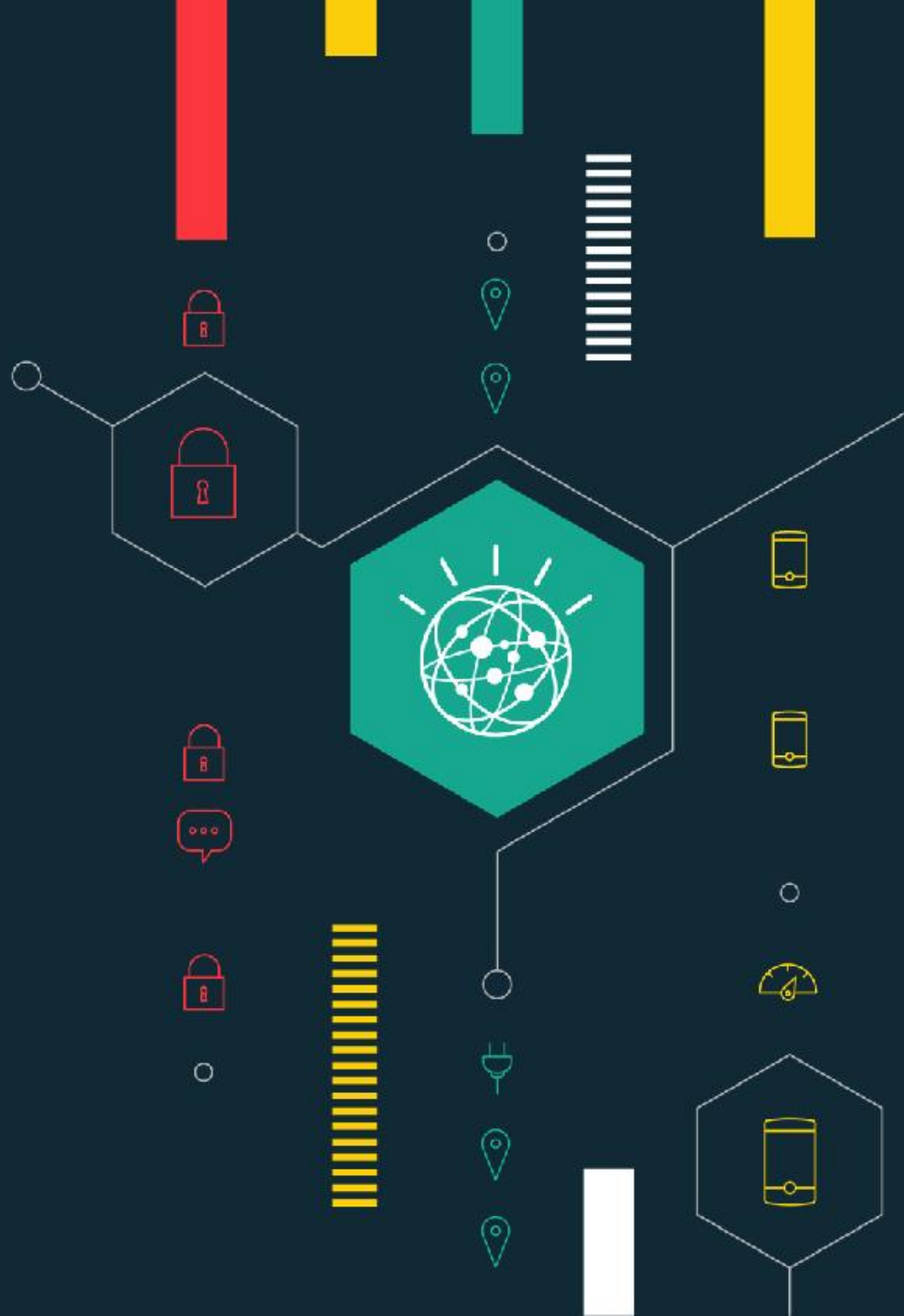
IBM Research | Africa

Cognitive-Assisted Interactive Labeling

Francois Luus
Research Scientist
IBM Research - Africa

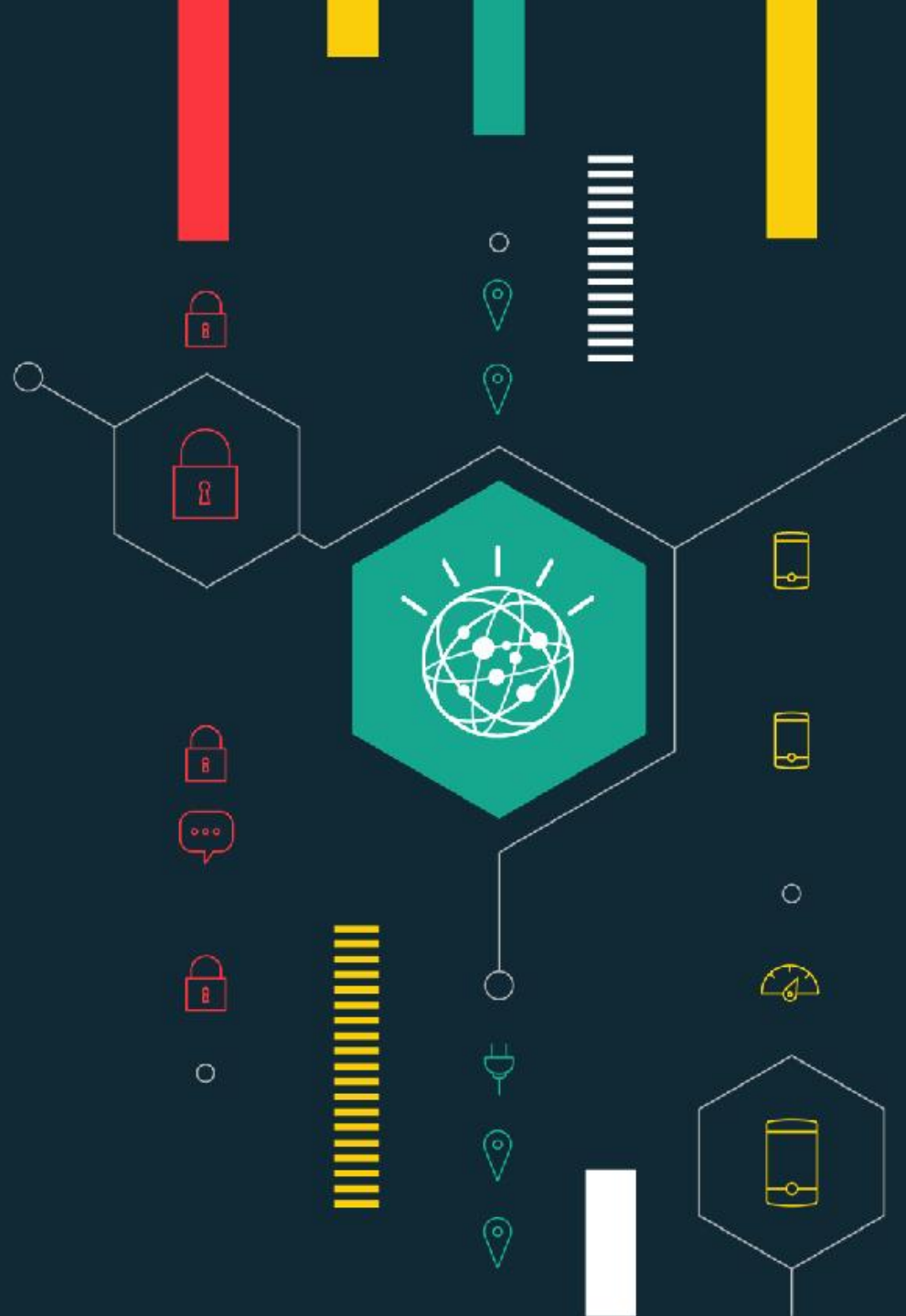
IBM



IBM **Research** | Africa

Overview

1. **SETI RFI signal classification**
2. **RFI signal clustering**
3. **Toolchain for Software 2.0**
4. **Cognitive-assisted interactive labeling**



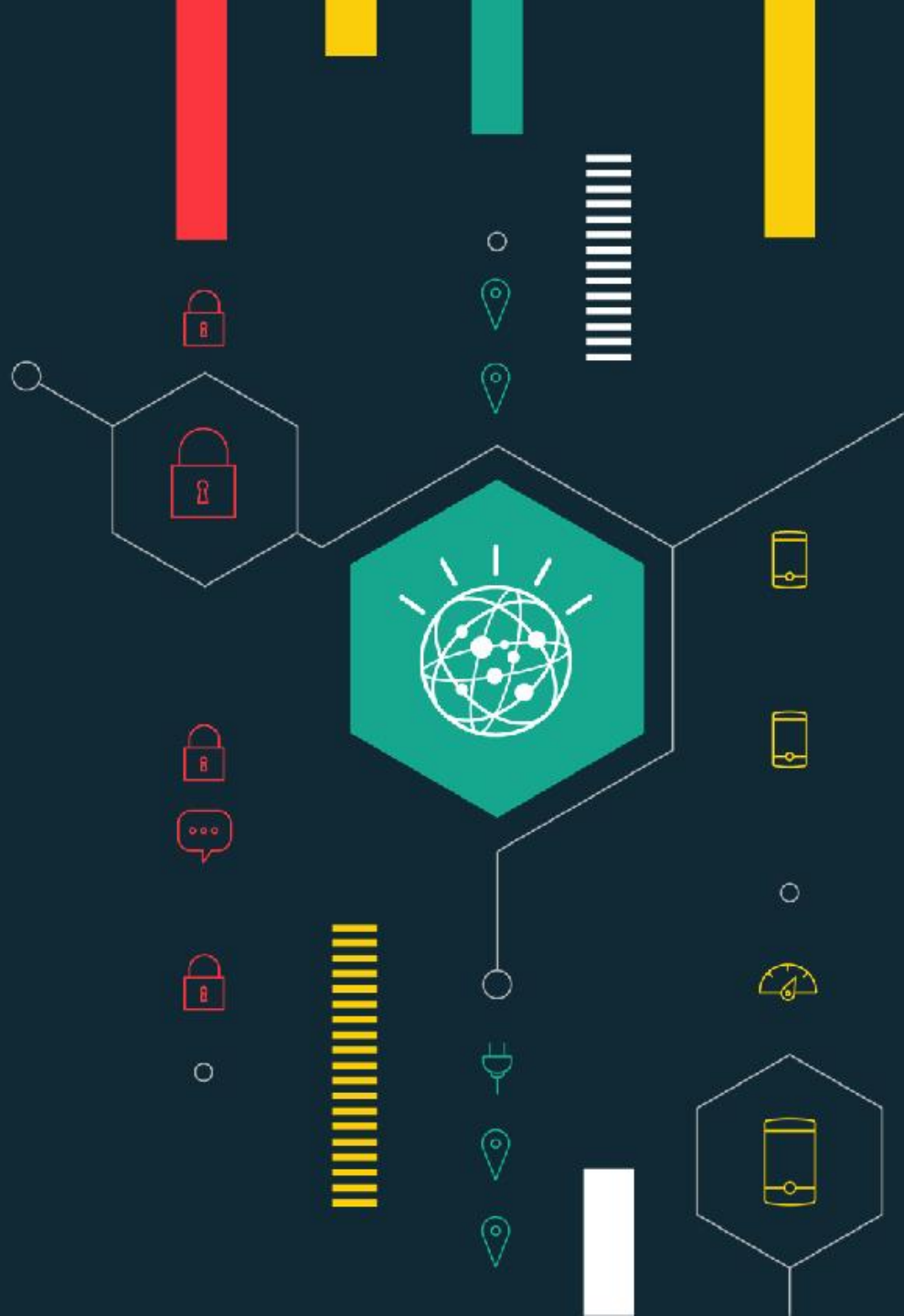
IBM

IBM Research | Africa

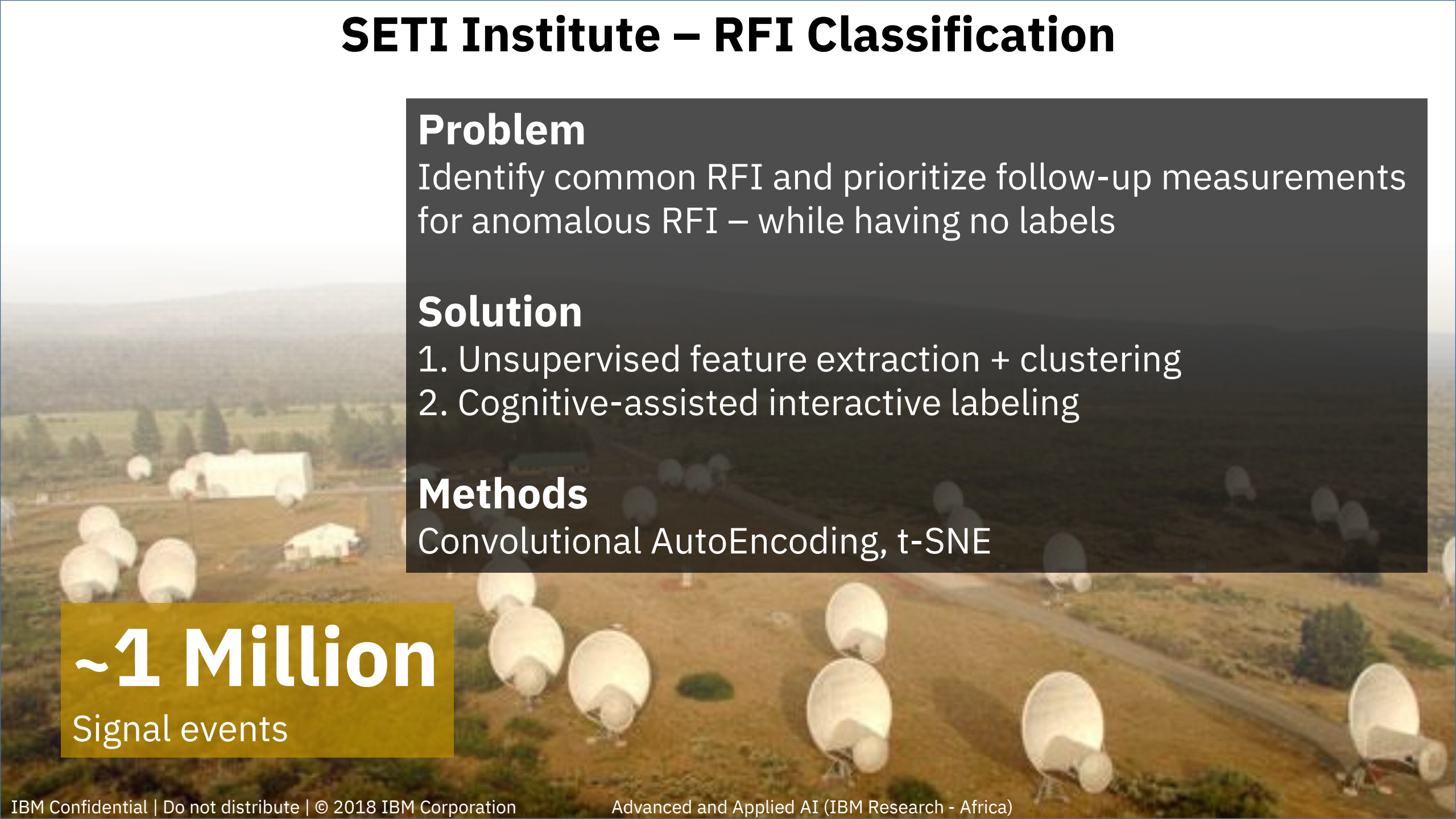
SETI RFI Signal Classification

IBM Research | Africa

IBM



SETI Institute – RFI Classification



Problem

Identify common RFI and prioritize follow-up measurements for anomalous RFI – while having no labels

Solution

1. Unsupervised feature extraction + clustering
2. Cognitive-assisted interactive labeling

Methods

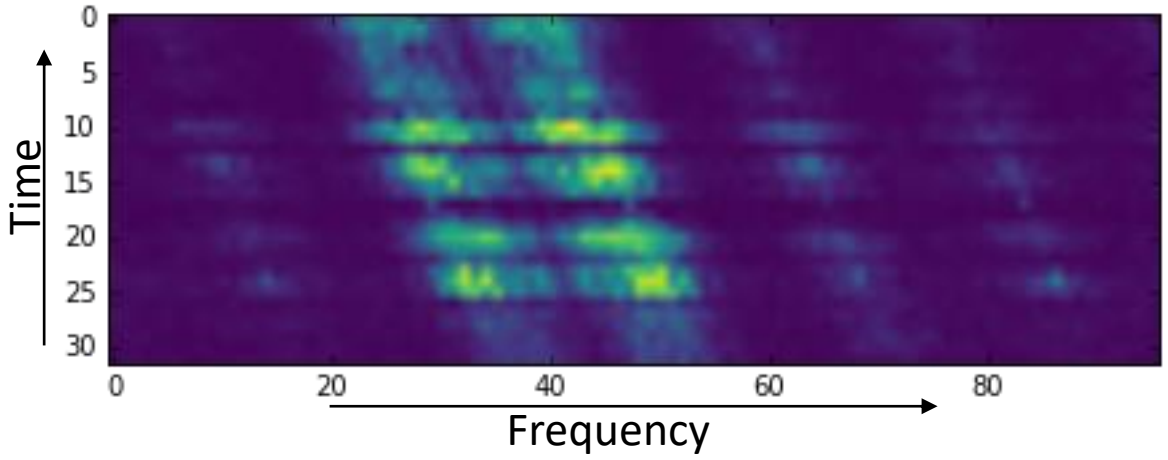
Convolutional AutoEncoding, t-SNE

~1 Million

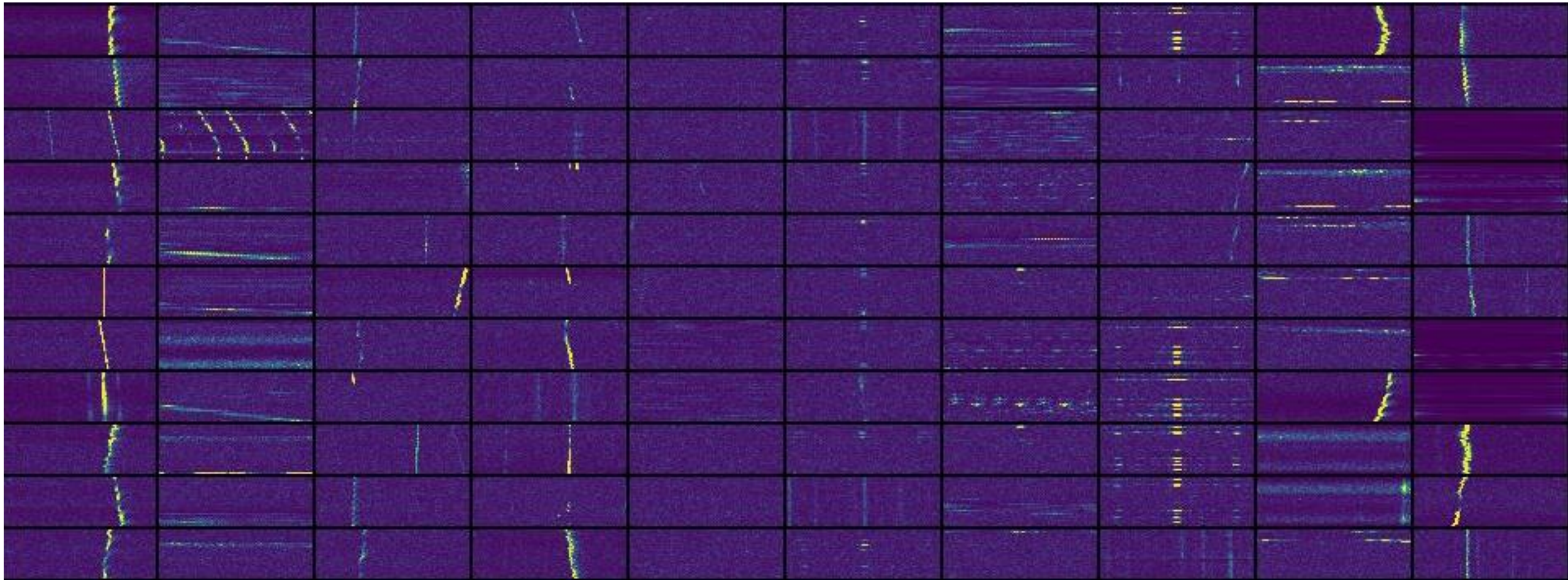
Signal events

Unsupervised RFI Classification

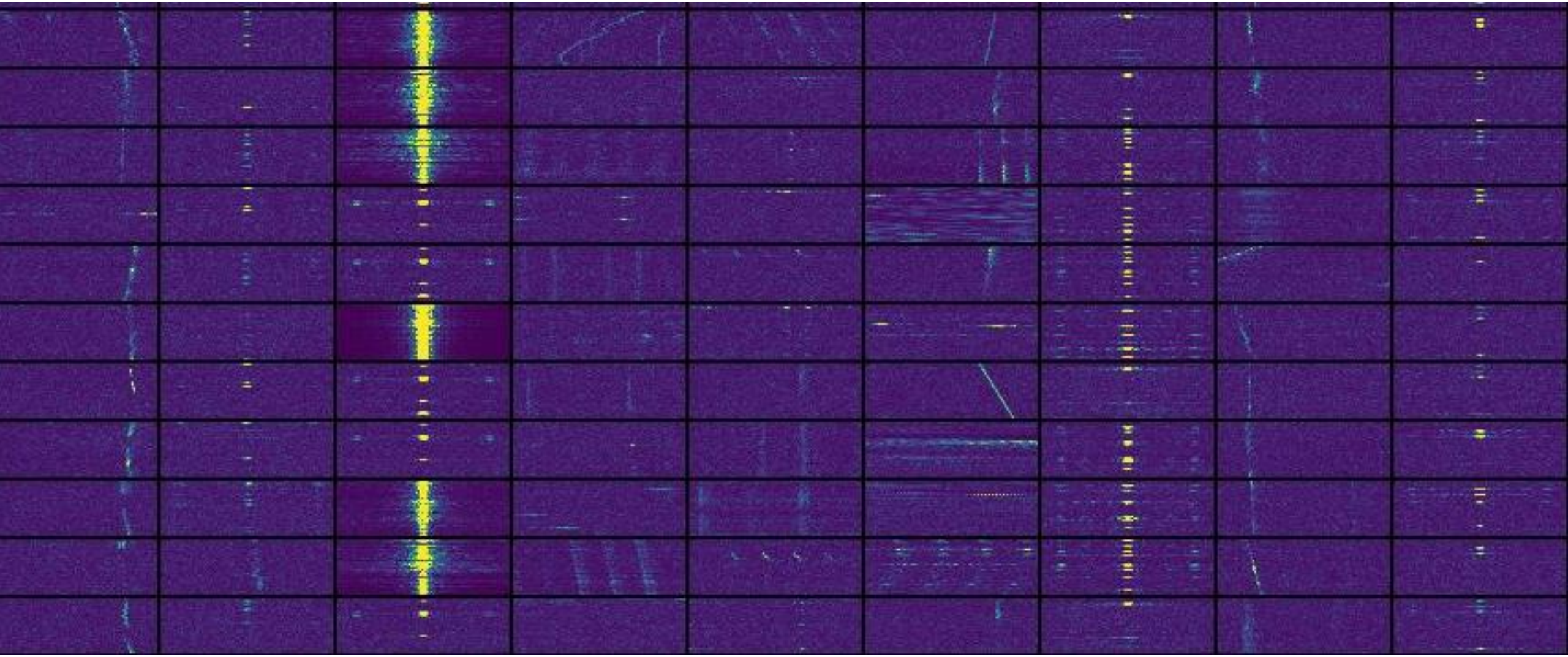
- 14 Million subband spectrogram/images
- 90-95% no signal
- Reoccurring RFI is prevalent
- ~20 common types of RFI spectra
- No labels



