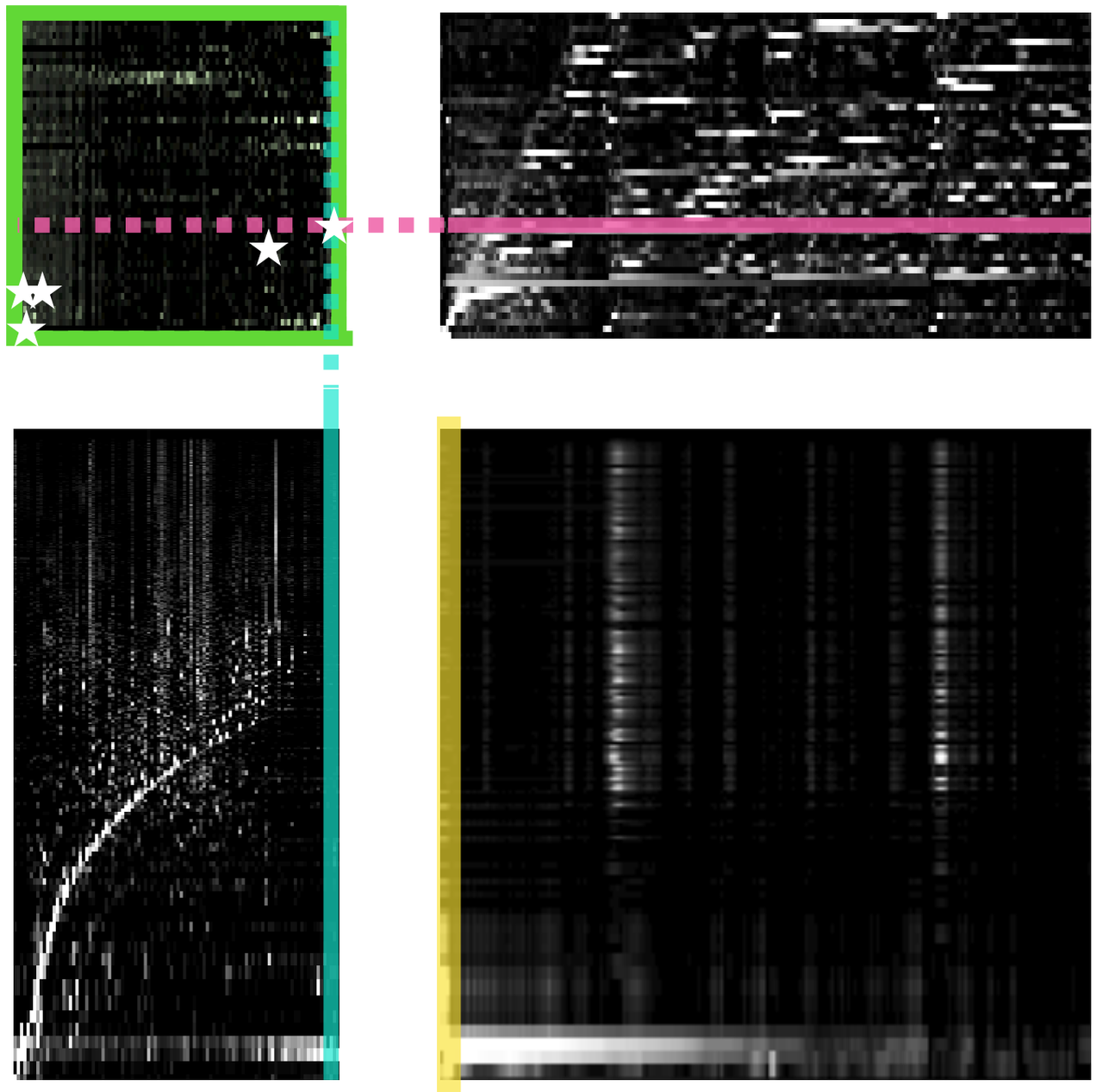
Loop 1 based on 3 components



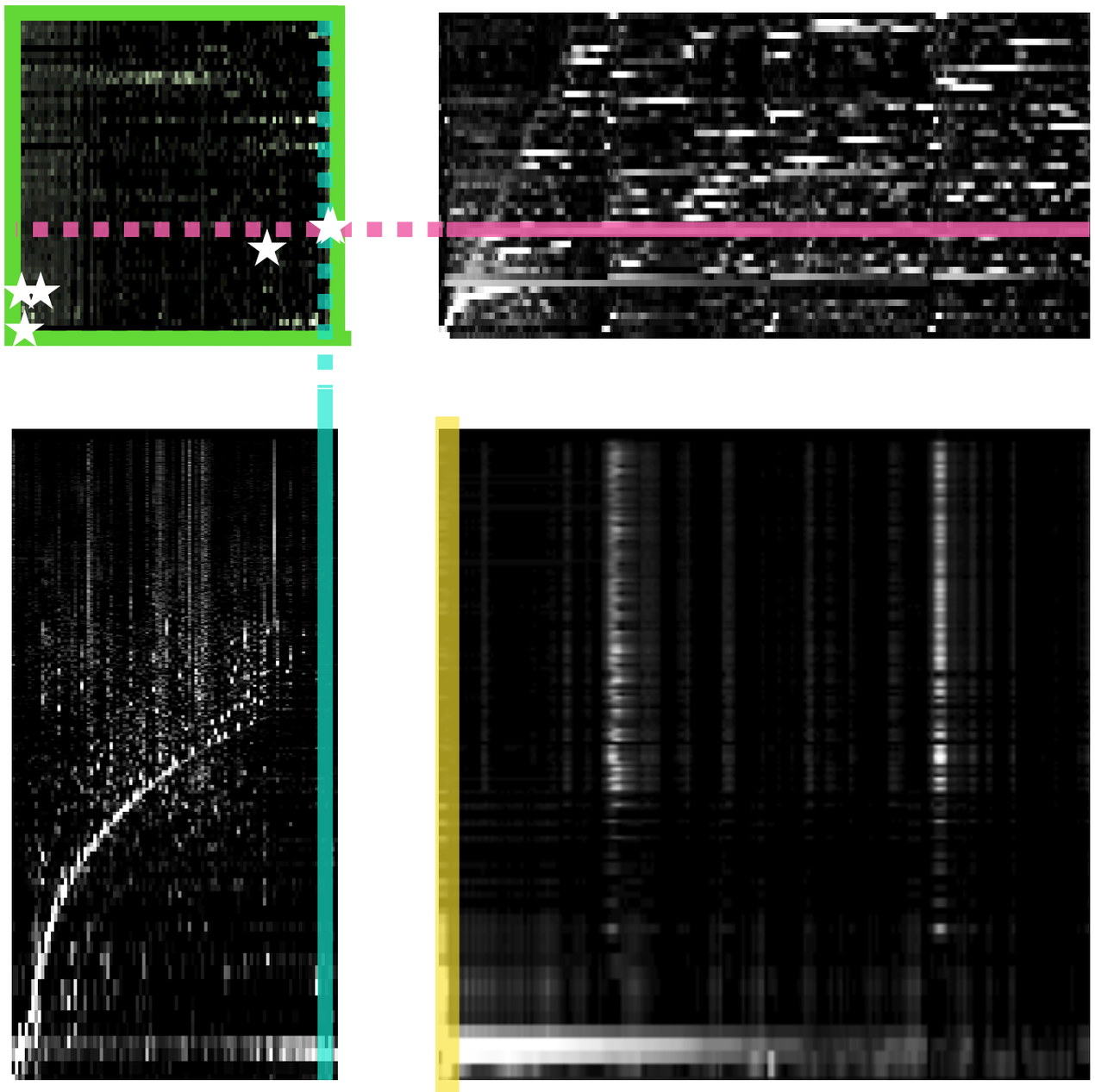
Loop 1 based on 4 components



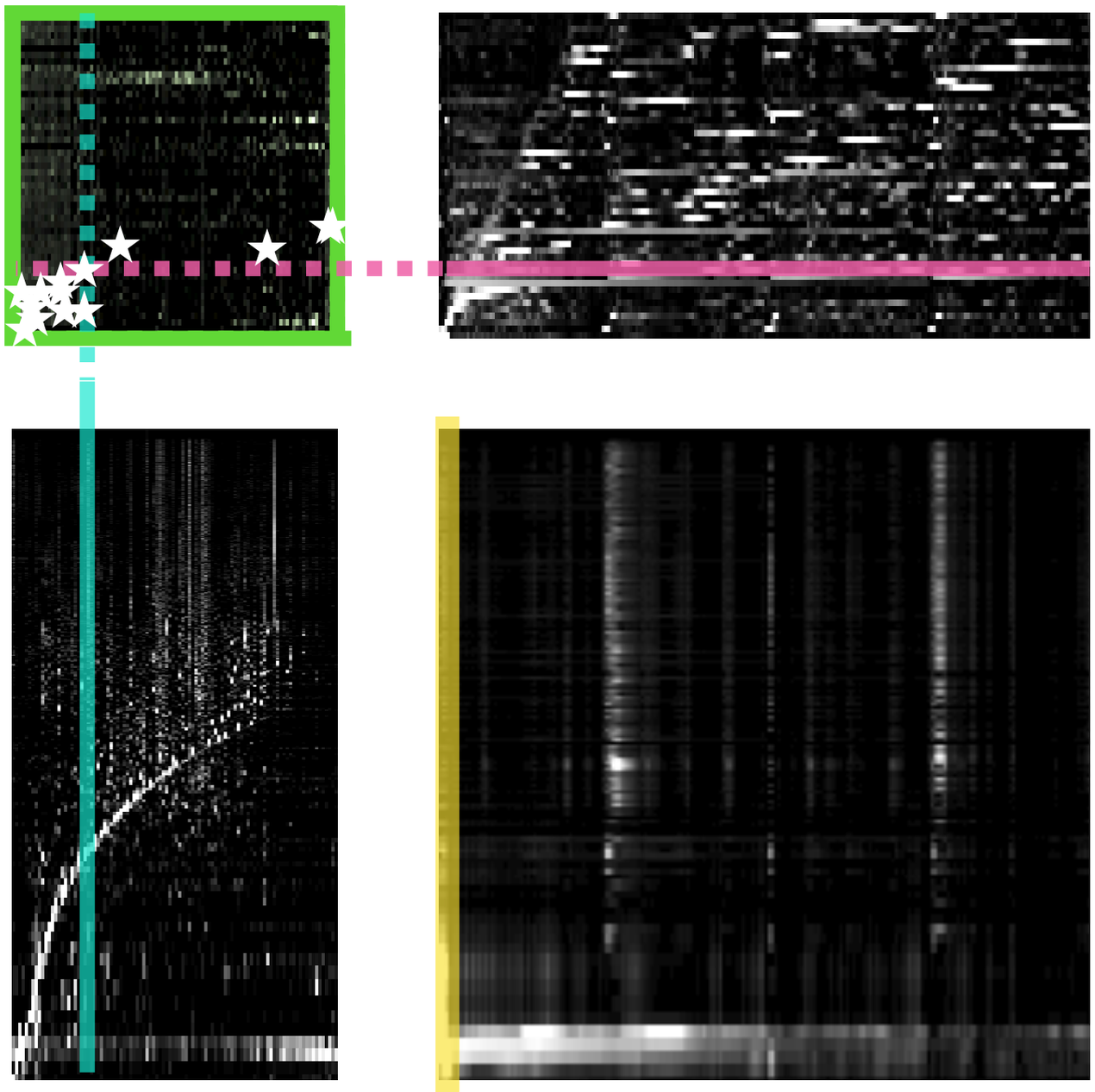
Loop 1 based on 5 components



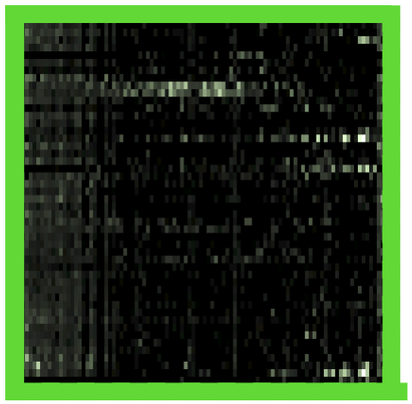
