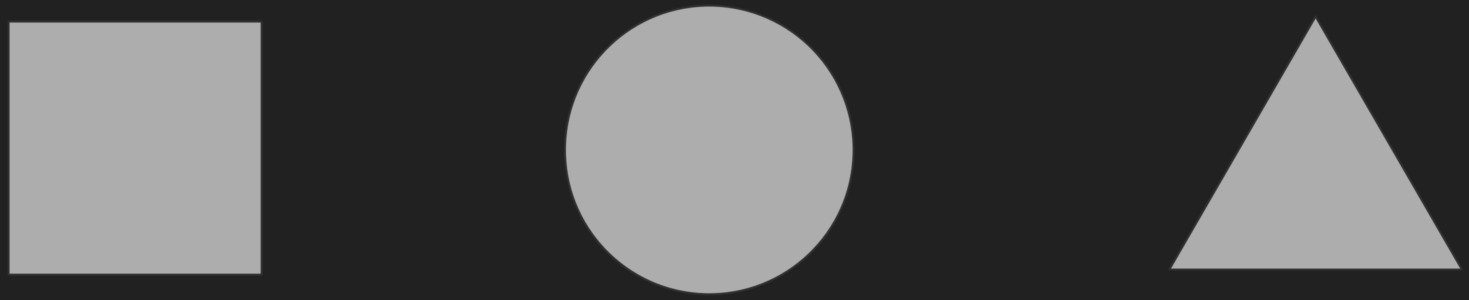
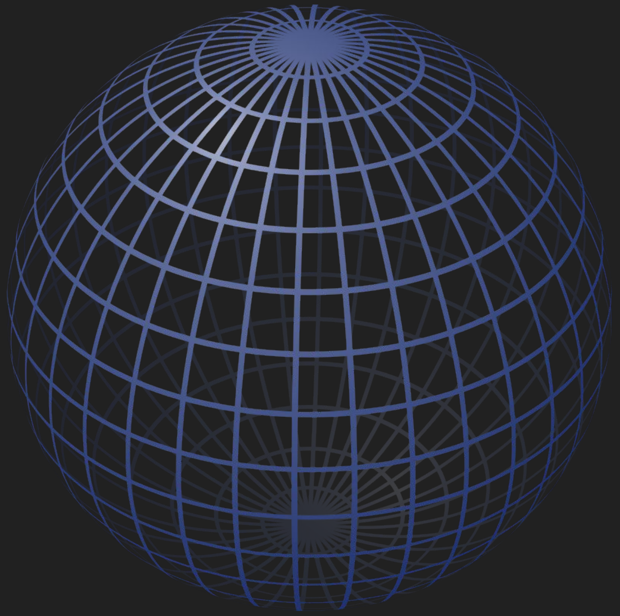
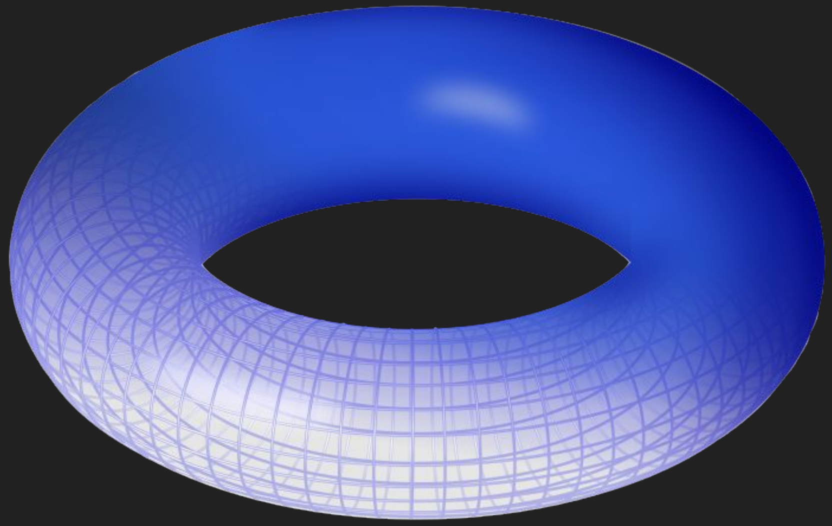
Topological Data Analysis to understand Convolutional Neural Networks

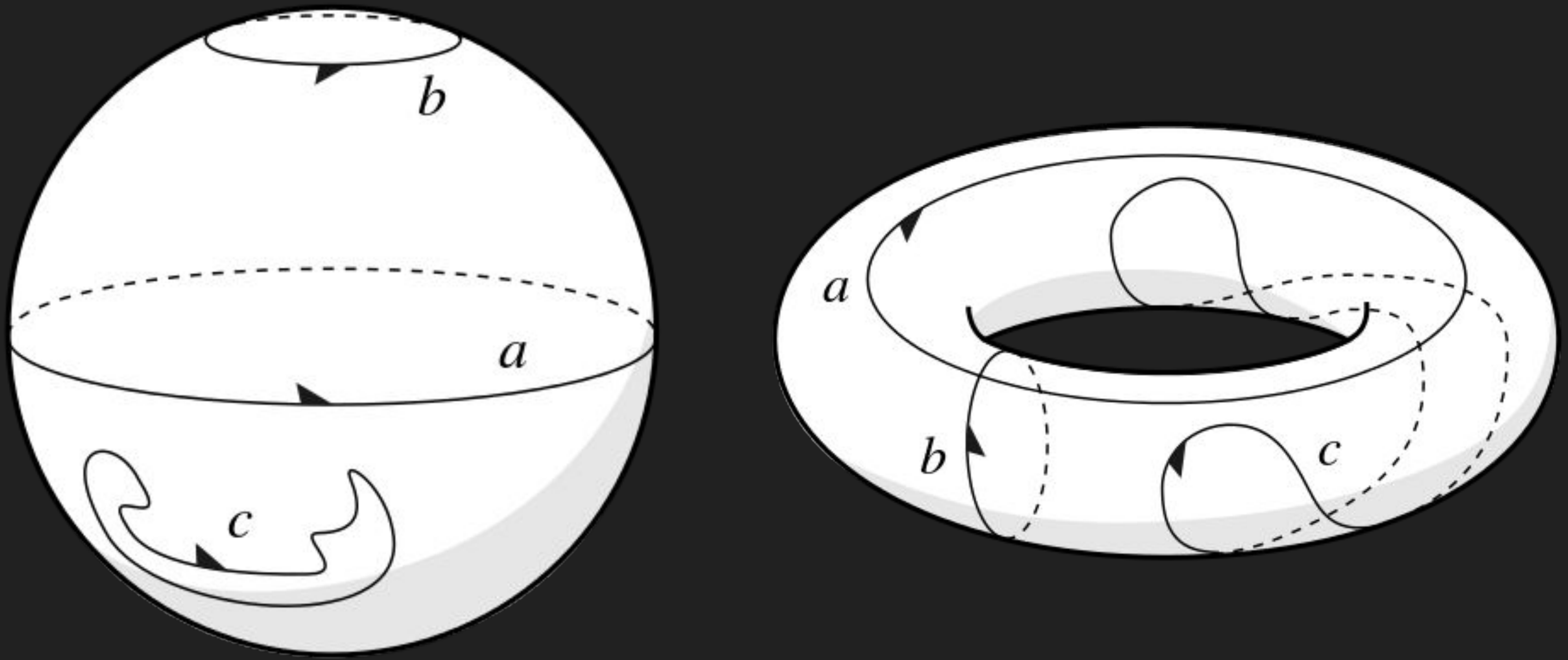
Aleksei Prokopev, SeoulAI, 2018

Shape

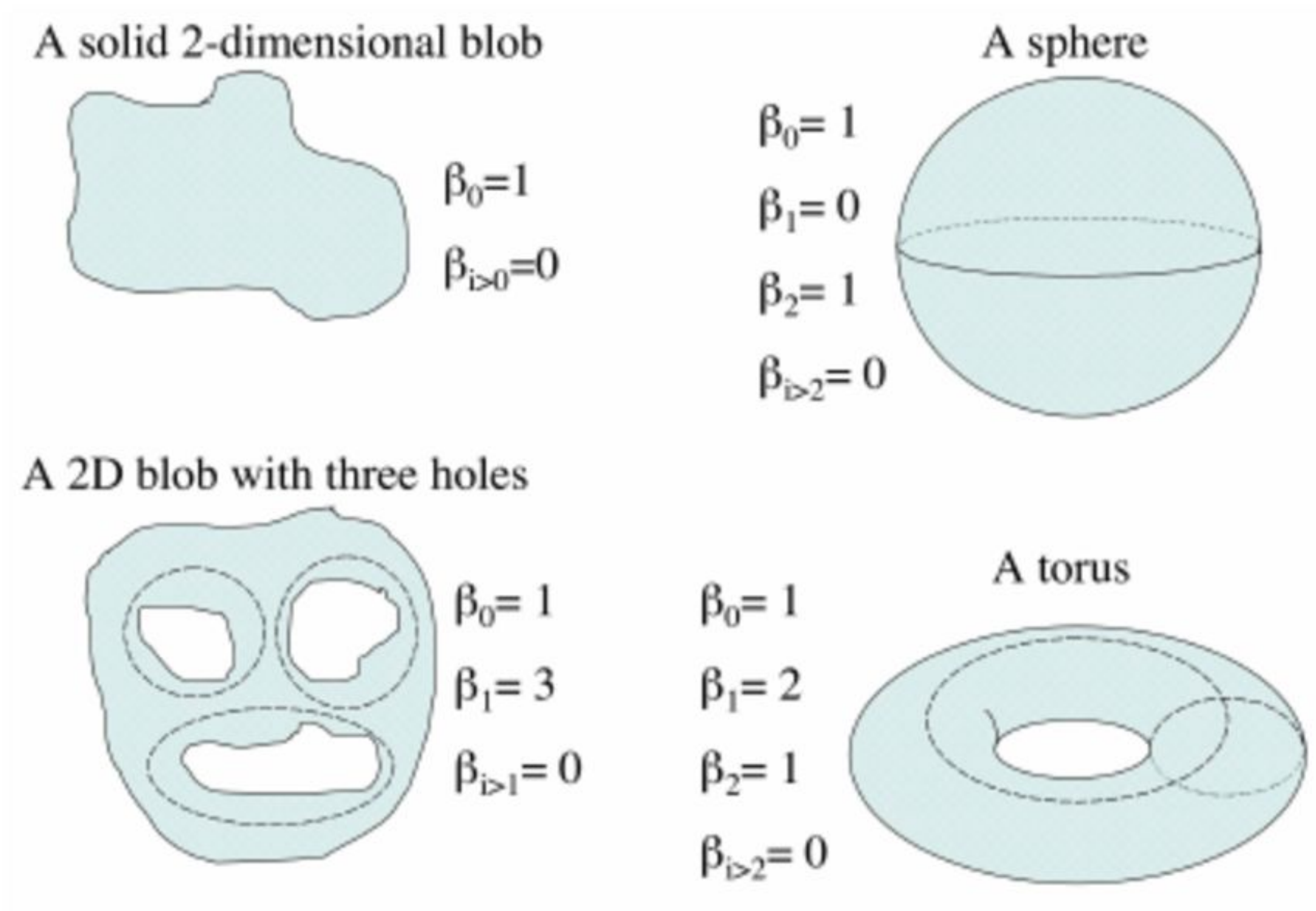


Topology

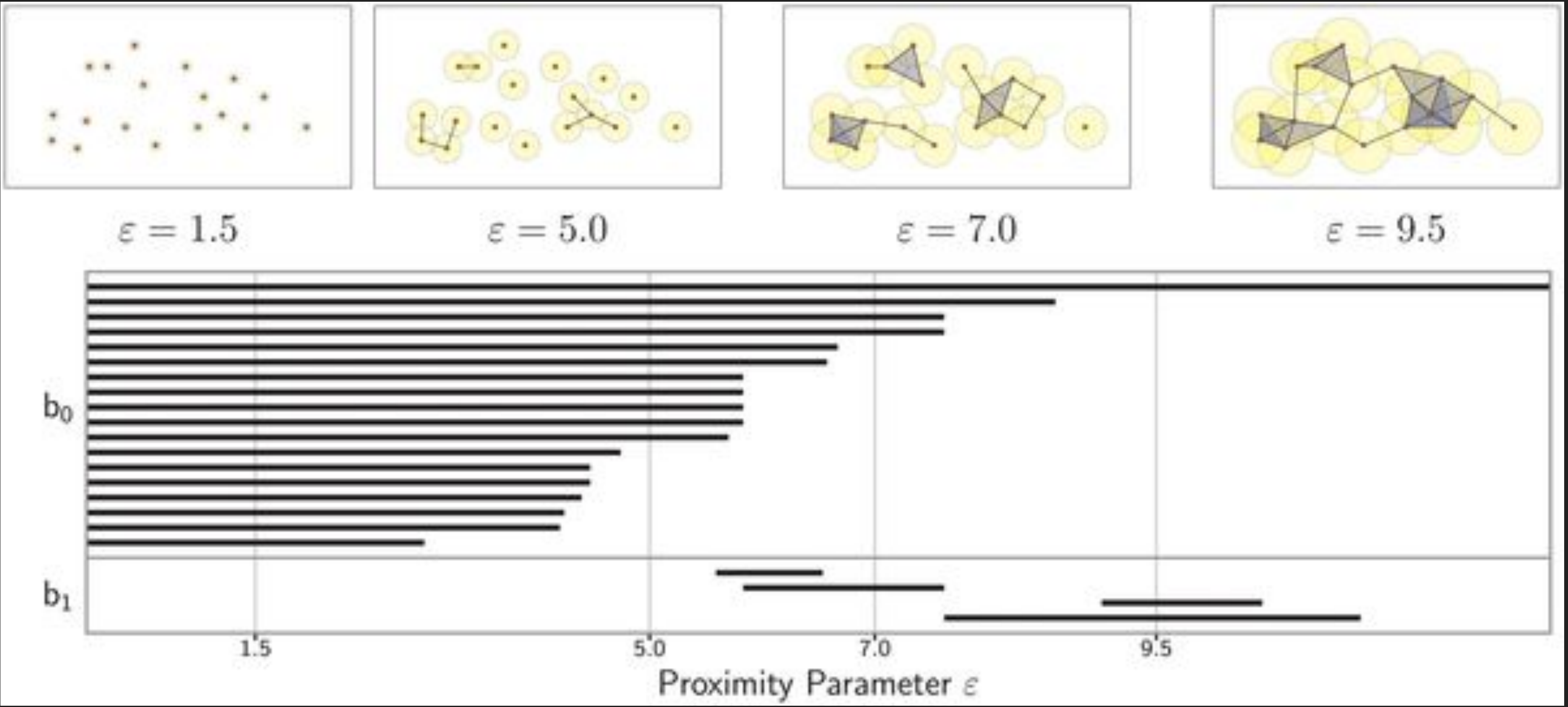





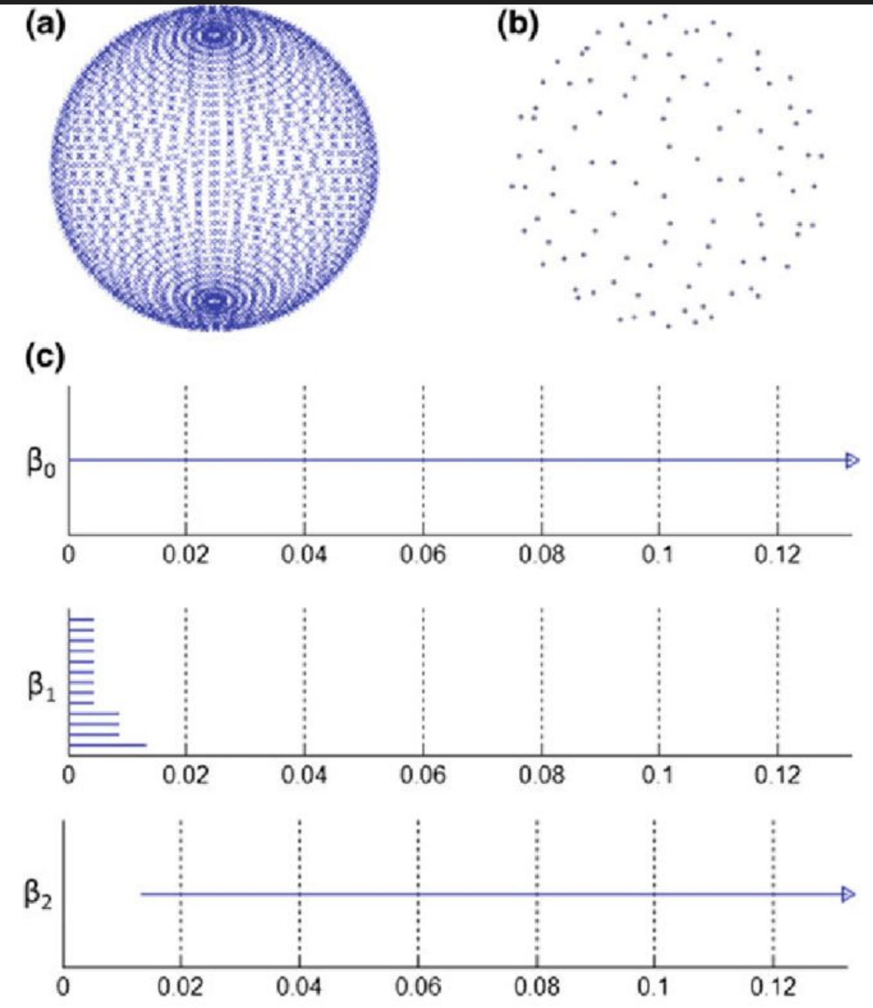
b
a
c
a
b
c


A solid 2-dimensional blob
$\beta_0=1$
$\beta_{i>0}=0$
A sphere
$\beta_0= 1$
$\beta_1= 0$
$\beta_2= 1$
$\beta_{i>2}= 0$
A 2D blob with three holes
$\beta_0= 1$
$\beta_1= 3$
$\beta_{i>1}= 0$
A torus
$\beta_0= 1$
$\beta_1= 2$
$\beta_2= 1$
$\beta_{i>2}= 0$

Topological Data Analysis

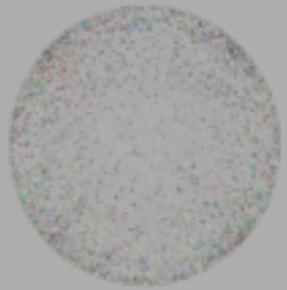


$\varepsilon = 1.5$
$\varepsilon = 5.0$
$\varepsilon = 7.0$
$\varepsilon = 9.5$
b_0
b_1
1.5
5.0
7.0
9.5
Proximity Parameter ε



(a)
(b)
(c)
β0
0
0.02
0.04
0.06
0.08
0.1
0.12
β1
0
0.02
0.04
0.06
0.08
0.1
0.12
β2
0
0.02
0.04
0.06
0.08
0.1
0.12

[redacted]

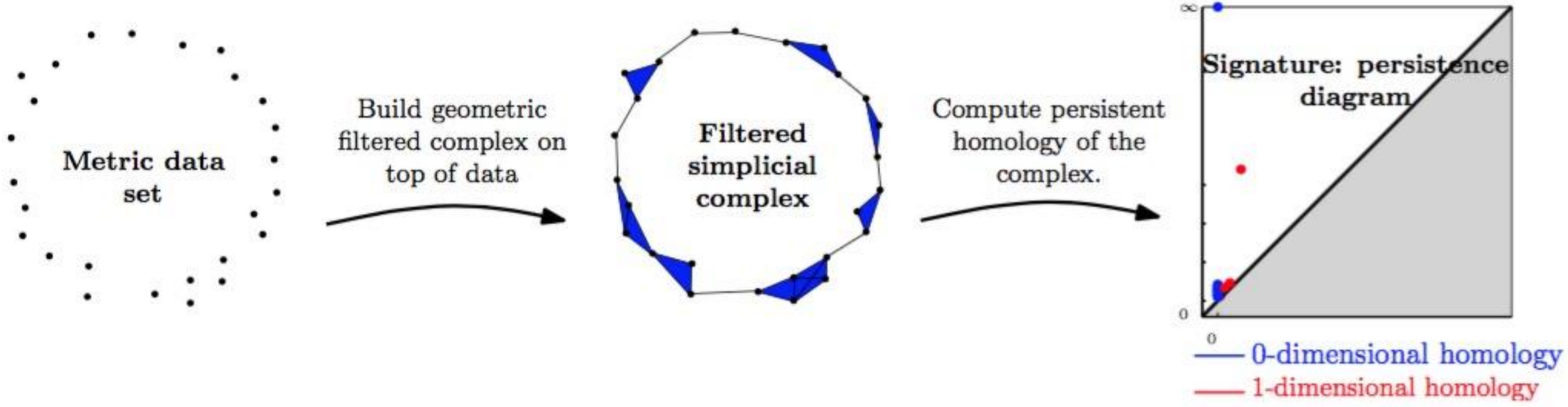


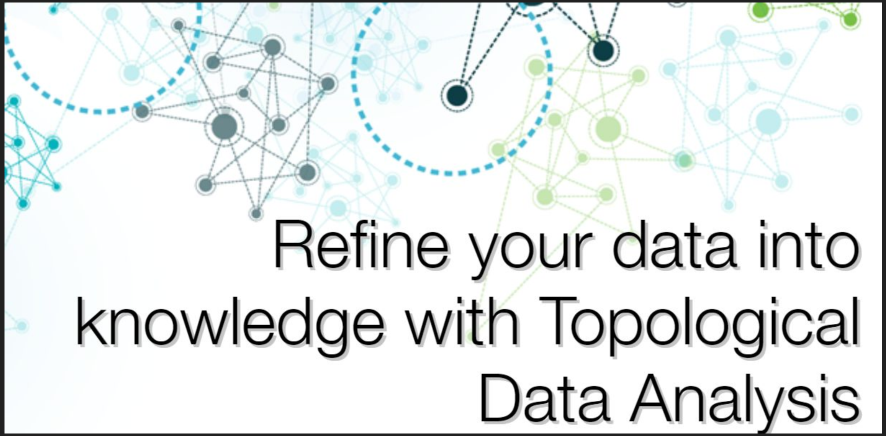
[redacted]

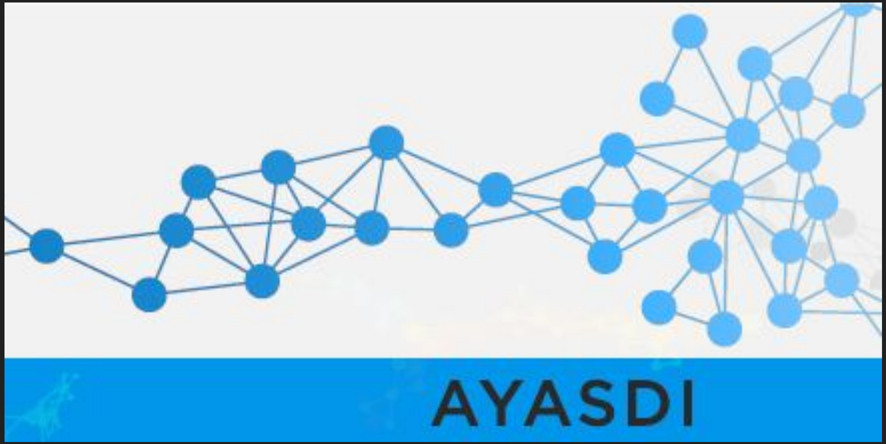
[redacted]

