When Machine Learning Takes over Audio Signal Processing

Paris Smaragdis
paris@illinois.edu
paris.cs.illinois.edu

University of Illinois at Urbana Champaign

What is this talk really about?

- Machine Learning vs. Signal Processing?
  - Not quite, they are the same thing really
- It's about breaking away from textbook adherence
  - Borrowing ideas from other fields, and incorporating them in SP
- The goal is to inspire you to look around more
  - The specific techniques here are irrelevant, the approach is

Three stories to tell

- Array processing from a different viewpoint
  - Powerful alternatives to beamforming / localization
- Non-negative audio models
  - Dictionary models for processing on mixtures of sounds
- (and the obligatory) Deep learning
  - Supervised methods for signal enhancement
  - Quantized networks for fast/cheap audio processing

Array methods

- Standard approaches
  - Beamforming (Delay & sum, MVDR, the GSC, etc.)
- Some problems
  - We need a lot of mics to get a lot of gain
  - We need precise calibration
- We are already pushing the limits of mic arrays
  - 300 mic arrays are amazing, but expensive!

A different approach

- 2-mic array inter-phase features
  - Phase difference between channels

Spectrogram of mic 1

Phase difference between each time/frequency bin

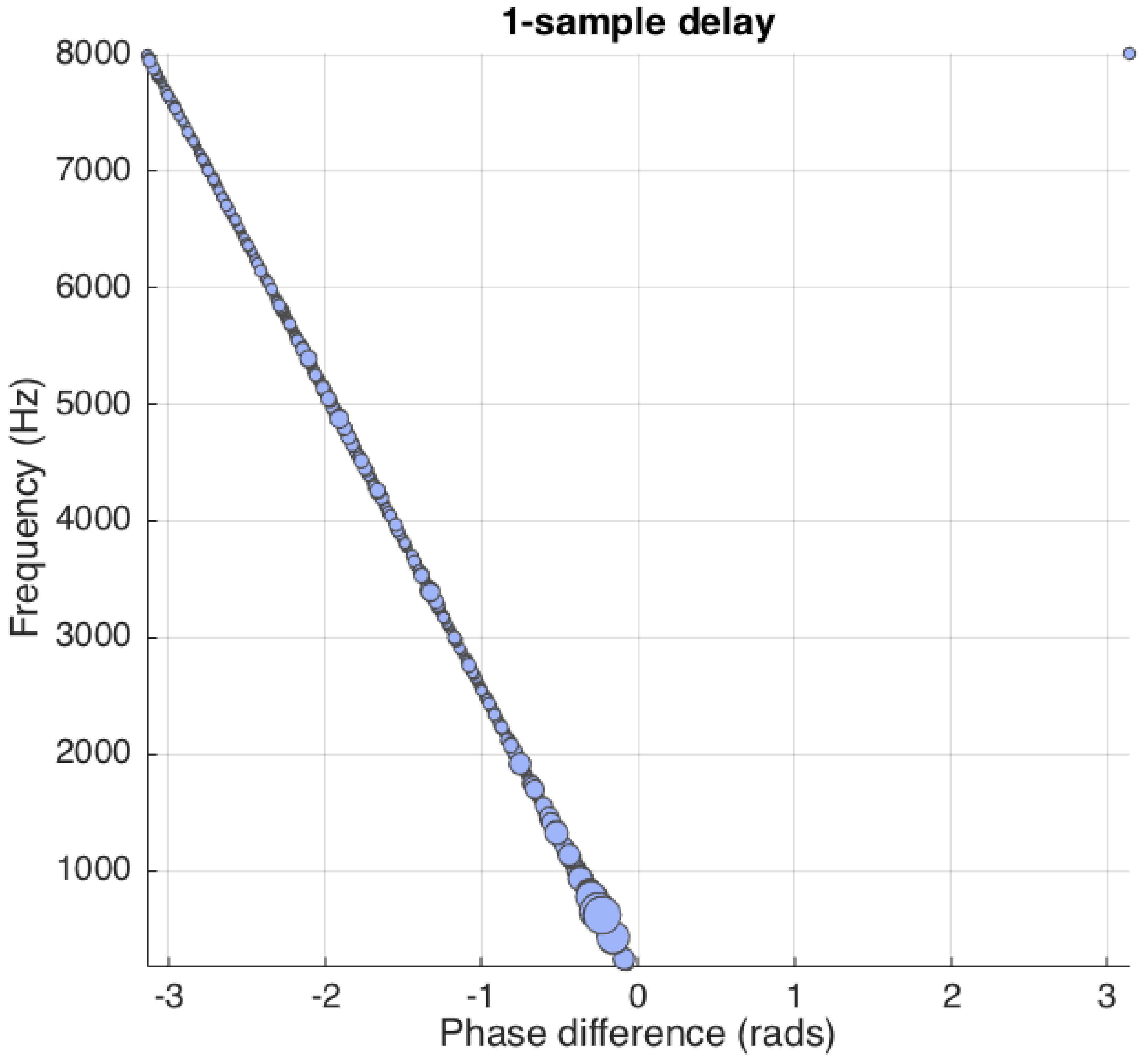
$$\delta_{f,t} = \angle F_{f,t}^{(1)} - \angle F_{f,t}^{(2)}$$

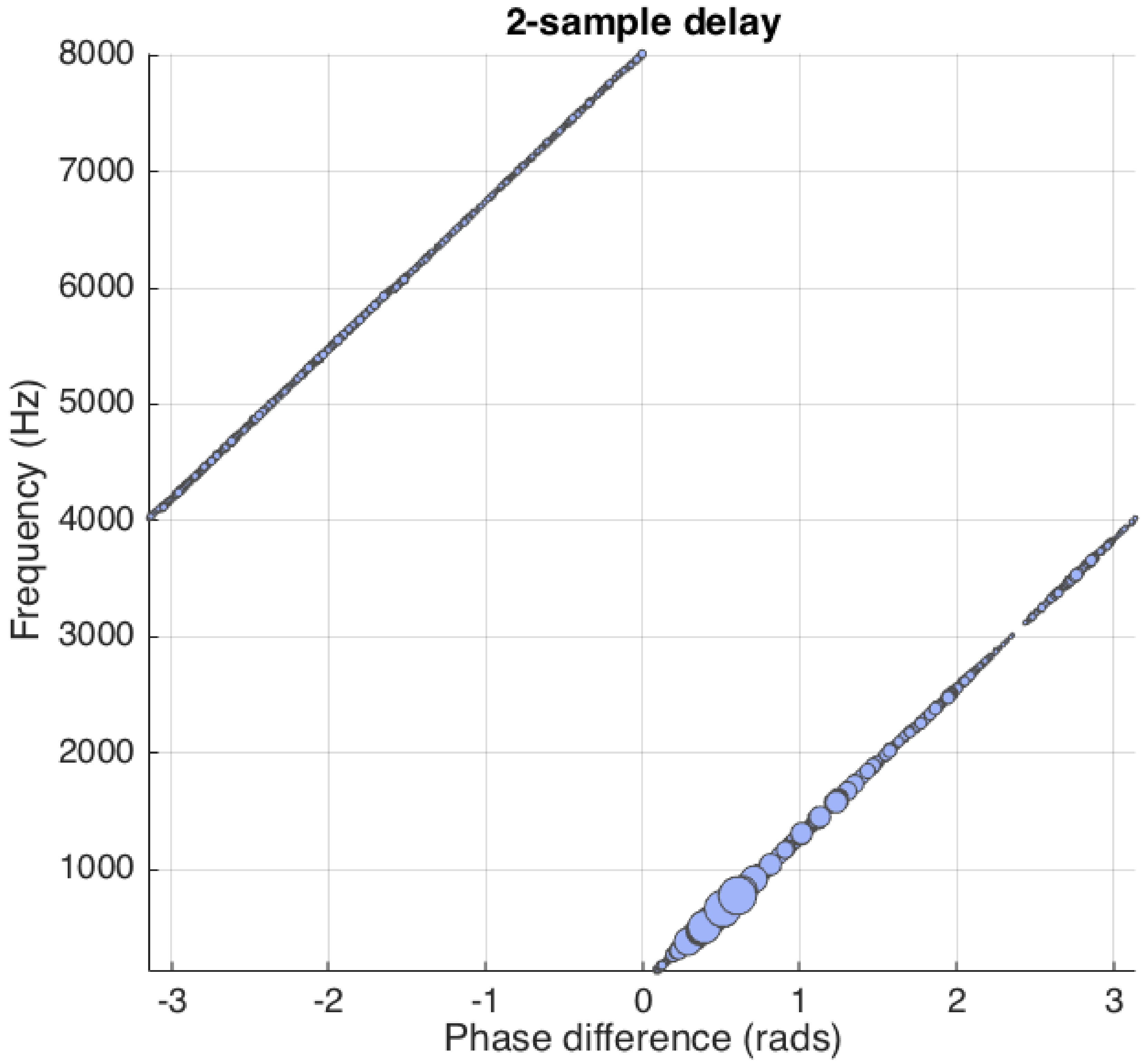
Spectrogram of mic 2

- Each spatial location has it's own set of values over f
  - Note that these are values between -π and π

What does this look like?

- For one source: scatter plot of points along a line
  - The line slope denotes the delay between the two channels
    - Each point corresponds to a time/frequency bin

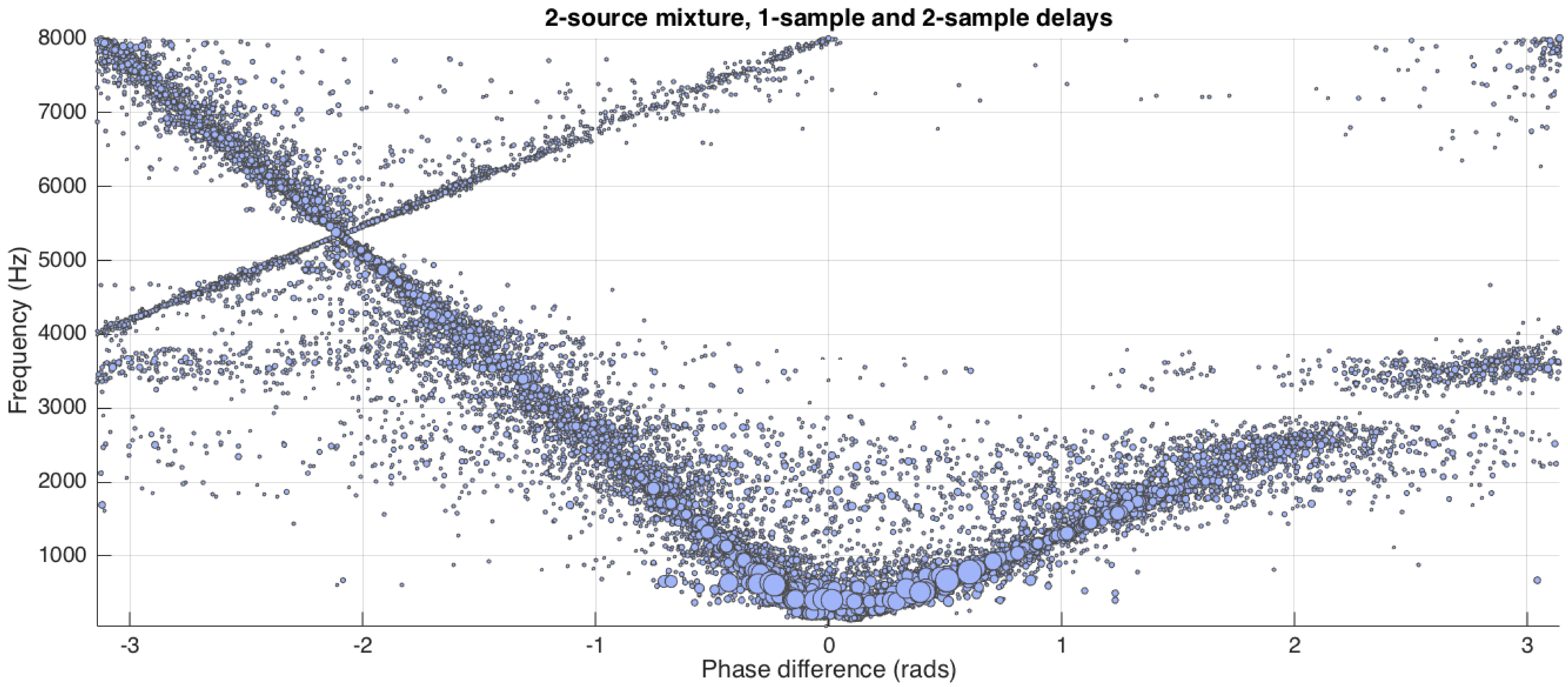




UNIVERSITY OF ILLINOIS AT URBANA CHAMPAIGN

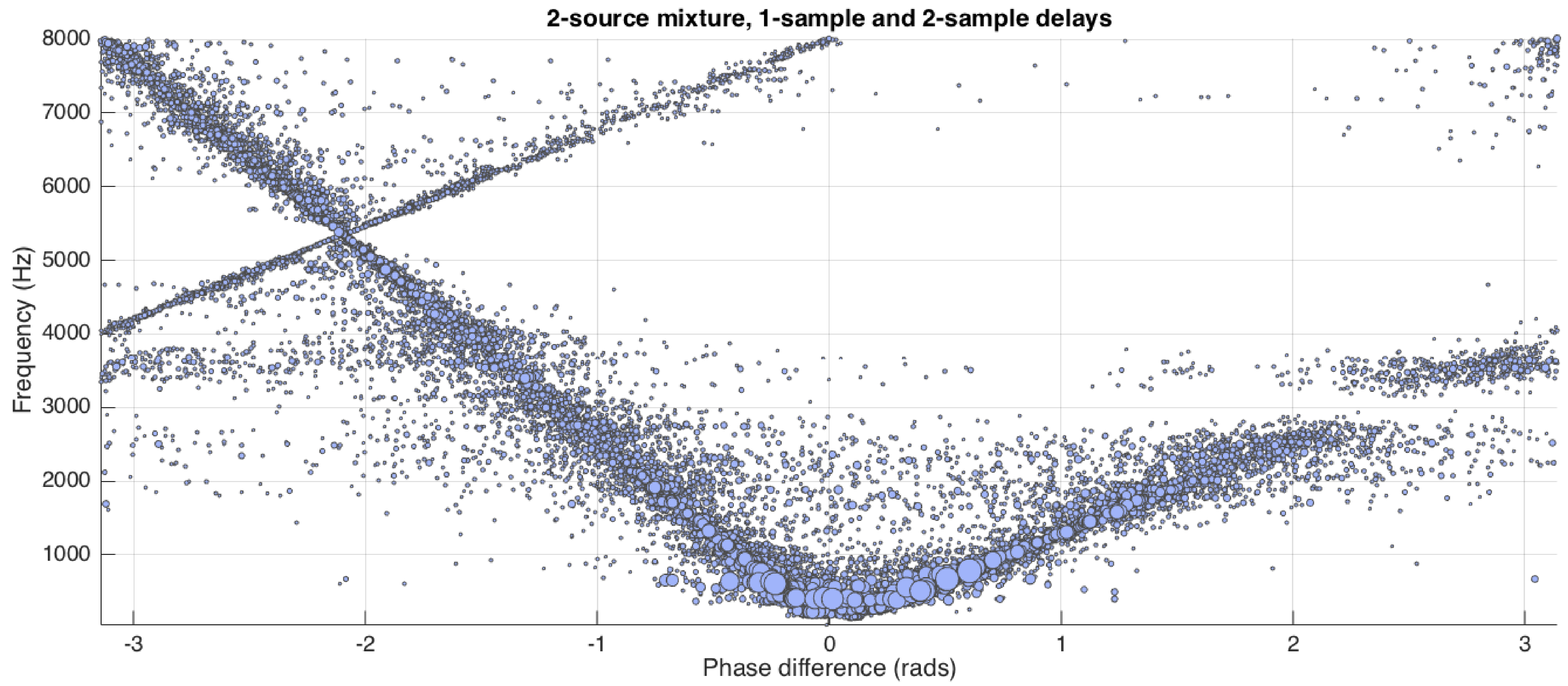
What about mixtures?

- Each source gets its own line depending on the delays
  - That is thanks to time/frequency disjointness between sources



A thought …

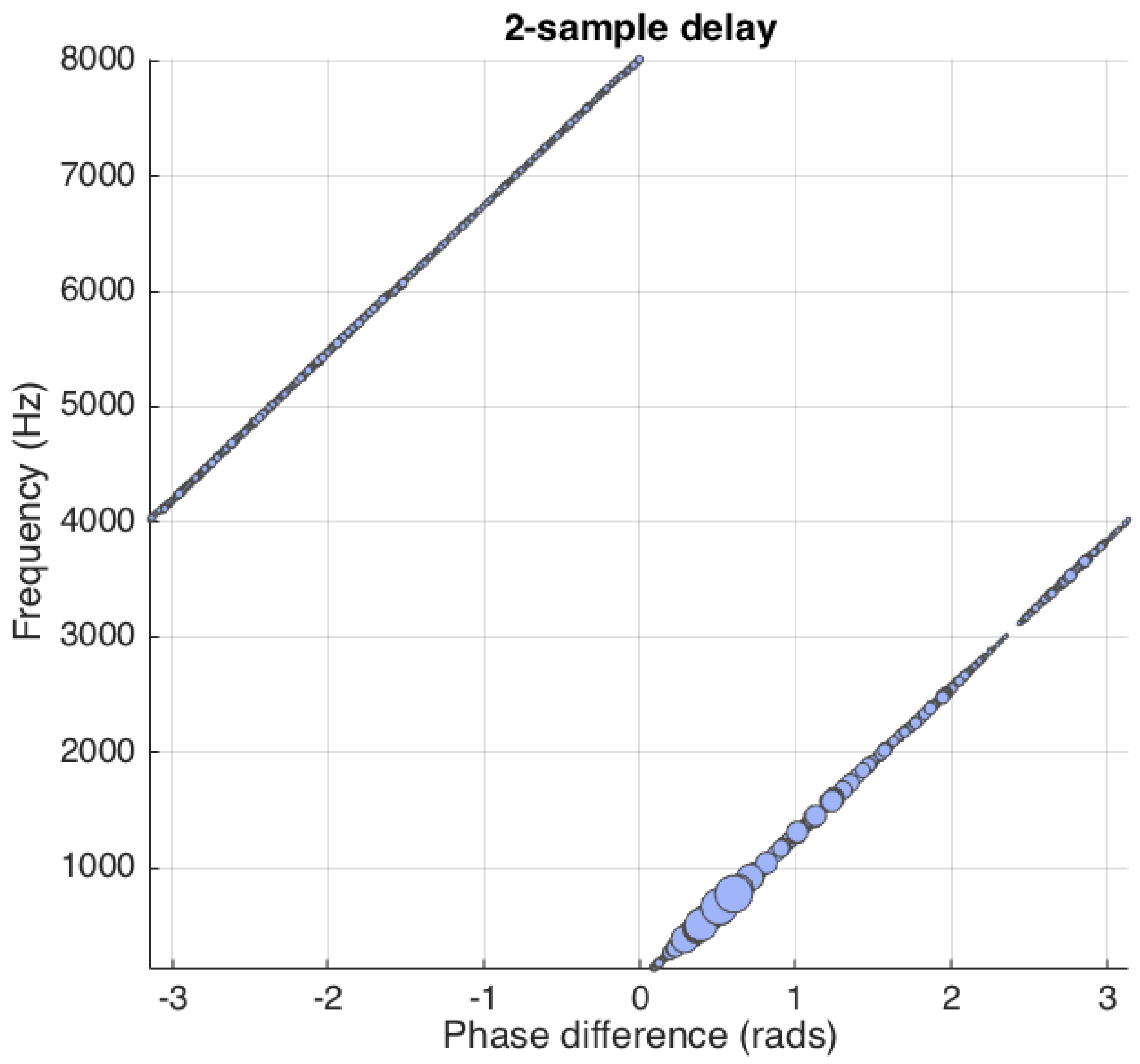
- Each source's dominant t/f bins lie on a wrapped line
  - We can make masks for each source using that information



- A problem:
  - Find the number and the slope of the wrapped lines
    - Number of lines ⟶ numbers sources
    - Slope of line ⟶ location of source

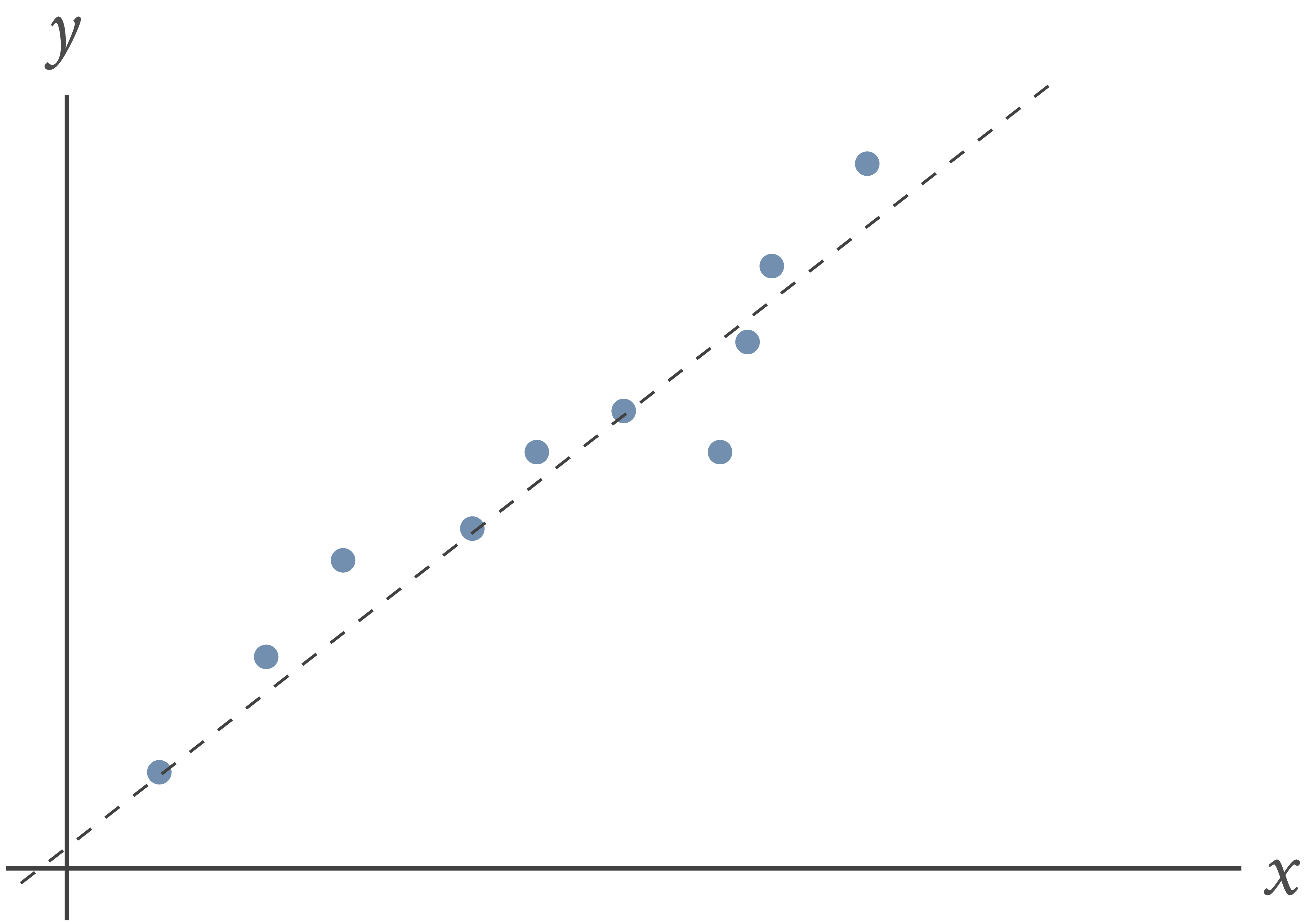
Models for wrapped data

- Linear-circular regression
  - Predict the phase difference values from the frequency index
  - i.e. a linear model on *f*, which then gets wrapped as a phase
- Problem: Not a linear model!
  - Phase values wrap inside {-*π*,*π*}
- We need something else
  - Multiple options available