Loop 1 based on 6 components

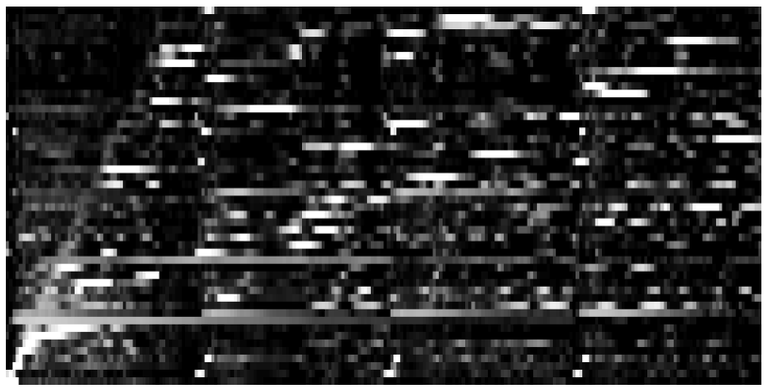


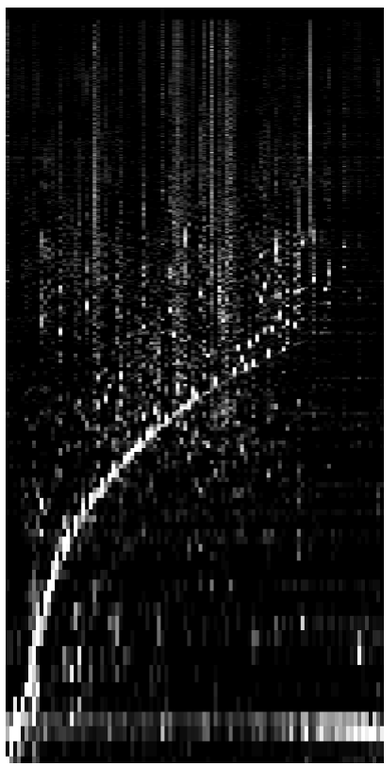
Loop 1 based on 15 components

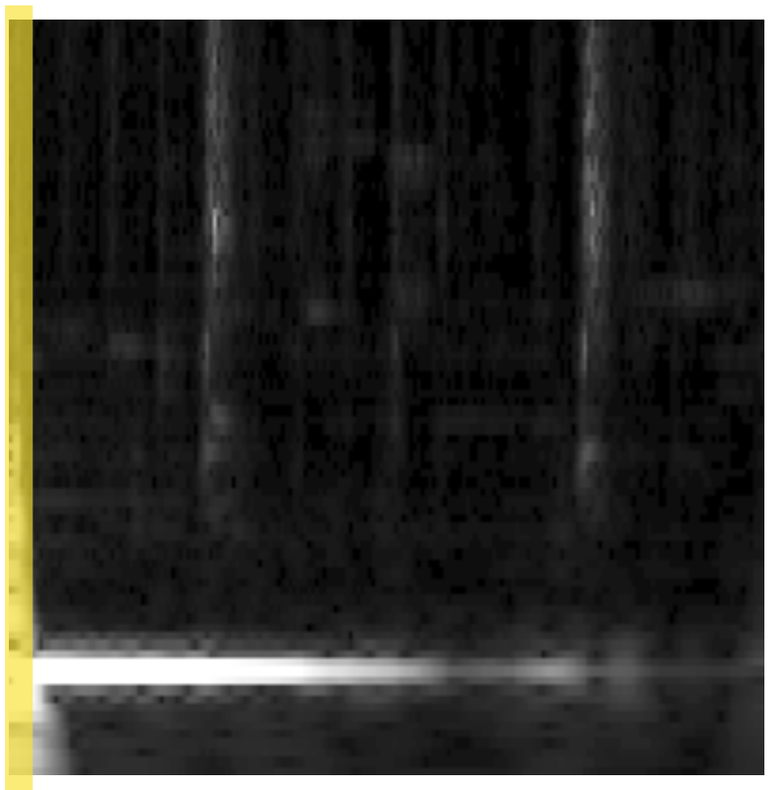


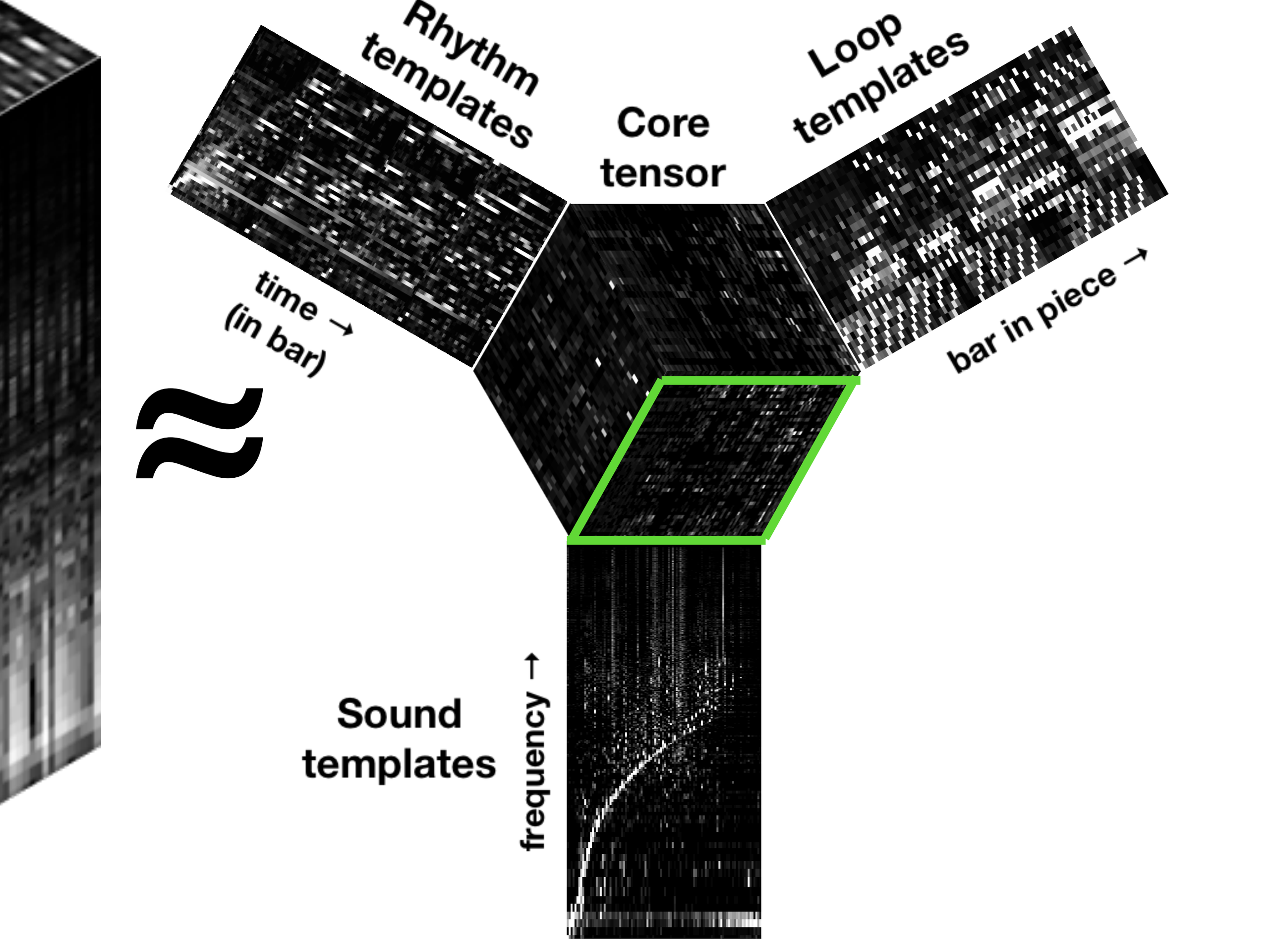
Loop 1 based on all components