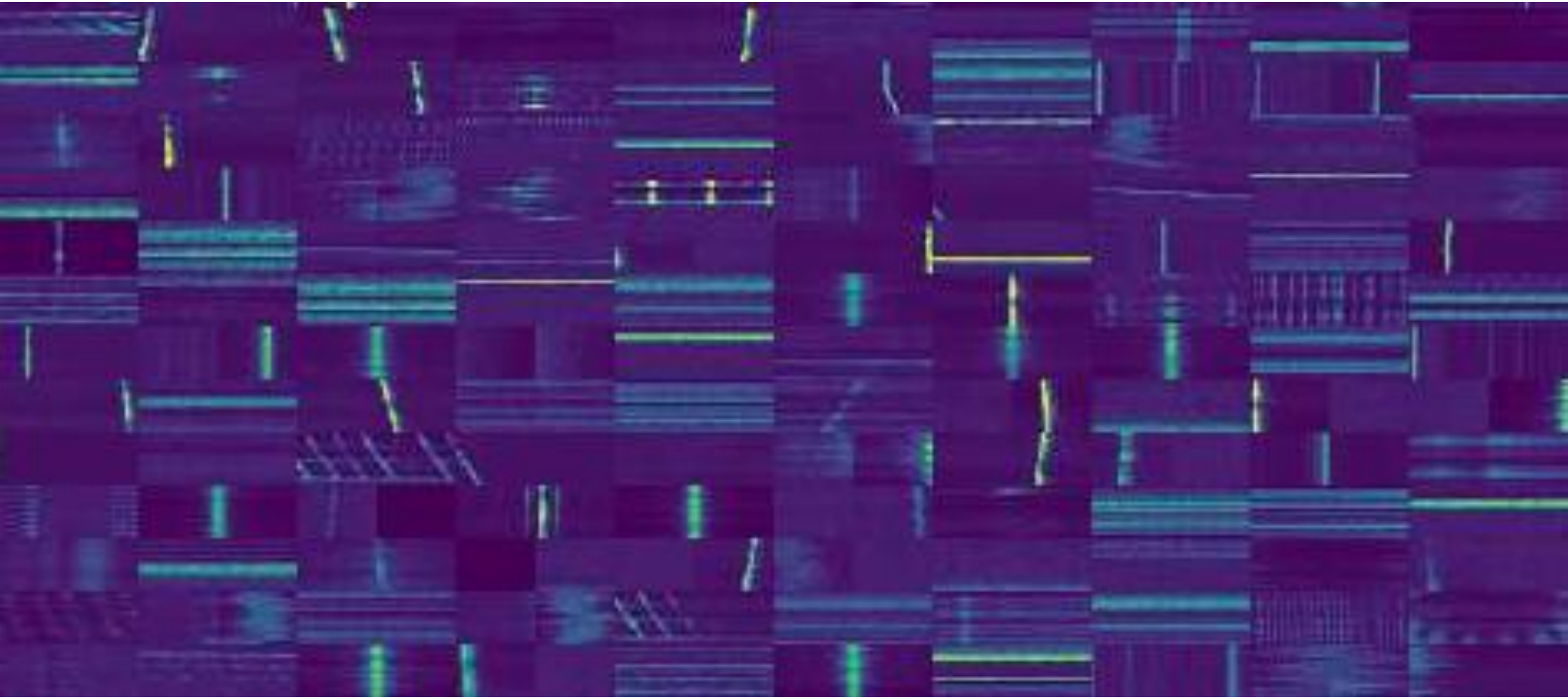
RFI Spectrograms

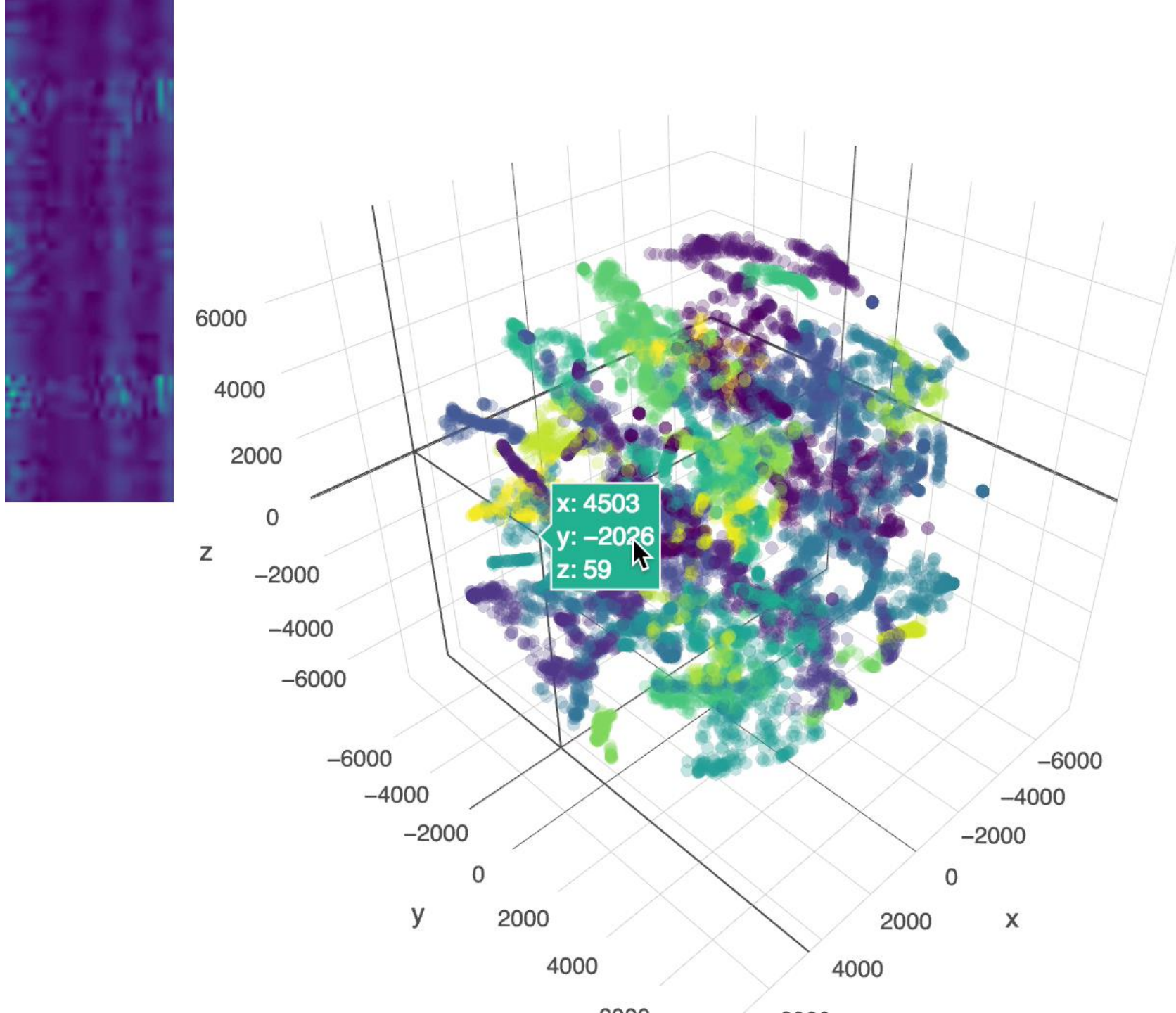


RFI Spectrograms



RFI Spectrograms



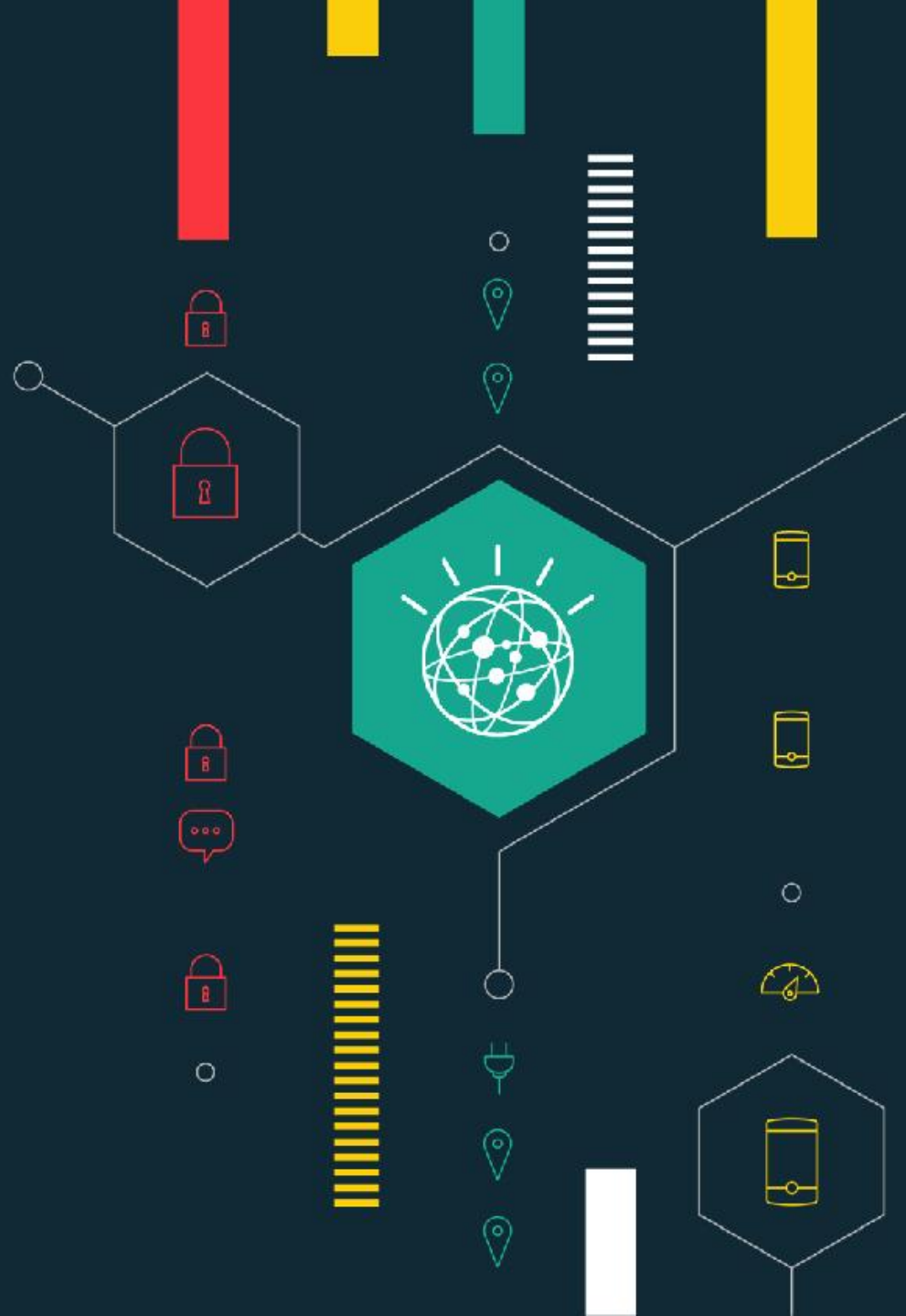

x: 4503
y: −2026
z: 59
6000
4000
2000
0
z
−2000
−4000
−6000
−6000
−4000
−2000
0
y
2000
4000
−6000
−4000
−2000
0
2000
x
4000

IBM Research | Africa

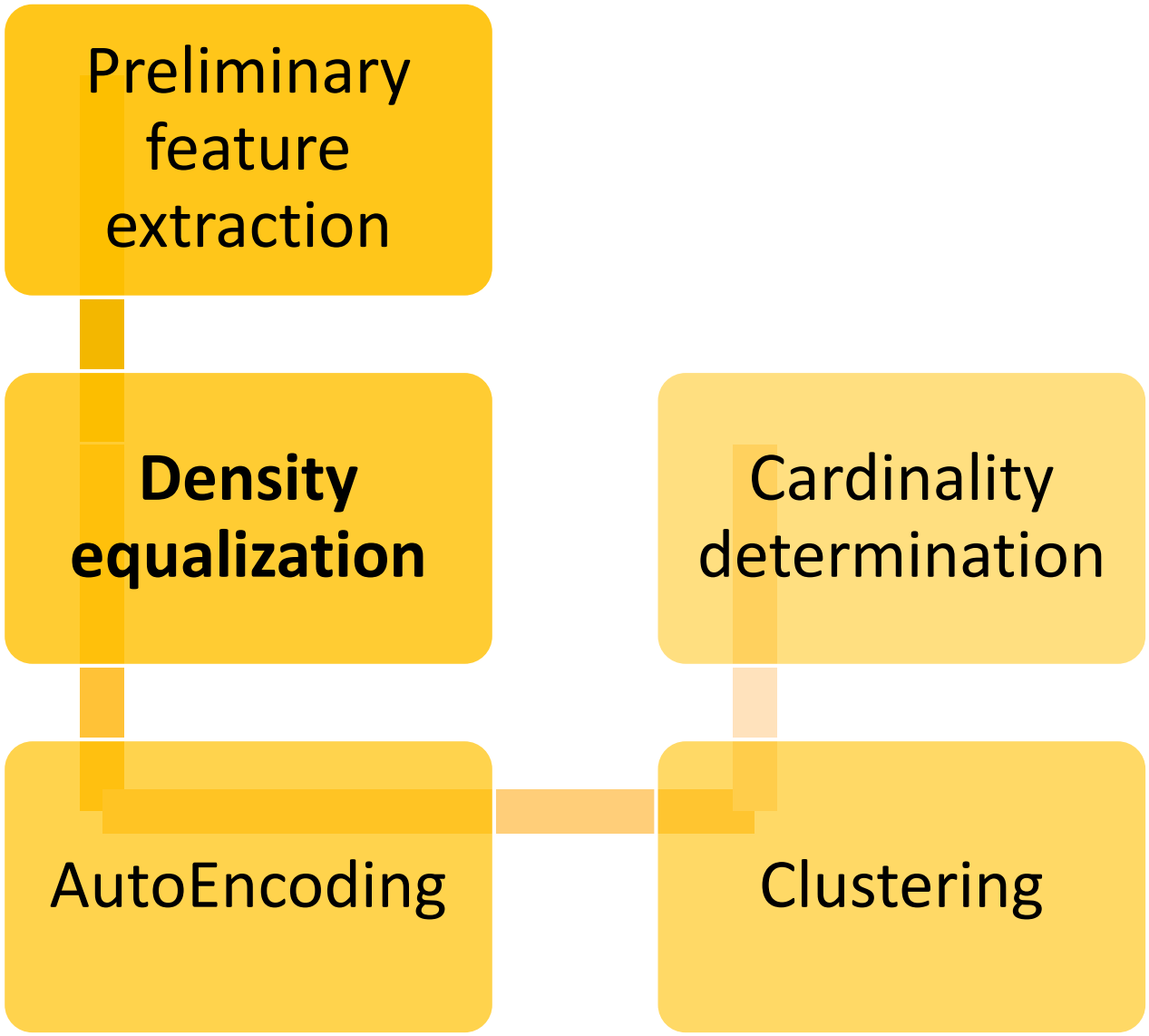
RFI Signal Clustering

IBM Research | Africa

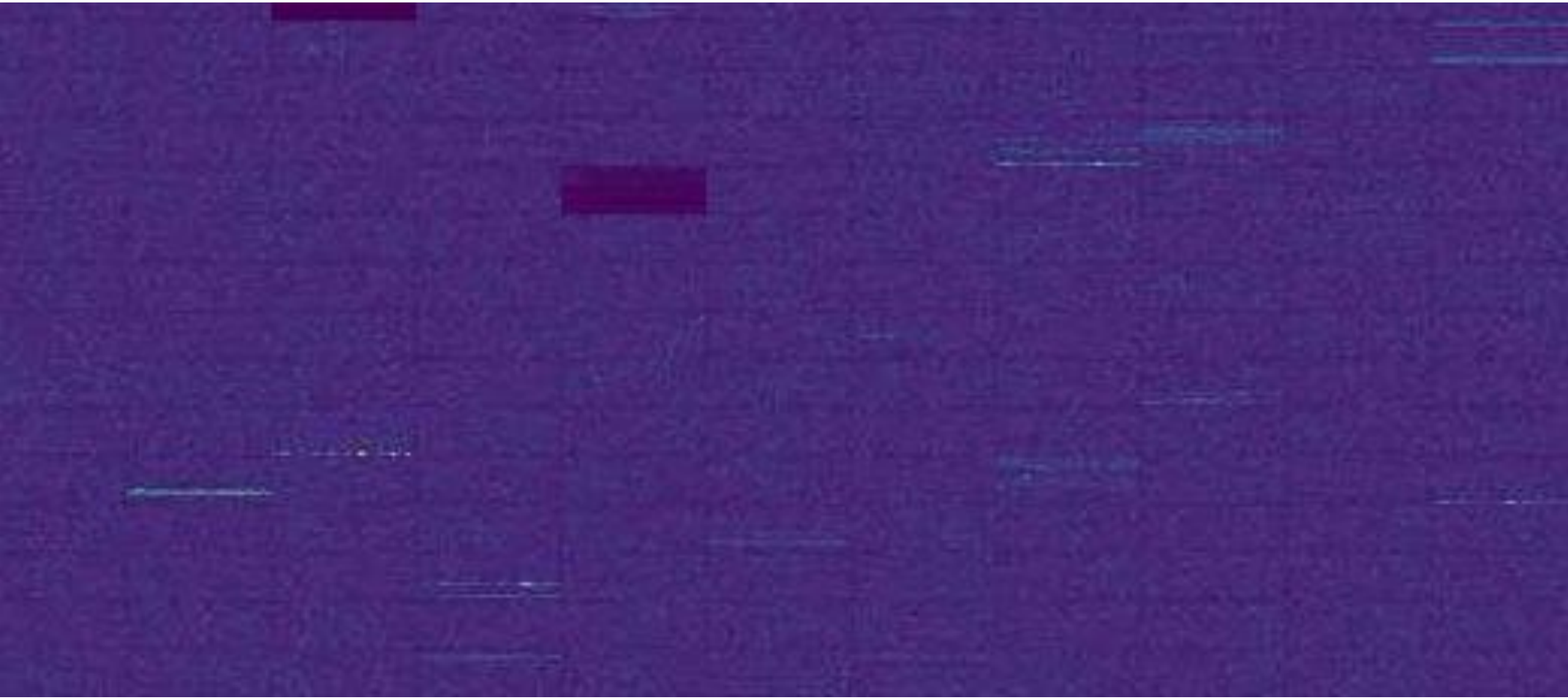
IBM



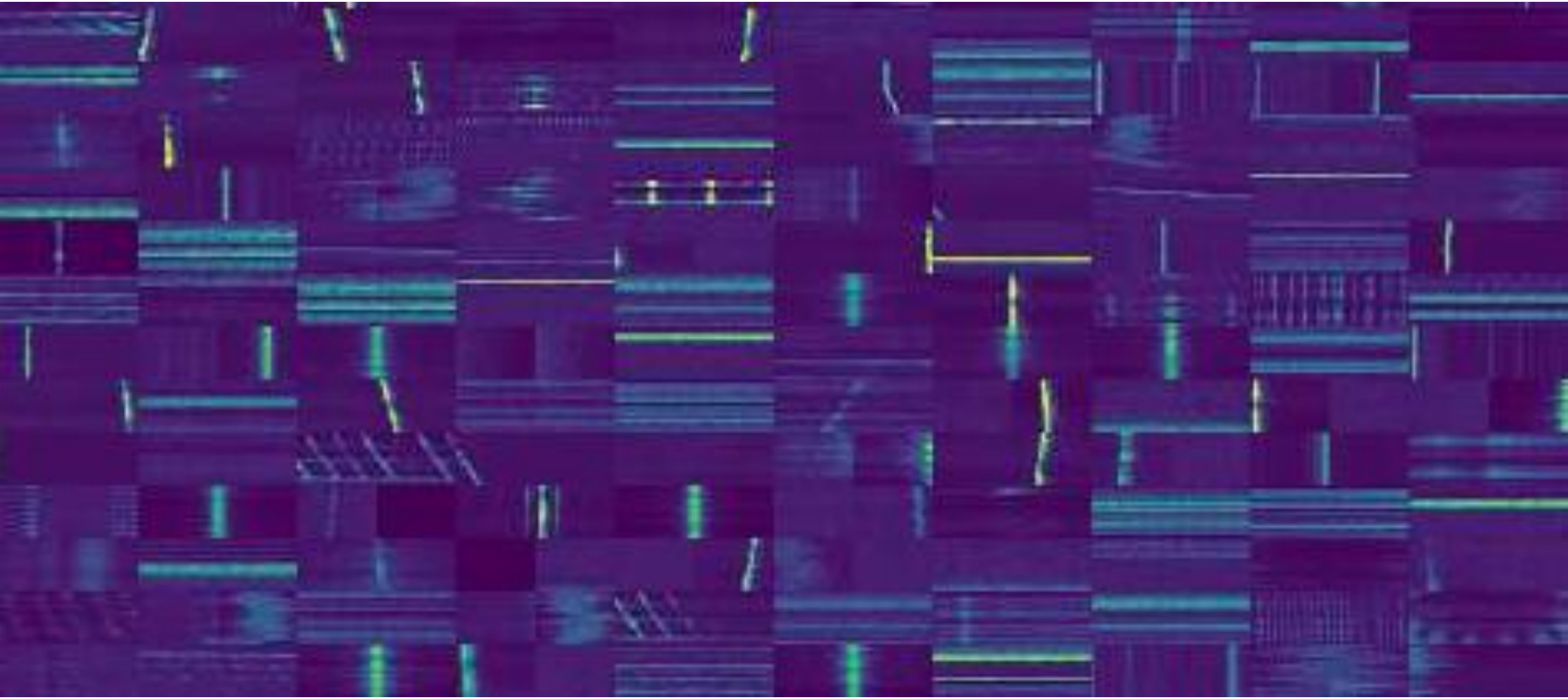
RFI Clustering



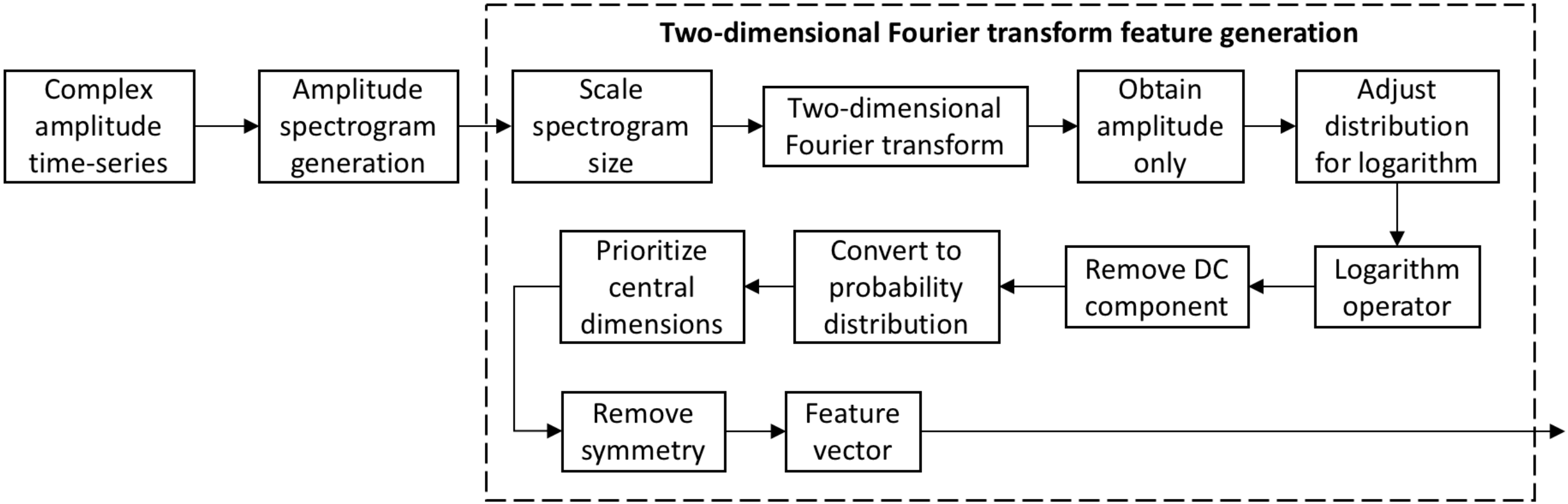
Spectrogram samples - Original density

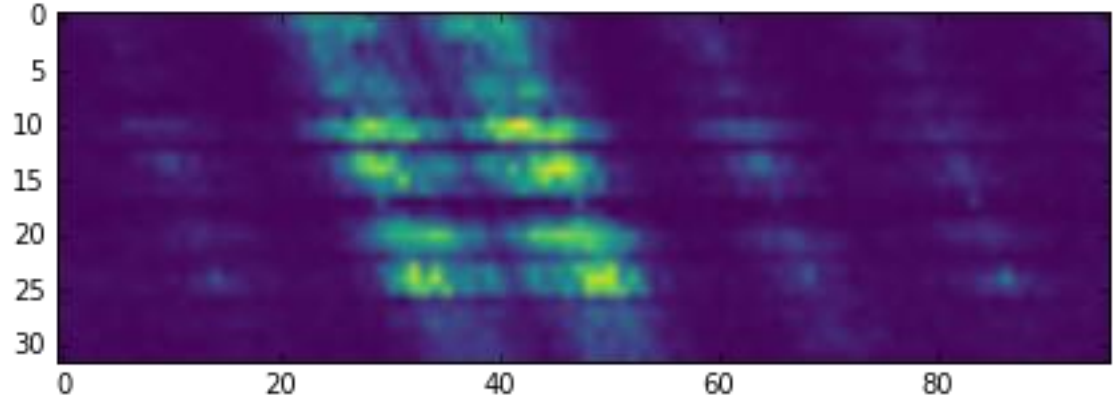


Spectrogram samples – Equalized density (Stat+PCA)

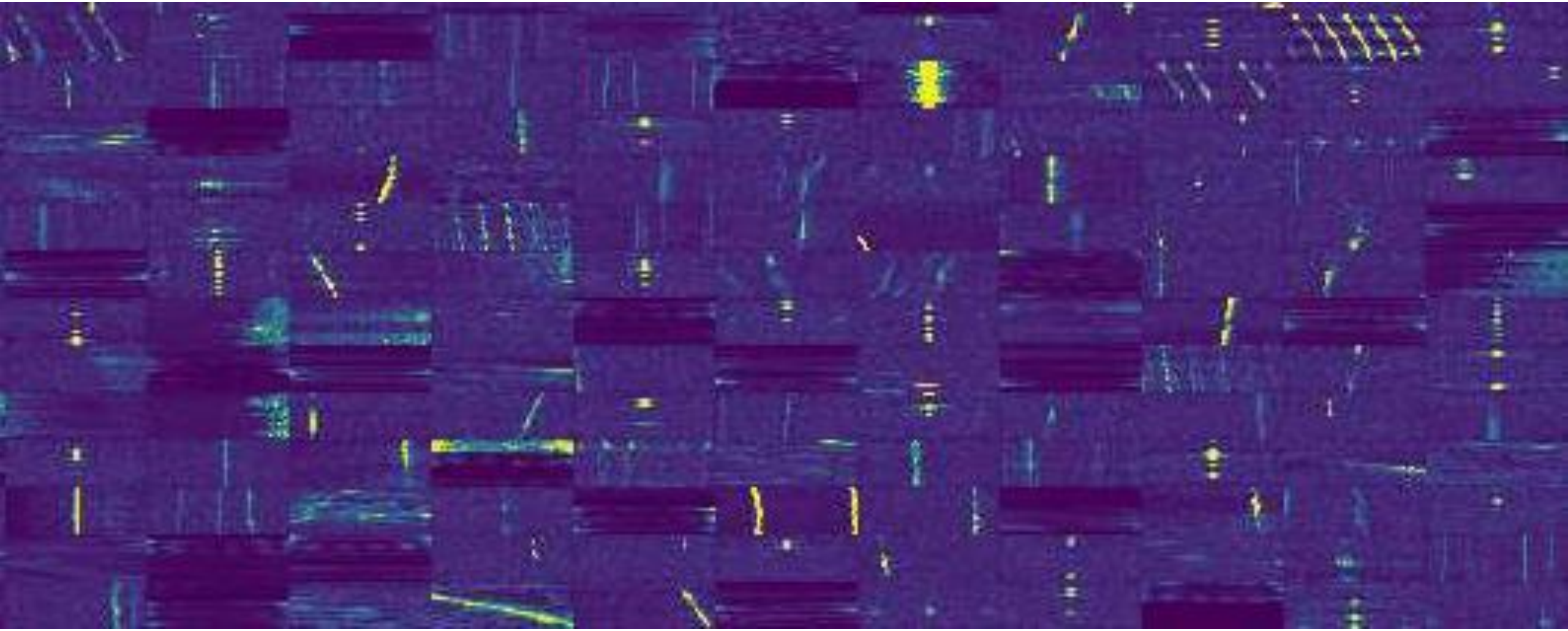


Spectrogram Fourier features

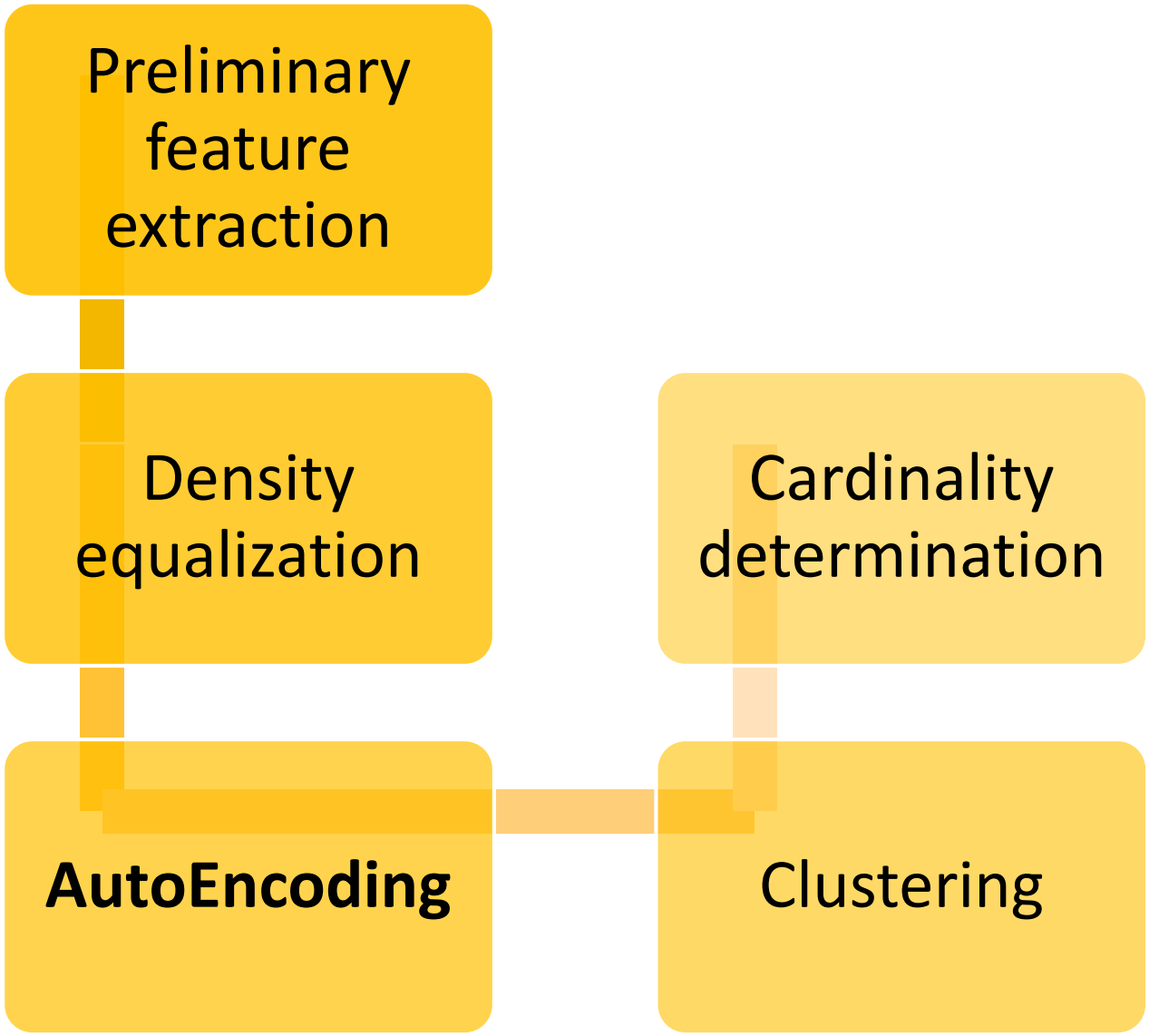




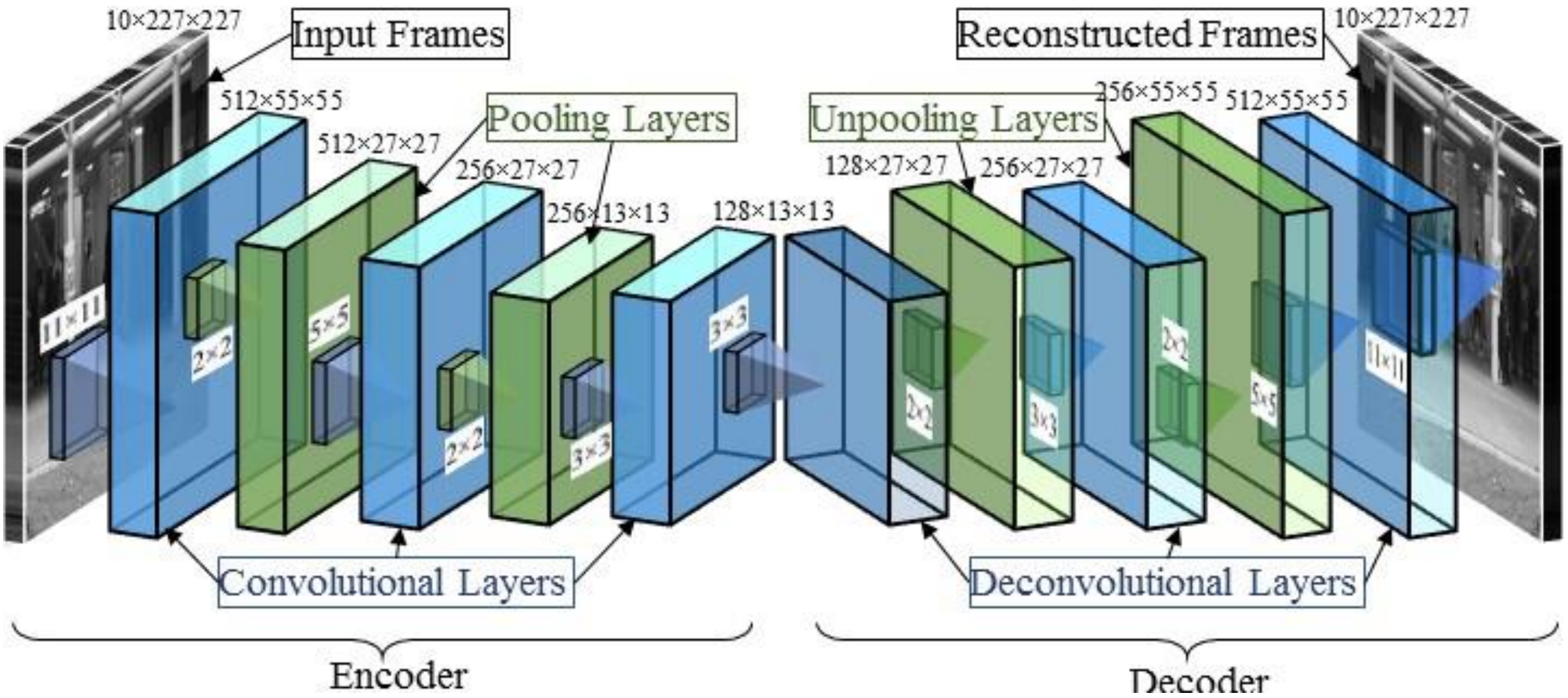
Spectrogram samples – Equalized density (Feature engineering)



