# Preserving User-Defined Expression through Dimensionality Reduction

Ted Moore, Doctoral Fellow in Music Composition
University of Chicago

*“new control strategies for an aging electronic music instrument” -Lauren*

Ted Moore, Doctoral Fellow in Music Composition
University of Chicago

*“the sweet spots are all over the place, they’re not contiguous, are they?” -Owen*

Ted Moore, Doctoral Fellow in Music Composition
University of Chicago

# The Goal

Using sound generators that have a high dimension of control inputs,

find expressively meaningful combinations of input settings,

then intelligently organize those combinations into fewer dimensions (using unsupervised learning).

# Hypothesis

Dimensionality reduction can enhance musical expression by enabling quick, interpolated, and gestural movements through high dimensional spaces.

Unsupervised learning strategies, will create more useful and meaningful (e.g., expressive) low dimensional latent spaces than supervised strategies.

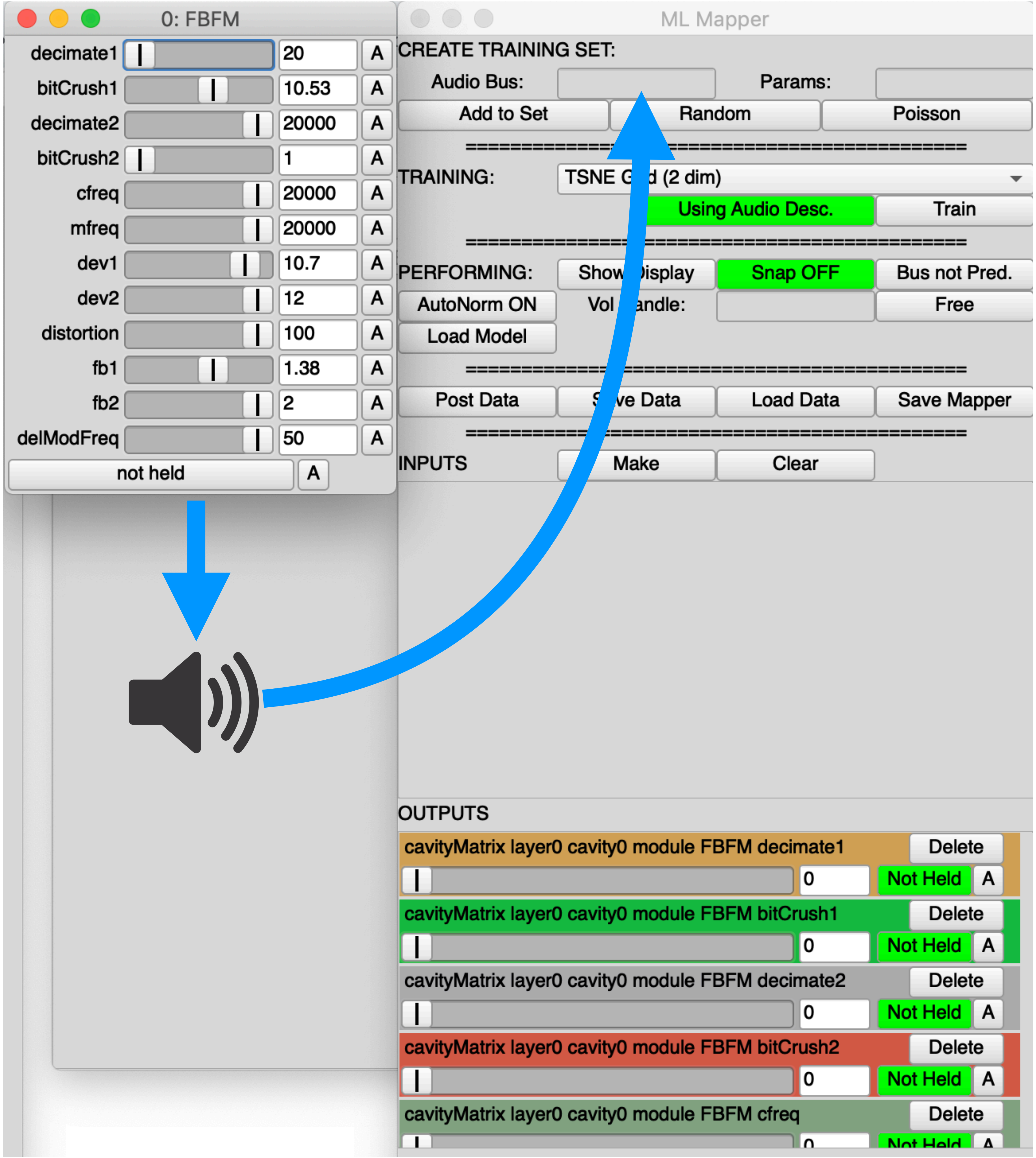
Supervised strategies require one to know the structure of the high dimensional space and low dimensional space ahead of time…

# Related Work

- Fasciani and Wyse (2012)
  - “The optimal mapping is defined as the one allowing the ***widest*** a sonic exploration”
  - “we assume a ***deterministic*** behaviour, excluding the presence of any stochastic component within the chain.”

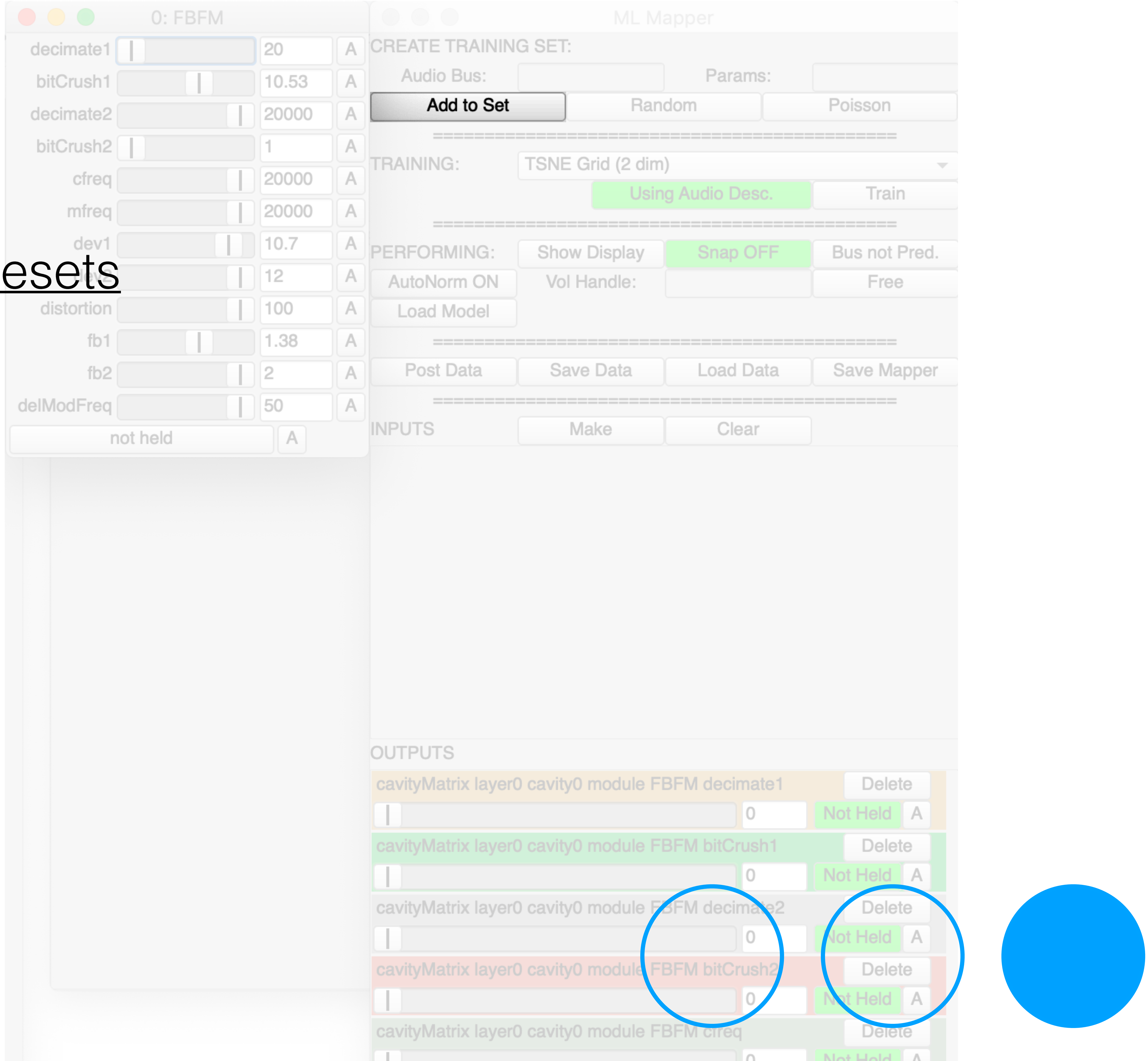

0: FBFM
decimate1 20 A
bitCrush1 10.53 A
decimate2 20000 A
bitCrush2 1 A
cfreq 20000 A
mfreq 20000 A
dev1 10.7 A
dev2 12 A
distortion 100 A
fb1 1.38 A
fb2 2 A
delModFreq 50 A
not held A
ML Mapper
CREATE TRAINING SET:
Audio Bus:
Params:
Add to Set
Random
Poisson
TRAINING:
TSNE Grid (2 dim)
Using Audio Desc.
Train
PERFORMING:
Show Display
Snap OFF
Bus not Pred.
AutoNorm ON
Vol Handle:
Free
Load Model
Post Data
Save Data
Load Data
Save Mapper
INPUTS
Make
Clear
OUTPUTS
cavityMatrix layer0 cavity0 module FBFM decimate1 Delete
0 Not Held A
cavityMatrix layer0 cavity0 module FBFM bitCrush1 Delete
0 Not Held A
cavityMatrix layer0 cavity0 module FBFM decimate2 Delete
0 Not Held A
cavityMatrix layer0 cavity0 module FBFM bitCrush2 Delete
0 Not Held A
cavityMatrix layer0 cavity0 module FBFM cfreq Delete


0: FBFM
decimate1 20 A
bitCrush1 10.53 A
decimate2 20000 A
bitCrush2 1 A
cfreq 20000 A
mfreq 20000 A
dev1 10.7 A
dev2 12 A
distortion 100 A
fb1 1.38 A
fb2 2 A
delModFreq 50 A
not held A
ML Mapper
CREATE TRAINING SET:
Audio Bus:
Params:
Add to Set
Random
Poisson
TRAINING:
TSNE Grid (2 dim)
Using Audio Desc.
Train
PERFORMING:
Show Display
Snap OFF
Bus not Pred.
AutoNorm ON
Vol Handle:
Free
Load Model
Post Data
Save Data
Load Data
Save Mapper
INPUTS
Make
Clear
OUTPUTS
cavityMatrix layer0 cavity0 module FBFM decimate1
Delete
0 Not Held A
cavityMatrix layer0 cavity0 module FBFM bitCrush1
Delete
0 Not Held A
cavityMatrix layer0 cavity0 module FBFM decimate2
Delete
0 Not Held A
cavityMatrix layer0 cavity0 module FBFM bitCrush2
Delete
0 Not Held A
cavityMatrix layer0 cavity0 module FBFM cfreq
Delete