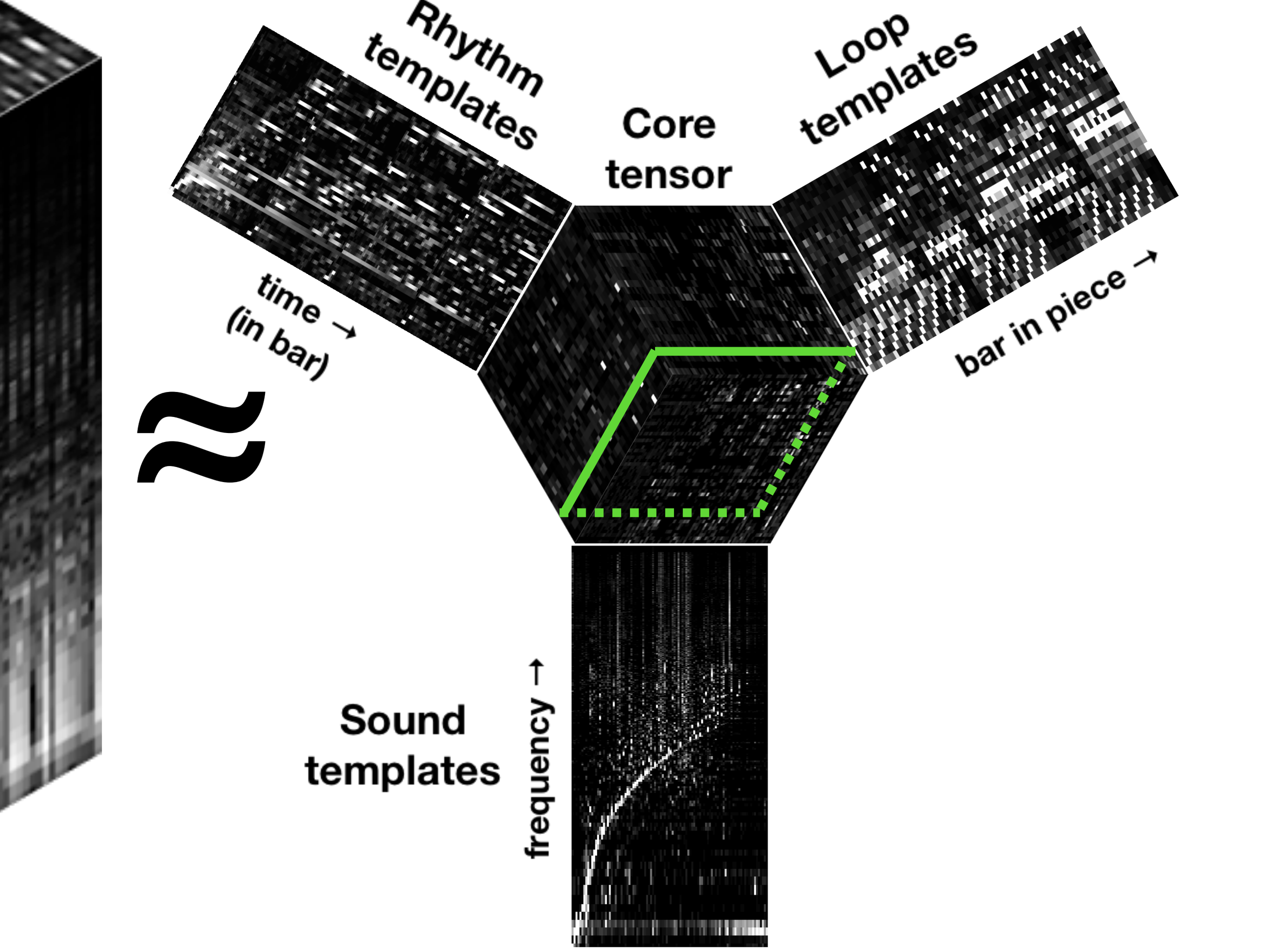




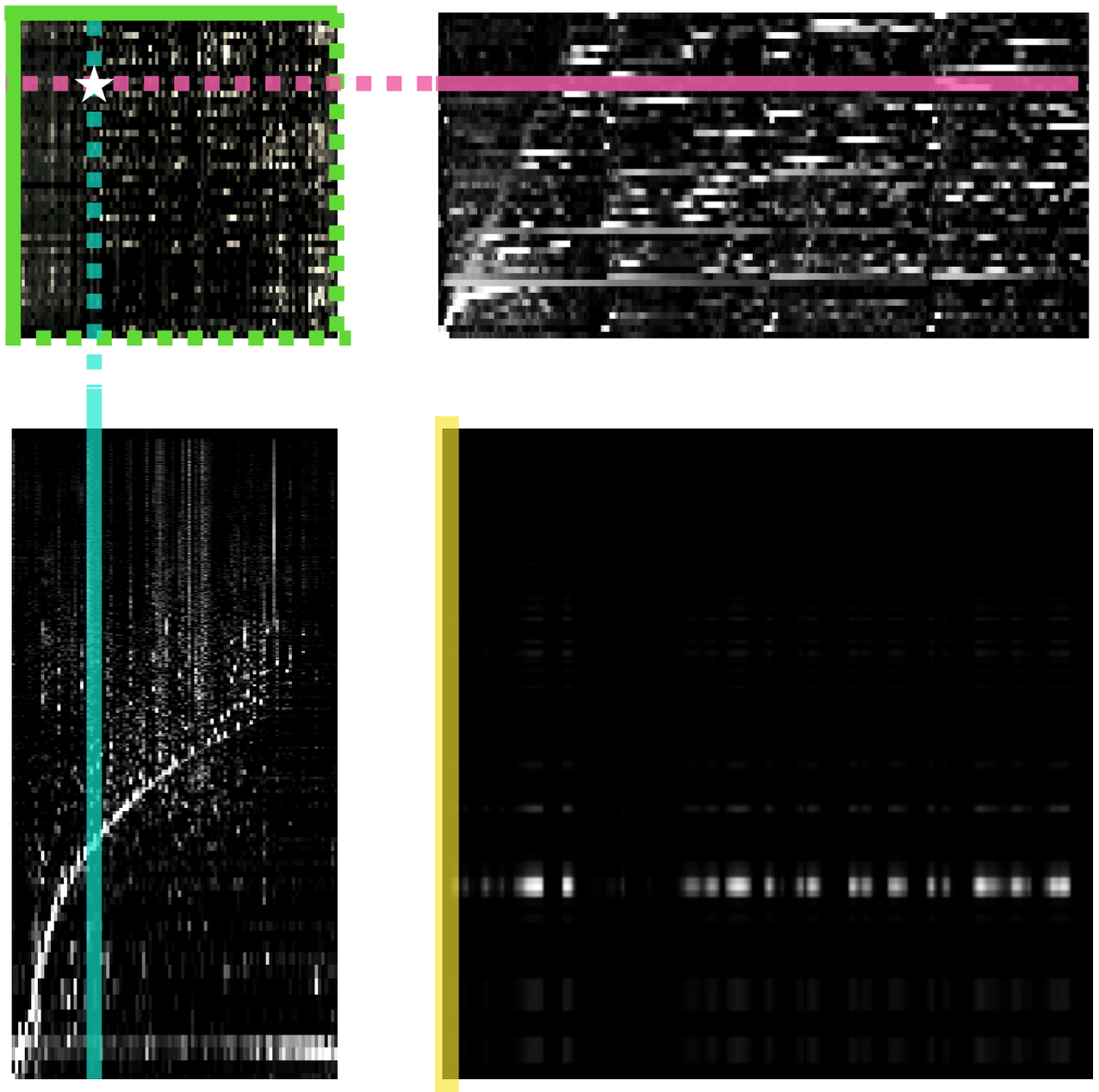




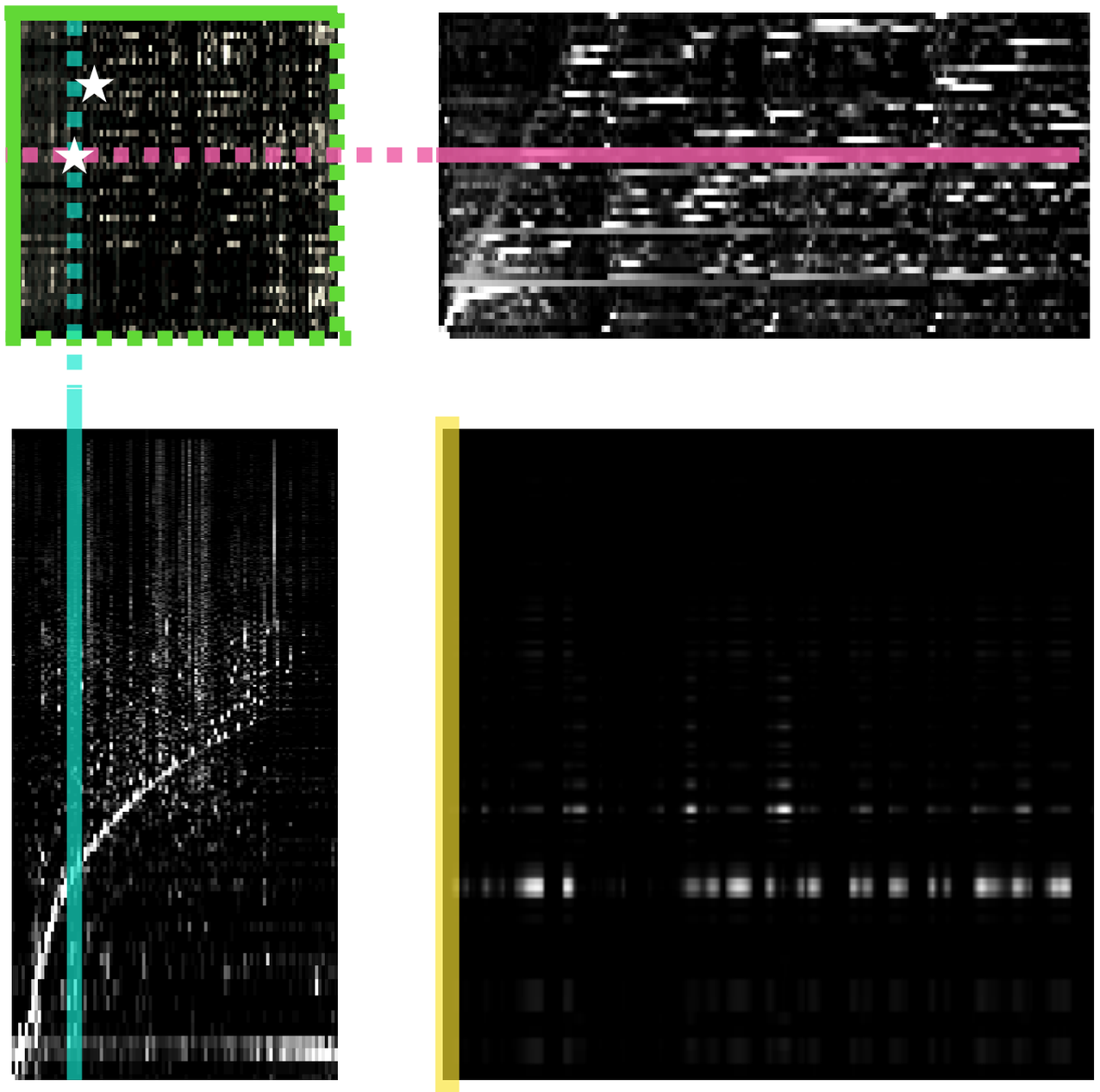

Rhythm templates
Core tensor
Loop templates
time →
(in bar)
bar in piece →
Sound templates
frequency →