RFI Clustering

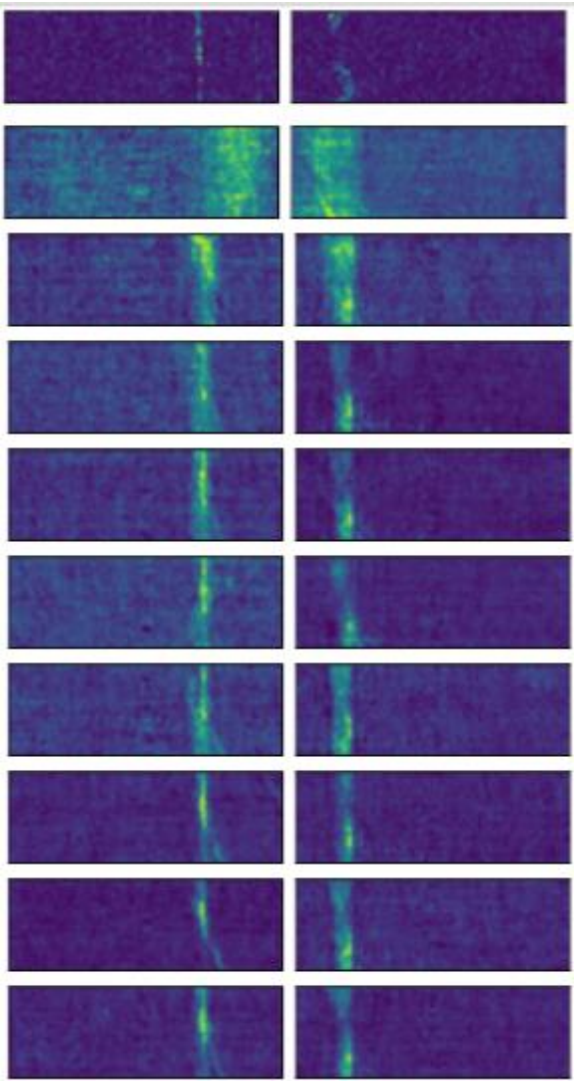


Convolutional Autoencoding: Architectures

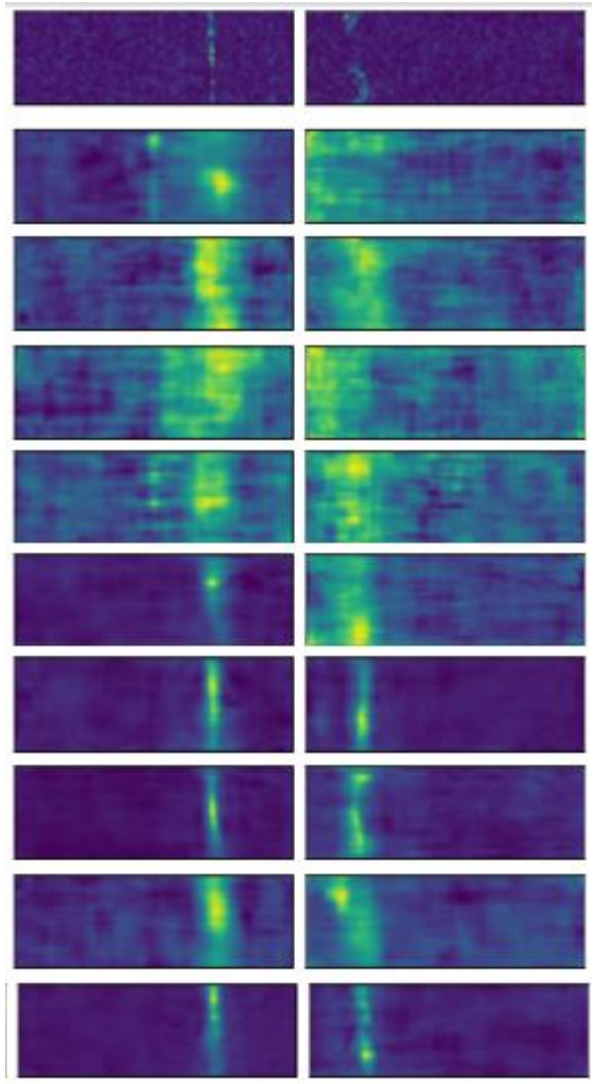


Autoencoding Comparison: Encoding Capacity

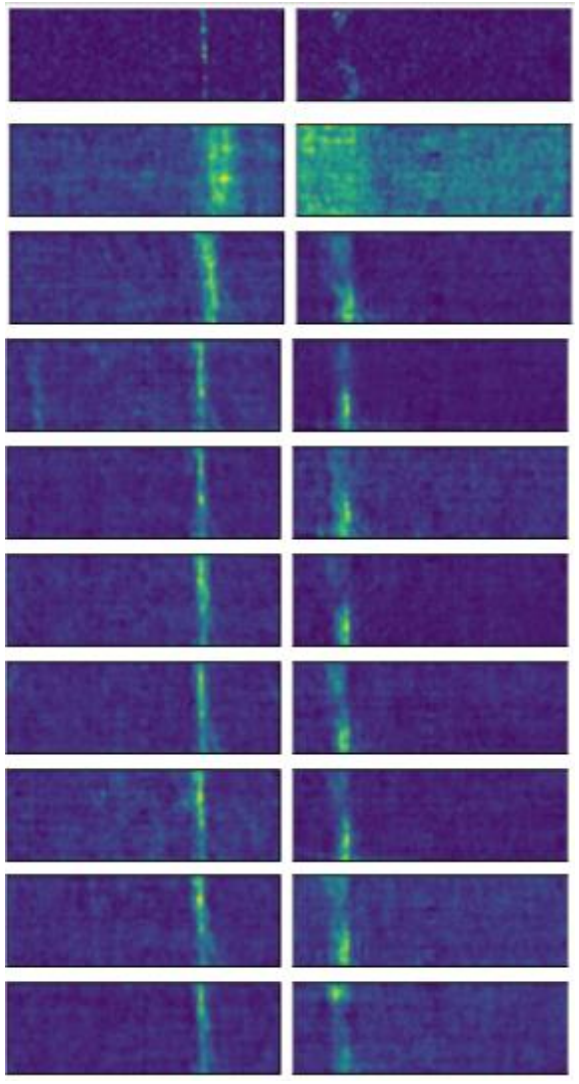
Dense



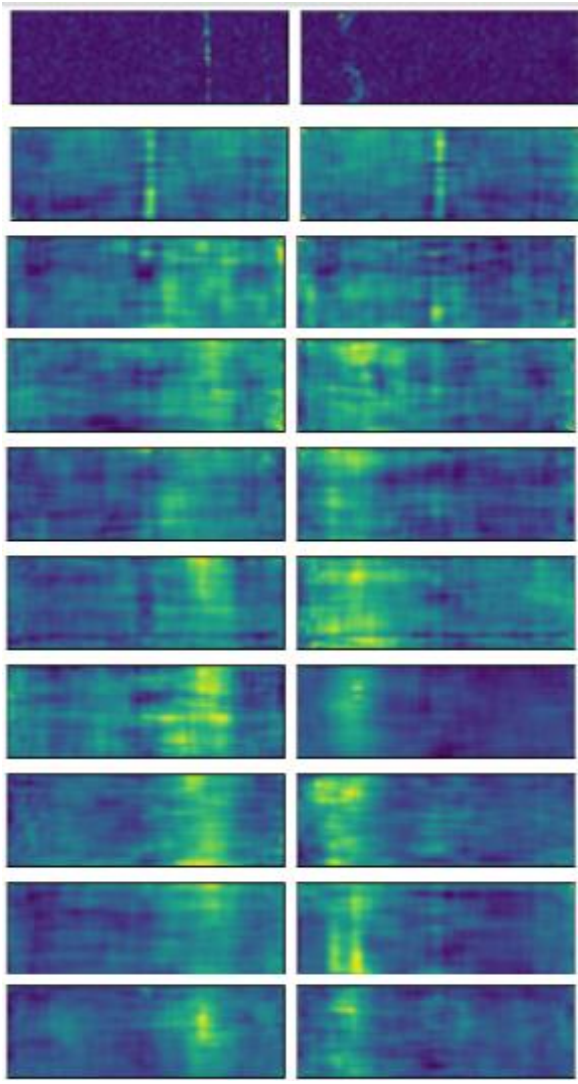
Convolutional



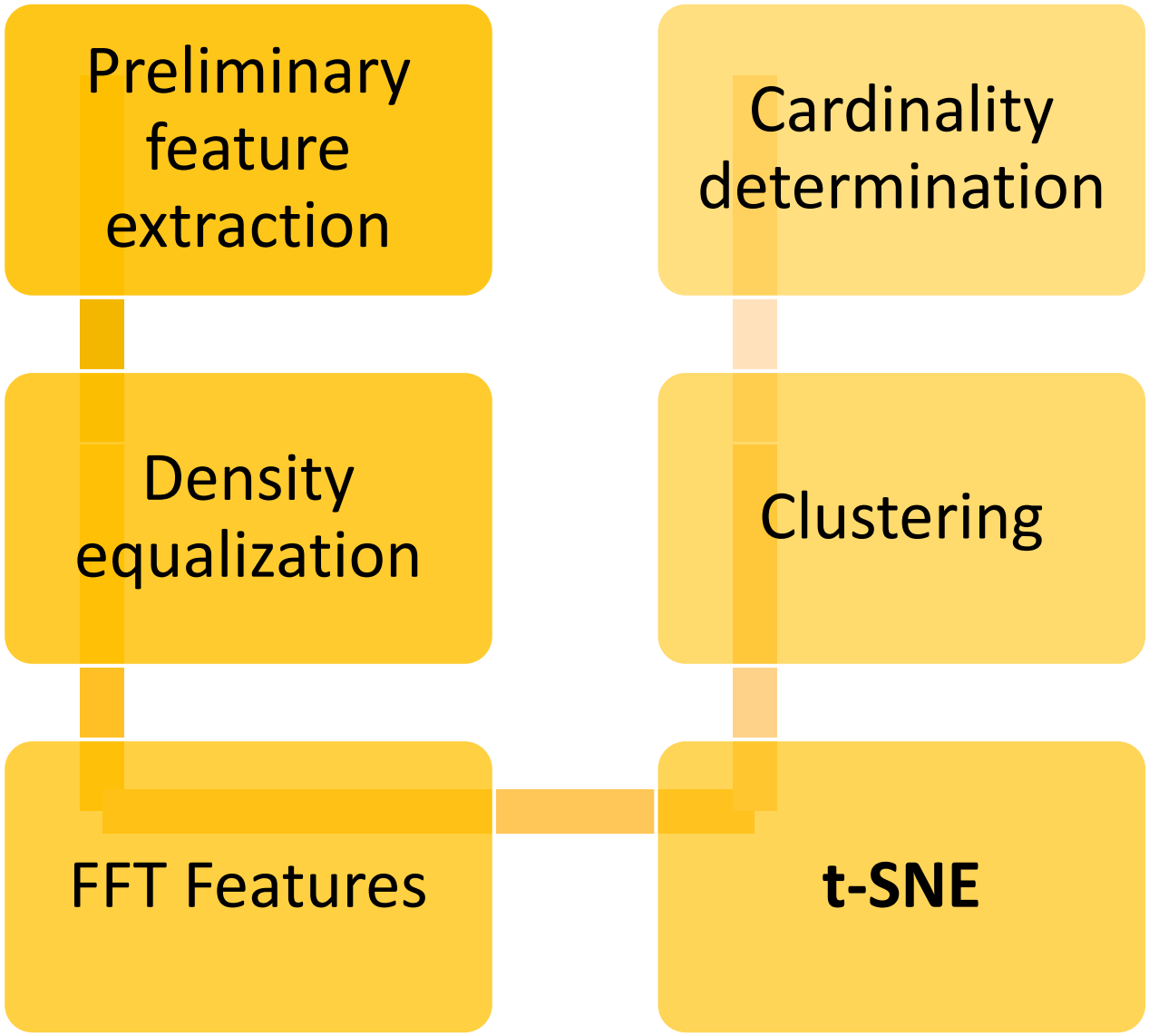
Dense Variational



Convolutional Variational



RFI Clustering



Laurens van der Maaten

Research scientist in machine learning and computer vision.

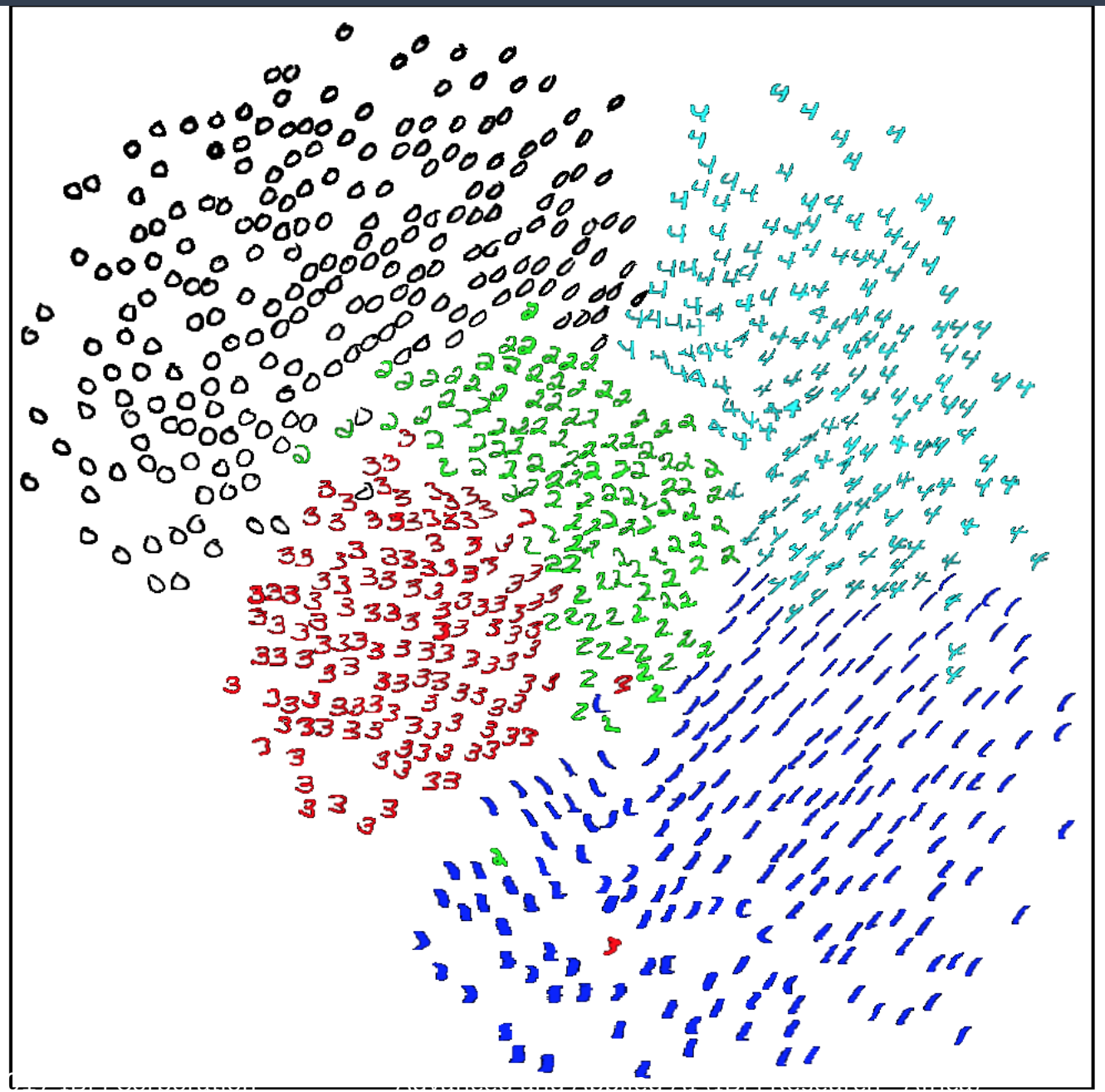
Email
Facebook
Google+
Github

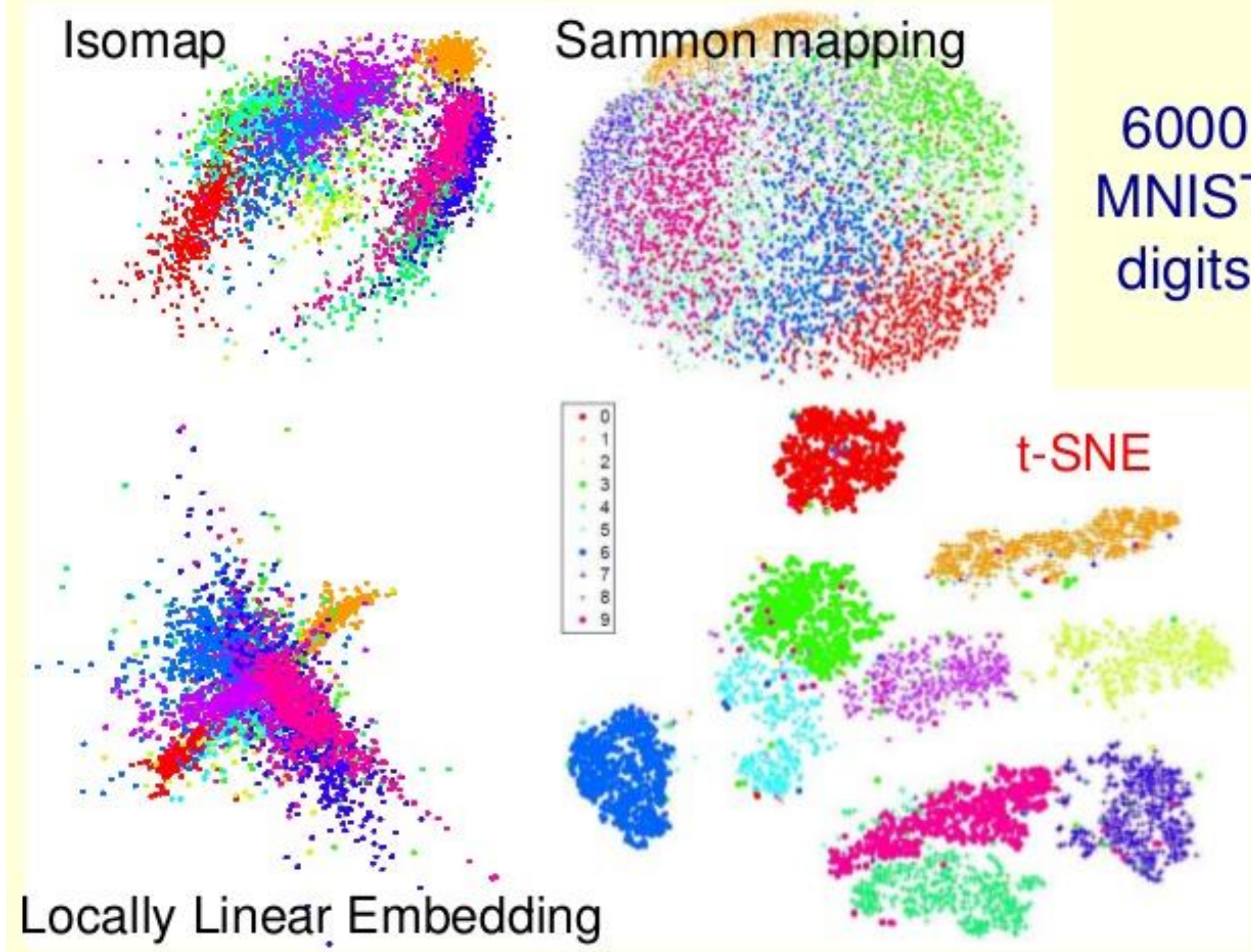
t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a (prize-winning) technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets. The technique can be implemented via Barnes-Hut approximations, allowing it to be applied on large real-world datasets. We applied it on data sets with up to 30 million examples. The technique and its variants are introduced in the following papers:

- L.J.P. van der Maaten. **Accelerating t-SNE using Tree-Based Algorithms**. *Journal of Machine Learning Research* 15(Oct):3221-3245, 2014. PDF [Supplemental material]
- L.J.P. van der Maaten and G.E. Hinton. **Visualizing Non-Metric Similarities in Multiple Maps**. *Machine Learning* 87(1):33-55, 2012. PDF
- L.J.P. van der Maaten. **Learning a Parametric Embedding by Preserving Local Structure**. In *Proceedings of the Twelfth International Conference on Artificial Intelligence & Statistics (AI-STATS), JMLR W&CP* 5:384-391, 2009. PDF
- L.J.P. van der Maaten and G.E. Hinton. **Visualizing High-Dimensional Data Using t-SNE**. *Journal of Machine Learning Research* 9(Nov):2579-2605, 2008. PDF [Supplemental material] [Talk]

SNE (2002) of MNIST




Isomap
Sammon mapping
6000
MNIST
digits
0
1
2
3
4
5
6
7
8
9
t-SNE
Locally Linear Embedding

MDS: Multi Dimensional Scaling

- Global geometry reconstruction

$$Cost = \sum_{i<j} (d_{ij} - \hat{d}_{ij})^2$$

LLE: Locally Linear Embedding

- Local geometry reconstruction

$$Cost = \sum_i \| \mathbf{x}_i - \sum_{j \varepsilon N(i)} w_{ij} \mathbf{x}_j \|^2, \qquad \sum_{j \varepsilon N(i)} w_{ij} = 1$$

2002

SNE: Stochastic Neighborhood Embedding

- Probabilistic version of LLE

$$p_{j|i} = \frac{e^{-\|x_i - x_j\|^2 / 2\sigma_i^2}}{\sum_k e^{-\|x_i - x_k\|^2 / 2\sigma_i^2}} \quad q_{j|i} = \frac{e^{-\|y_i - y_j\|^2}}{\sum_k e^{-\|y_i - y_k\|^2}}$$

2007

UNI-SNE

- Alleviates crowding problem

2008

t-SNE

- Faster and more stable

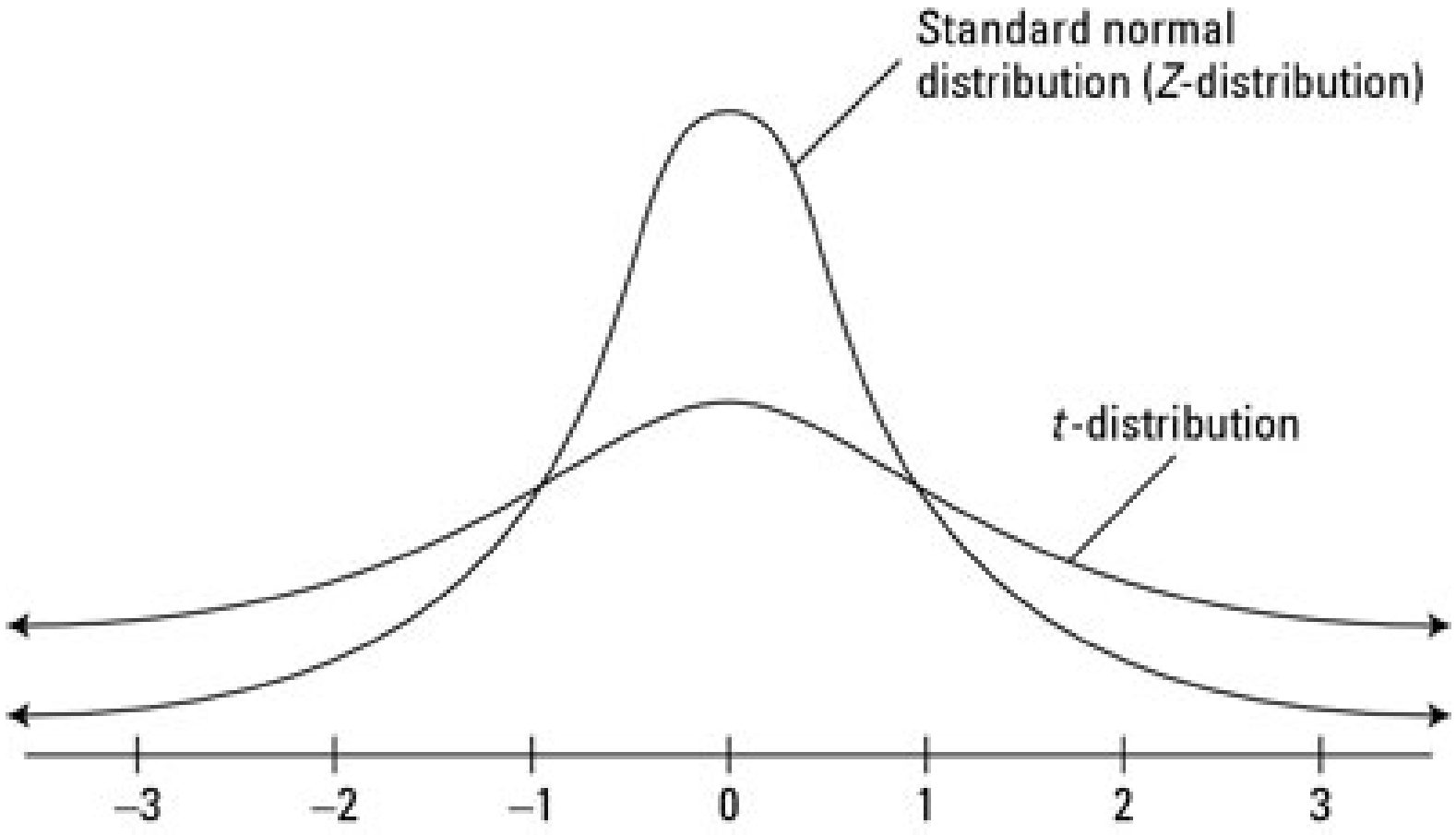
Low-dimensional embedding

SNE

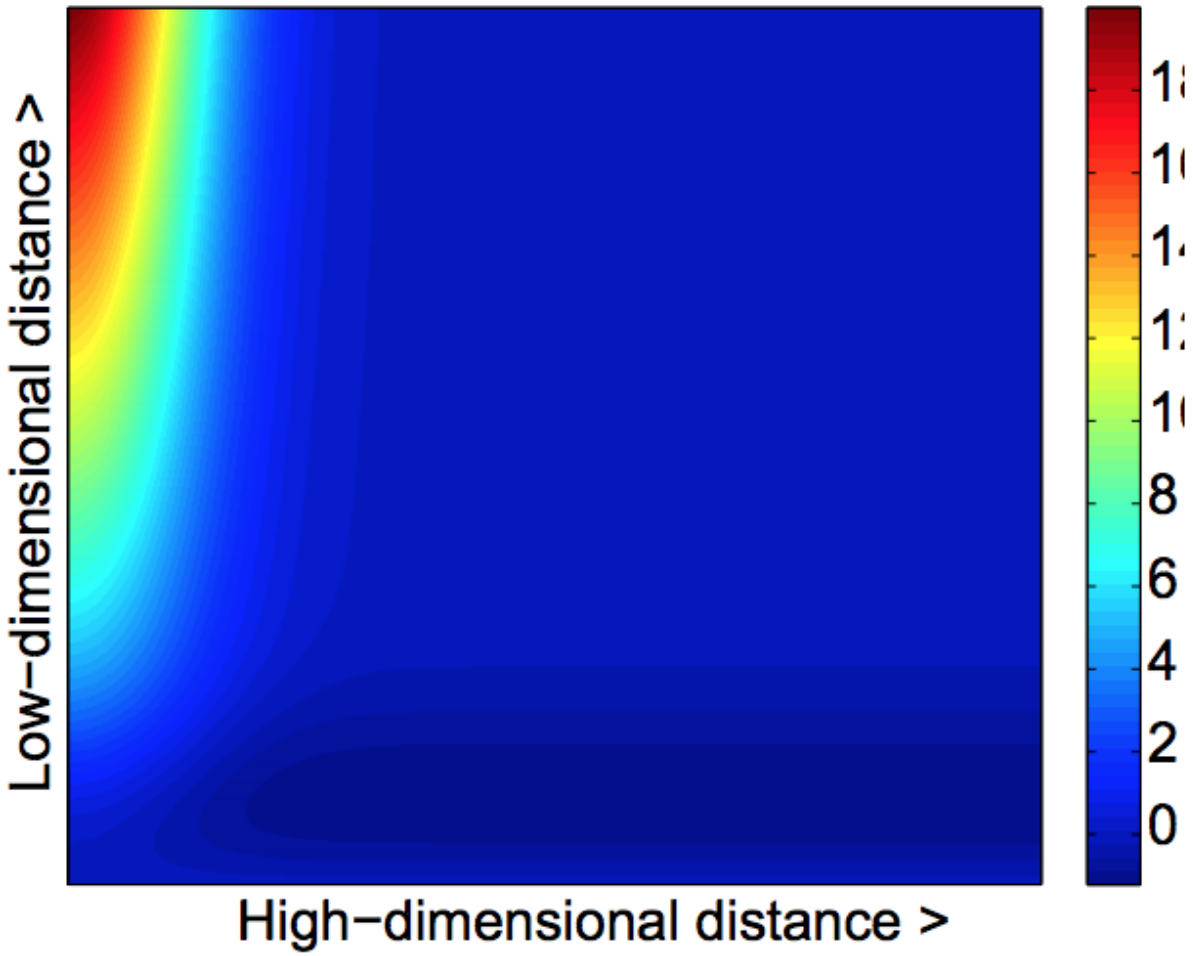
$$q_{ij} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq l} \exp\left(-\|y_k - y_l\|^2\right)},$$