Bayesian regression nomenclature

- Linear regression = minimize Gaussian error likelihood



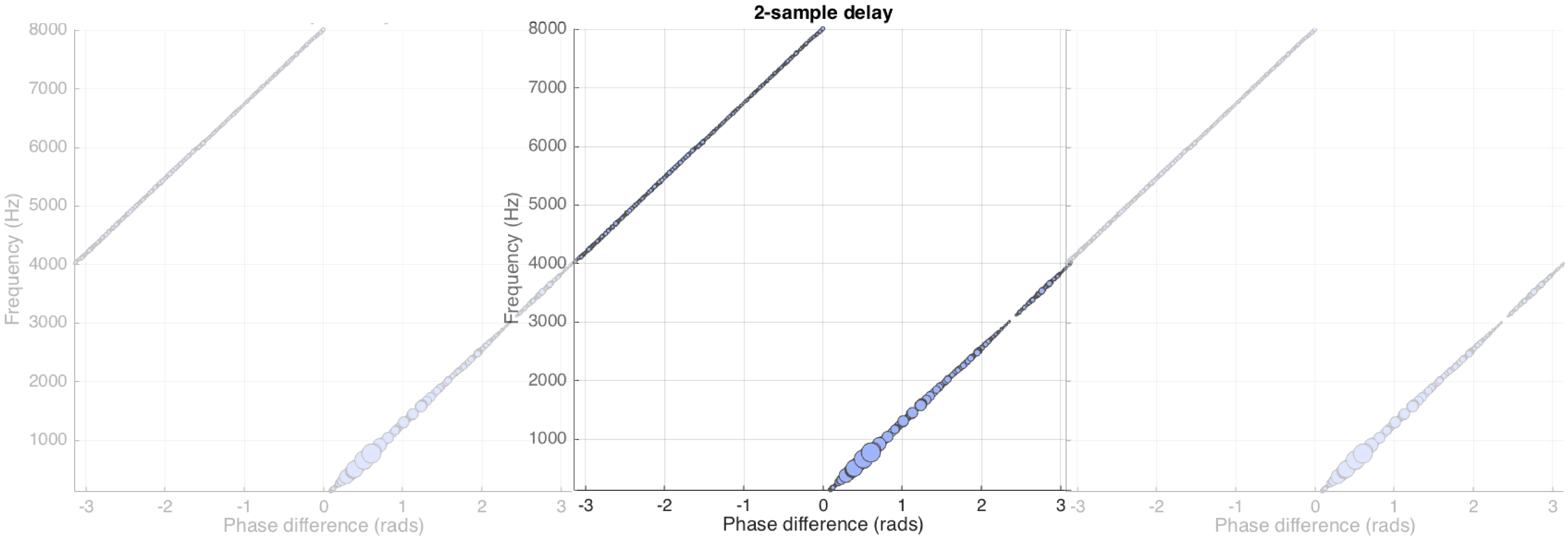
$$y_i = \alpha x_i + n_i$$

$$p(y;\alpha;\sigma^2) = \prod_{i=1}^{N} \mathcal{N}\left(y_i;\alpha x_i,\sigma^2\right)$$

Using the wrapped Gaussian

- Model the data as repeating regressions every 2π
  - Effectively use a sum of infinitely repeating Gaussians

$$p(\delta;\alpha;\sigma^2) = \prod_{f=1}^{D} \sum_{l=-\infty}^{\infty} \mathcal{N}\left(\delta_f;\alpha f + 2\pi l, \sigma_f^2\right)$$



A model for multiple sources

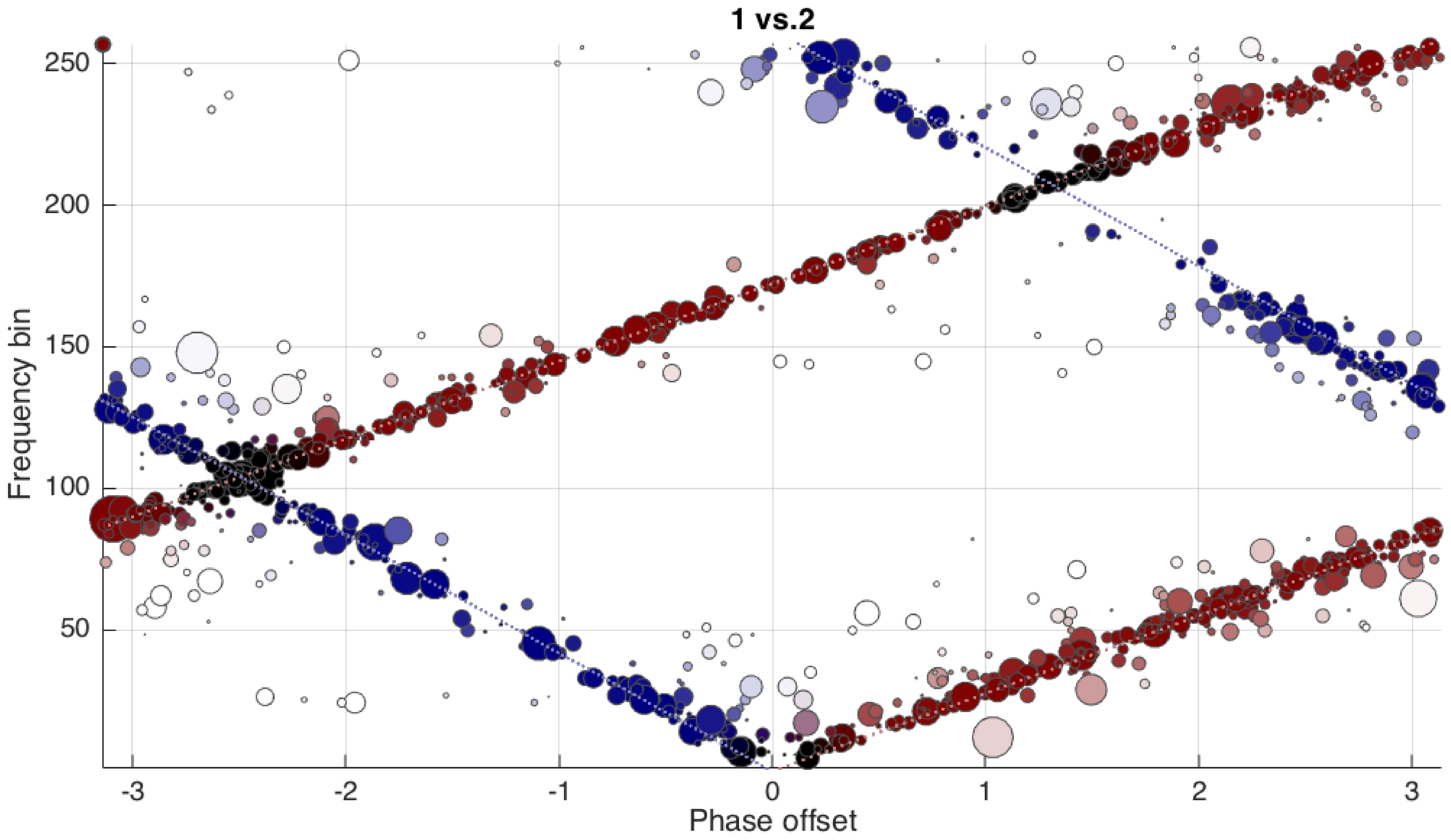
- Likelihood for explaining a mixture of K sources
  - Each source has its own line

$$p(\delta;\alpha;\sigma^2,q) = \prod_{f=1}^{D}\sum_{j=1}^{K} q_{i,j} \sum_{l=-\infty}^{\infty} \mathcal{N}\left(\delta_f;\alpha_j f + 2\pi l, \sigma_{j,f}^2\right)$$

- We can learn this in a variety of ways
  - Expectation-Maximization (accurate, slow)
  - RANSAC (accurate enough, really fast)

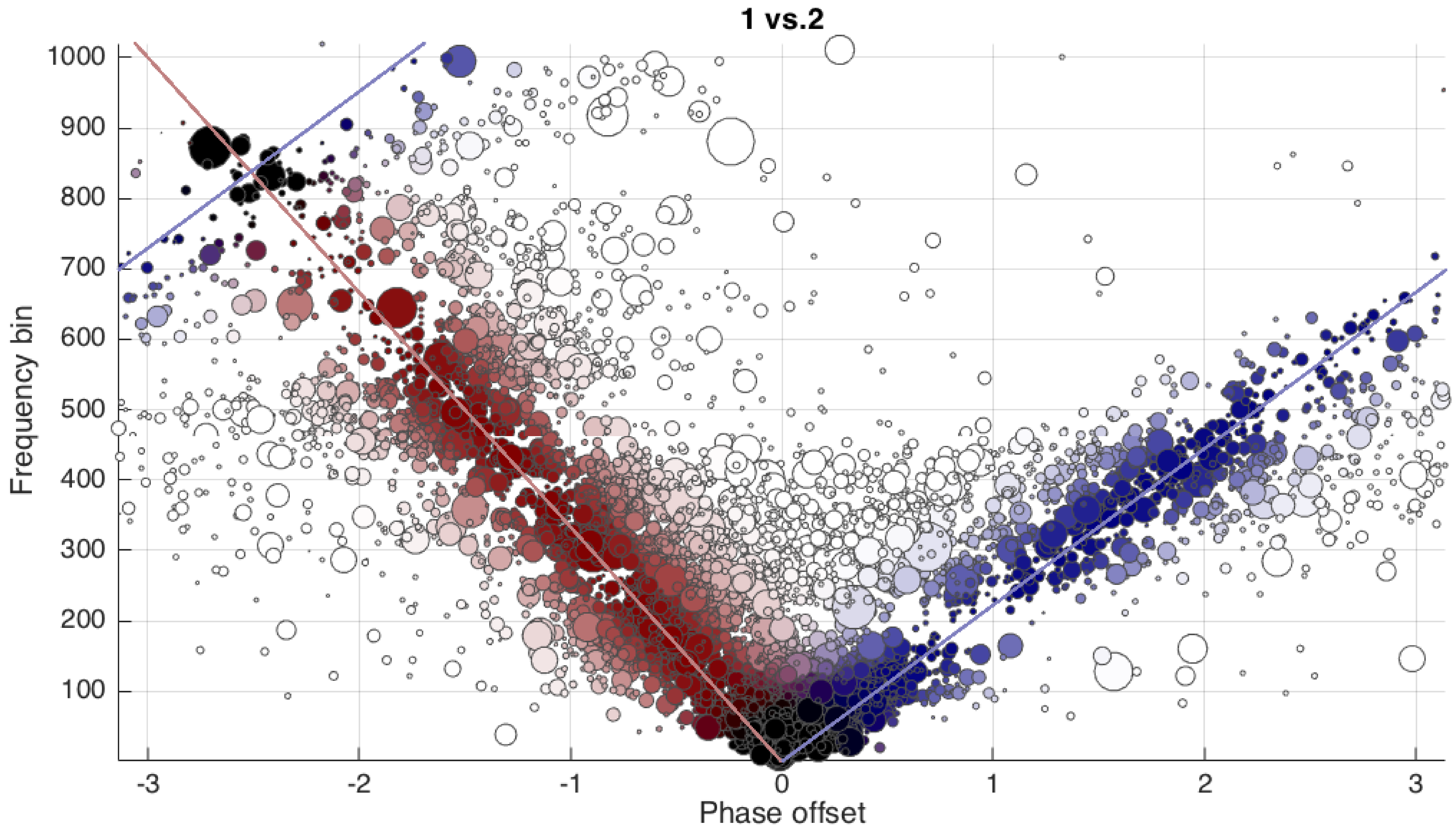
A toy example

- Simple {+2,−3} delay mixture



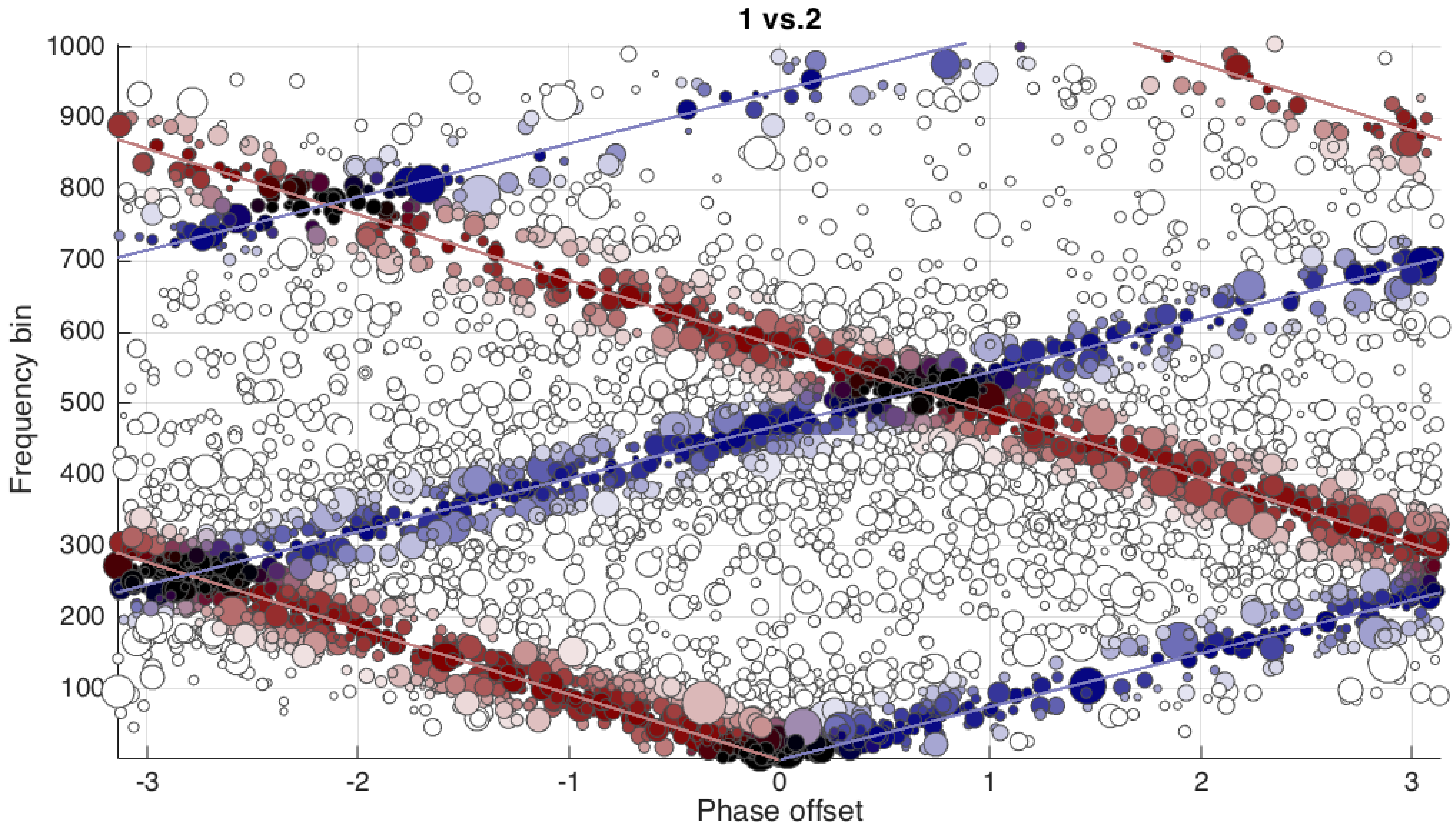
In real-life

- Two people in one office, small delays



Extreme case

- Stairwell with strong reverberation, larger delays




Mix


Paris


Johannes

Some interesting features

- Arrays should be shorter
  - We like short delays
    - Too much wrapping can be a problem
- Sample rate should be low
  - Again keeps the delays shorter
- No need for tedious calibration
  - Also generalizes easily for multiple microphones

But …

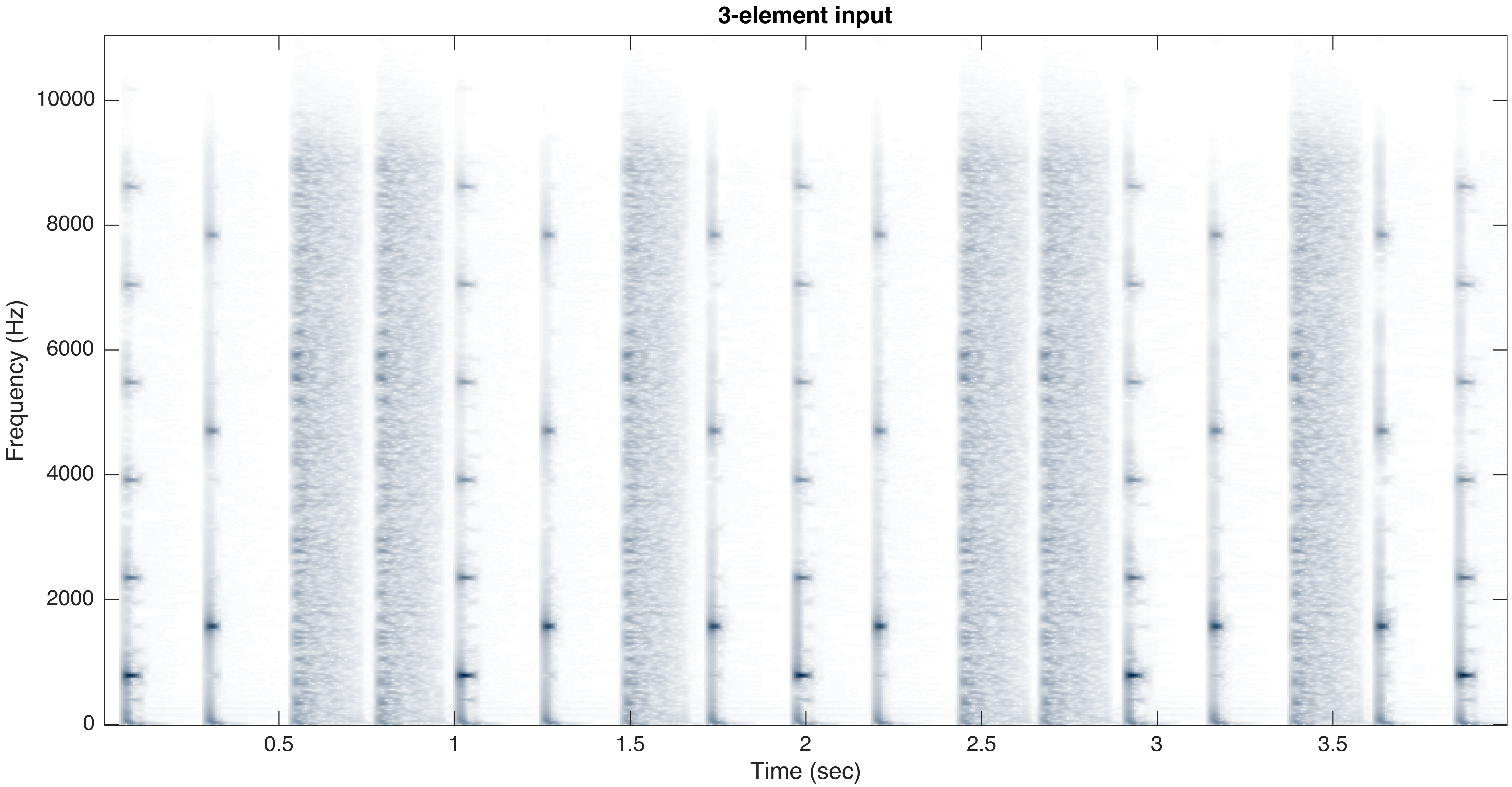
- Arrays are good, but can be expensive
  - Not so much an issue today, but still not as widespread
- The holy grail is single-channel signal processing
- A complication
  - No spatial domain, no good way to "point" to a source
    - So let's do that!

Models on magnitude spectra

- Phase can be uninformative so we will ignore it
  - We can do a lot of denoising on magnitude spectrograms instead
- Key property here: Data is non-negative
  - Which means we can't use typical MSE-based methods
- Promising area: Non-Negative Factorizations
  - Lots of work on this area in the last 10 years

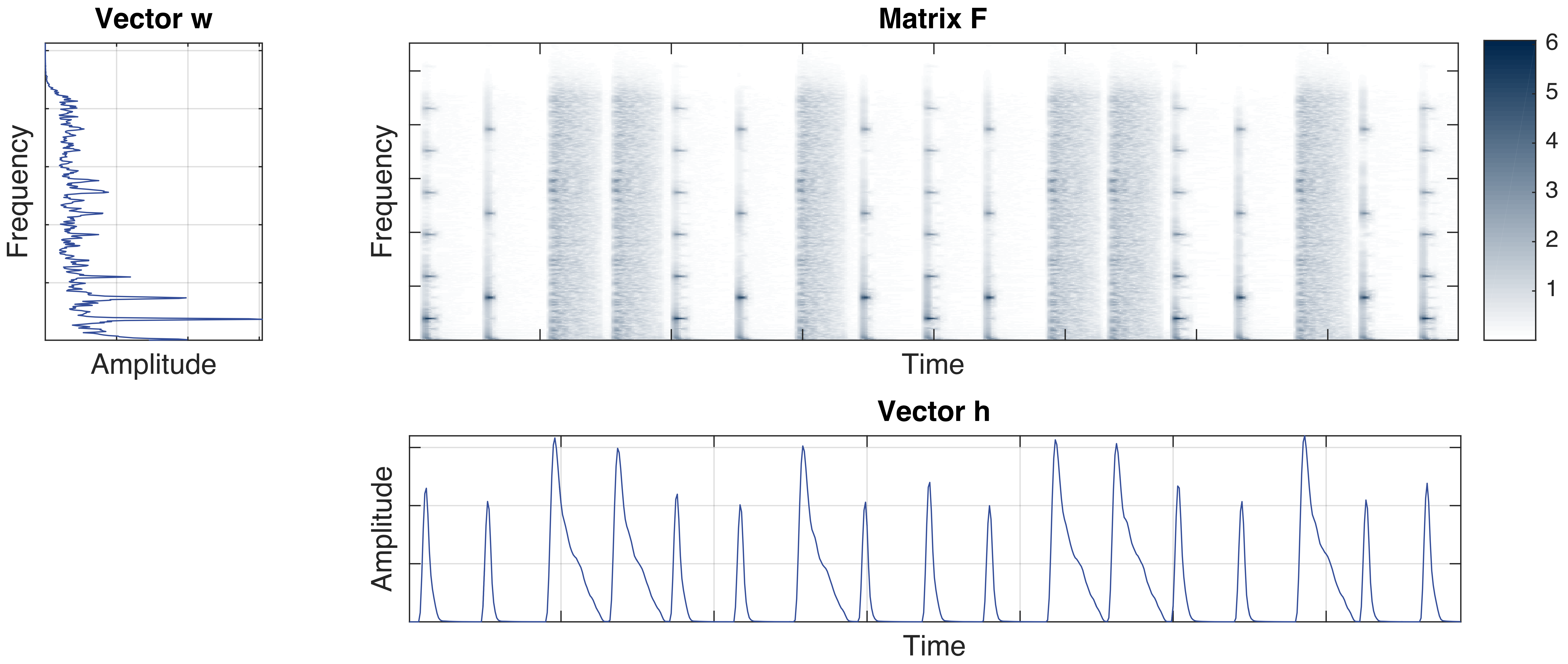
Simple example

- A drum loop with three distinct tones
  - Two blips, one snare



A simple factorization model

- Factorize magnitude spectrogram as: $F_{i,j} \approx w_i h_j$
  - All three quantities are non-negative



Towards a richer model

- Simple factor model learns spectrum & envelope
  - Doesn't help much in interpreting the input

- We can instead use multiple spectra/envelopes:

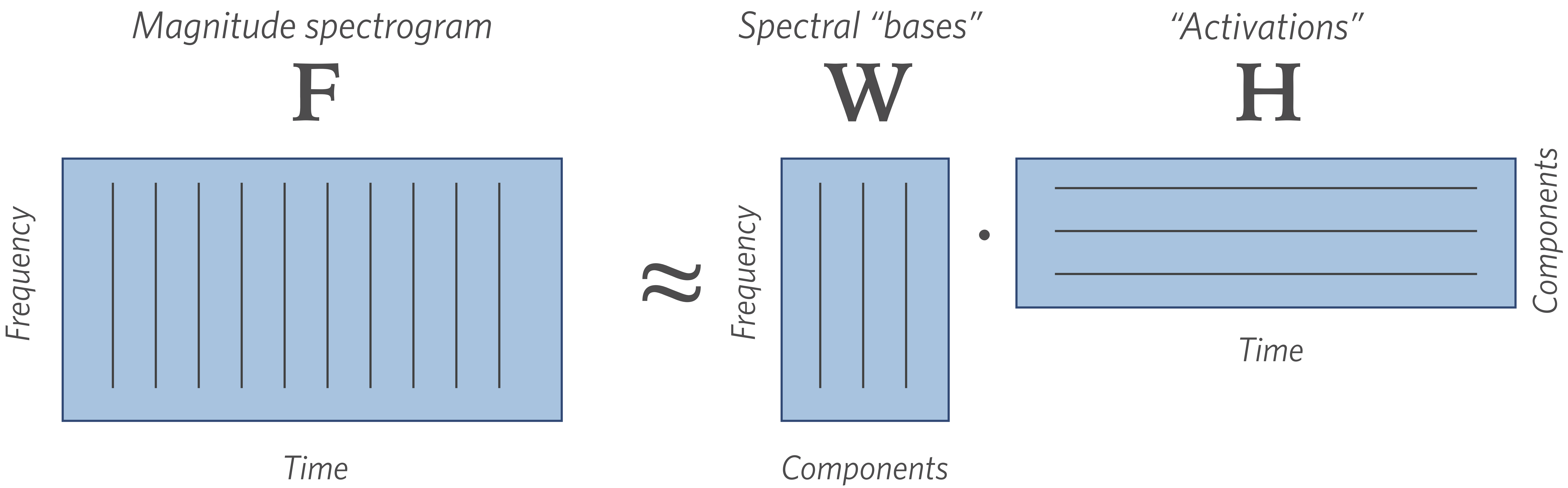
$$F_{i,j} \approx w_i^{(1)} h_j^{(1)} + w_i^{(2)} h_j^{(2)} + \ldots$$

$$\Rightarrow \mathbf{F} \approx \mathbf{w}^{(1)} \cdot \mathbf{h}^{(1)} + \mathbf{w}^{(2)} \cdot \mathbf{h}^{(2)} + \ldots$$

$$\Rightarrow \mathbf{F} \approx \mathbf{W} \cdot \mathbf{H}, \quad \mathbf{F} \in \mathbb{R}_+^{(M \times N)}, \mathbf{W} \in \mathbb{R}_+^{(M \times K)}, \mathbf{H} \in \mathbb{R}_+^{(K \times N)}$$