Rhythm templates
Core tensor
Loop templates
time →
(in bar)
bar in piece →
Sound templates
frequency →

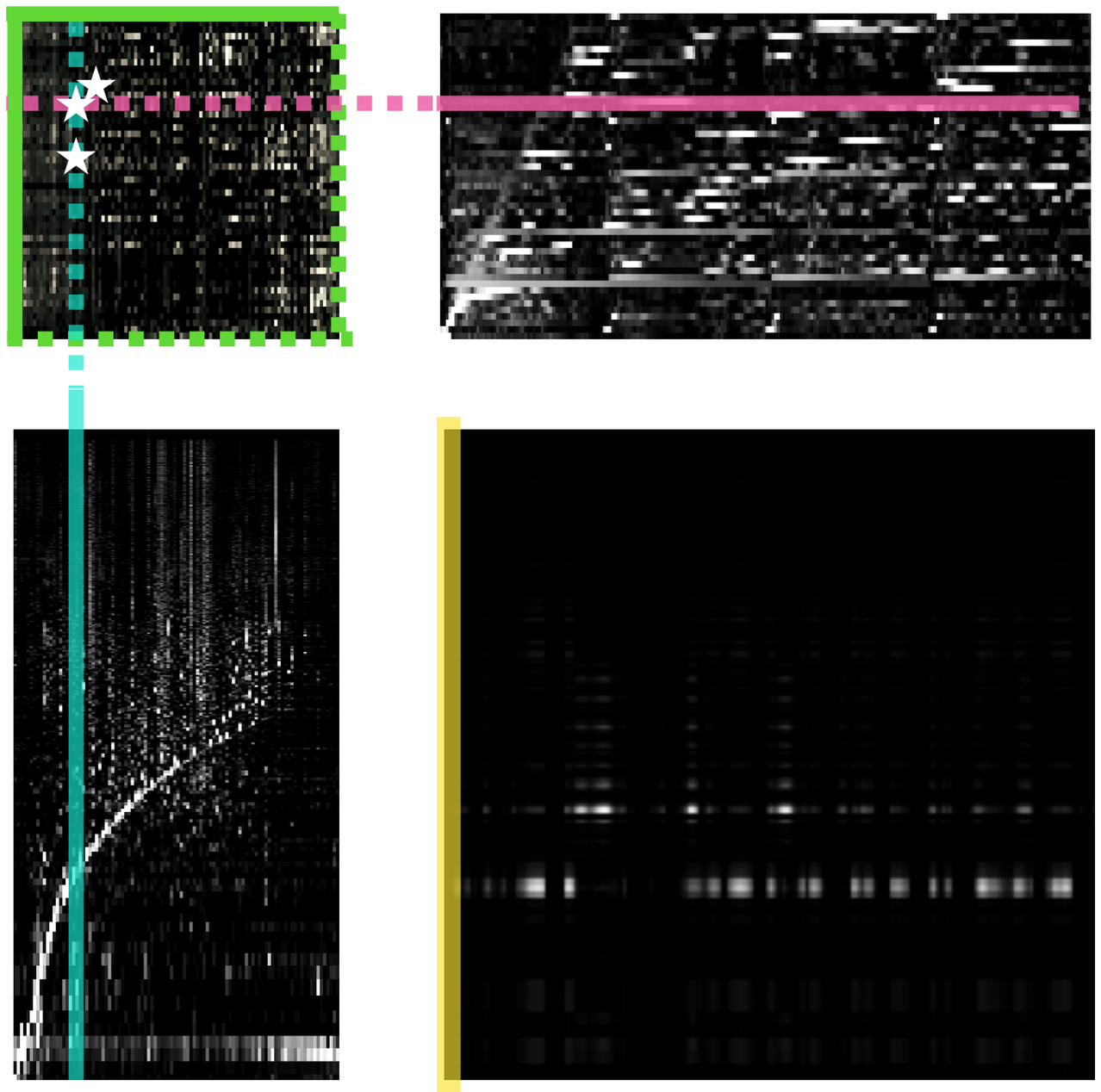
Loop #5 based on 1 component



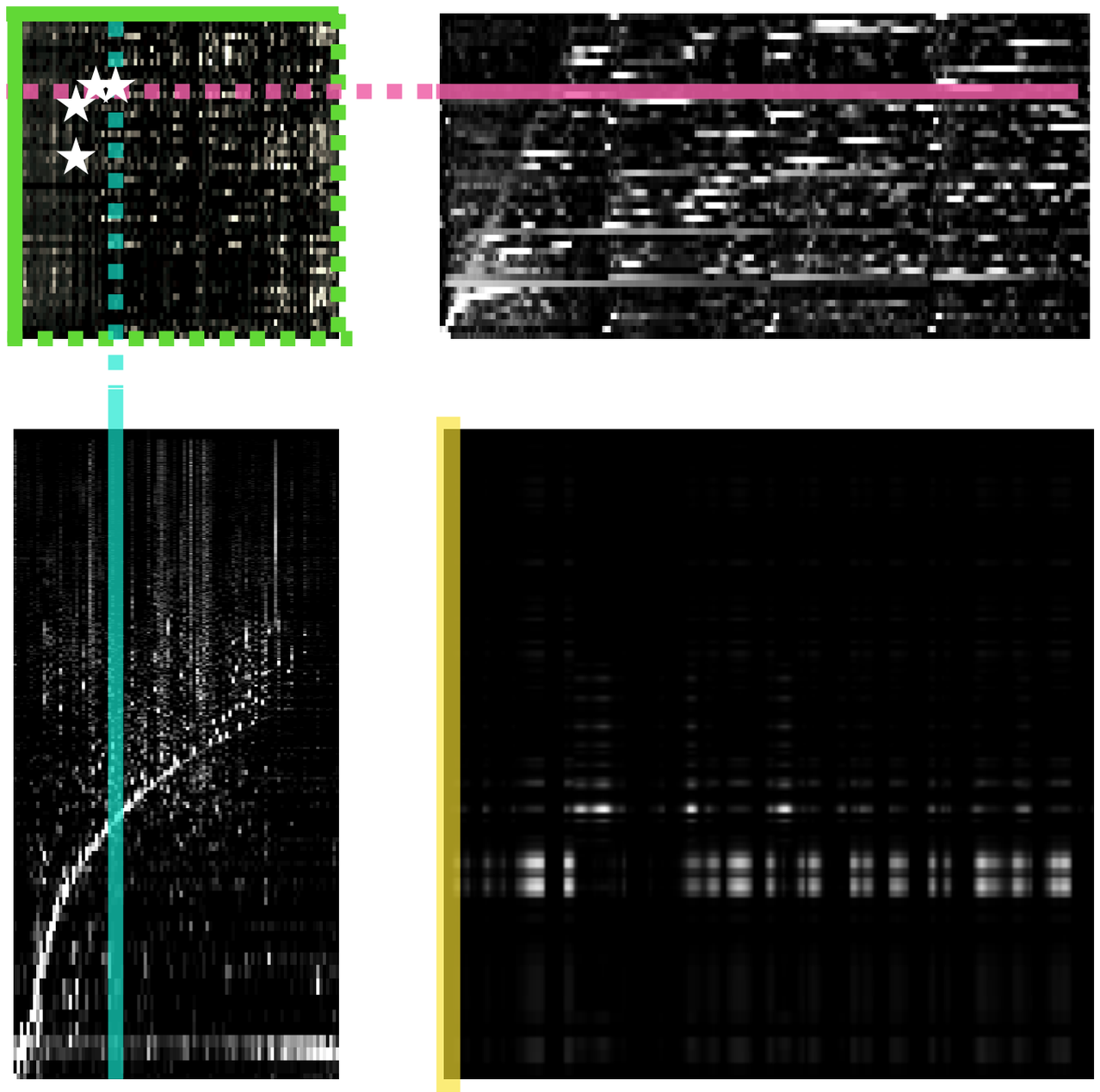
Loop #5 based on 2 components



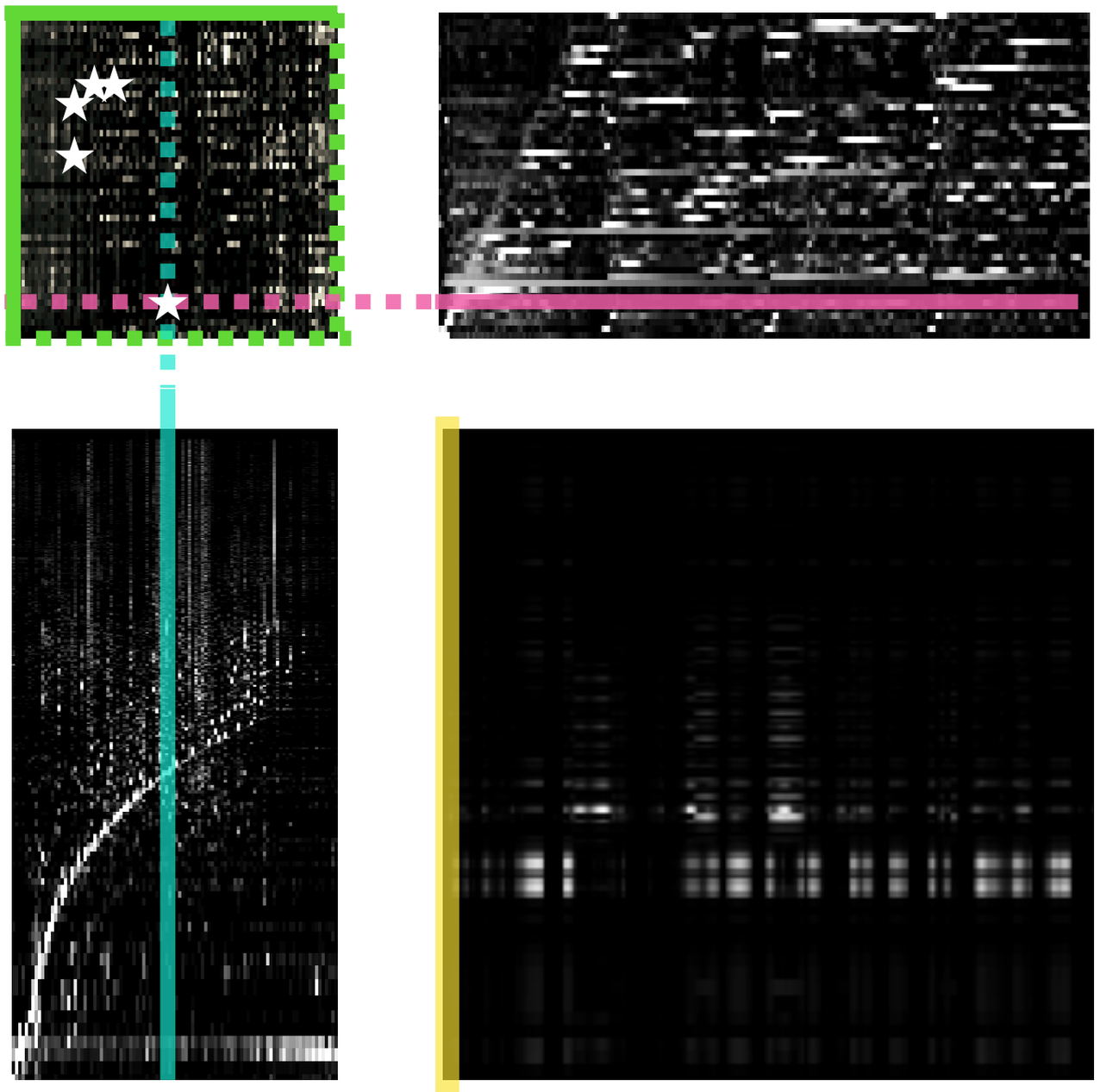