0: FBFM
decimate1 20 A
bitCrush1 10.53 A
decimate2 20000 A
bitCrush2 1 A
cfreq 20000 A
mfreq 20000 A
dev1 10.7 A
dev2 12 A
distortion 100 A
fb1 1.38 A
fb2 2 A
delModFreq 50 A
not held A
ML Mapper
CREATE TRAINING SET:
Audio Bus:
Params:
Add to Set
Random
Poisson
TRAINING:
Using Audio Desc.
Train
PERFORMING:
Snap OFF
Bus not Pred.
AutoNorm ON
Free
Load Model
Post Data
Load Data
Save Mapper
INPUTS
Make
Clear
OUTPUTS
cavityMatrix layer0 cavity0 module FBFM decimate1
Delete
0 Not Held A
cavityMatrix layer0 cavity0 module FBFM bitCrush1
Delete
0 Not Held A
cavityMatrix layer0 cavity0 module FBFM decimate2
Delete
0 Not Held A
cavityMatrix layer0 cavity0 module FBFM bitCrush2
Delete
0 Not Held A
cavityMatrix layer0 cavity0 module FBFM cfreq
Delete

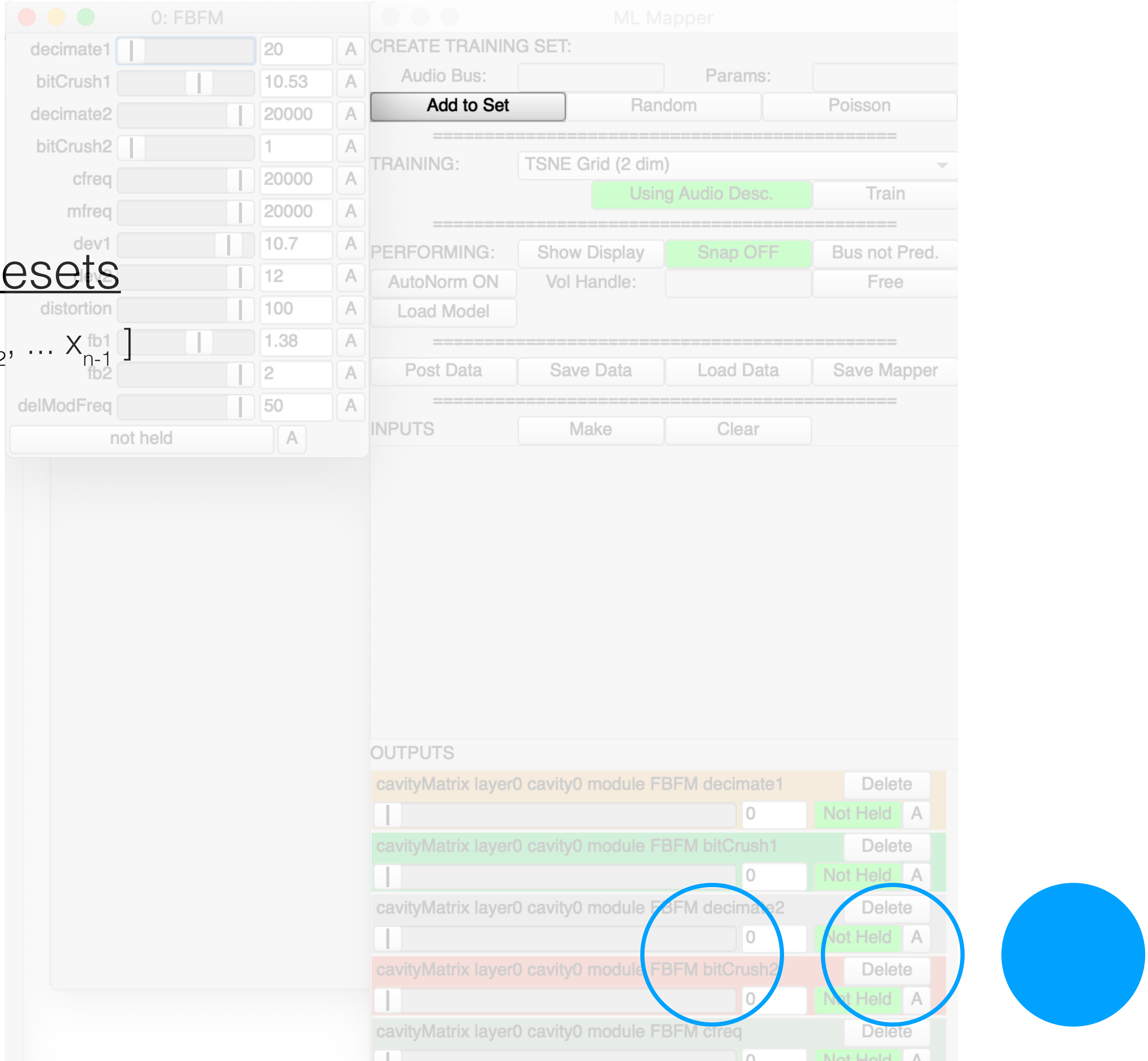

0: FBFM
decimate1 20 A
bitCrush1 10.53 A
decimate2 20000 A
bitCrush2 1 A
cfreq 20000 A
mfreq 20000 A
dev1 10.7 A
dev2 12 A
distortion 100 A
fb1 1.38 A
fb2 2 A
delModFreq 50 A
not held A
ML Mapper
CREATE TRAINING SET:
Audio Bus:
Params:
Add to Set
Random
Poisson
TRAINING:
TSNE Grid (2 dim)
Using Audio Desc.
Train
PERFORMING:
Show Display
Snap OFF
Bus not Pred.
AutoNorm ON
Vol Handle:
Free
Load Model
Post Data
Save Data
Load Data
Save Mapper
INPUTS
Make
Clear
OUTPUTS
cavityMatrix layer0 cavity0 module FBFM decimate1
Delete
0 Not Held A
cavityMatrix layer0 cavity0 module FBFM bitCrush1
Delete
0 Not Held A
cavityMatrix layer0 cavity0 module FBFM decimate2
Delete
0 Not Held A
cavityMatrix layer0 cavity0 module FBFM bitCrush2
Delete
0 Not Held A
cavityMatrix layer0 cavity0 module FBFM cfreq
Delete

# Vector Presets



# Vector Presets

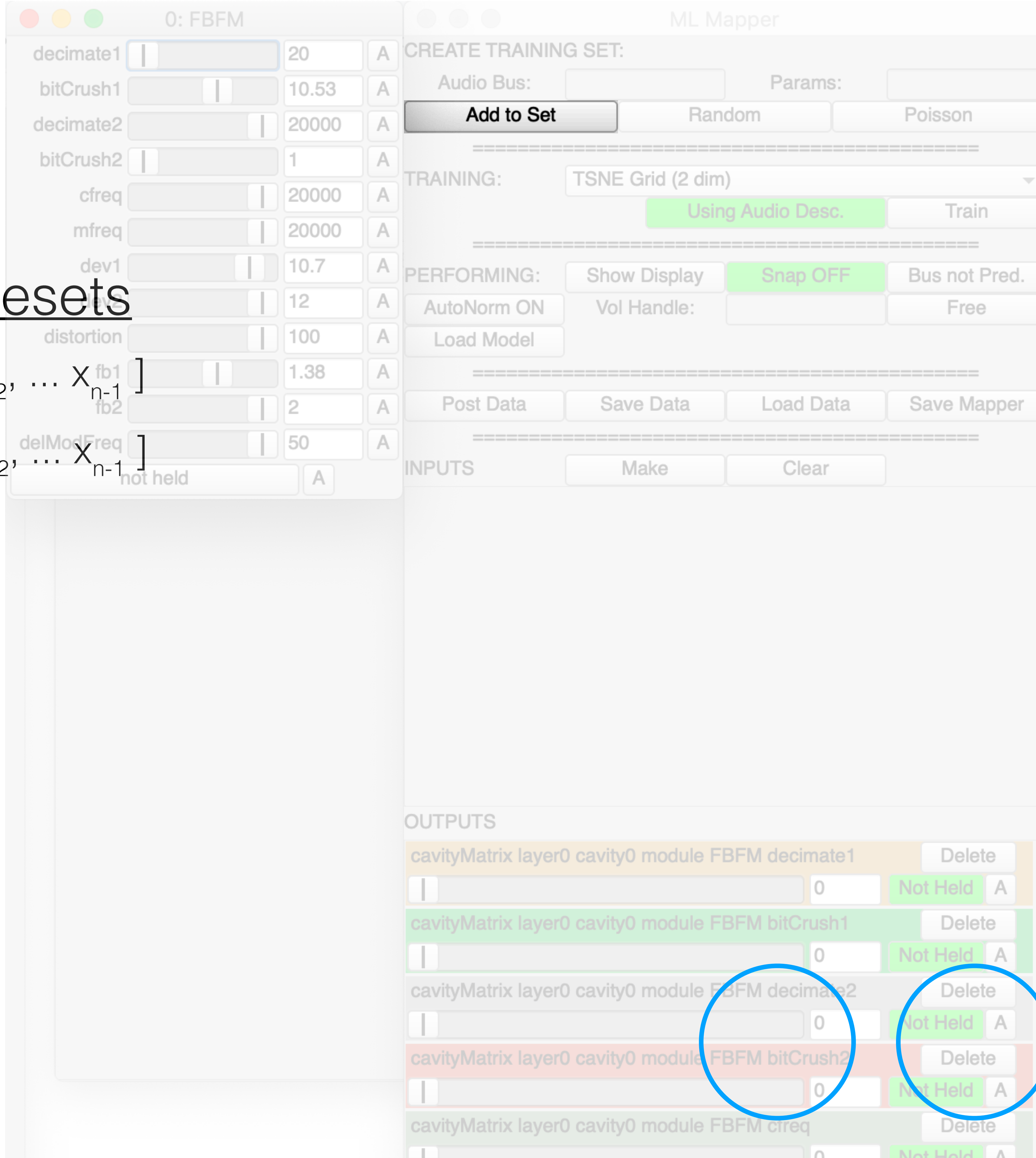
$$a = [\ x_0,\ x_1,\ x_2,\ \ldots\ x_{n-1}\ ]$$



