Factorsynth

A tool for analysis/resynthesis based on matrix factorization

Juan José Burred

Independent researcher, Paris, France

Séminaire Recherche et Technologie, IRCAM, Paris

October 19th 2016

Factorsynth: what is it?

- A prototype software tool for sound modification and creation based on matrix factorization.

- Matrix factorization (or matrix decomposition) is a set of linear algebra methods widely used in machine learning and data mining applications.

- In audio applications, matrix factorization techniques are used in compression, source separation and music information retrieval (MIR). Largely unexplored for musical creation.

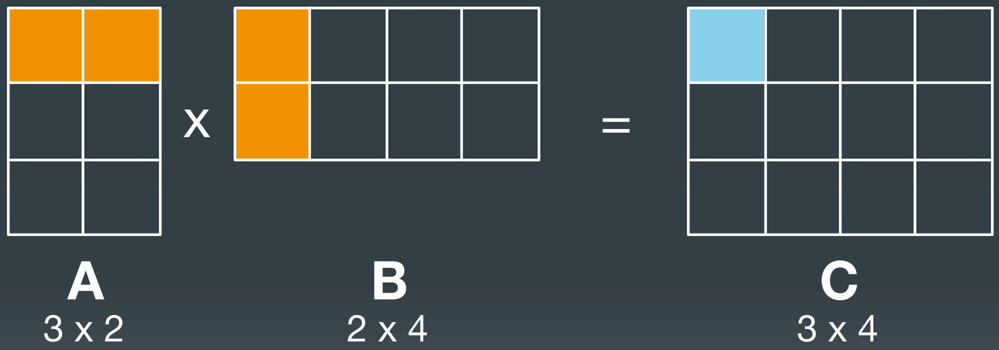
- Factorsynth aims at bringing the ideas and possibilities of matrix factorization to a wider audience of composers and sound designers.

Factorization

- The opposite of multiplication: decomposition into factors
  - 100 = 20 x 5
  - 100 = 25 x 4
  - 100 = 5 x 2 x 10
  - 100 = 3 x 33.3333...
- There is always an infinite number of possible solutions
- The chosen factorization method will depend on the desired form of the output factors
  - E.g.: factorization into prime factors, the basis of cryptography
- Factorization = deconstruction into “building blocks”

Matrix multiplication

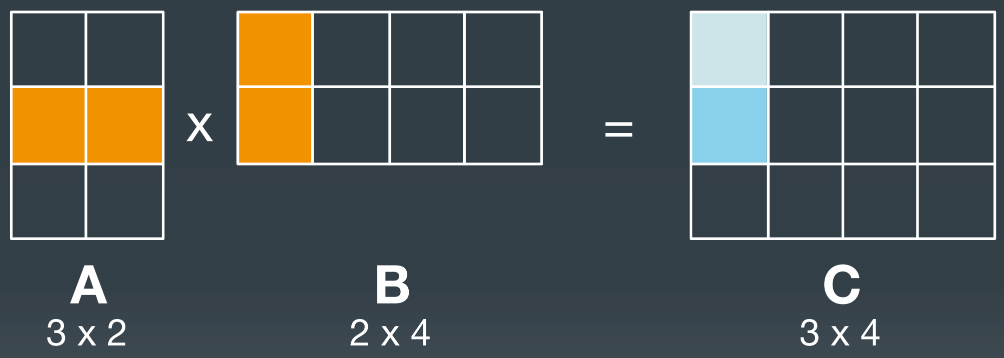
- Factorization can be applied to matrices
- Remember that matrix multiplication is not equal to element-wise multiplication



Matrix multiplication

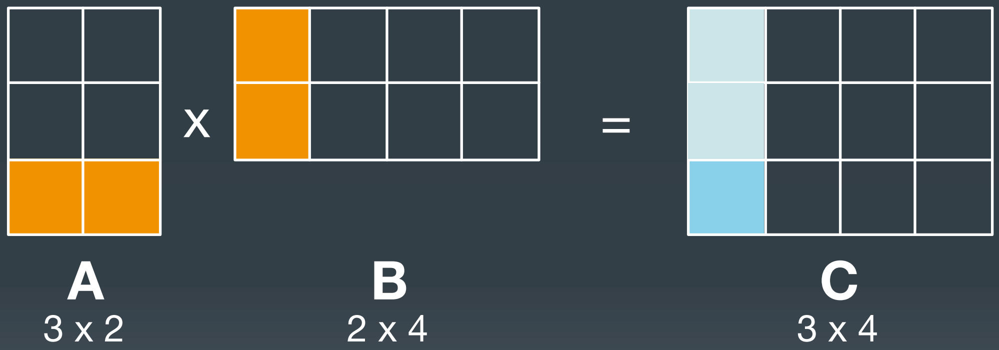
- Factorization can be applied to matrices

- Remember that matrix multiplication is not equal to element-wise multiplication



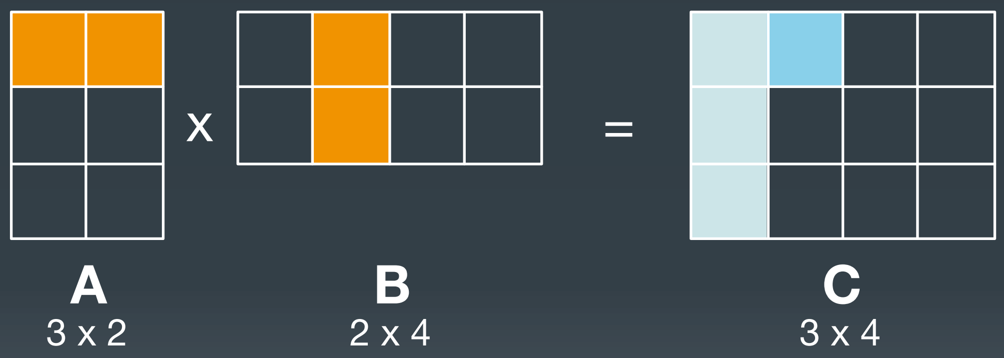
Matrix multiplication

- Factorization can be applied to matrices
- Remember that matrix multiplication is not equal to element-wise multiplication



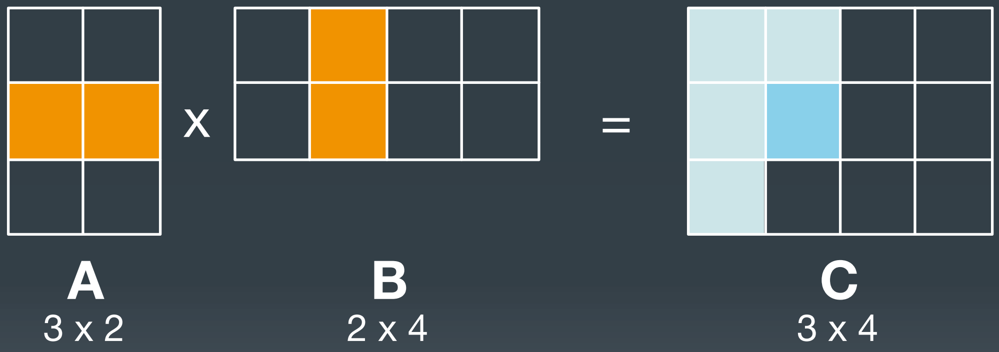
Matrix multiplication

- Factorization can be applied to matrices
- Remember that matrix multiplication is not equal to element-wise multiplication



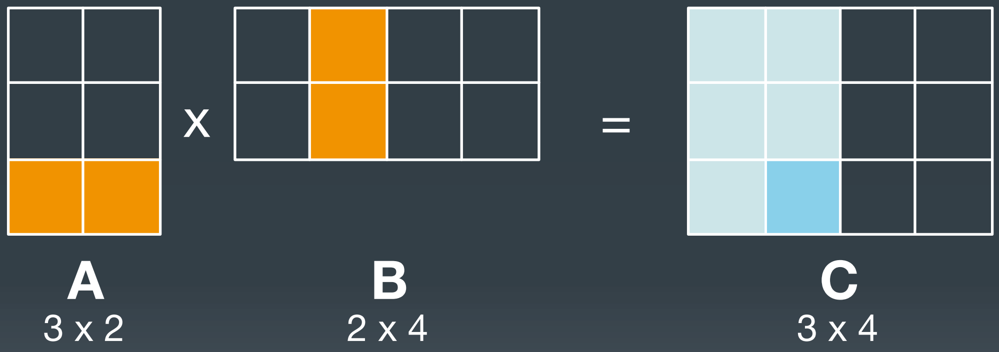
Matrix multiplication

- Factorization can be applied to matrices
- Remember that matrix multiplication is not equal to element-wise multiplication



Matrix multiplication

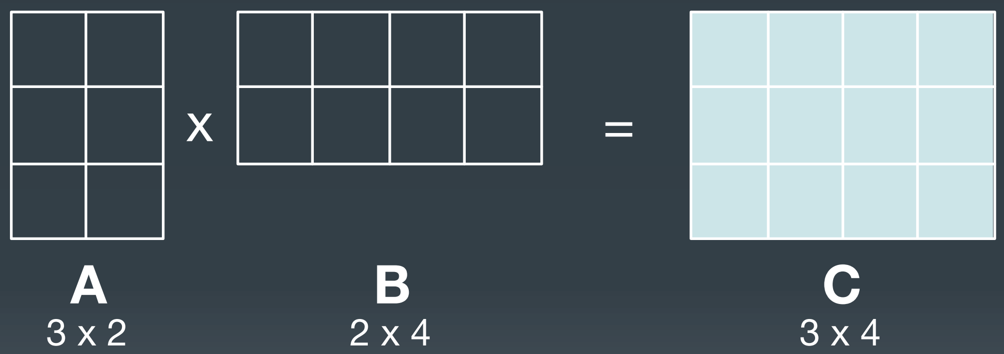
- Factorization can be applied to matrices
- Remember that matrix multiplication is not equal to element-wise multiplication



Matrix multiplication

- Factorization can be applied to matrices

- Remember that matrix multiplication is not equal to element-wise multiplication



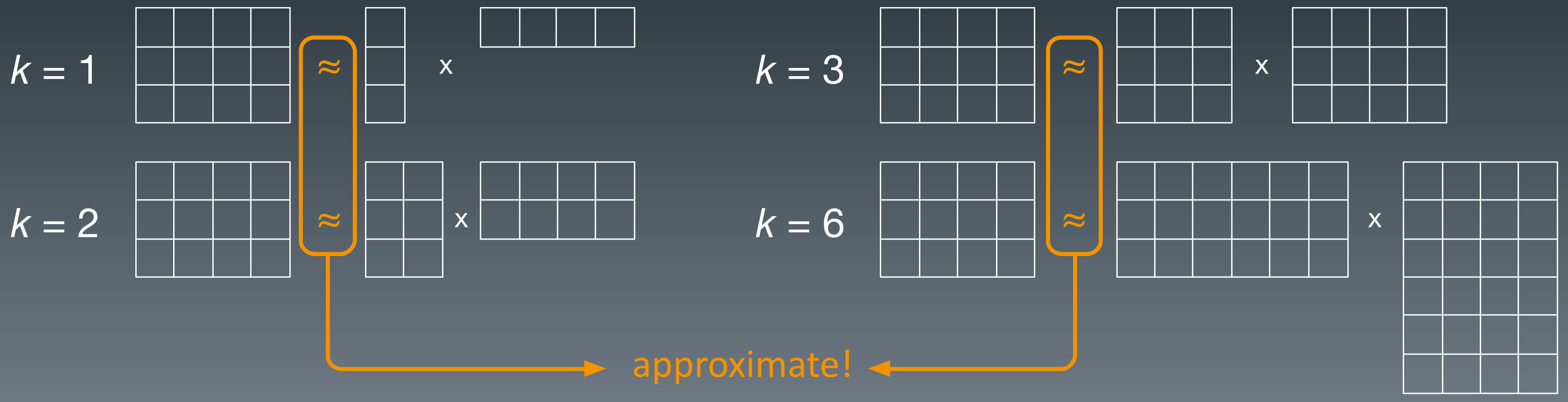
- Matrix multiplication is defined this way so that it corresponds to chaining linear transformations (projections, rotations, scalings...)