[redacted]

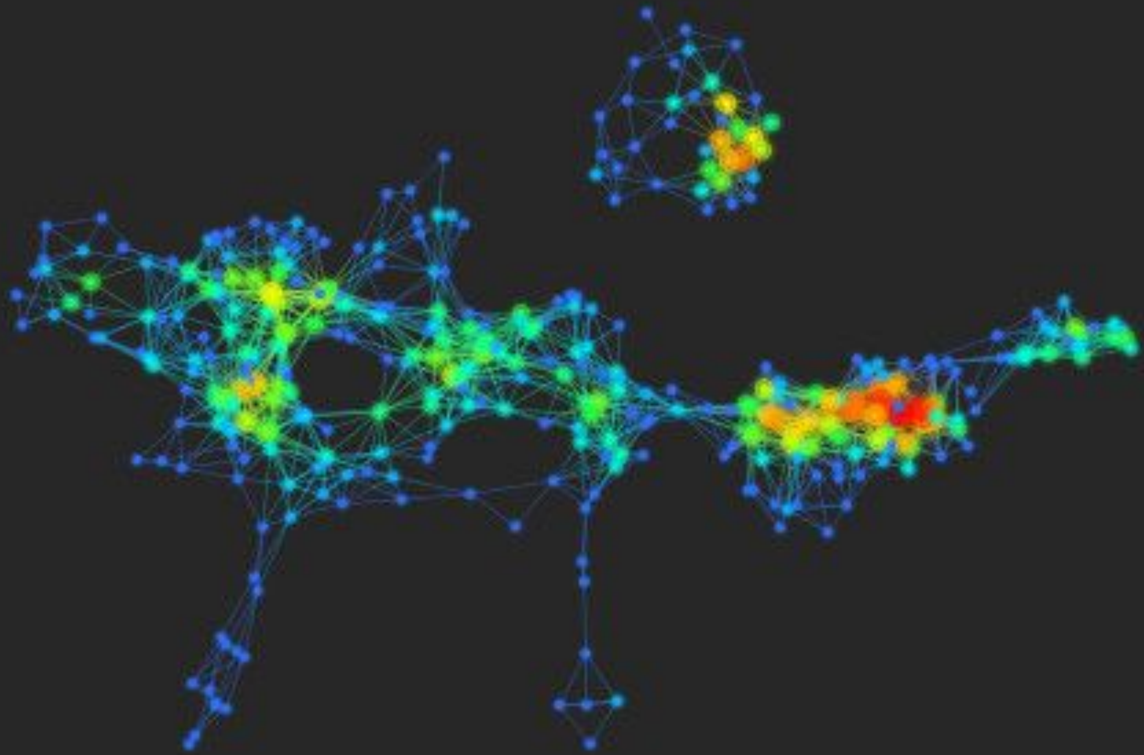
[redacted]


Metric data set
Build geometric filtered complex on top of data
Filtered simplicial complex
Compute persistent homology of the complex.
Signature: persistence diagram
∞
0
0
0-dimensional homology
1-dimensional homology

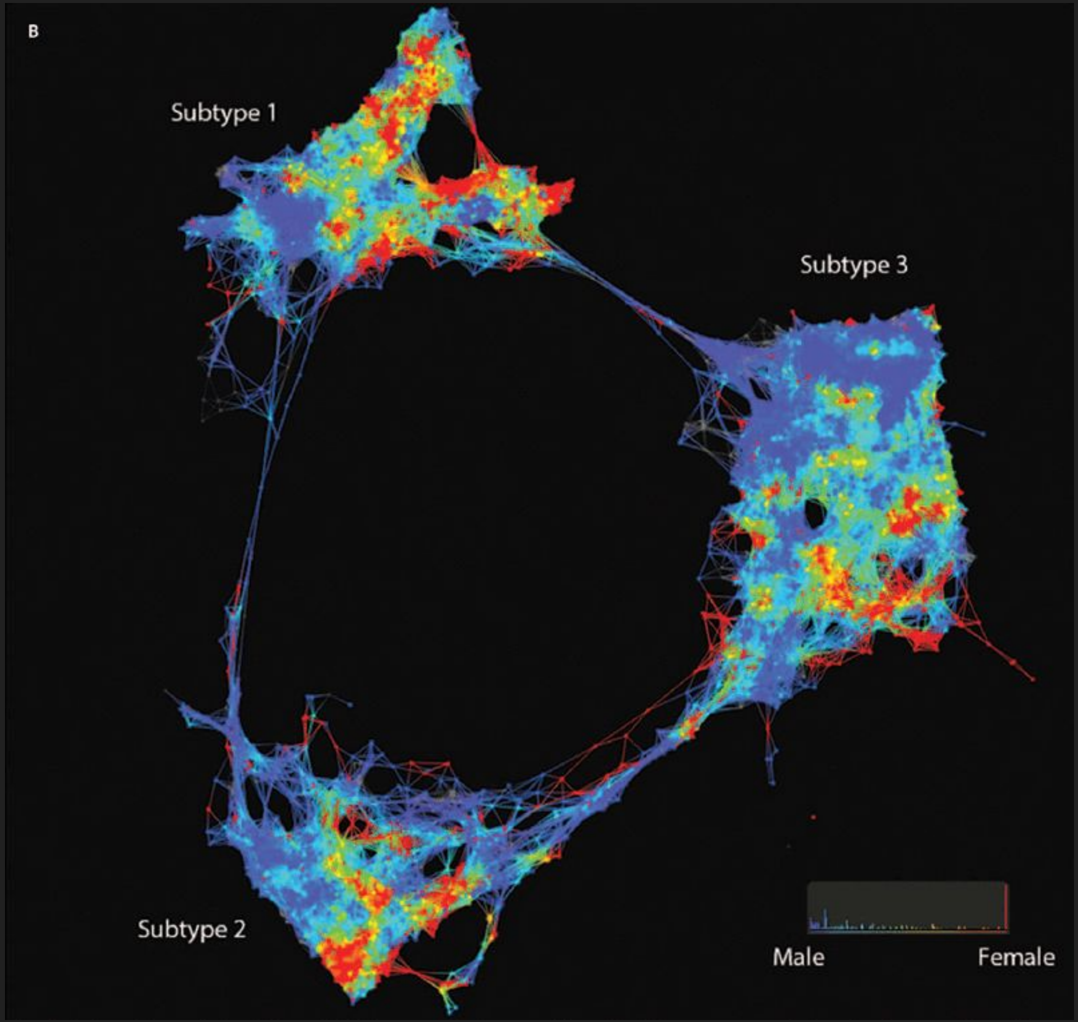

Refine your data into knowledge with Topological Data Analysis


AYASDI

Examples



AYASDI

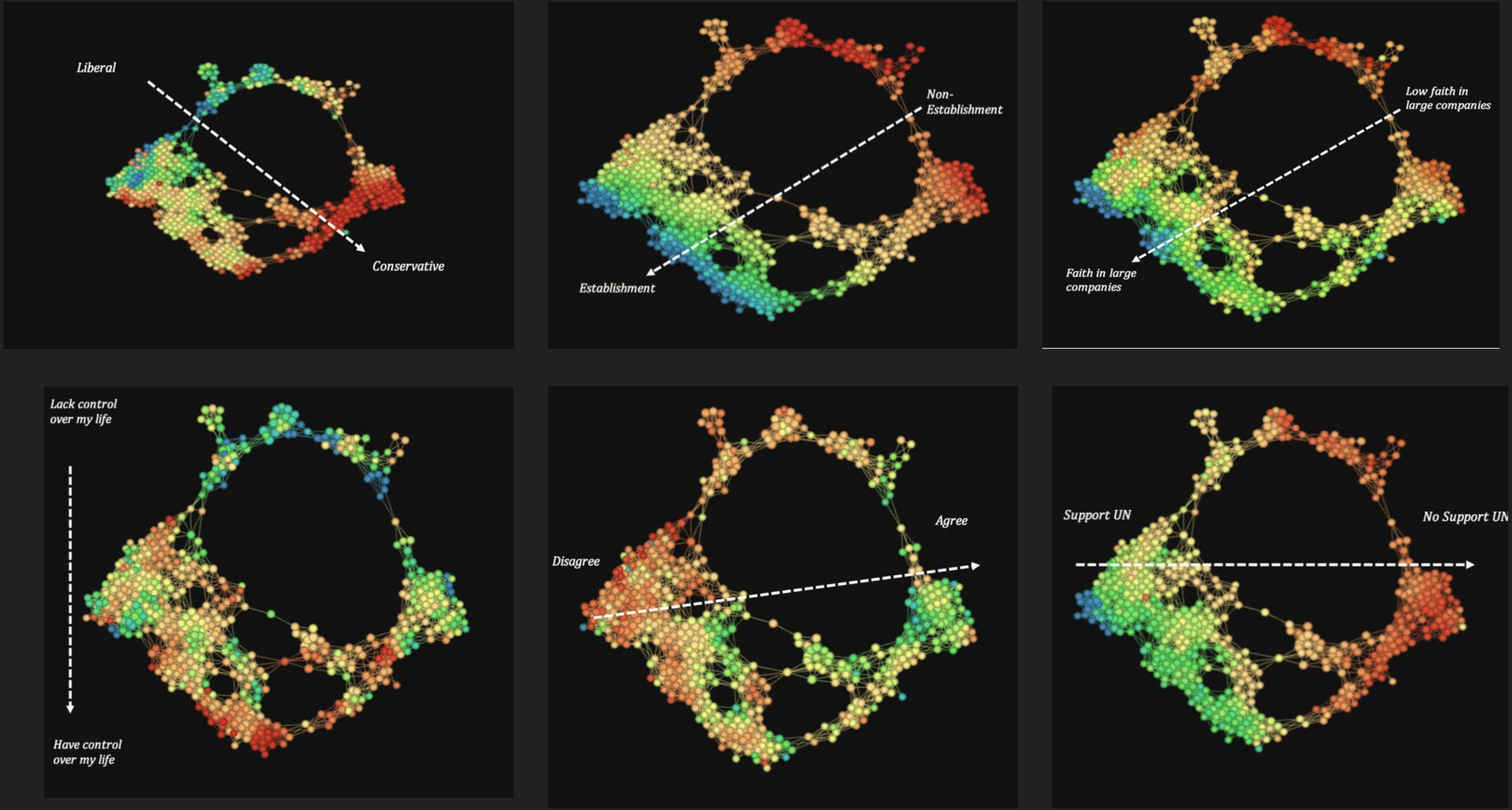

B
Subtype 1
Subtype 3
Subtype 2
Male
Female

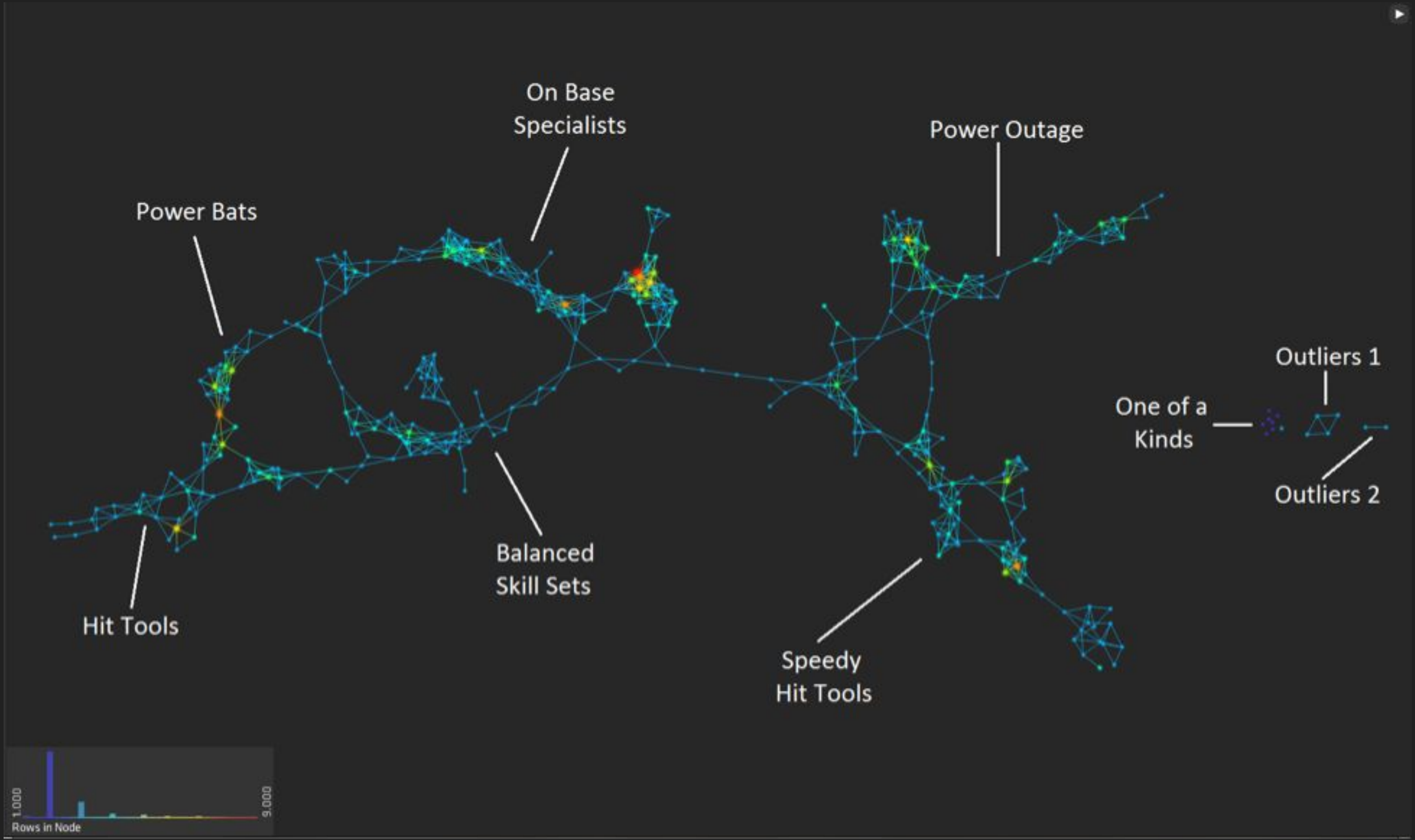
Nodes are groups of similar data points

Edges connect similar nodes

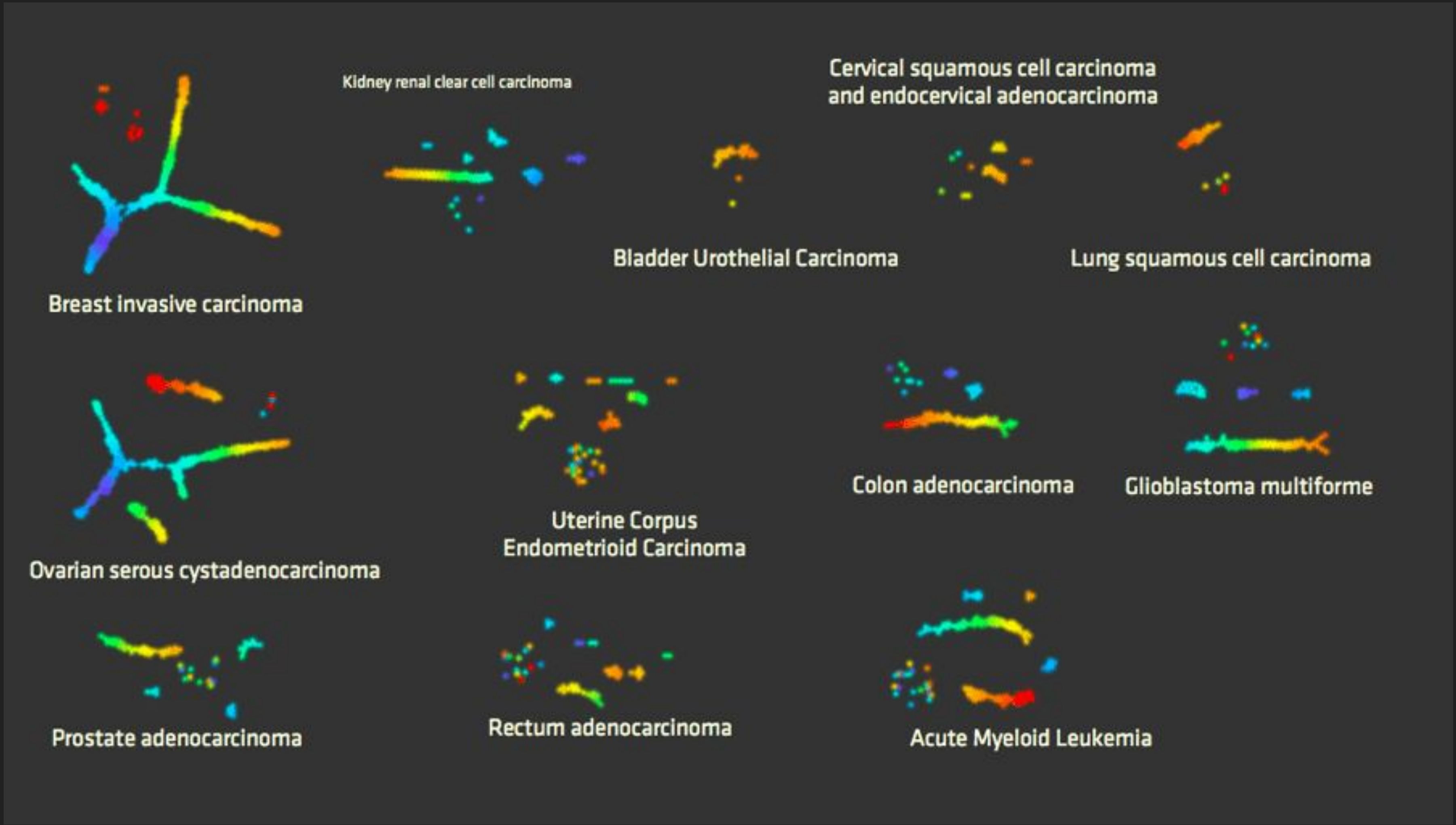
Colors let you see values of interest

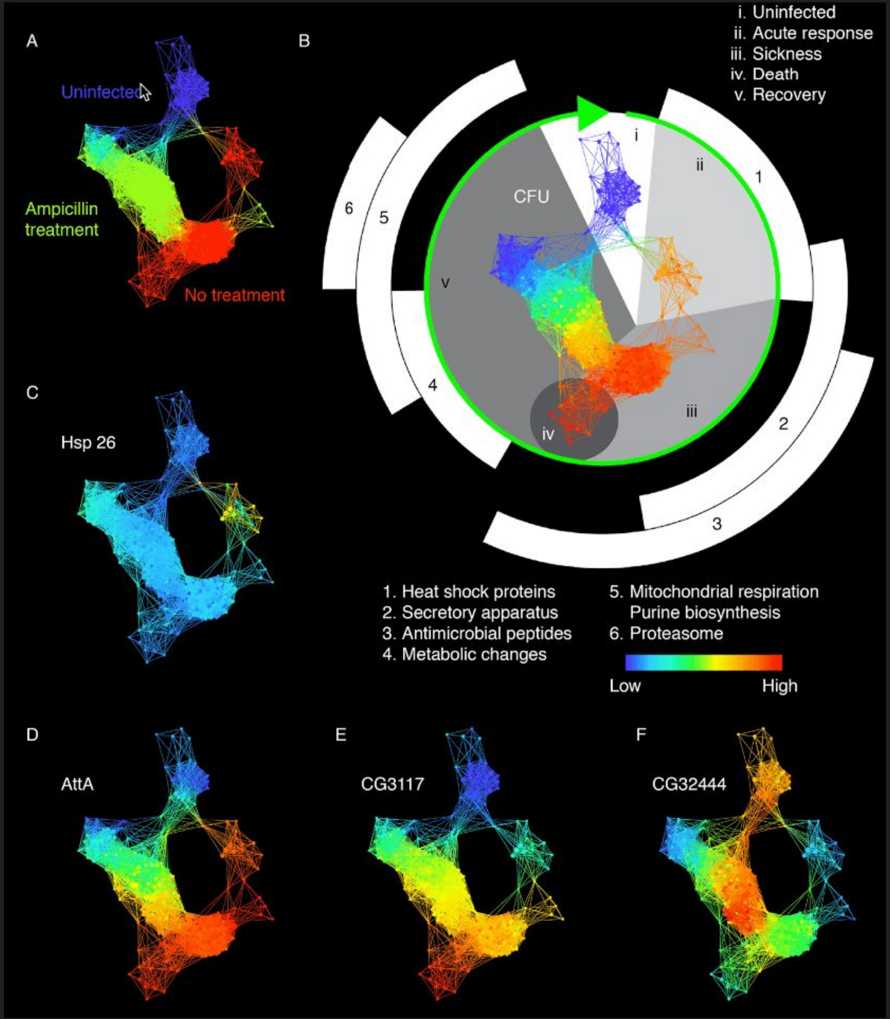
Position of a node on the screen doesn’t matter

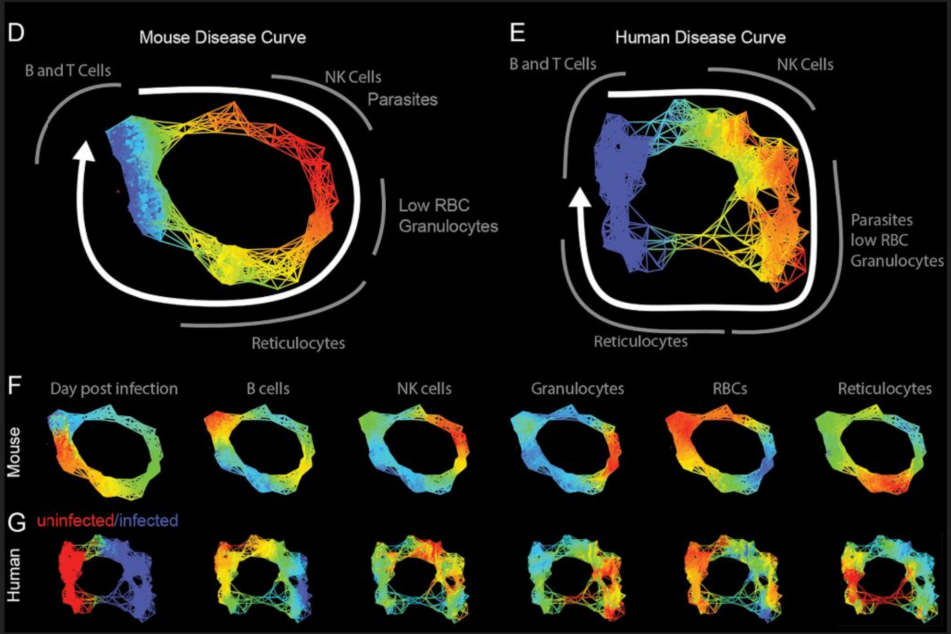

Liberal
Conservative
Non-
Establishment
Establishment
Low faith in
large companies
Faith in large
companies
Lack control
over my life
Have control
over my life
Disagree
Agree
Support UN
No Support UN