# Vector Presets

$$a = [\ x_0,\ x_1,\ x_2,\ \ldots\ x_{n-1}\ ]$$

$$b = [\ x_0,\ x_1,\ x_2,\ \ldots\ x_{n-1}\ ]$$

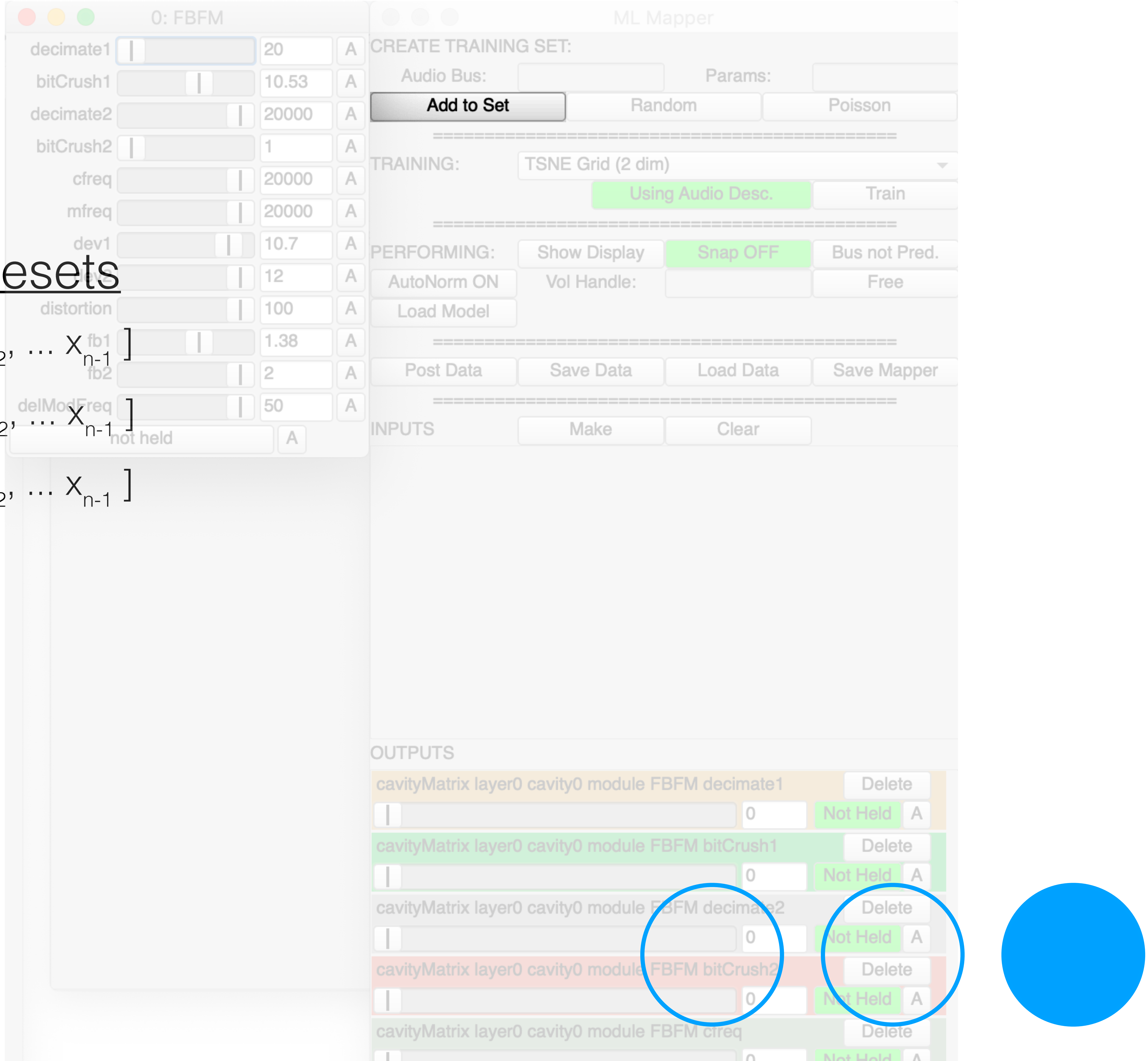


# Vector Presets

$$a = [\ x_0,\ x_1,\ x_2,\ \ldots\ x_{n-1}\ ]$$

$$b = [\ x_0,\ x_1,\ x_2,\ \ldots\ x_{n-1}\ ]$$

$$c = [\ x_0,\ x_1,\ x_2,\ \ldots\ x_{n-1}\ ]$$



## Vector Presets

$a = [\ x_0, x_1, x_2, \ldots\ x_{n-1}\ ]$

$b = [\ x_0, x_1, x_2, \ldots\ x_{n-1}\ ]$

$c = [\ x_0, x_1, x_2, \ldots\ x_{n-1}\ ]$

$d = [\ x_0, x_1, x_2, \ldots\ x_{n-1}\ ]$

## Vector Presets

$a = [\ x_0, x_1, x_2, \dots\ x_{n-1}\ ]$

$b = [\ x_0, x_1, x_2, \dots\ x_{n-1}\ ]$

$c = [\ x_0, x_1, x_2, \dots\ x_{n-1}\ ]$

$d = [\ x_0, x_1, x_2, \dots\ x_{n-1}\ ]$

$e = [\ x_0, x_1, x_2, \dots\ x_{n-1}\ ]$

$\vdots$

Vector Presets

$a = [\ x_0, x_1, x_2, \ldots\ x_{n-1}\ ]$

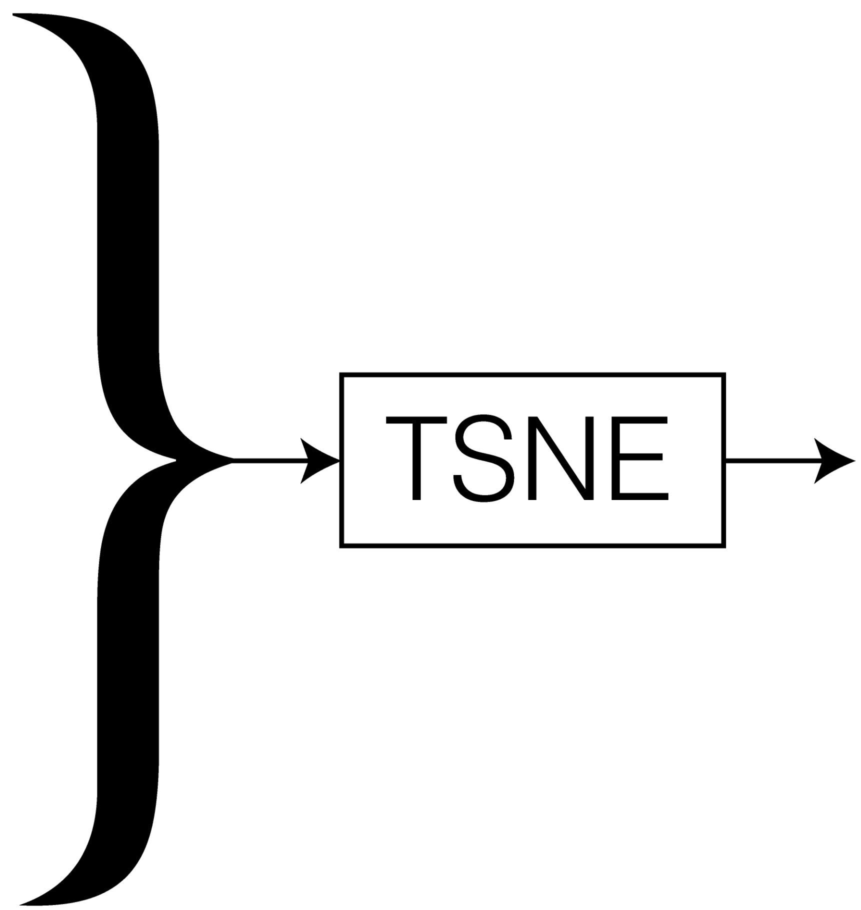
$b = [\ x_0, x_1, x_2, \ldots\ x_{n-1}\ ]$

$c = [\ x_0, x_1, x_2, \ldots\ x_{n-1}\ ]$

$d = [\ x_0, x_1, x_2, \ldots\ x_{n-1}\ ]$

$e = [\ x_0, x_1, x_2, \ldots\ x_{n-1}\ ]$

$\vdots$



# TSNE

- t-Distributed Stochastic Neighbor Embedding
- Dimensionality Reduction Algorithm
- Vectors that are similar in high dimensional space are embedded near each other in low dimensional space, while vectors dissimilar in high dimensional space are embedded far away in low dimensional space

```
//https://github.com/karpathy/tsnejs/blob/master/tsne.js

// create main global object
TSNE {
    var /*return_v,
    v_val,*/
    iter,
    perplexity,
    dim,
    epsilon,
    sizeOfDataSet,
    <y,
    gains,
    ystep;

    *new {
        arg perplexity = 30, dim = 2, epsilon = 10;
        ^super.new.init(perplexity, dim, epsilon);
    }

    init {
        arg perplexity_ = 30, dim_ = 2, epsilon_ = 10;

        perplexity = perplexity_; // effective number of nearest neighbors
        dim = dim_; // by default 2-D tSNE
        epsilon = epsilon_; // learning rate
```

## Vector Presets

$a = [\ x_0,\ x_1,\ x_2,\ \ldots\ x_{n-1}\ ]$

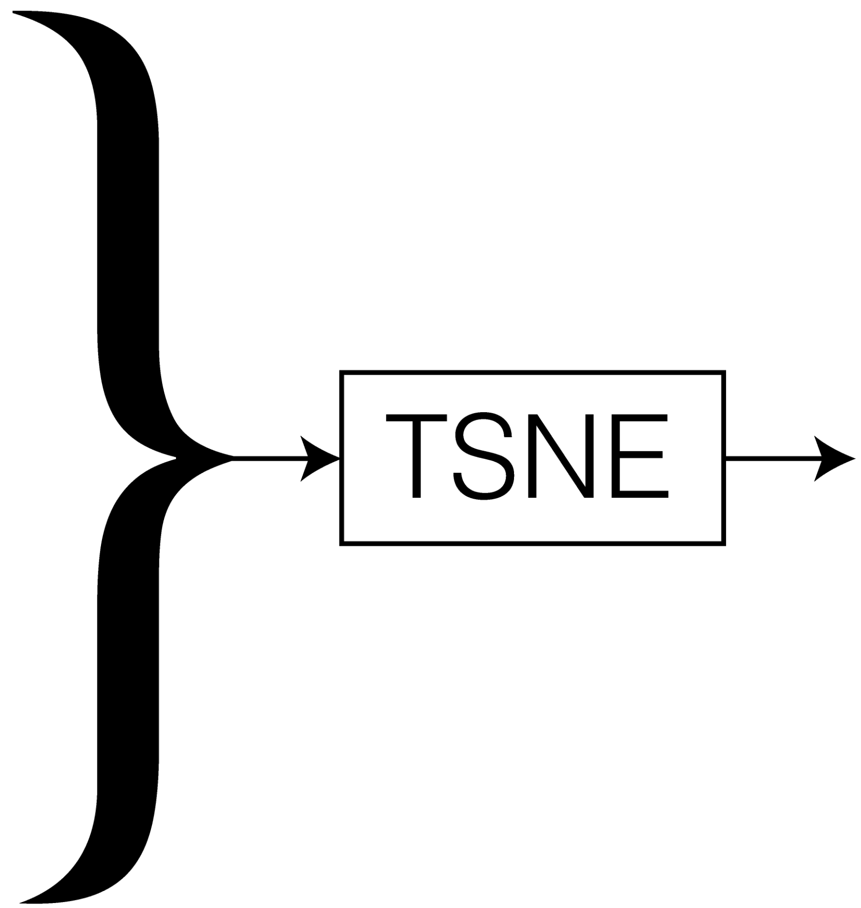
$b = [\ x_0,\ x_1,\ x_2,\ \ldots\ x_{n-1}\ ]$