t-SNE

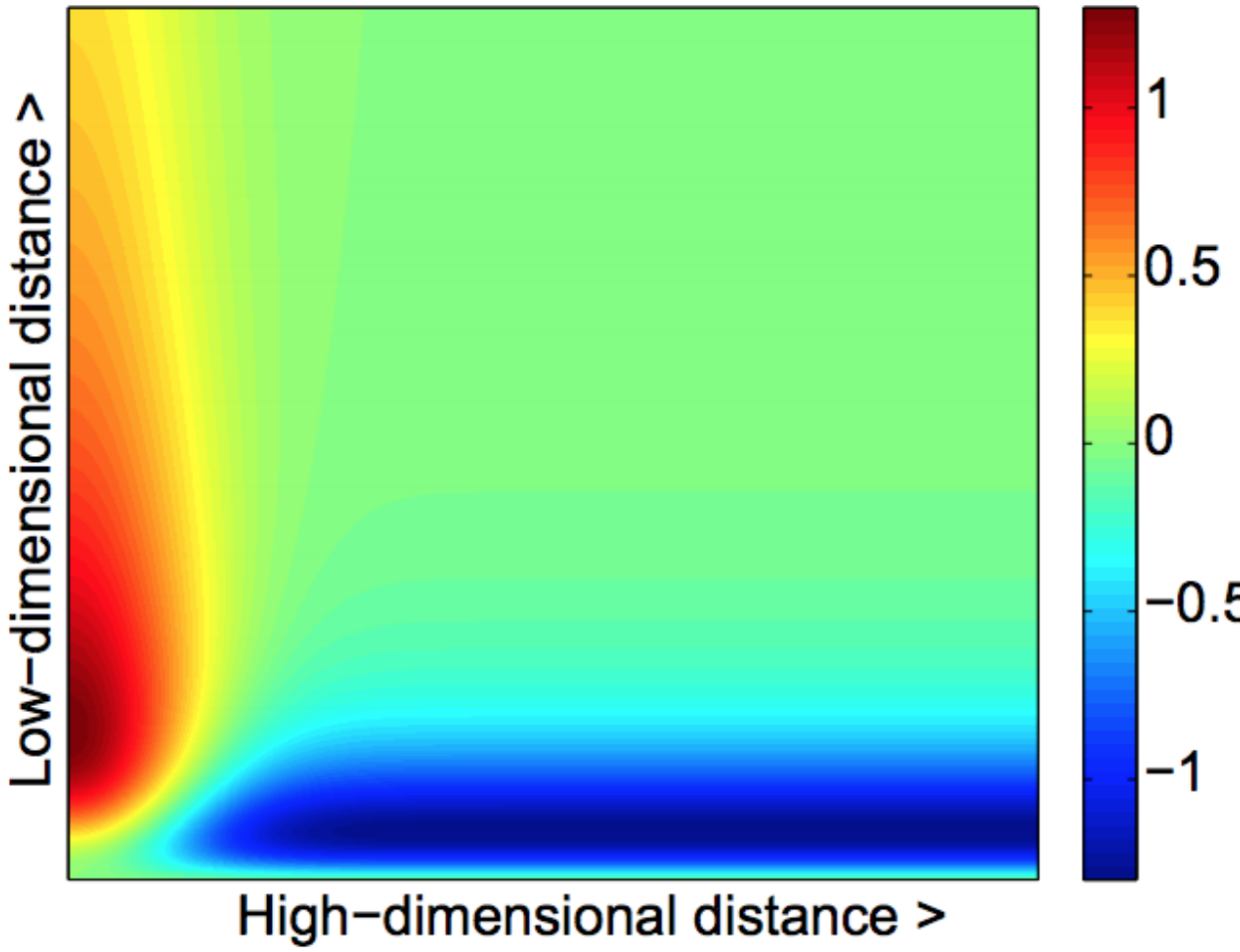
$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y_k - y_l\|^2\right)^{-1}}.$$



Gradient comparison



(a) Gradient of SNE.



(c) Gradient of t-SNE.

From: van der Maaten, 2008

High-dimension & Similarity reproduction

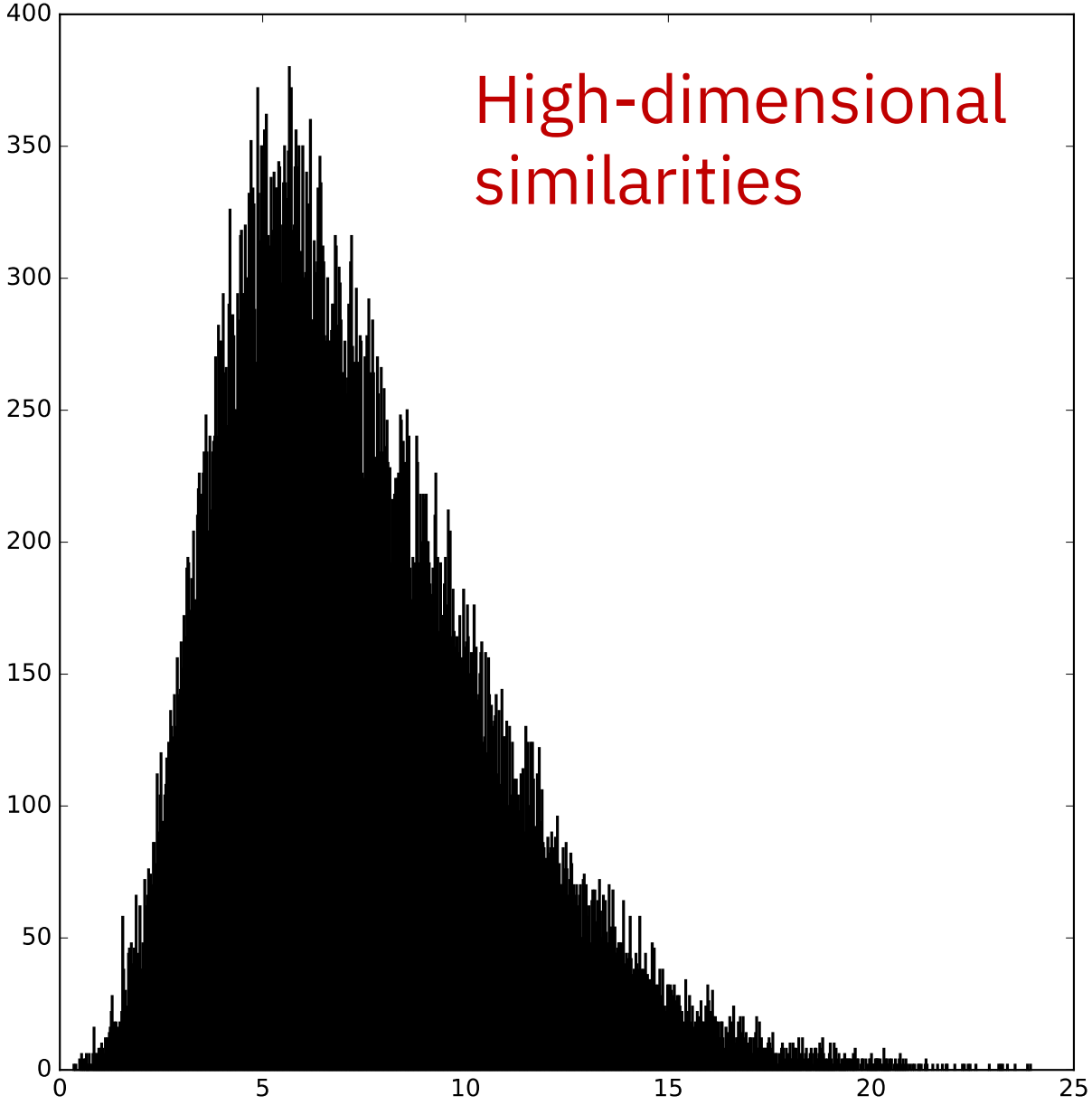
High-dimensional input

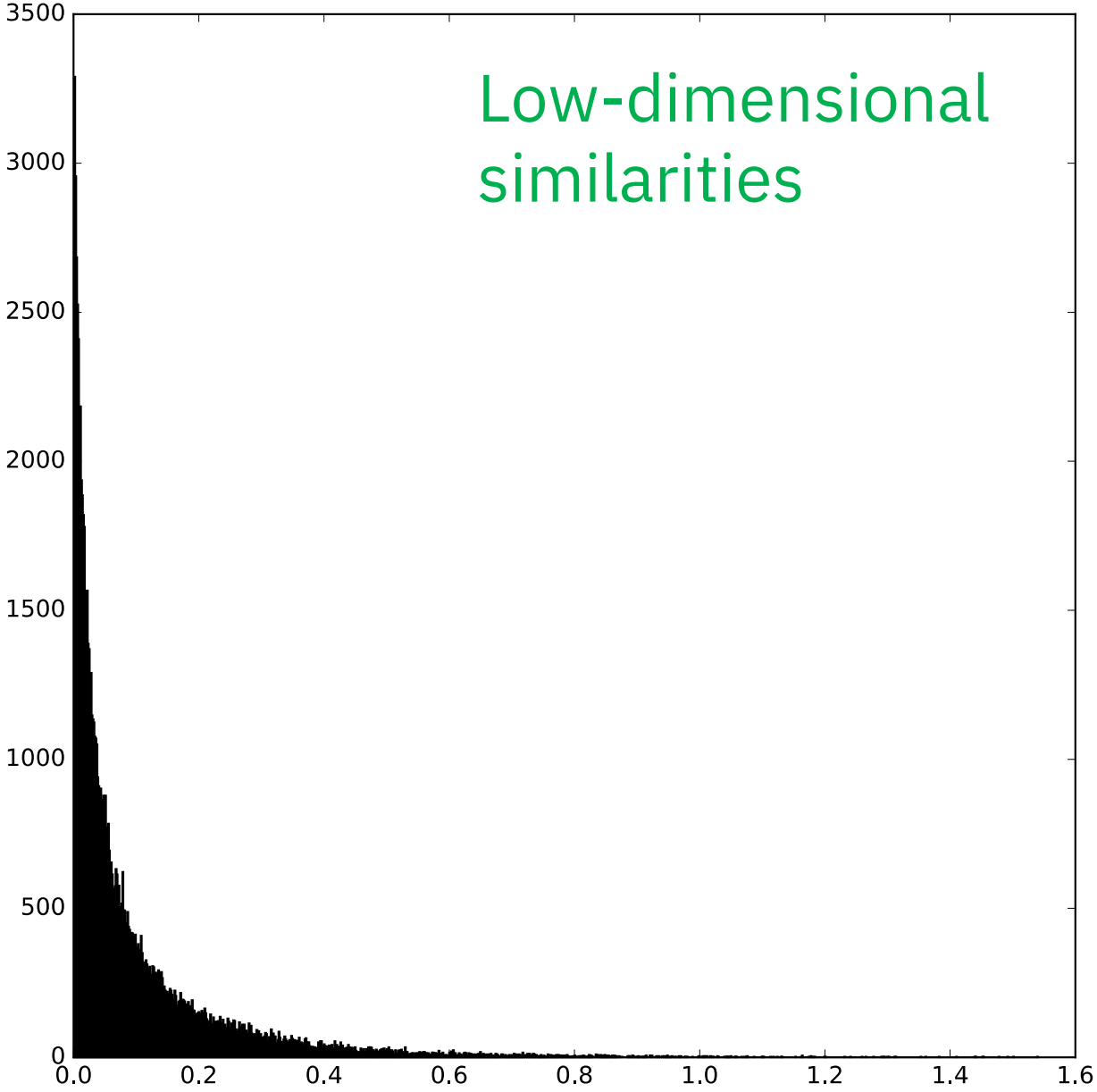
$$p_{ij} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma^2\right)}{\sum_{k \neq l} \exp\left(-\|x_k - x_l\|^2 / 2\sigma^2\right)},$$

Reproduce similarities

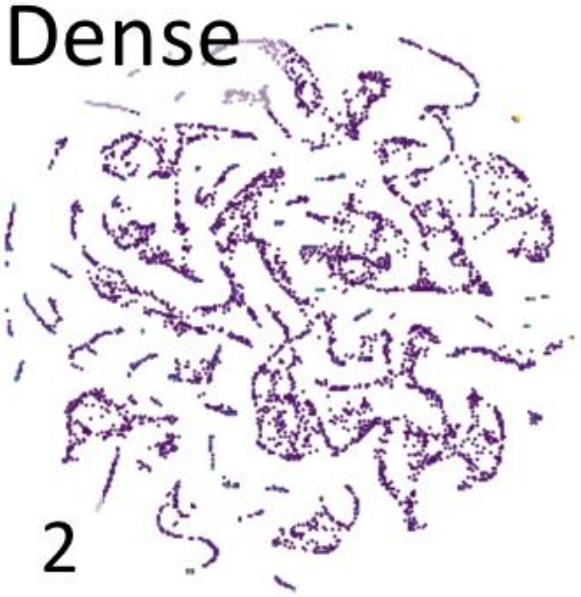
$$C = \sum_i KL(P_i || Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}},$$