- The number of columns of **A** must match the number of rows of **B**

Matrix factorization

- Also infinite solutions. Each algorithm is defined following the desired characteristics of the output factors. Examples:
  - LU and QR decompositions: to solve systems of linear equations
  - EVD (EigenValue Decomposition) and SVD (Singular Value Decomp.):
    for data fitting, statistics, matrix inversion...

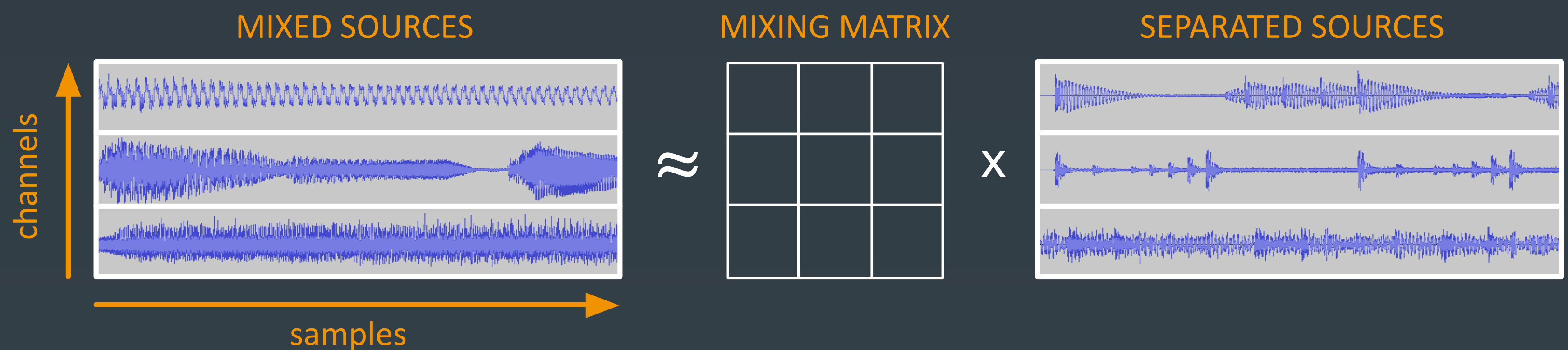
- Furthermore, many algorithms allow to freely choose *k*, the "internal dimension" of the factorization



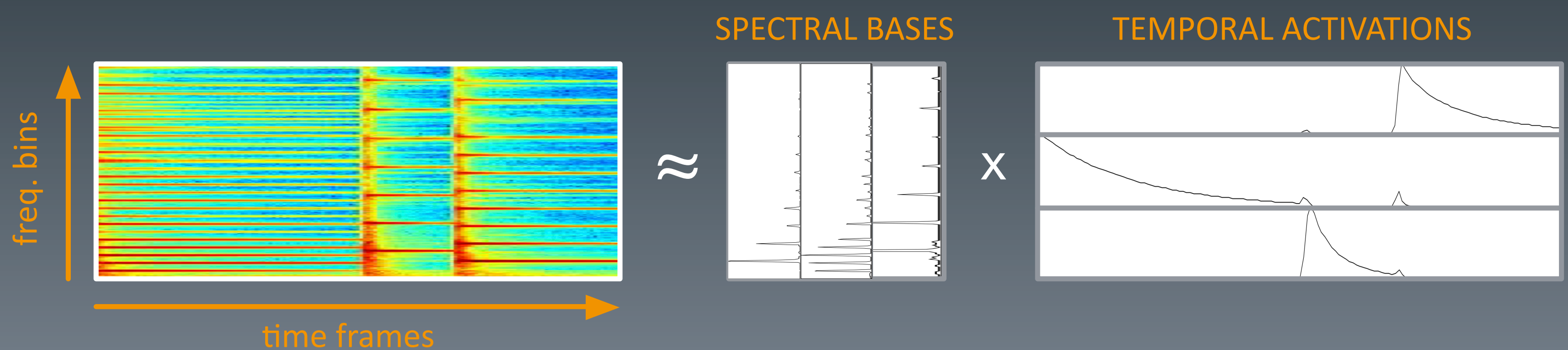
Matrix factorization in audio

- Any audio data that can be arranged as a matrix can be subjected to factorization. 2 typical scenarios:

1. Input matrix is a multichannel time-domain signal (e.g. ICA)

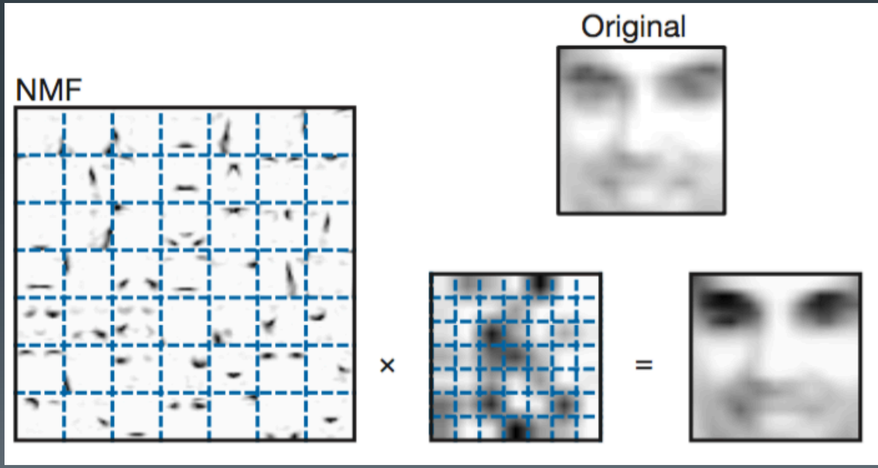


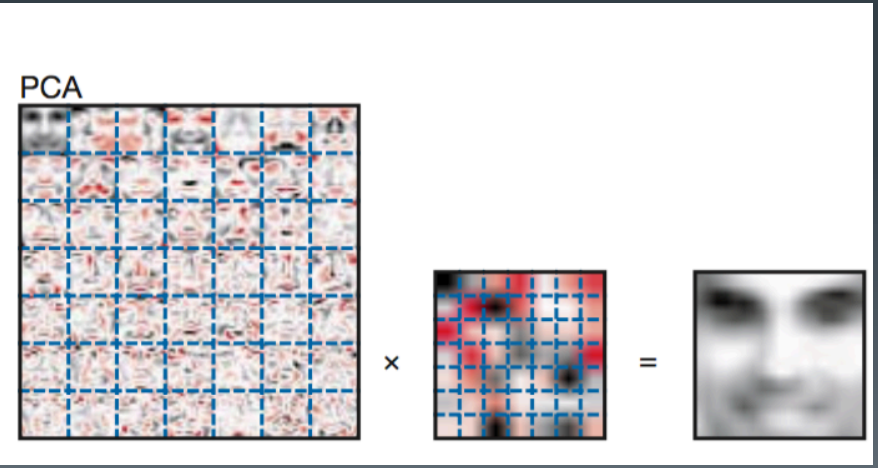
2. Input matrix is a magnitude spectrogram (e.g. ISA, NMF)



Non-negative Matrix Factorization (NMF)

- All elements of all 3 matrices involved have to be 0 or positive
- This simple constraint is enough to make the output factors more easily interpretable: it is a “parts-based” decomposition
- D. Lee (Bell Labs) and S. Seung (MIT), 1999