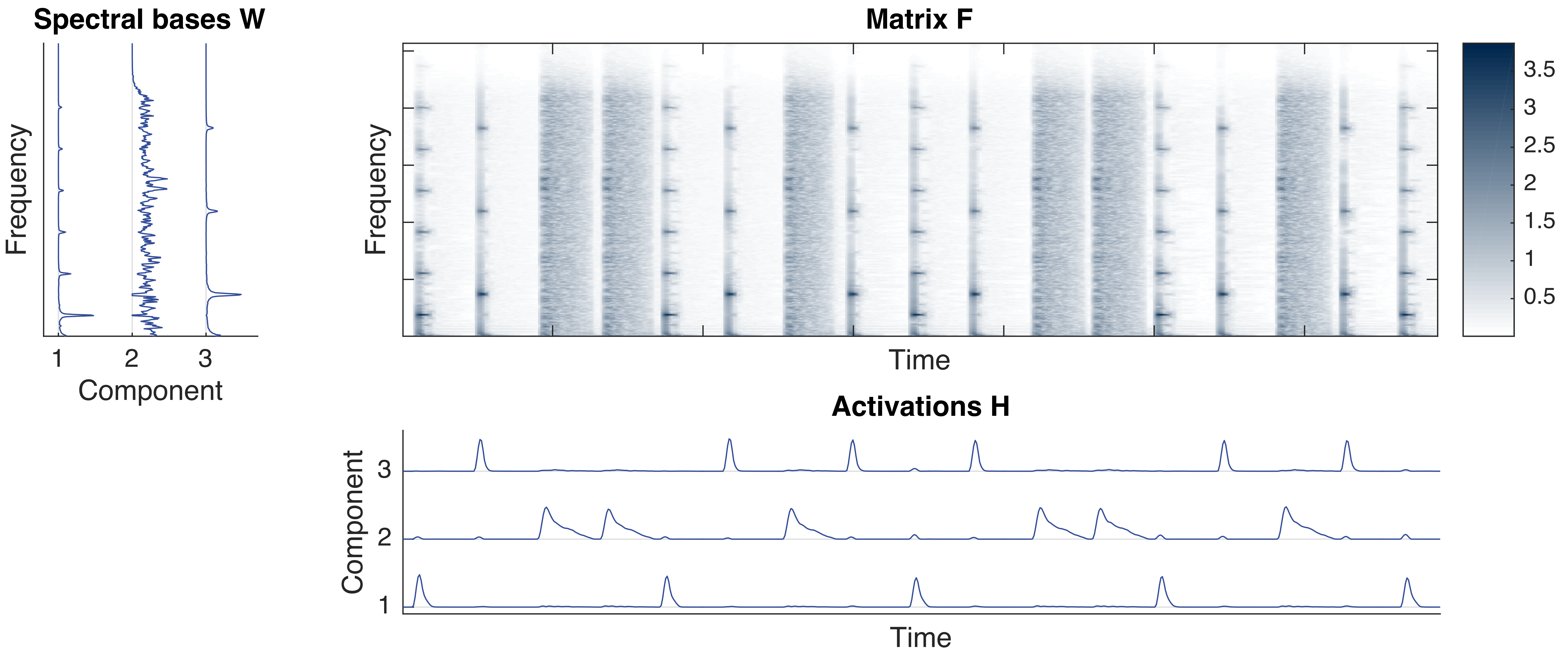
The pretty picture version

- The input is decomposed as a combination of spectral bases $\mathbf{W}$ and their corresponding activations $\mathbf{H}$
  - Each pair of spectrum/activation makes a "component"



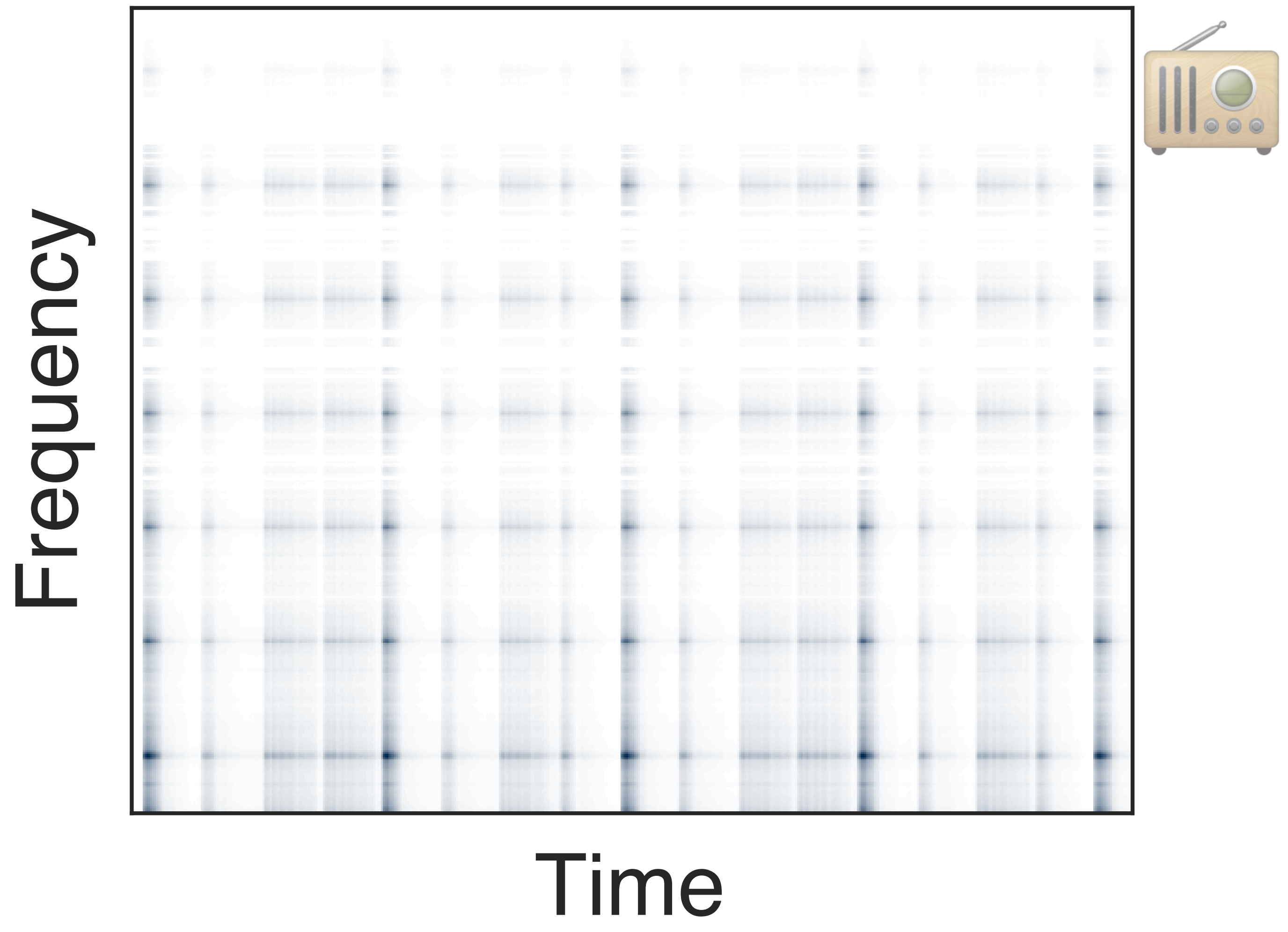
Back to the original example

- This model results in a more descriptive output
  - Each component describes a different sound in the mix