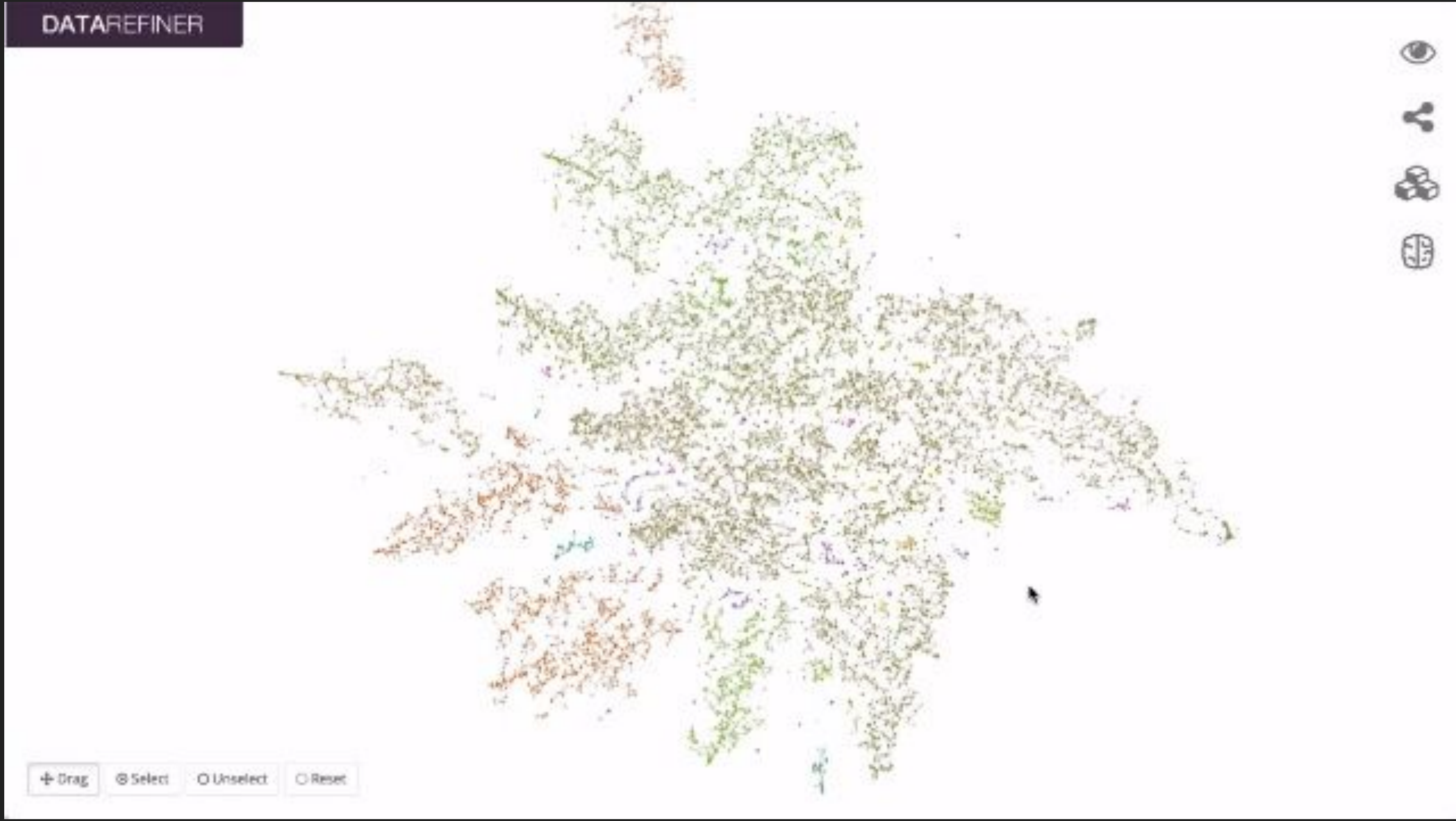


On Base Specialists
Power Outage
Power Bats
Outliers 1
One of a Kinds
Outliers 2
Balanced Skill Sets
Hit Tools
Speedy Hit Tools
1.000
9.000
Rows in Node


Breast invasive carcinoma
Kidney renal clear cell carcinoma
Bladder Urothelial Carcinoma
Cervical squamous cell carcinoma and endocervical adenocarcinoma
Lung squamous cell carcinoma
Ovarian serous cystadenocarcinoma
Uterine Corpus Endometrioid Carcinoma
Colon adenocarcinoma
Glioblastoma multiforme
Prostate adenocarcinoma
Rectum adenocarcinoma
Acute Myeloid Leukemia

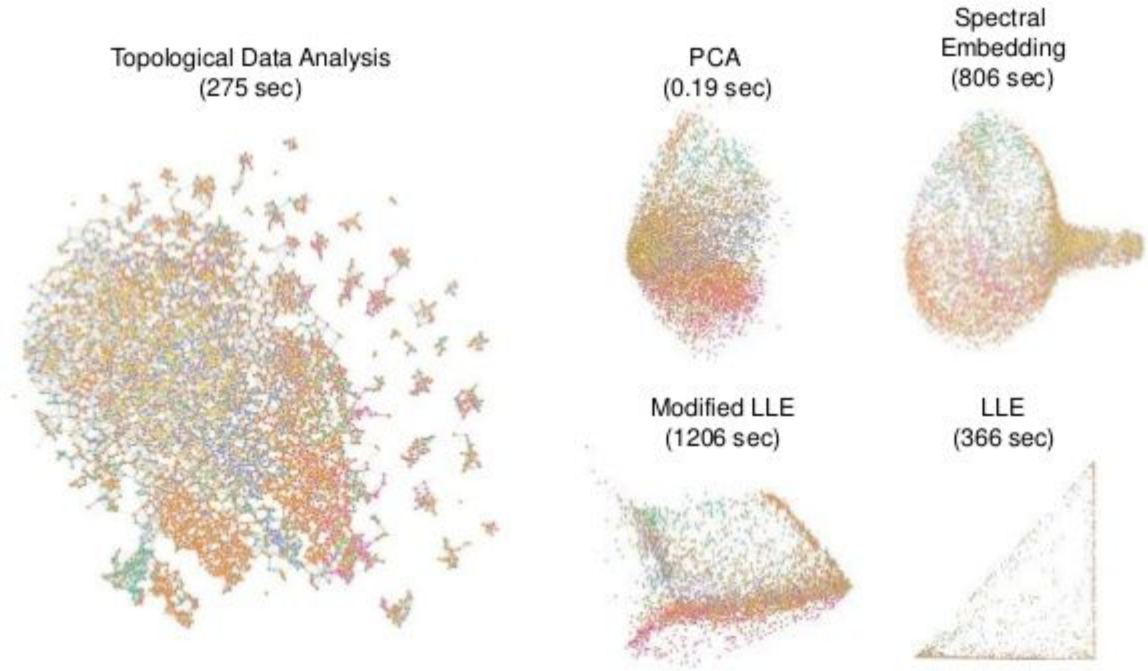

A
Uninfected
Ampicillin treatment
No treatment
B
i. Uninfected
ii. Acute response
iii. Sickness
iv. Death
v. Recovery
CFU
i
ii
iii
iv
v
1
2
3
4
5
6
1. Heat shock proteins
2. Secretory apparatus
3. Antimicrobial peptides
4. Metabolic changes
5. Mitochondrial respiration
Purine biosynthesis
6. Proteasome
Low
High
C
Hsp 26
D
AttA
E
CG3117
F
CG32444


D
Mouse Disease Curve
B and T Cells
NK Cells
Parasites
Low RBC
Granulocytes
Reticulocytes
E
Human Disease Curve
B and T Cells
NK Cells
Parasites
low RBC
Granulocytes
Reticulocytes
F
Day post infection
B cells
NK cells
Granulocytes
RBCs
Reticulocytes
Mouse
G
uninfected/infected
Human


DATAREFINER
Drag
Select
Unselect
Reset

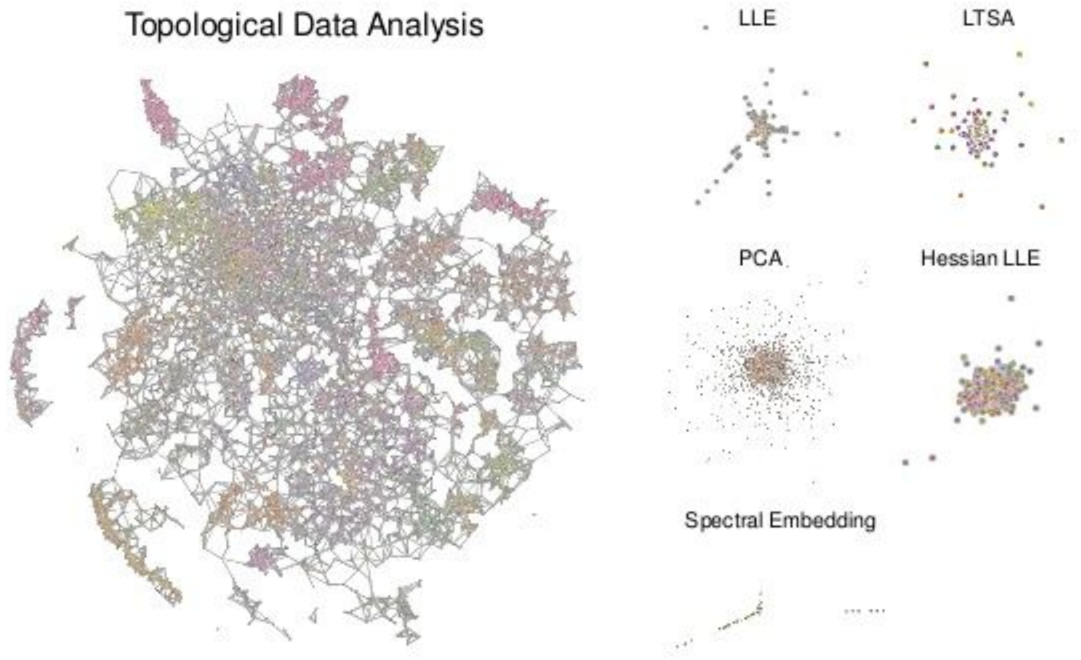
Case study: Yelp Dataset Challenge

Result comparison: TDA with other techniques

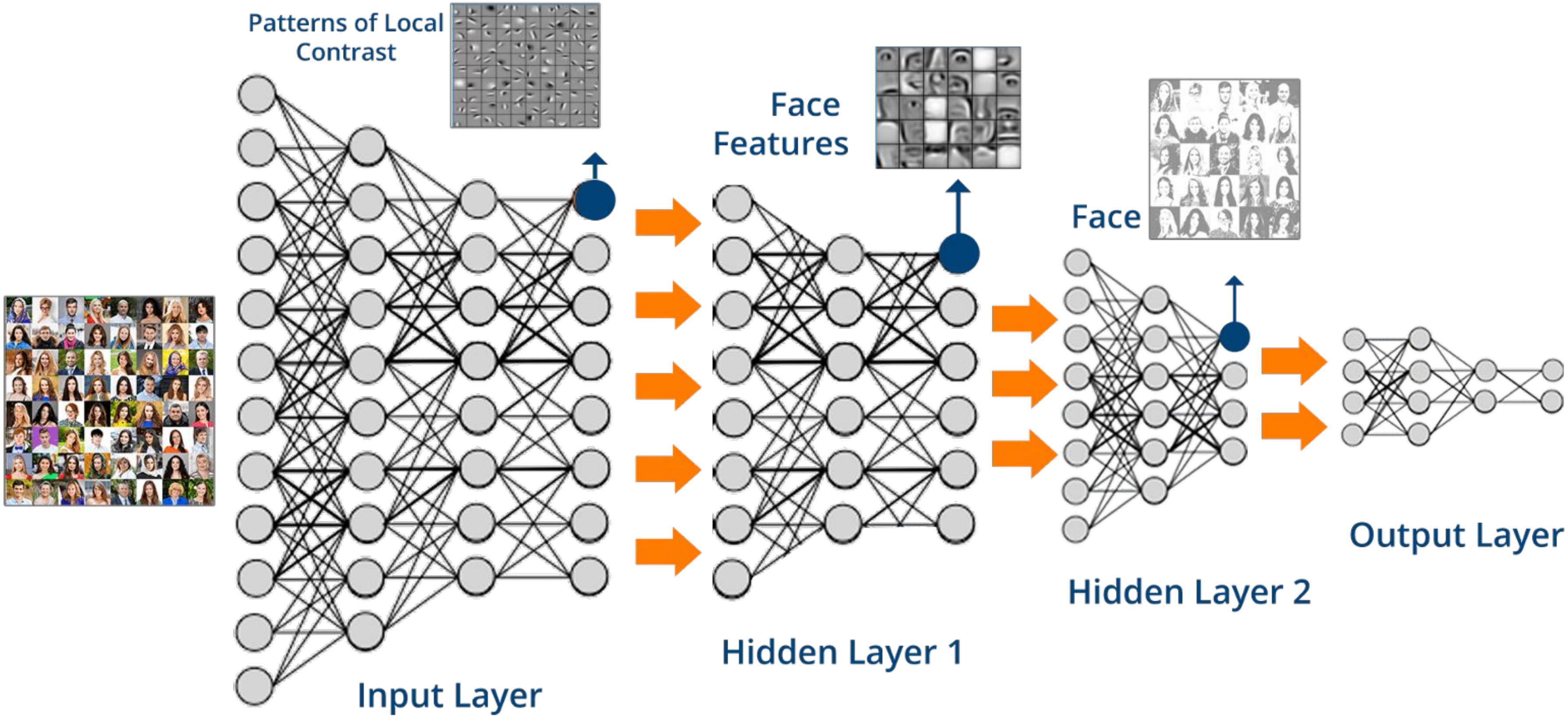


Case study: Netflix competition

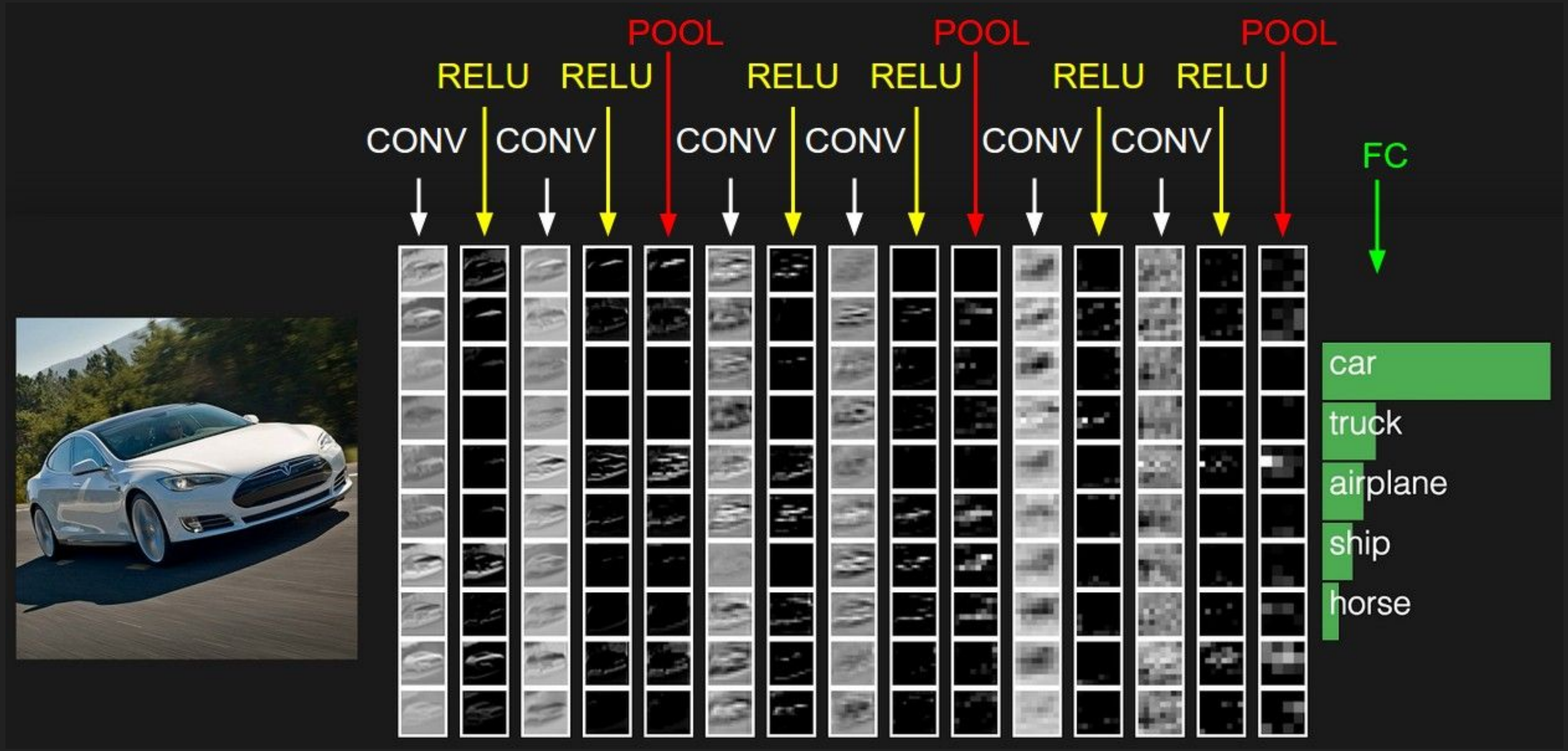
Result comparison: TDA with other techniques



Convolutional Neural Networks



Patterns of Local Contrast
Face Features
Face
Input Layer
Hidden Layer 1
Hidden Layer 2
Output Layer

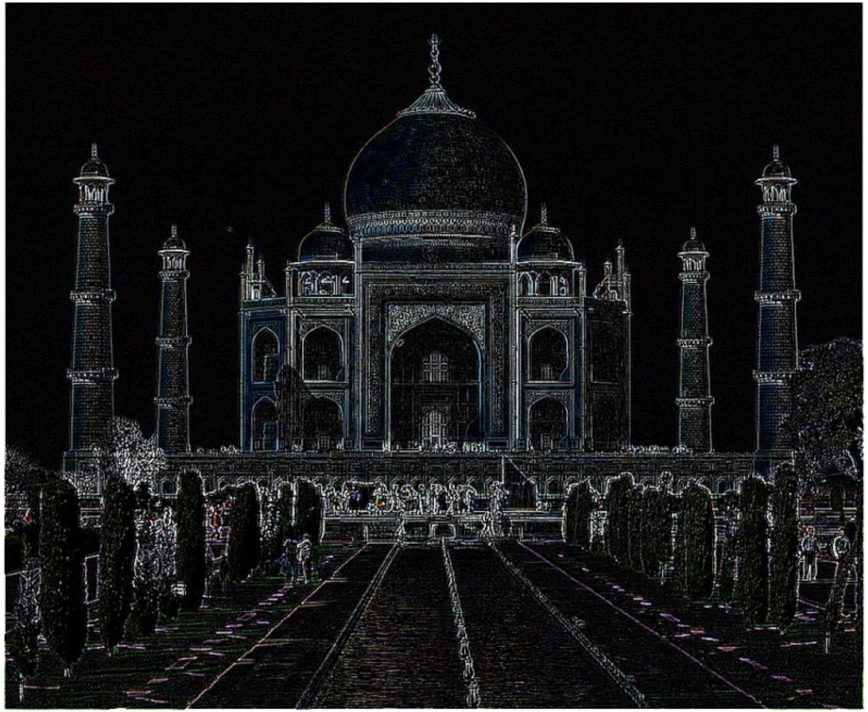


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

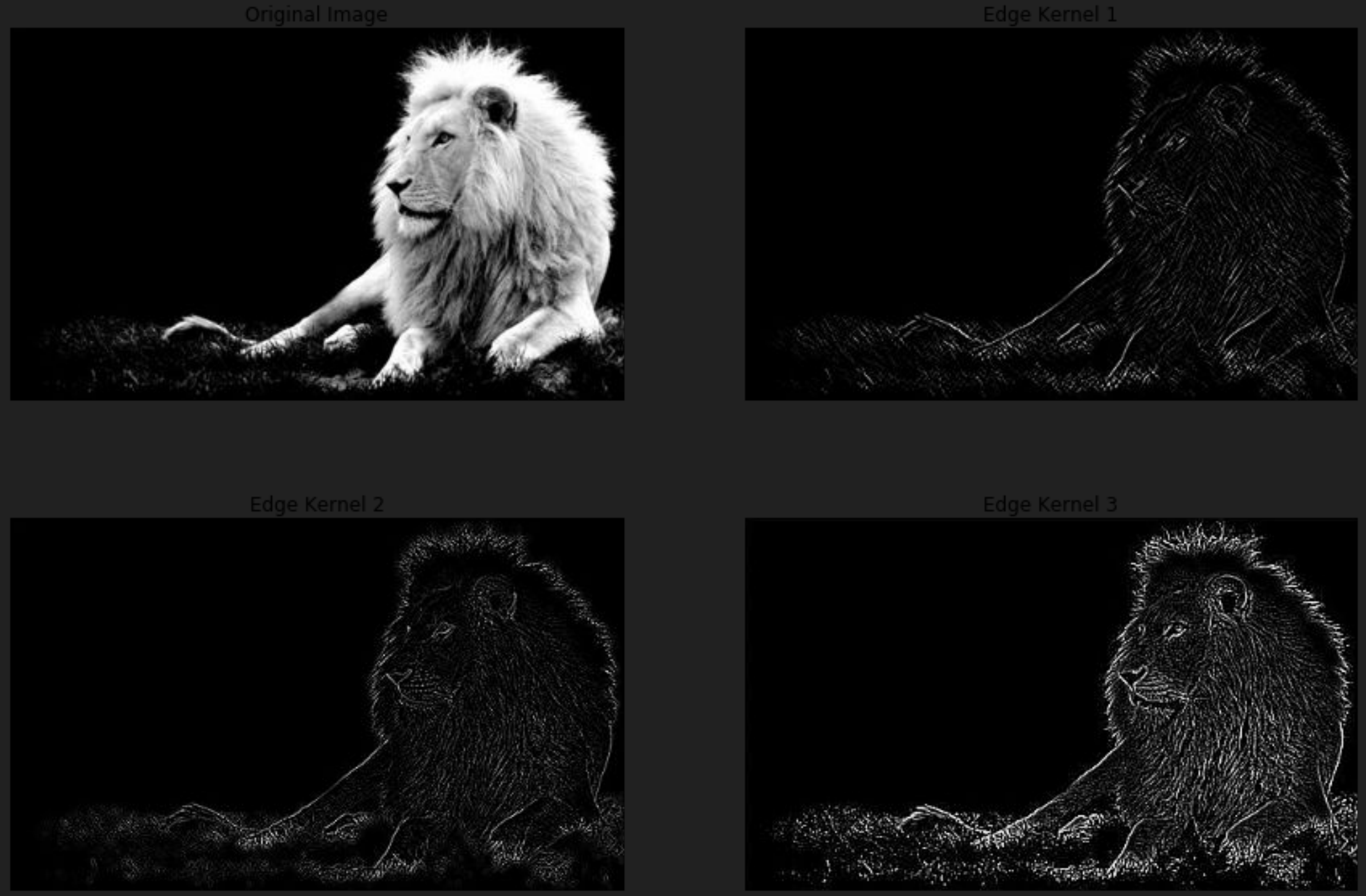
Convolution

$$
\begin{array}{|c|c|c|c|c|}
\hline
35 & 40 & 41 & 45 & 50 \\ \hline
40 & 40 & 42 & 46 & 52 \\ \hline
42 & 46 & 50 & 55 & 55 \\ \hline
48 & 52 & 56 & 58 & 60 \\ \hline
56 & 60 & 65 & 70 & 75 \\ \hline
\end{array}
\times
\begin{array}{|c|c|c|c|c|}
\hline
 & & & & \\ \hline
 & 0 & 1 & 0 & \\ \hline
 & 0 & 0 & 0 & \\ \hline
 & 0 & 0 & 0 & \\ \hline
 & & & & \\ \hline
\end{array}
=
\begin{array}{|c|c|c|c|c|}
\hline
 & & & & \\ \hline
 & & & & \\ \hline
 & & 42 & & \\ \hline
 & & & & \\ \hline
 & & & & \\ \hline
\end{array}
$$