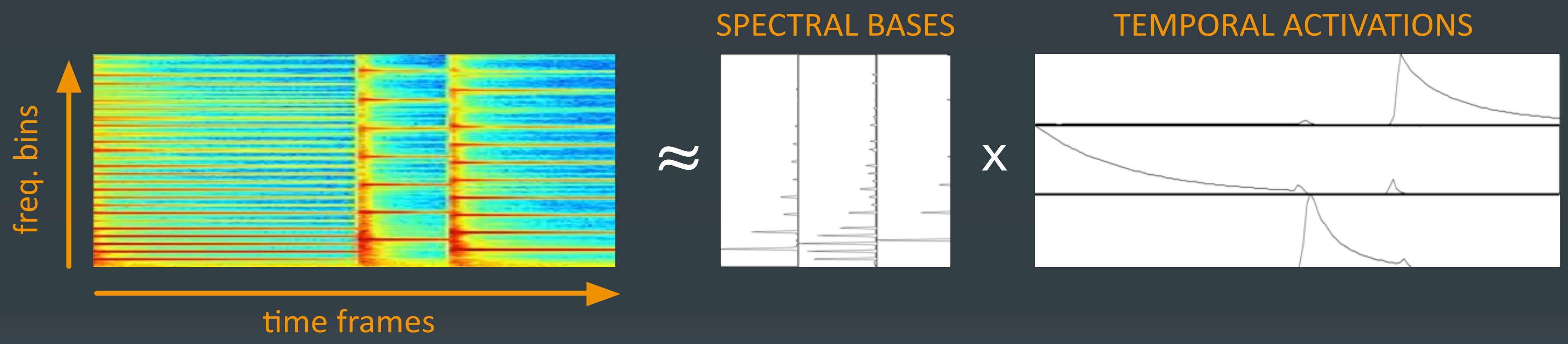




D. Lee and S. Seung, “Learning the parts of objects by non-negative matrix factorization”, Nature, 1999

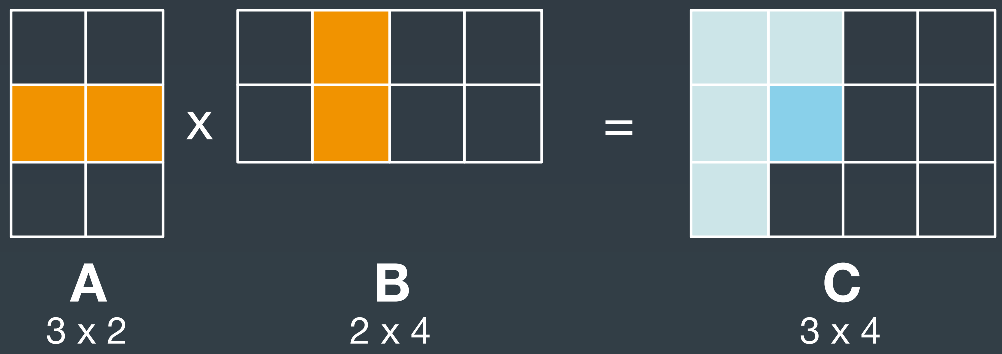
Spectogram factorization with NMF

- Toy example: 3 piano notes, NMF with k=3

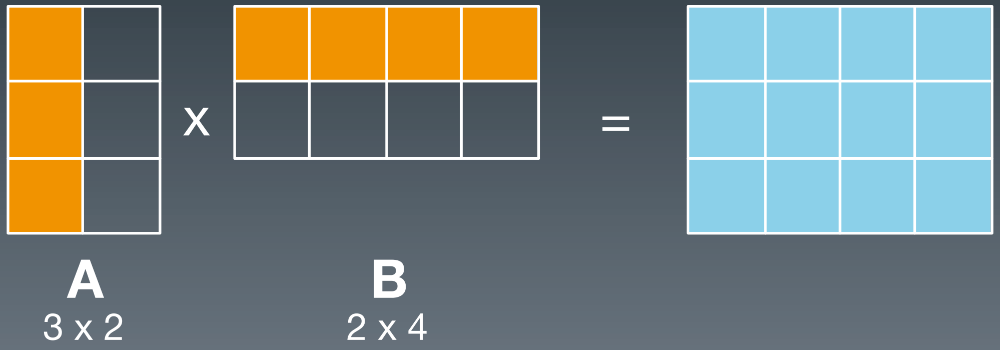


Layered view of matrix multiplication

- Traditional (dot product) view

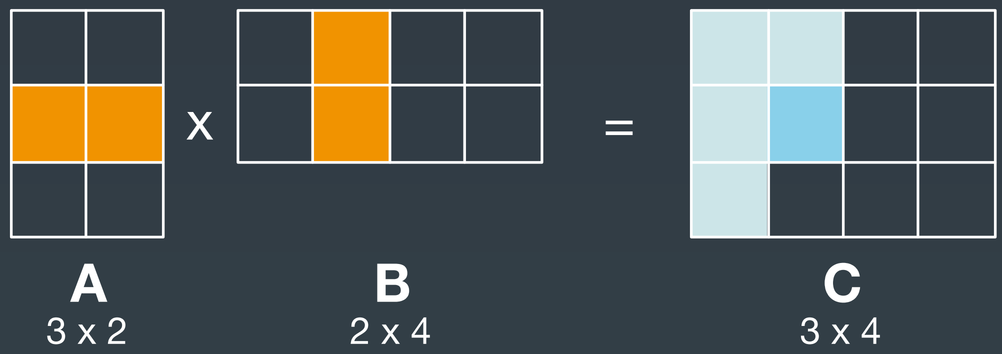


- Layered (outer product) view