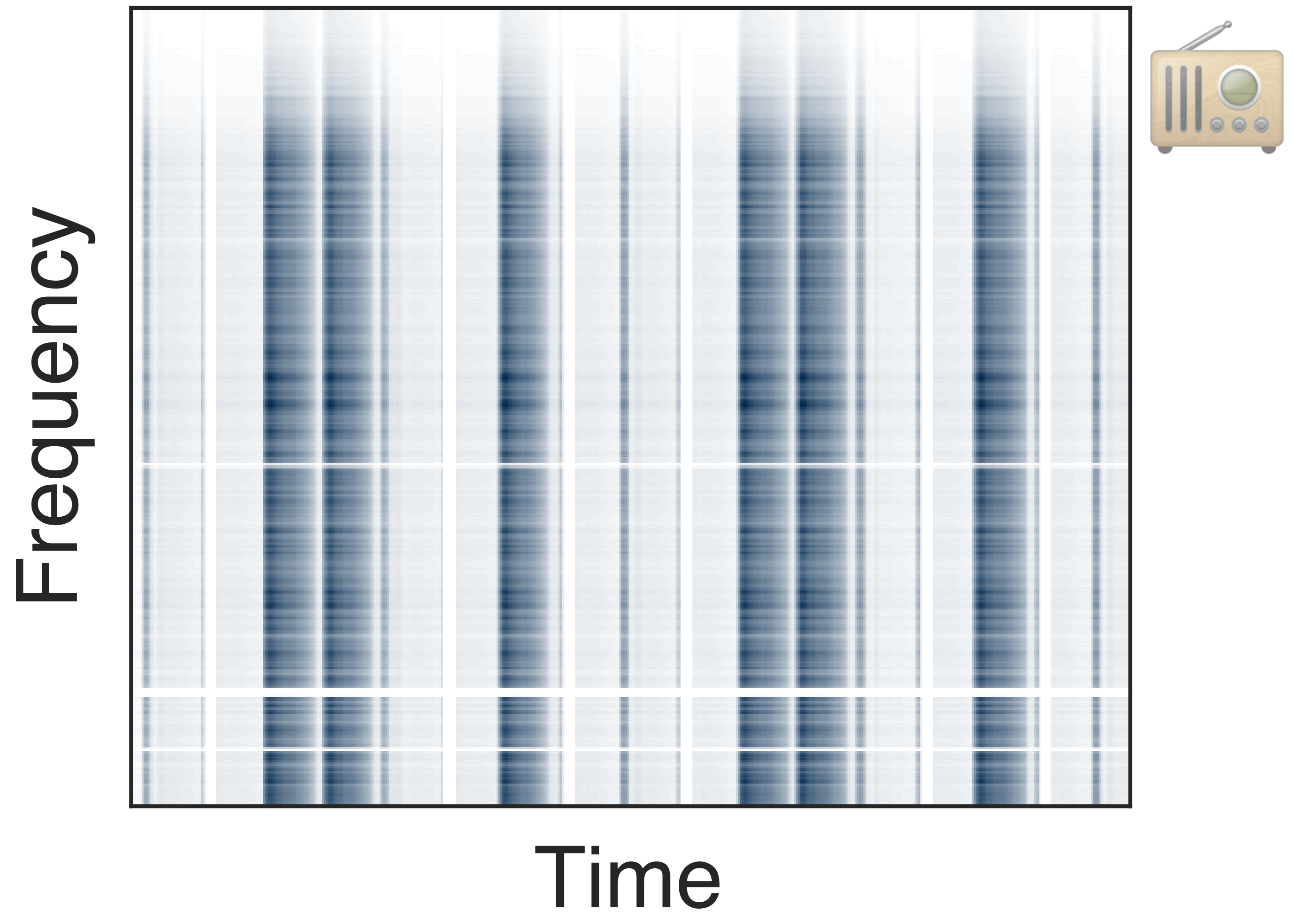


Component contributions

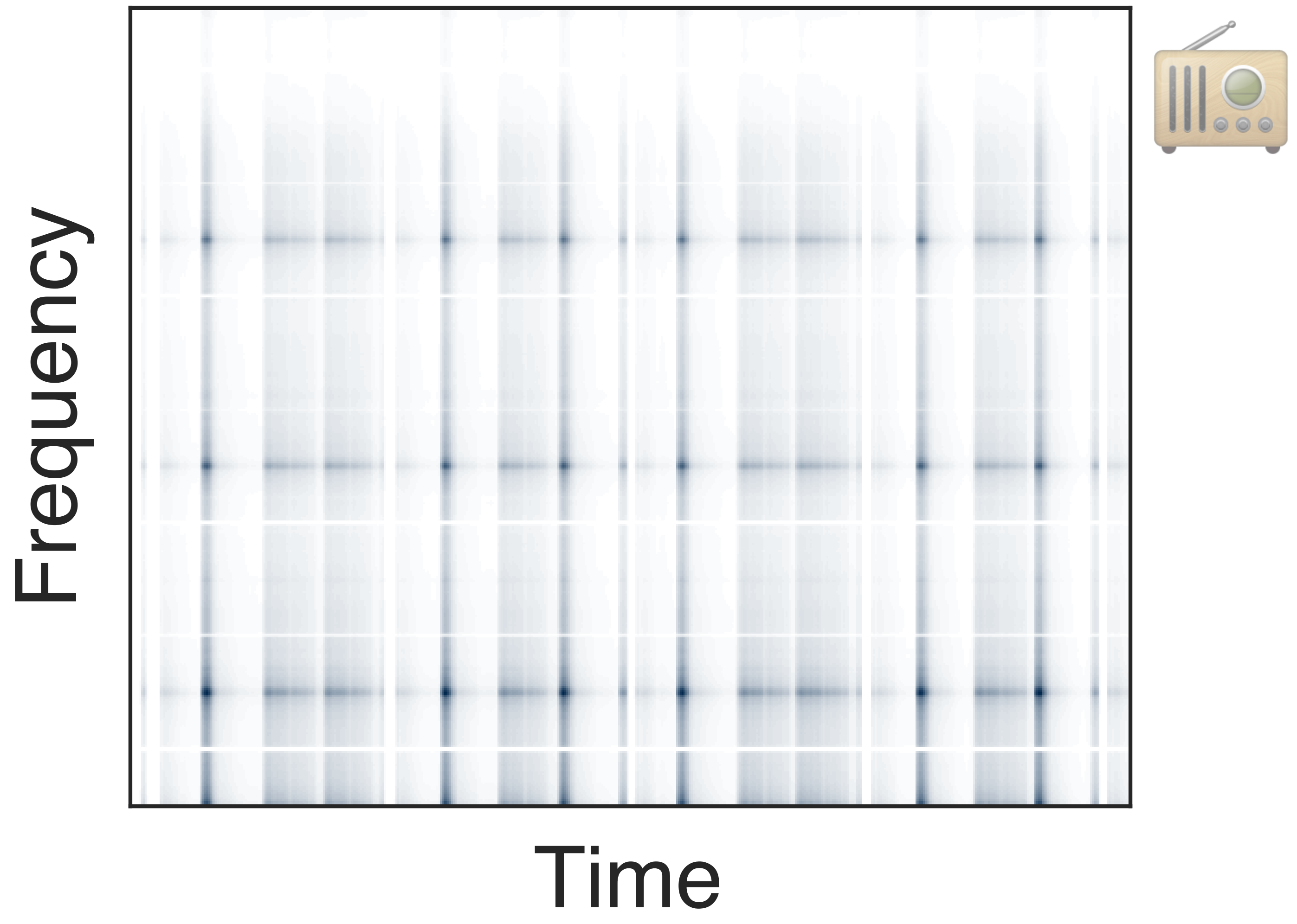
Component 1: $F_1 = w_1 \cdot h_1$



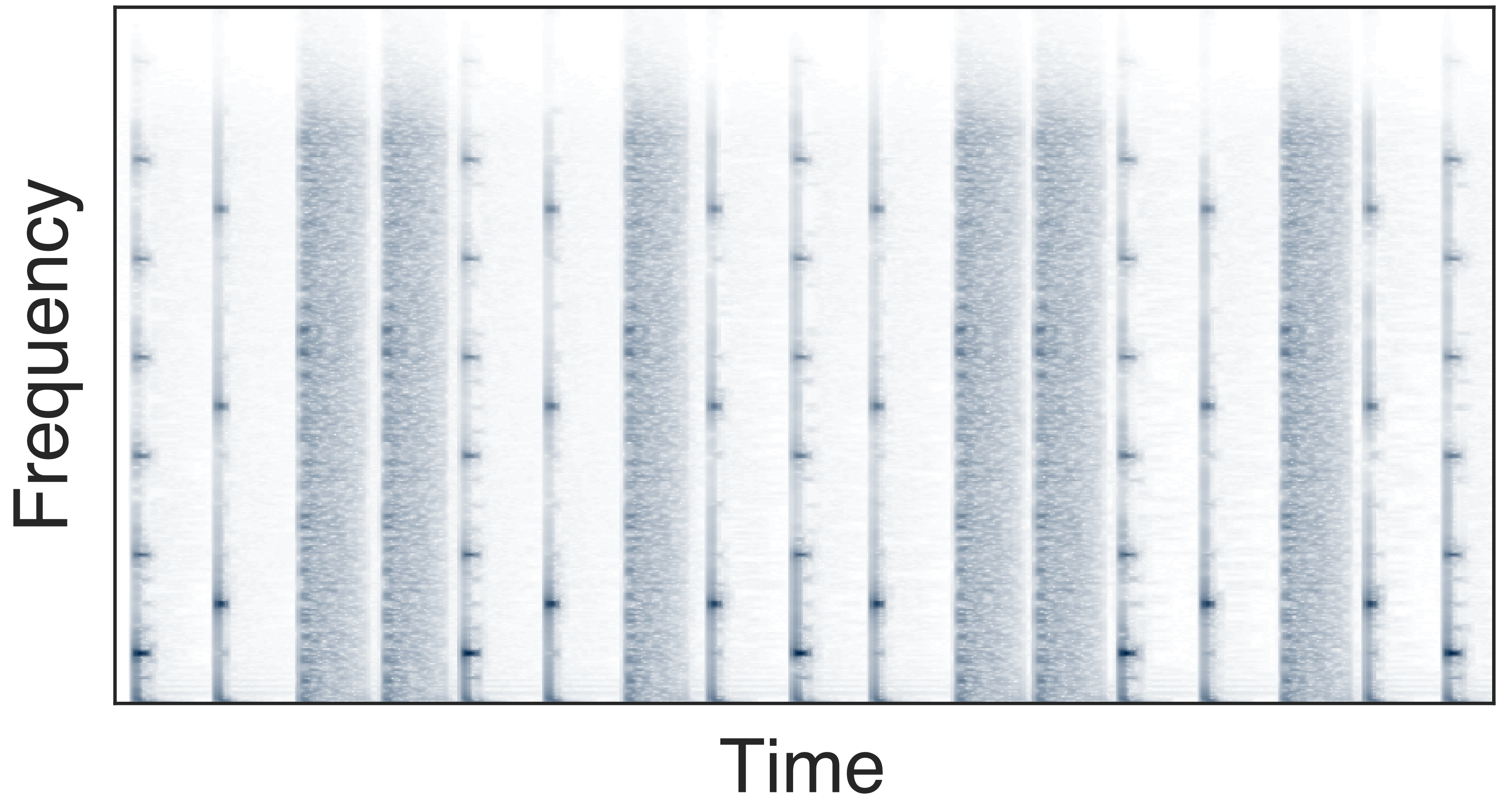
Component 2: $F_2 = w_2 \cdot h_2$



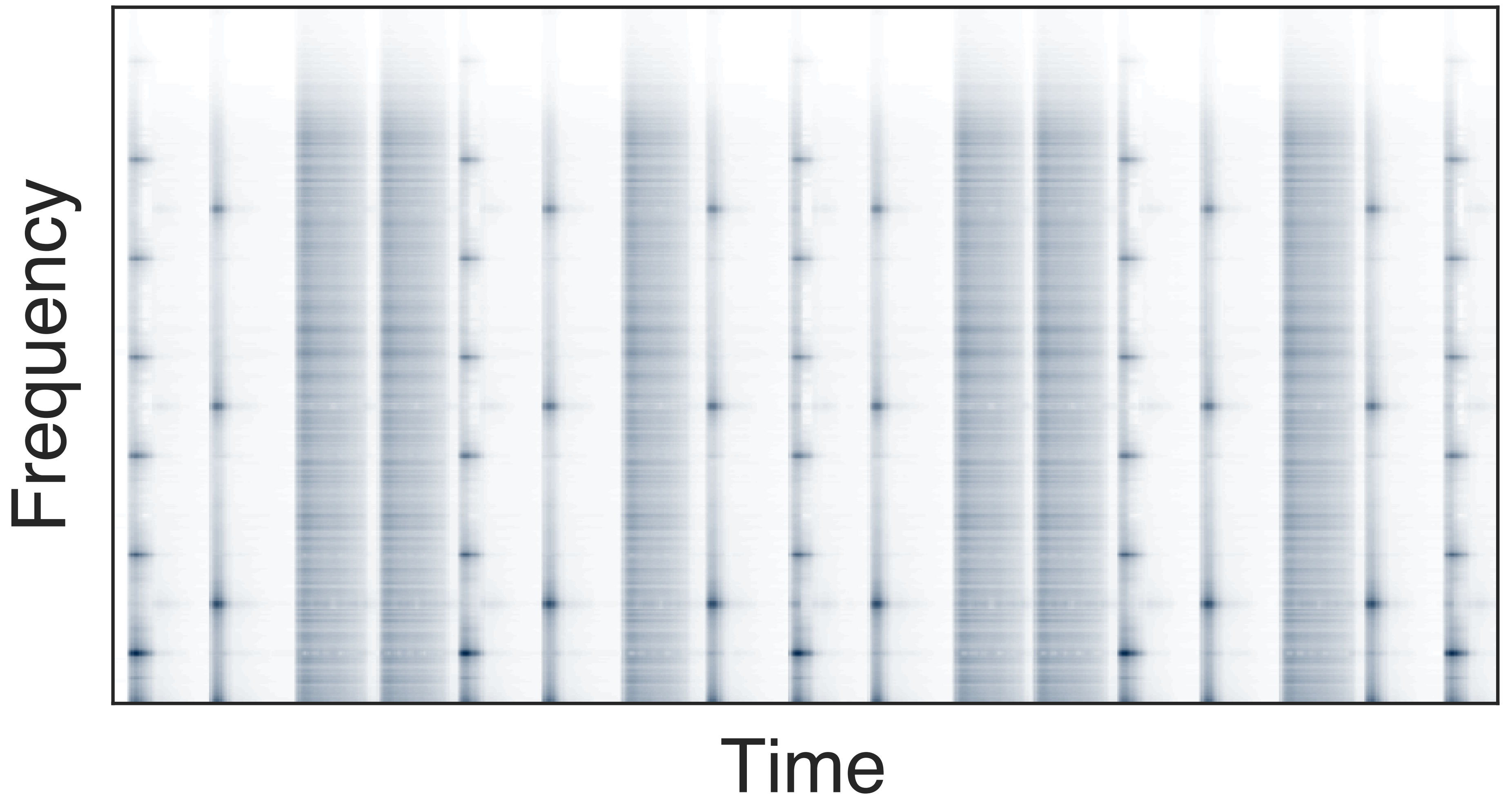
Component 3: $F_3 = w_3 \cdot h_3$



Original input



Sum of components: $F \approx F_1 + F_2 + F_3$

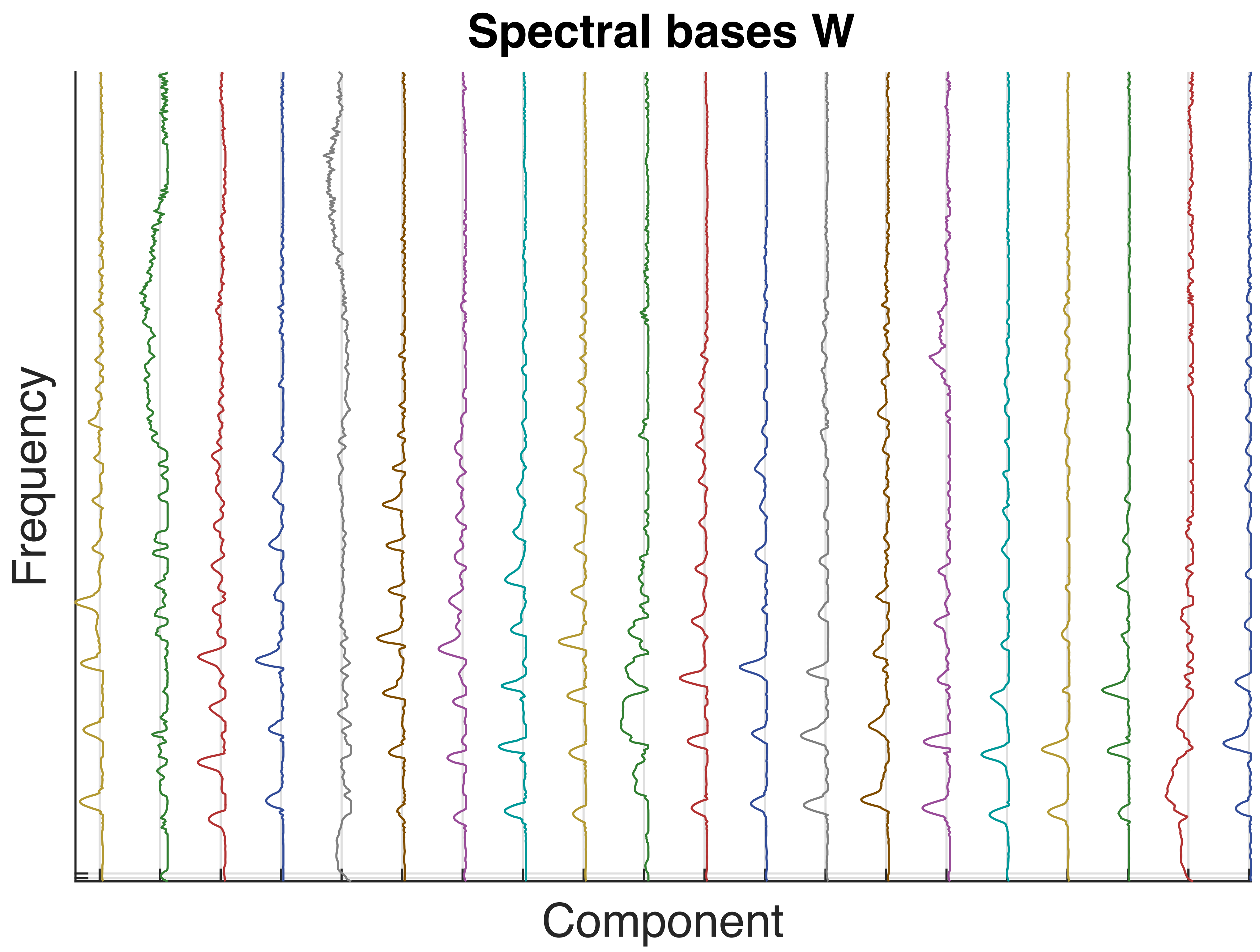


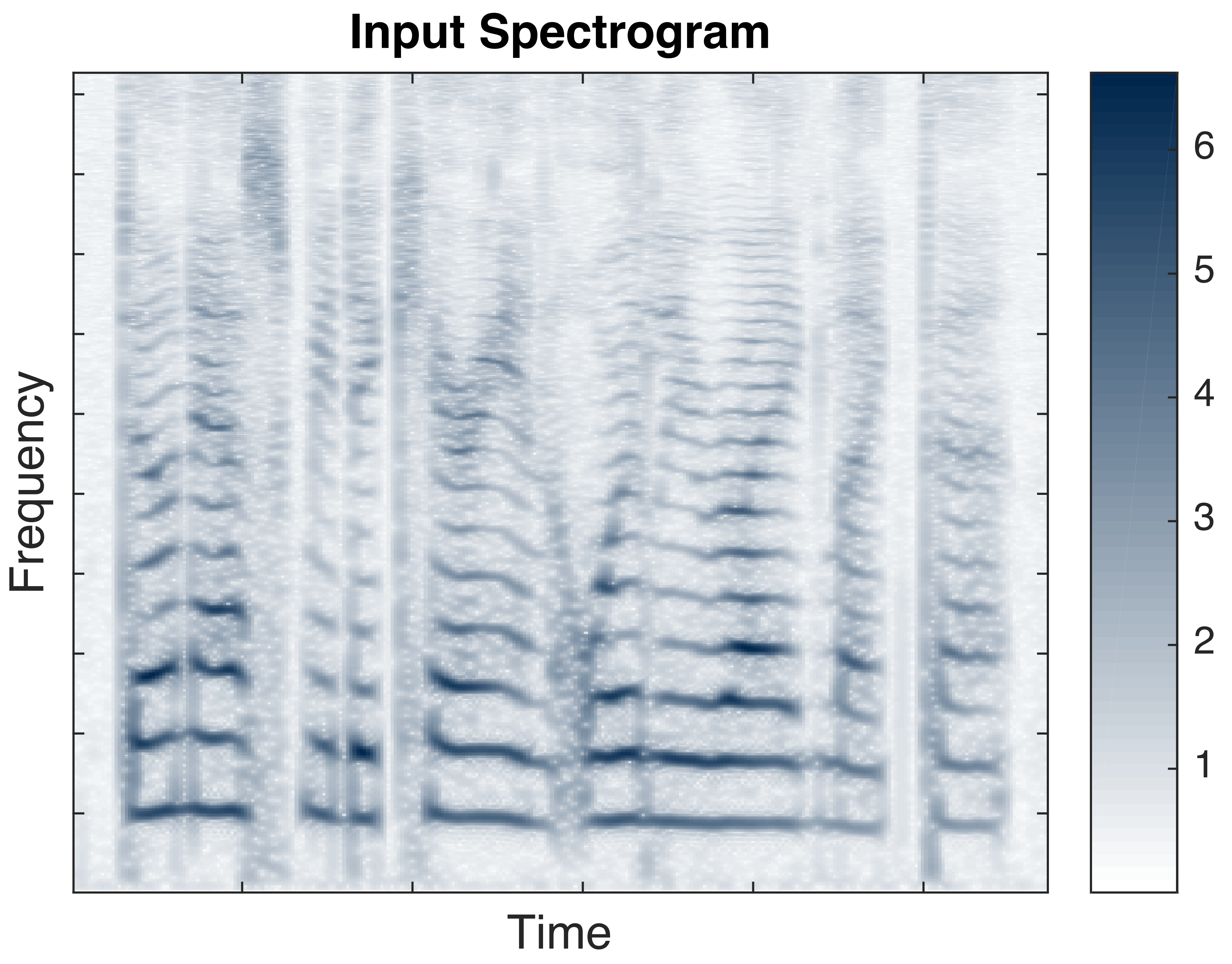
Can’t do miracles (yet)

- This model has some limitations
  - Components can only have a static spectrum
    - Fine for stationary sounds (drums, piano, etc), but not useful for speech
- We can however use this model to construct better ones
  - Non-Negative dictionary models

Learning a speech dictionary

- When applied on speech we (sort of) learn phonemes
  - Each component describes a characteristic spectrum of the input





Reconstruction of similar sounds

- Speaker-dependent dictionaries
  - Factorize spectra from training data of a speaker and get $\mathbf{W}$

$$\mathbf{X}_{train} \approx \mathbf{W} \cdot \mathbf{H}$$

    - Different speakers would have somewhat different spectral bases

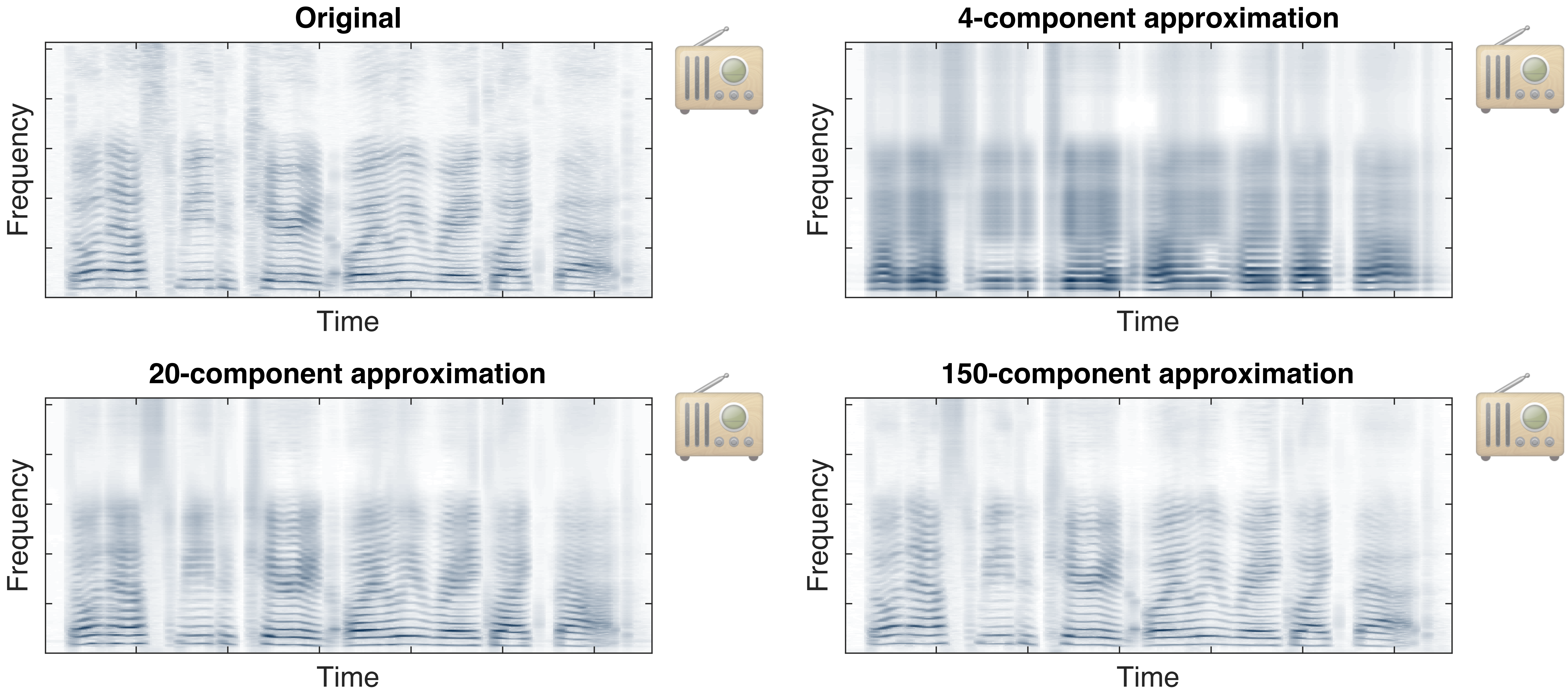
- We can resynthesize that speaker's voice using $\mathbf{W}$ only

$$\mathbf{X}_{test} \approx \mathbf{W} \cdot \mathbf{H}_{test}$$

  - Think of it as a complicated form of VQ coding

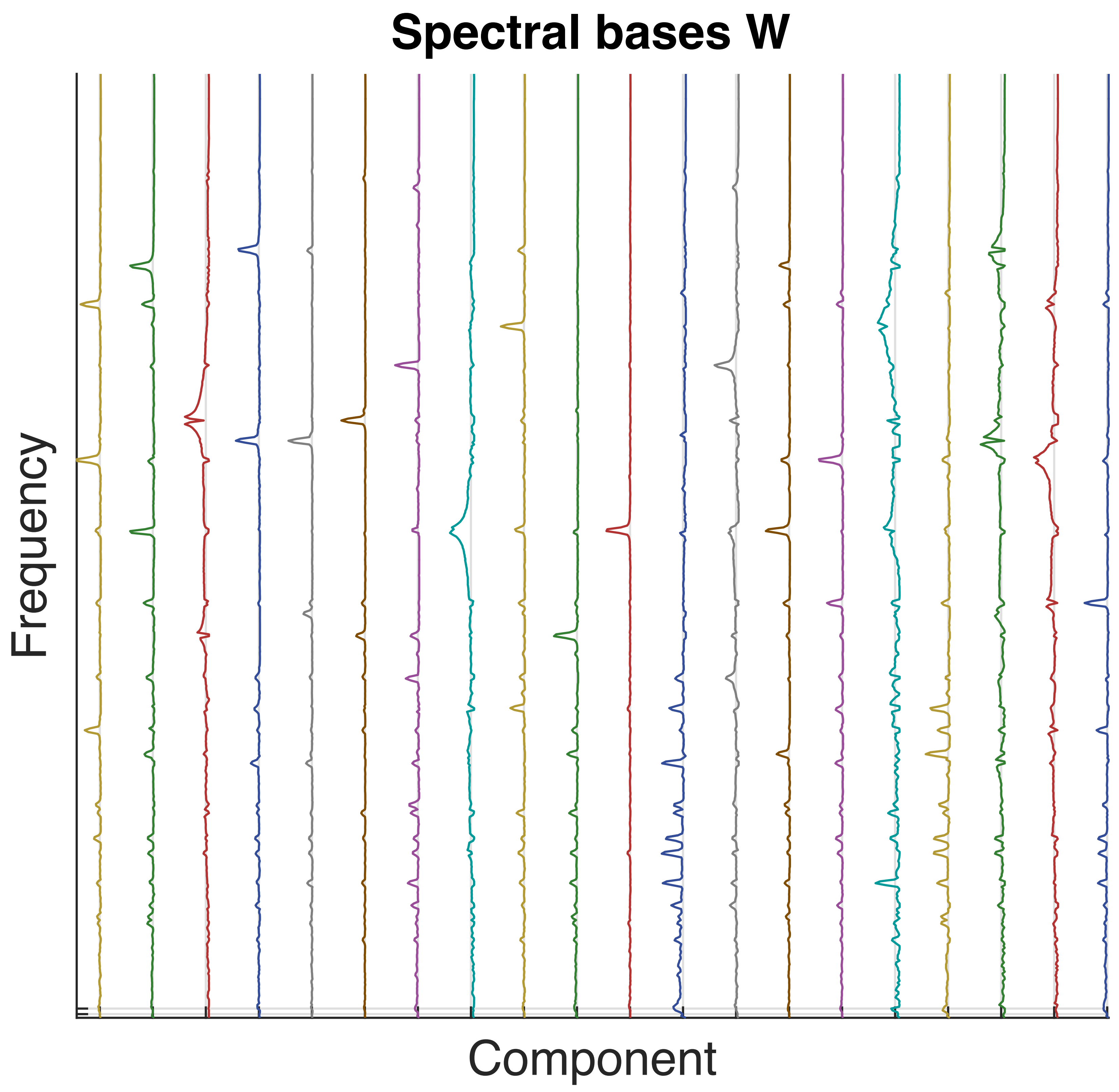
Pointless example

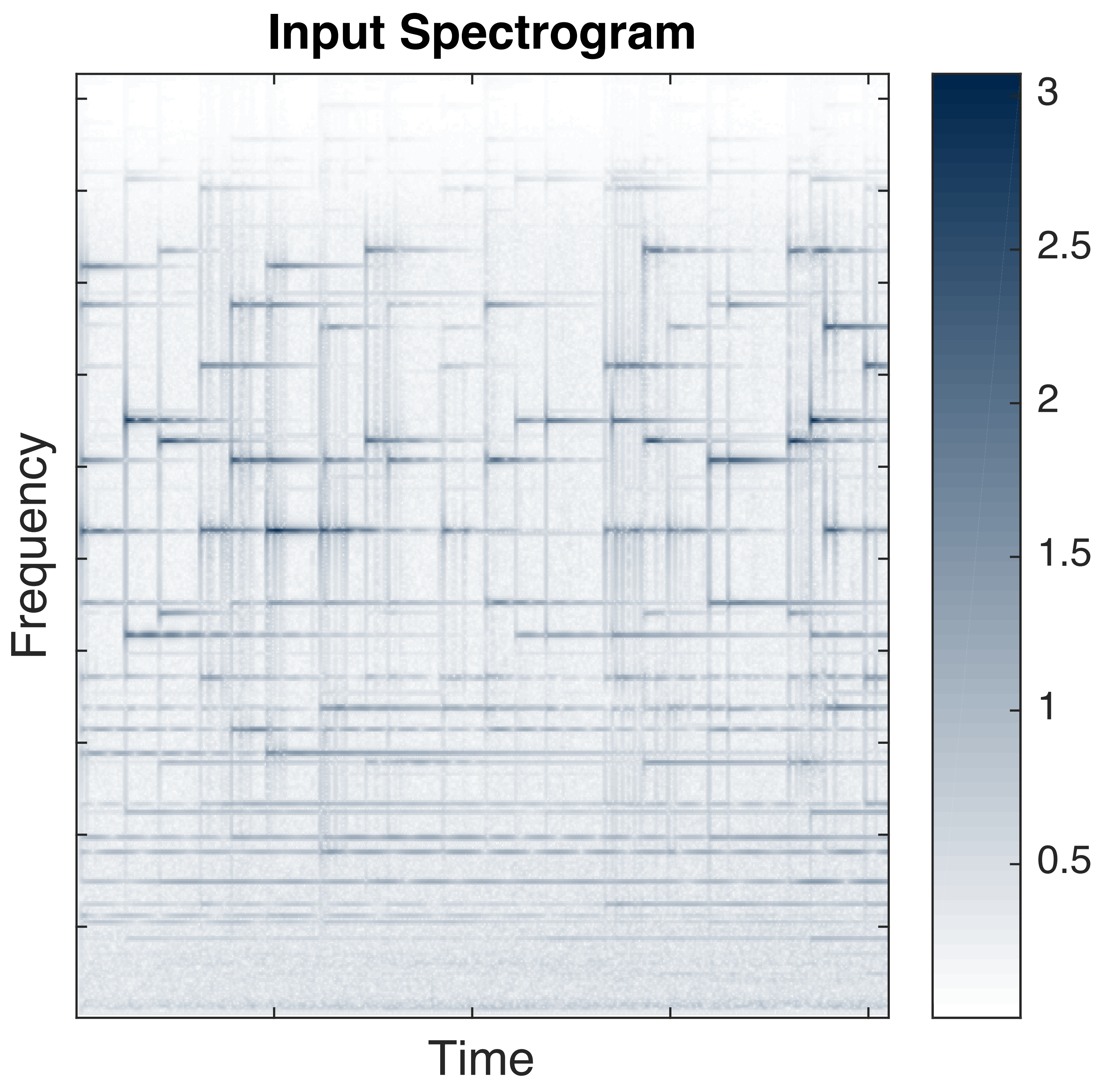
- Keep the phase; approximate the magnitude
  - Train on 9 sentences for $\mathbf{W}$, use it to approximate 10th sentence



Learning a different sound class

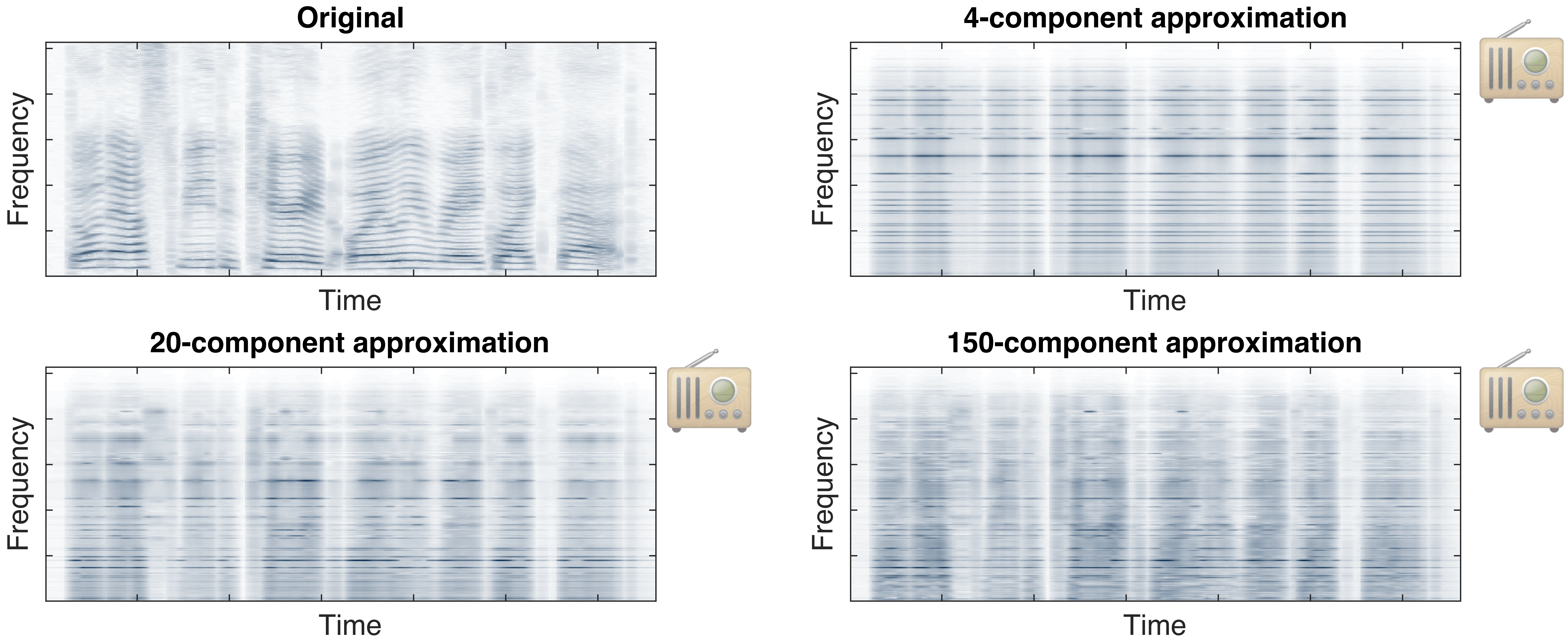
- Different types of sound have distinctly different bases
  - E.g. the chime bases below are very different from speech bases





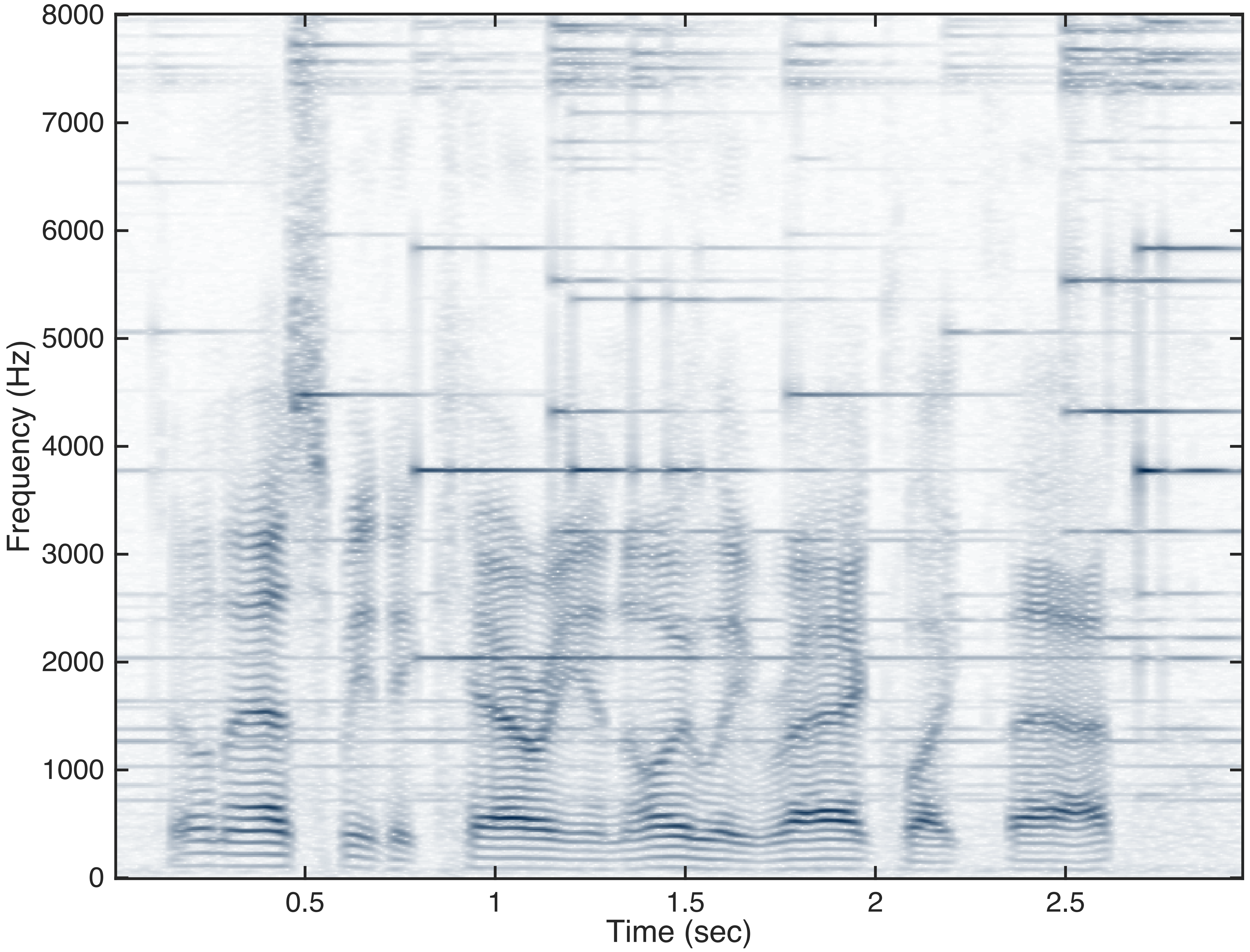
And these bases are not speech bases!

- If we approximate speech with the chime bases it produces a very poor approximation



An idea ...

- What if I have a mixture of two known sound classes?
  - How would I approximate this one?