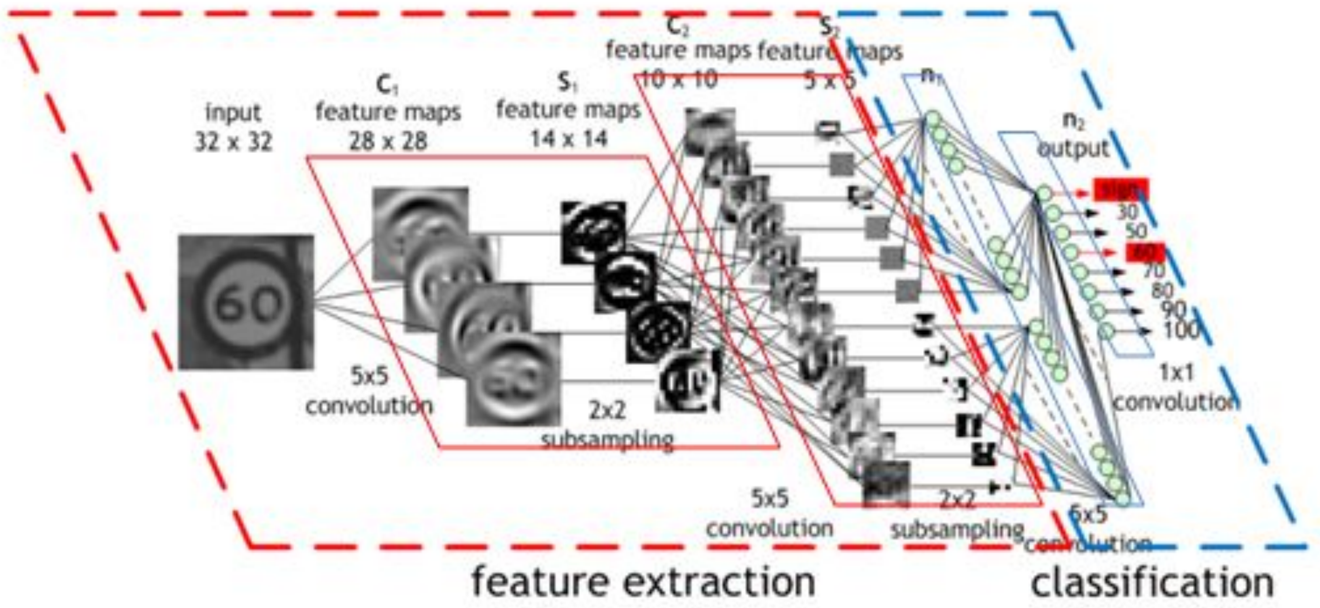
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	-2	-2	-2	0	0
0	0	-2	16	-2	0	0
0	0	-2	-2	-2	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

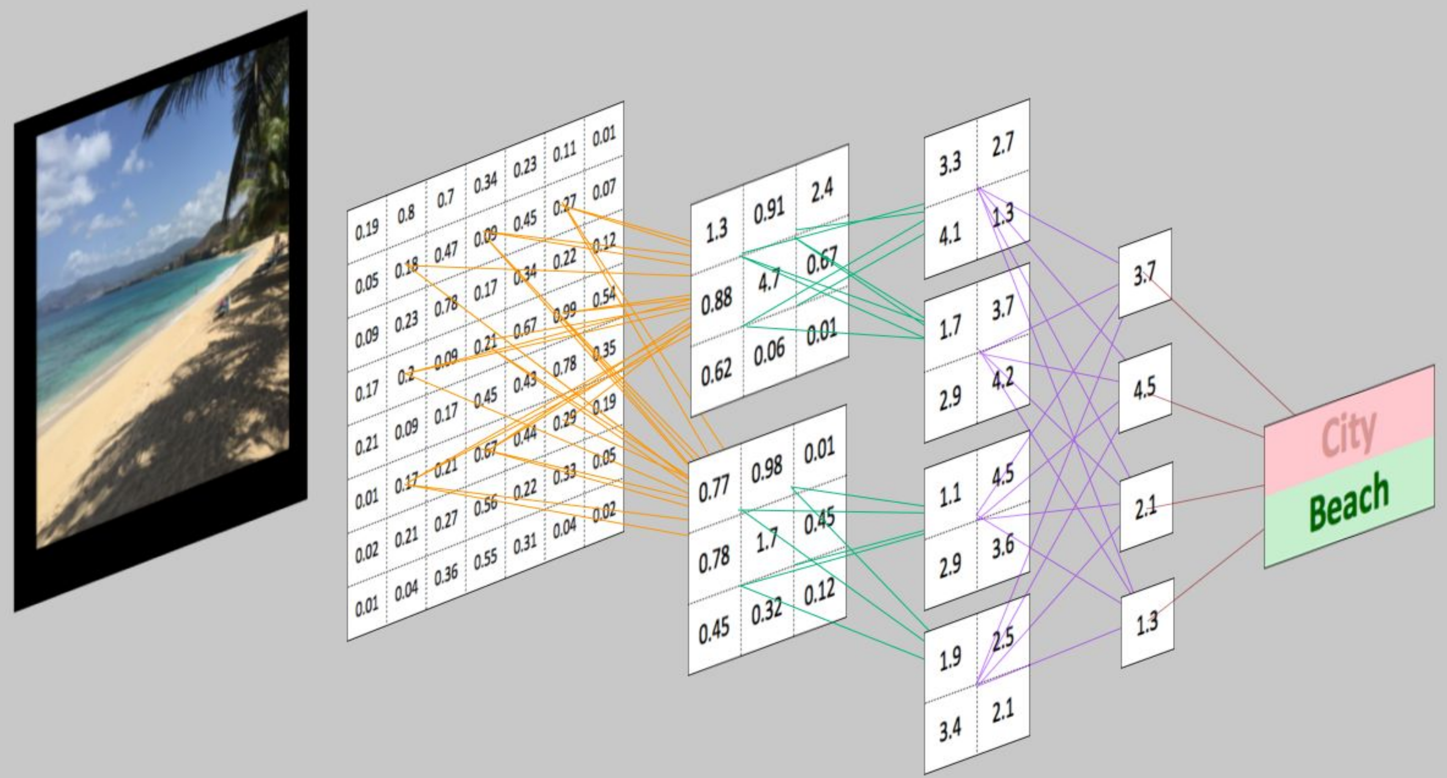


Original	Gaussian Blur	Sharpen	Edge Detection
$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$
			

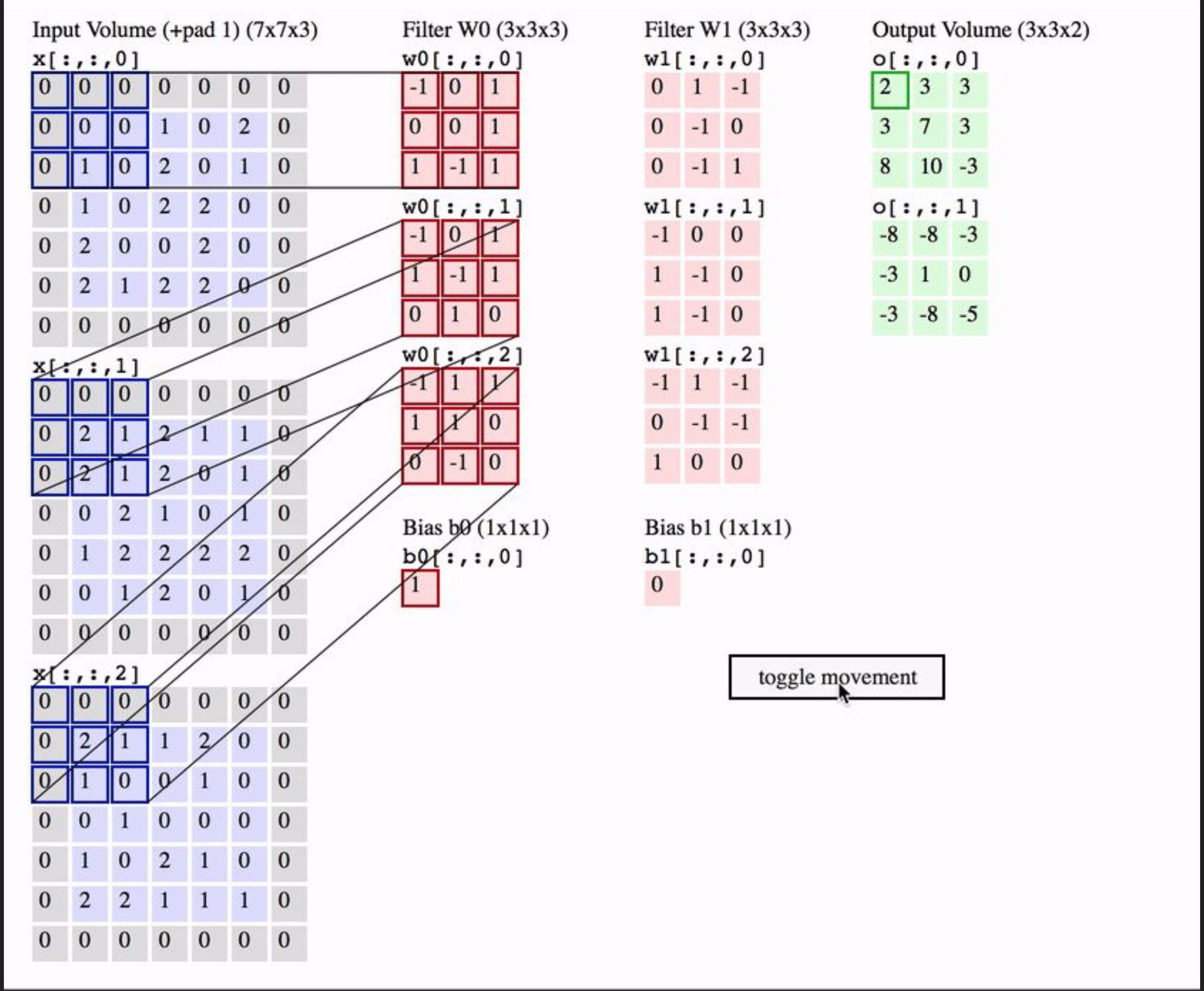

Original Image
Edge Kernel 1
Edge Kernel 2
Edge Kernel 3

Weights

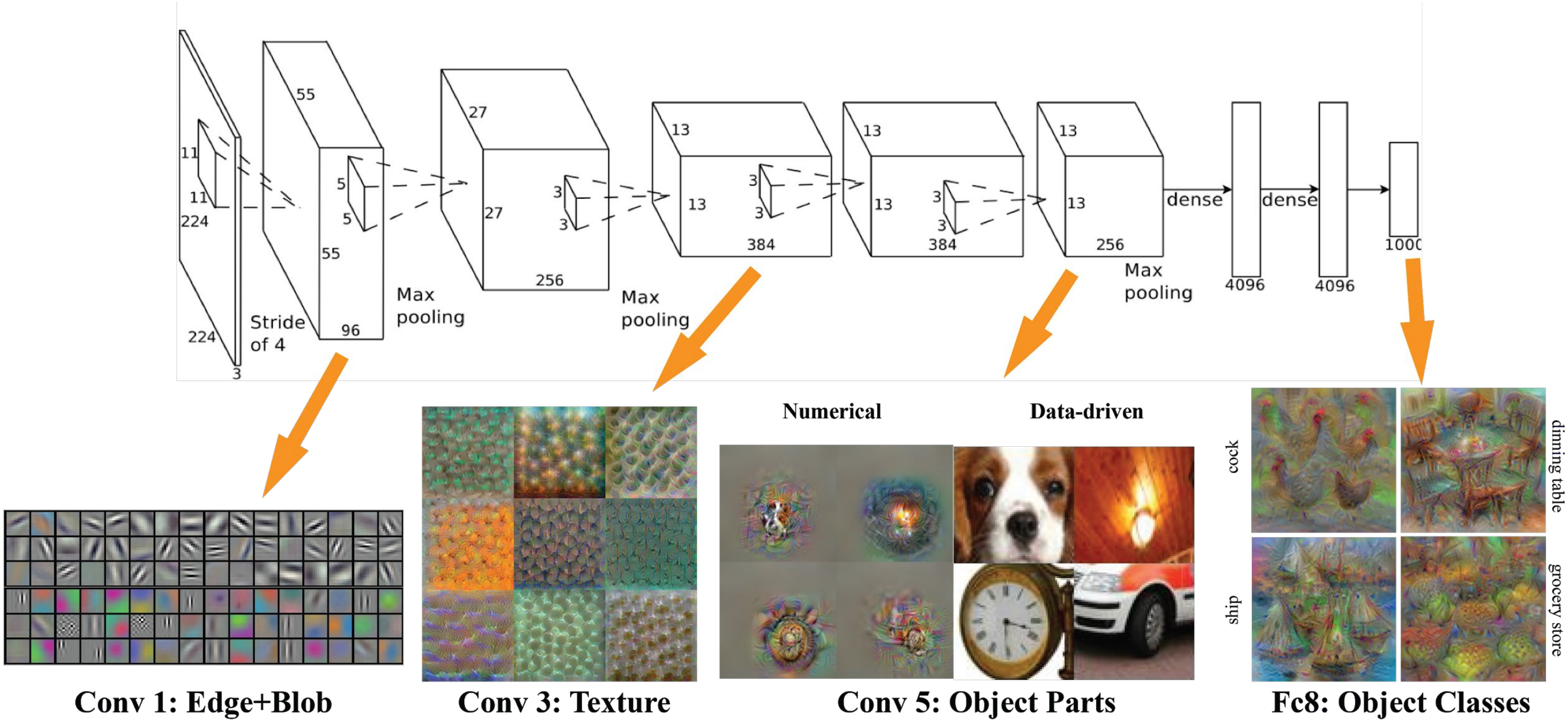

input
32 x 32
C_1
feature maps
28 x 28
S_1
feature maps
14 x 14
C_2
feature maps
10 x 10
S_2
feature maps
5 x 5
n_1
n_2
output
30
50
60
70
80
90
100
5x5
convolution
2x2
subsampling
5x5
convolution
2x2
subsampling
6x5
convolution
1x1
convolution
feature extraction
classification

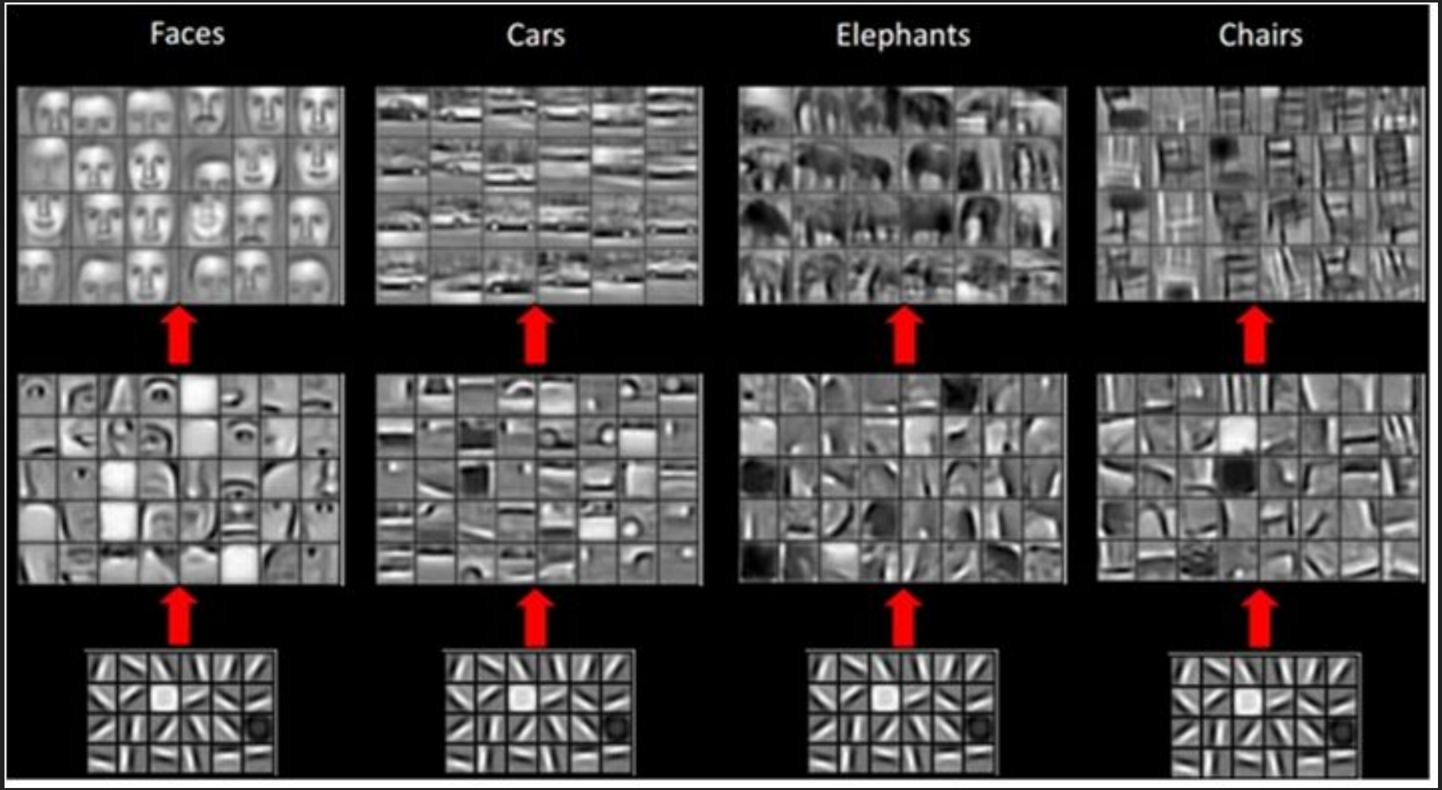


0.19 0.8 0.7 0.34 0.23 0.11 0.01
0.05 0.18 0.47 0.09 0.45 0.27 0.07
0.09 0.23 0.78 0.17 0.34 0.22 0.12
0.17 0.2 0.09 0.21 0.67 0.99 0.54
0.21 0.09 0.17 0.45 0.43 0.78 0.35
0.01 0.17 0.21 0.67 0.44 0.29 0.19
0.02 0.21 0.27 0.56 0.22 0.33 0.05
0.01 0.04 0.36 0.55 0.31 0.04 0.02
1.3 0.91 2.4
0.88 4.7 0.67
0.62 0.06 0.01
0.77 0.98 0.01
0.78 1.7 0.45
0.45 0.32 0.12
3.3 2.7
4.1 1.3
1.7 3.7
2.9 4.2
1.1 4.5
2.9 3.6
1.9 2.5
3.4 2.1
3.7
4.5
2.1
1.3
City
Beach