Loop #5 based on 3 components



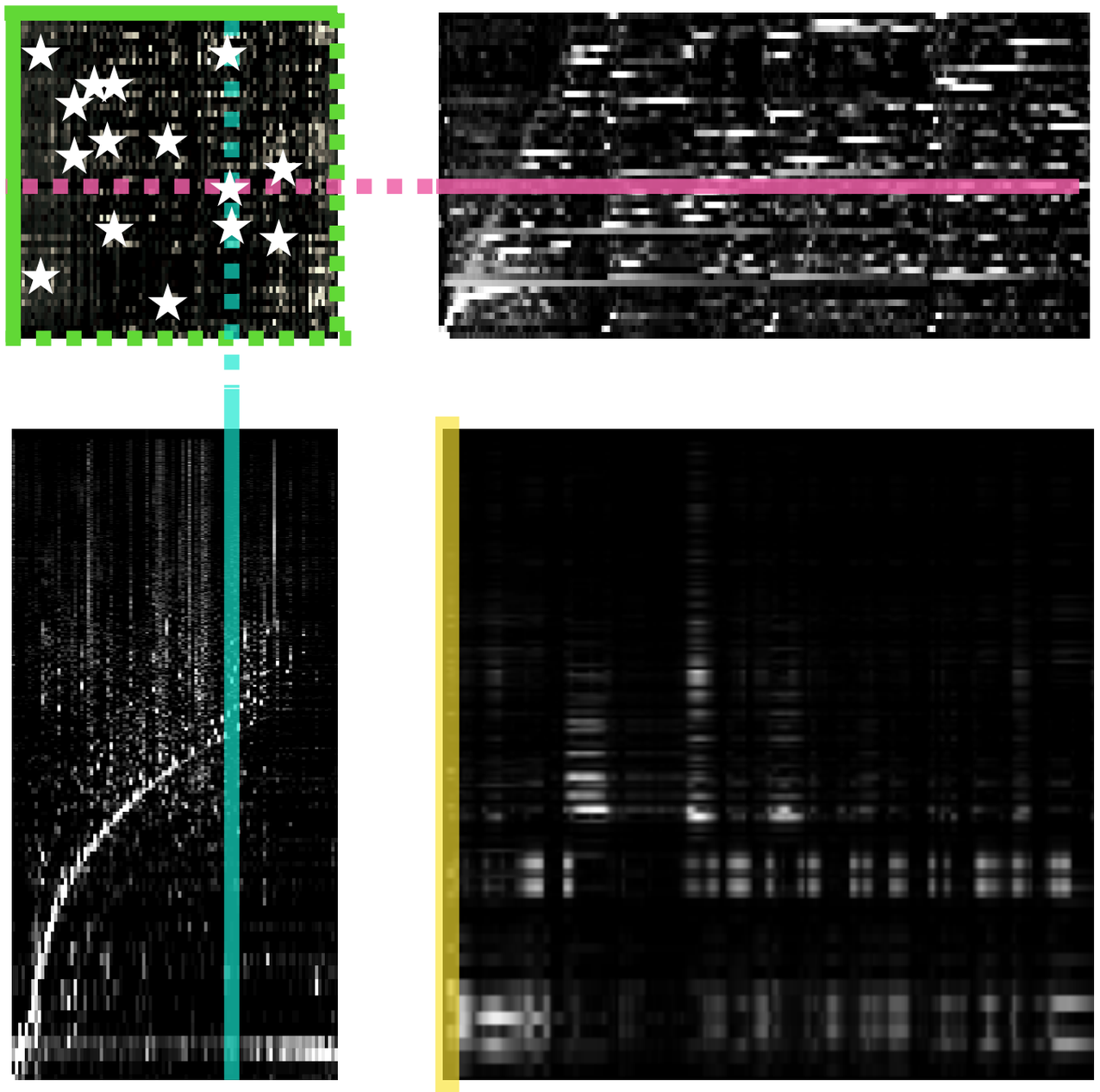
Loop #5 based on 4 components



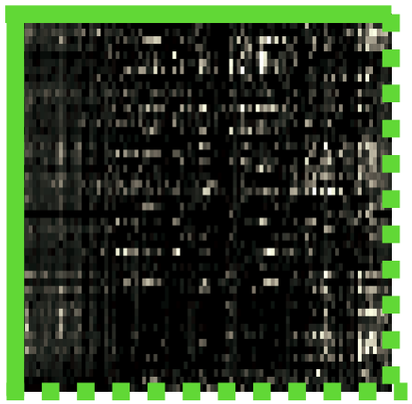
Loop #5 based on 5 components

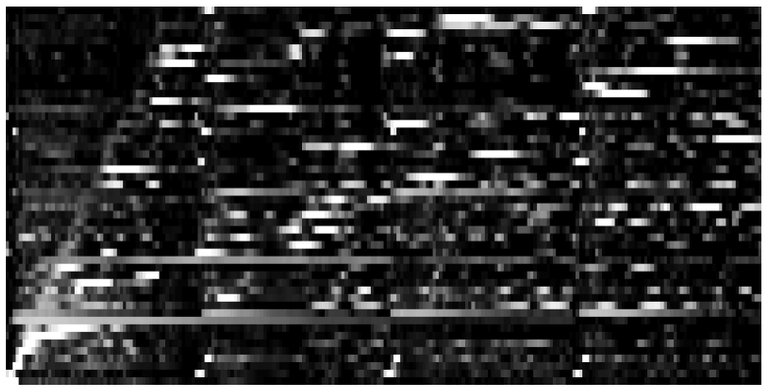


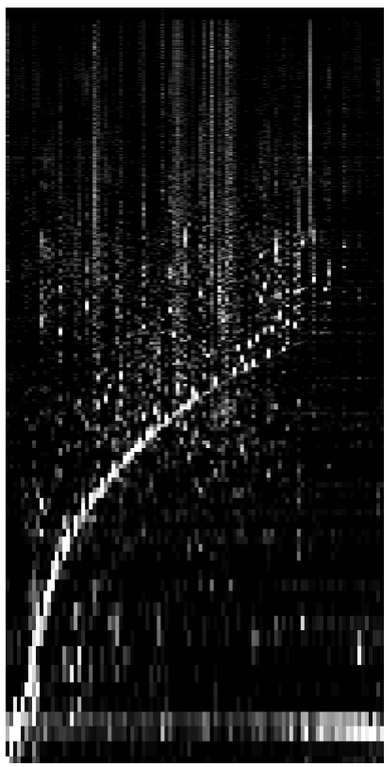
Loop #5 based on 15 components