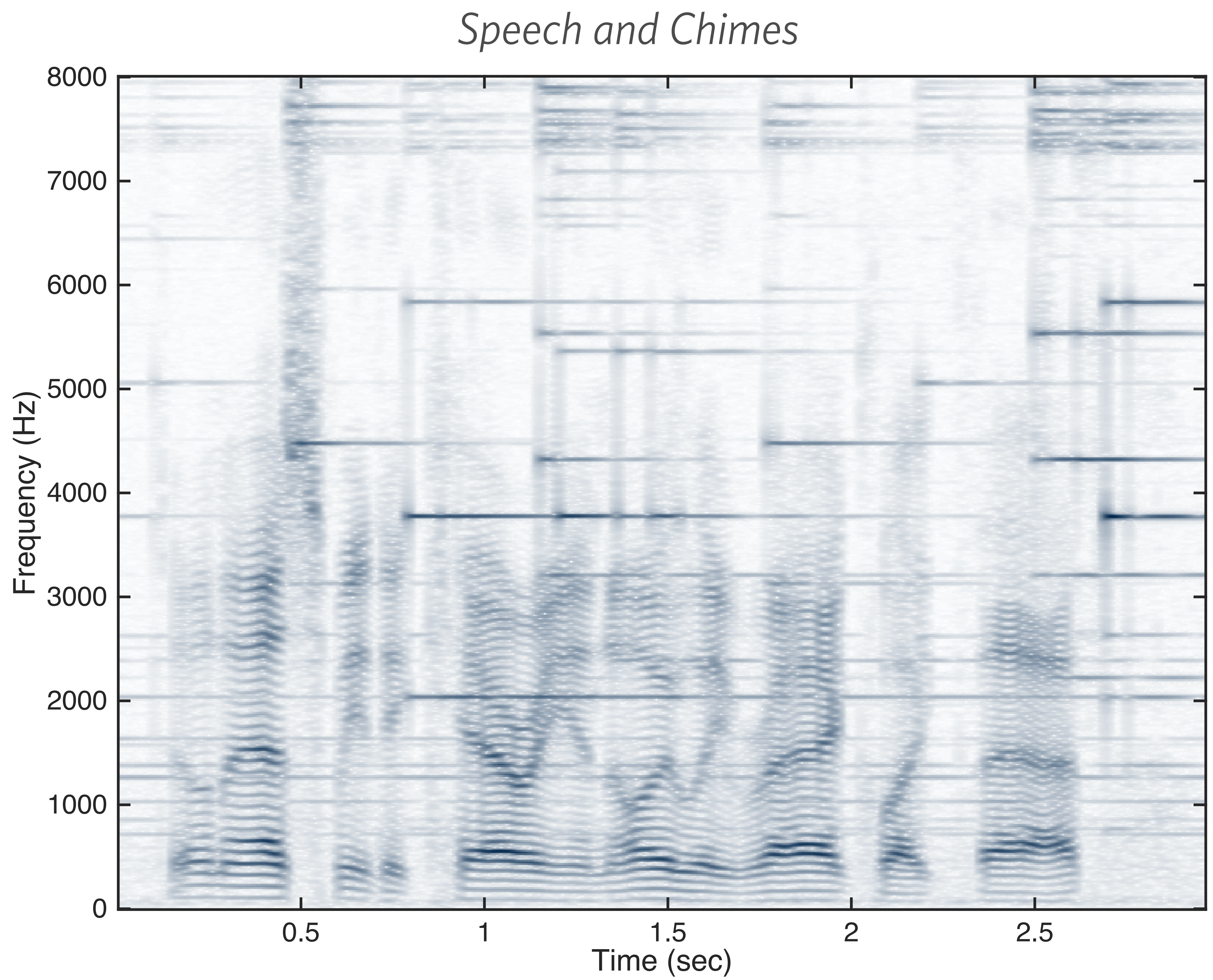


Mixtures of sounds

- Use spectrogram additivity
  - combine models to explain mixture

$$\mathbf{F} = \begin{bmatrix} \underbrace{\mathbf{W}_{chimes}}_{\text{Known/fixed}} & \mathbf{W}_{speech} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{H}_{chimes} \\ \underbrace{\mathbf{H}_{speech}}_{\text{Estimated}} \end{bmatrix}$$

- Estimate only the activations
- The spectral bases claim only parts that they can explain best



Separation

- Recompose sources individually

$$\mathbf{F}_{speech} = \mathbf{W}_{speech} \cdot \mathbf{H}_{speech}$$
$$\mathbf{F}_{chimes} = \mathbf{W}_{chimes} \cdot \mathbf{H}_{chimes}$$

- And convert spectrograms to time domain
  - Use the original phase of the mixture
  - This is effectively a *soft mask*
- Sounds have to have different **W**'s!
  - But not dramatically so

Mixture

Extracted speech

Extracted chimes

Speech mixture

Extracted speaker 1

Extracted speaker 2

Separation with some unknown sounds

- Same as before, use only one model:

$$\mathbf{F} = \begin{bmatrix} \underbrace{\mathbf{W}_{known}}_{\text{Known/fixed}} & \mathbf{W}_{unknown} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{H}_{known} \\ \mathbf{H}_{unknown} \end{bmatrix}$$

Known/fixed

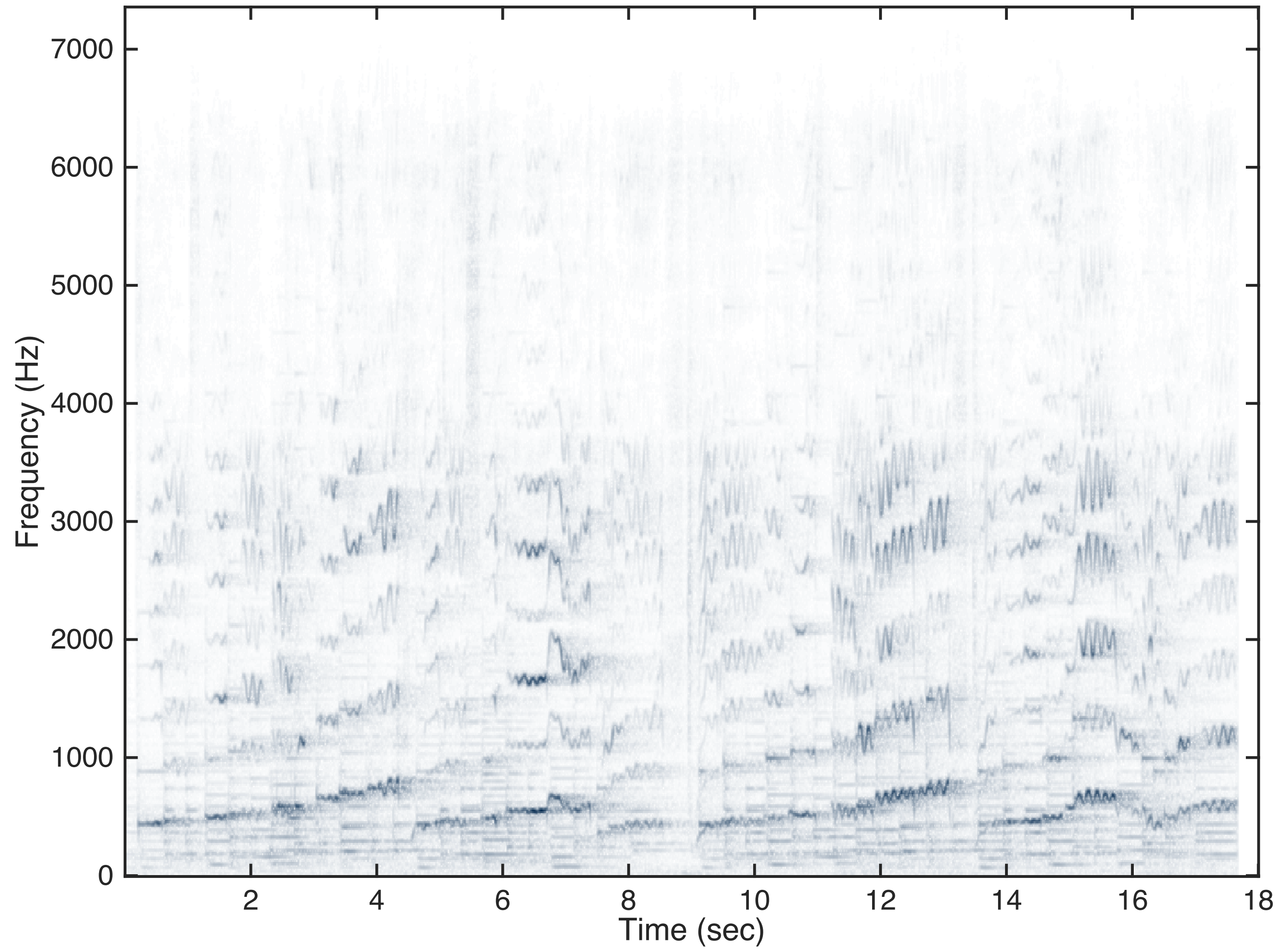
Estimated

- Learn weights and unknown bases
  - Unknown bases converge to the unknown parts in the mixture

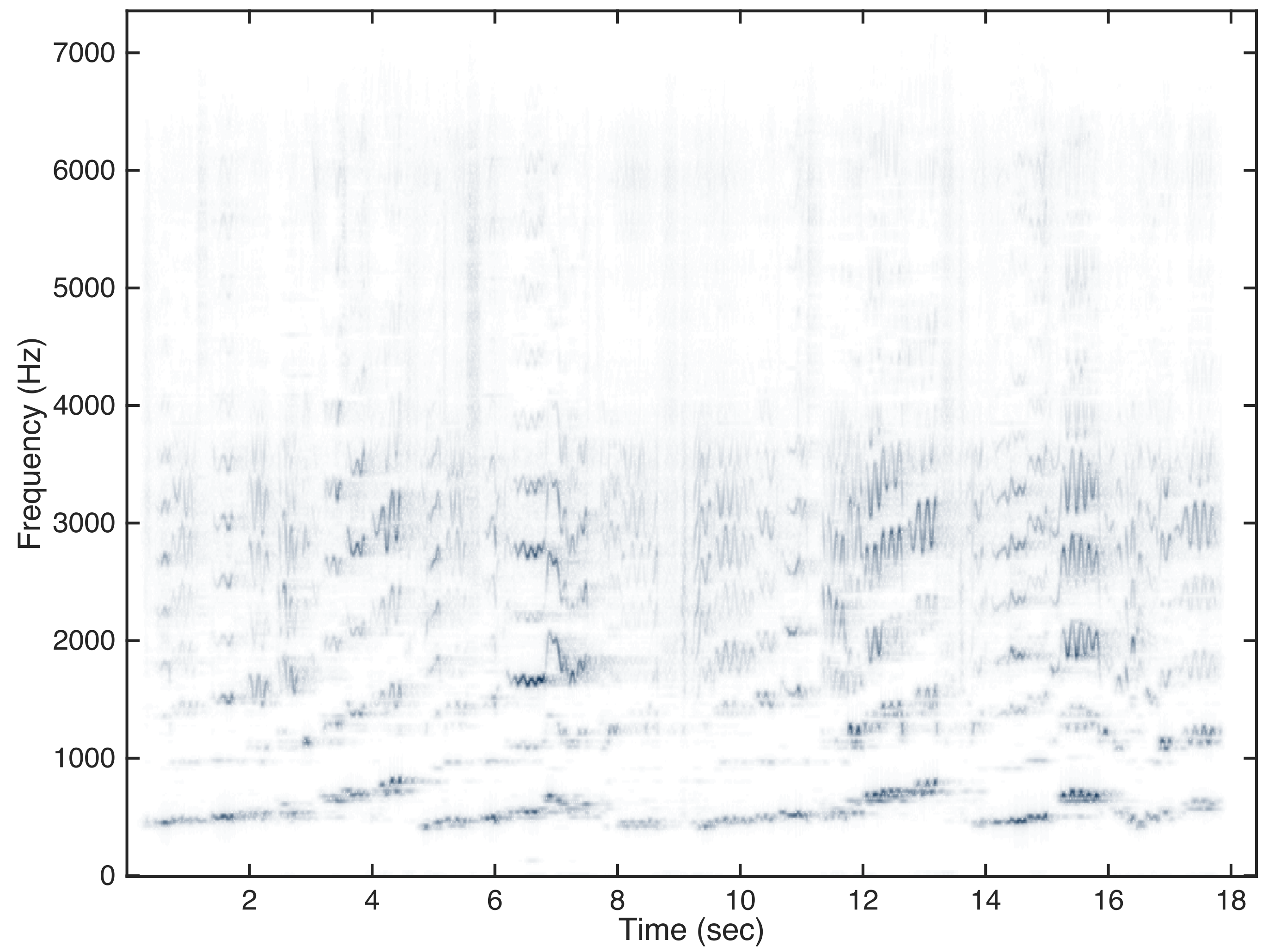
Soprano & Piano

Extracted soprano

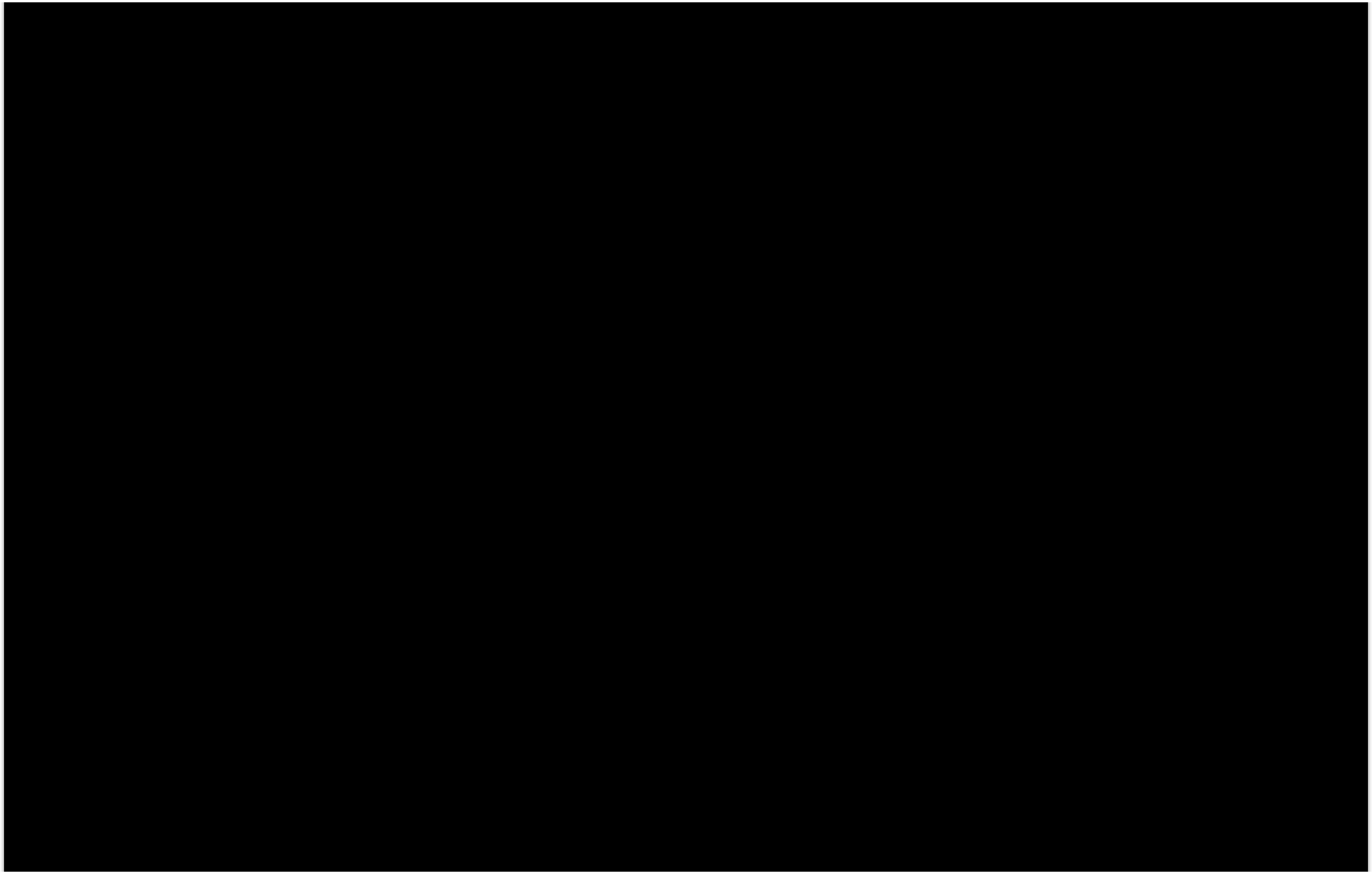
Soprano+Piano



Extracted soprano



How robust is this?

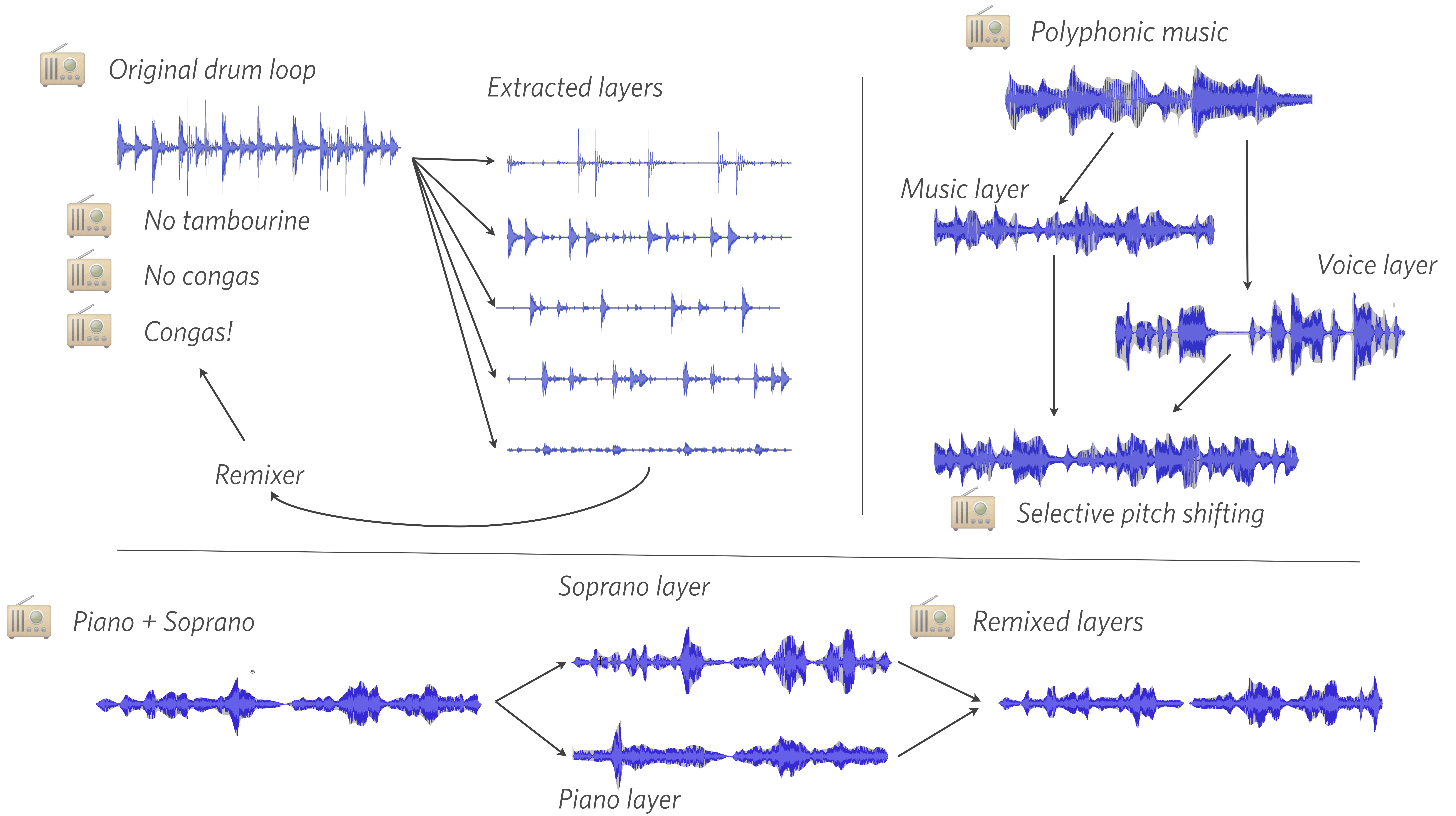


Many more applications

- Non-Negative models have been pretty successful when it comes to processing magnitude spectrograms
  - Very effective when dealing with mixtures of sounds!

- Some applications that are out there
  - Sound detection from mixtures, polyphonic music transcription, missing data restoration, remixing tools, multi-channel enhancements, dereverberation, compression models, …

Audio layer editing



Video Content Analysis

- Detecting sounds in mixtures
  - Measure activation of known dictionaries to estimate presence

Input

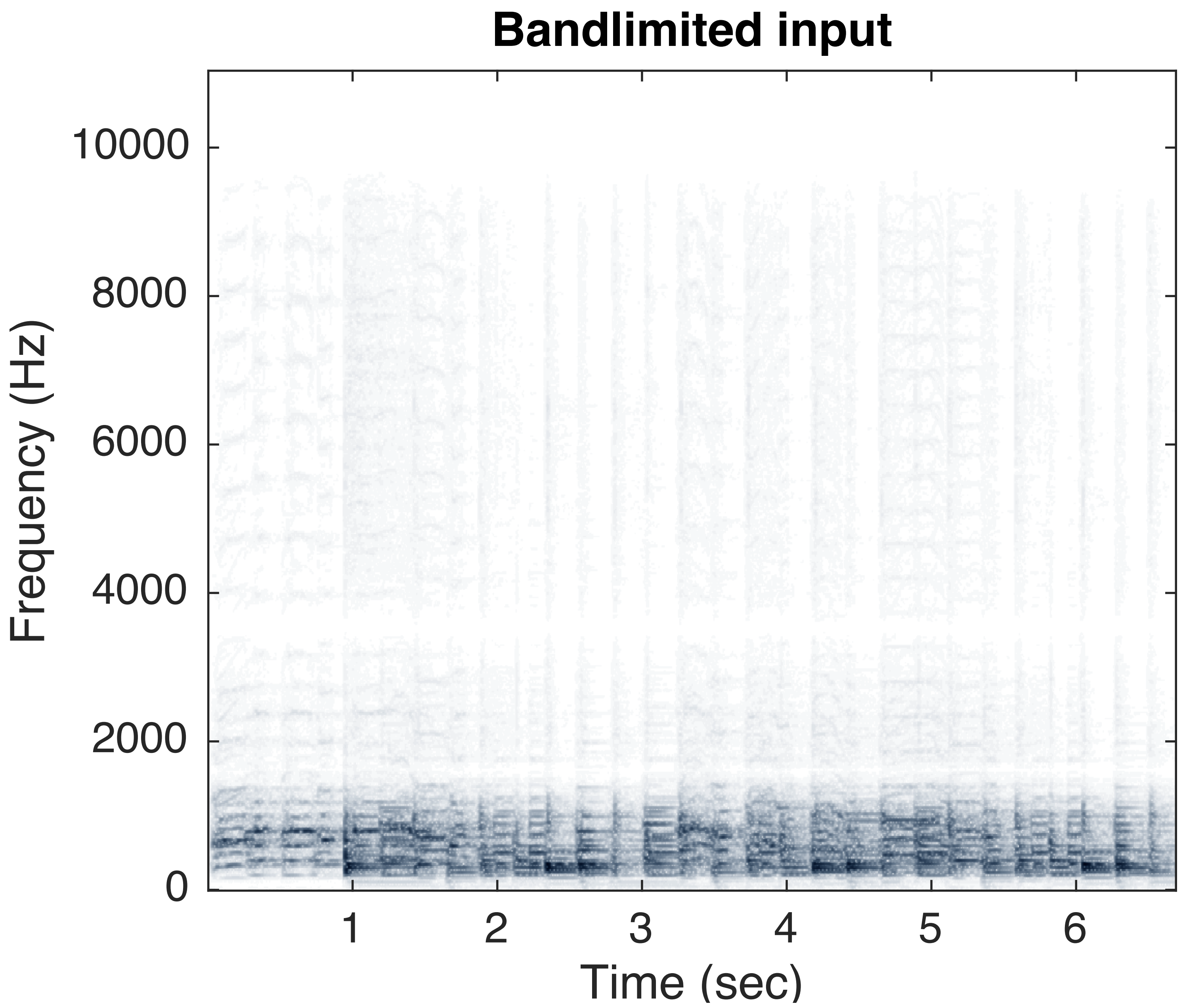
Pre-trained models

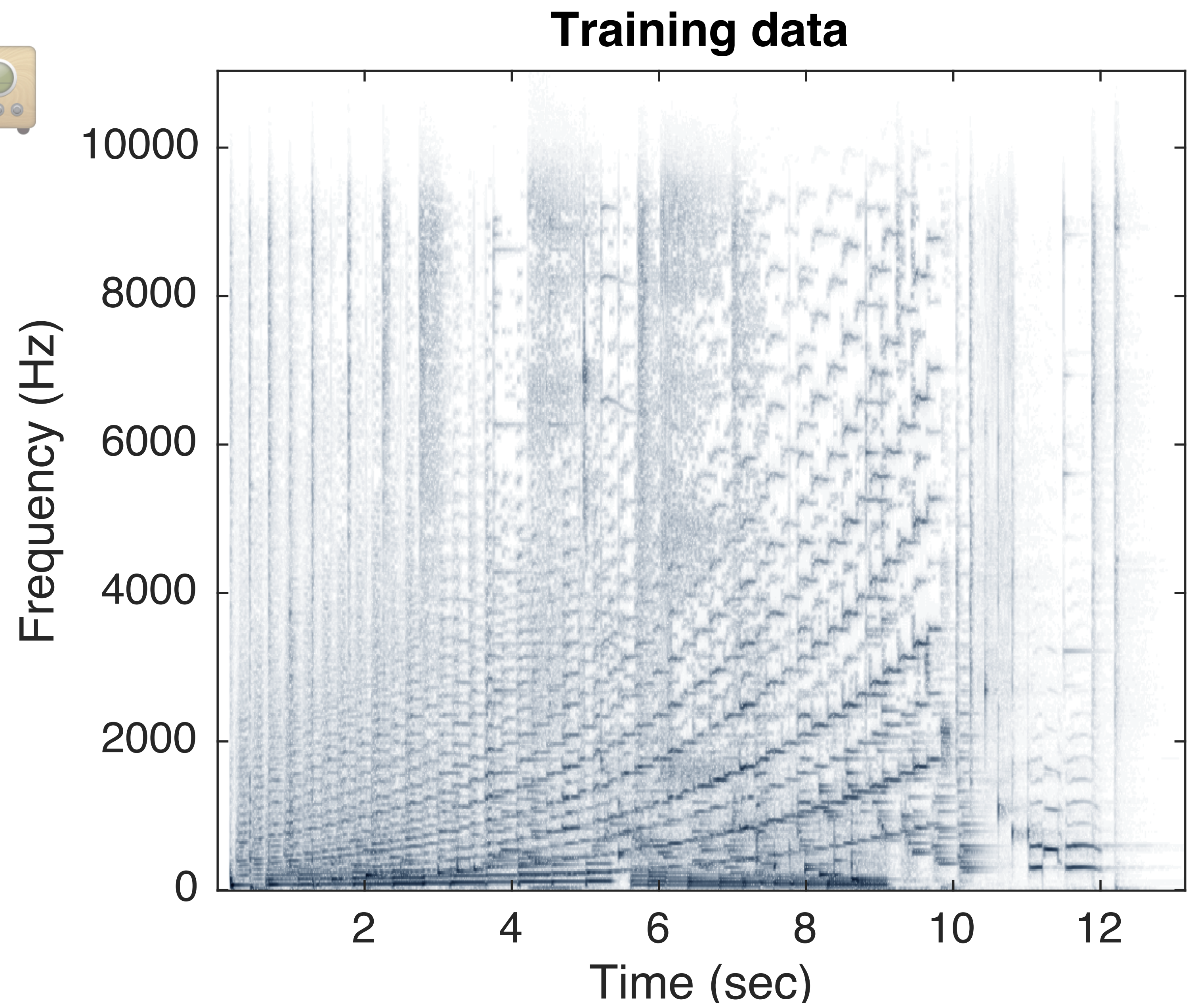
$$\mathbf{F} \approx \begin{bmatrix} \mathbf{W}_1 & \mathbf{W}_2 & \cdots \end{bmatrix} \cdot \begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \\ \cdots \end{bmatrix}$$