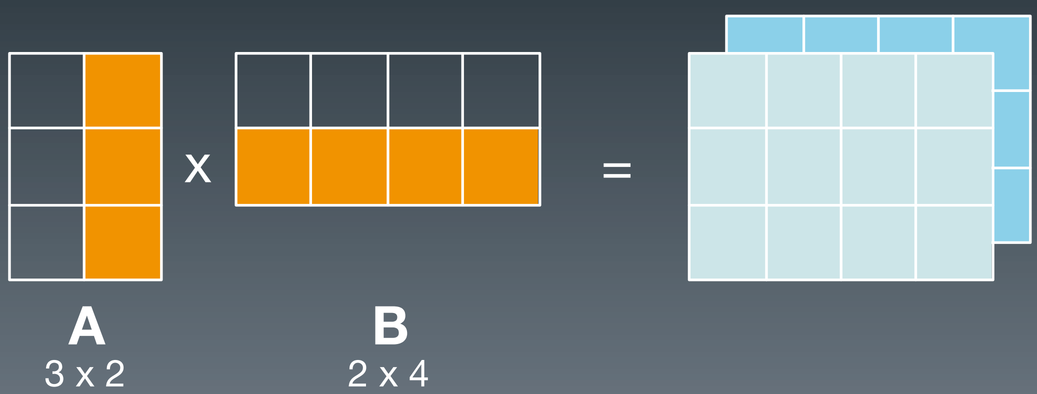


Layered view of matrix multiplication

- Traditional (dot product) view

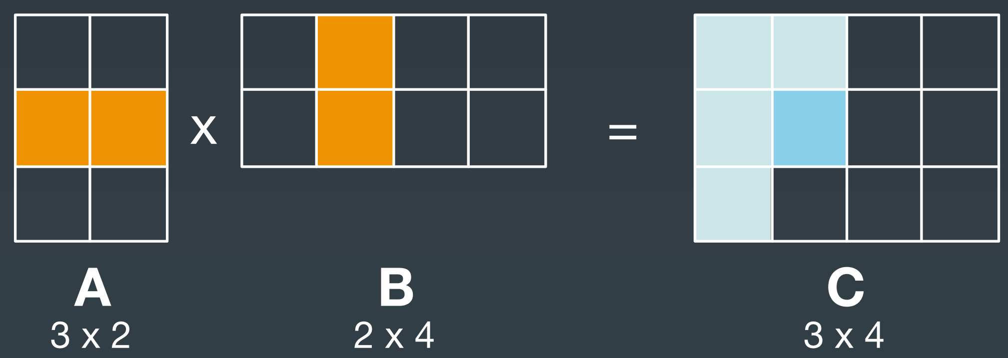


- Layered (outer product) view

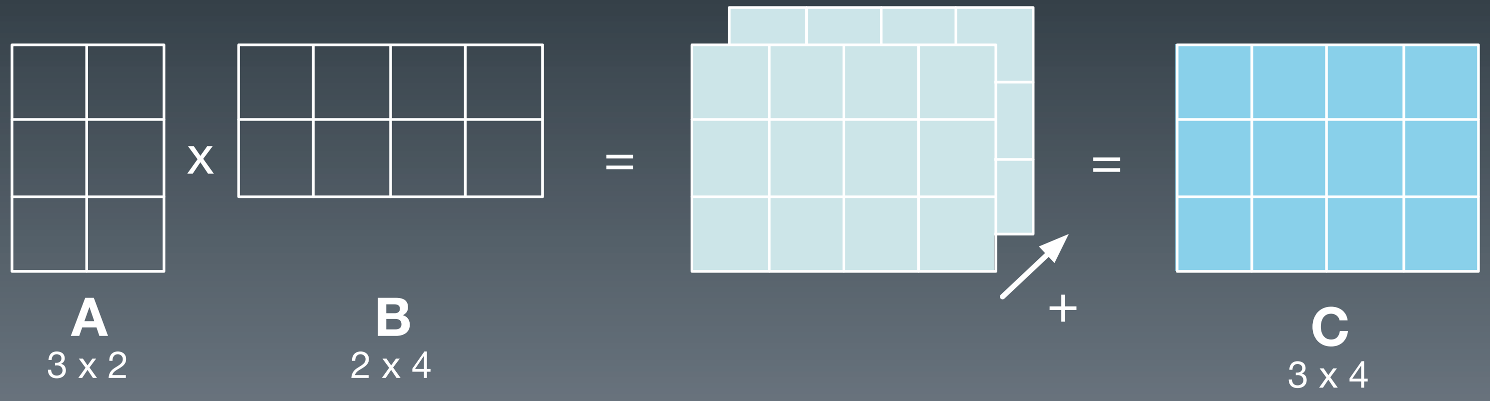


Layered view of matrix multiplication

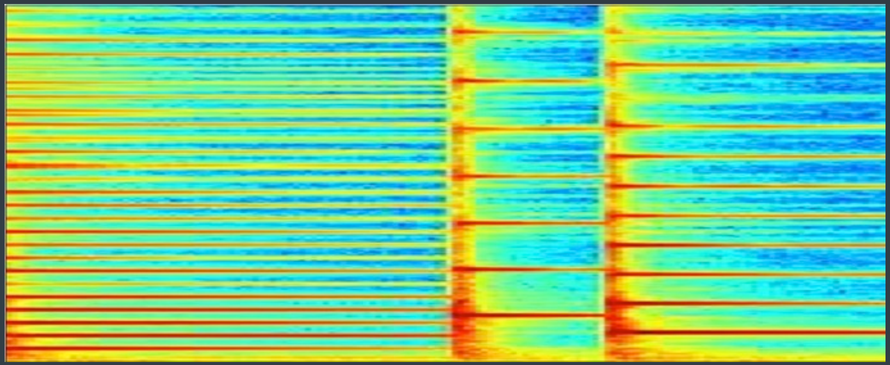
- Traditional (dot product) view

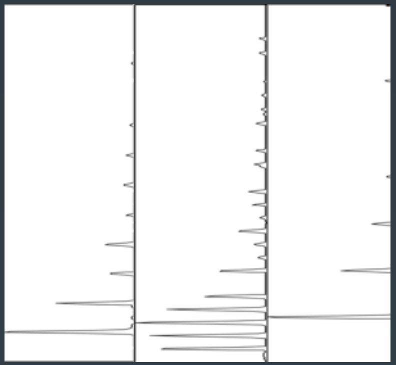


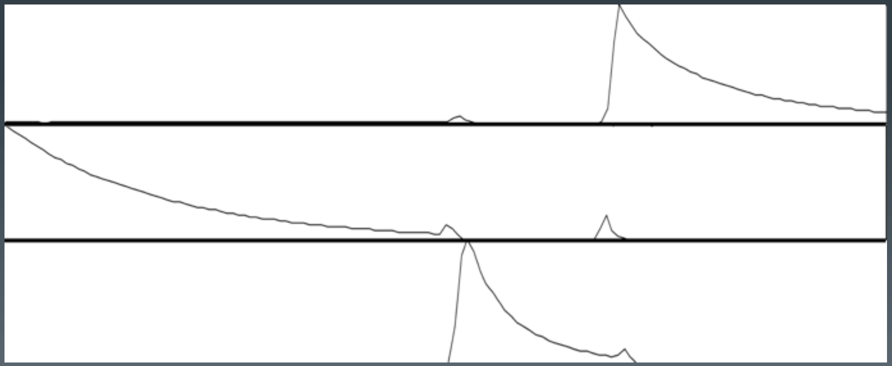
- Layered (outer product) view



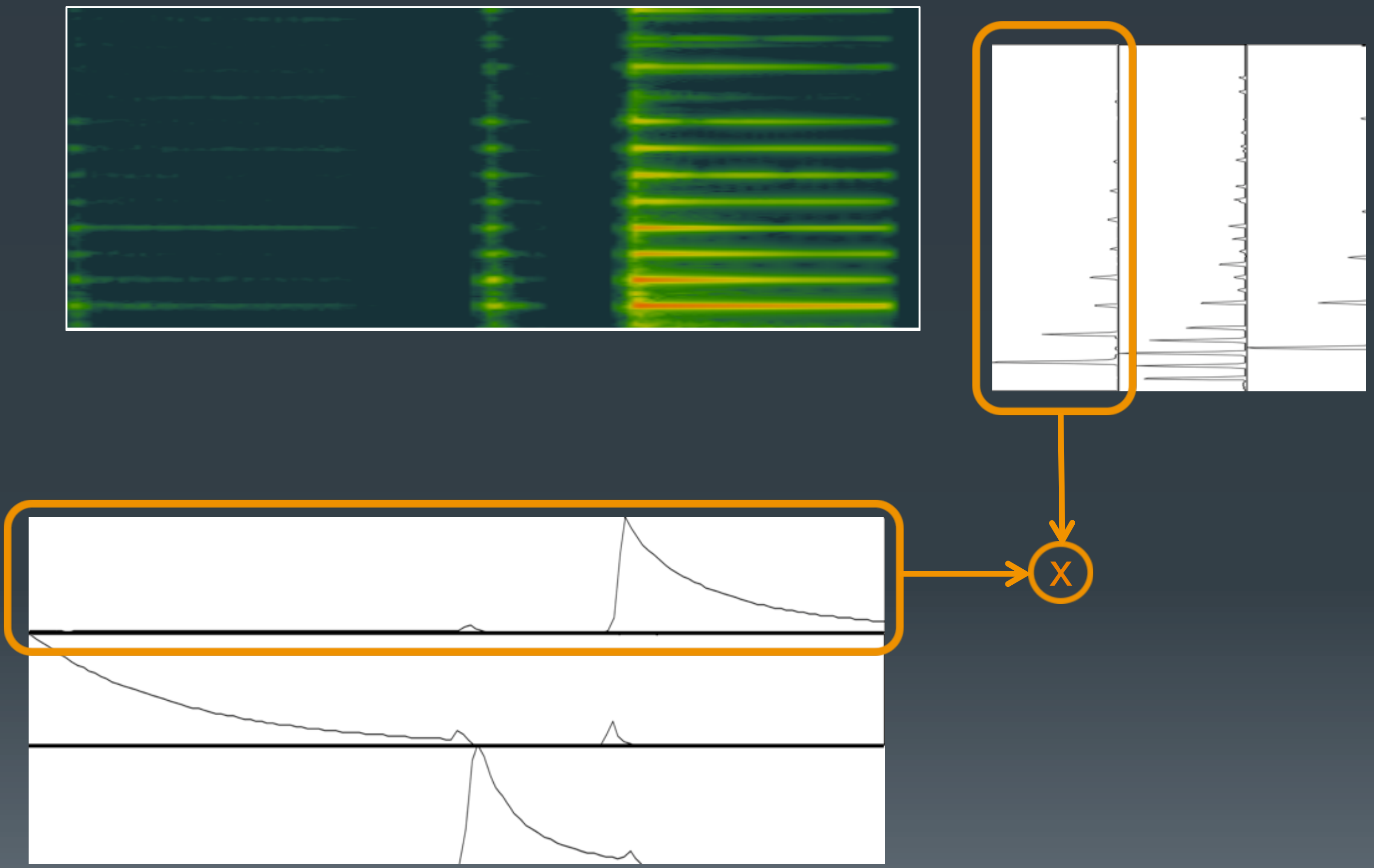