$c = [\ x_0,\ x_1,\ x_2,\ \ldots\ x_{n-1}\ ]$

$d = [\ x_0,\ x_1,\ x_2,\ \ldots\ x_{n-1}\ ]$

$e = [\ x_0,\ x_1,\ x_2,\ \ldots\ x_{n-1}\ ]$

$\vdots$



# Vector Presets

$$a = [\ x_0, x_1, x_2, \ldots\ x_{n-1}\ ]$$

$$b = [\ x_0, x_1, x_2, \ldots\ x_{n-1}\ ]$$

$$c = [\ x_0, x_1, x_2, \ldots\ x_{n-1}\ ]$$

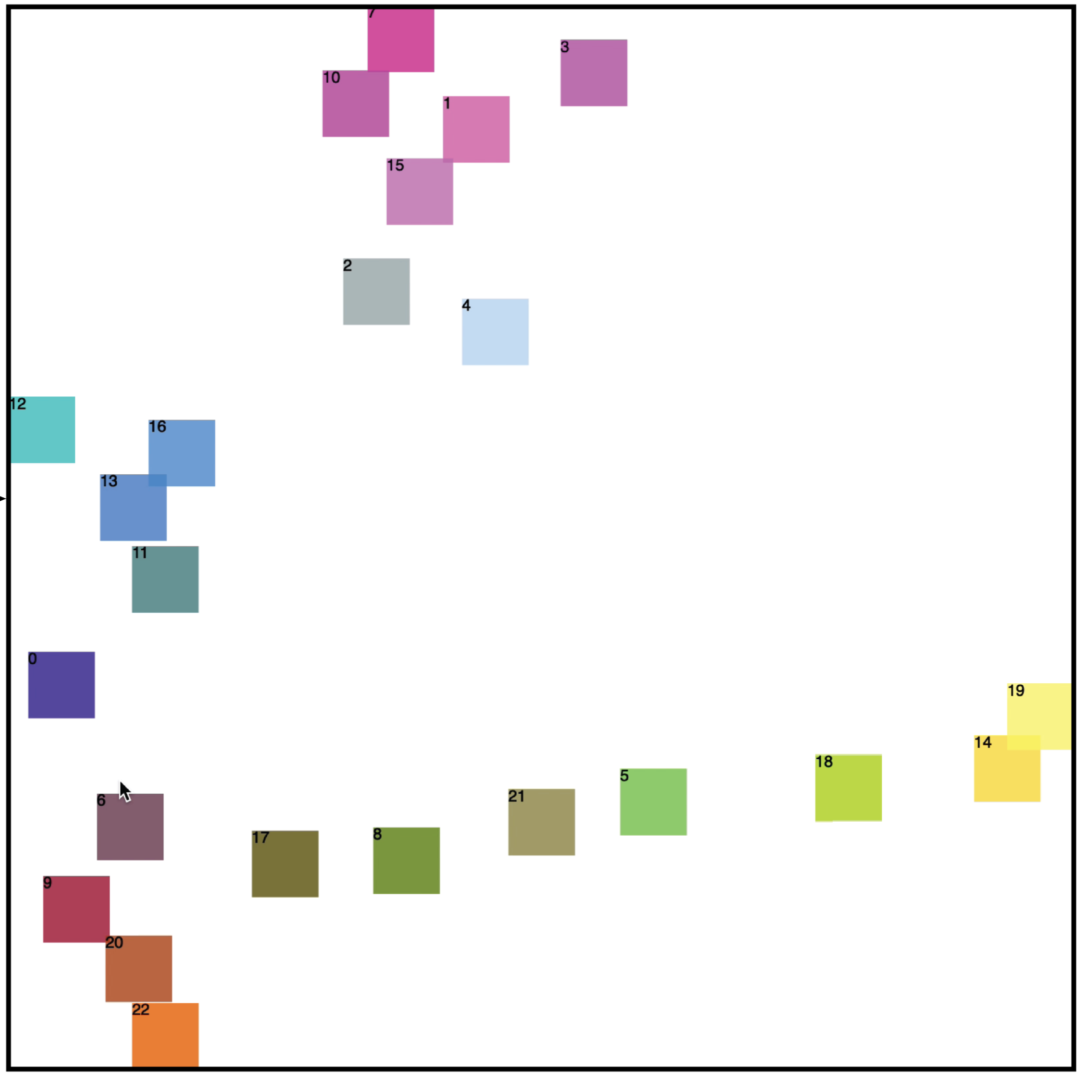
$$d = [\ x_0, x_1, x_2, \ldots\ x_{n-1}\ ]$$

$$e = [\ x_0, x_1, x_2, \ldots\ x_{n-1}\ ]$$

$$\vdots$$

TSNE

# TSNE Solution



## Vector Presets

$a = [\ x_0, x_1, x_2, \ldots\ x_{n-1}\ ]$

$b = [\ x_0, x_1, x_2, \ldots\ x_{n-1}\ ]$

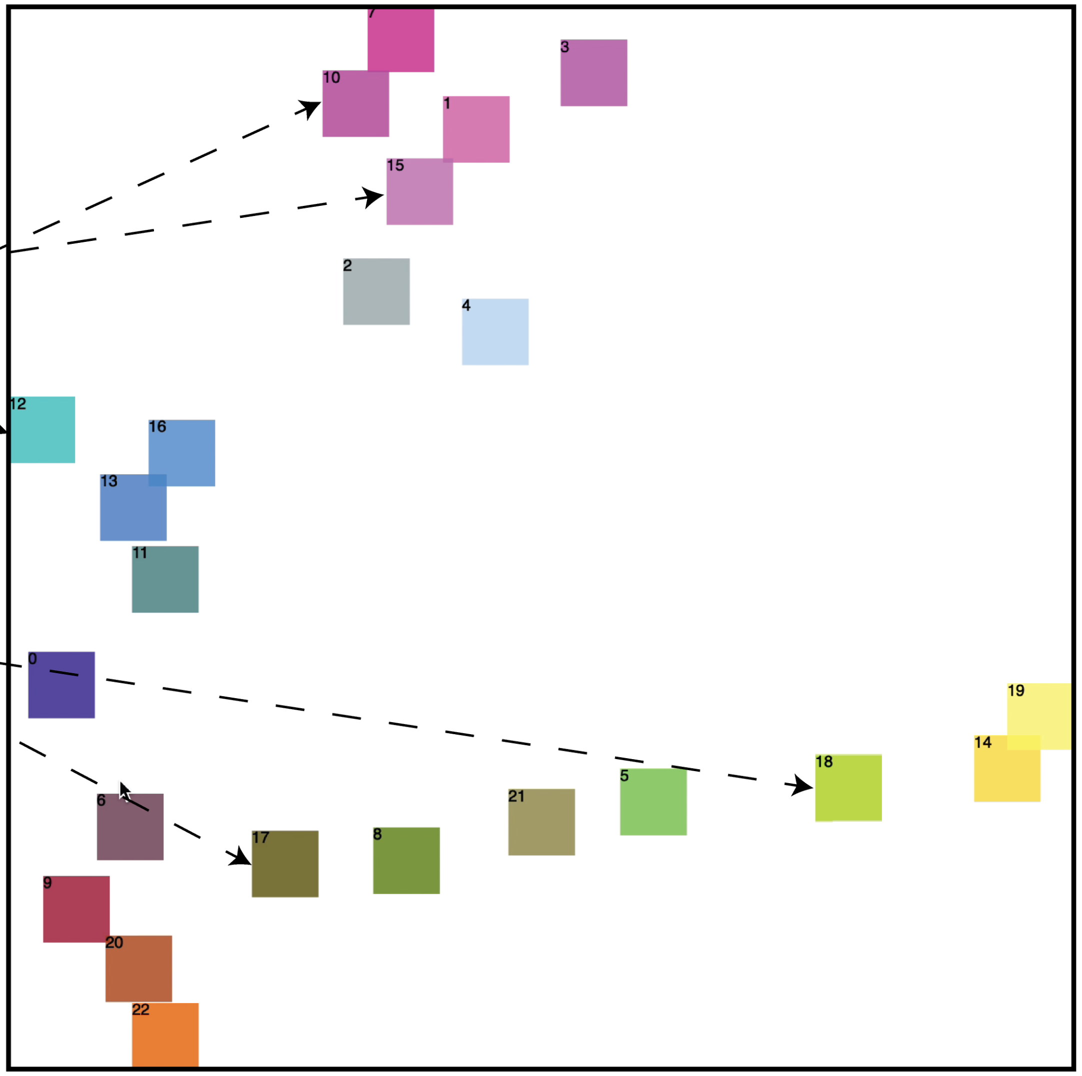
$c = [\ x_0, x_1, x_2, \ldots\ x_{n-1}\ ]$