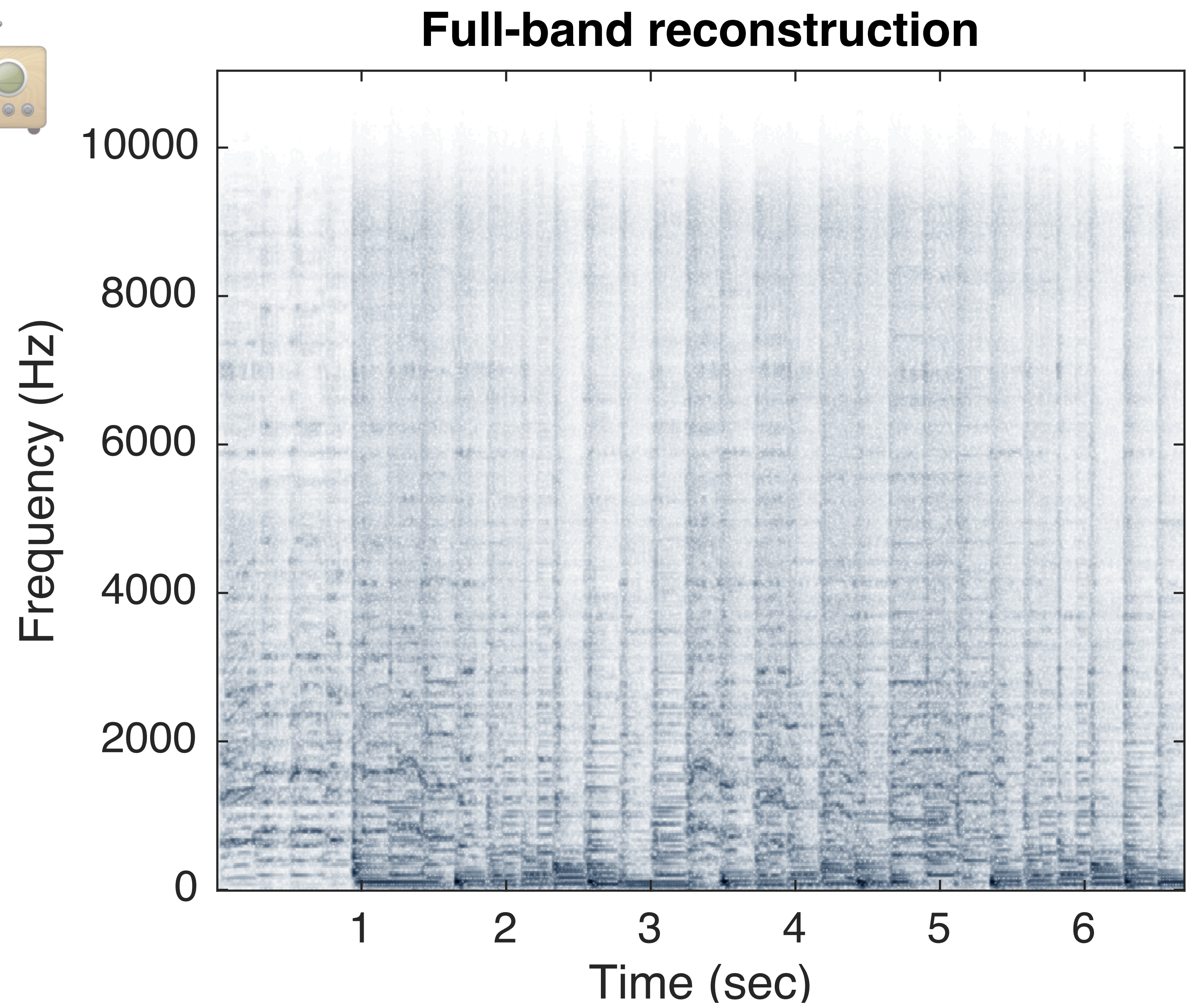
Activations to estimate

Bandwidth Expansion

- Filling in missing data
  - Learn full-band dictionary $\mathbf{W}$ from example sounds
  - Fit $\mathbf{W}$ on input recording using only the available bands
  - Reconstruct input using full-bandwidth bases

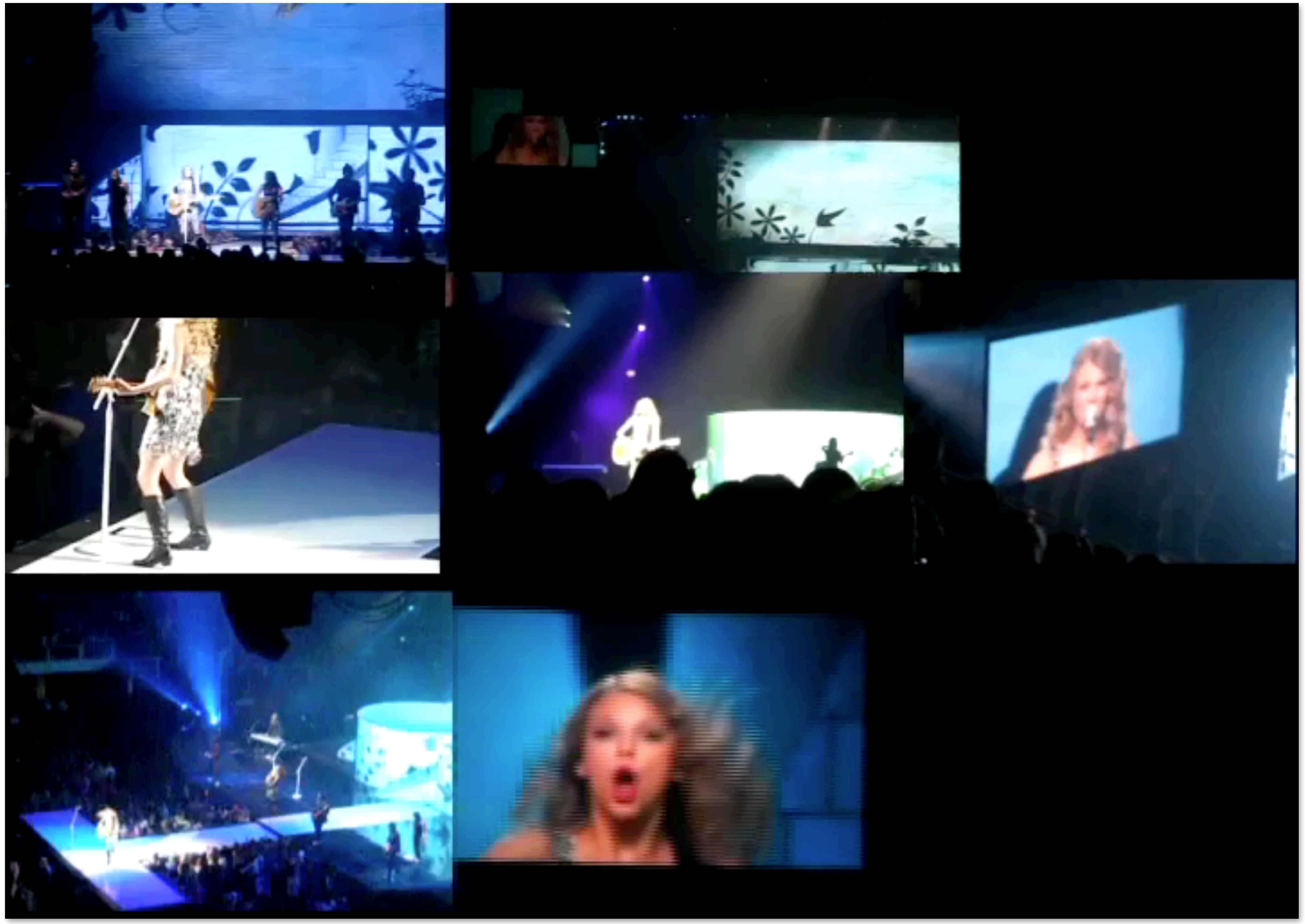






Multi-channel methods

- 700 videos of YouTube
  - Taylor Swift at the (then) San Jose HP Pavillon
    - As dirty as data gets!
- Can we beamform it?
  - Two problems:
    - Sync and combine

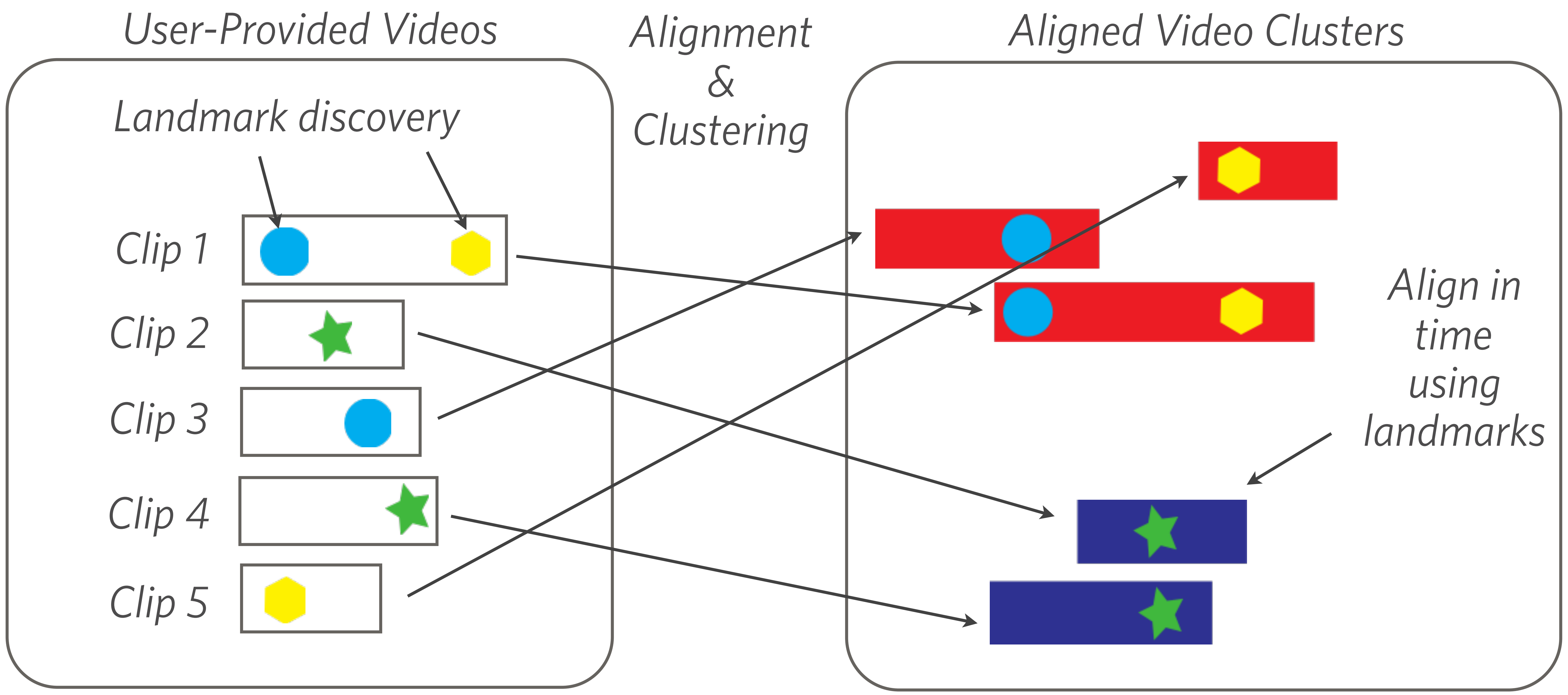


Sync issues

- Super heavy using traditional processing
  - 200 people out of 5,000
  - 4 videos per person = 800 videos
  - 800^2 = 640,000 correlations
  - 2 min per average video = 1,600 minutes = 26 hours of footage
  - 44,100 samples per sec = 5,292,000 samples per clip
  - 1 correlation = 28 Trillion = 28 TeraFLOPS
  - Total cost = 17 Quintillion FLOPS = 17 ExaFLOPS!!

Landmark-based sync

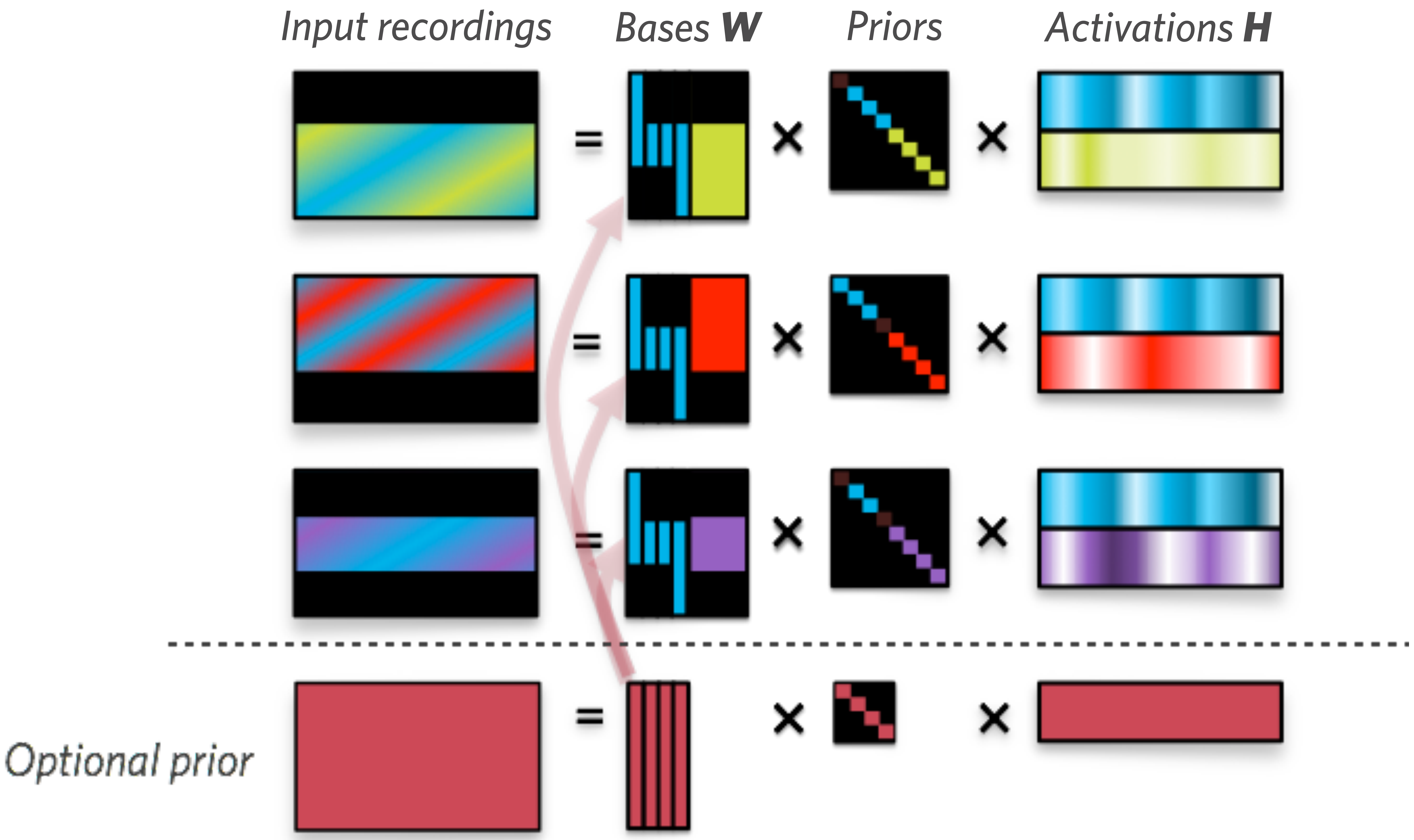
- Forget correlations
  - Hash spectral peaks and match their locations across recordings
    - ~30sec on my laptop!



Does pretty well

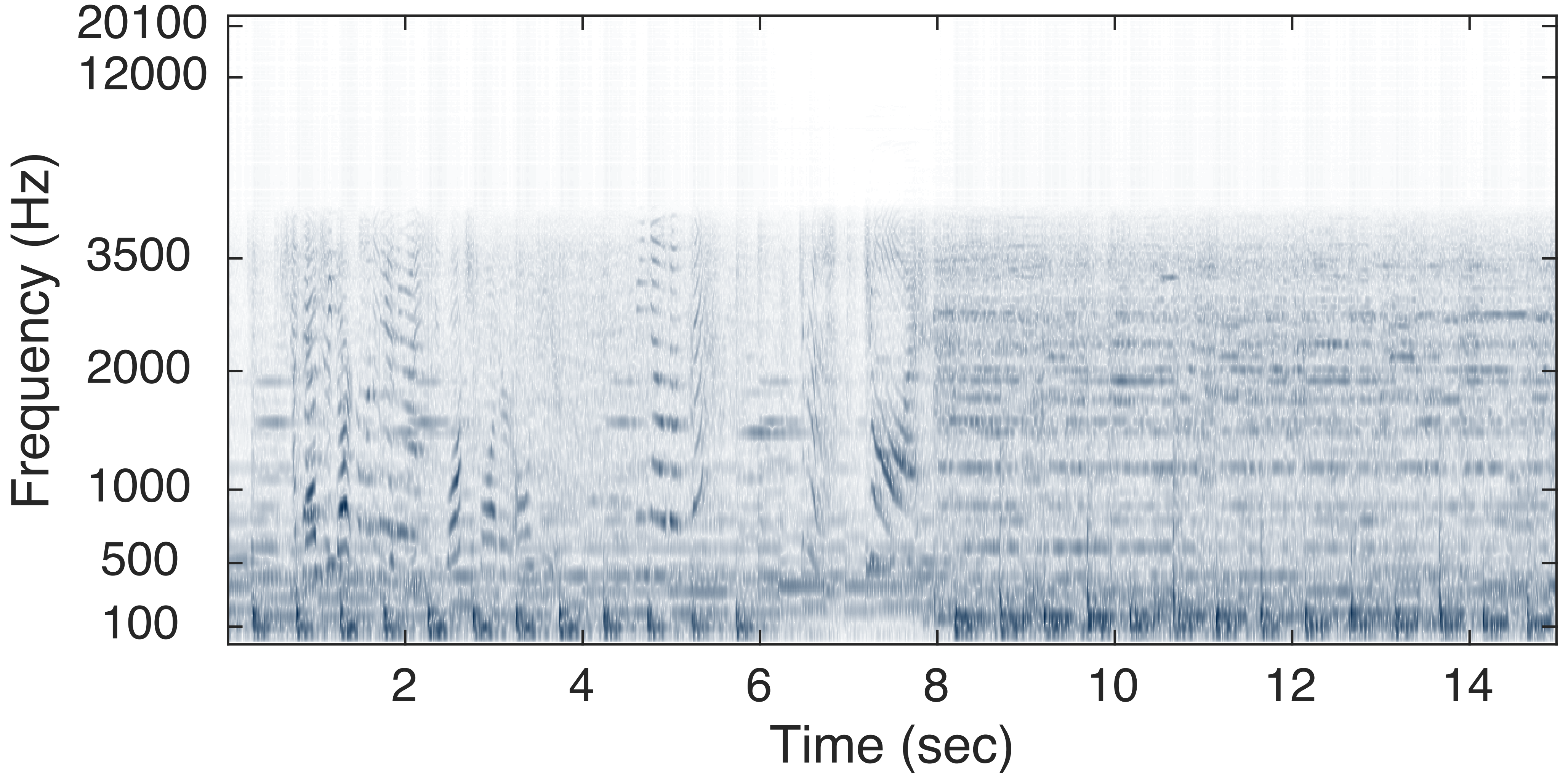
Using co-factorization

- What if all the recordings are of poor quality? (they are!)
  - Can we combine them to get a better reconstruction?