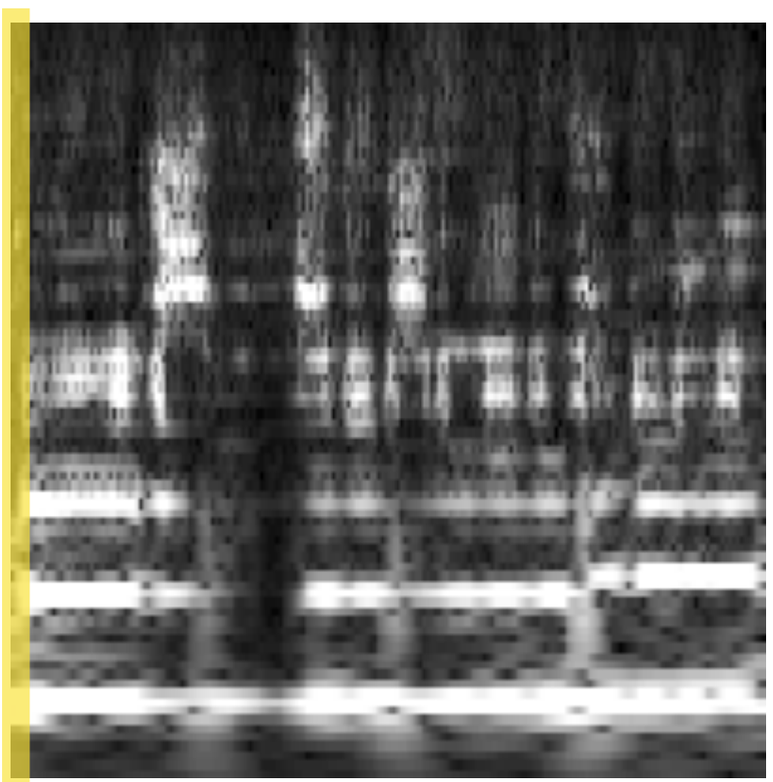


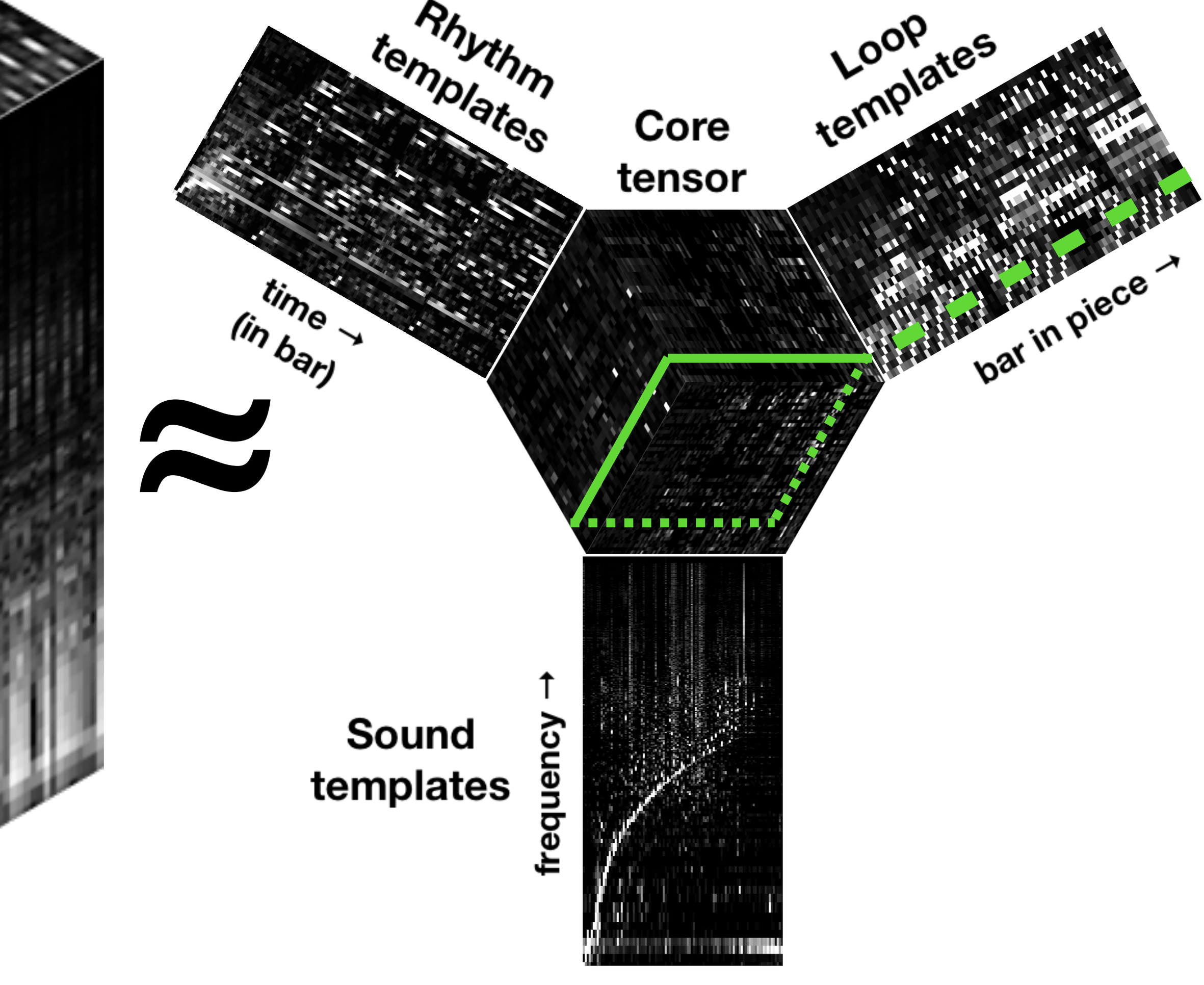
Loop #5 based on all components



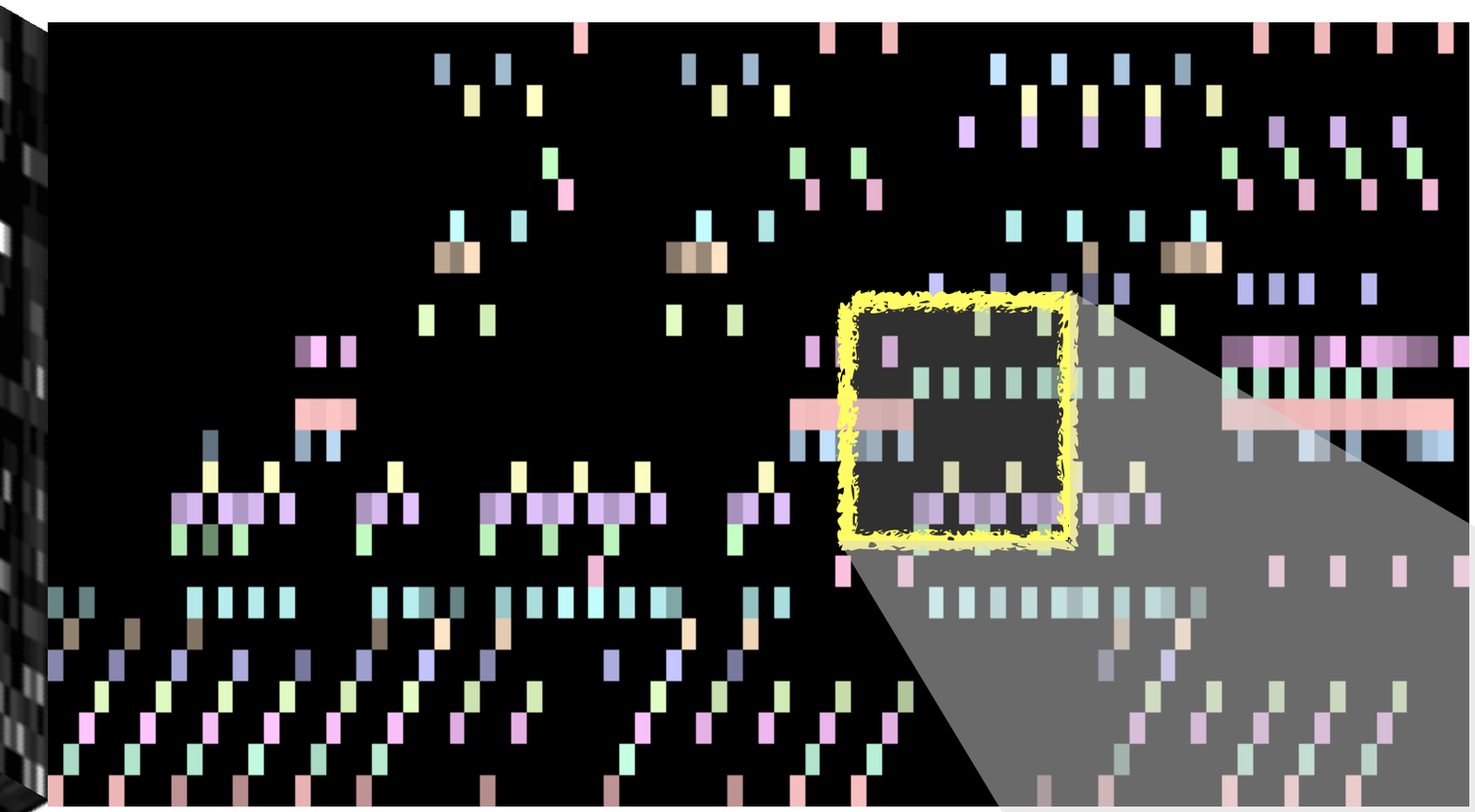




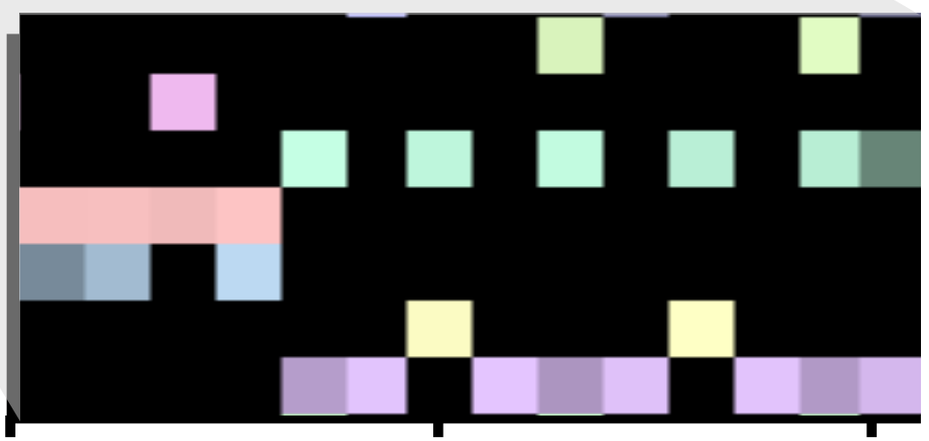



Rhythm templates
Core tensor
Loop templates
time →
(in bar)
bar in piece →
Sound templates
frequency →