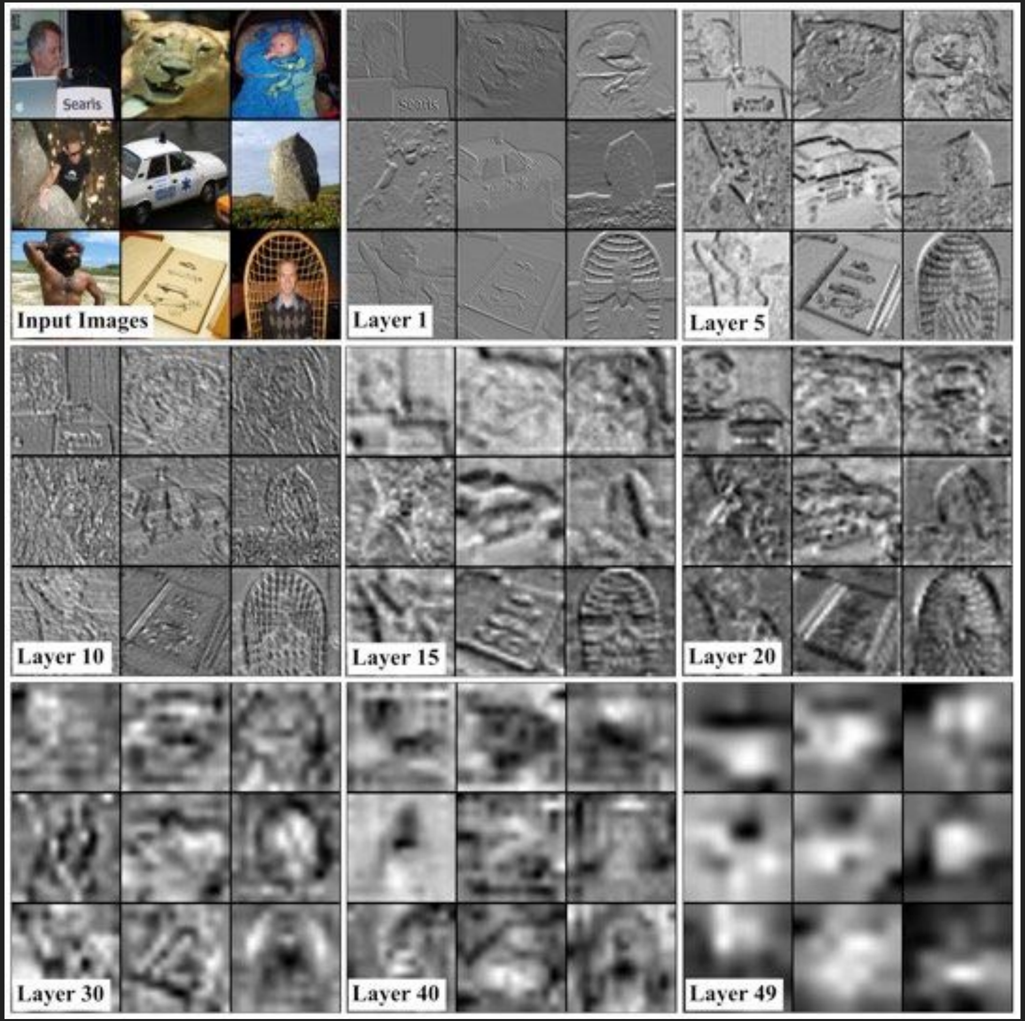


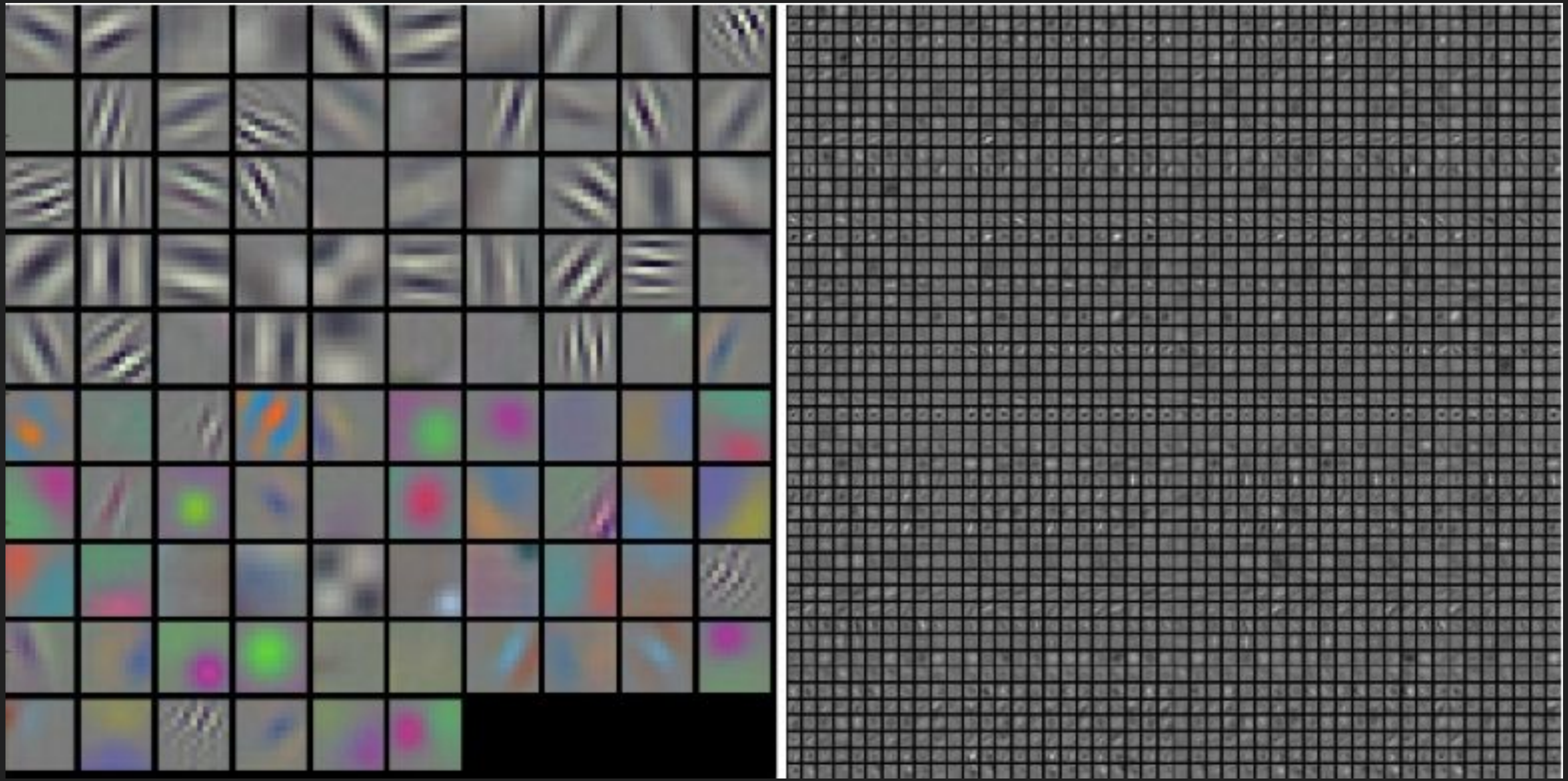
Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 0 0 1 0 2 0
0 1 0 2 0 1 0
0 1 0 2 2 0 0
0 2 0 0 2 0 0
0 2 1 2 2 0 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 2 1 2 1 1 0
0 2 1 2 0 1 0
0 0 2 1 0 1 0
0 1 2 2 2 2 0
0 0 1 2 0 1 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 2 1 1 2 0 0
0 1 0 0 1 0 0
0 0 1 0 0 0 0
0 1 0 2 1 0 0
0 2 2 1 1 1 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
-1 0 1
0 0 1
1 -1 1
w0[:,:,1]
-1 0 1
1 -1 1
0 1 0
w0[:,:,2]
-1 1 1
1 1 0
0 -1 0
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
0 1 -1
0 -1 0
0 -1 1
w1[:,:,1]
-1 0 0
1 -1 0
1 -1 0
w1[:,:,2]
-1 1 -1
0 -1 -1
1 0 0
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
2 3 3
3 7 3
8 10 -3
o[:,:,1]
-8 -8 -3
-3 1 0
-3 -8 -5
toggle movement

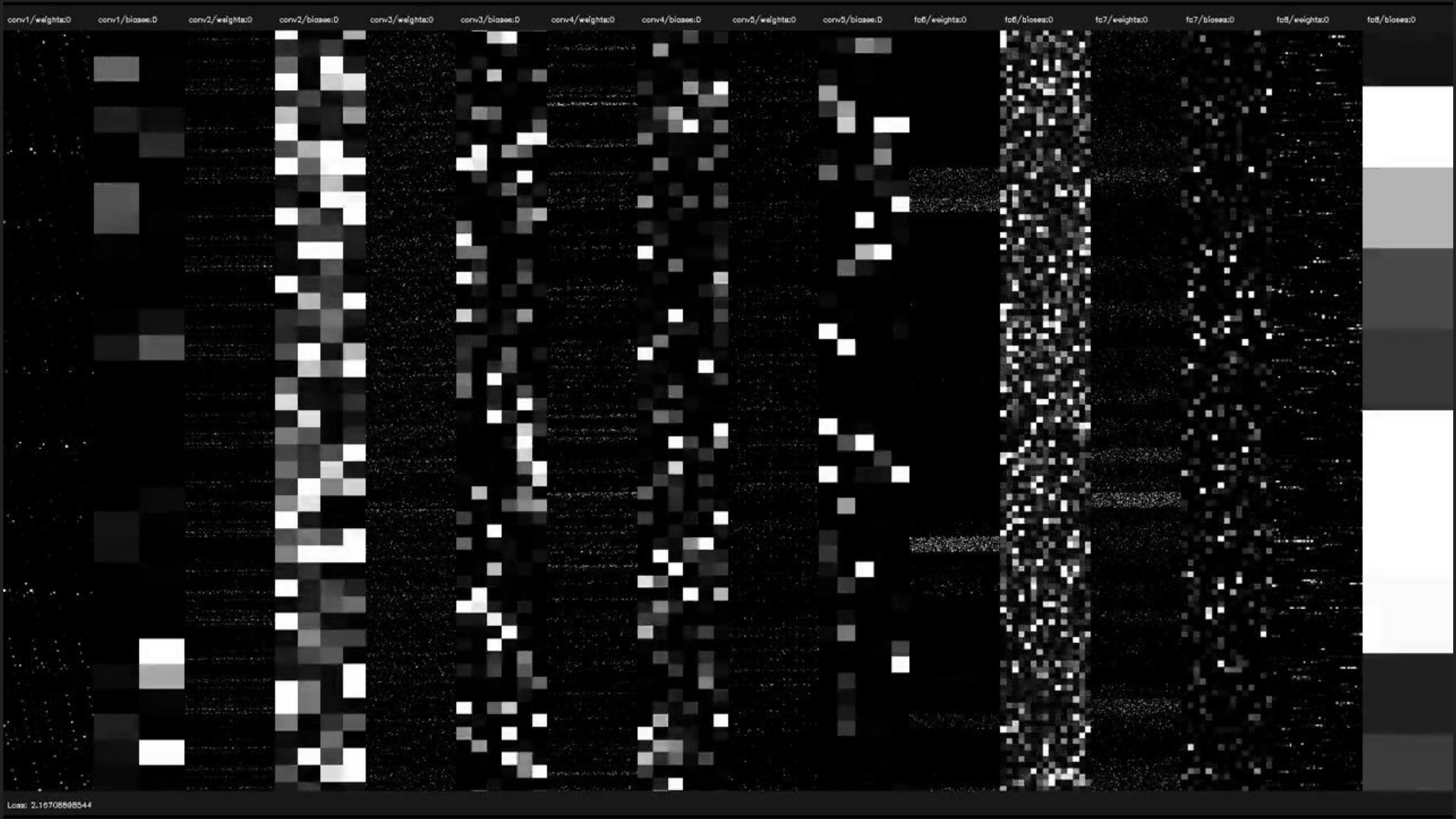

224
224
3
11
11
Stride of 4
55
55
96
5
5
Max pooling
27
27
256
3
3
Max pooling
13
13
384
13
13
384
13
13
256
Max pooling
dense
dense
4096
4096
1000
Numerical
Data-driven
cock
dinning table
ship
grocery store
Conv 1: Edge+Blob
Conv 3: Texture
Conv 5: Object Parts
Fc8: Object Classes


Faces
Cars
Elephants
Chairs

Problems


Input Images
Searis
Layer 1
Layer 5
Layer 10
Layer 15
Layer 20
Layer 30
Layer 40
Layer 49




conv1/weights:0
conv1/biases:0
conv2/weights:0
conv2/biases:0
conv3/weights:0
conv3/biases:0
conv4/weights:0
conv4/biases:0
conv5/weights:0
conv5/biases:0
fc6/weights:0
fc6/biases:0
fc7/weights:0
fc7/biases:0
fc8/weights:0
fc8/biases:0
Loss: 2.16708698544

TDA for CNN

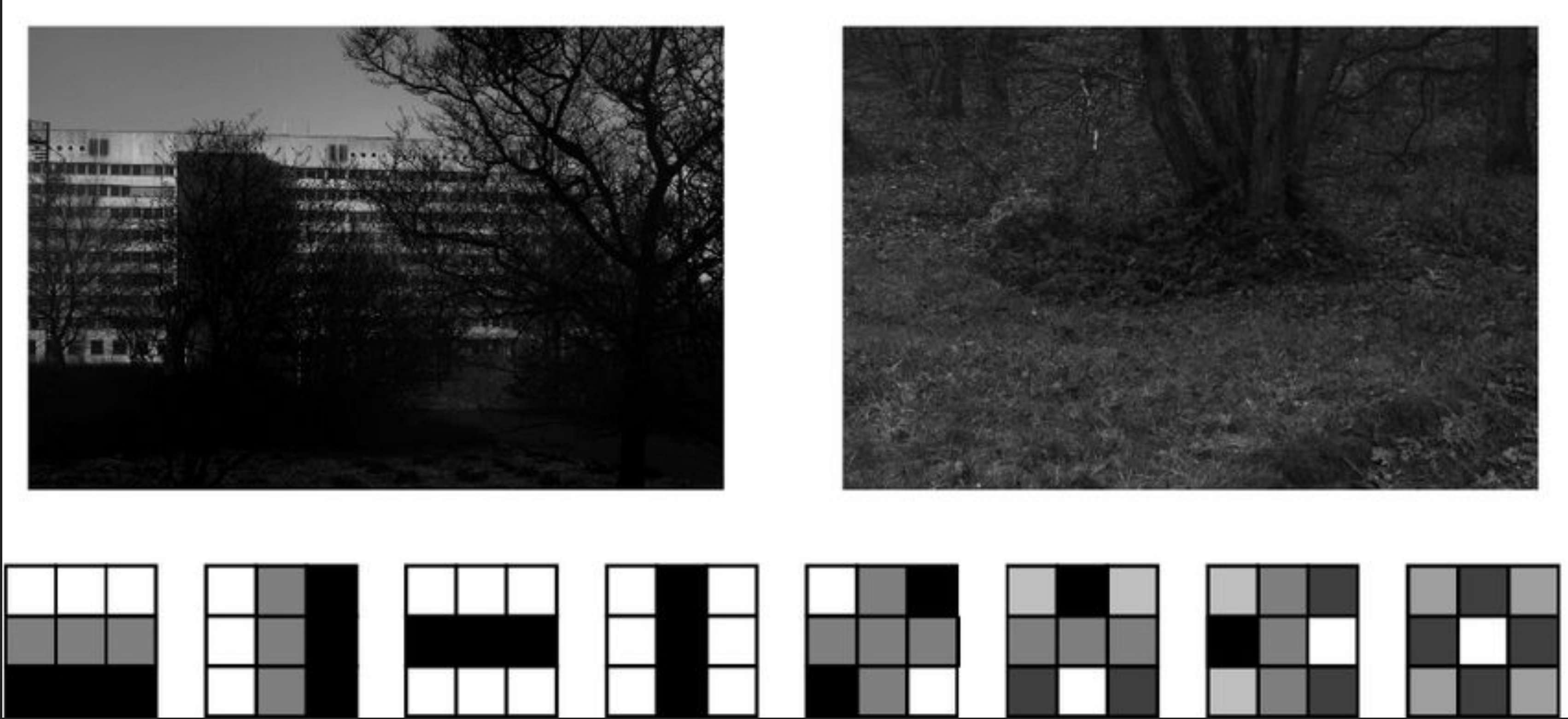
Int J Comput Vis (2008) 76: 1–12
DOI 10.1007/s11263-007-0056-x

POSITION PAPER

On the Local Behavior of Spaces of Natural Images

Gunnar Carlsson · Tigran Ishkhanov · Vin de Silva · Afra Zomorodian

Received: 19 May 2006 / Accepted: 27 March 2007 / Published online: 30 June 2007



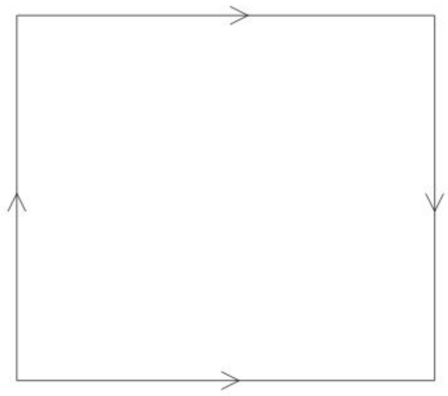


Fig. 4 Klein bottle representation as a rectangle with opposite edges identified

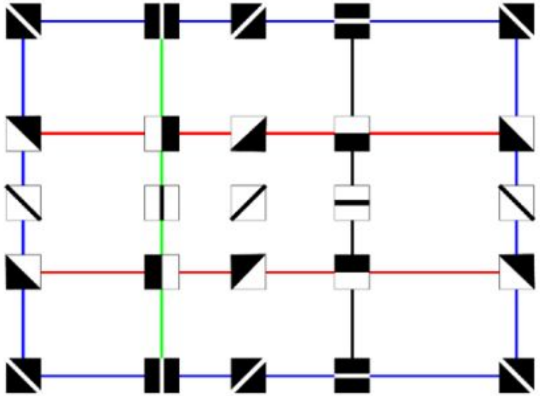


Fig. 6 3 by 3 patches parametrized by the Klein bottle

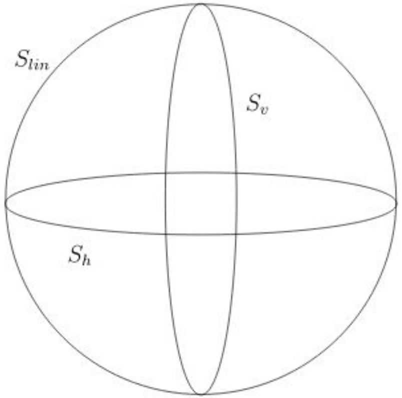


Fig. 5 The 'three circle' space

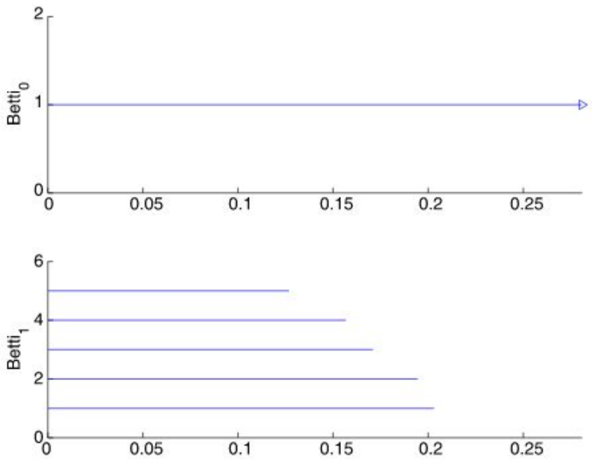
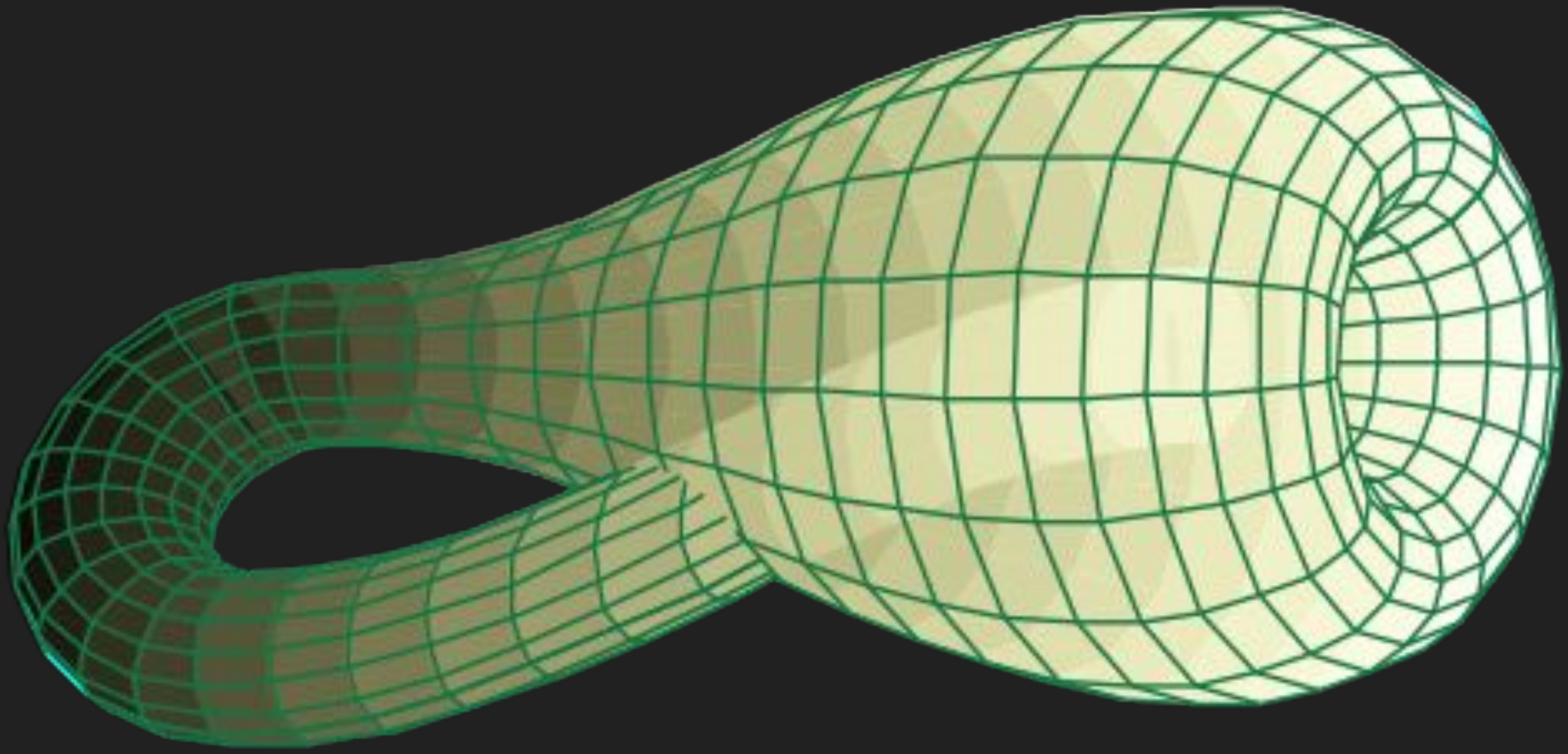
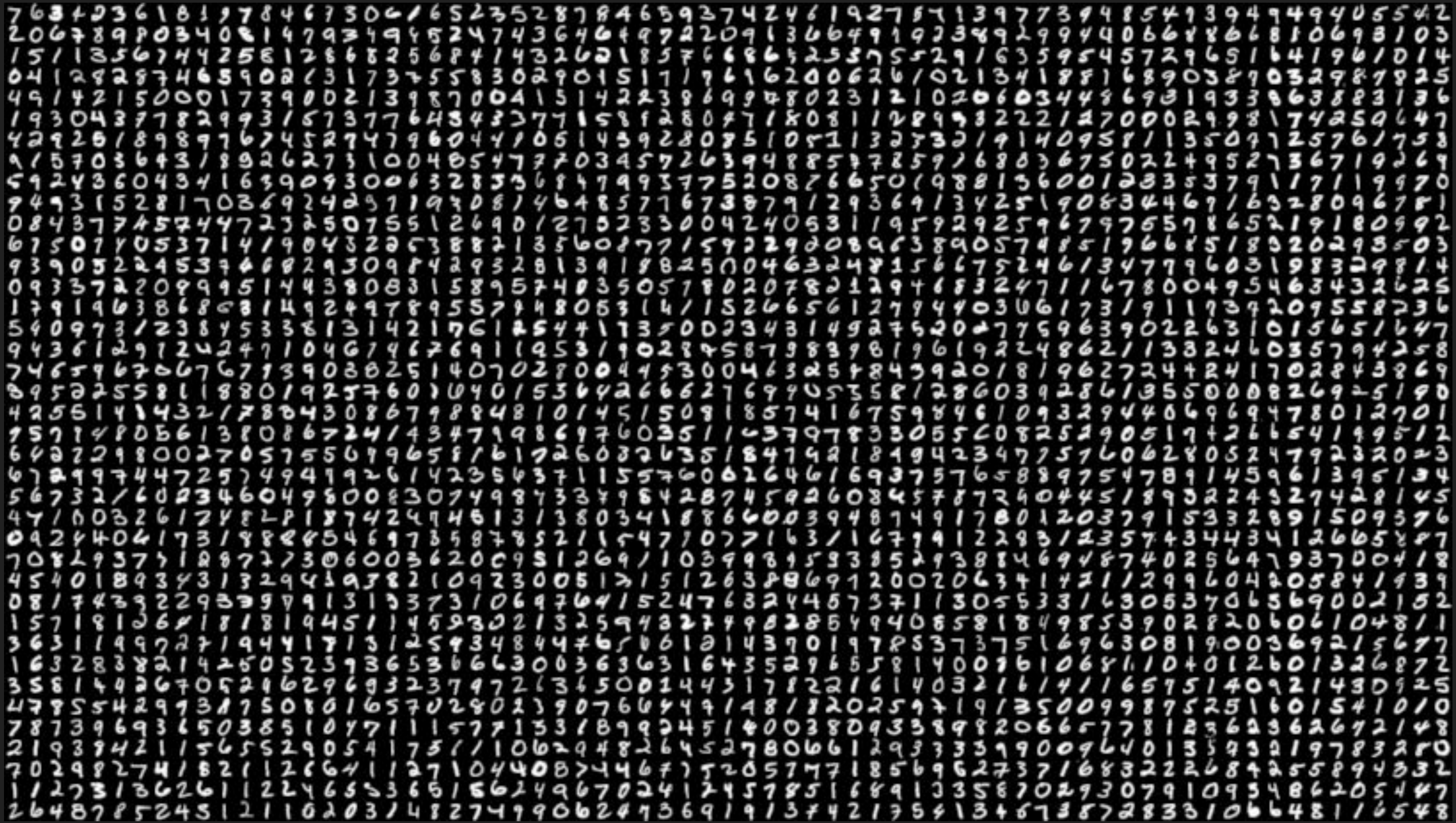
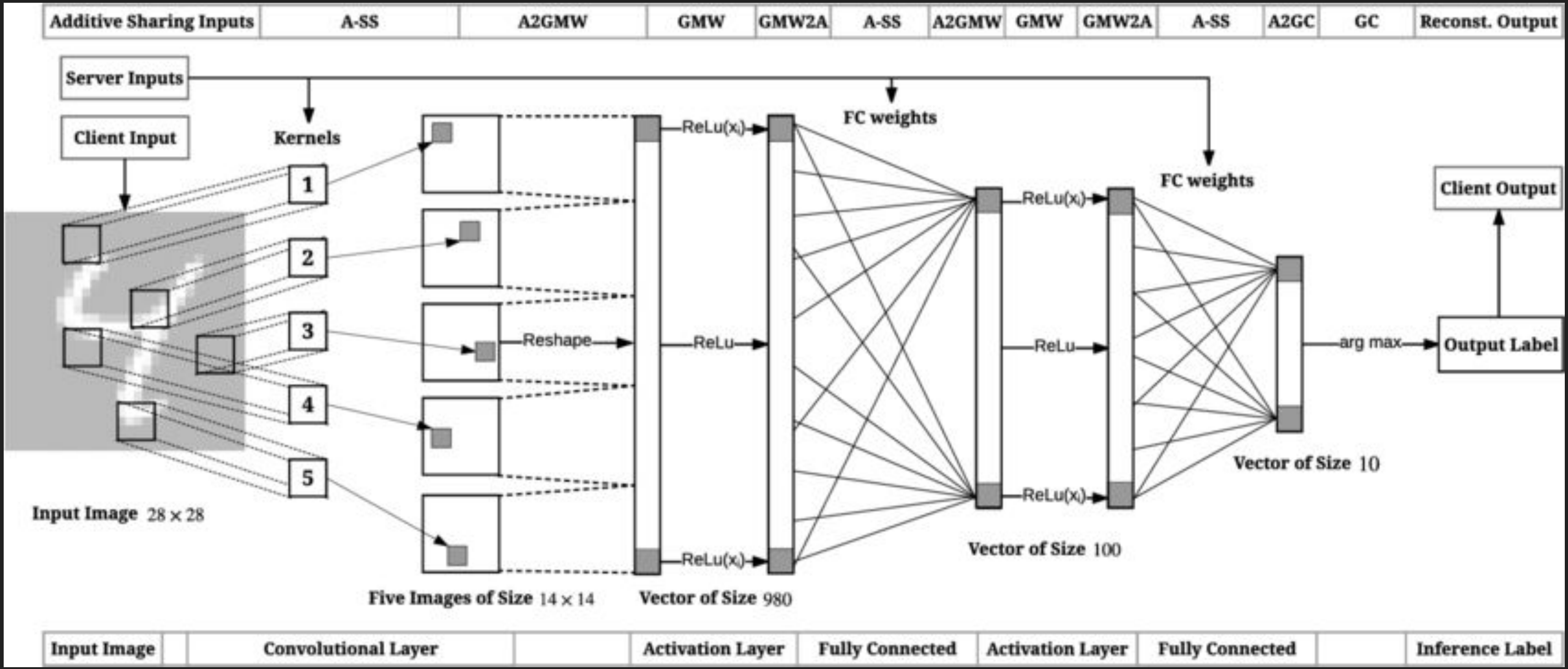