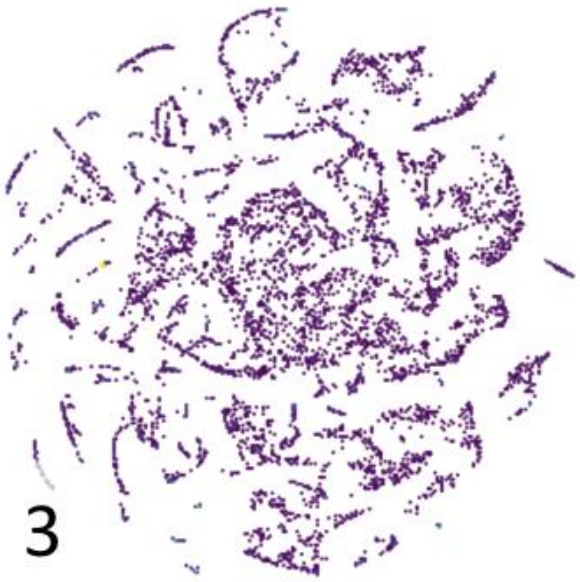
Curse of dimensionality



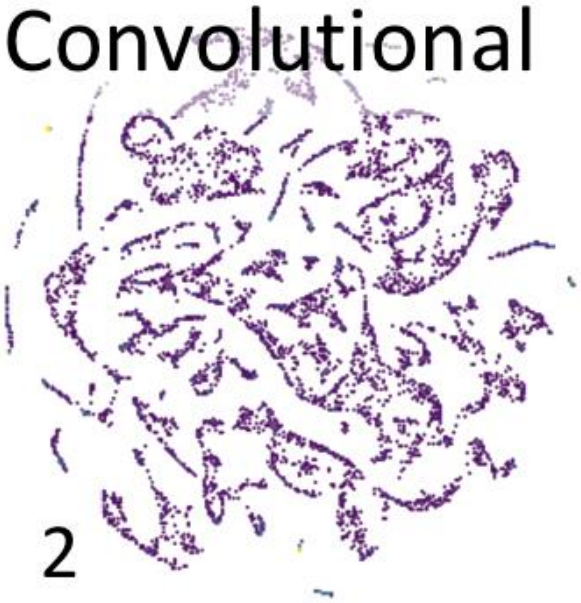


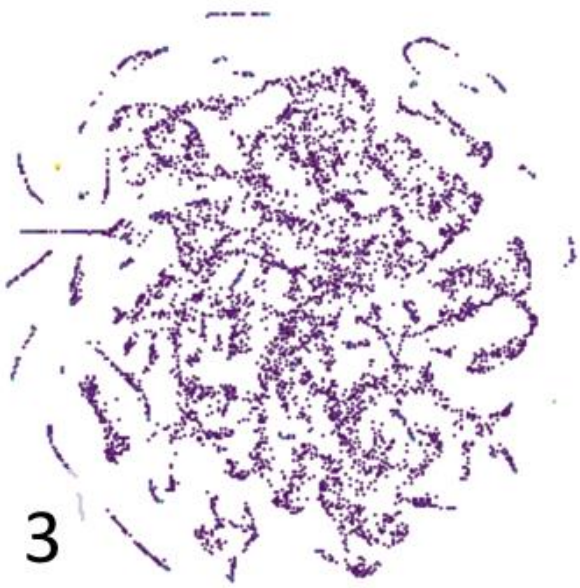
Dense



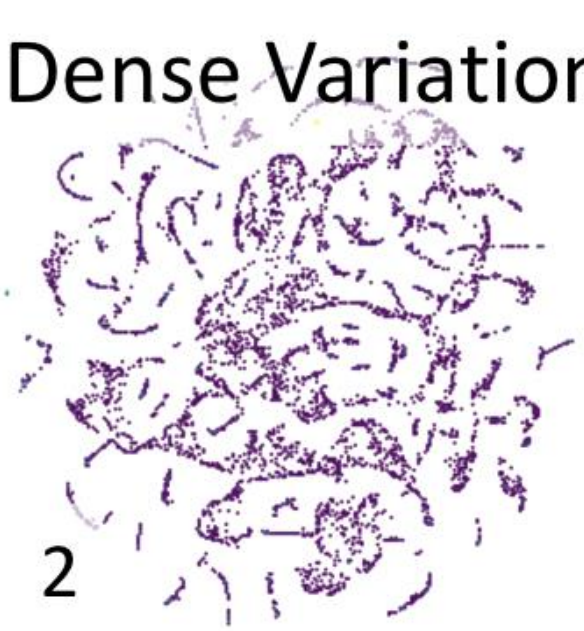


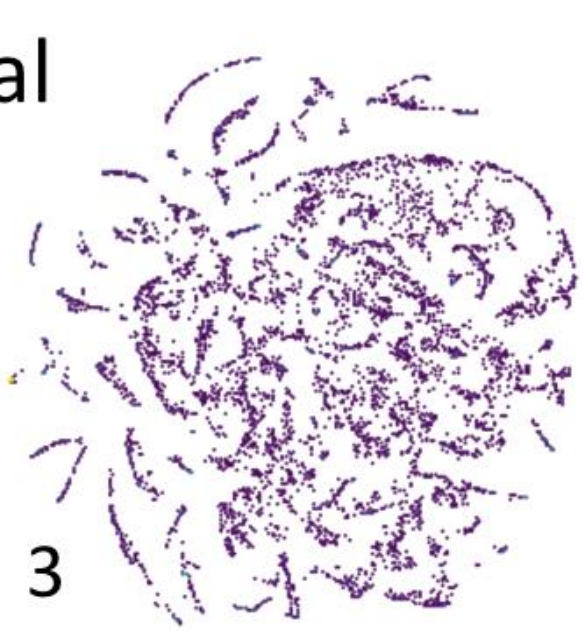
Convolutional



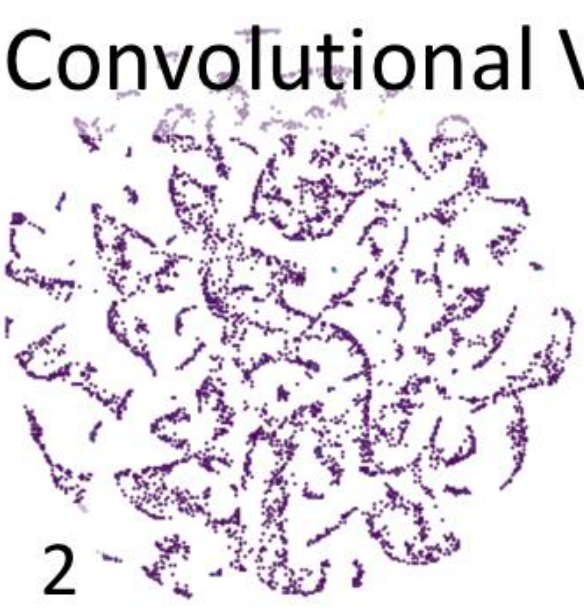


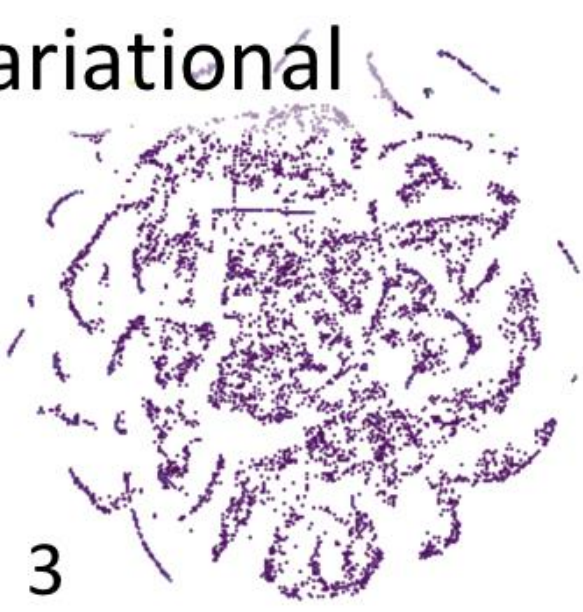
Dense Variational





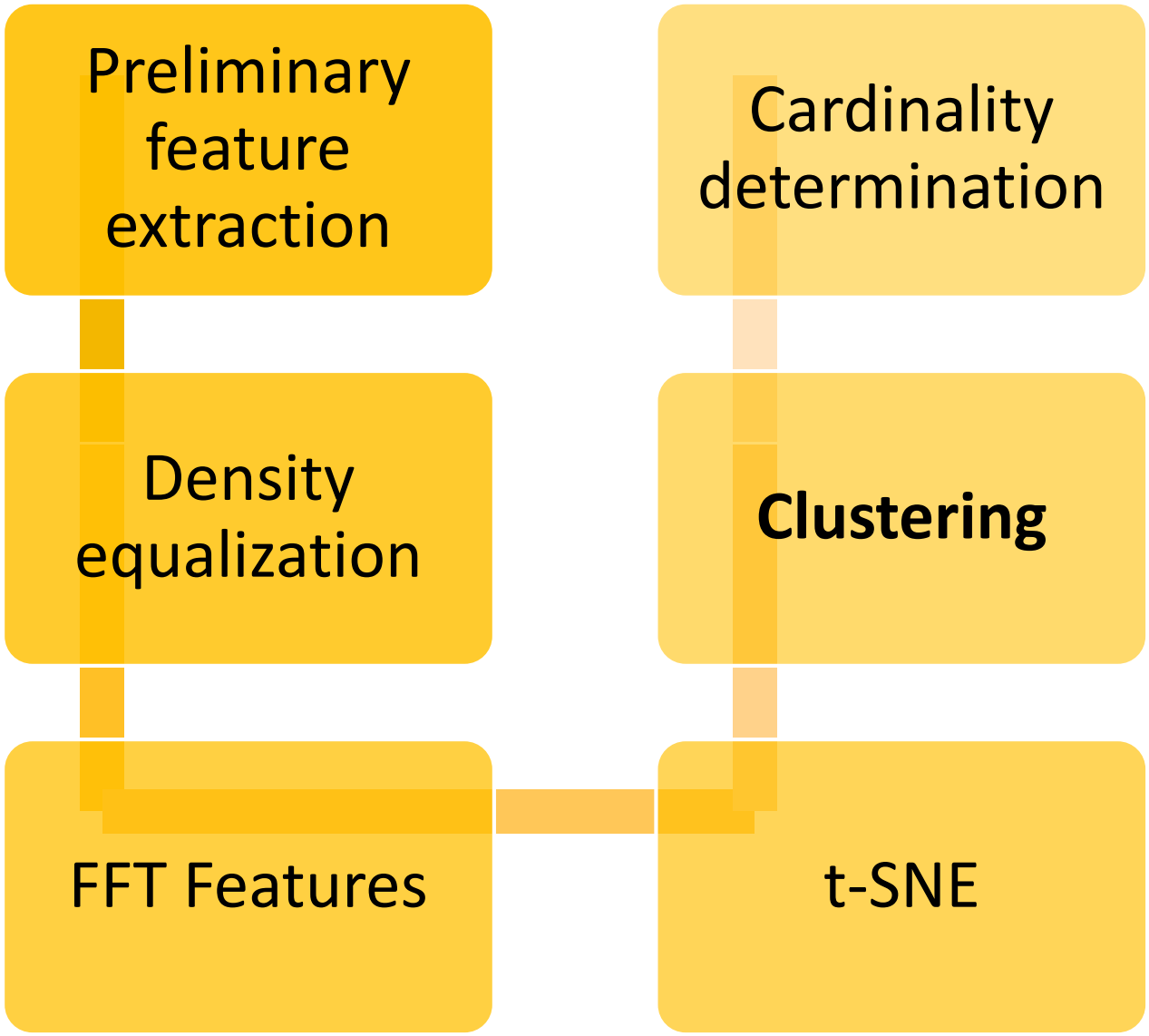
Convolutional Variational

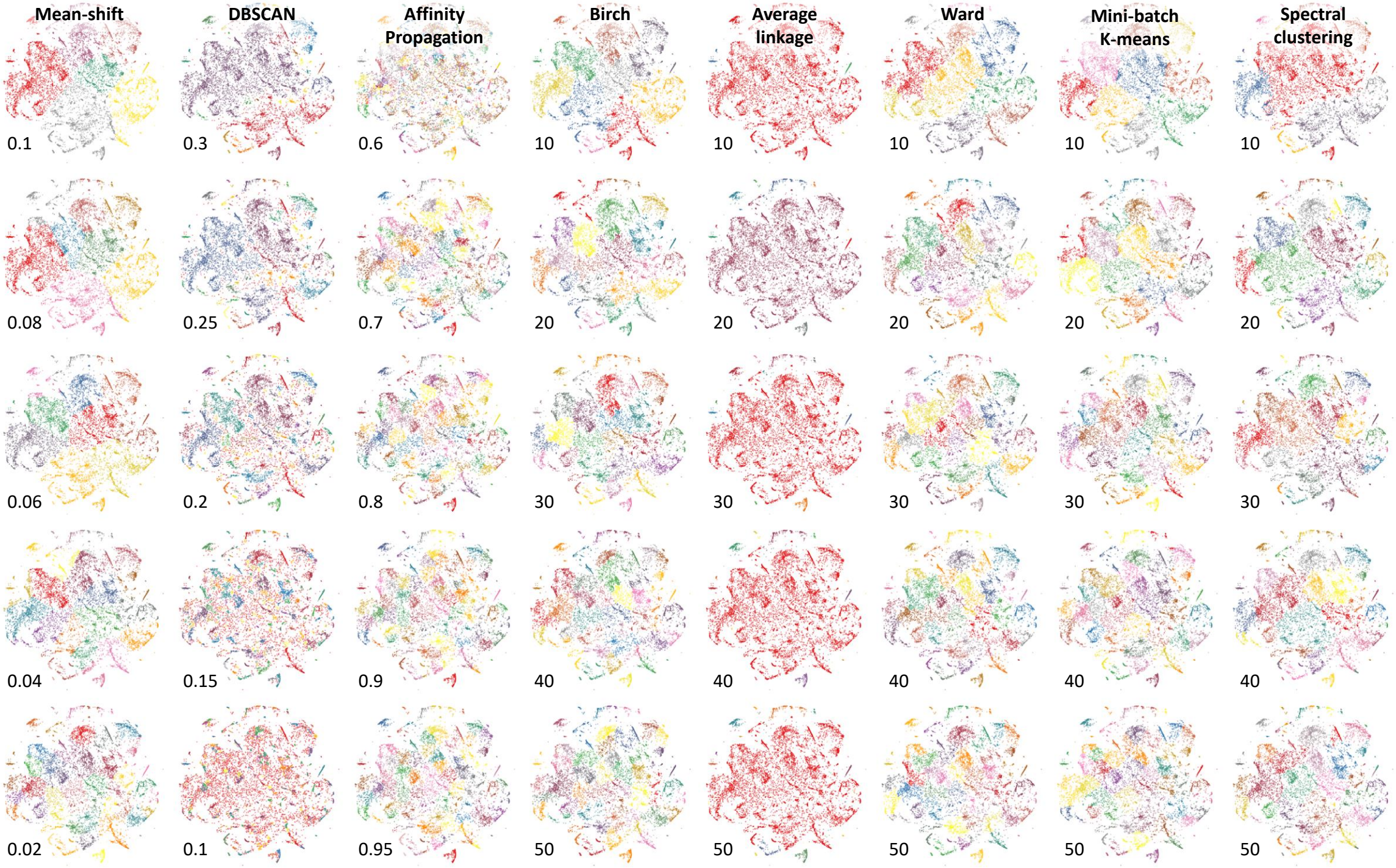




Autoencoding: tSNE plots

RFI Clustering

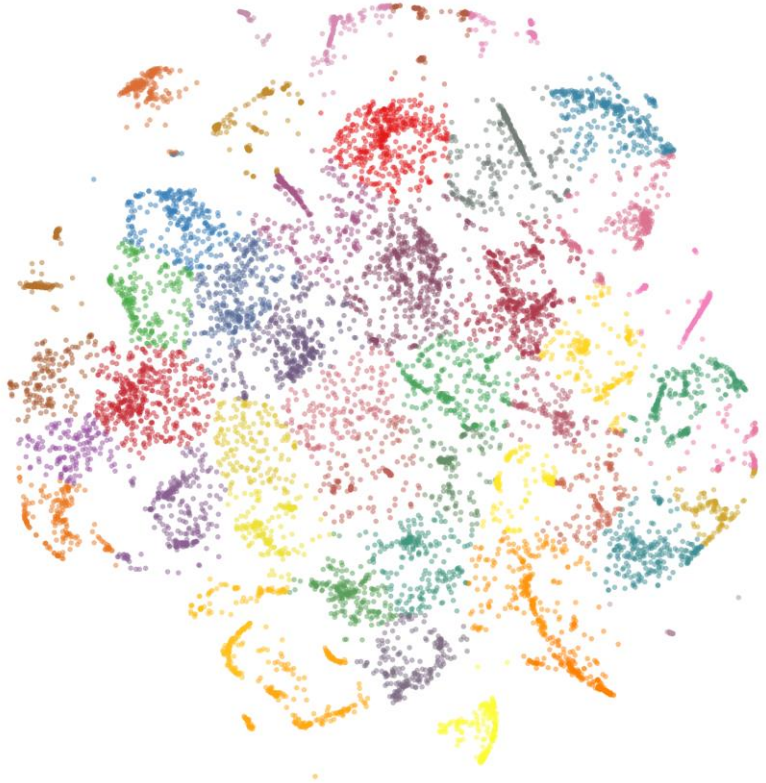




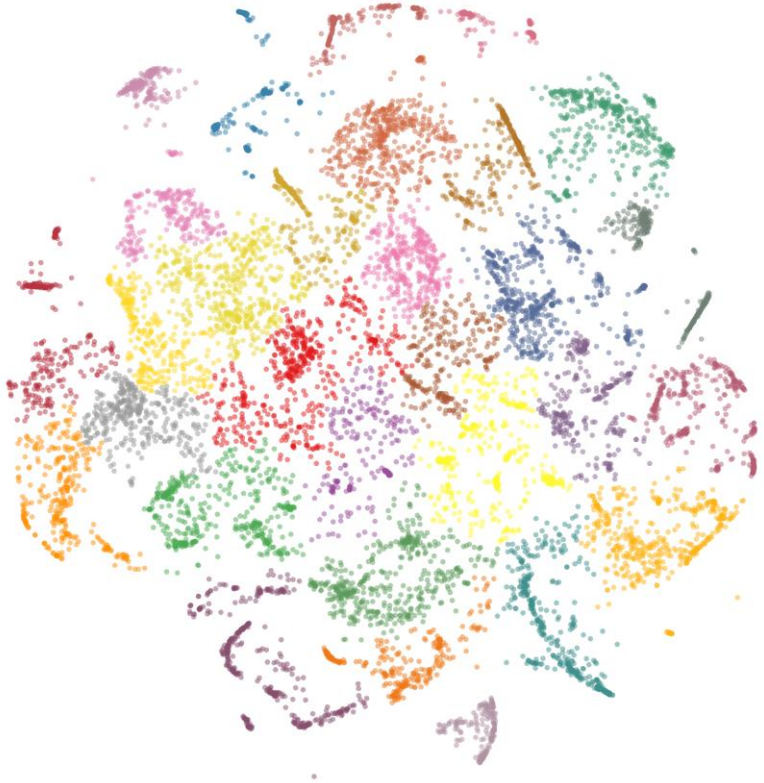
Clustering comparison of tSNE

Clustering: Qualitative inspection

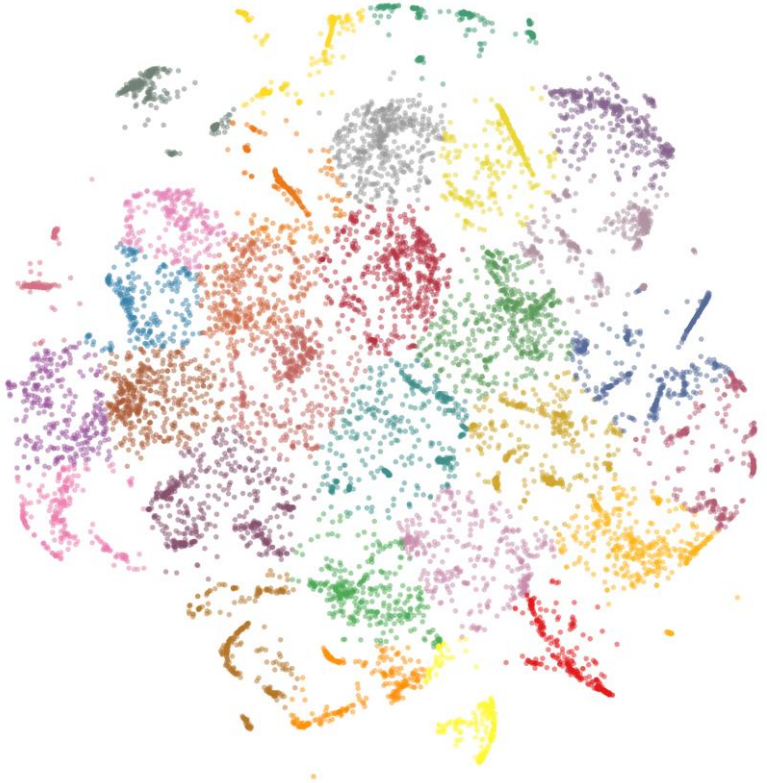
Mean-shift



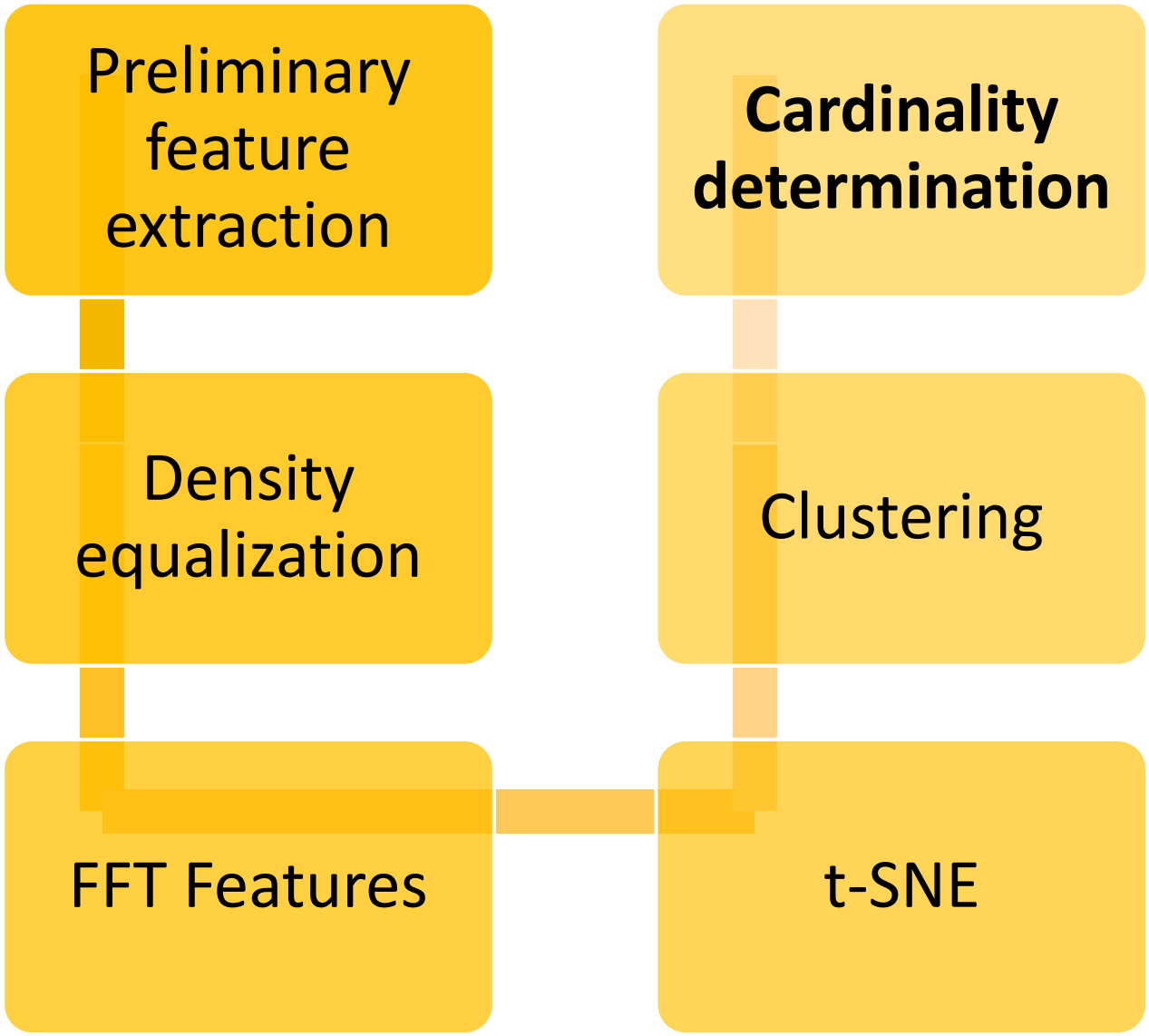
Ward



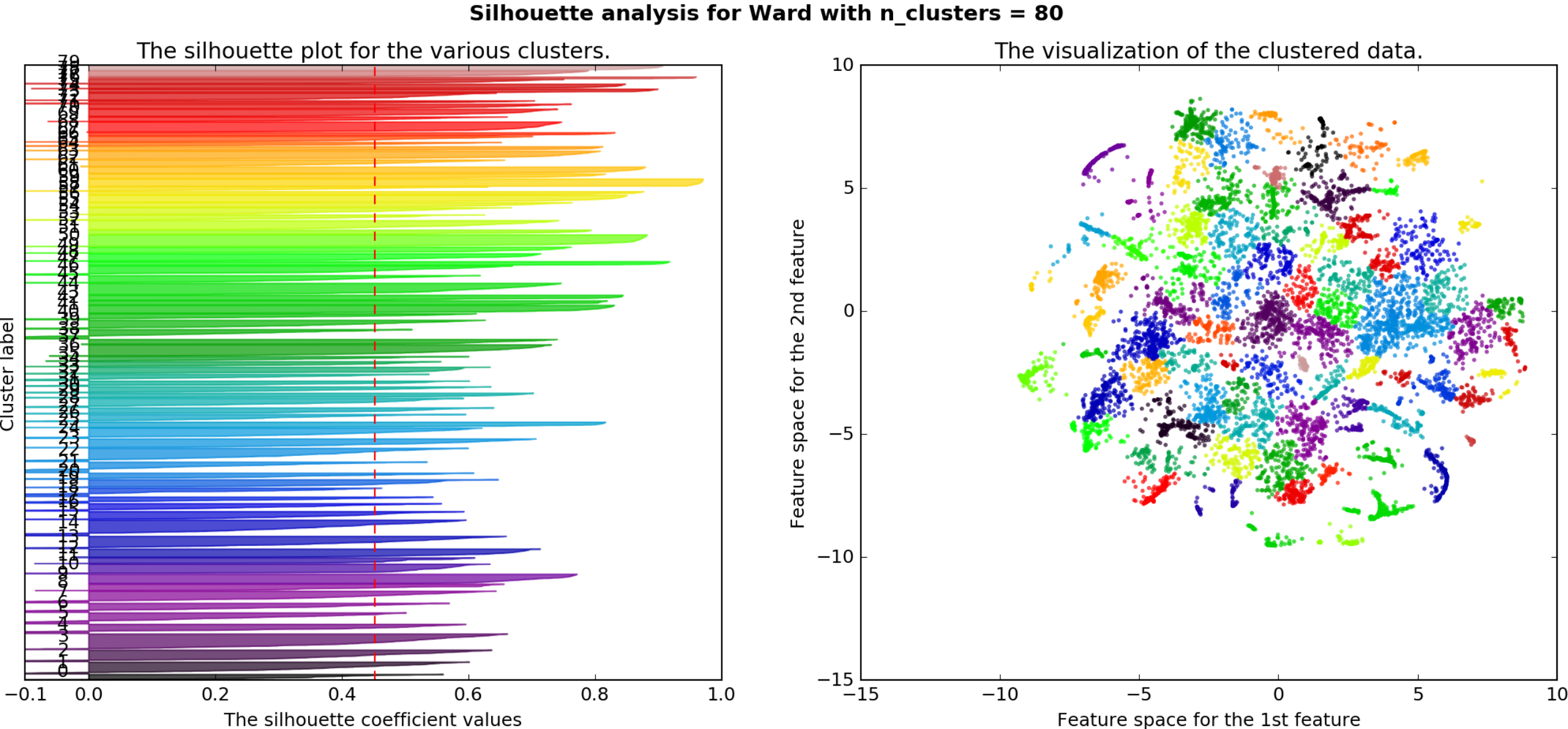
Mini-batch K-means



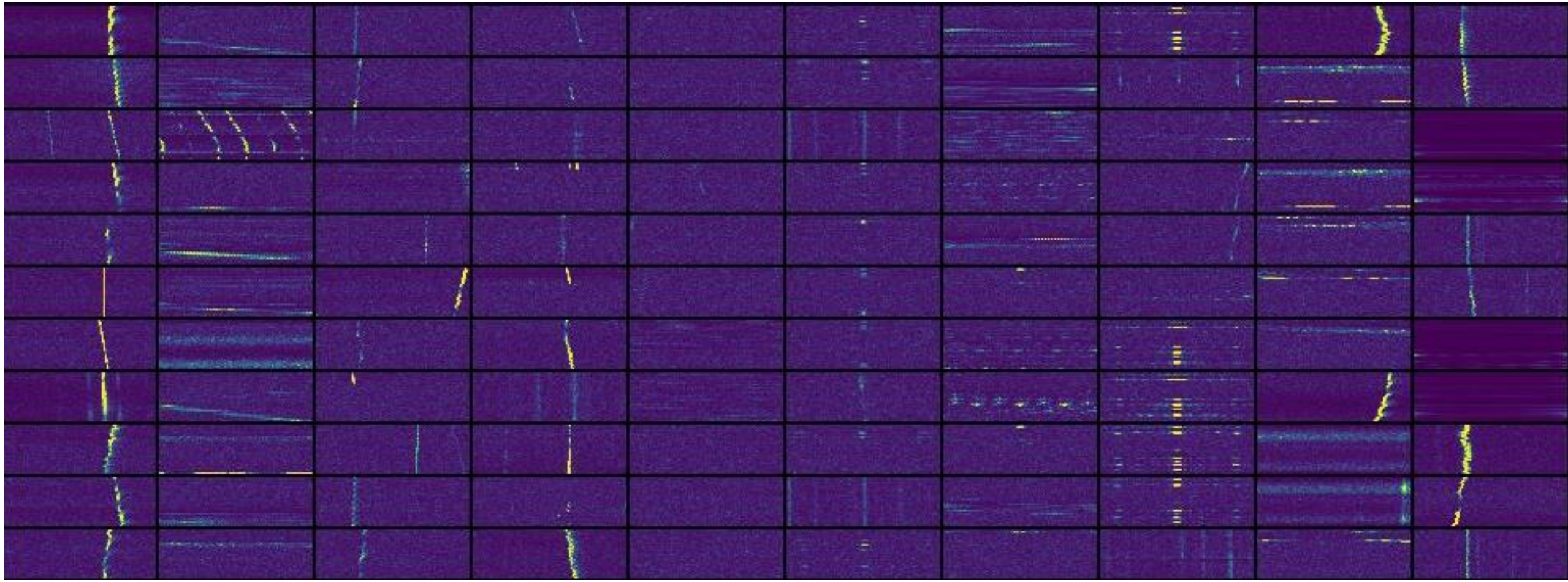
RFI Clustering



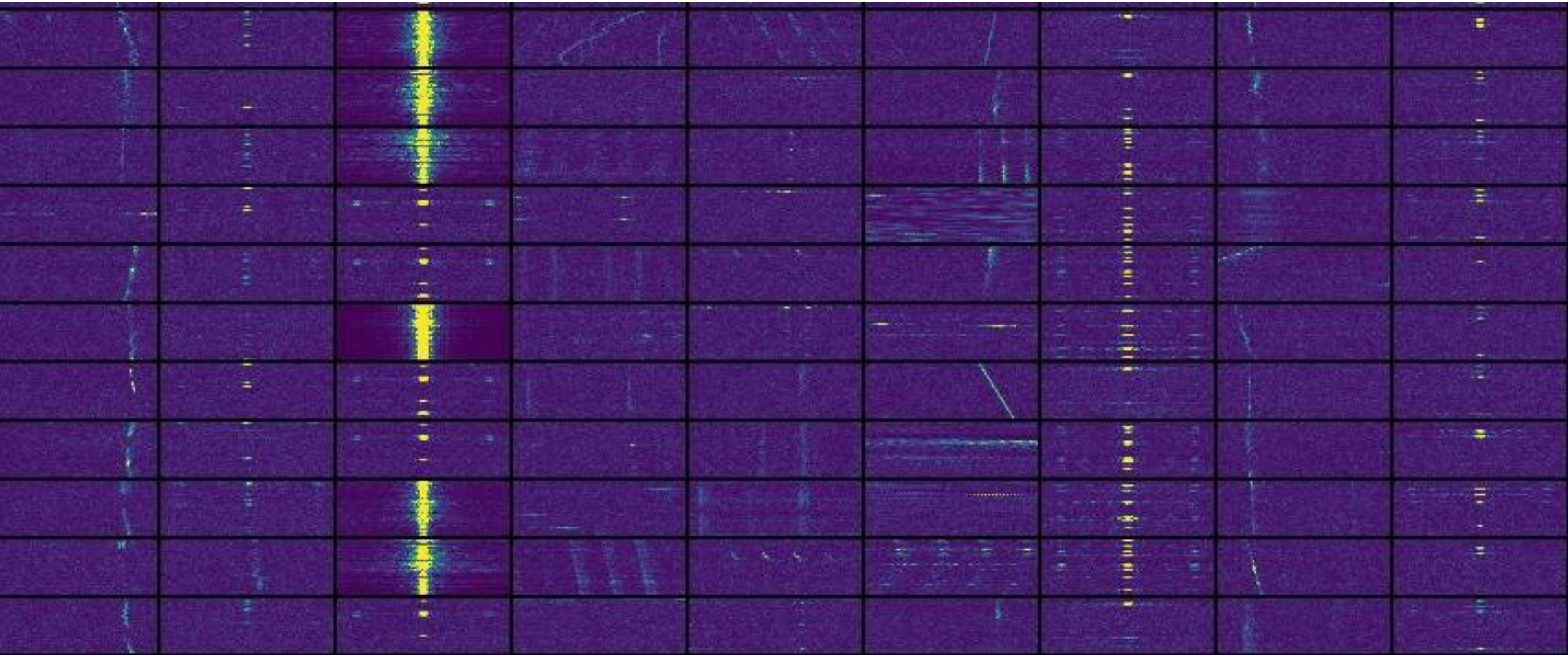
For n_clusters = 80 The average silhouette_score is : 0.45

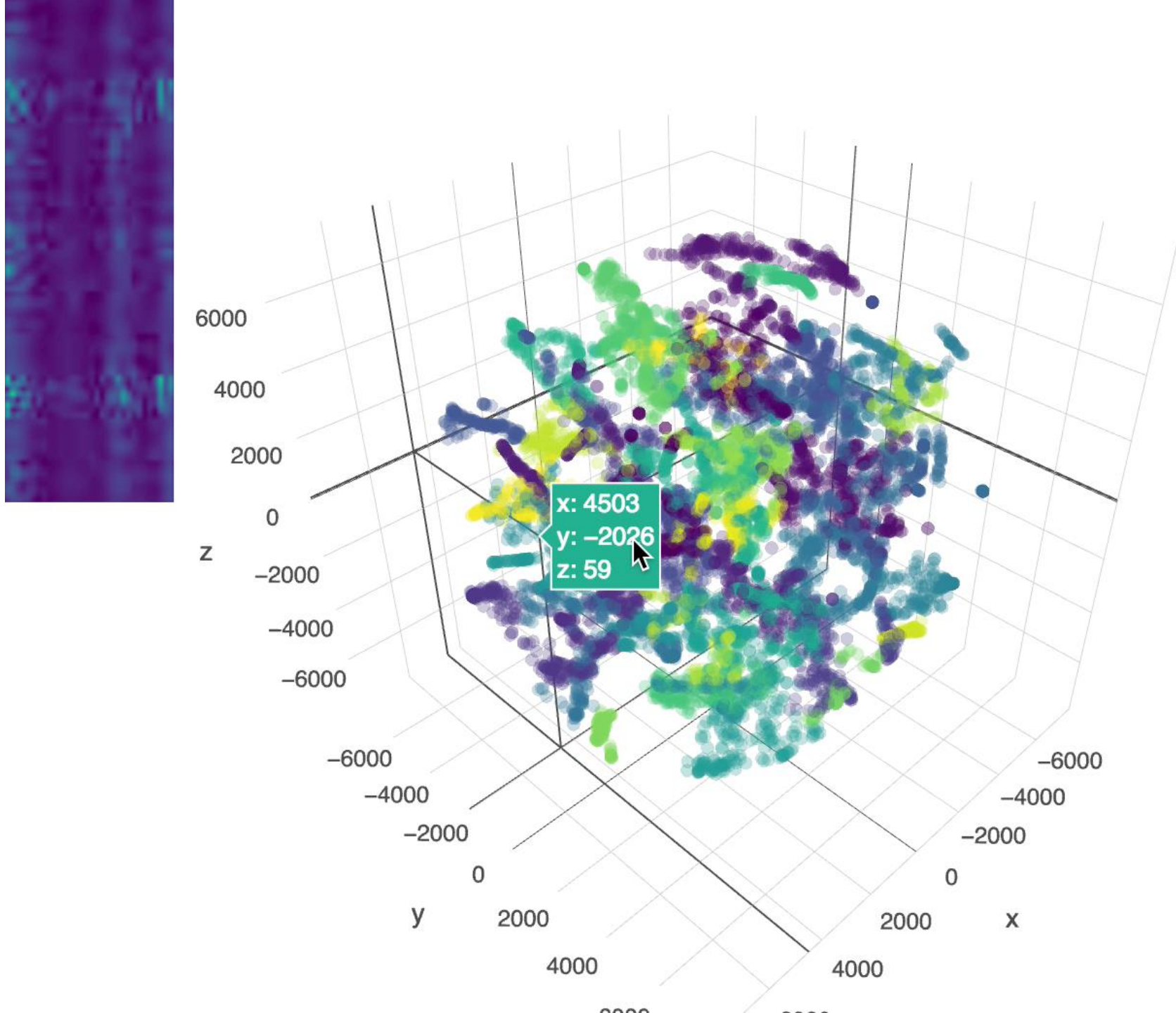


Cluster sample 1



Clusters sample 2



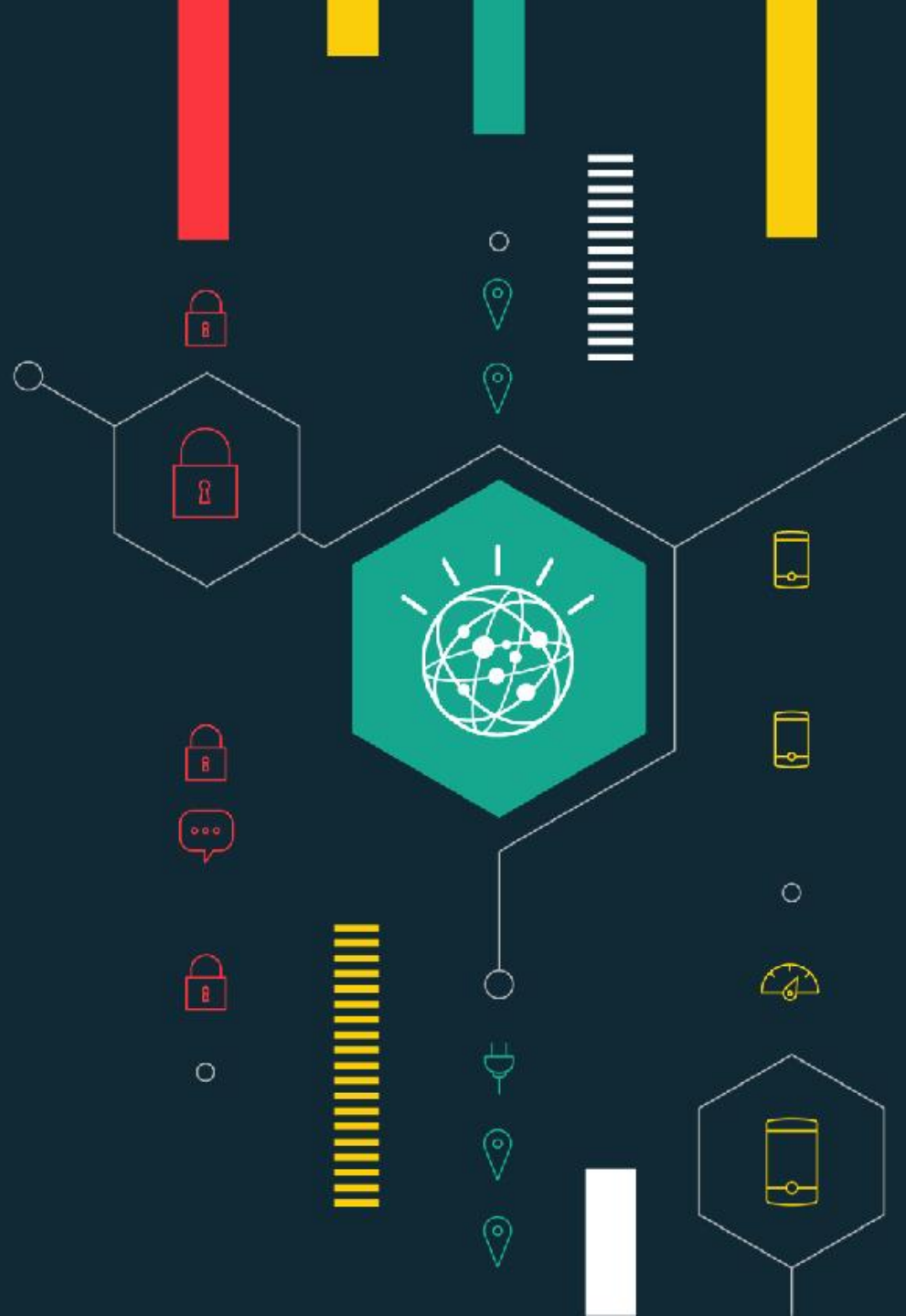

x: 4503
y: −2026
z: 59
6000
4000
2000
0
z
−2000
−4000
−6000
−6000
−4000
−2000
0
y
2000
4000
−6000
−4000
−2000
0
2000
x
4000