Layered view of matrix multiplication



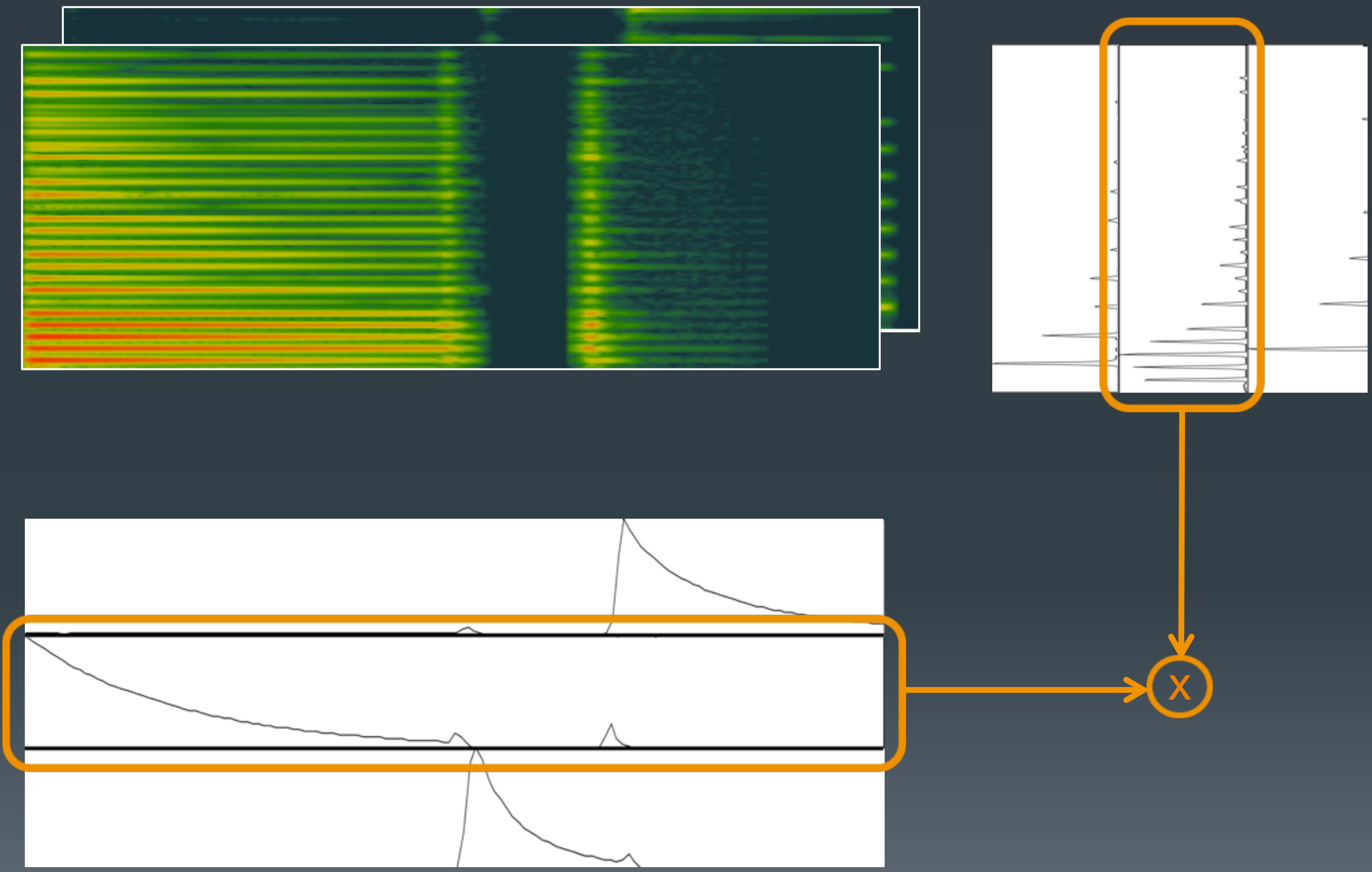




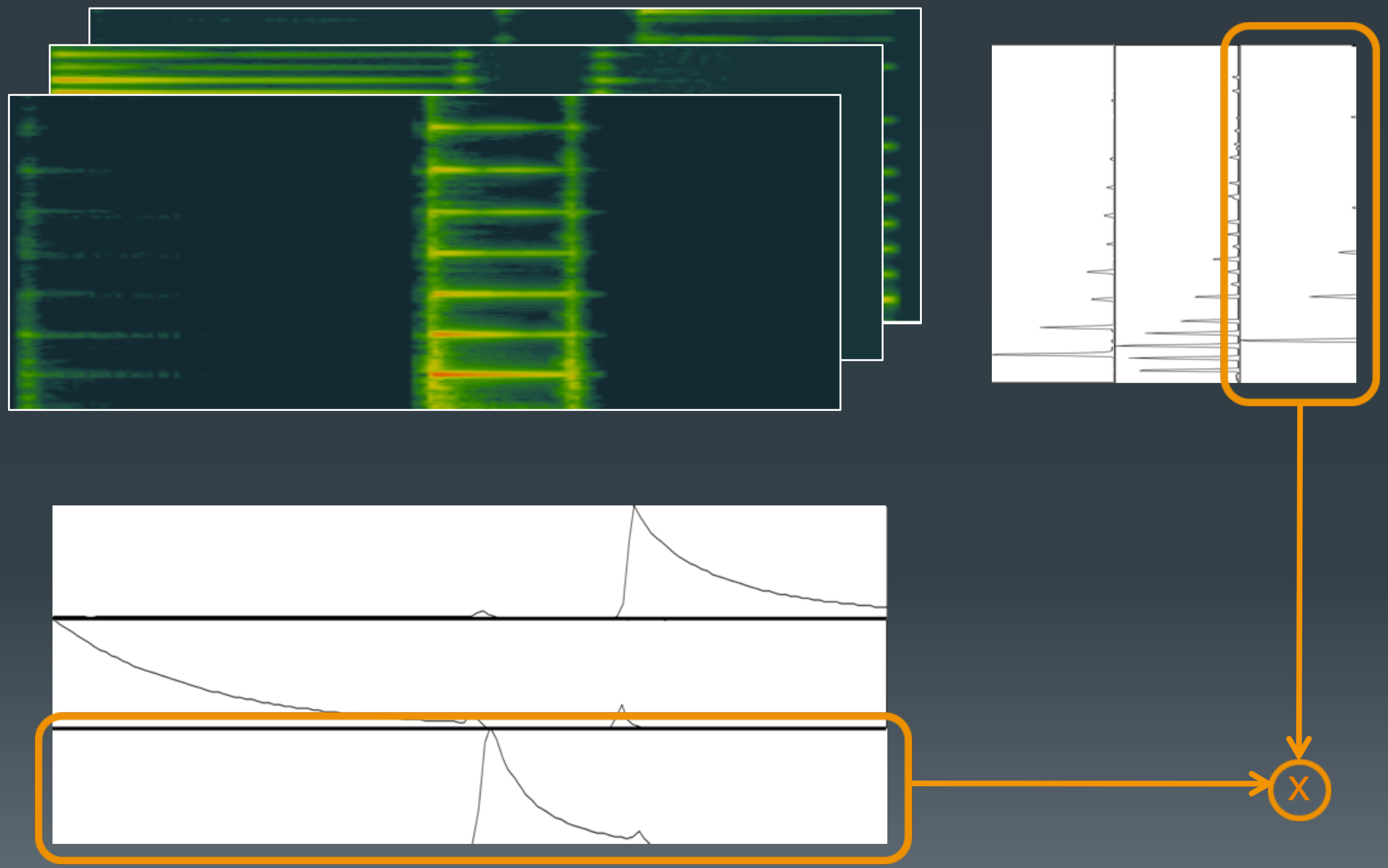
Layered view of matrix multiplication



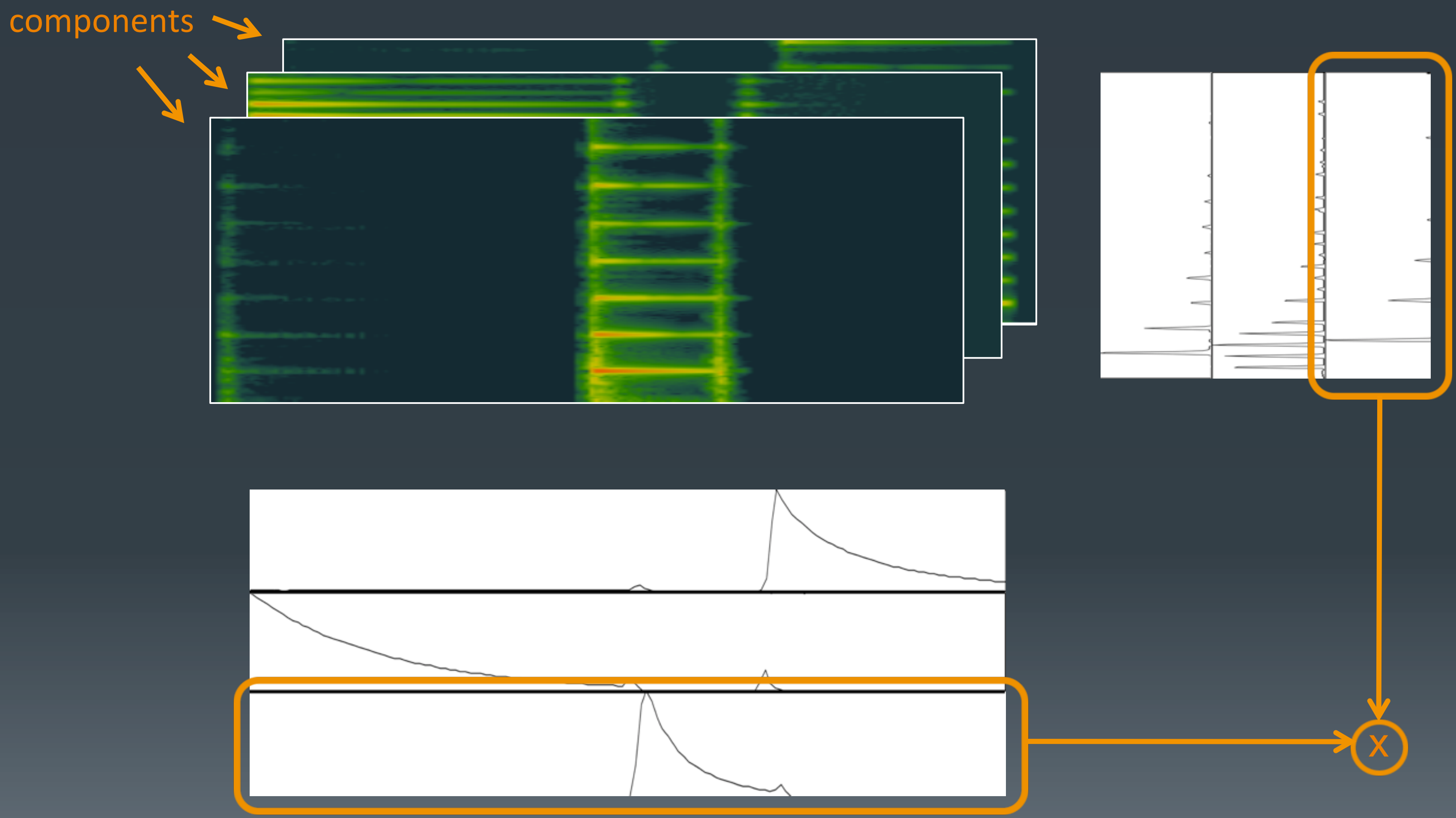
Layered view of matrix multiplication



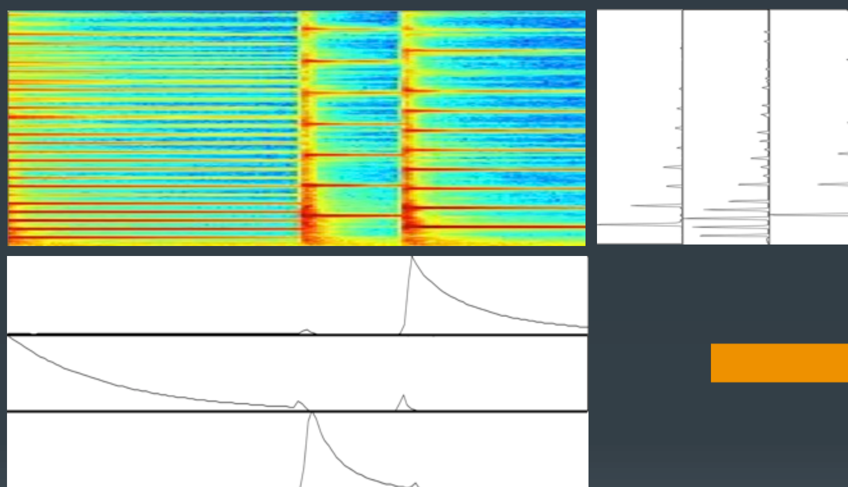
Layered view of matrix multiplication



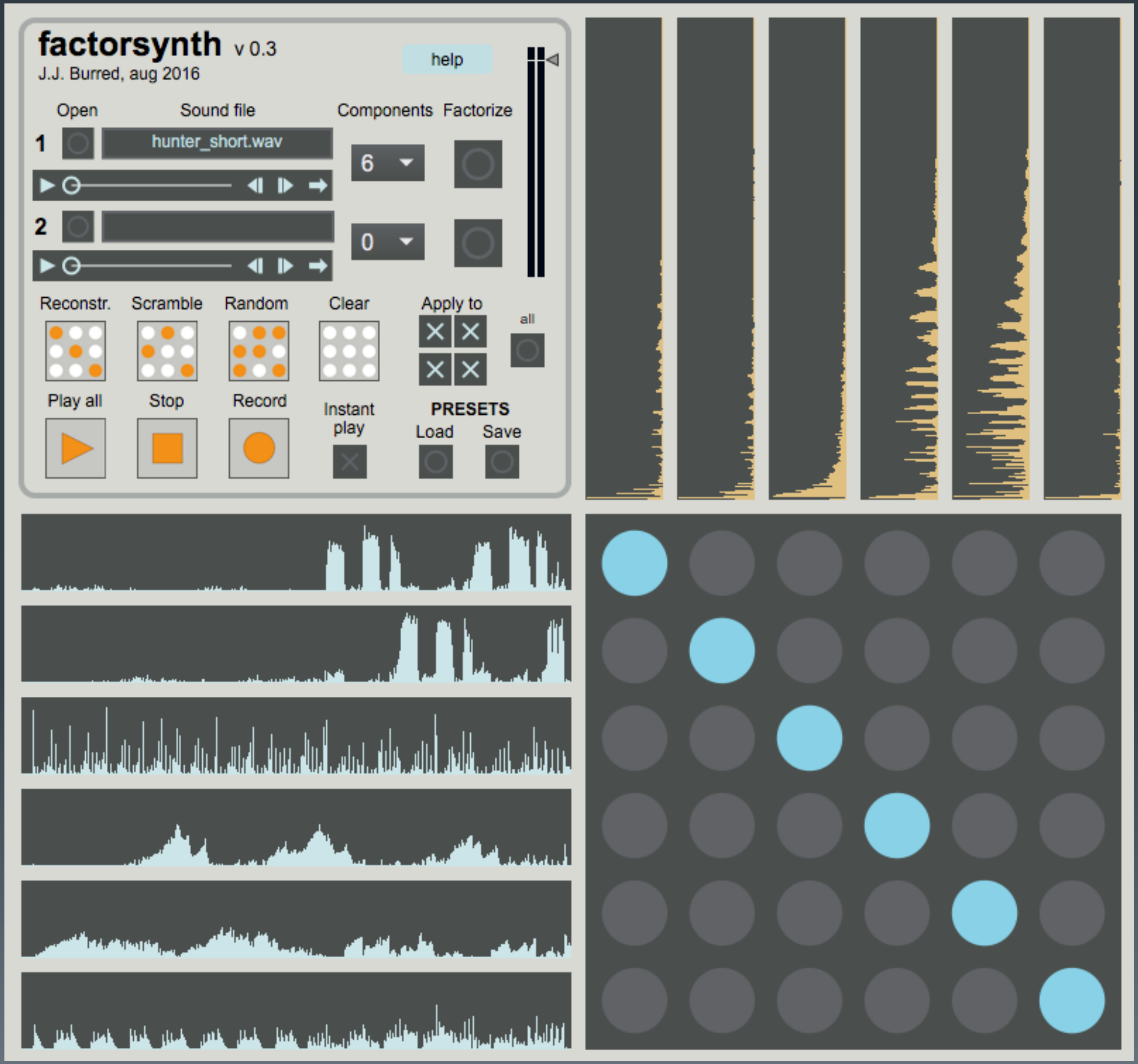
Layered view of matrix multiplication



Factorsynth display

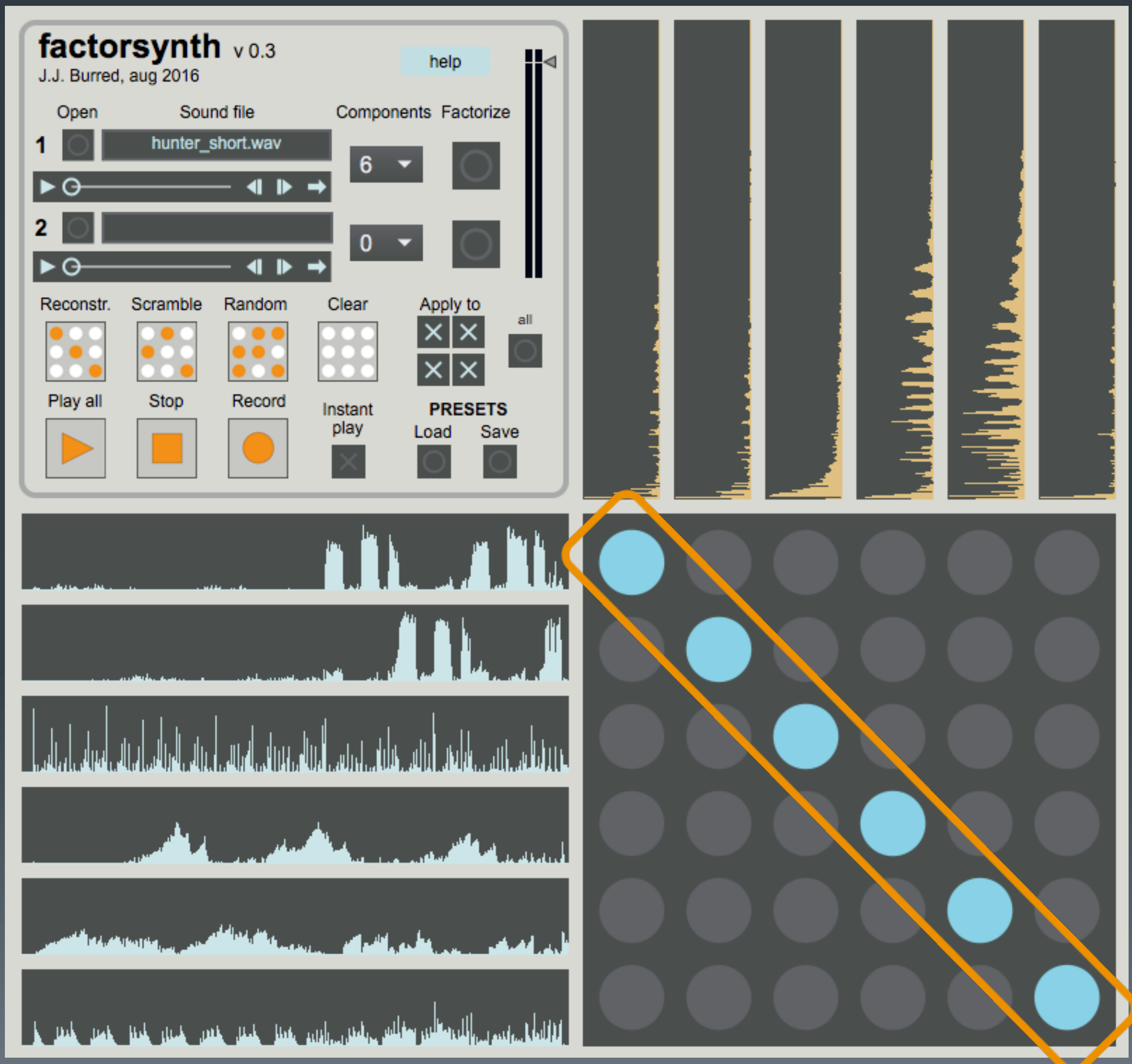