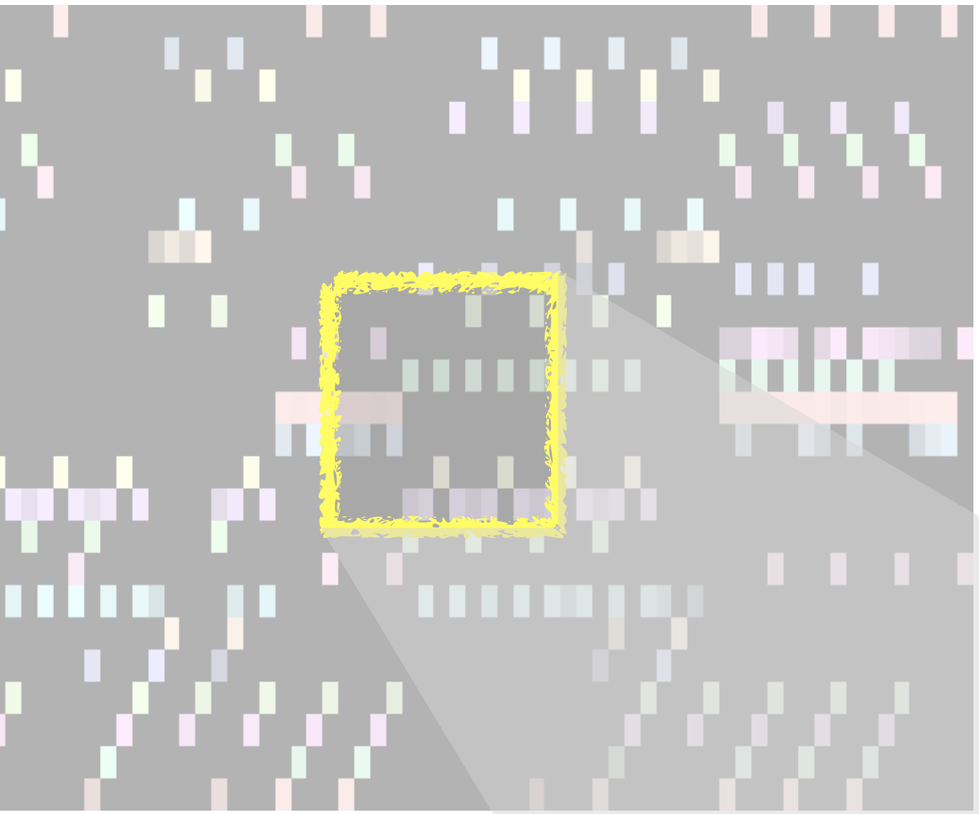
Loop templates

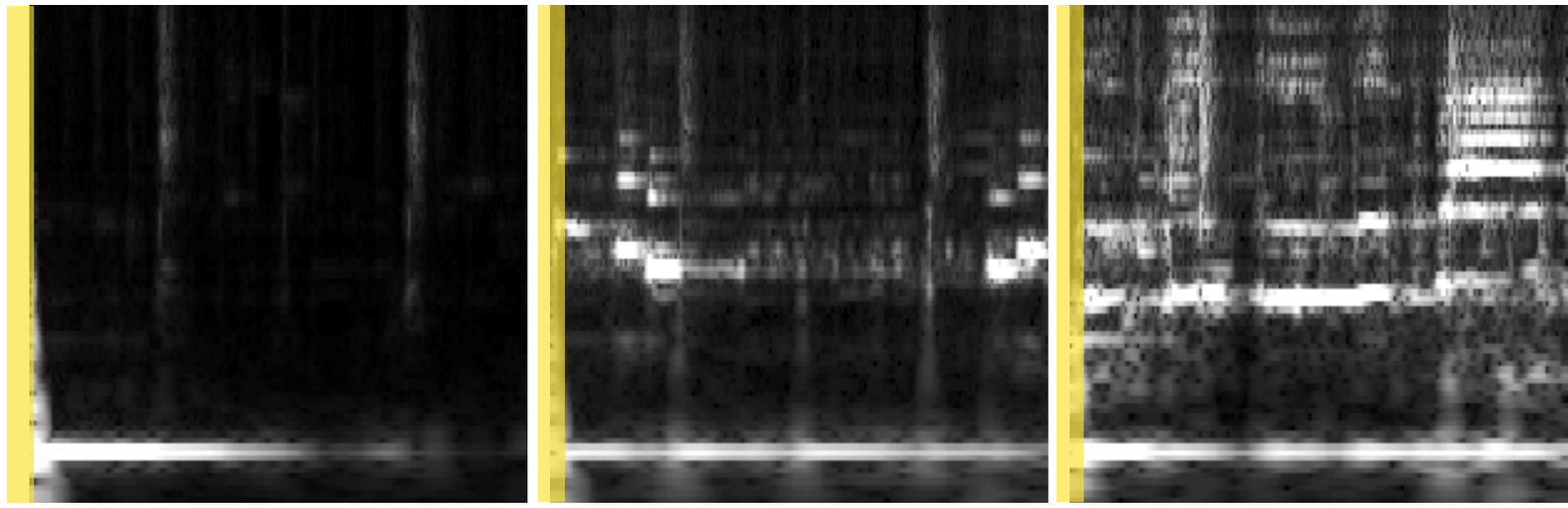


bar in piece →

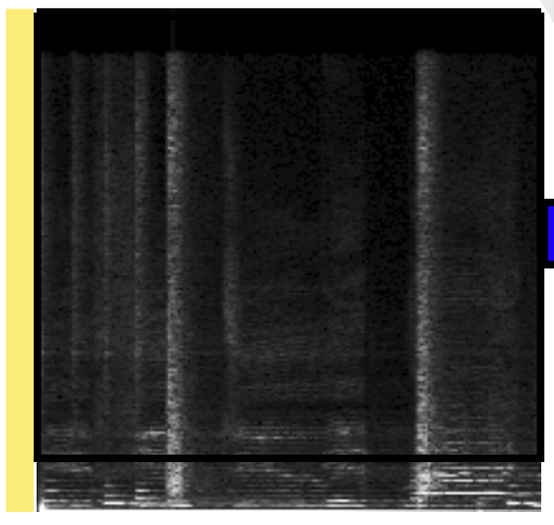


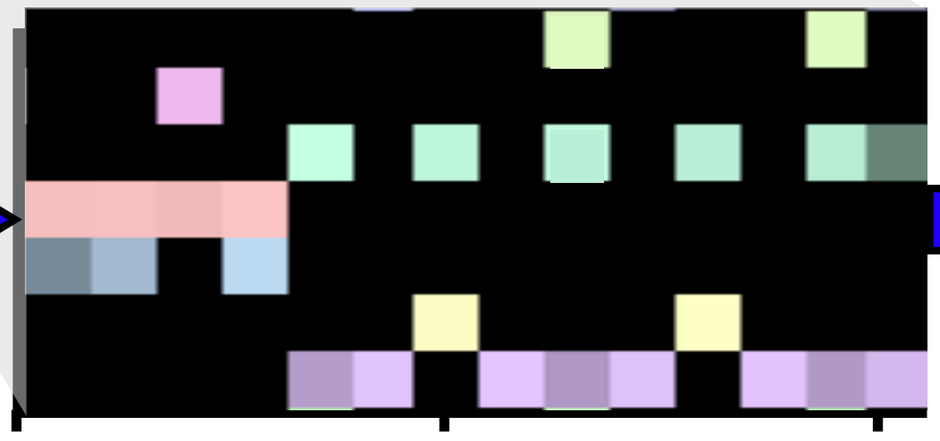
Bar 52 Bar 58 Bar 65





Bar #60



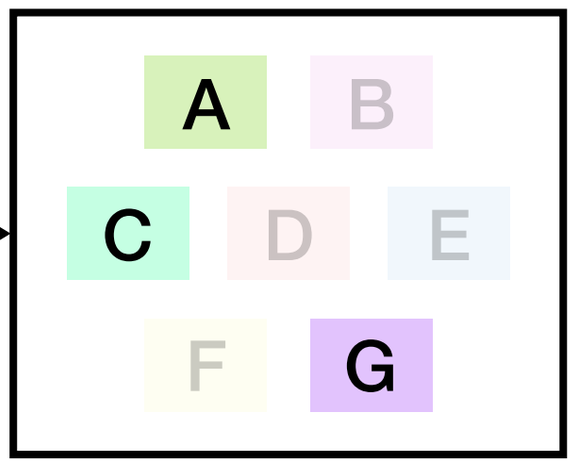


Bar 52

Bar 58

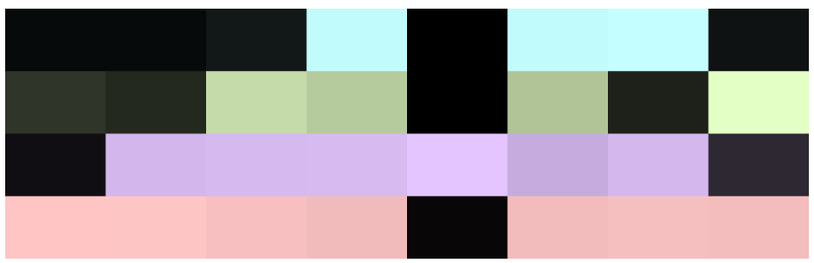
Bar 65

Ingredients for bar #60:



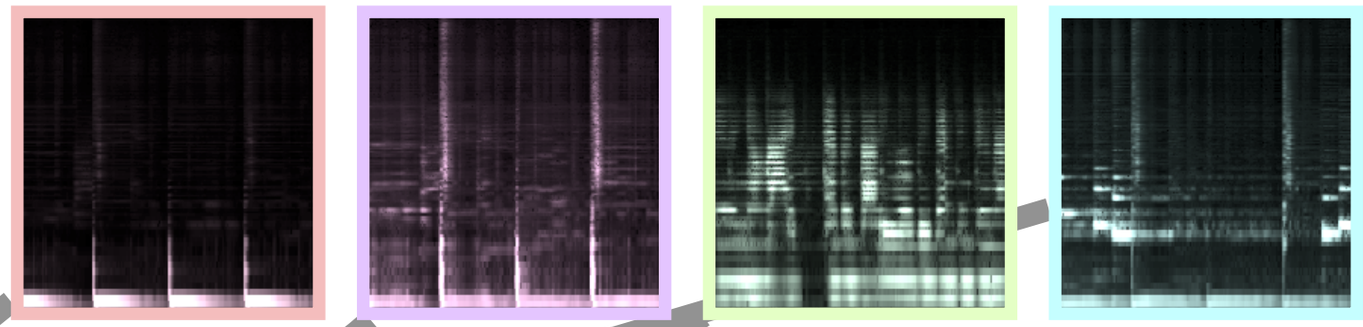
Choosing the bar to reconstruct

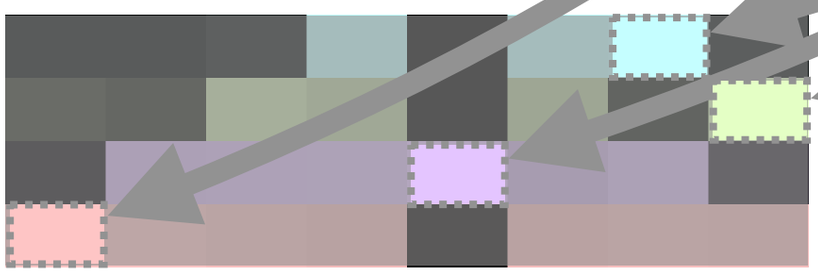
Estimated map of loops:



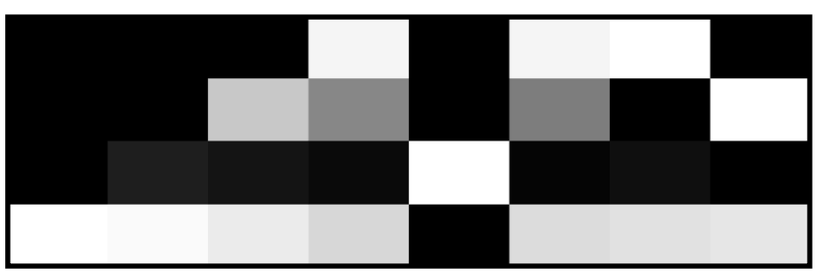
Which bar ↑ to reconstruct?