IBM Research | Africa

Toolchain for Software 2.0

IBM Research | Africa

IBM



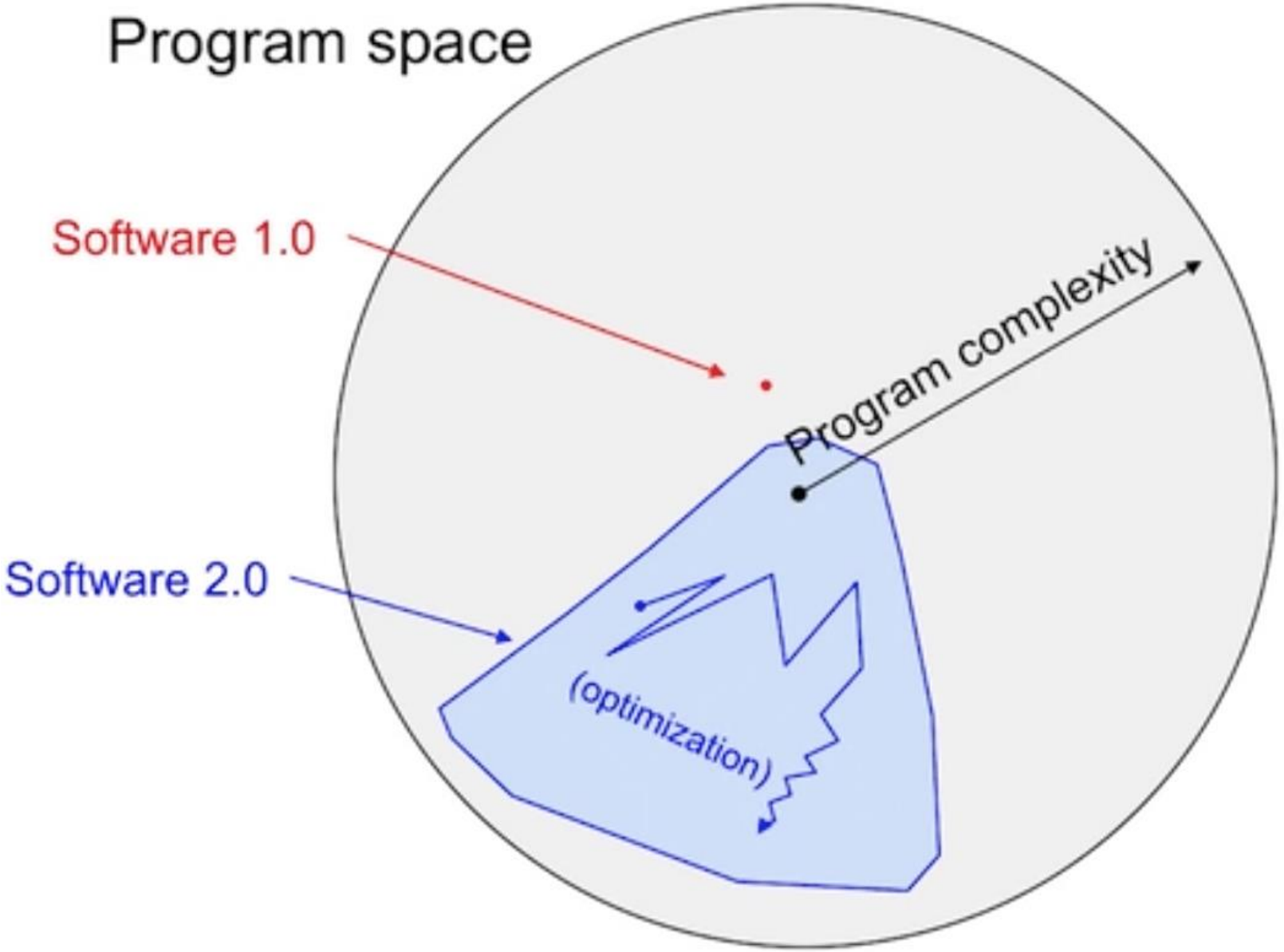
Building the Software 2.0 Stack by Andrej Karpathy from Tesla

June 8, 2018

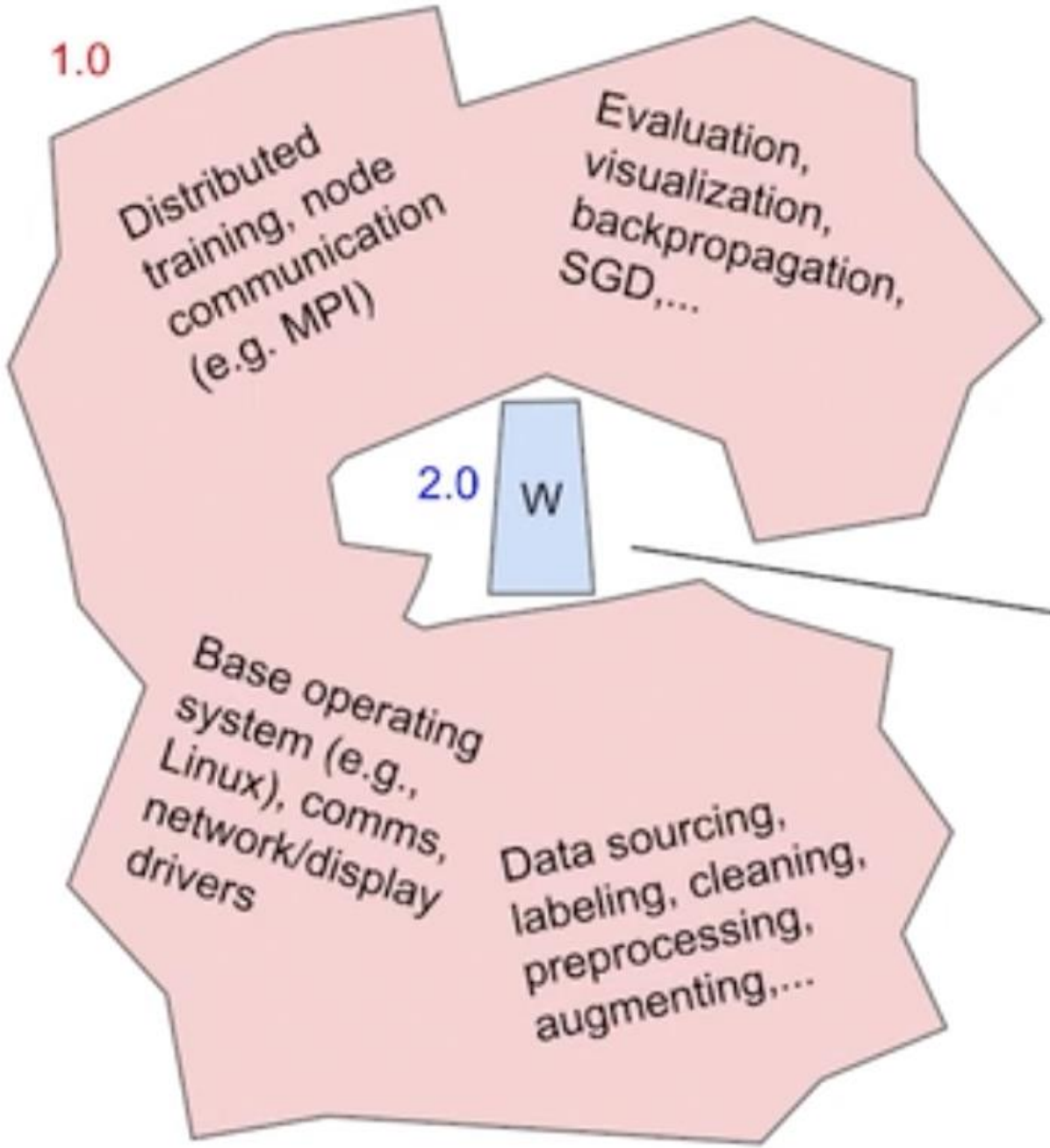
Autonomous Vehicles, Machine Learning, TRAIN AI 2018

Share

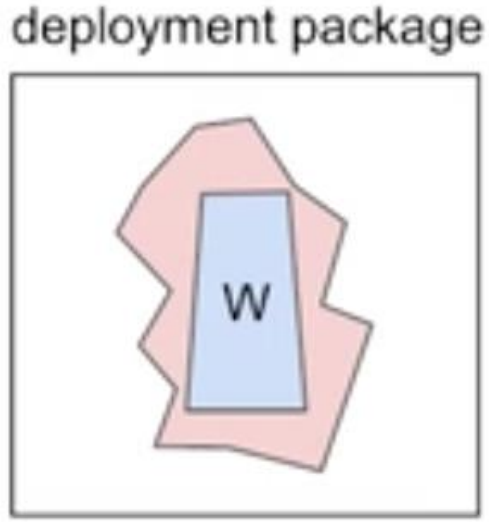
Software 2.0



(Karpathy, 2017)



Software 1.0 is not going anywhere...



(Karpathy, 2017)

Software 2.0

Andrej Karpathy (2017)

Software 2.0:

- “Software written by optimization in the language of weights”
- Engineered systems (1.0) -> Learned systems (2.0)
- End-to-end optimized systems
- Model skeletons + optimizers + supervised data
- Data labelers are the programmers

One Model To Learn Them All

Łukasz Kaiser
Google Brain
lukaszkaiser@google.com

Aidan N. Gomez*
University of Toronto
aidan@cs.toronto.edu

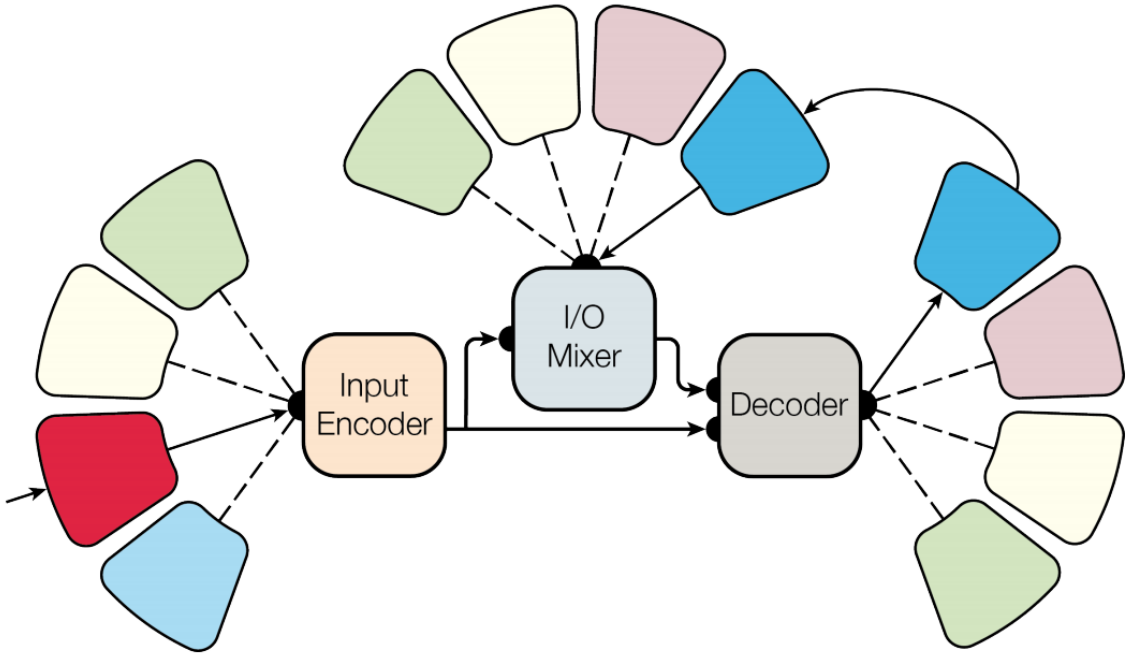
Noam Shazeer
Google Brain
noam@google.com

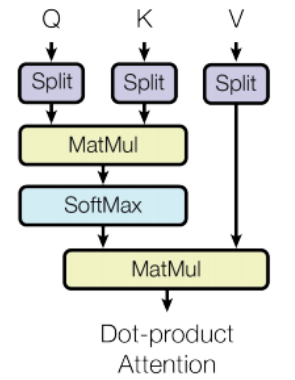
Ashish Vaswani
Google Brain
avaswani@google.com

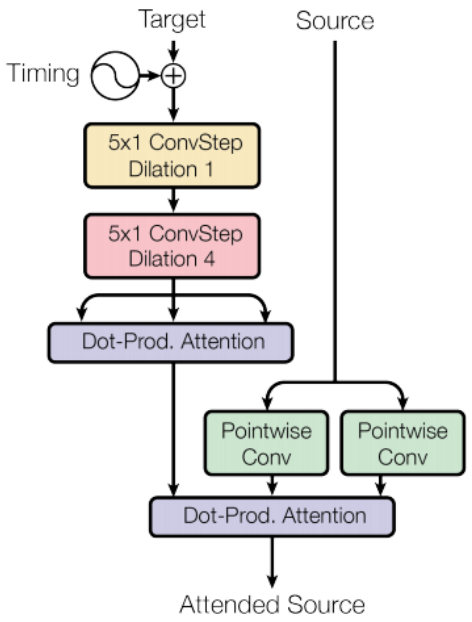
Niki Parmar
Google Research
nikip@google.com

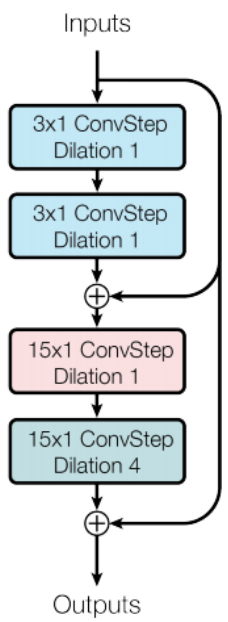
Llion Jones
Google Research
llion@google.com

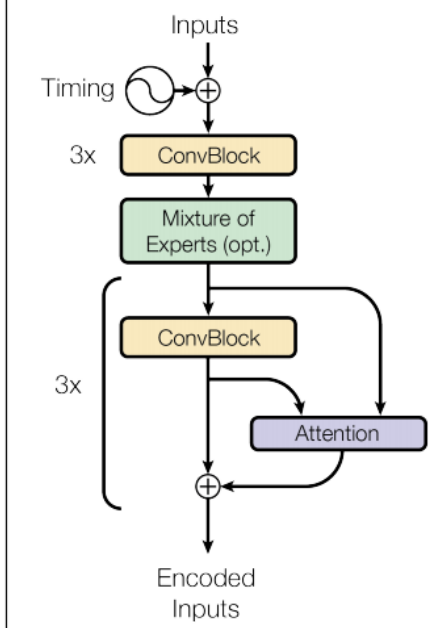
Jakob Uszkoreit
Google Research
usz@google.com

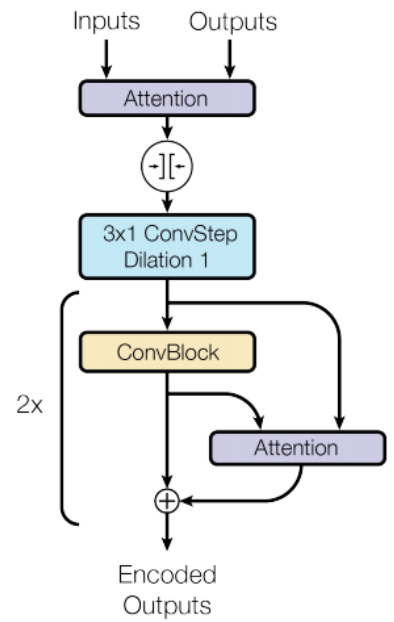


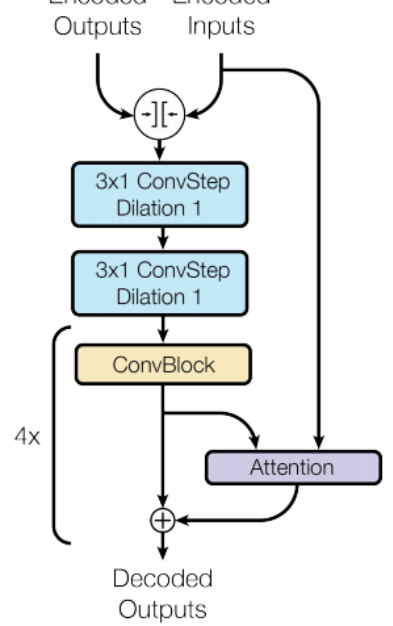












Toolchain for Software 2.0

Surrounding dataset/supervision infrastructure

IDE for Software 2.0 (Karpathy, 2017)

- Show full inventory/stats of current dataset
- Create/edit annotation layers for any datapoint
- Flag, escalate resolve discrepancies in multiple labels
- Flag, escalate datapoints thate are likely to be mislabeled
- Display predictions on an arbitrary set of test datapoints
- Autosuggest datapoints that should be labeled

Cognitive-assisted interactive labeling

Objectives

1. Visualize the quality of feature space
2. Define constraints/characteristics, could be through direct labeling
3. Update the feature space

Requirements

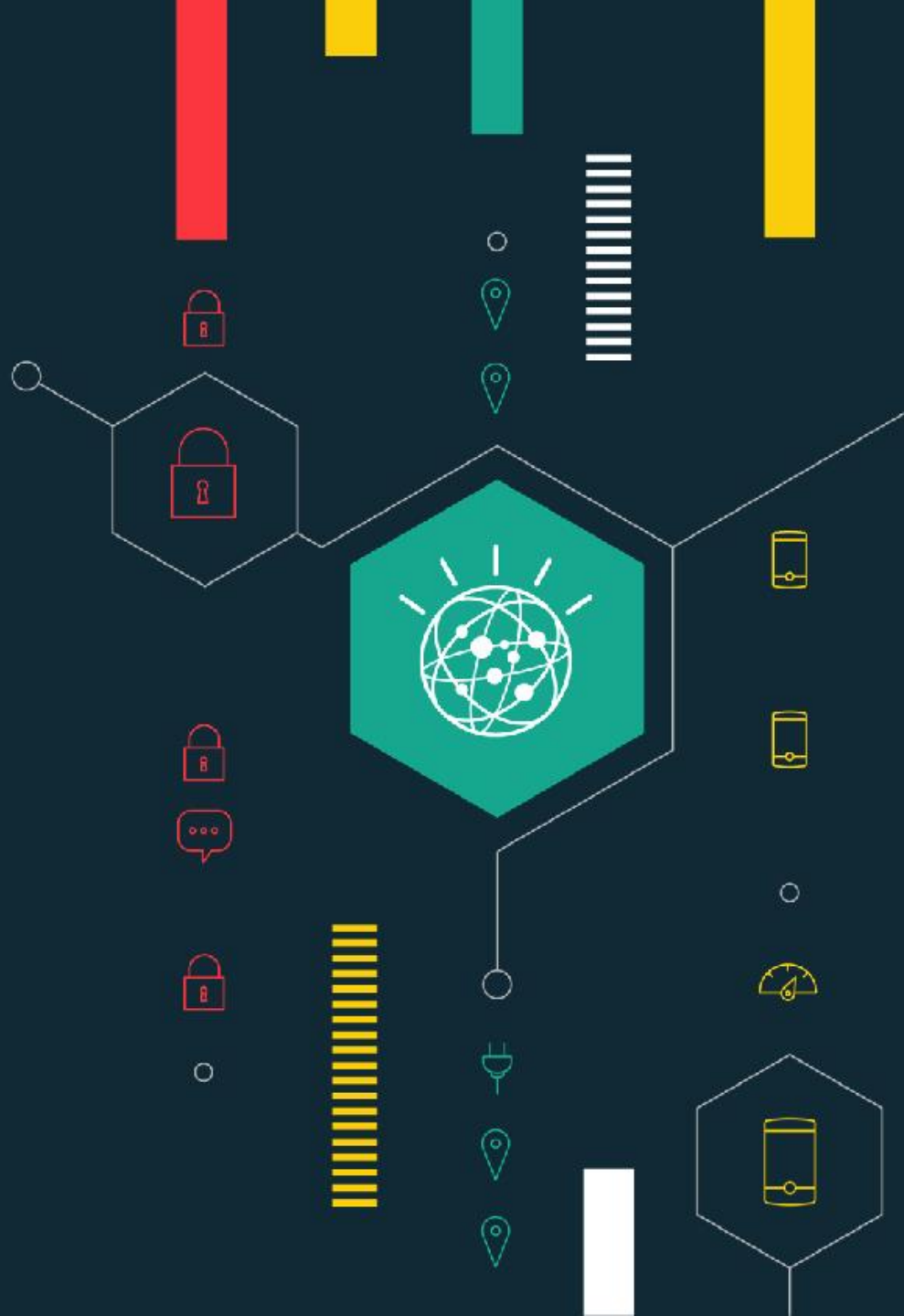
1. Iterative optimization
2. Dimensionality reduction for visualization
3. Fine-grained definition of sample characteristics