The Factorsynth switchboard



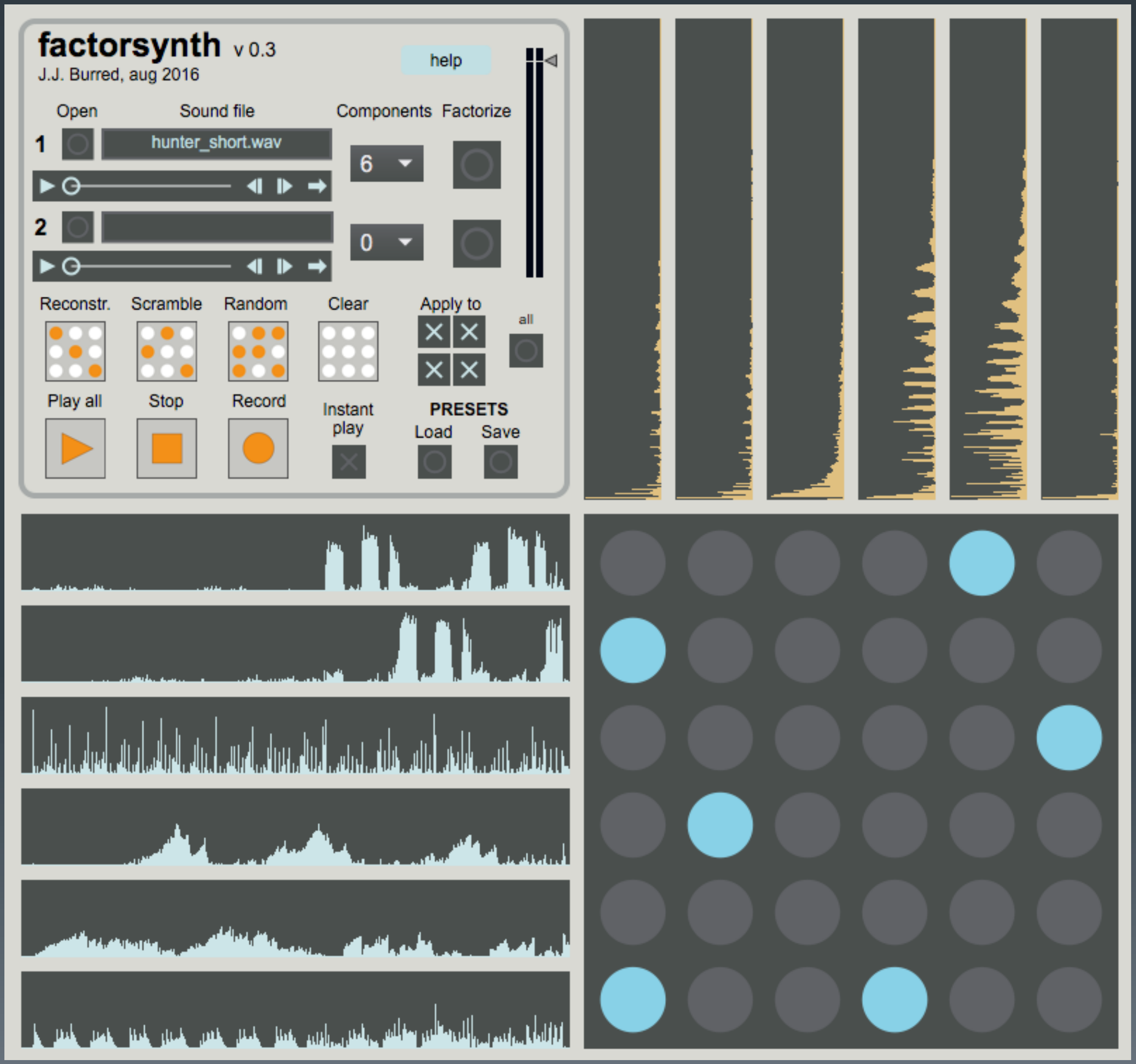
- Each selected button on the switchboard creates a component (time-frequency layer) by multiplying the activation to its left with the spectrum above it

The Factorsynth switchboard



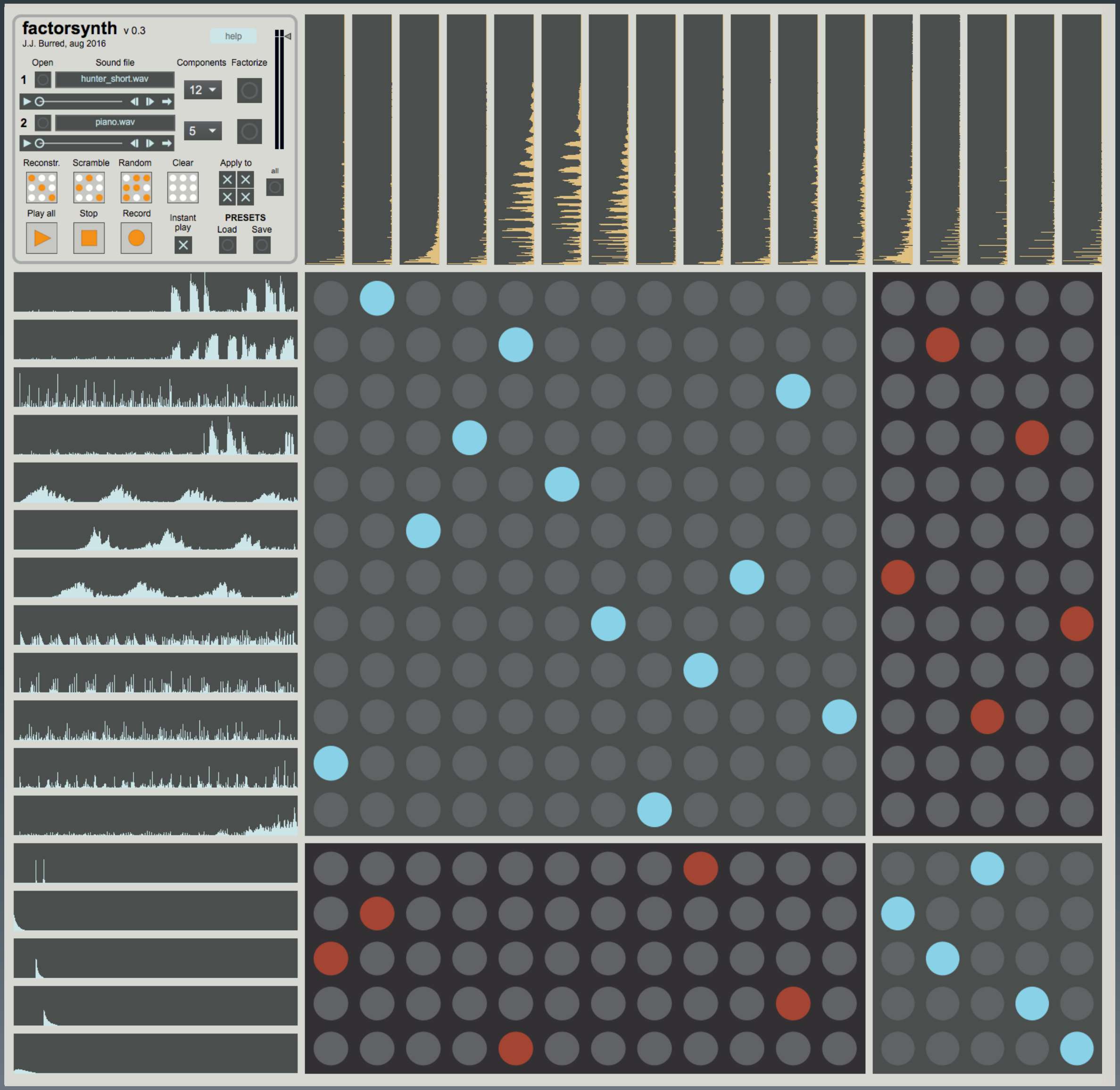
- Diagonal buttons correspond to the original components
- This is the situation in traditional uses of factorization (source separation, compression, transcription...)