Reconstructed magnitude spectra:





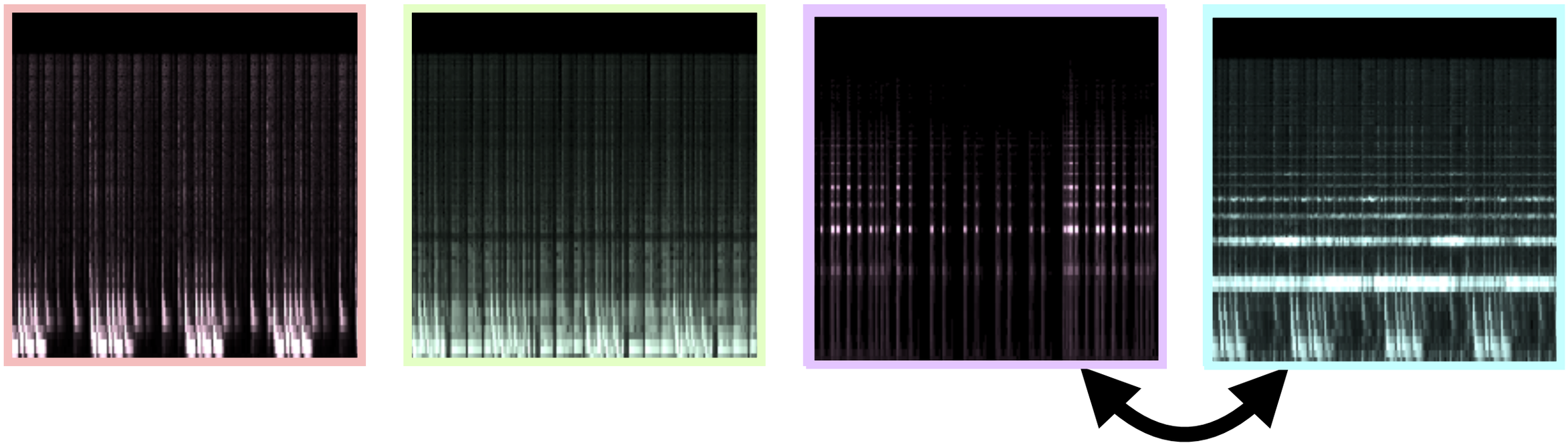
← Answer: choose a bar that maximizes activation loudness, minimizes crosstalk



← How to do this?
= [activation value × softmask share]

Doing even better: sparse factorization

Problem: sometimes several loops sound similar.

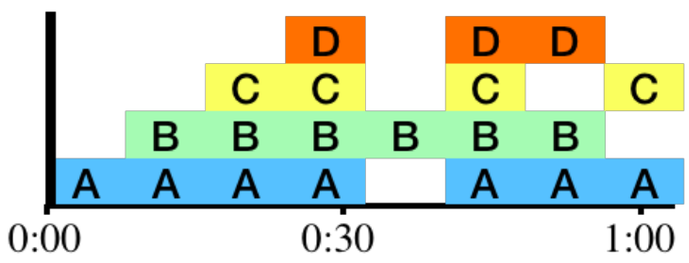


These are similar (now they're less similar!)

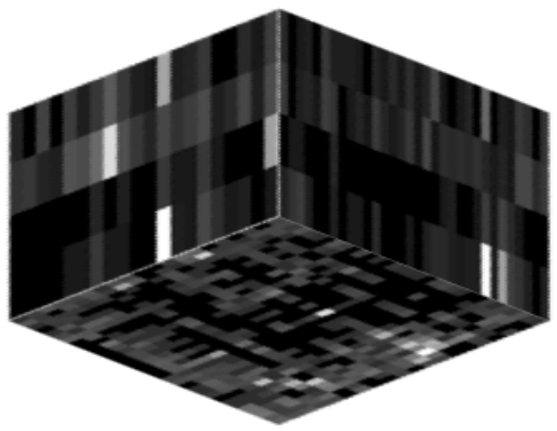
Doing even better: sparse factorization

Standard approach: acceptable match to ground truth, but some loops redundant.

True layout:



core tensor



Loop templates (4 estimated)

Doing even better: sparse factorization

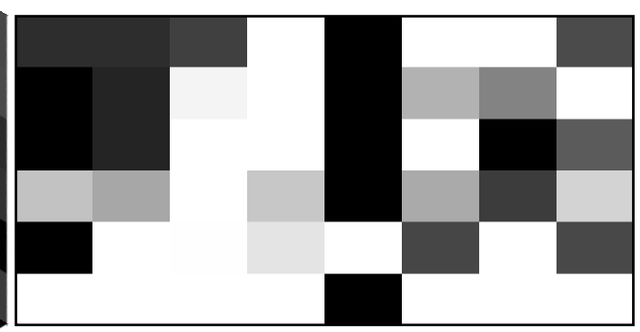
Standard approach: acceptable match to ground truth, but some loops redundant.

Additional sparse-NMF step to condense core tensor:

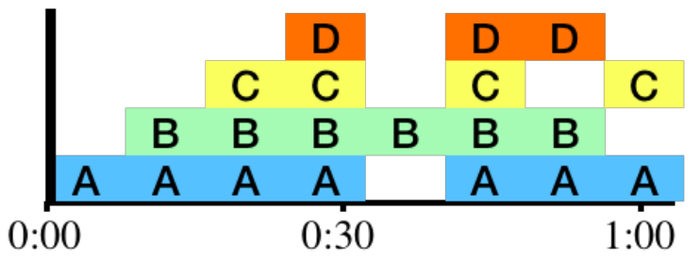
→ fewer redundant loops, better match to ground truth

core tensor

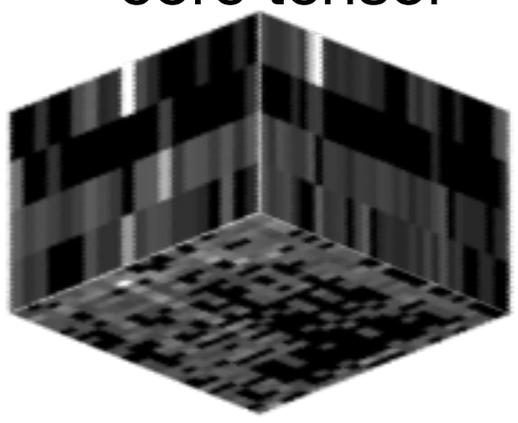
Loop templates (6 estimated):



True layout:



core tensor



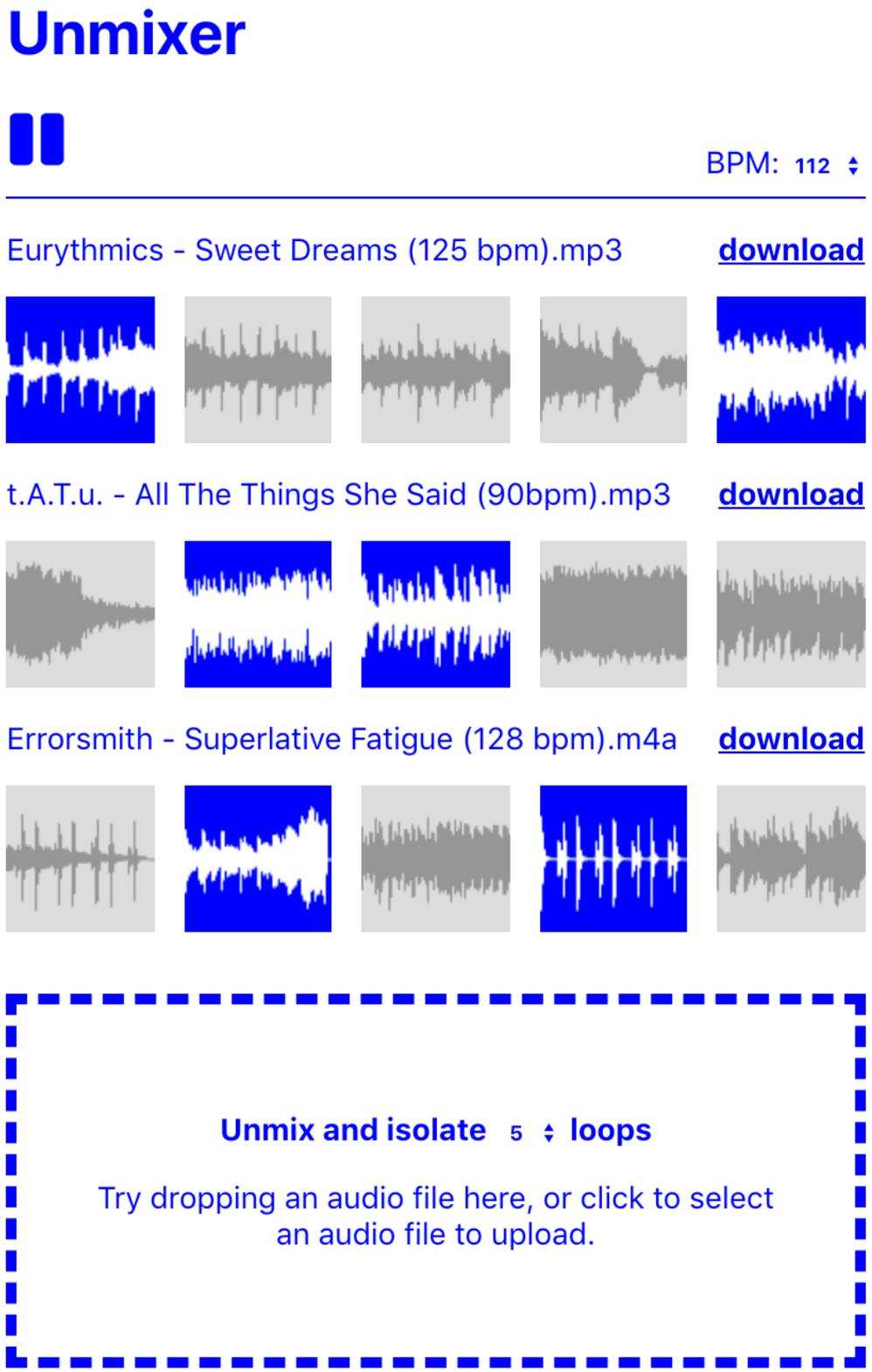
Loop templates (6 estimated, condensed to 4):

Main points

- Tucker decomposition (NTD) is a natural de-composition model for music
  - Spectral cube ≈ sounds x rhythms x loop activations
    - NTD groups any sounds that always repeat together
- **Unmixer** use NTD to extract loops from uploaded songs
  - Performs important groundwork for remix artists
  - Allows rapid exploration and experimentation of remix and mashup possibilities

https://unmixer.ongaaccel.jp

Thank you!



This work was supported in part by JST ACCEL Grant Number JPMJAC1602, Japan.