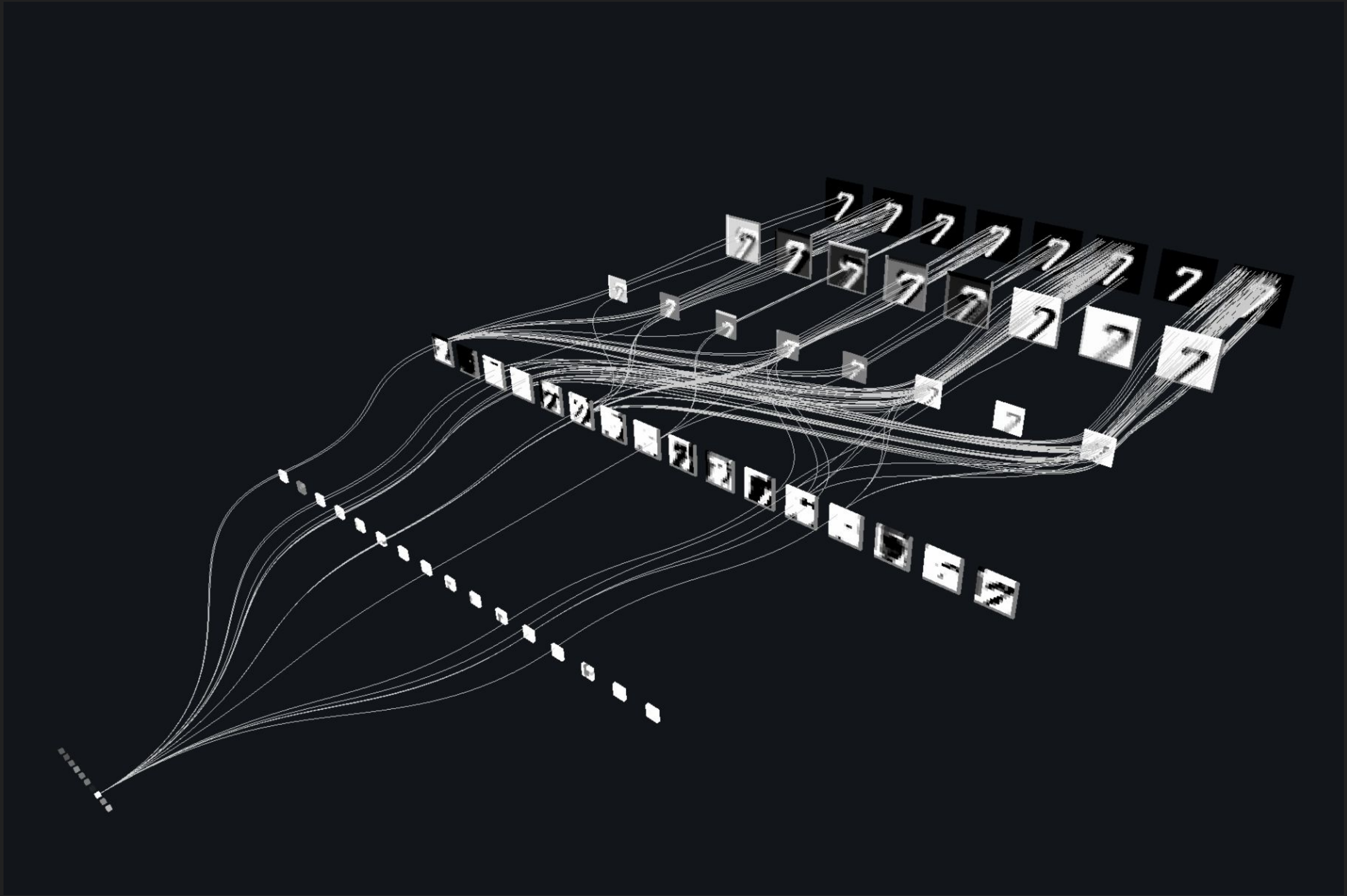
Fig. 7 $PLEX$ results for $X(15, 30)$

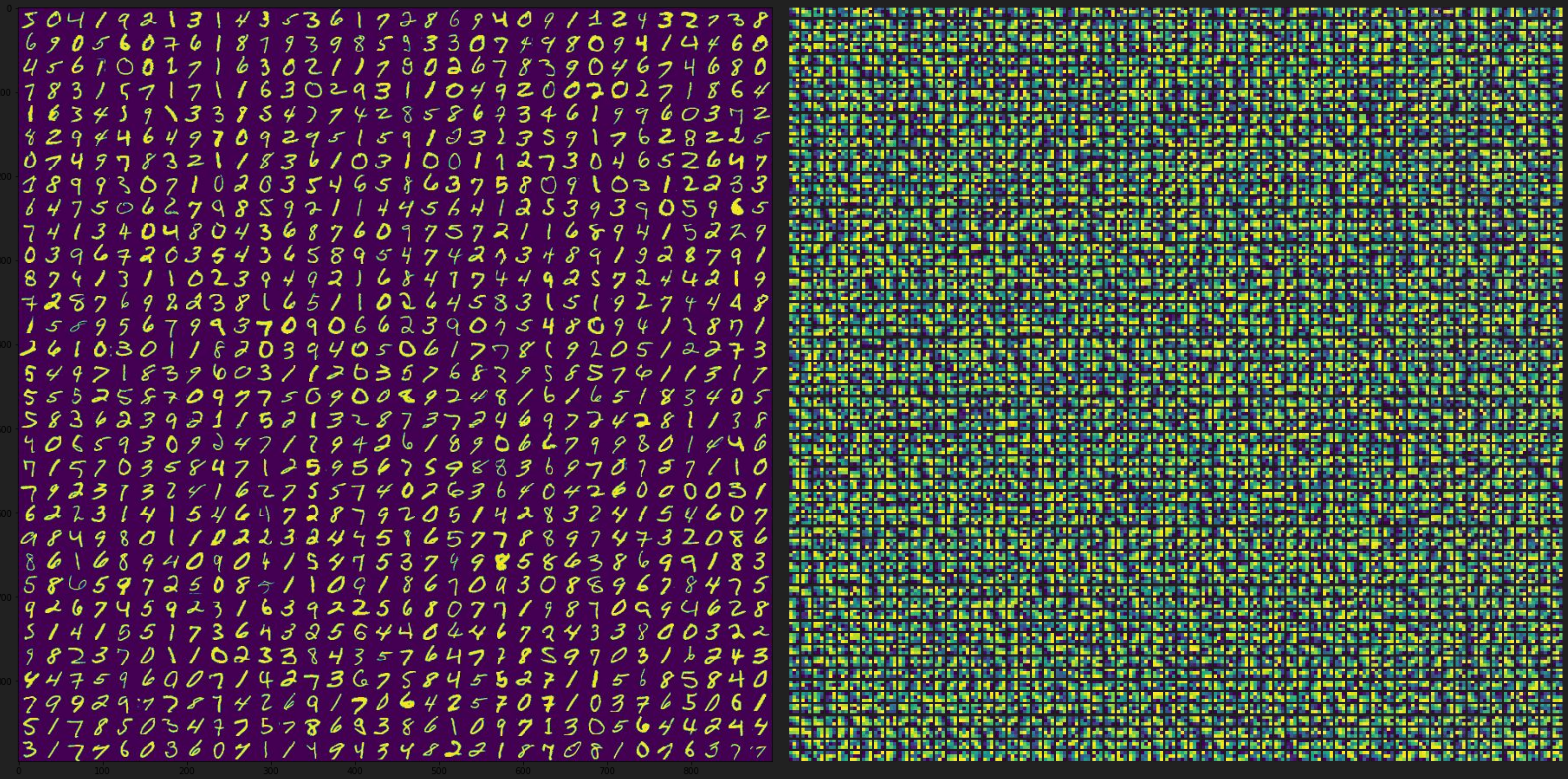






Additive Sharing Inputs
A-SS
A2GMW
GMW
GMW2A
A-SS
A2GMW
GMW
GMW2A
A-SS
A2GC
GC
Reconst. Output
Server Inputs
Client Input
Kernels
FC weights
FC weights
Client Output
1
2
3
4
5
ReLu(xi)
Reshape
ReLu
ReLu
arg max
Output Label
Input Image 28 × 28
Five Images of Size 14 × 14
Vector of Size 980
Vector of Size 100
Vector of Size 10
Input Image
Convolutional Layer
Activation Layer
Fully Connected
Activation Layer
Fully Connected
Inference Label




0
100
200
300
400
500
600
700
800

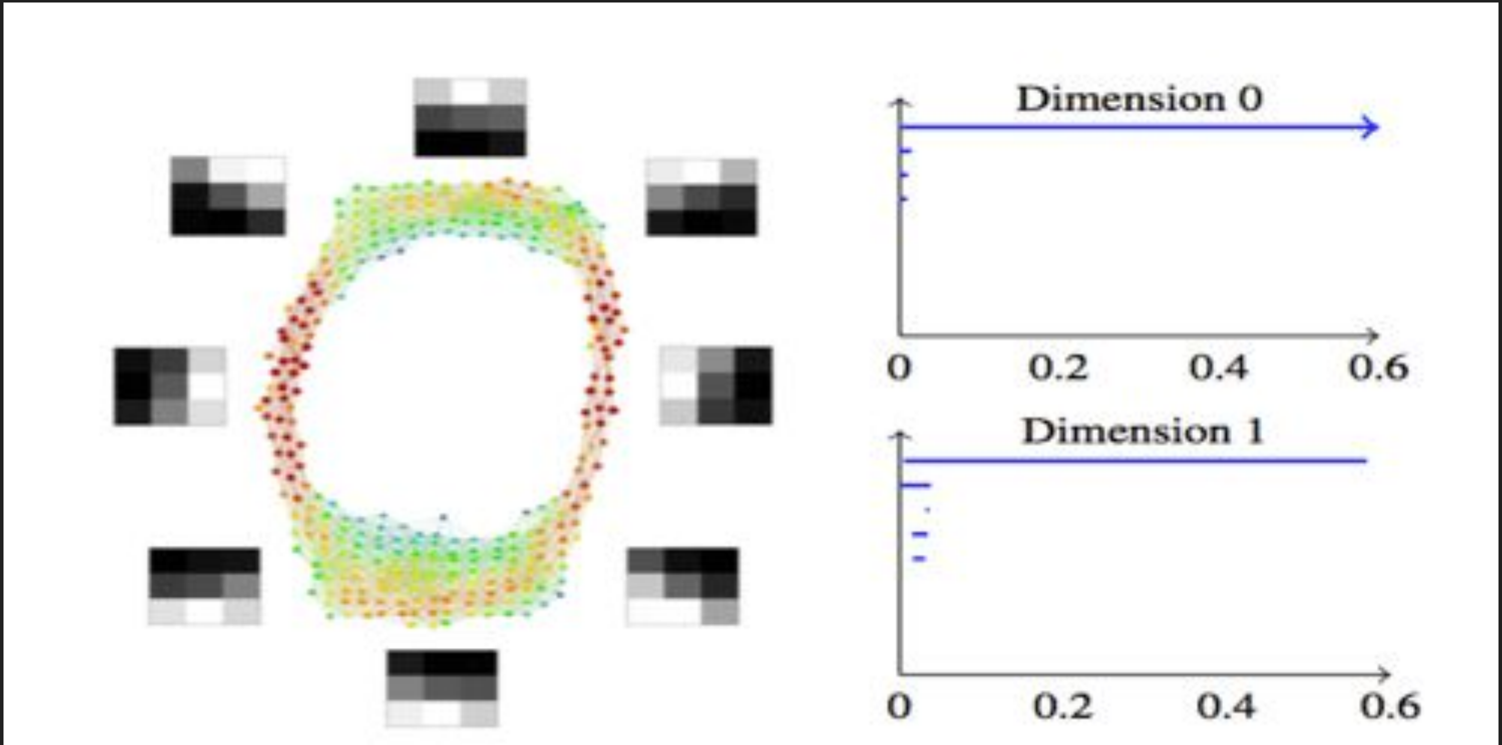
Using Topological Data Analysis to Understand the Behavior of Convolutional Neural Networks

By Gunnar Carlsson

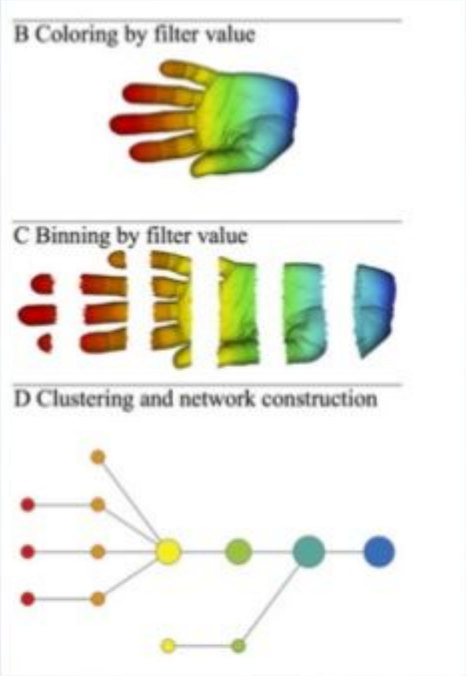
June 21, 2018

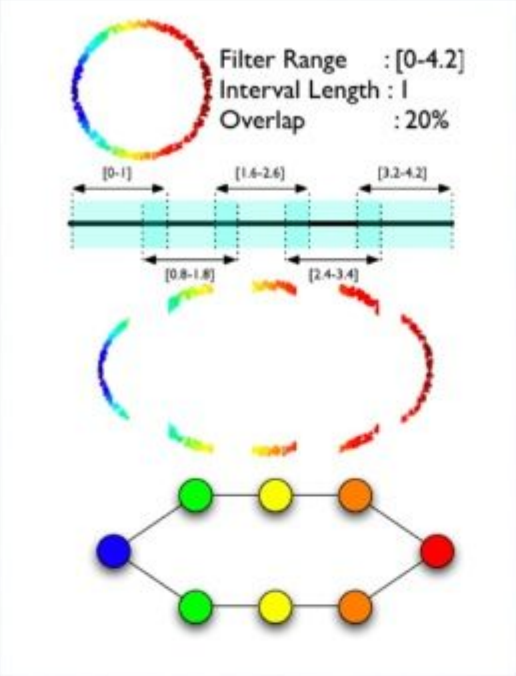
ARTIFICIAL INTELLIGENCE, MACHINE INTELLIGENCE, MACHINE LEARNING, TOPOLOGY

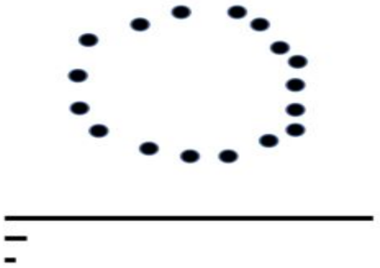
TLDR: Neural Networks are powerful but complex and opaque tools. Using Topological Data Analysis, we can describe the functioning and learning of a convolutional neural network in a compact and understandable way. The implications of the findings are profound and will accelerate the development of a wide range of applications from self-driving cars and drones to complying with things like GDPR.

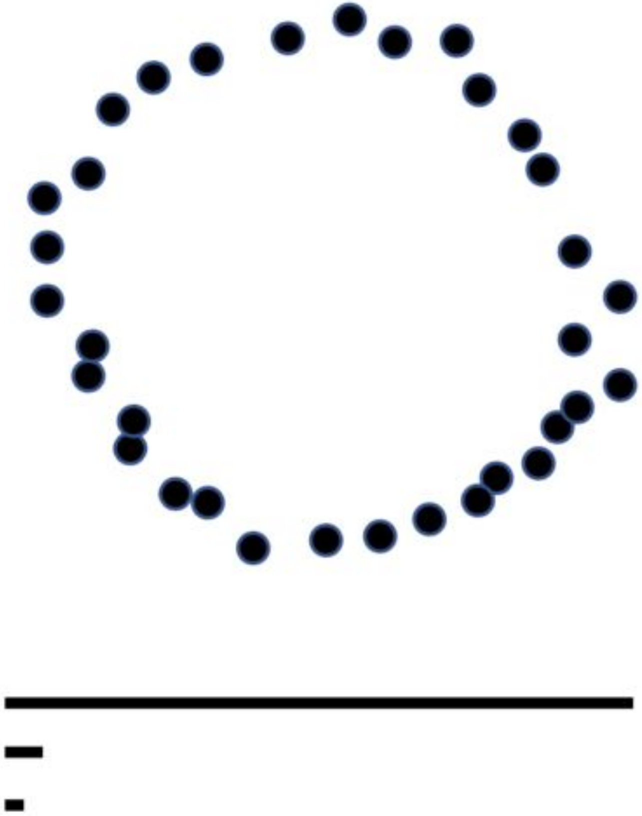

Dimension 0
0
0.2
0.4
0.6
Dimension 1
0
0.2
0.4
0.6

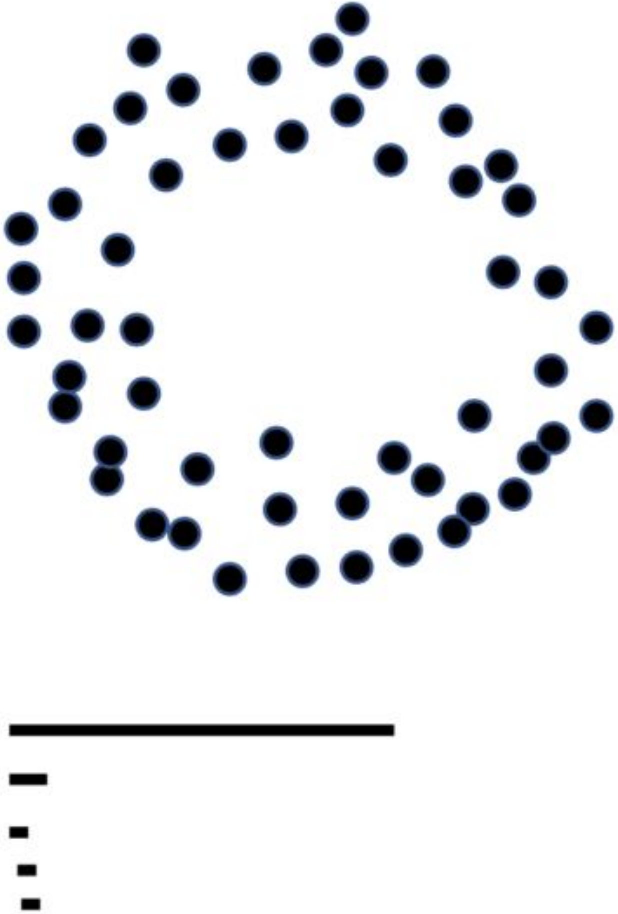
MAPPER IV

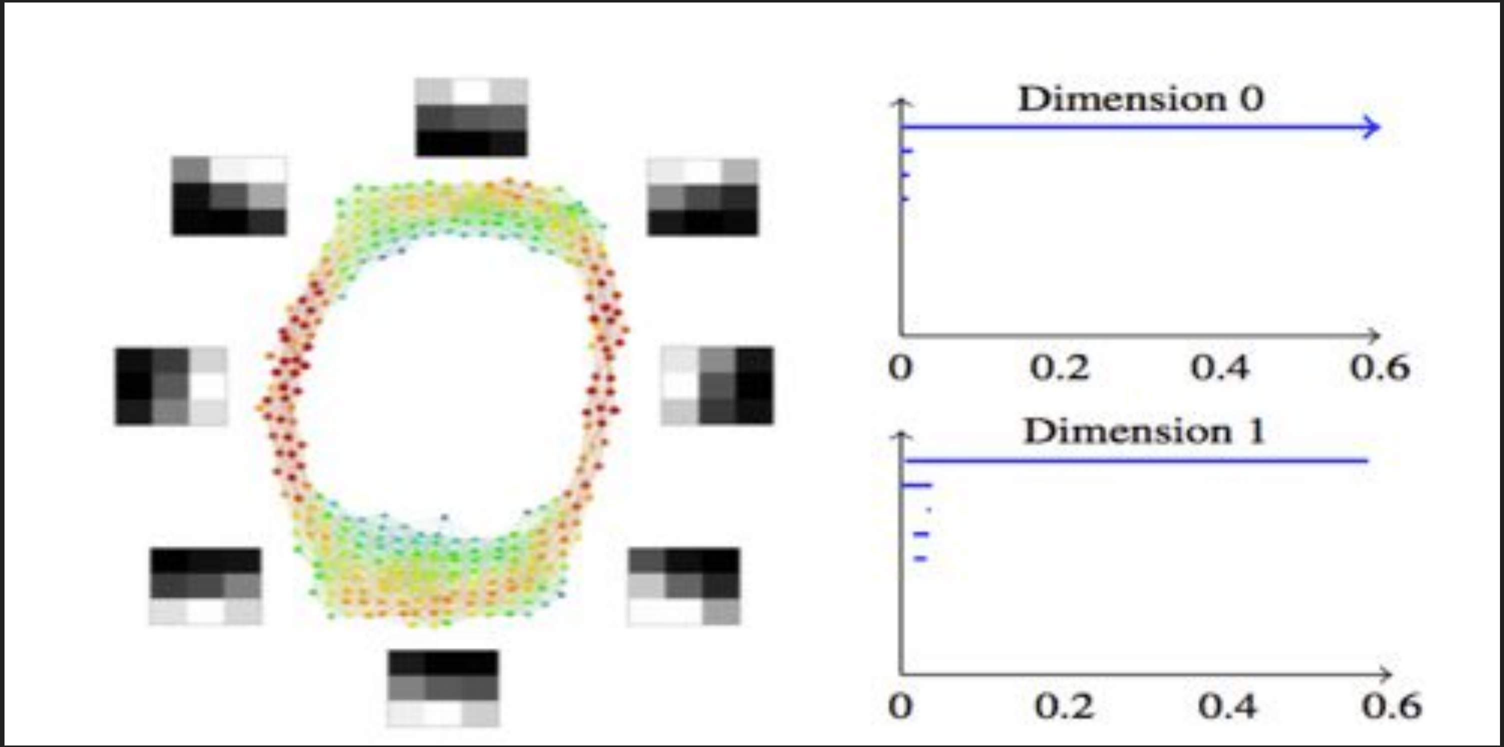

B Coloring by filter value
C Binning by filter value
D Clustering and network construction