$d = [\ x_0, x_1, x_2, \ldots\ x_{n-1}\ ]$

$e = [\ x_0, x_1, x_2, \ldots\ x_{n-1}\ ]$

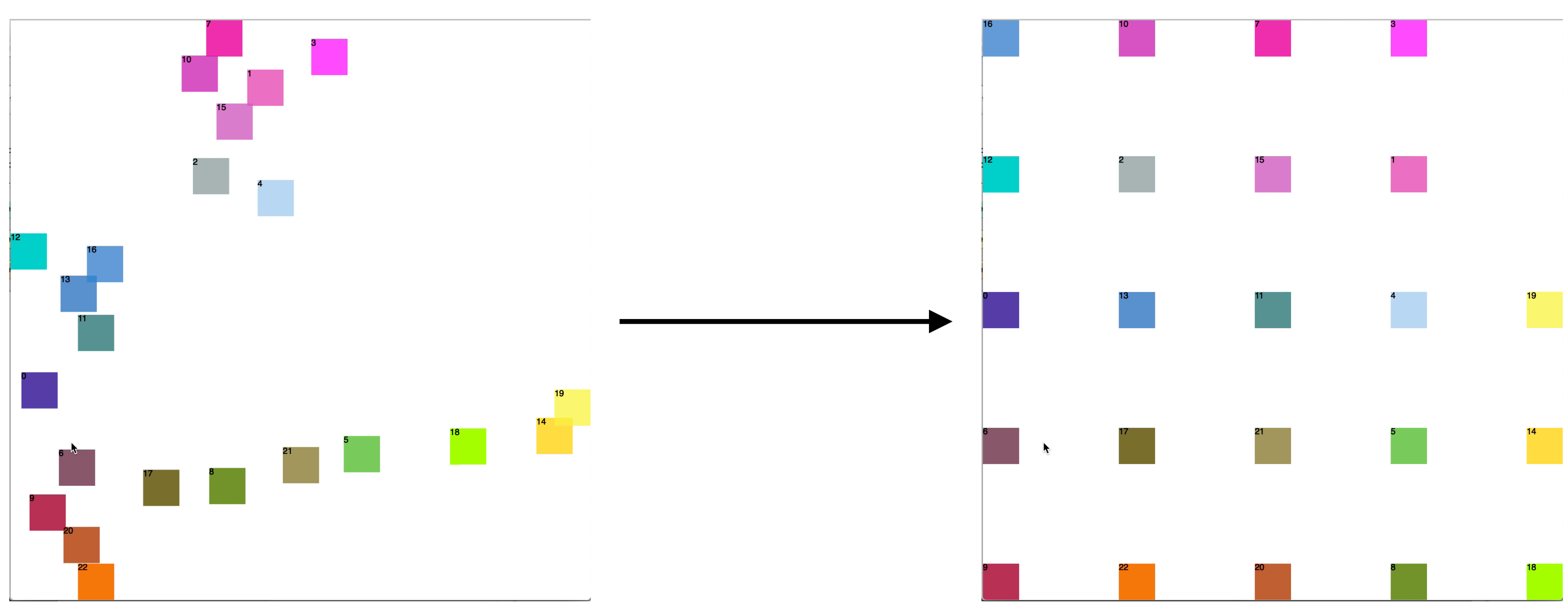
$\vdots$

## TSNE Solution



# Munkres Algorithm

- aka “Hungarian Algorithm” or “Kuhn-Munkres Algorithm”
- Optimal solution to linear assignment problem
- Every element in the **t-SNE embeddings** must be assigned to one unique element in the **grid of locations**



# Vector Presets

$a = [\ x_0,\ x_1,\ x_2,\ \ldots\ x_{n-1}\ ]$

$b = [\ x_0,\ x_1,\ x_2,\ \ldots\ x_{n-1}\ ]$

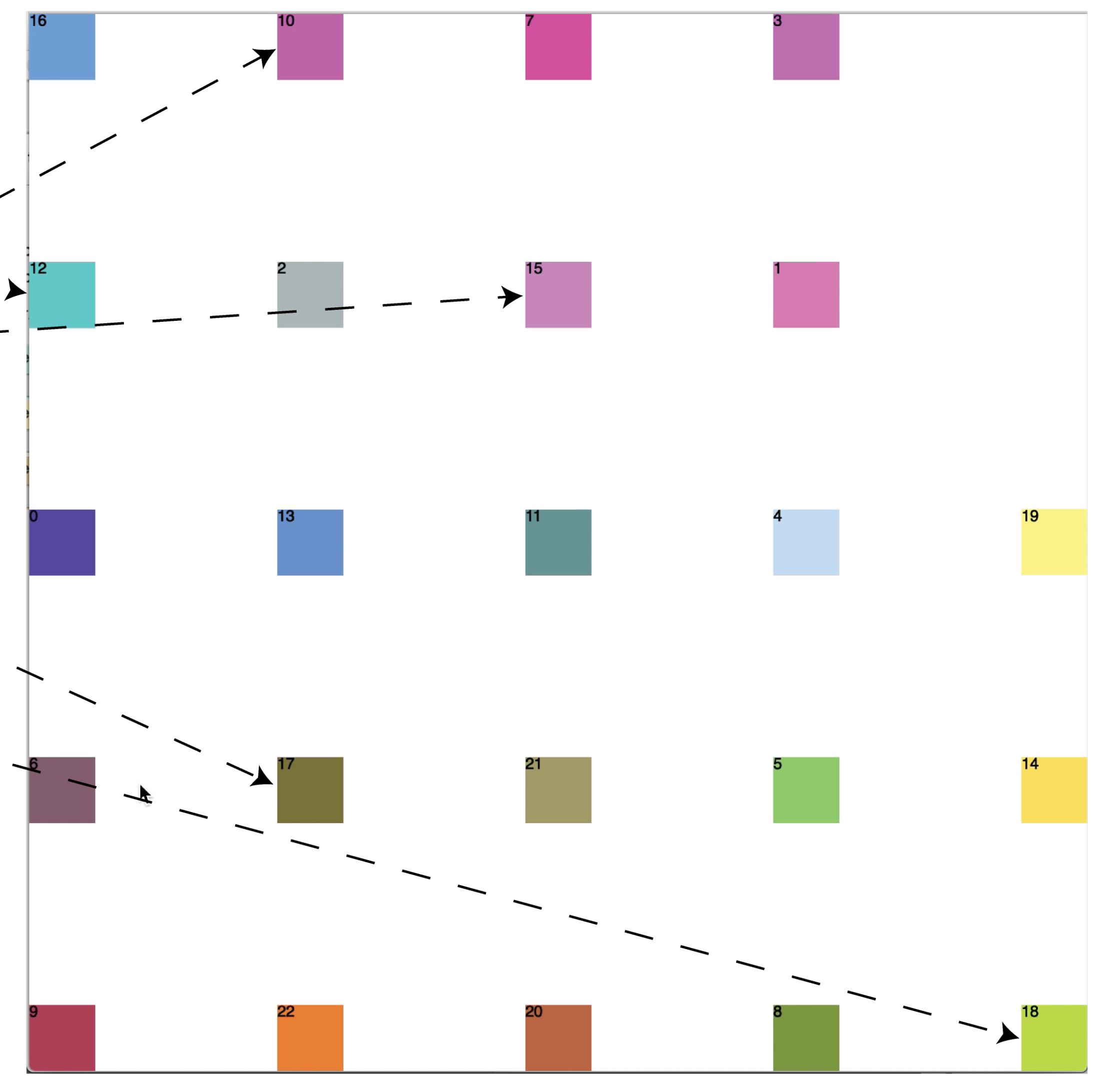
$c = [\ x_0,\ x_1,\ x_2,\ \ldots\ x_{n-1}\ ]$

$d = [\ x_0,\ x_1,\ x_2,\ \ldots\ x_{n-1}\ ]$

$e = [\ x_0,\ x_1,\ x_2,\ \ldots\ x_{n-1}\ ]$

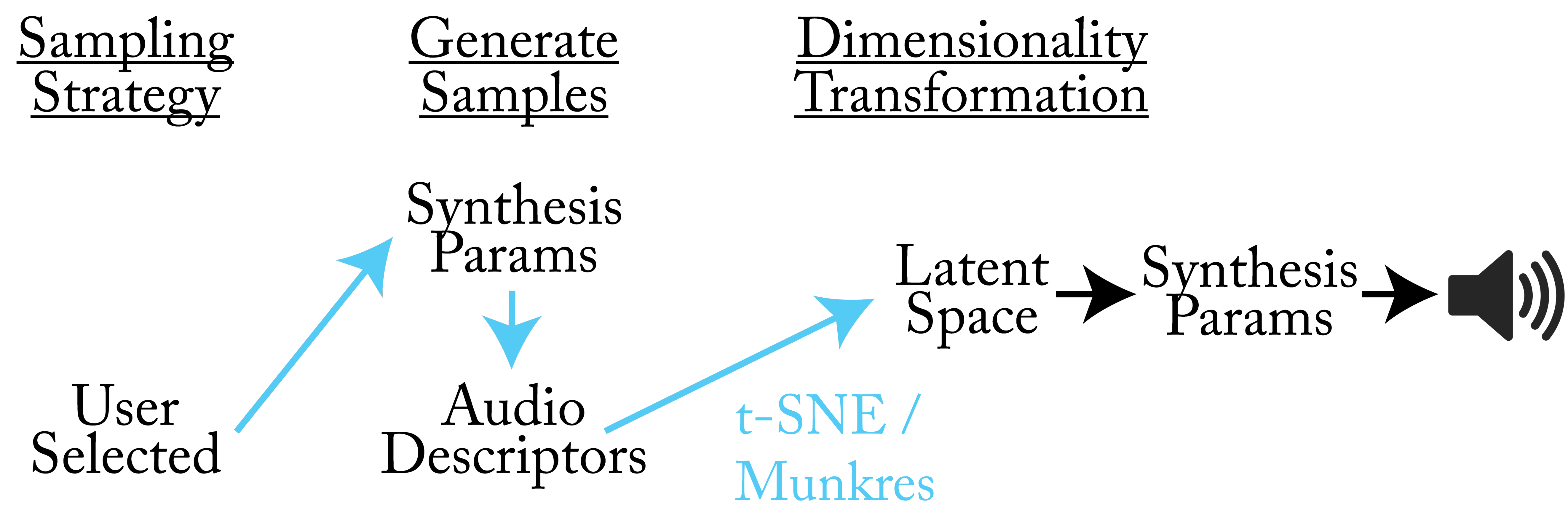
$\vdots$

# Munkres Solution

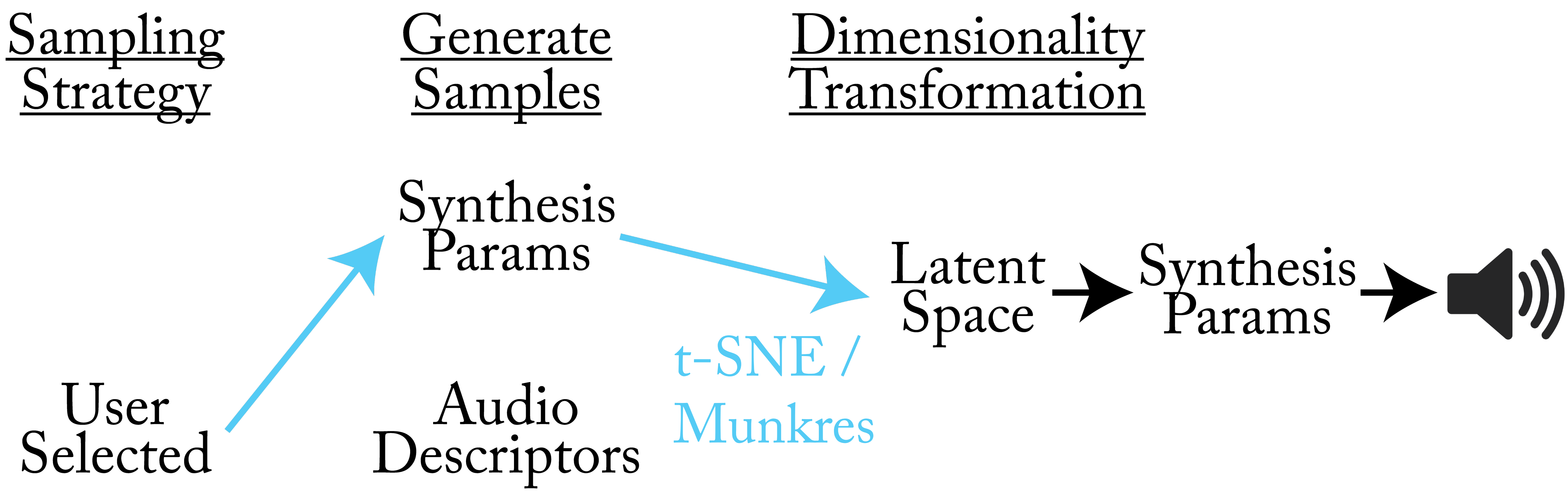


# t-SNE / Munkres demo

## User-Defined (no stochasticity)



**User-Defined (has stochasticity)**



# t-SNE / Munkres demo 2

# Benefits of TSNE / Munkres approach

- Preserves user-defined presets
- TSNE recognized as superior dimensionality reduction
- Munkres finds optimal solution
- “Non-linear” latent space requires practice to learn