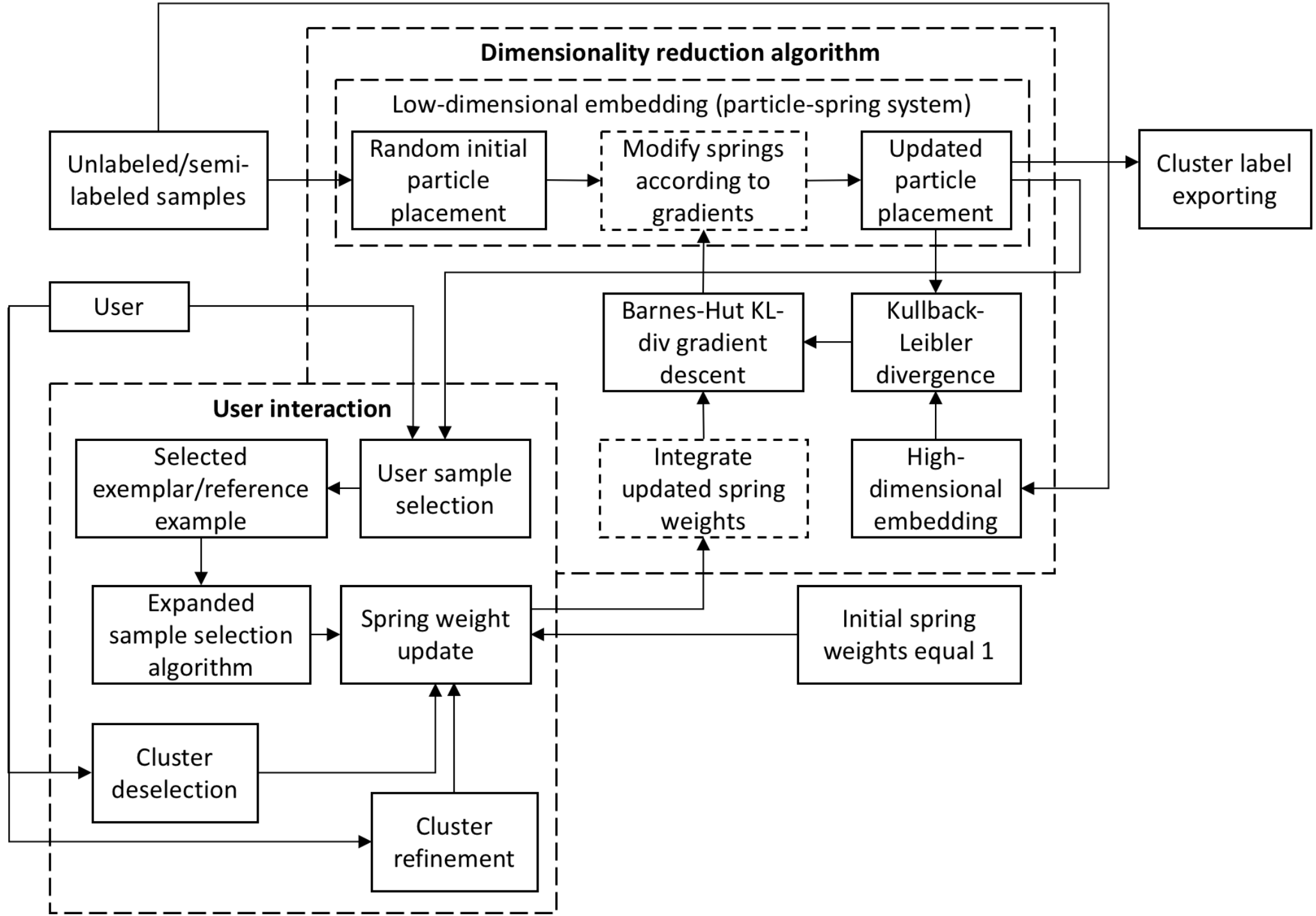
IBM Research | Africa

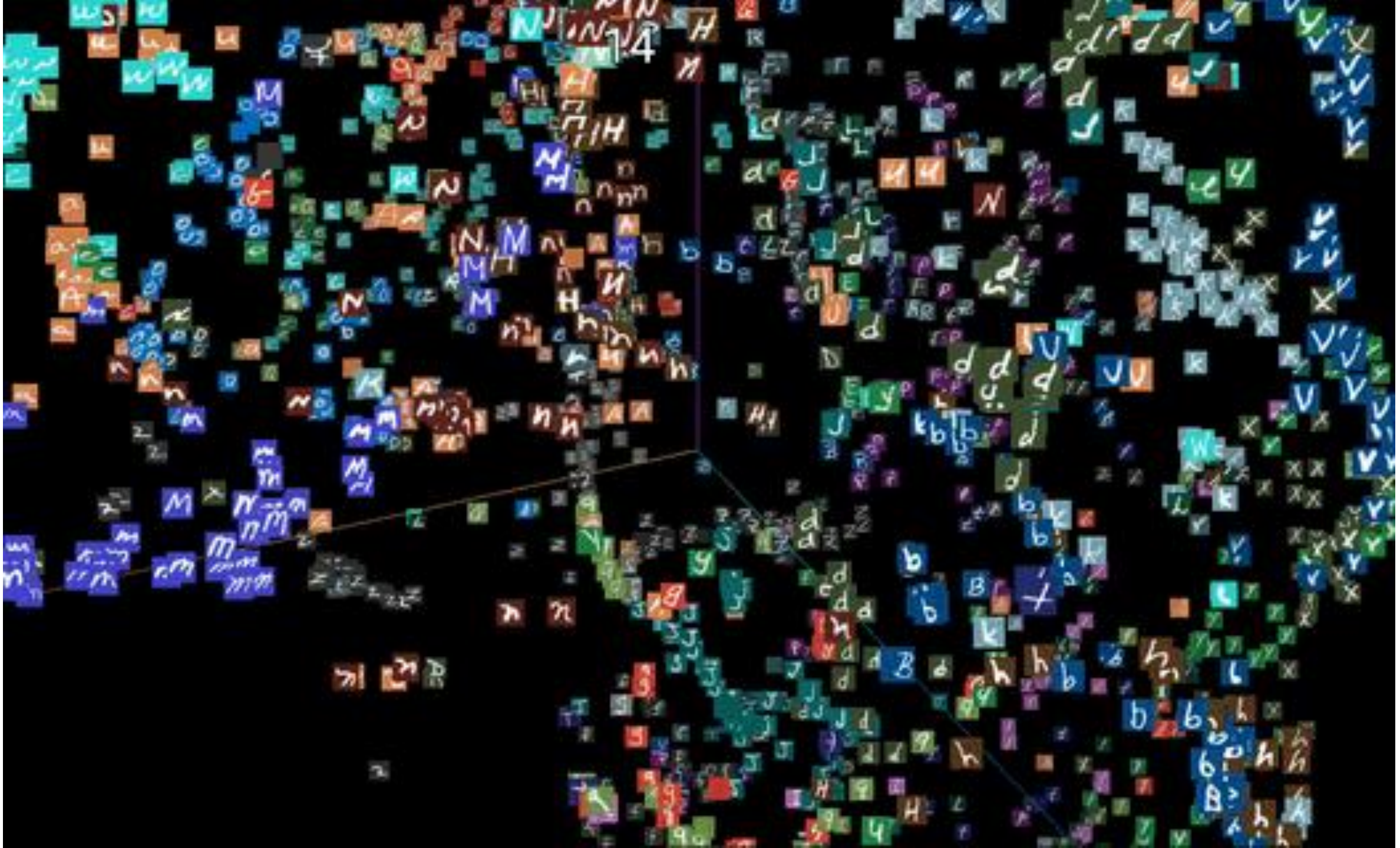
Cognitive-assisted interactive labeling

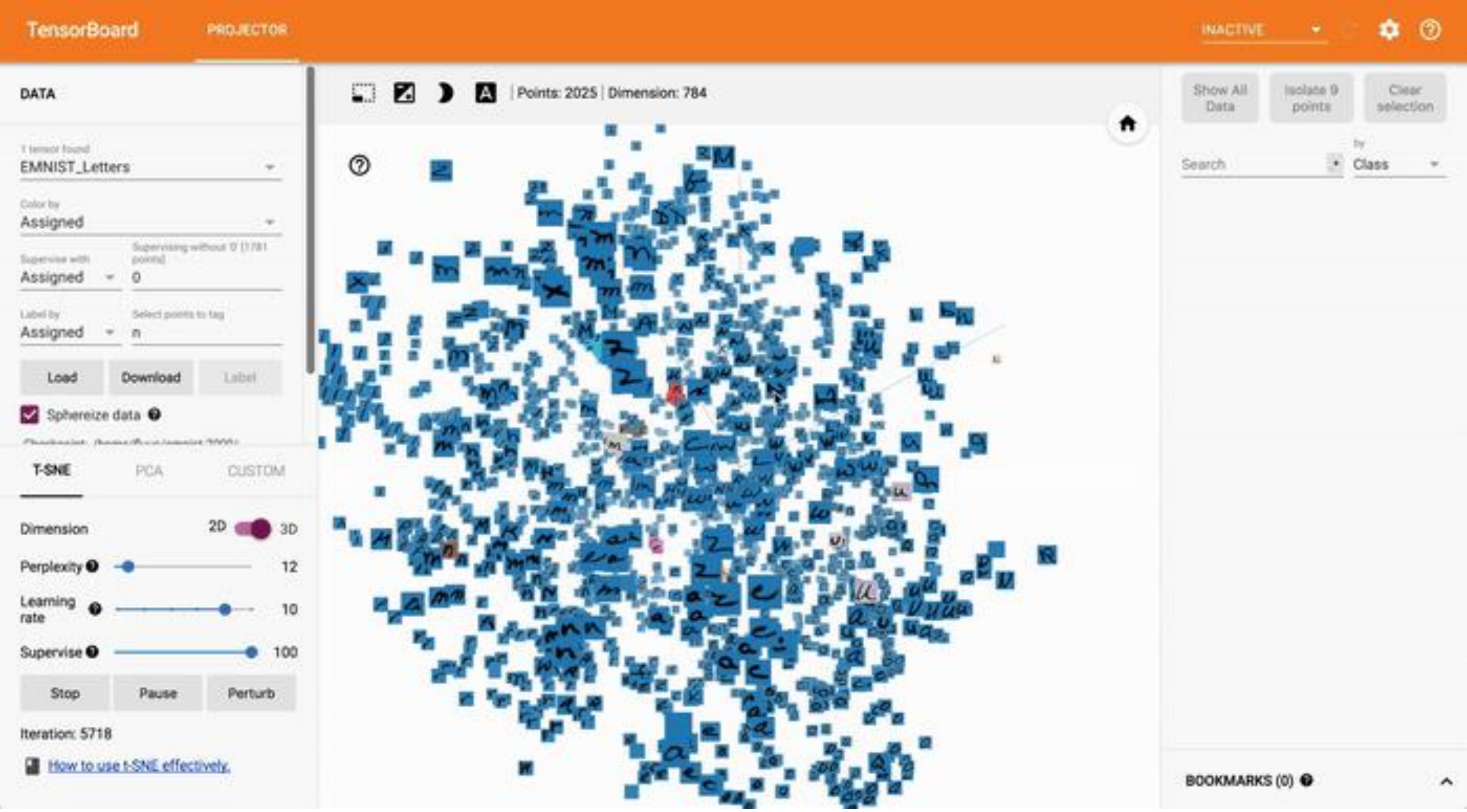
IBM Research | Africa

IBM



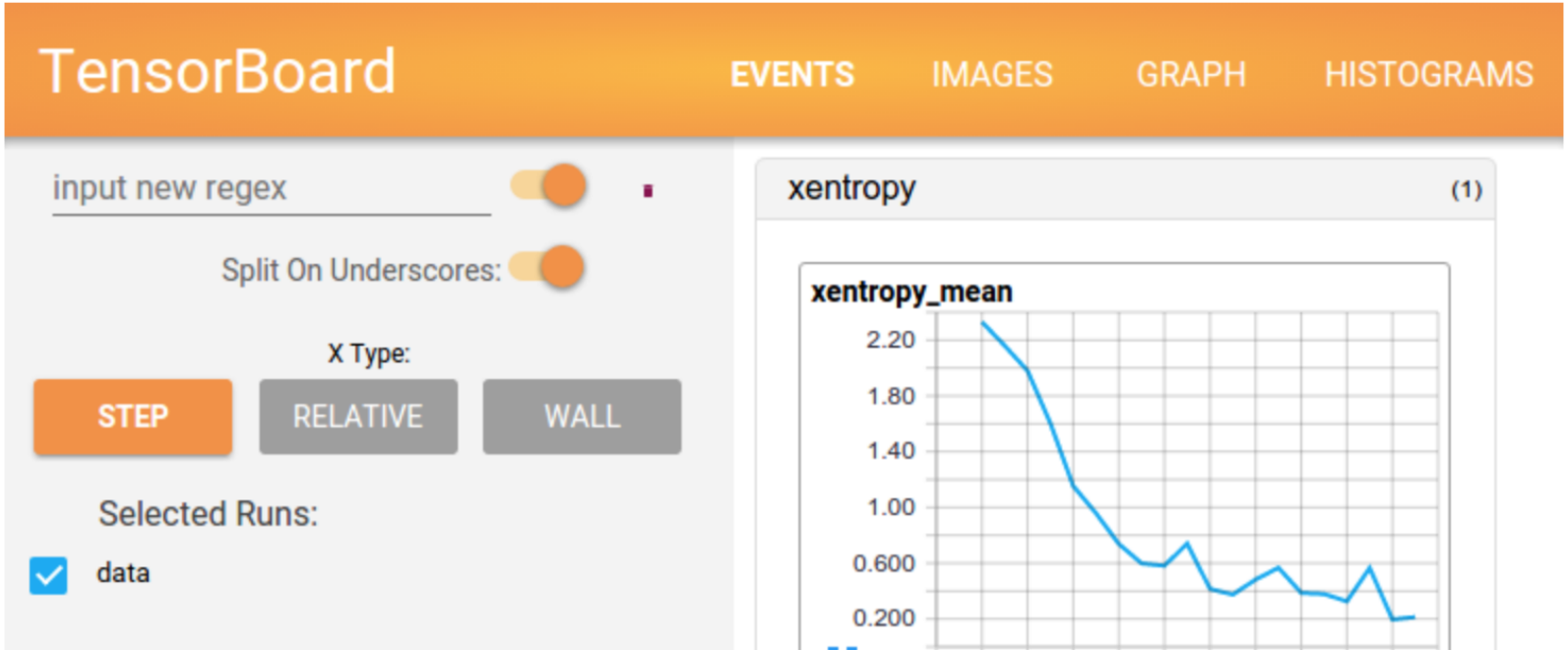

Dimensionality reduction algorithm
Low-dimensional embedding (particle-spring system)
Unlabeled/semi-labeled samples
Random initial particle placement
Modify springs according to gradients
Updated particle placement
Cluster label exporting
User
Barnes-Hut KL-div gradient descent
Kullback-Leibler divergence
User interaction
Selected exemplar/reference example
User sample selection
Integrate updated spring weights
High-dimensional embedding
Expanded sample selection algorithm
Spring weight update
Initial spring weights equal 1
Cluster deselection
Cluster refinement




TensorBoard
PROJECTOR
INACTIVE
DATA
1 tensor found
EMNIST_Letters
Color by
Assigned
Supervise with
Assigned
Supervising without 0 [1781 points]
0
Label by
Assigned
Select points to tag
n
Load
Download
Label
Sphereize data
T-SNE
PCA
CUSTOM
Dimension
2D
3D
Perplexity
12
Learning rate
10
Supervise
100
Stop
Pause
Perturb
Iteration: 5718
How to use t-SNE effectively.
Points: 2025 | Dimension: 784
Show All Data
Isolate 9 points
Clear selection
Search
by
Class
BOOKMARKS (0)

TensorBoard: Visualizing Learning

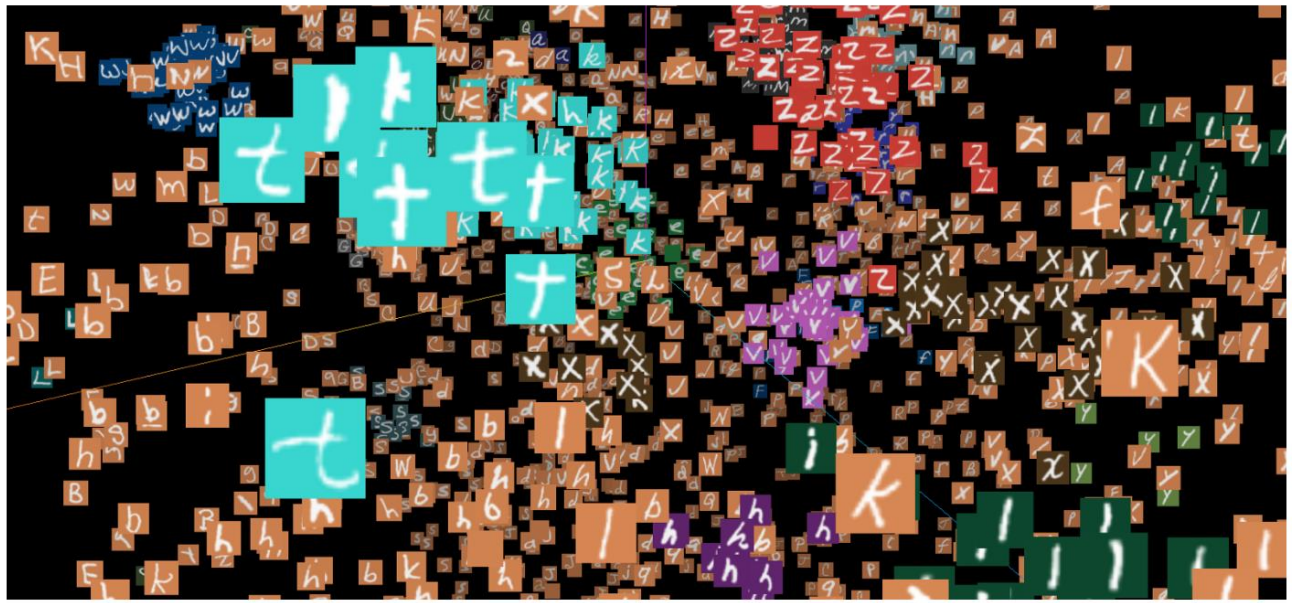
The computations you'll use TensorFlow for - like training a massive deep neural network - can be complex and confusing. To make it easier to understand, debug, and optimize TensorFlow programs, we've included a suite of visualization tools called TensorBoard. You can use TensorBoard to visualize your TensorFlow graph, plot quantitative metrics about the execution of your graph, and show additional data like images that pass through it. When TensorBoard is fully configured, it looks like this:



Francois Luus
Nov 25, 2017 · 7 min read

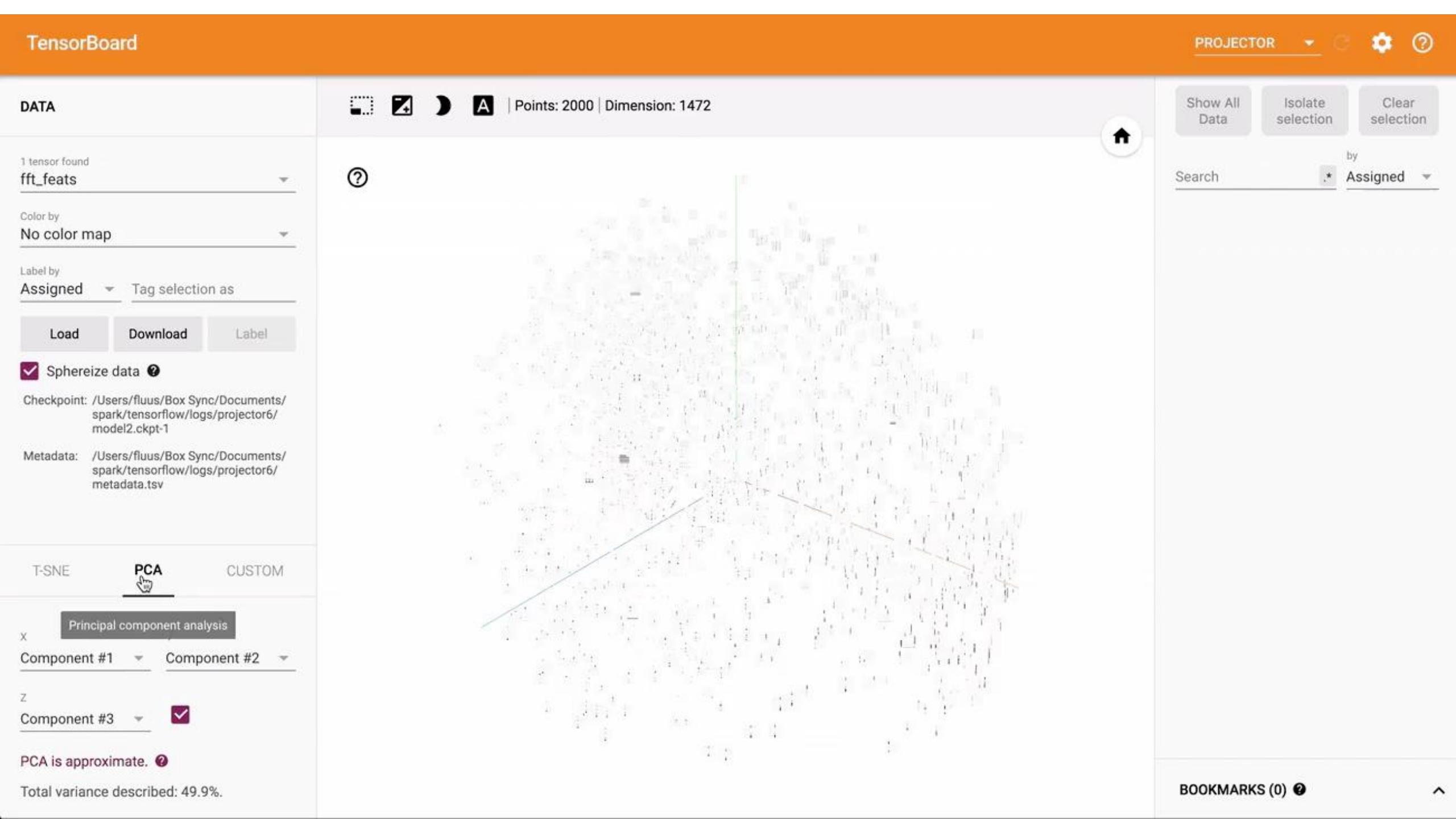
Interactive supervision with TensorBoard

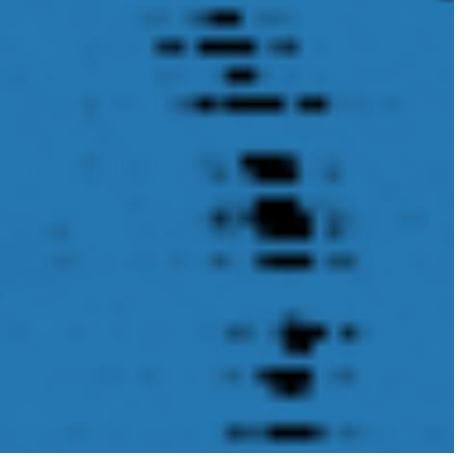
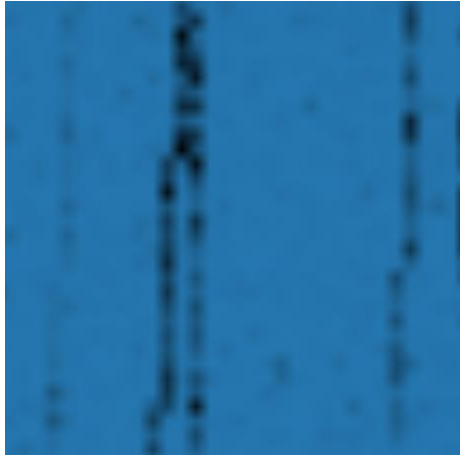
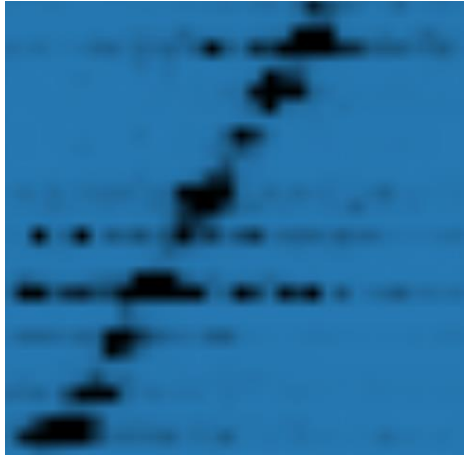
IBM Research AI

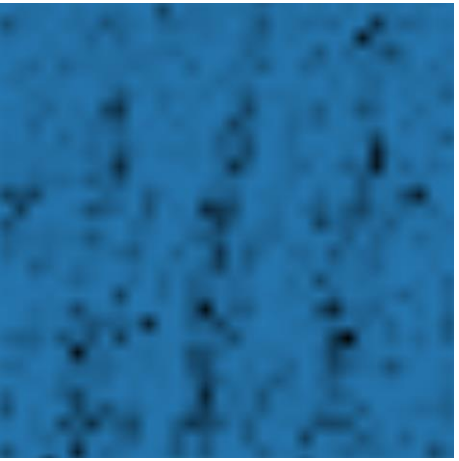
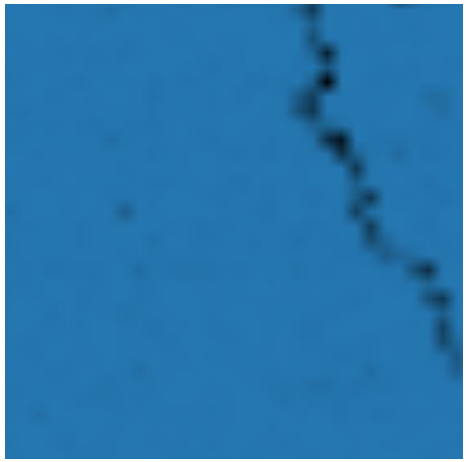


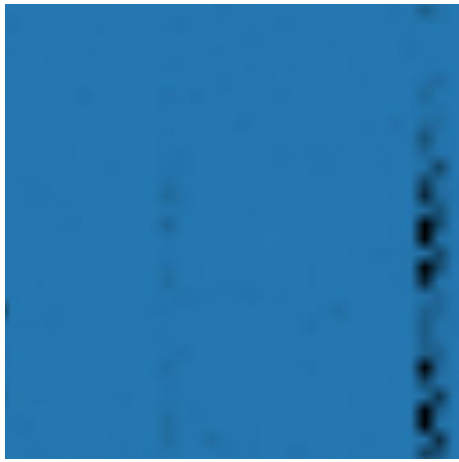
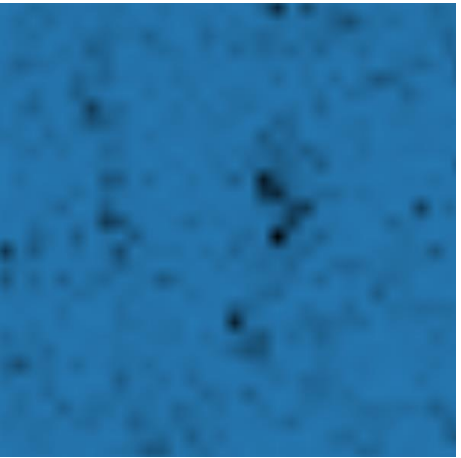
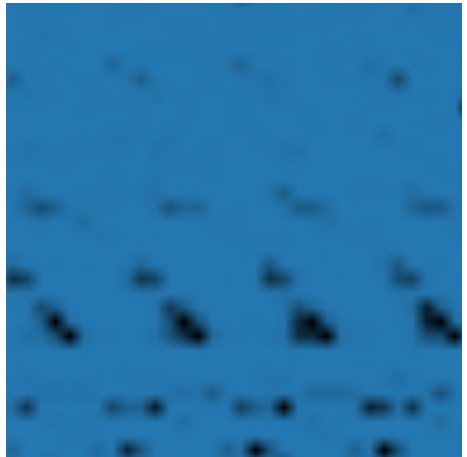
TensorBoard Projector showing labeled t-SNE

Originally published at www.ibm.com

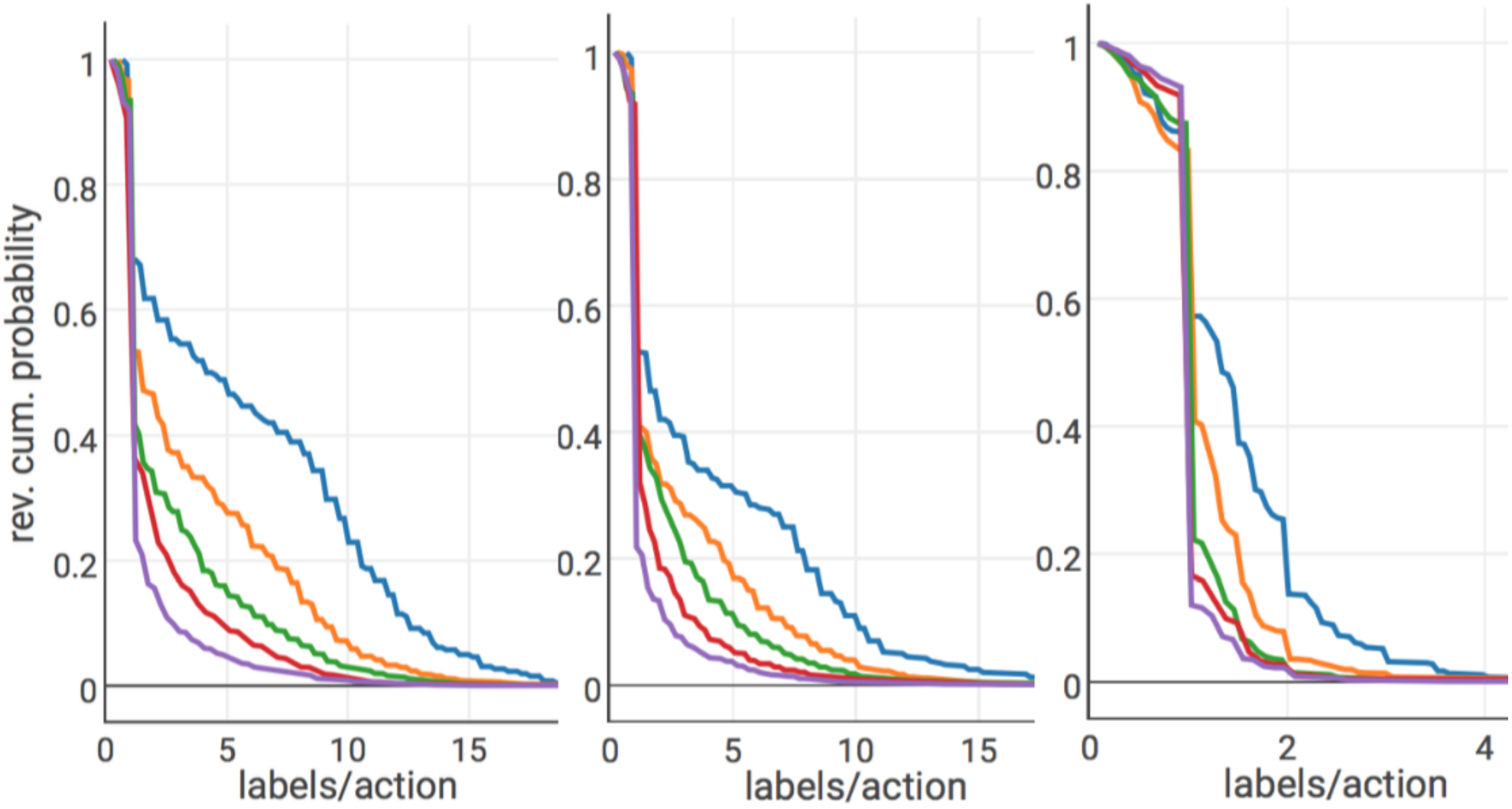

TensorBoard
PROJECTOR
DATA
1 tensor found
fft_feats
Color by
No color map
Label by
Assigned
Tag selection as
Load
Download
Label
Sphereize data
Checkpoint: /Users/fluus/Box Sync/Documents/spark/tensorflow/logs/projector6/model2.ckpt-1
Metadata: /Users/fluus/Box Sync/Documents/spark/tensorflow/logs/projector6/metadata.tsv
T-SNE
PCA
CUSTOM
Principal component analysis
X
Component #1
Component #2
Z
Component #3
PCA is approximate.
Total variance described: 49.9%.
Points: 2000 | Dimension: 1472
Show All Data
Isolate selection
Clear selection
by
Search
Assigned
BOOKMARKS (0)

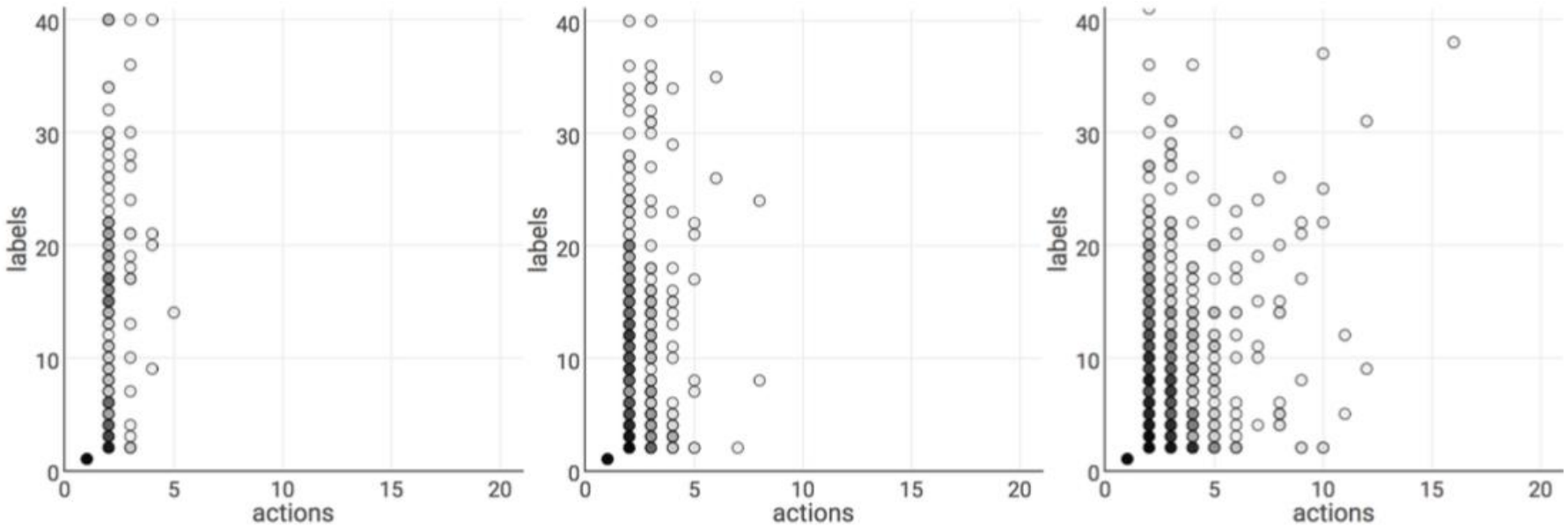



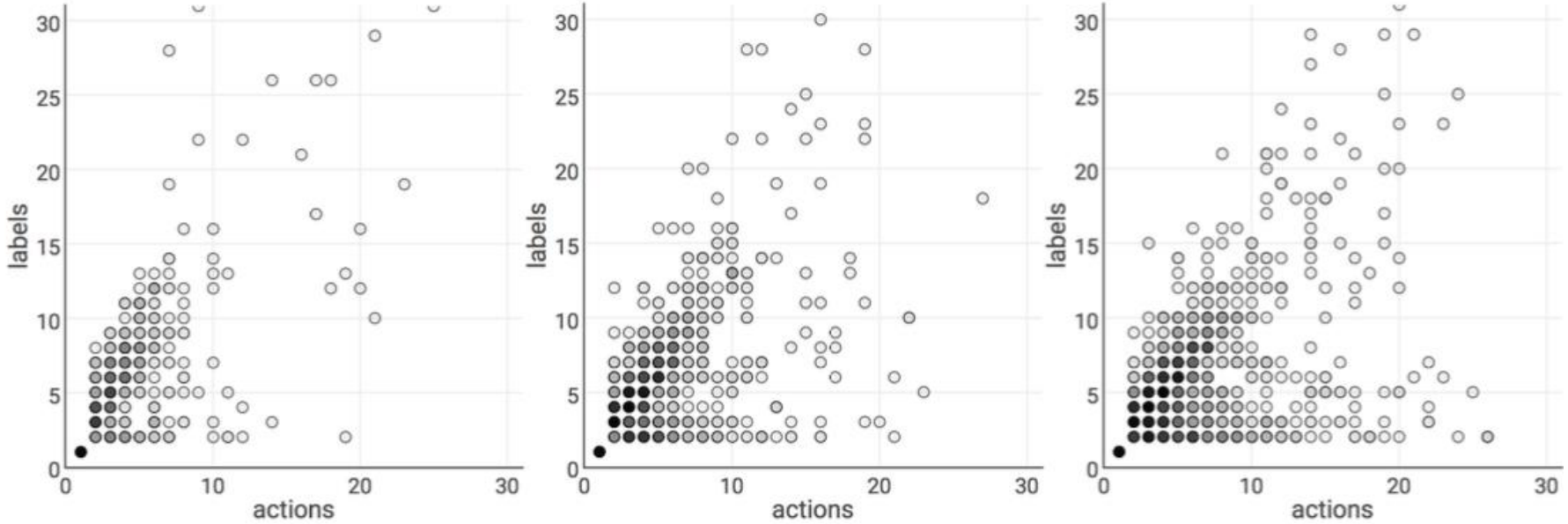
Interactive Supervision with t-SNE



(a) MNIST (b) Fashion (c) CIFAR10



(f) 20% [Fashion] (g) 40% [Fashion] (h) 60% [Fashion]