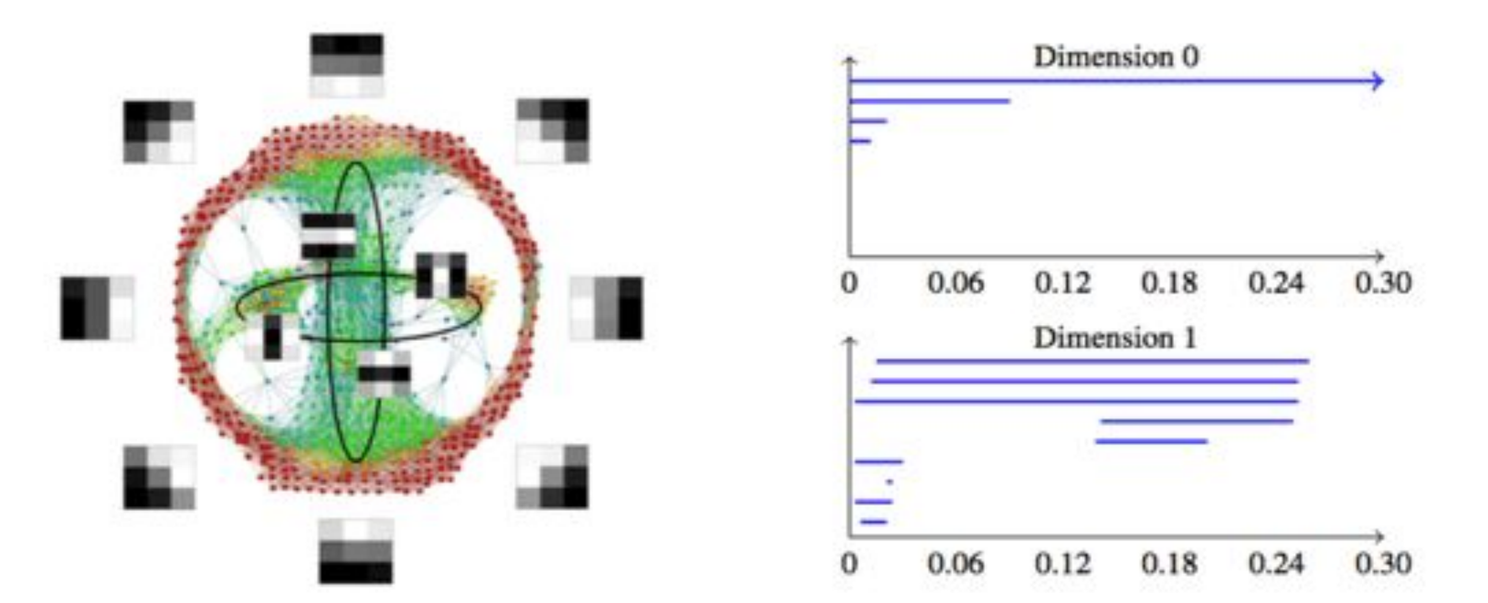

Filter Range : [0-4.2]
Interval Length : 1
Overlap : 20%
[0-1]
[1.6-2.6]
[3.2-4.2]
[0.8-1.8]
[2.4-3.4]



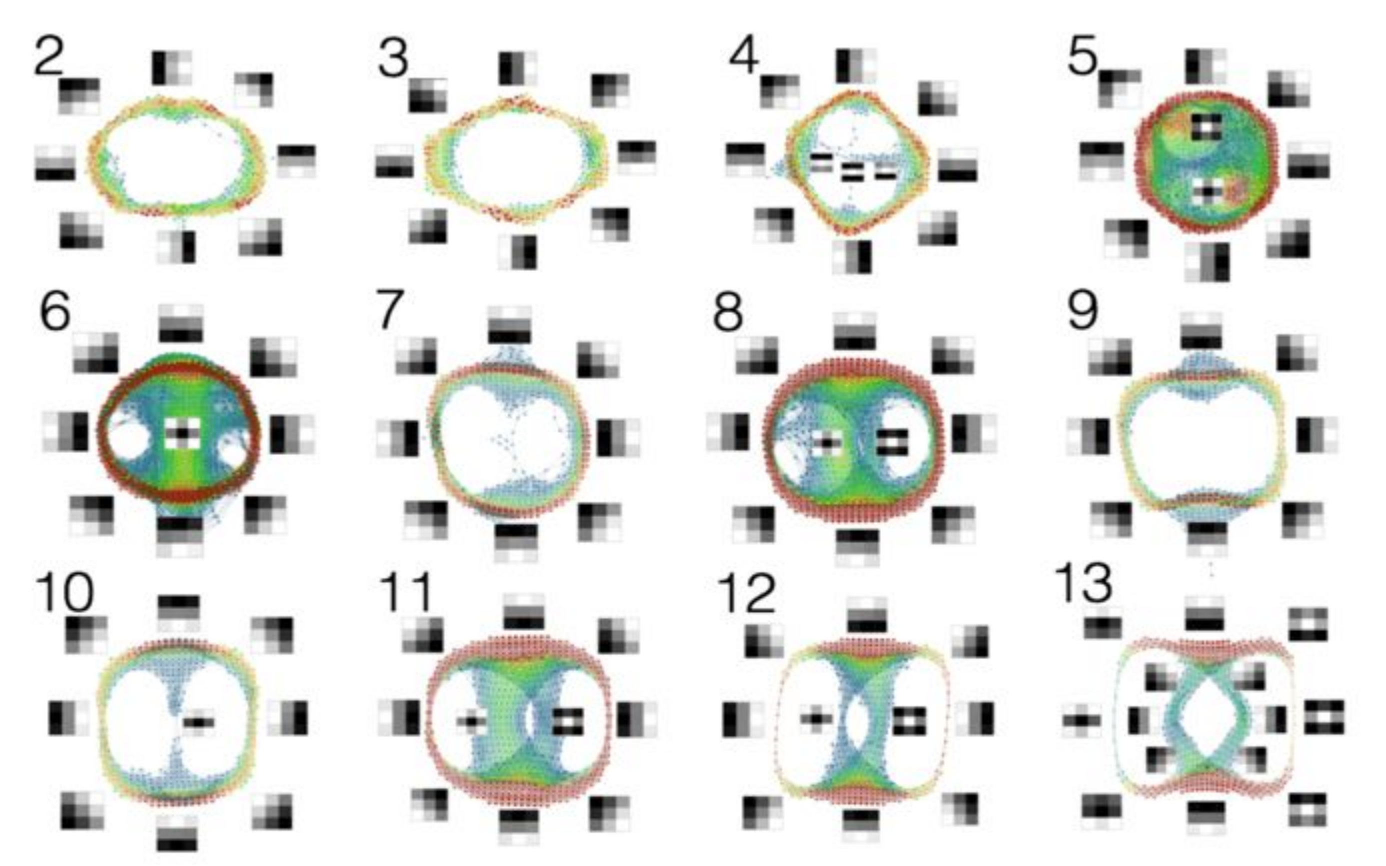


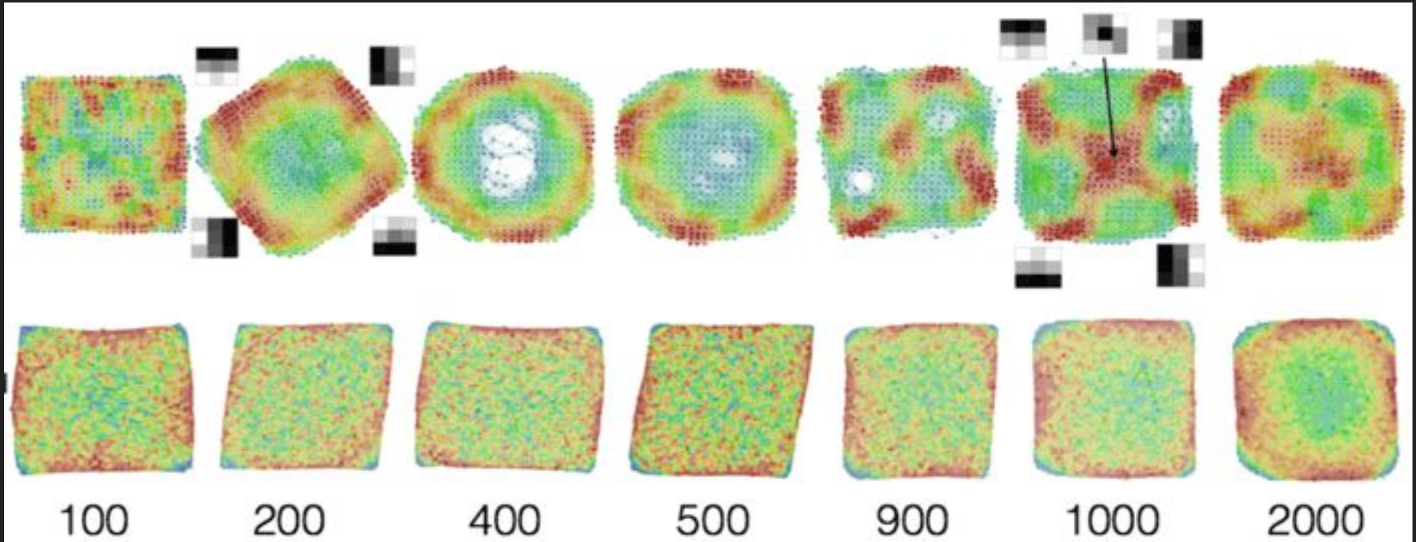



Dimension 0
0
0.2
0.4
0.6
Dimension 1
0
0.2
0.4
0.6



Dimension 0
0
0.06
0.12
0.18
0.24
0.30
Dimension 1
0
0.06
0.12
0.18
0.24
0.30


2
3
4
5
6
7
8
9
10
11
12
13


100
200
400
500
900
1000
2000

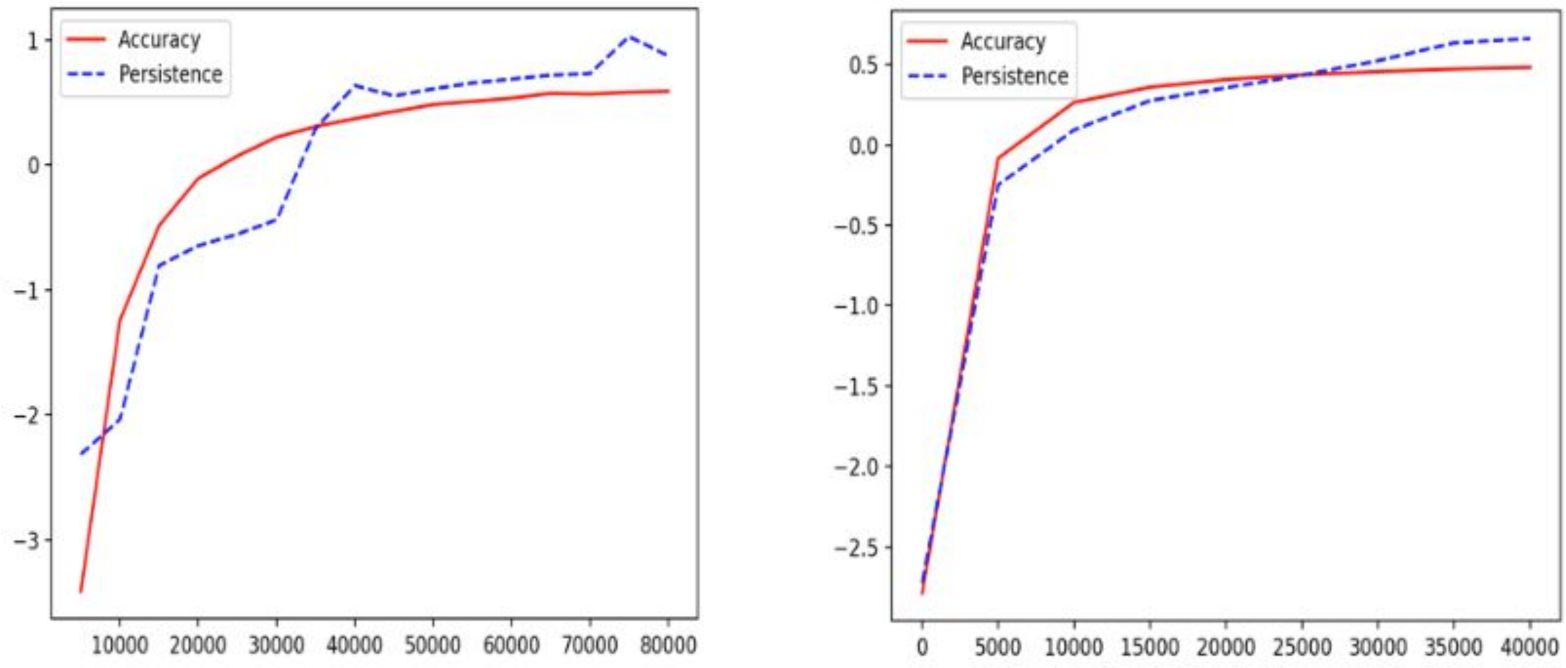
Going Deeper: Understanding How Convolutional Neural Networks Learn Using TDA

By Gunnar Carlsson

August 9, 2018

ARTIFICIAL INTELLIGENCE, MACHINE INTELLIGENCE, MACHINE LEARNING, TOPOLOGY

In my earlier post I discussed how performing topological data analysis on the weights learned by convolutional neural nets (CNN's) can give insight into what is being learned and how it is being learned.



Accuracy
Persistence
1
0
−1
−2
−3
10000
20000
30000
40000
50000
60000
70000
80000
Accuracy
Persistence
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
0
5000
10000
15000
20000
25000
30000
35000
40000

Mathematical Acceleration: Incorporating Prior Information to Make Neural Nets Learn 3.5X Faster

By Gunnar Carlsson

August 30, 2018

ARTIFICIAL INTELLIGENCE, MACHINE LEARNING, TOPOLOGY

Validation Accuracy	# Batch iterations Boosted	# Batch iterations standard
.8	187	293
.9	378	731
.95	1046	2052
.96	1499	2974
.97	2398	4528
.98	5516	8802
.985	9584	16722

Validation Accuracy	# Batch iterations Boosted	# Batch iterations standard
.25	303	1148
.5	745	2464
.75	1655	5866
.8	2534	8727
.83	4782	13067
.84	6312	15624
.85	8426	21009

Thank you !


PLEASE STAND BY
325
50
575
300
35
45
30
20

Weapon of choice

गुढी GUDHI Geometry Understanding in Higher Dimensions

The GUDHI library is a generic open source **C++ library**, with a **Python interface**, for Topological Data Analysis (**TDA**) and Higher Dimensional Geometry Understanding. The library offers state-of-the-art data structures and algorithms to construct simplicial complexes and compute persistent homology.

The library comes with data sets, demos, examples and test suites.

The GUDHI library is developed as part of the **GUDHI project** supported by the **European Research Council**.

NEW RELEASE

GUDHI version 2.2.0

As a major new feature, the GUDHI library now offers a Čech complex module, a sparse version of the Rips complex and a utility to build the Rips complex from a correlation matrix (no Python interface yet).

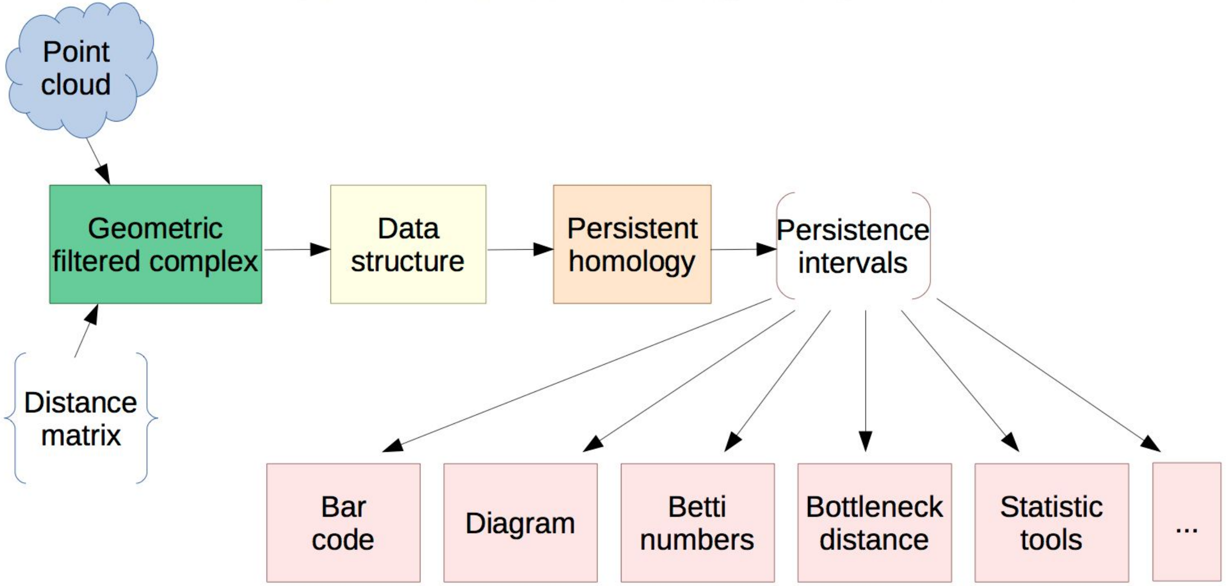
More Articles

- New release · **GUDHI version 2.1.0 Debian package**
- New release · **GUDHI version 2.1.0**
- New release · **GUDHI version 2.0.1**

More ›

गुढी GUDHI

Geometric Understanding in Higher Dimensions

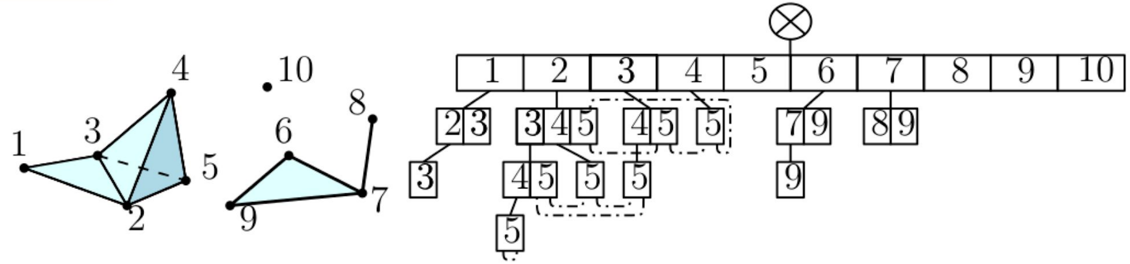


गुडी GUDHI

Geometric Understanding in Higher Dimensions

Data structure

Filtered simplicial complexes – Simplex tree

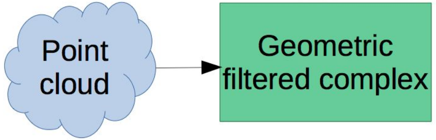


- Memory and time–efficient data structure to store simplicial complexes.
- Every simplex is a word stored in the tree.
- The nodes corresponding to simplices of the same dimension having the same maximal vertex are stored in a cyclic list.
- It is a base of all algorithms to compute persistence of weighted simplicial complexes in GUDHI.

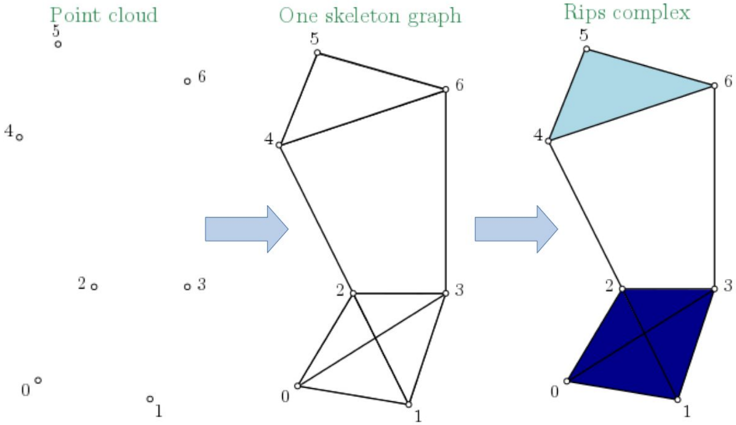
by Clément Maria

गुढी GUDHI

Geometric Understanding in Higher Dimensions



Geometric filtered complex – Rips from a point cloud



by Clément Maria

mapper 0.1.17

```
pip install mapper
```

Latest version

Last released: Apr 19, 2017

Python Mapper: an open source tool for exploration, analysis and visualization of data.

Navigation

- Project description
- Release history
- Download files

Project links

- Homepage

Project description

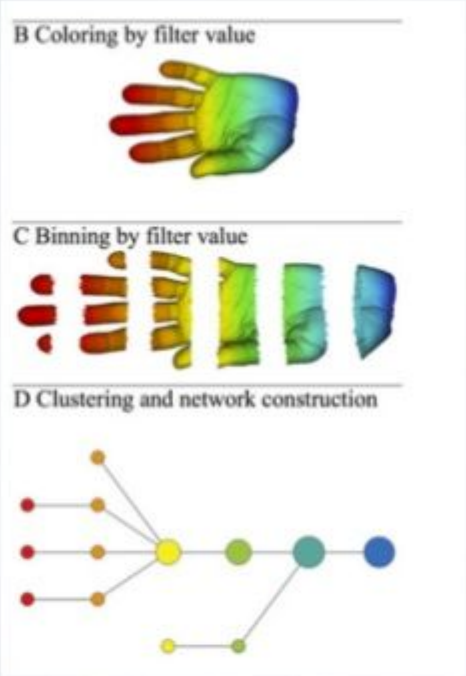
See the project home page http://danifold.net/mapper for a detailed description and documentation.