The Factorsynth switchboard



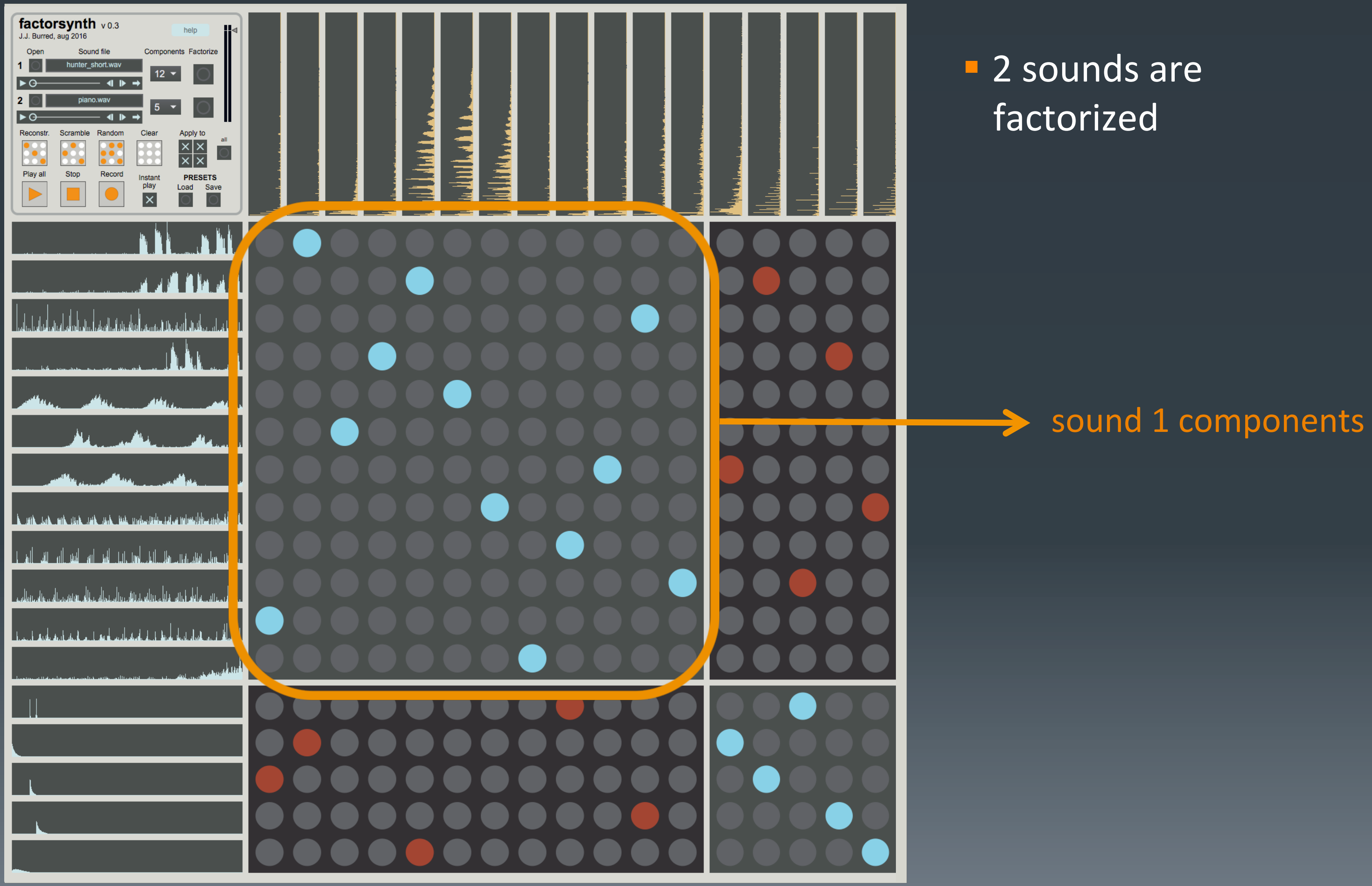
- Key of Factorsynth: allow off-diagonal connections
- This creates new components, not present in the original sound
- Furthermore, activations and bases can be manually edited