# Rejected Alternatives

- Neural Network - supervised learning requires knowing the desired low dimensional structure before training
    - Neural Networks generally need a lot of data
- Self-Organizing Maps - doesn't guarantee that exact user-defined presets are preserved

# Reflections

| | | |
|---|---|---|
| **quick** | ✅ | |
| **interpolated** | ✅ | |
| **gestural** | ✅ | |
| **musical** | **?** | ***more practice needed*** |

# Reflections

| **Indexical Control** | **Symbolic Control** |
|---|---|
| Interface maps to<br>The instrument | Mapping is arbitrary |
| filter cutoff knob | latent space |

**Previous Strategy** $\xrightarrow[\textit{reduction}]{\textit{dimensionality}}$ **ML Strategy**

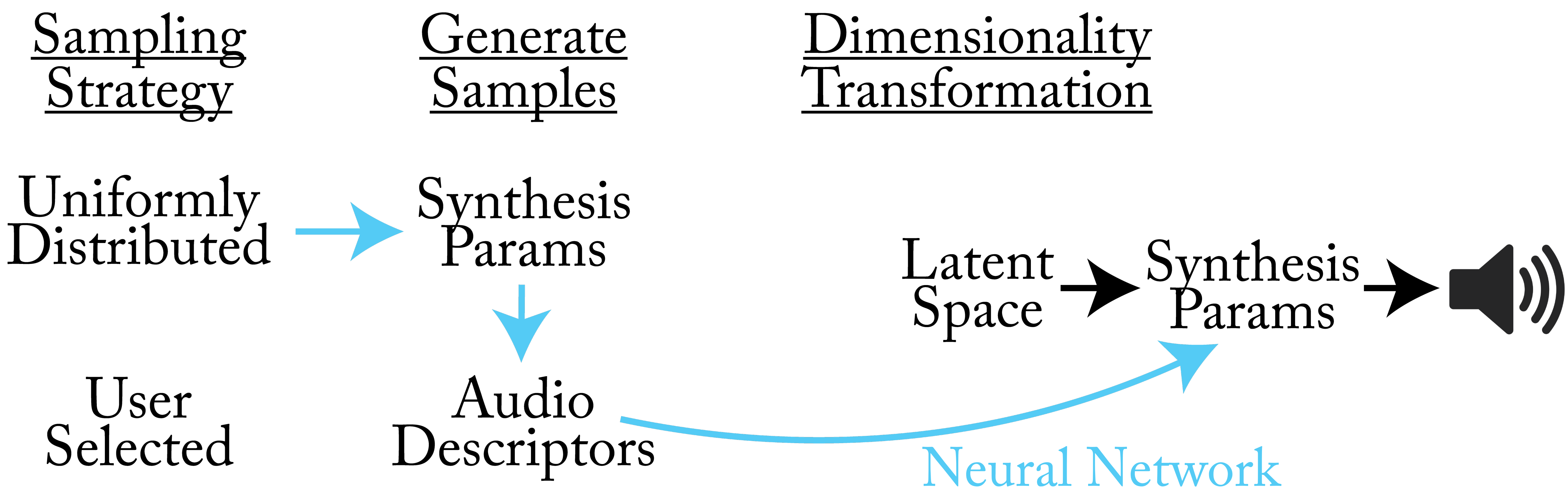
***flow state*** — ***no flow state***

*practice*

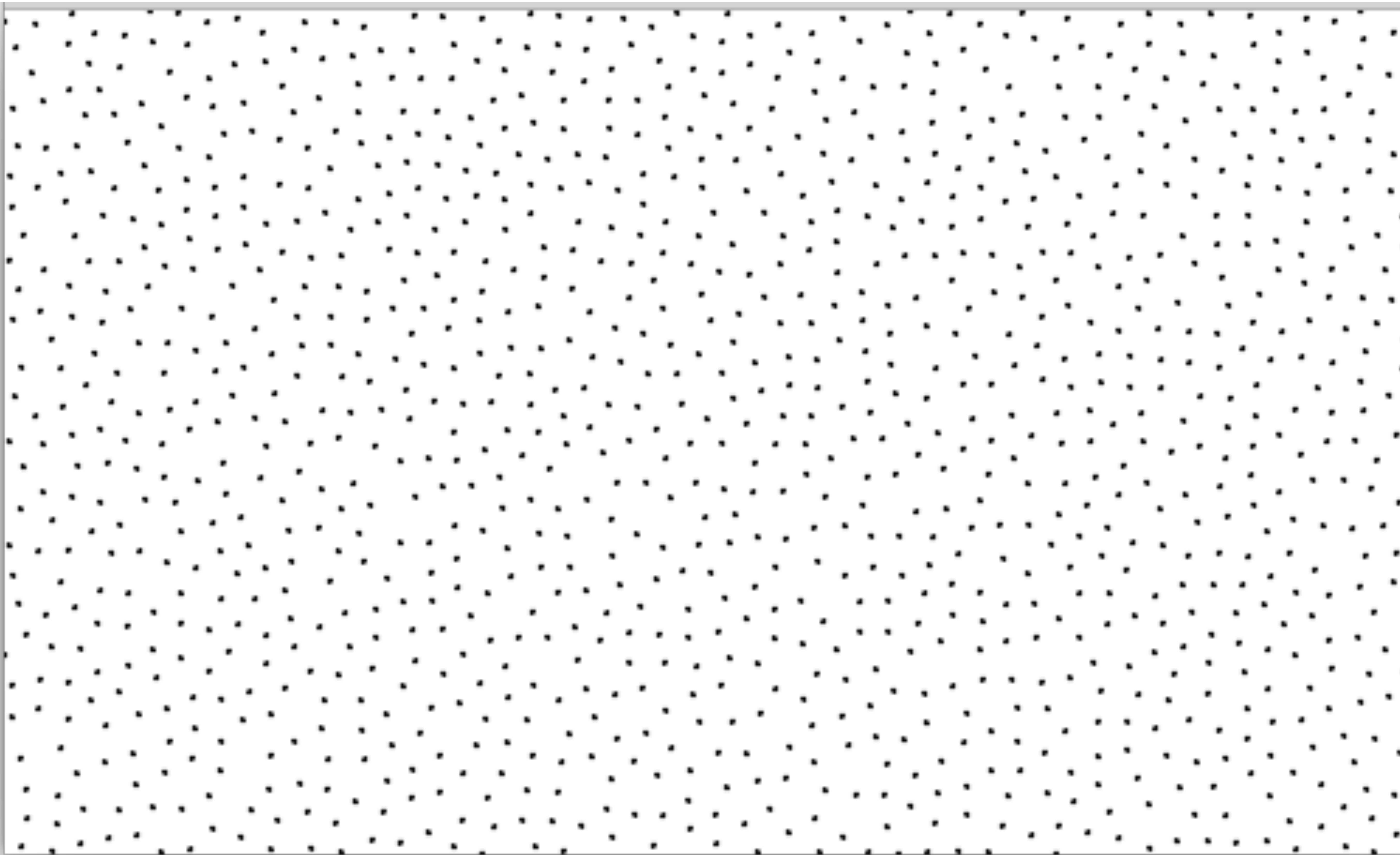
Pierce (1955), Magnusson (2018)