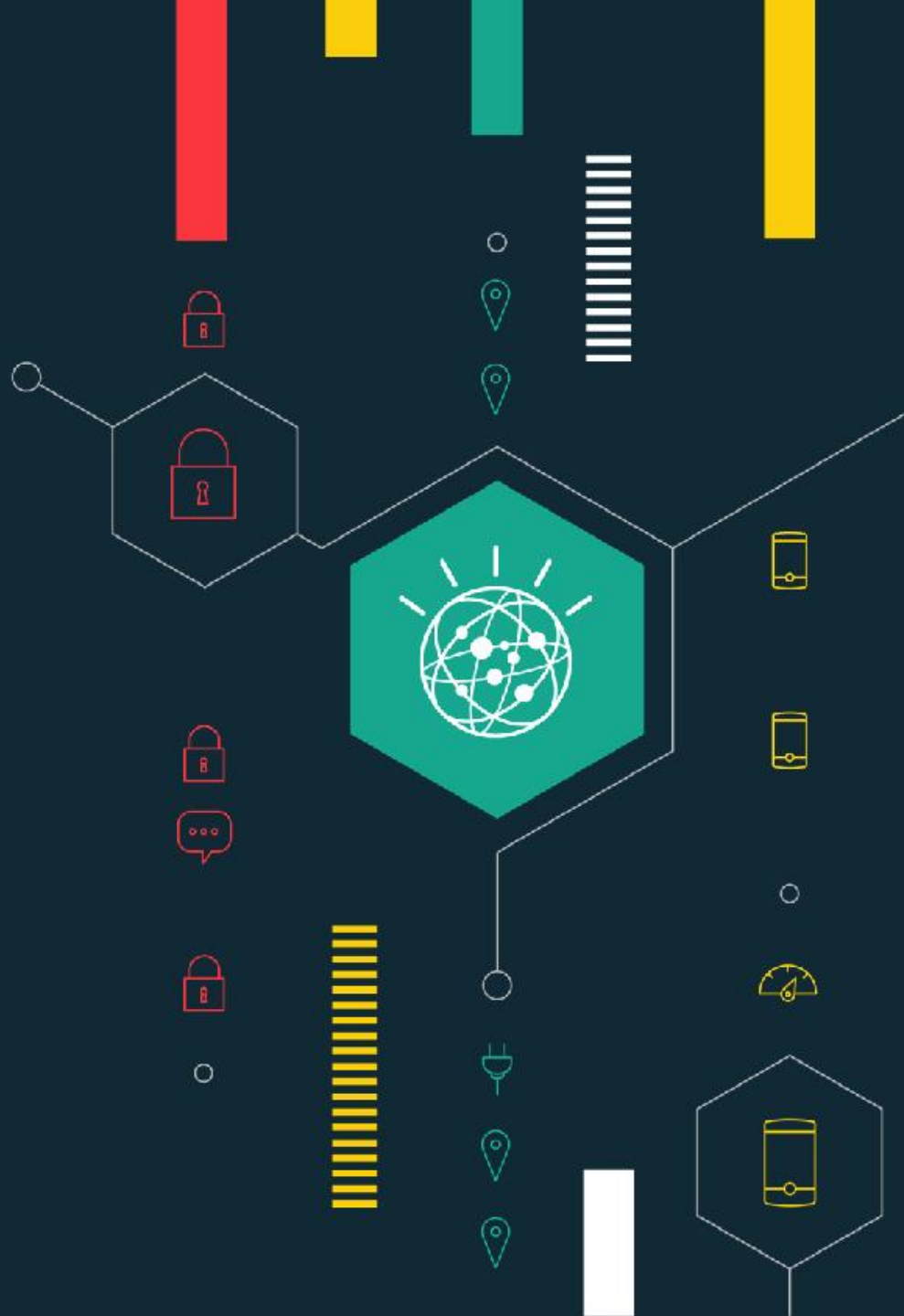
(k) 20% [CIFAR10] (l) 40% [CIFAR10] (m) 60% [CIFAR10]

Cognitive-assisted interactive labeling

1. Higher quality feature space means more homogeneous clusters
2. More homogeneous clusters means greater labeling efficiency
3. More labels means greater homogeneity
4. Edge cases and anomalies are obtained
5. Common classes defined

IBM **Research** | Africa

Questions



IBM

[D] Suggestion by Salesforce chief data scientist Discussion (i.redd.it)

submitted 21 hours ago by Prooffread3r to r/MachineLearning

72 comments share save hide report

Richard
@RichardSocher

Follow

Rather than spending a month figuring out an unsupervised machine learning problem, just label some data for a week and train a classifier.

RETWEETS 114 LIKES 289

5:47 PM - 10 Mar 2017

12 114 289

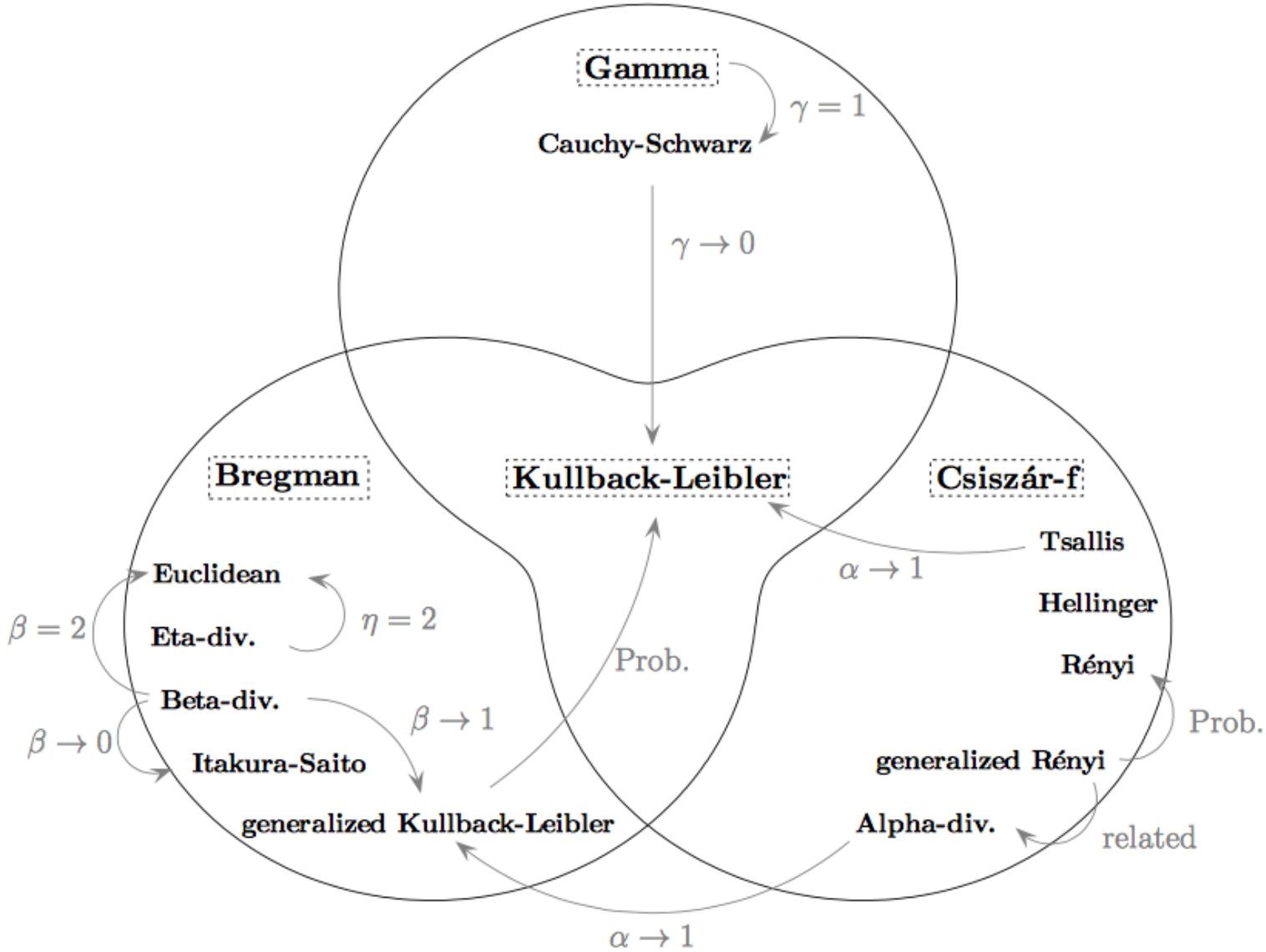


Figure 1: Overview over the families of divergences and their relationship to each other. The shortcut *Prob.* denotes the special case of probability densities. For sake of clarity we show the most important relations only and do not claim completeness.

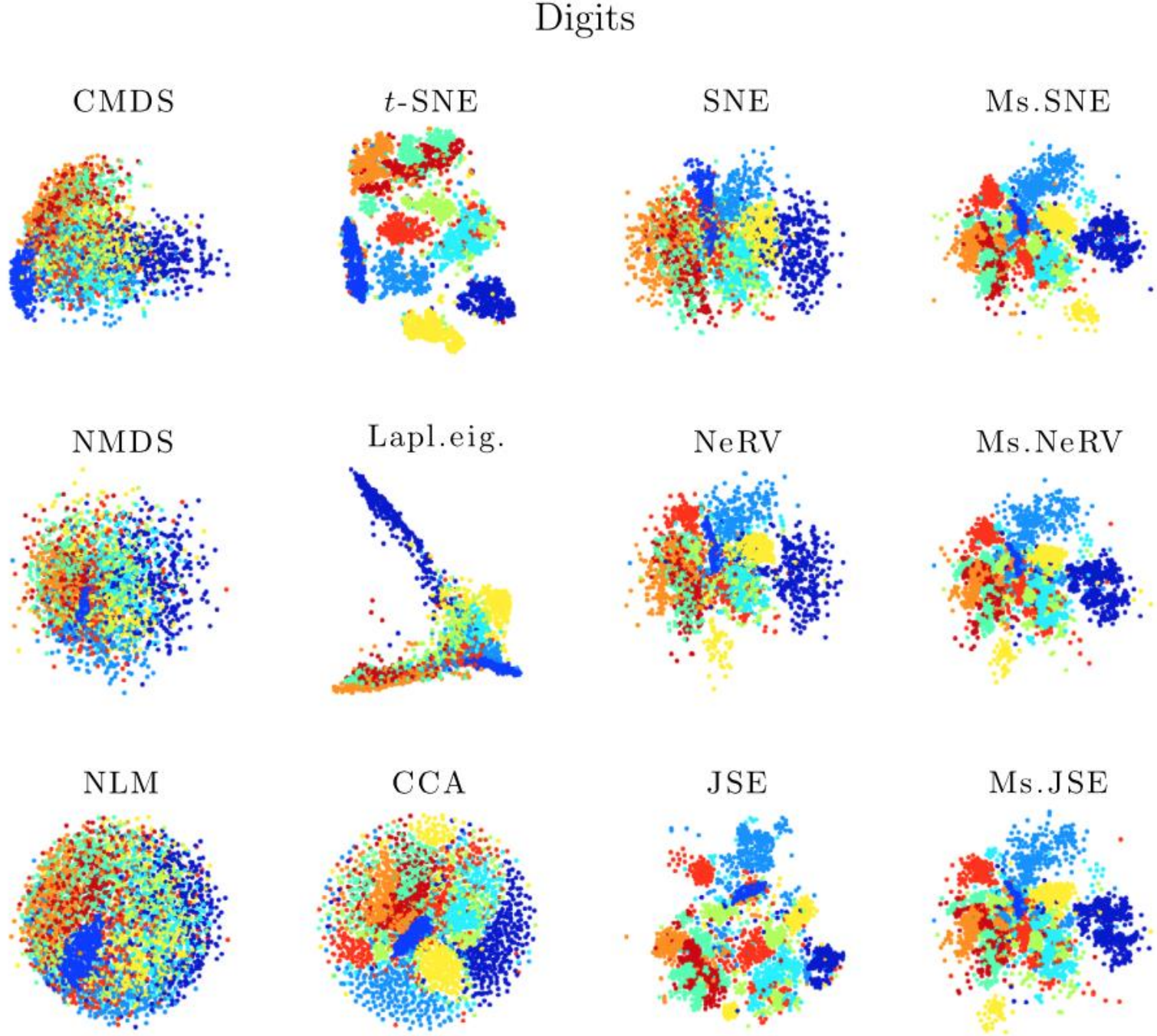


Fig. 6. Embeddings of all compared methods for Faces and Digits.

Barnes-Hut-SNE

Laurens van der Maaten
Pattern Recognition and Bioinformatics Group, Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands
lvdmaaten@gmail.com

Abstract

The paper presents an $\mathcal{O}(N \log N)$-implementation of t-SNE — an embedding technique that is commonly used for the visualization of high-dimensional data in scatter plots and that normally runs in $\mathcal{O}(N^2)$. The new implementation uses vantage-point trees to compute sparse pairwise similarities between the input data objects, and it uses a variant of the Barnes-Hut algorithm to approximate the forces between the corresponding points in the embedding. Our experiments show that the new algorithm, called Barnes-Hut-SNE, leads to substantial computational advantages over standard t-SNE, and that it makes it possible to learn embeddings of data sets with millions of objects.

https://arxiv.org/pdf/1301.3342.pdf

4.2 Approximating t-SNE Gradients

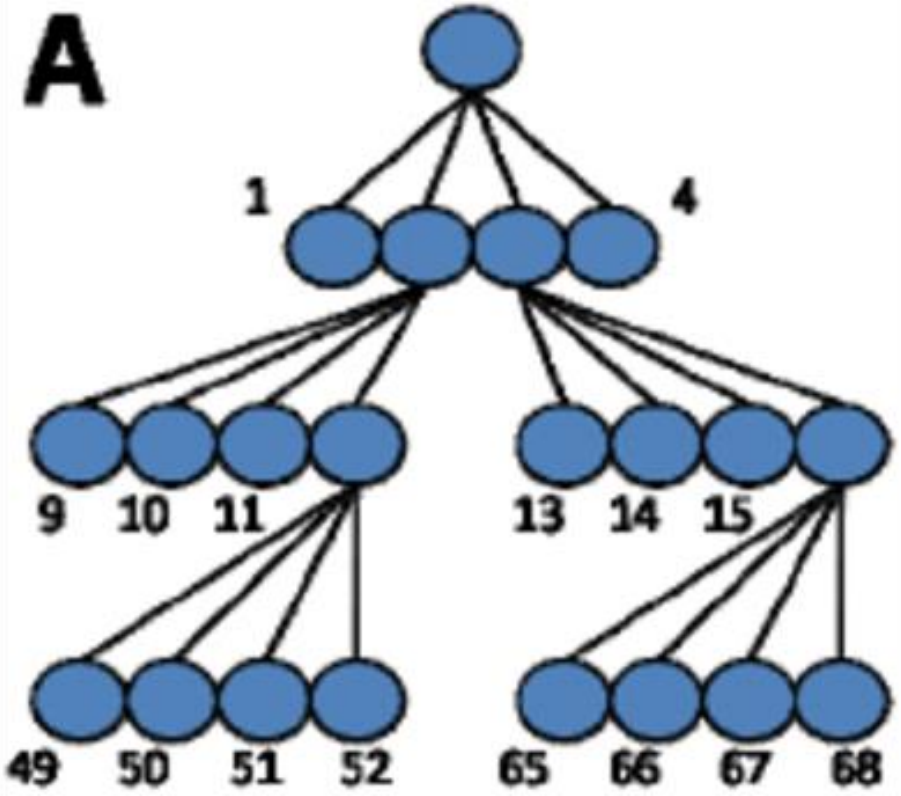
To approximate the t-SNE gradient, we start by splitting the gradient into two parts as follows:

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4(F_{attr} - F_{rep}) = 4\left(\sum_{j \neq i} p_{ij} q_{ij} Z(\mathbf{y}_i - \mathbf{y}_j) - \sum_{j \neq i} q_{ij}^2 Z(\mathbf{y}_i - \mathbf{y}_j)\right), \tag{8}$$

where F_{attr} denotes the sum of all attractive forces (the left sum), whereas F_{rep} denotes the sum of all repulsive forces (the right sum). Computing the sum of all attractive forces, F_{attr}, is computationally efficient; it can be done by summing over all non-zero elements of the sparse distribution P in $\mathcal{O}(uN)$. (Note that the term $q_{ij}Z = (1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}$ can be computed in $\mathcal{O}(1)$.) However, a naive computation of the sum of all repulsive forces, F_{rep}, is $\mathcal{O}(N^2)$. We now develop a Barnes-Hut algorithm to approximate F_{rep} efficiently in $\mathcal{O}(N \log N)$.

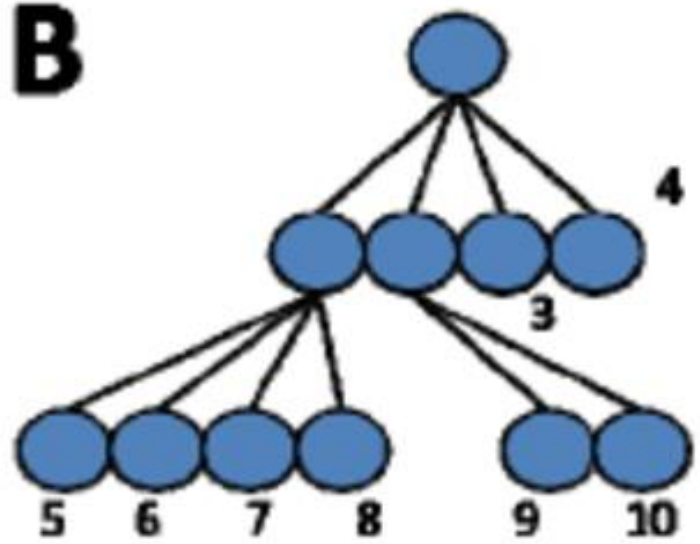
Consider three points $\mathbf{y}_i$, $\mathbf{y}_j$, and $\mathbf{y}_k$ with $\|\mathbf{y}_i - \mathbf{y}_j\| \approx \|\mathbf{y}_i - \mathbf{y}_k\| \gg \|\mathbf{y}_j - \mathbf{y}_k\|$. In this situation, the contributions of $\mathbf{y}_j$ and $\mathbf{y}_k$ to F_{rep} will be roughly equal. The Barnes-Hut algorithm [1] exploits this by i) constructing a quadtree on the current embedding, ii) traversing the quadtree using a depth-first search, and iii) at every node in the quadtree, deciding whether the corresponding cell can be used as a "summary" for the gradient contributions of all points in that cell.

A



1		9	10
		11	49 50 51 52
13	14	4	
15	65 66 67 68		

B



5	6	9	10
7	8		
3		4	

https://arxiv.org/pdf/1301.3342.pdf

Quadtree. A quadtree is a tree in which each node represents a rectangular *cell* with a particular center, width, and height. Non-leaf nodes have four children that split up the cell into four smaller cells (quadrants) that lie "northwest", "northeast", "southwest", and "southeast" of the center of the parent node (see Figure 1 for an illustration). Leaf nodes represent cells that contain at most one point of the embedding; the root node represents the cell that contains the complete embedding. In each node, we store the center-of-mass of the embedding points that are located inside the corresponding cell, $\mathbf{y}_{cell}$, and the total number of points that lie inside the cell, N_{cell}. A quadtree has $\mathcal{O}(N)$ nodes and can be constructed in $\mathcal{O}(N)$ time by inserting the points one-by-one, splitting a leaf node whenever a second point is inserted in its cell, and updating $\mathbf{y}_{cell}$ and N_{cell} of all visited nodes.

Approximating the gradient. To approximate the repulsive part of the gradient, F_{rep}, we

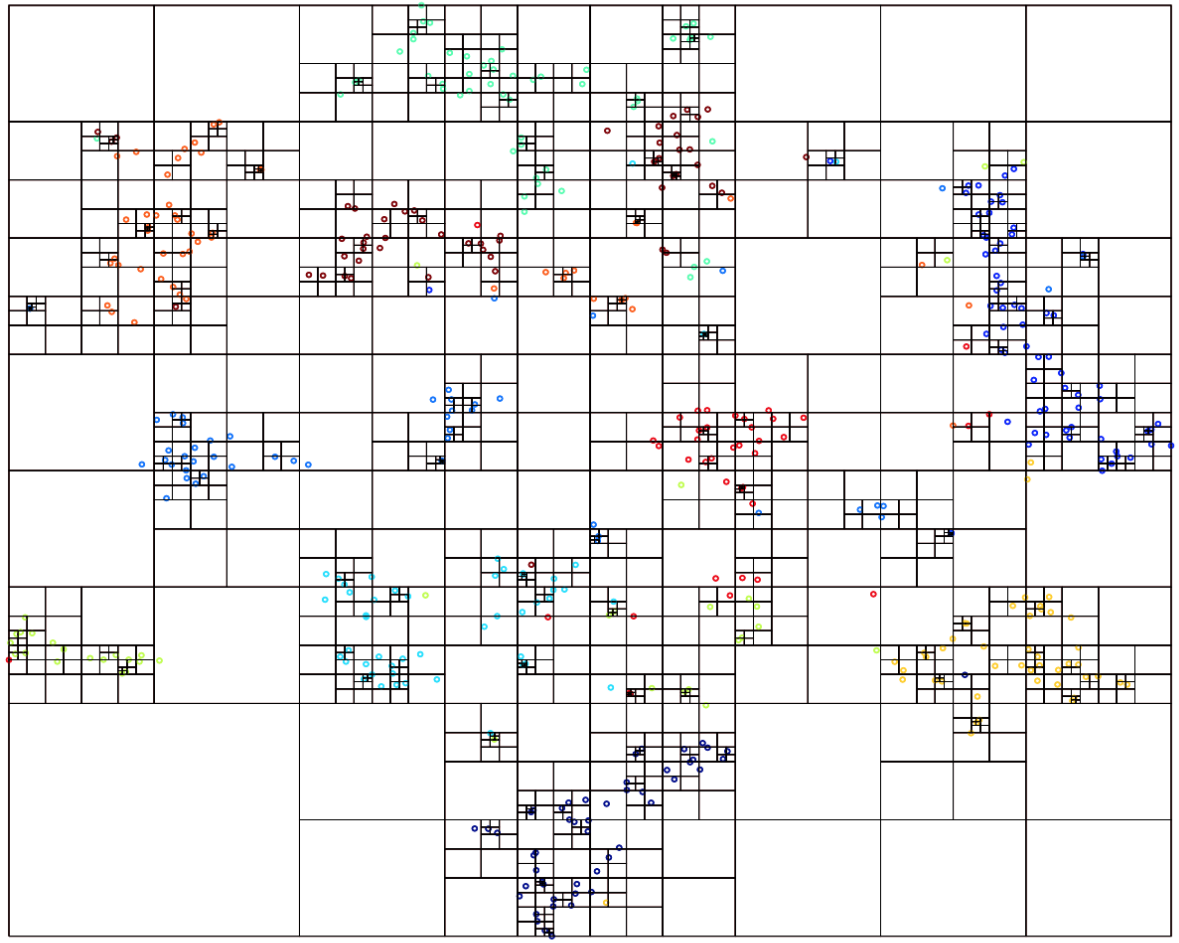


Figure 1: Quadtree constructed on a two-dimensional t-SNE embedding of 500 MNIST digits (the colors of the points correspond to the digit classes). Note how the quadtree adapts to the local point density in the embedding.

Optimization Equivalence of Divergences Improves Neighbor Embedding

Zhirong Yang[2] ZHIRONG.YANG@AALTO.FI
Jaakko Peltonen[1,4] JAAKKO.PELTONEN@AALTO.FI
Samuel Kaski[1,3] SAMUEL.KASKI@AALTO.FI
[1]Helsinki Institute for Information Technology HIIT, [2]Department of Information and Computer Science, Aalto University, Finland, [3]Department of Computer Science, University of Helsinki, and [4]University of Tampere

Abstract

Visualization methods that arrange data objects in 2D or 3D layouts have followed two main schools, methods oriented for graph layout and methods oriented for vectorial embedding. We show the two previously separate approaches are tied by an optimization equivalence, making it possible to relate methods from the two approaches and to build new methods that take the best of both worlds. In detail, we prove a

1. Introduction

We address two research problems: nonlinear dimensionality reduction (NLDR) of vectorial data and graph layout. In NLDR, given a set of data points represented with high-dimensional feature vectors or a distance matrix between such vectors, low-dimensional coordinates are sought for each data point. In graph layout, given a set of nodes (vertices) and a set of edges between node pairs, the task is to place the nodes on a 2D or 3D display. Solutions to both research problems are widely used in data visualization.

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4(F_{attr} - F_{rep}) = 4\left(\sum_{j\neq i} p_{ij}q_{ij}Z(\mathbf{y}_i - \mathbf{y}_j) - \sum_{j\neq i} q_{ij}^2 Z(\mathbf{y}_i - \mathbf{y}_j)\right)$$

amples). In contrast, ws-SNE uses edge repulsion to handle such cases of imbalanced degrees. When the d_i are not uniform, ws-SNE behaves differently from conventional s-SNE, which can be explained by its gradient $\frac{\partial \mathcal{J}_{\text{ws-SNE}}(Y)}{\partial y_i} = \sum_j p_{ij}q_{ij}^{\theta}(y_i - y_j) - cd_id_jq_{ij}^{1+\theta}(y_i - y_j)$, where $c = \sum_{ij} p_{ij}/(\sum_{ij} d_id_jq_{ij})$ is the connection scalar, and $\theta = 0$ for Gaussian q and $\theta = 1$ for Cauchy q. The first term in the summation is for attraction of nodes and the second for repulsion. Compared with the s-SNE gradient, the repulsion part is weighted by d_id_j in ws-SNE. That is, important nodes have extra repulsive force with the others and thus tend to be placed farther. This edge-repulsion strategy has been shown to be effective in graph drawing to overcome the "crowding problem", namely, many mapped points becoming crowded in the center of the display.

egy from Noack (2007), we insert weights M in the repulsive term: $\mathcal{J}_{\text{weighted-EE}}(Y) = \sum_{ij} p_{ij}\|y_i - y_j\|^2 + \lambda \sum_{ij} M_{ij} \exp\left(-\|y_i - y_j\|^2\right)$, where $M_{ij} = d_id_j$, and the vector d measures importance of the nodes. We use degree centrality as the measurement, i.e., d_i=degree of the i-th node. $\mathcal{J}_{\text{weighted-EE}}(Y)$ has downsides: it needs a user-set edge repulsion weight λ, and is not invariant to scaling of p. By Theorem 1 we create a corresponding improved method, ws-SNE, minimizing a nonseparable divergence.

Proposition 5. *Weighted EE is a separable divergence minimizing method and its non-separable variant is ws-SNE.* **Proof:** Writing $q_{ij} = \exp(-\|y_i - y_j\|^2)$ with $q_{ii} = 0$, we have $\mathcal{J}_{\text{weighted-EE}}(Y) = D_{\text{I}}(p||\lambda M \circ q) + F(\lambda)$, where $\circ$ denotes element-wise product and $F(\lambda) = C(\lambda) + \sum_{ij} p_{ij} \ln d_id_j$ is a constant to Y. In the final and most important step, by a special case of Theorem 1: $\arg\min_Y D_{\text{KL}}(p||M \circ q) = \arg\min_Y [\min_{\lambda\geq 0} D_{\text{I}}(p||\lambda M \circ q)]$, we obtain a new variant of SNE which minimizes $D_{\text{KL}}(p||M \circ q)$ over Y:

$$\mathcal{J}_{\text{ws-SNE}}(Y) = -\sum_{ij} p_{ij} \ln q_{ij} + \ln \sum_{ij} M_{ij}q_{ij} + \text{constant}$$

http://proceedings.mlr.press/v32/yange14.pdf