Cross-synthesis mode



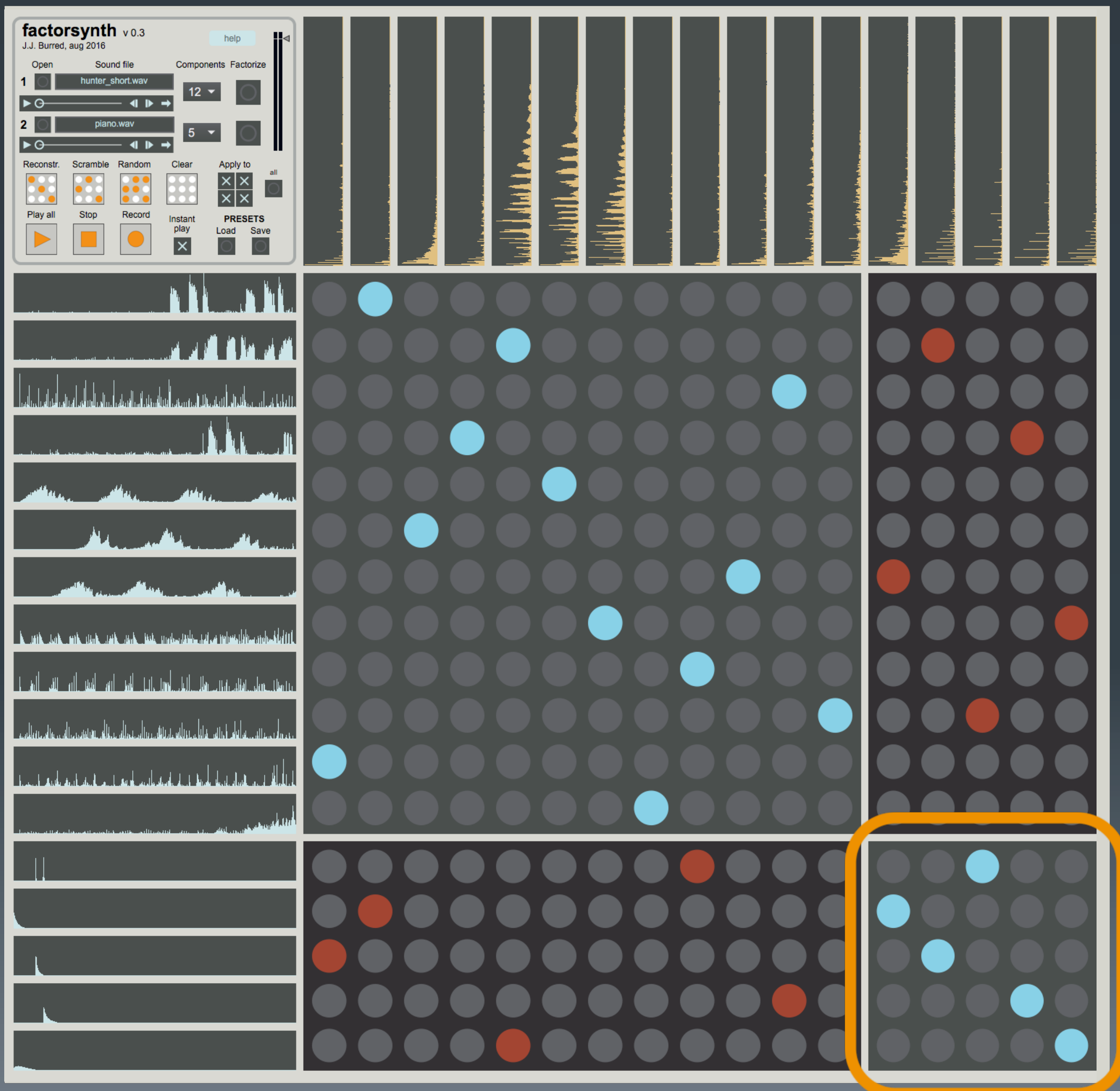
- 2 sounds are factorized

Cross-synthesis mode



- 2 sounds are factorized

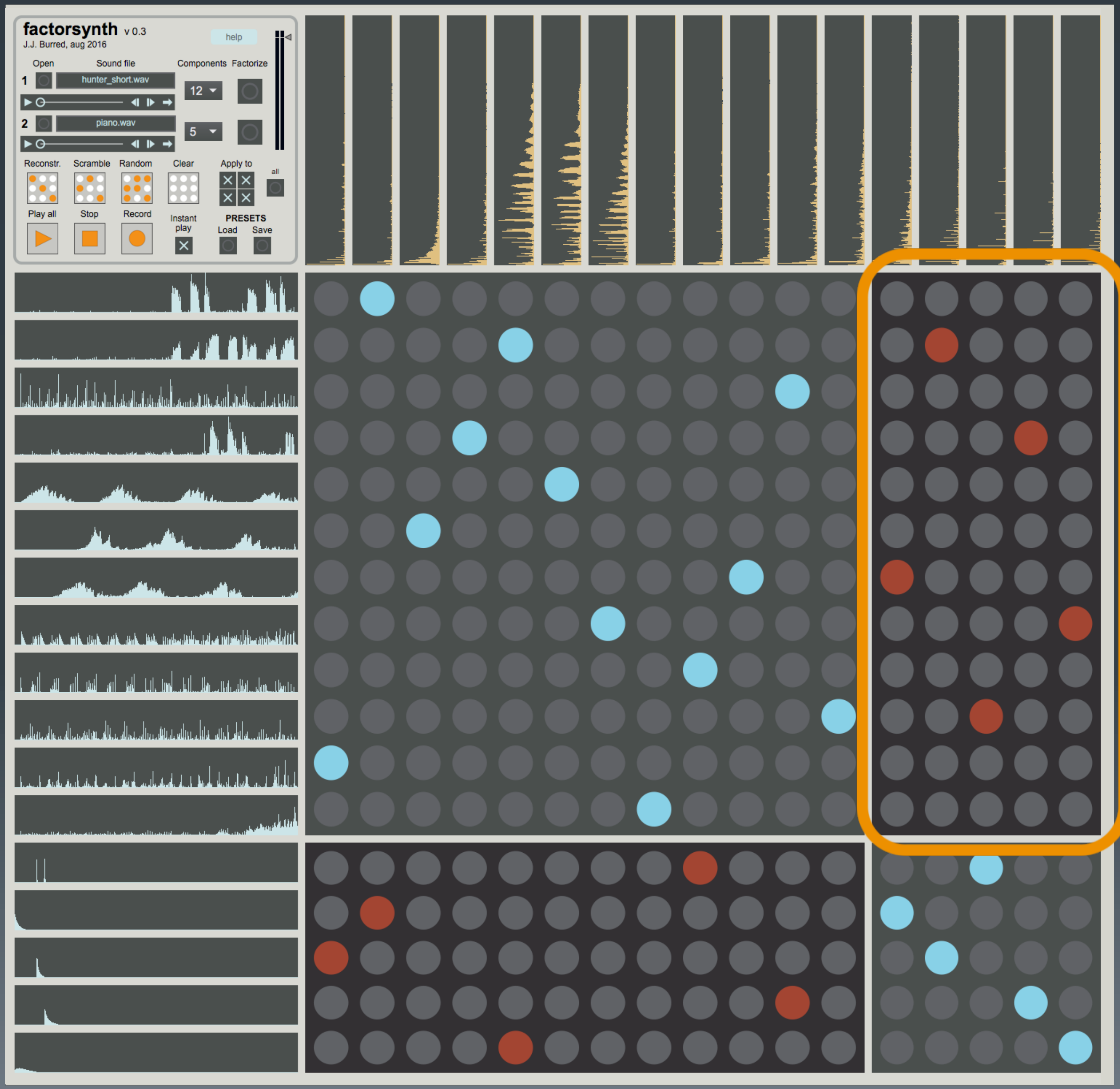
Cross-synthesis mode



sound 2 components

- 2 sounds are factorized

Cross-synthesis mode

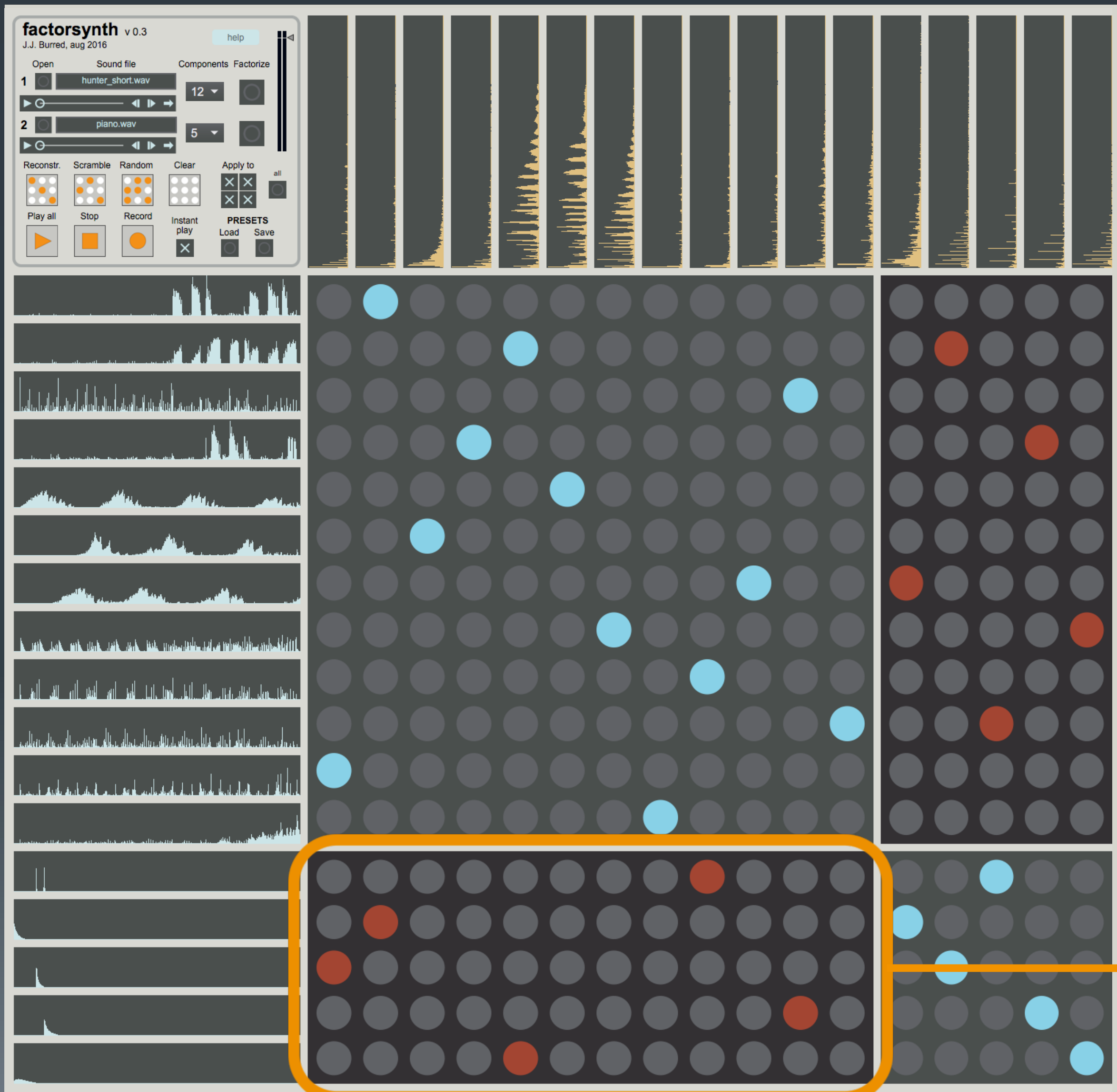


- 2 sounds are factorized

→ 1➔2 cross-components

(activations of sound 1, spectra of sound 2)

Cross-synthesis mode



- 2 sounds are factorized

2→1 cross-components
(activations of sound 2, spectra of sound 1)

Resynthesis

- We have seen that each connection generates a time-frequency layer, which is a magnitude spectrogram.
- For resynthesis, all layers are added to create the final synthesis spectrogram.
- NMF only works on real numbers, so phases (needed for resynthesis) are missing.
- 2 options:
  1. Generate phases from scratch → additive resynthesis
  2. Take phases from input → subtractive resynthesis

Subtractive resynthesis

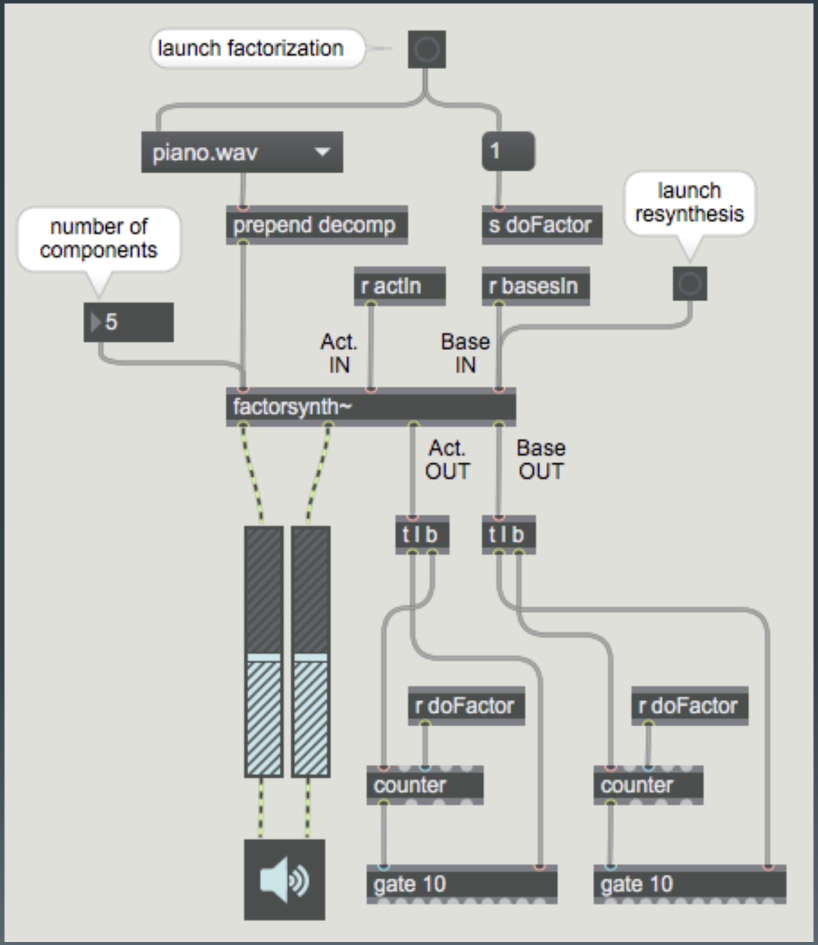
- The final resynthesis spectrogram is normalized and applied as a time-frequency mask to the input spectrogram.
- The implementation uses a ad-hoc modified Wiener mask, the method of choice in source separation, due to its better sound quality (transient preservation).
- Thus, it can be seen as an adaptive subtractive synthesis
- This has an important implication in Factorsynth: sometimes, output components will be softer than expected, if the original frequency areas they are filtering contain little energy.

Implementation

- Implemented and tested as a patcher for Max 7 (Mac OS).
- Heavy use of JavaScript for the GUI.
- For now, loads and stores WAV files.
- Prototype version (v0.3) as Creative Commons freeware:
  - download at `jjburred.com`
- Stable, stand-alone versions to come.
  - Comments and bug reports are more than welcome!
- A command-line executable version is also available.

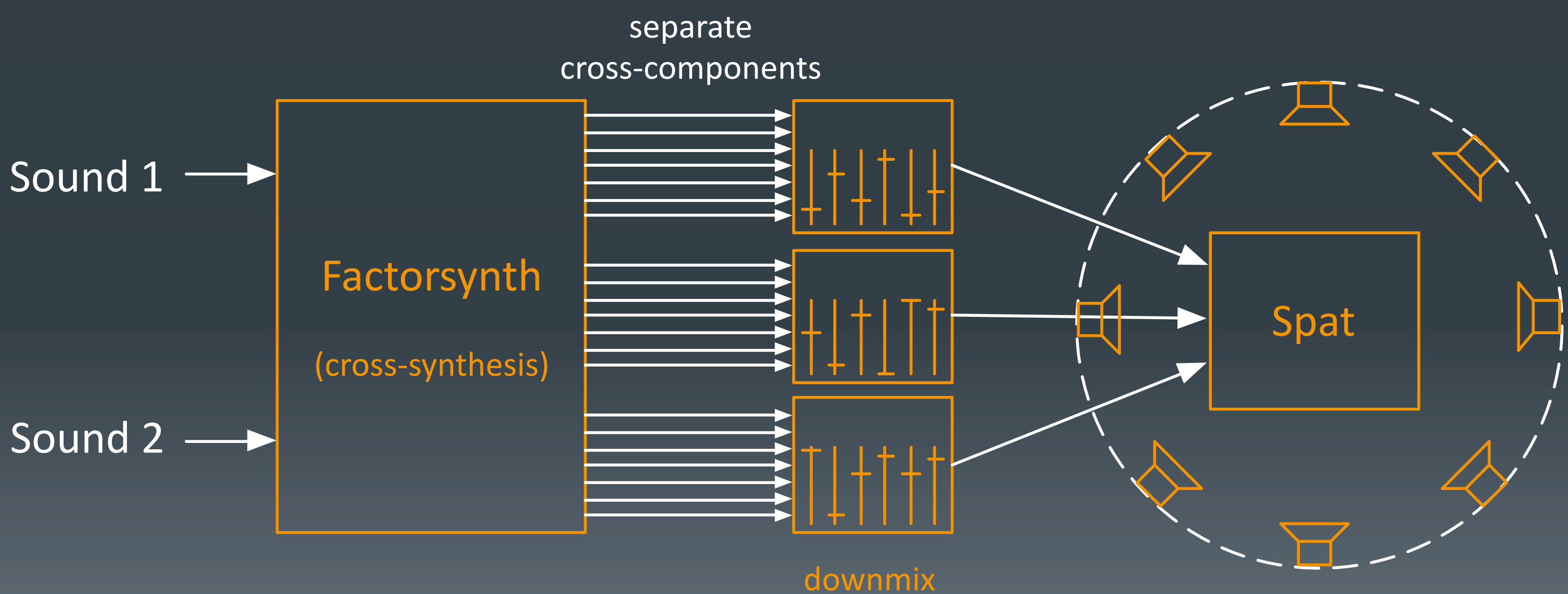
The factorsynth~ external

- The core of the patch is the factorsynth~ external object.
- Implements both NMF factorization and modified Wiener resynthesis.
- Efficient implementation that makes use of Apple’s vDSP library.
  - Factorization time: 25% of length of input file
  - Resynthesis time: almost instantaneous



Example of real-life usage

- Usage in “Artaud Overdrive” by Emanuele Palumbo
  - Premiered June 2016 at Manifest festival (IRCAM), Centre Pompidou



Factorsynth: take-home messages

- Factorsynth is can be understood a spectral editing tool in which the elements to be manipulated are of a relatively high abstraction level (notes, transients, impulsive events, spectral structures...).

- In other words, the representation bases are full spectra, instead of sinusoids.

- Beyond editing of existing elements, Factorsynth can also create new sound elements by combining unrelated spectral and temporal shapes.

- It implements a new kind of cross-synthesis at the level of internal sound events.

Future developments

- Additive resynthesis (white noise input / phase vocoder).
- Use buffers instead of files for input/output.
- Processing of real-time input (with pre-stored bases and activations).
- Automatic switchboard connections by spectral similarity (alternative scatter plot interface?)
- Multichannel output (component-based spatialization).