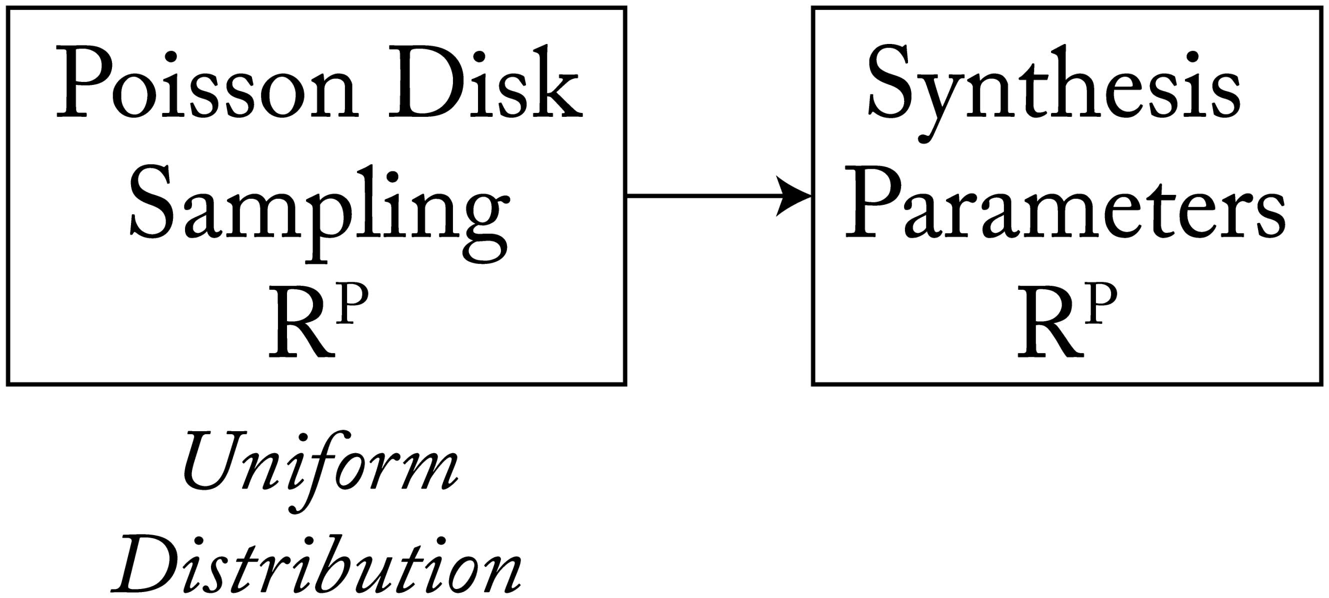
# Learned Synthesis Parameters

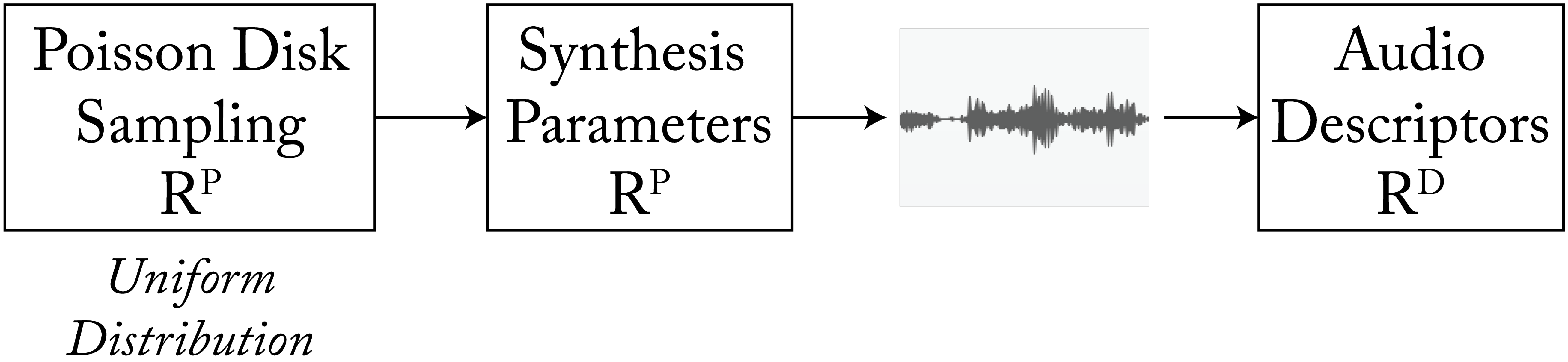
## Uniformly Distributed

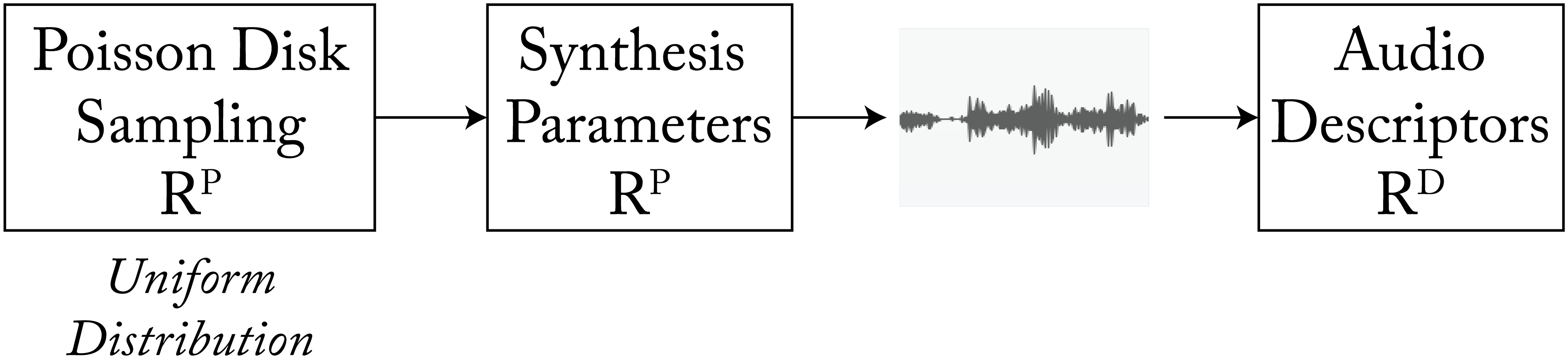


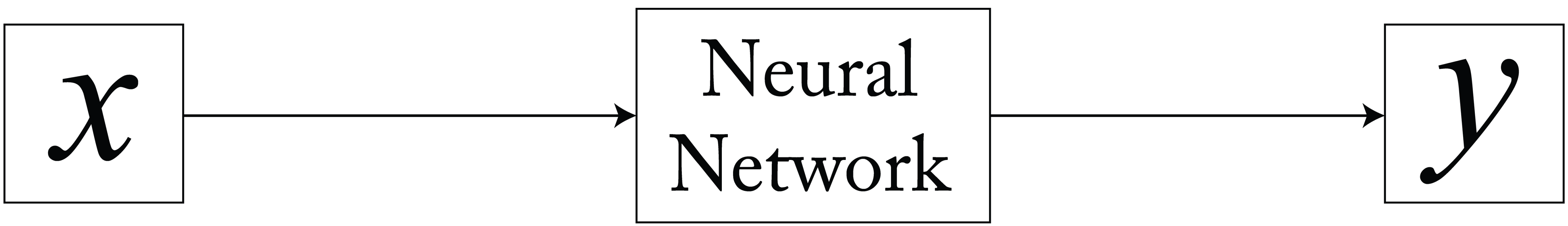
# Poisson Disk Sampling

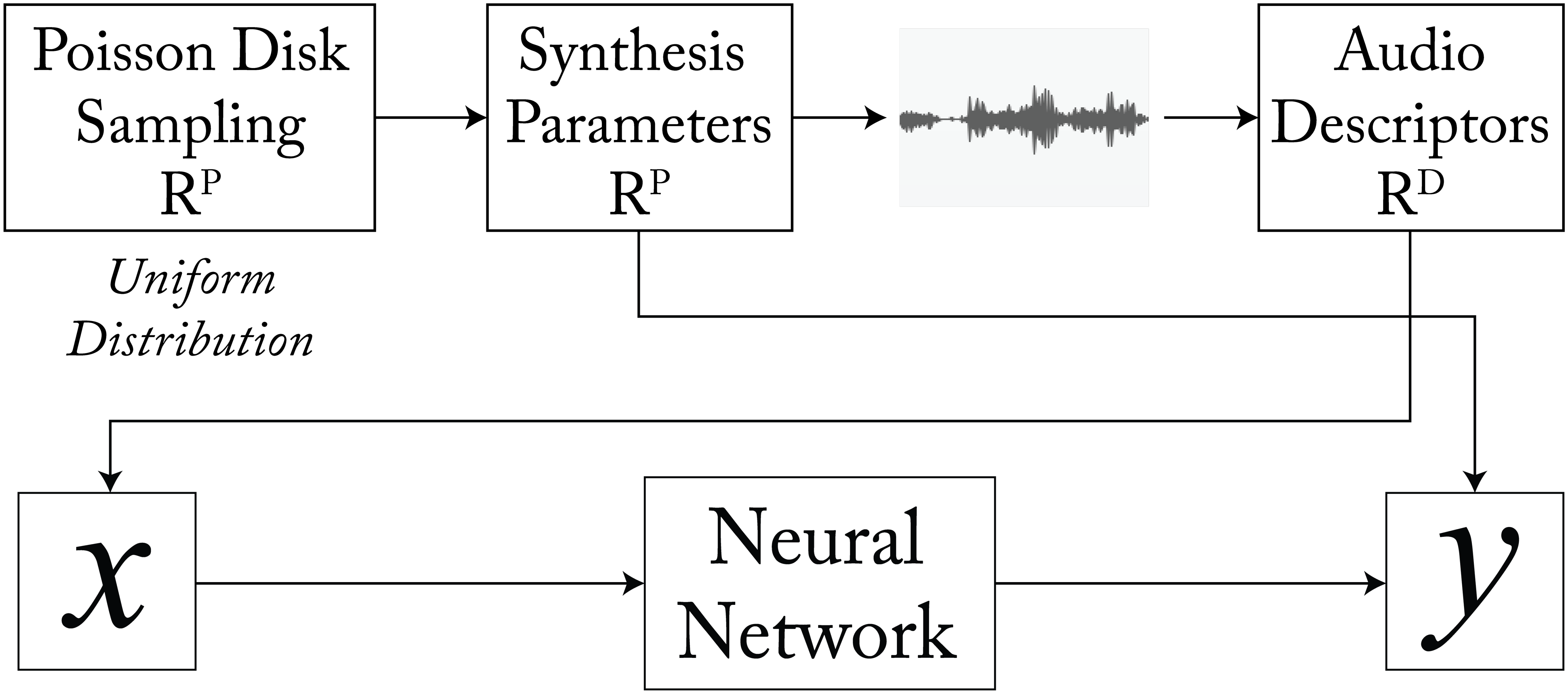


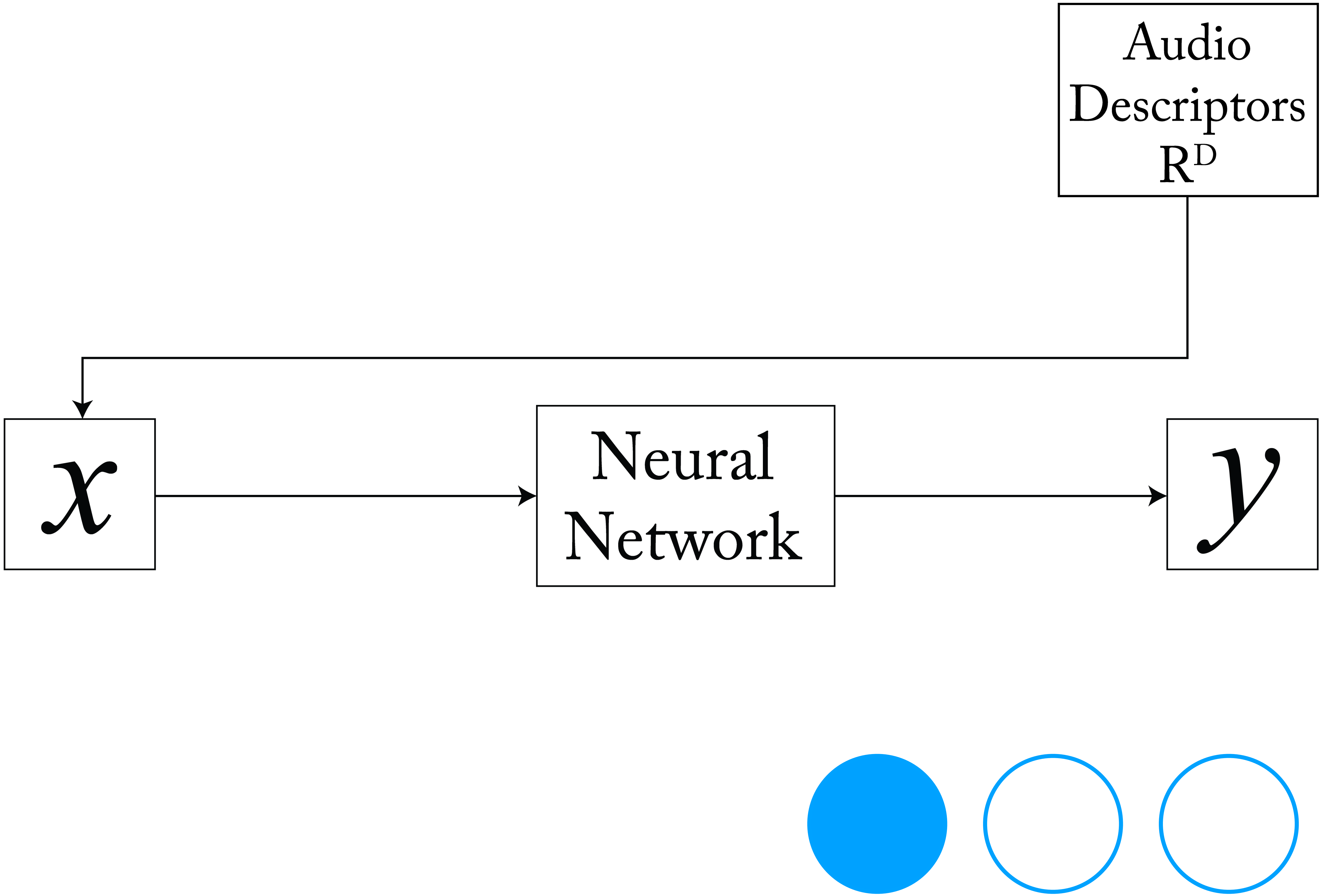

Poisson Disk Sampling
R^P
Uniform Distribution
Synthesis Parameters
R^P

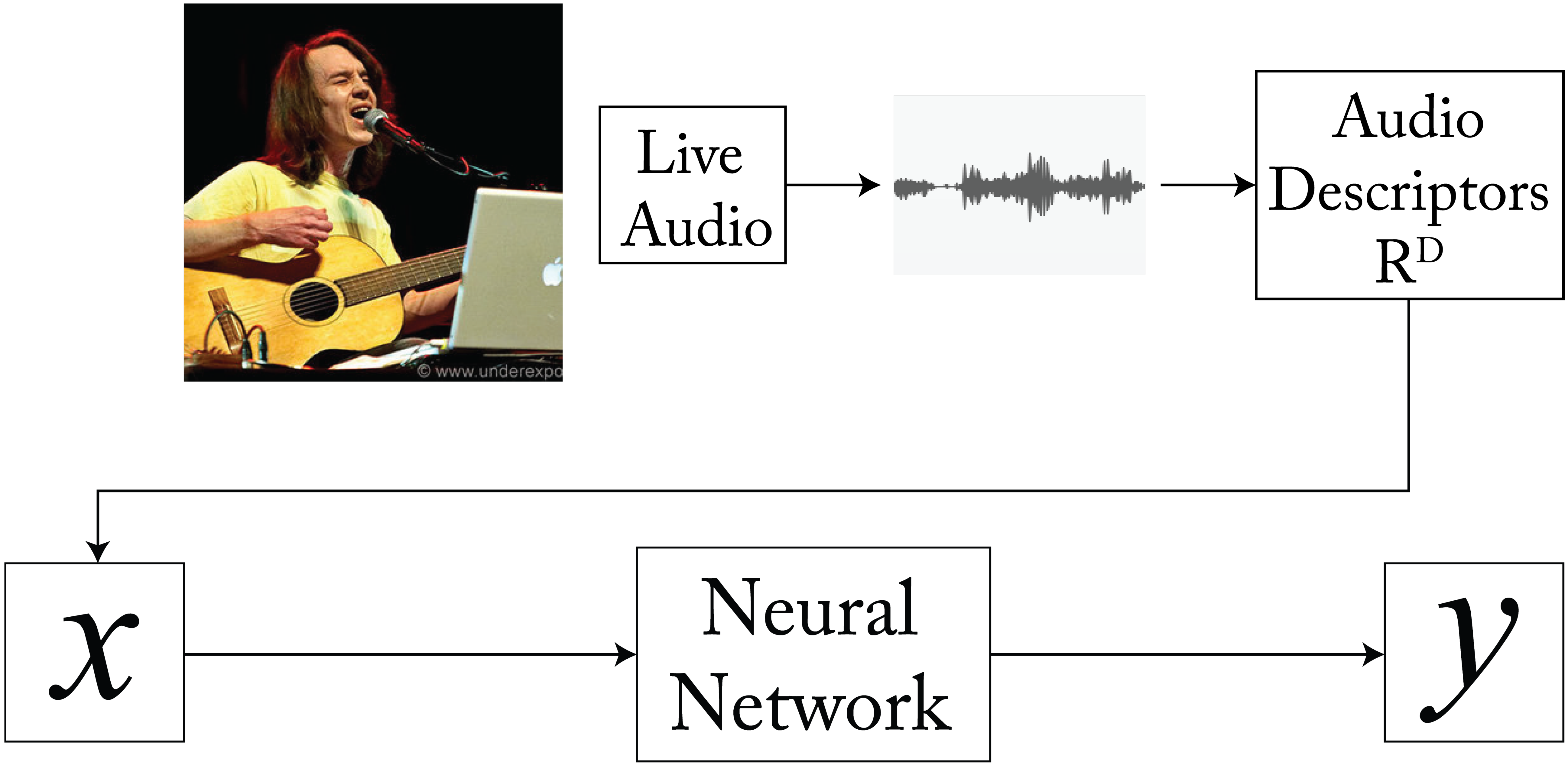

Poisson Disk Sampling $R^P$
Uniform Distribution
Synthesis Parameters $R^P$
Audio Descriptors $R^D$

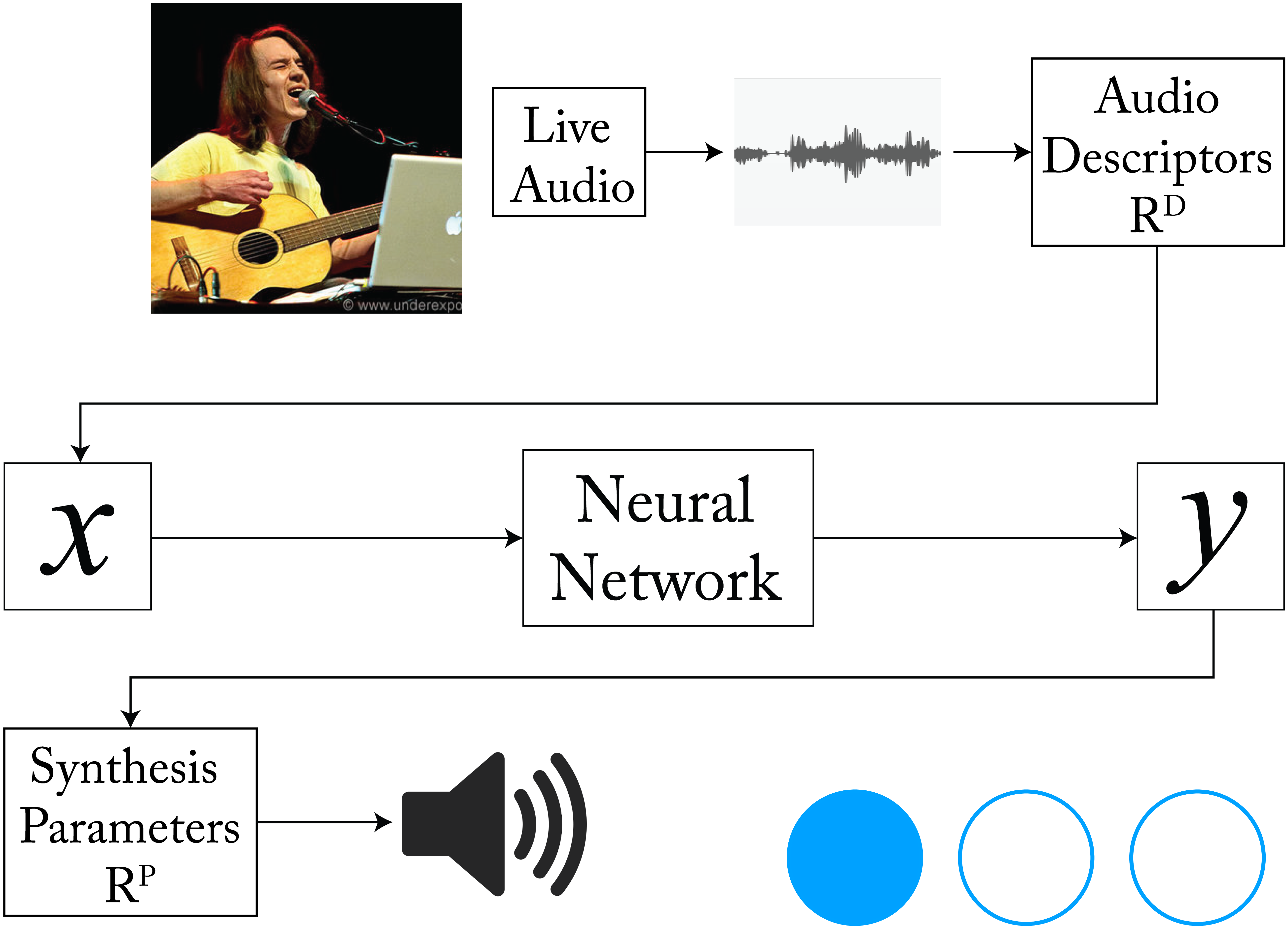

Poisson Disk Sampling $R^P$
Uniform Distribution
Synthesis Parameters $R^P$
Audio Descriptors $R^D$


$x$
Neural Network
$y$


Poisson Disk Sampling
$R^P$
Uniform Distribution
Synthesis Parameters
$R^P$
Audio Descriptors
$R^D$
$x$
Neural Network
$y$


Audio Descriptors $R^D$
$x$
Neural Network
$y$


Live Audio
Audio Descriptors $R^D$
$x$
Neural Network
$y$


© www.underexpo
Live Audio
Audio Descriptors $R^D$
$x$
Neural Network
$y$
Synthesis Parameters $R^P$

demo 3

# the Future

- More control strategies
- More practice performing
- Find more appropriate sound generators
- More audio descriptor options
- Neural Network learn modular synth
    - for live audio input mimicry
    - for gestural control

Thank you. Questions?

Fasciani, Stefano, and Lonce Wyse. 2012. "Adapting General Purpose Interfaces to Synthesis Engines Using Unsupervised Dimensionality Reduction Techniques and Inverse Mapping from Features to Parameters." *Proceedings of the International Computer Music Conference*, no. June 2016: 467–72.

Magnusson, Thor. 2018. "Ergomimesis Towards a Language Describing Instrumental Transductions." *ICLI PORTO 2018*.

Roma, Gerard, Owen Green, and Pierre Alexandre Tremblay. 2019. "Adaptive Mapping of Sound Collections for Data-Driven Musical Interfaces." *Proceedings of th International Conference on New Interfaces for Musical Expression*, 313–18. http://www.nime.org/proceedings/2019/nime2019_060.pdf.

Puckette, Miller. 2004. "Low-Dimensional Parameter Mapping Using Spectral Envelopes." *Proceedings of the International Computer Music Conference* 2004: 406–8.