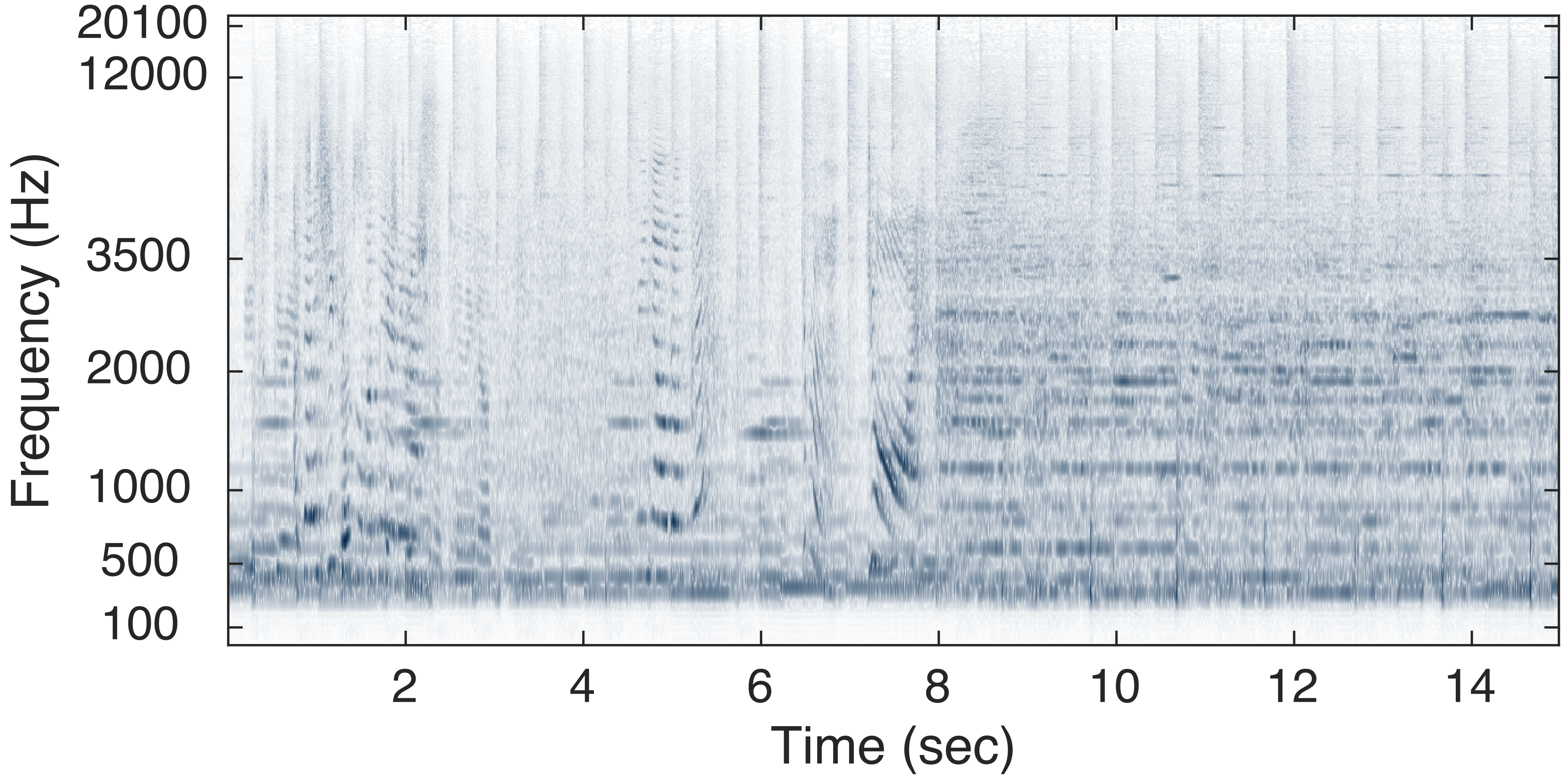


Example: Yuki – Joy, Live

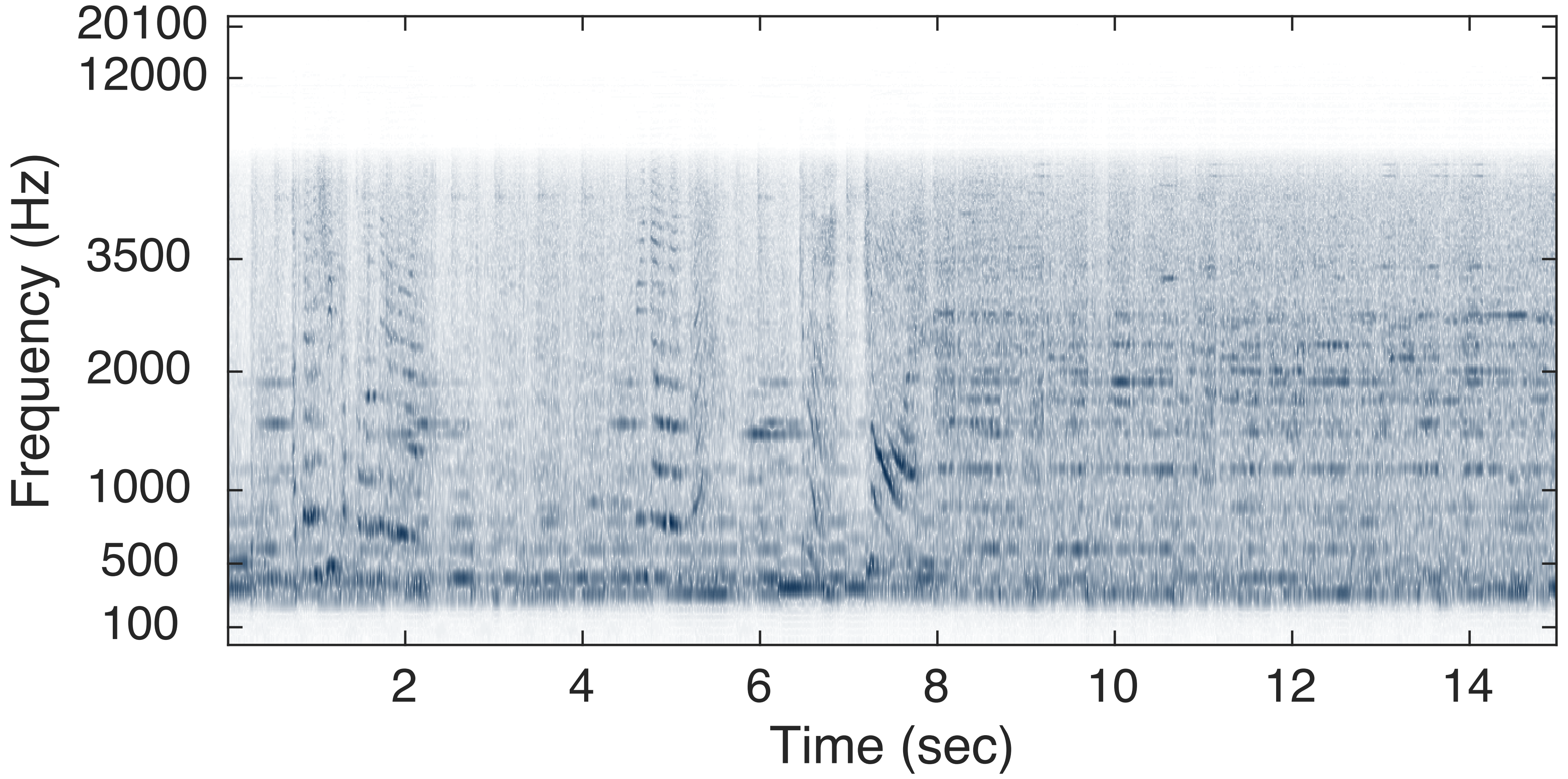
Interference + LPF



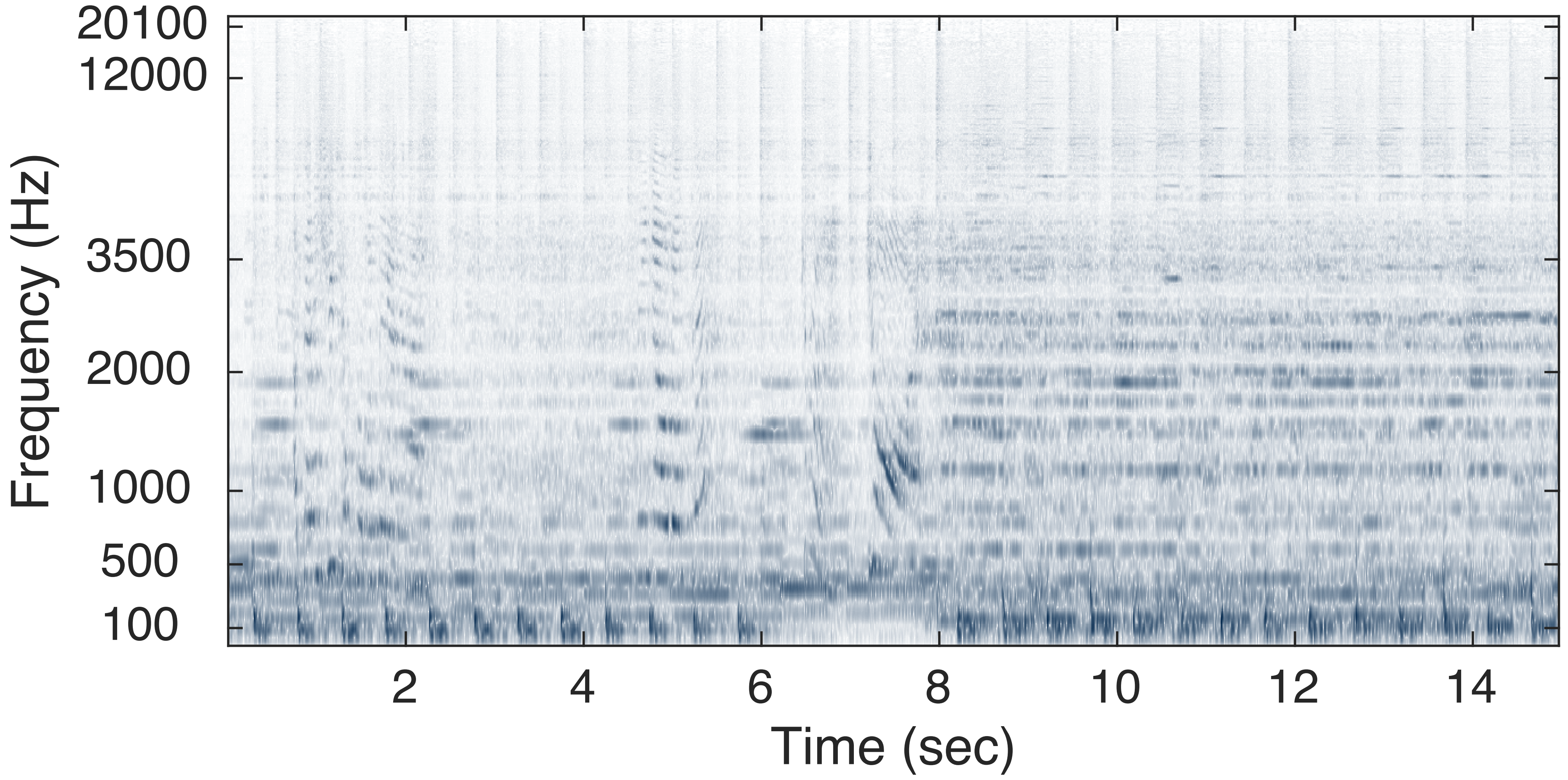
Interference + HPF



Distortion + BPF



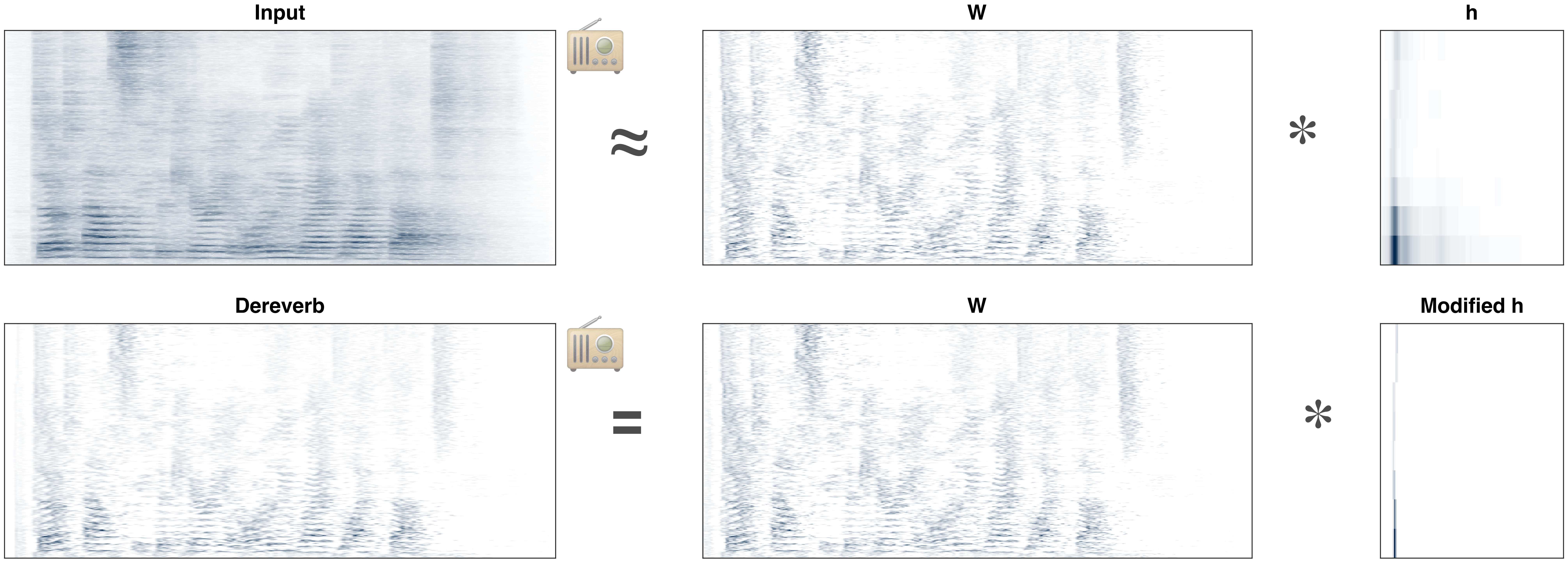
Reconstruction



Convolutional form of factorization

- Convolution is a product, we can use that instead
  - Allows us to deconvolve in the magnitude domain:

$$\mathbf{F} \approx \mathbf{W} * \mathbf{h}$$



Many more models for different jobs

- Online formulations
  - Facilitate real-time deployment
    -
- Universal Speaker Model
  - Doesn't require exact model for a speaker
- HMM / Dynamical models
  - Allow concurrent speech ASR

But …

- Matrix factorizations are not for the faint of heart
  - Heavy computational requirements (large matrix multiplies)
    - Might be ok for desktops, not for smaller devices
- Is there a way to avoid the costly weights estimation?
  - Can the runtime processing be a non-iterative process?
- Let's explore that option

Towards a more direct method

- Non-negative models were generative models
  - We modeled the data, and the rest was a side-effect

- We can instead explicitly aim for a task
  - Forget the models, teach a system to perform the needed task

Noisy autoencoders for enhancement

- Use a neural net with positive-only outputs

$$\mathbf{s}_t = f\left(\mathbf{W} \cdot \mathbf{m}_t + \mathbf{b}\right)$$

Output clean spectra

Positive-output activation

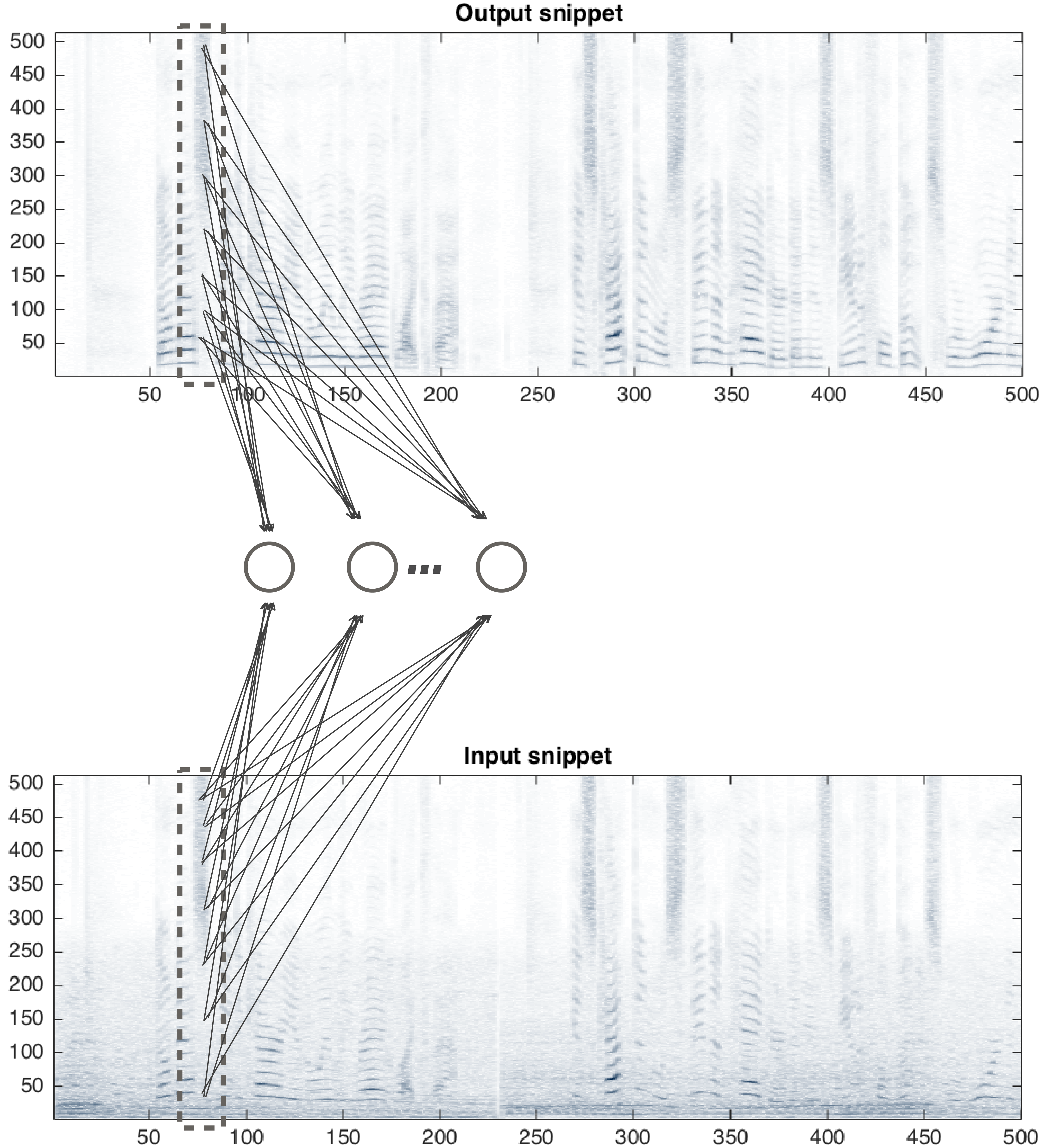
Input noisy spectra

- Train it to predict clean spectra from noisy spectra
  - We can easily do this by making artificial mixtures
  - Advantage: solves the problem directly
  - We can also use other flavors (multilayer, recurrent, convnet, ...)

Toy example – Training

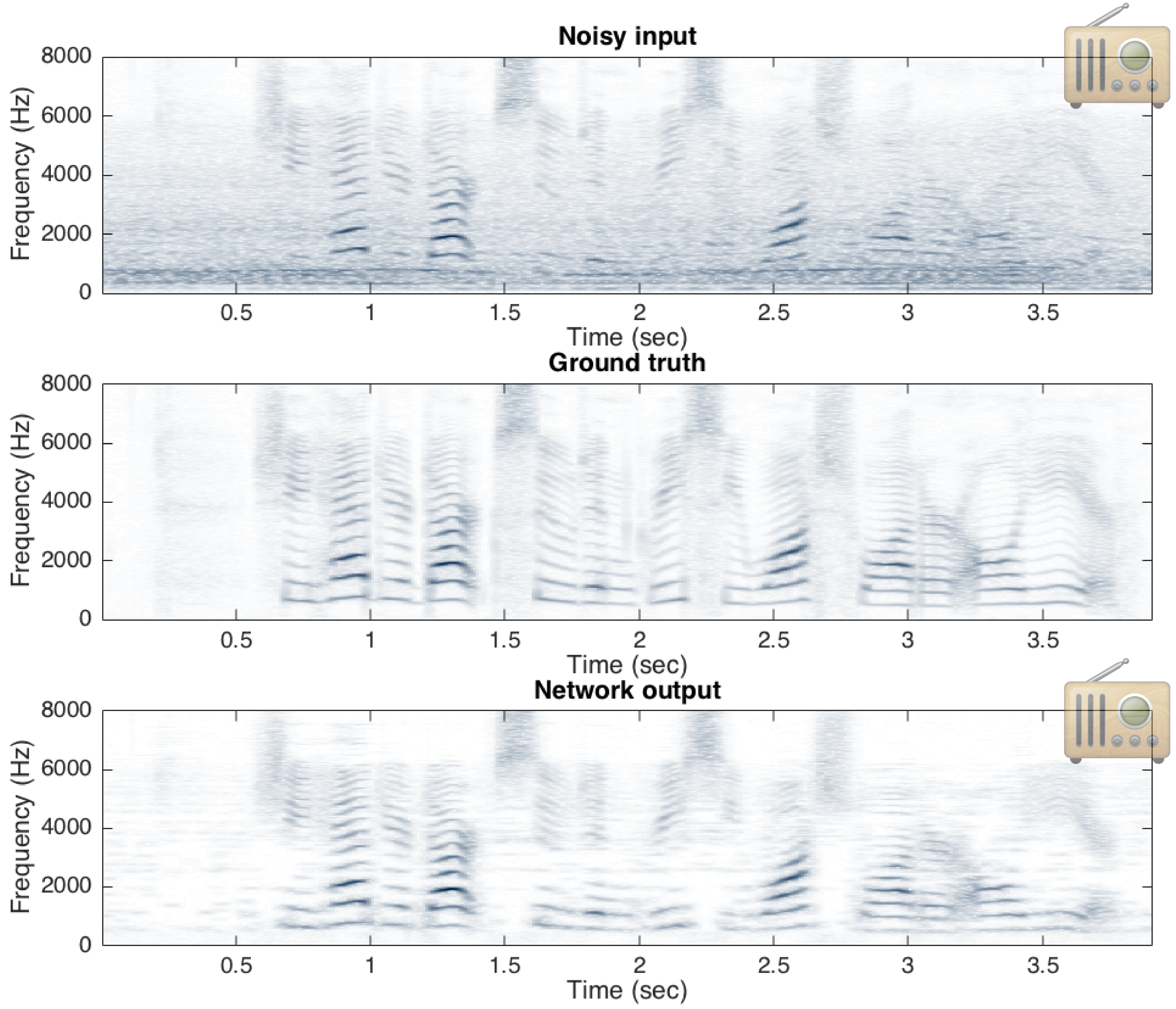
- Trained on 30sec inputs
  - Speech + street noise
  - Known speaker
  - Takes 30sec to train
    - on a laptop (2-3sec with GPU)

- Parameters
  - 1024pt spectra
  - 1 hidden layer, 100 nodes
  - Leaky ReLU activations



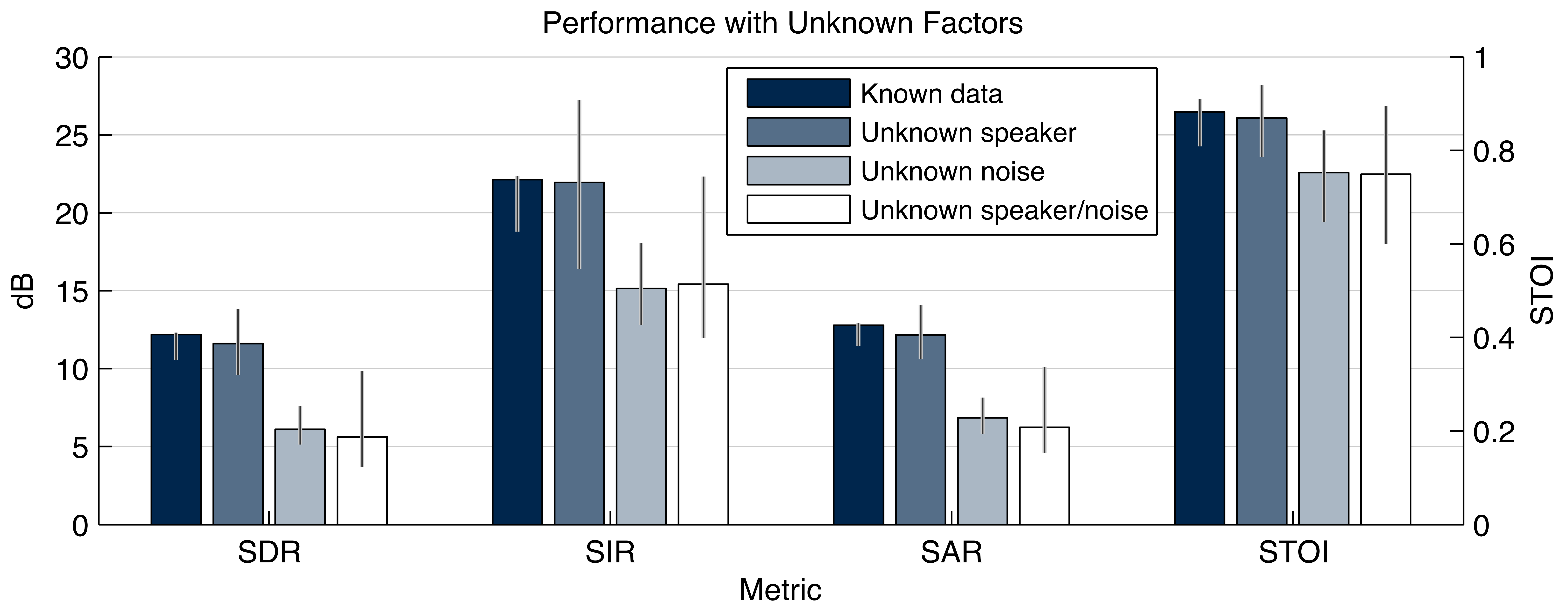
Toy example – Runtime

- Very lightweight process
  - ~300x real-time
    - 0.01sec in this case
- Strong performance
  - **SDR**: 12.7, **SIR**: 23.2, **SAR**: 13.1
  - **PEAQ**: -2.04, **PESQ**: 0.71
  - **STOI**: 0.86



What about unknown sounds?

- The more you know the better (no surprise here)



Thinning down the computations

- Running these floating-point operations is costly
  - Complex FP hardware $\longrightarrow$ more power consumption and cost
- "Binarizing" the feedforward pass
  - Key idea: replace FP operations with bits operations
- Problem: How do we map the operations?

Mapping to binary

- Typical unit operation:

$$y = \tanh\left(\sum_i w_i x_i\right)$$

- Binary re-interpretation:

$$y = \sum_i w_i \otimes x_i > \frac{N}{2}$$

Real products

	$x > 0$	$x < 0$
$w > 0$	$w\,x > 0$	$w\,x < 0$
$w < 0$	$w\,x < 0$	$w\,x > 0$

Non-linearity result

$\Sigma \gg 0$	$y \rightarrow +1$
$\Sigma \ll 0$	$y \rightarrow -1$

Binary XNOR

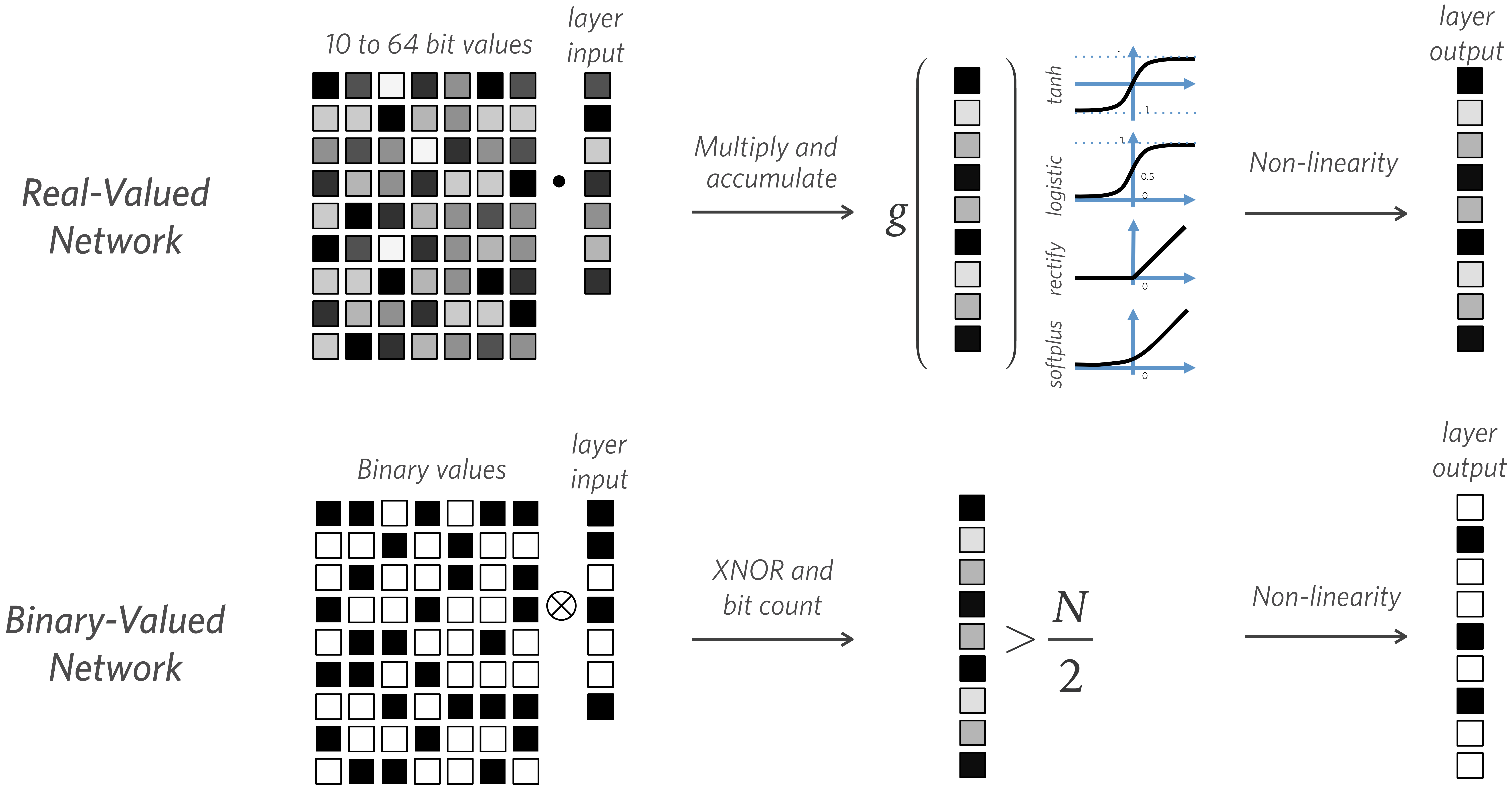
	$x = 0$	$x = 1$
$w = 0$	$w \otimes x = 1$	$w \otimes x = 0$
$w = 1$	$w \otimes x = 0$	$w \otimes x = 1$

Comparison result

$\Sigma > N/2$	$y = 1$
$\Sigma < N/2$	$y = 0$

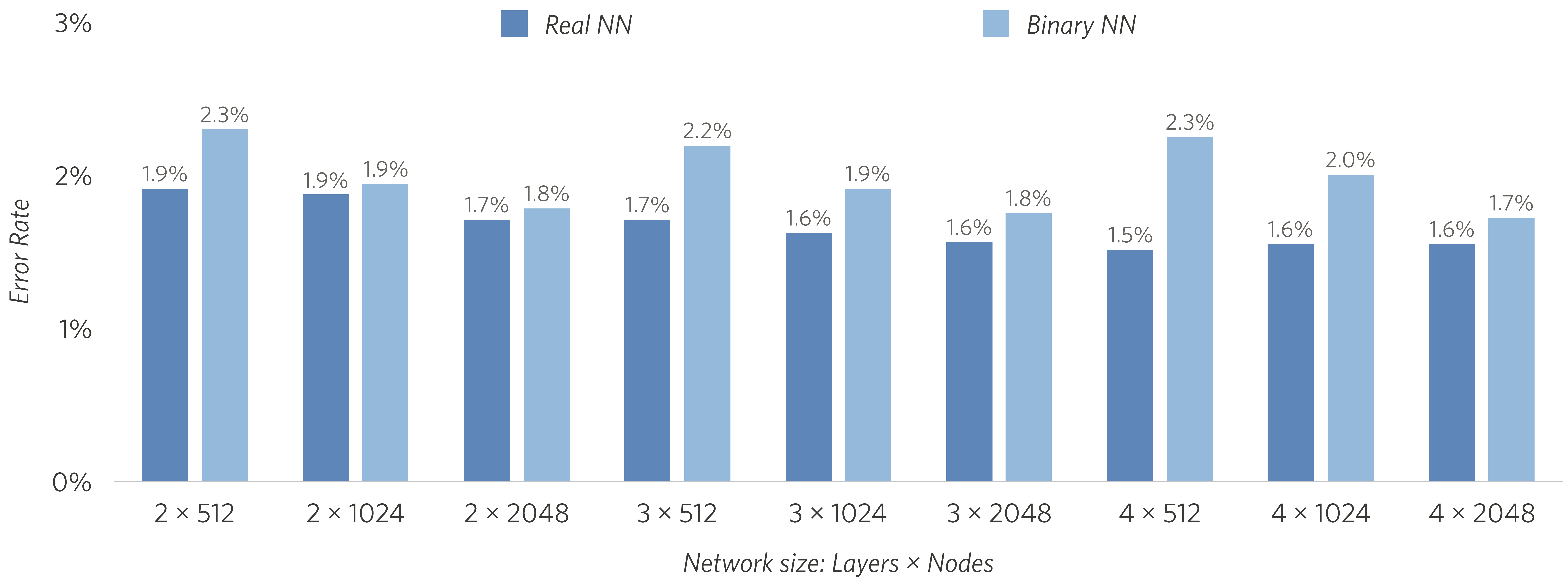
- Works fine as long as the w's are not close to zero
  - Hence we also maximize w's and apply a tanh to saturate them

Comparison of forward pass



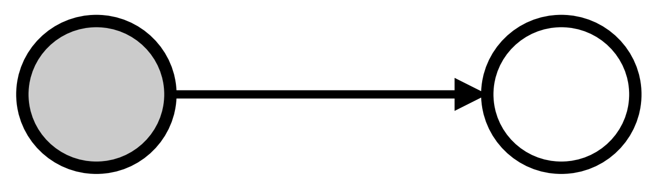
Does this work?

- Comparison using NMIST dataset (digit recognition)

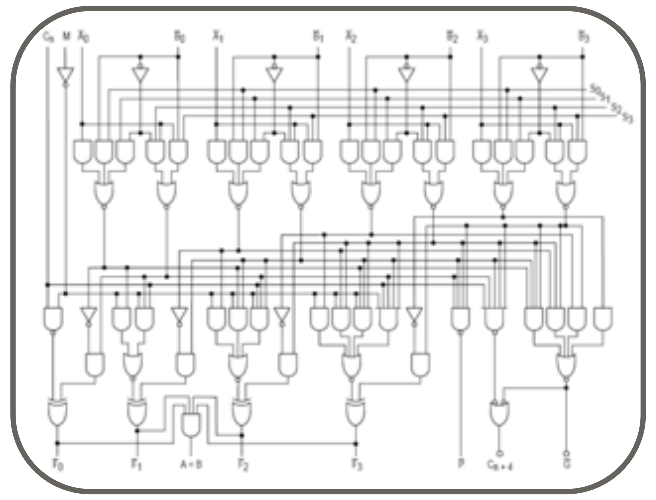


Hardware comparison

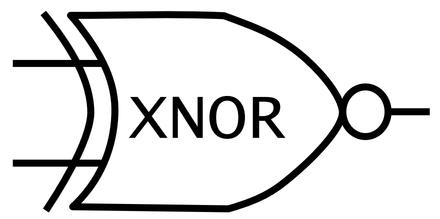
One connection



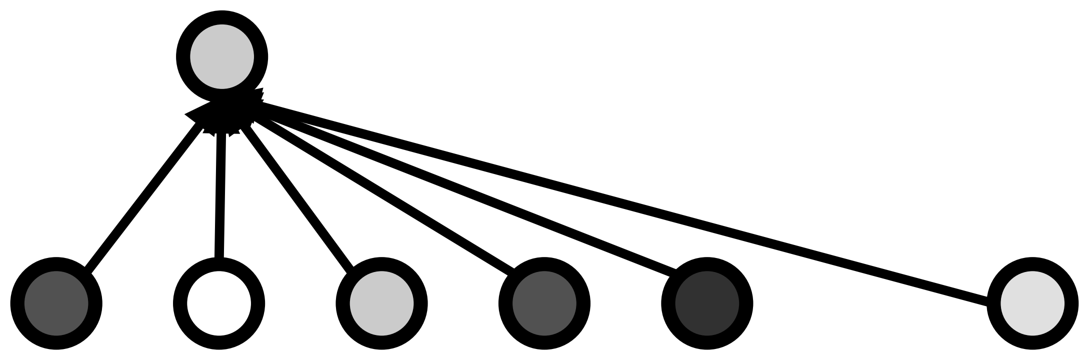
32 bit real multiplication



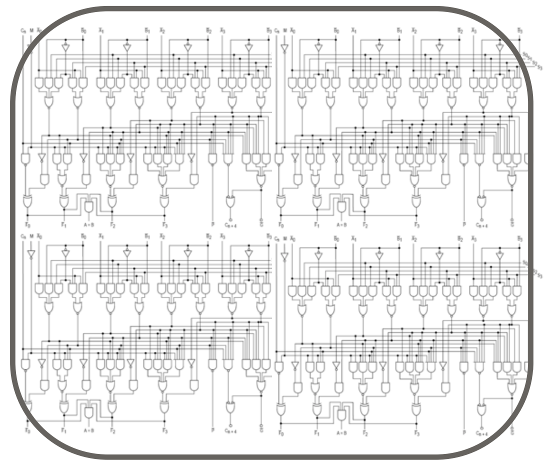
1 XNOR, 1 bit



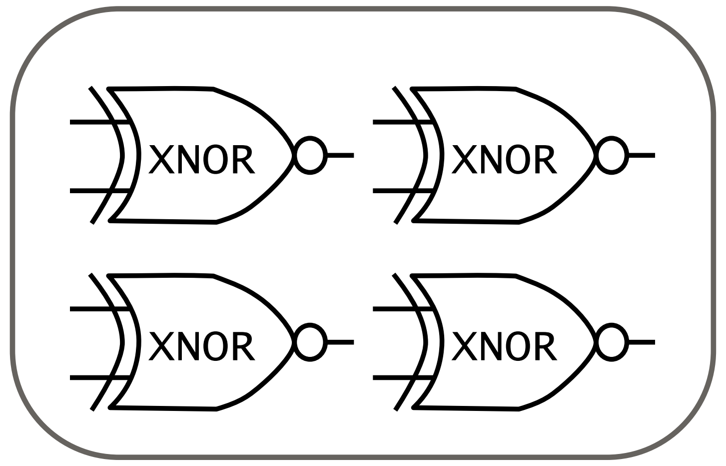
One node



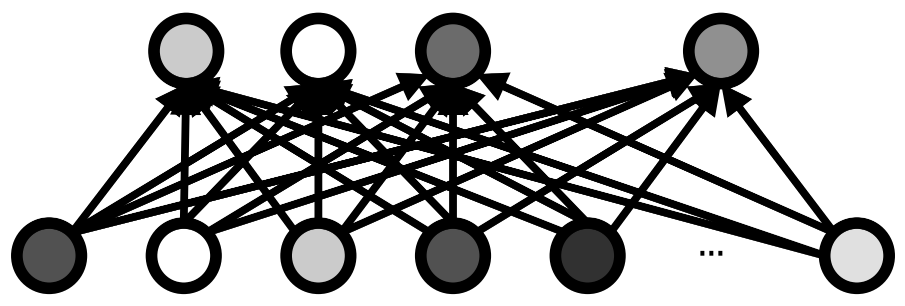
1K Multiply-adds and an FP function, 32K bits



1K XNORs + 1 pop count, 1K bits



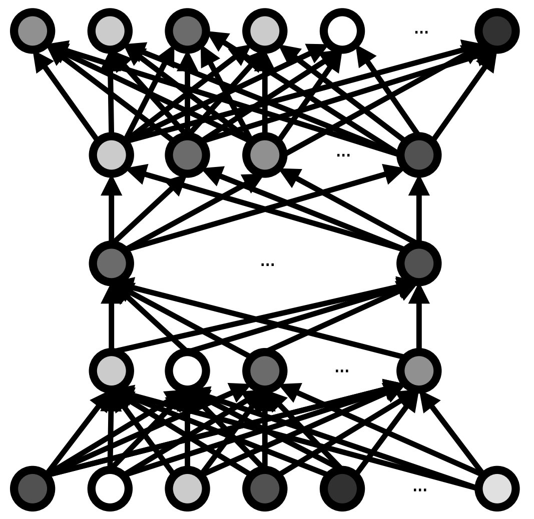
One layer



2 MFLOPS, 4 MBytes

2 Mbit-Ops, 0.125 MBytes

One network



8 MFLOPS, 16 MBytes

8 Mbit-Ops, 0.5 MBytes

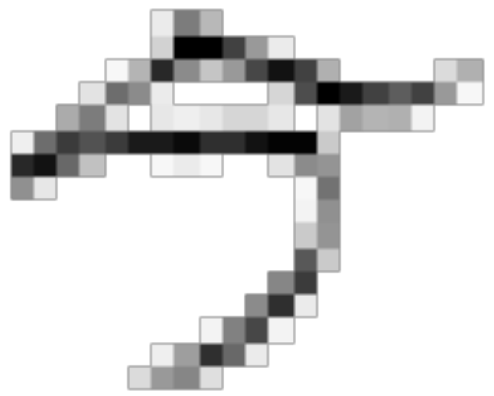
Estimated hardware comparison (per node)

	32bit float	16bit int	Binary
Area (μm^2)	6,000	1,000	100
Power (μW)	2,000	250	20

Under-the-rug issues

- Currently this model is for runtime only
  - Fortunately, learning is a one-time offline process
- Data needs to be in a binary format
  - Not a major problem, but requires additional thinking
    - We can simply quantize, or use hashing methods

Input data



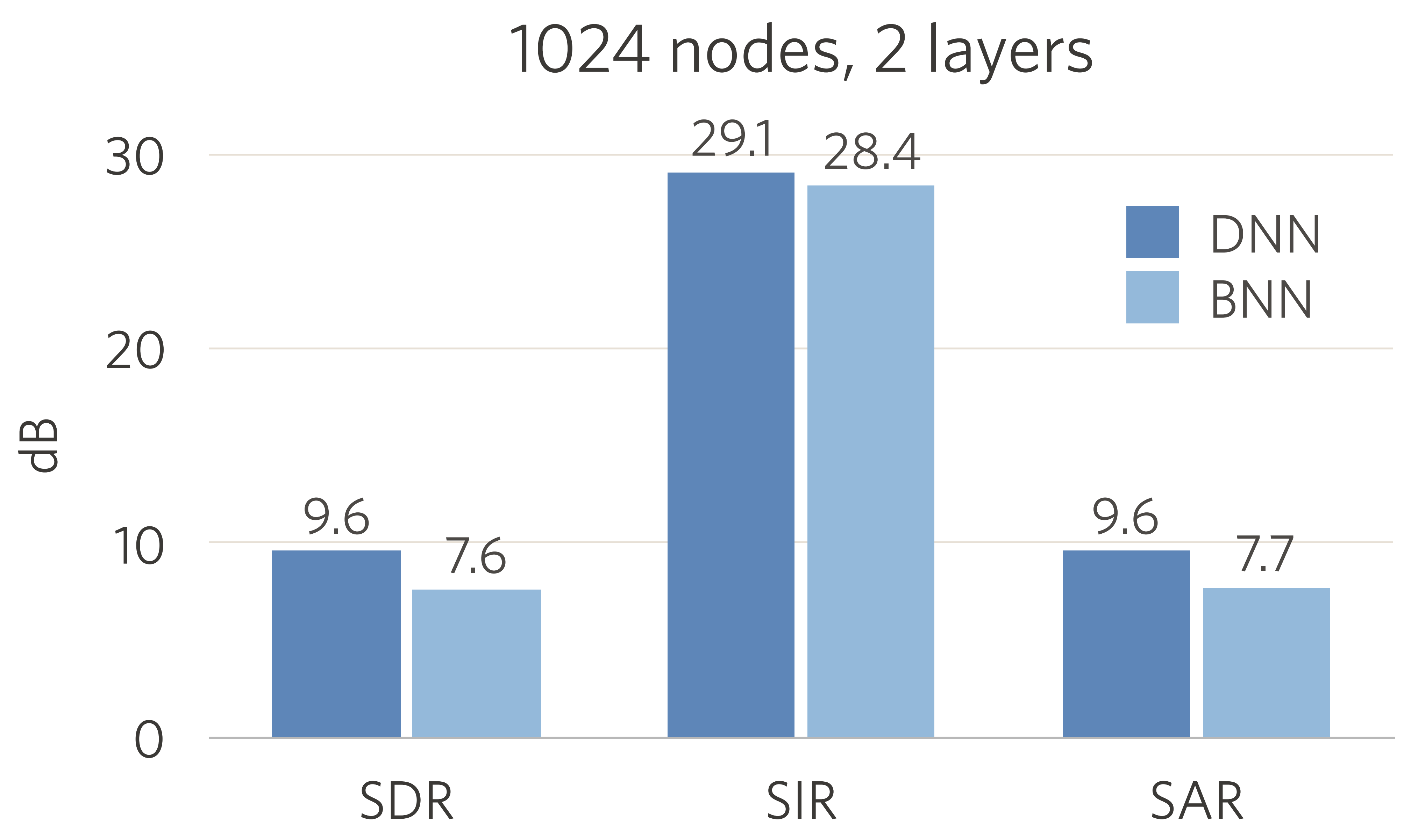
Quantization

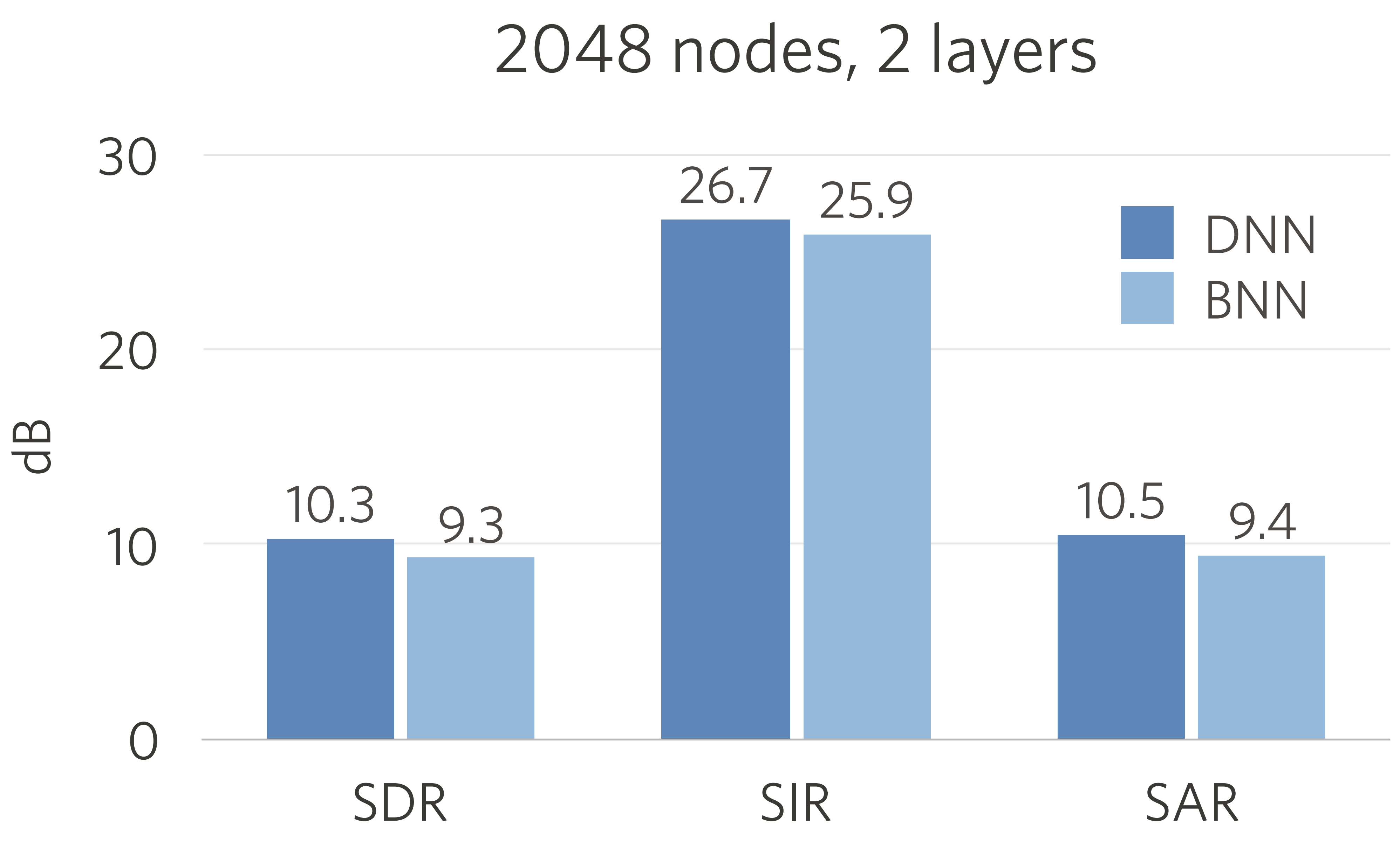
Hash representation

```
10011100001
11110010010
00100010101
00001001010
11000101101
```

What about our problem at hand?

- Slightly different structure to accommodate binary data
  - Inputs are quantized noisy spectra (4-bits per coefficient)
  - Output is a binary mask





So what have we gained from ML?

- We can improve on array methods
  - Simple models, better performance, less finicky setup
- We can explain mixtures intuitively
  - Allows us to manipulate sound in easier ways
- We can simplify processing complexity
  - Neural net enhancers using very simple hardware

In conclusion

- There is more to DSP than textbook approaches
  - Let's stop beating dead horses …
- Lots of neat ideas we can take from machine learning
  - Not that different from the DSP way of thinking
    - But definitely outside our comfort zone

Thanks to you, and to:

- Collaborators:
  - Johannes Traa (now at Lyric Labs)
  - Minje Kim (now at U Indiana)
  - Gautham Mysore (now at Adobe)
  - Madhu Shashanka (now at Charles Schwab)
  - Nicholas Bryan (now at Apple)
  - Bhiksha Raj (now at CMU)

- Funding support:
  - NSF, Adobe Research, MERL, Sony, Analog Devices, Intel

Bibliography

- Arrays
  - Traa, J. and P. Smaragdis. 2014. *Multichannel Source Separation and tracking with RANSAC and Directional Statistics*, in IEEE Transactions on Audio, Speech and Language Processing. [PDF]
  - Traa, J. and P. Smaragdis. 2014. *Multiple Speaker Tracking with the Factorial von Mises-Fisher Filter*, in Proceedings of the IEEE Machine Learning for Signal Processing Workshop [PDF]
- Non-Negative Modeling
  - Virtanen, T., J. Gemmeke, B. Raj and P. Smaragdis. 2014. *Compositional models for audio processing*, in IEEE Signal Processing Magazine [PDF]
  - Smaragdis, P., C. Fevotte, G. Mysore, N. Mohammadiha, M. Hoffman 2014. *Static and Dynamic Source Separation Using Nonnegative Factorizations: A unified view*, in IEEE Signal Processing Magazine [PDF]
  - Kim, M., and P. Smaragdis. 2014. Collaborative Audio Enhancement: Crowdsourced Audio Recording, in Neural Information Processing Systems (NIPS) 2014, Workshop on CrowdSourcing and Machine Learning. Montreal, Canada. [PDF]
- Neural Nets
  - Liu, D., P. Smaragdis, M. Kim. 2014. *Experiments on Deep Learning for Speech Denoising*, in Proceedings of the annual conference of the International Speech Communication Association [PDF]
  - Huang, P-S., M. Kim, M. Hasegawa-Johnson, P. Smaragdis. 2014. *Deep Learning for Monaural Speech Separation*, in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing [PDF]
  - Kim, M. and P. Smaragdis, "Bitwise Neural Networks," International Conference on Machine Learning (ICML) Workshop on Resource-Efficient Machine Learning [PDF]