This package features both a GUI and a Python package for custom scripts. The Python package itself works with Python 2 and 3. The GUI, however, depends on wxPython, which is available for Python 2 only. Therefore, the setup script will install the GUI only if it is executed by Python 2.

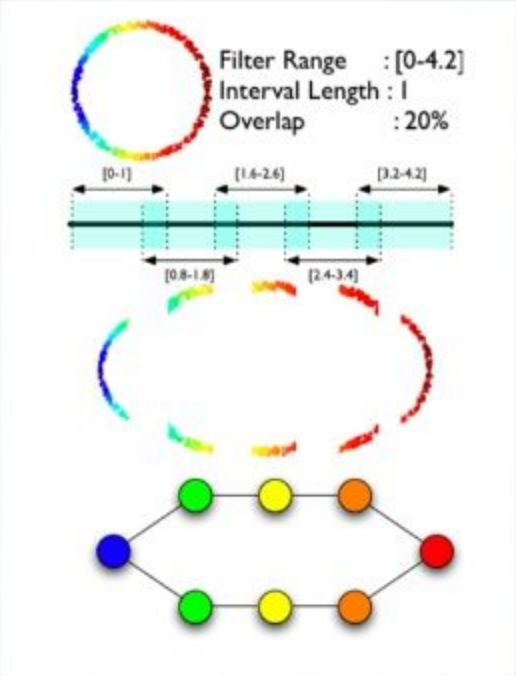
See also https://pypi.python.org/pypi/cmappertools for the companion package with fast C++ algorithms.

The authors of Python mapper are Daniel Müllner and Aravindakshan Babu. (PyPI apparently suppresses everything but the first name in the "author" field, hence only one author is displayed below.)

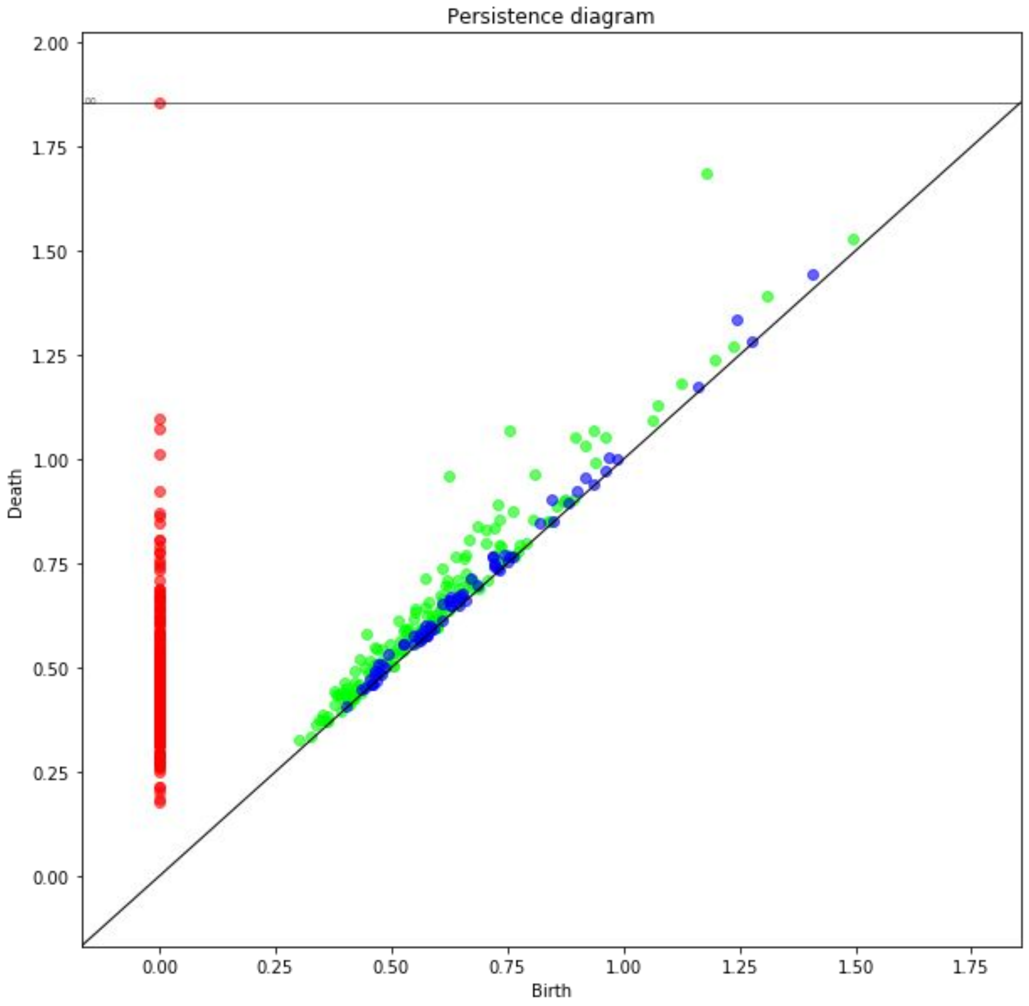
MAPPER IV



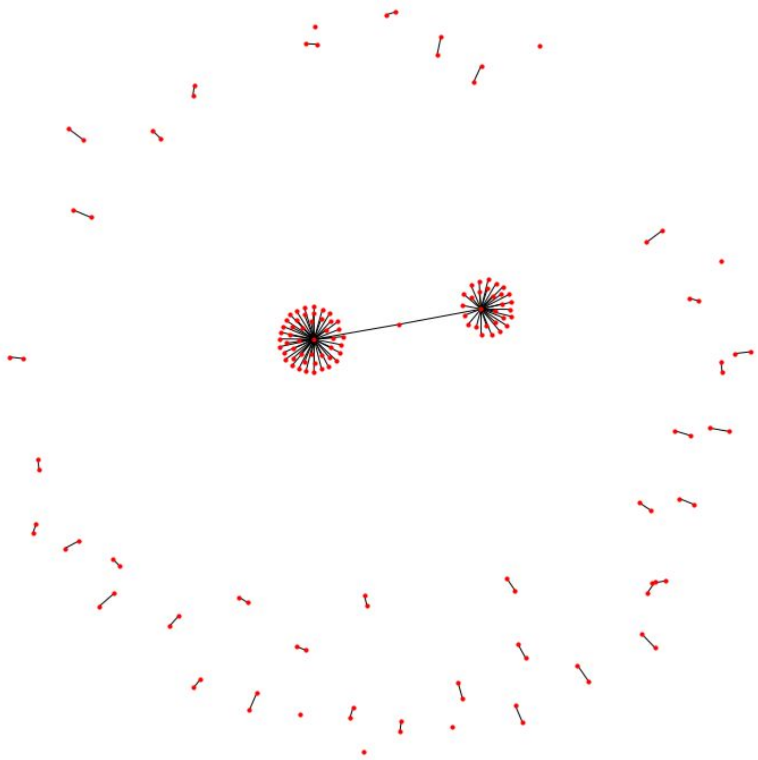
B Coloring by filter value
C Binning by filter value
D Clustering and network construction

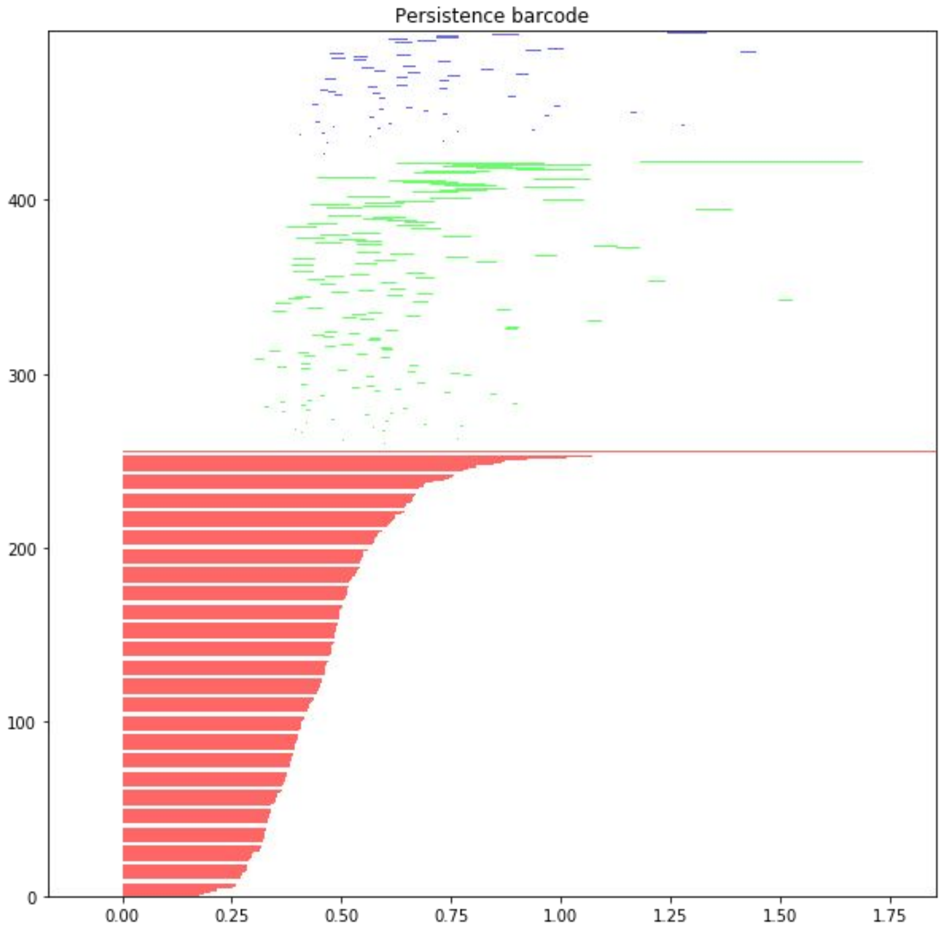


Filter Range : [0-4.2]
Interval Length : 1
Overlap : 20%
[0-1]
[1.6-2.6]
[3.2-4.2]
[0.8-1.8]
[2.4-3.4]

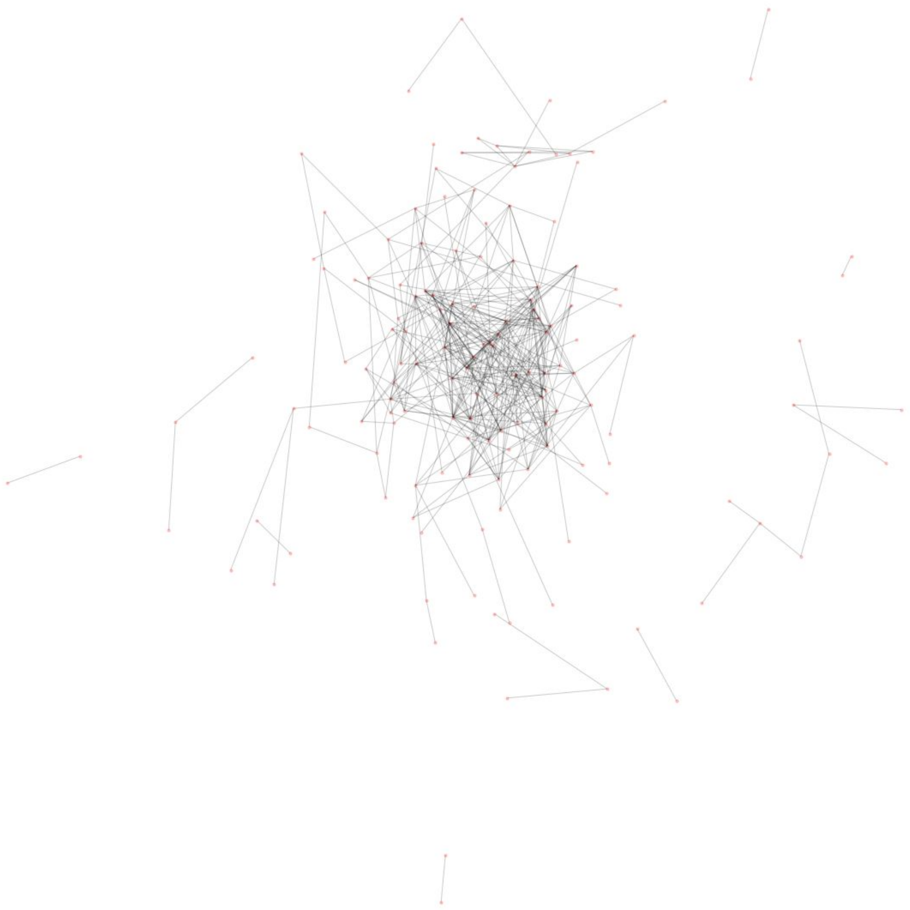


Persistence diagram
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Death
∞
-0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Birth

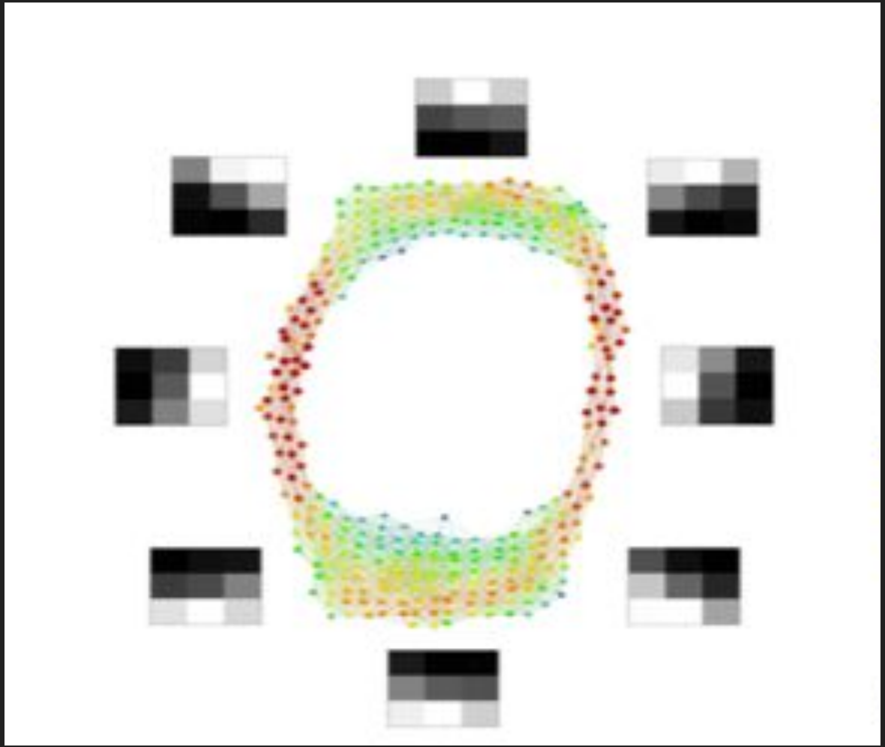




Persistence barcode
0
100
200
300
400
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75



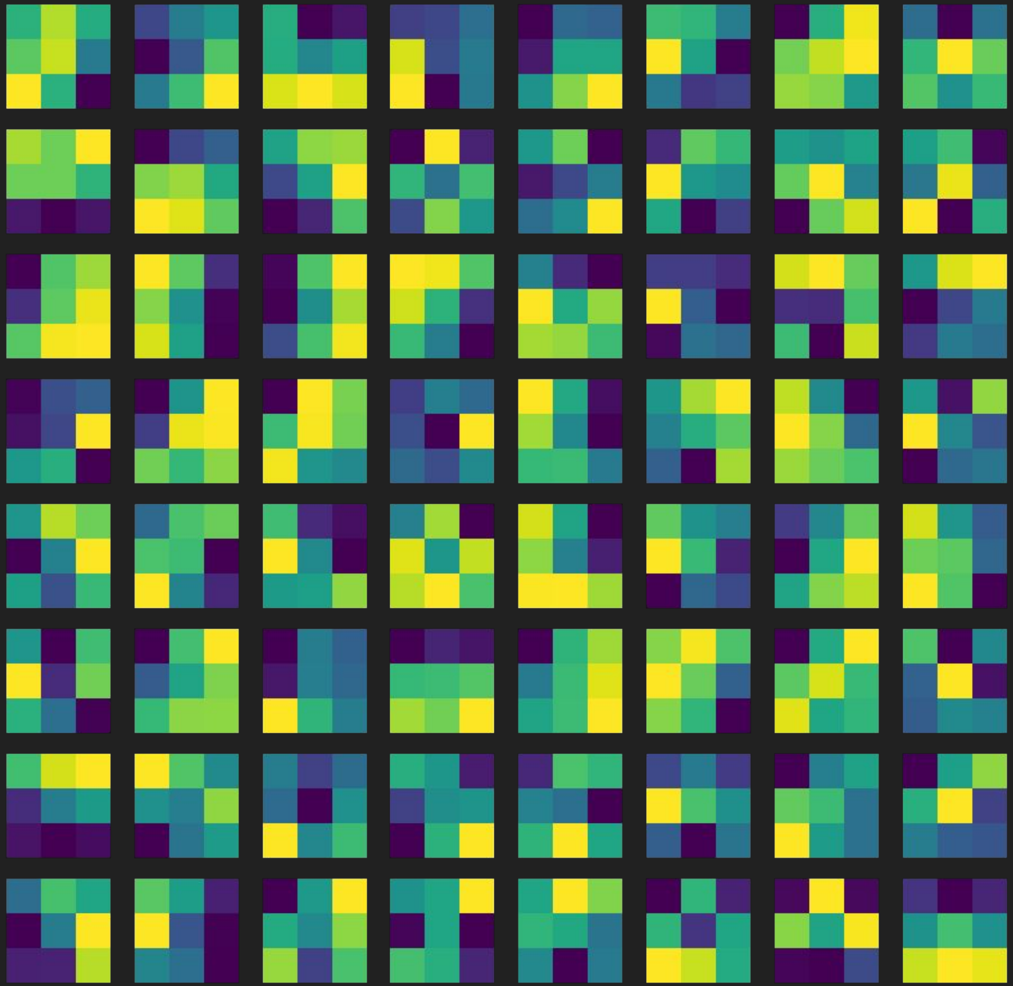
???

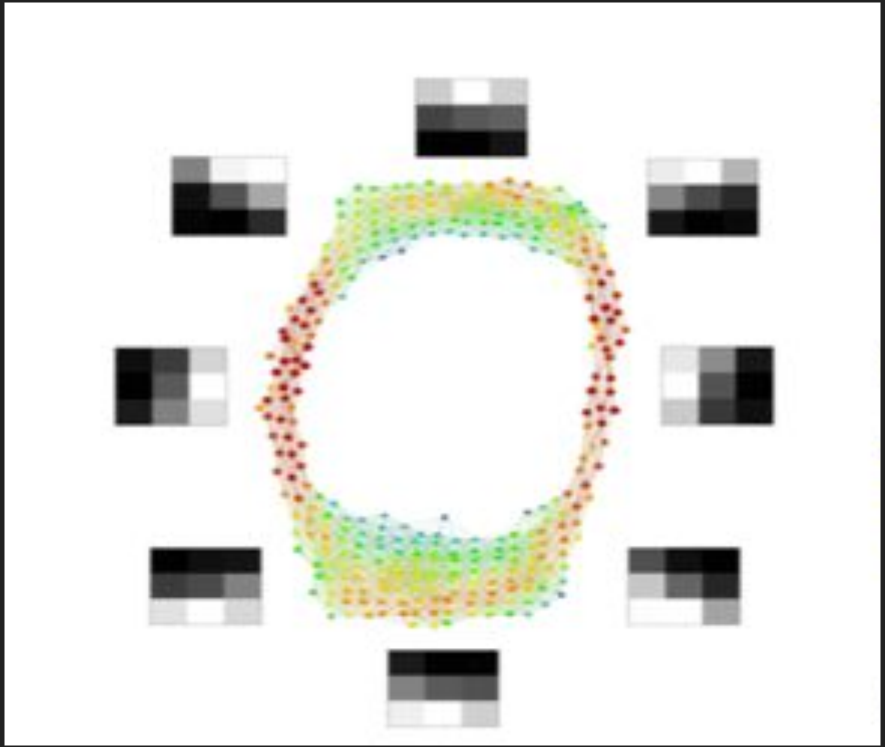


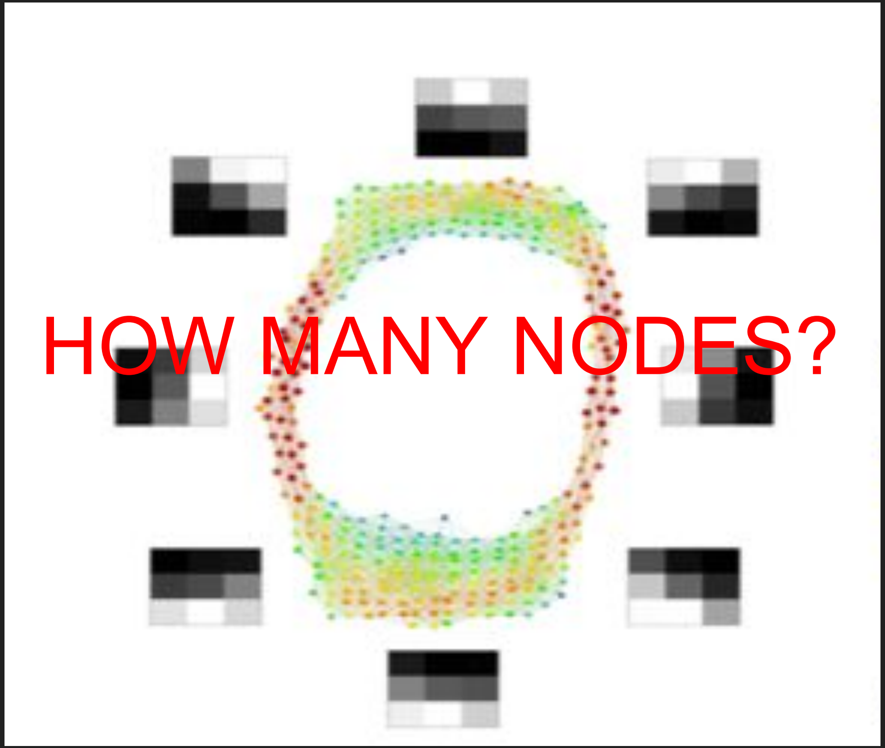
```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 64, 26, 26)        640
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 32, 13, 26)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 16, 12, 25)        2064
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 8, 6, 25)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 1200)              0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                12010
=================================================================
Total params: 14,714
Trainable params: 14,714
Non-trainable params: 0
_________________________________________________________________
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/8
60000/60000 [==============================] - 3s 53us/step - loss: 0.4136 - acc: 0.8756 - val_loss: 0.1806 - val_acc
: 0.9449
Epoch 2/8
60000/60000 [==============================] - 3s 47us/step - loss: 0.1528 - acc: 0.9548 - val_loss: 0.1083 - val_acc
: 0.9674
Epoch 3/8
60000/60000 [==============================] - 3s 48us/step - loss: 0.1086 - acc: 0.9673 - val_loss: 0.0814 - val_acc
: 0.9741
Epoch 4/8
60000/60000 [==============================] - 3s 48us/step - loss: 0.0862 - acc: 0.9739 - val_loss: 0.0671 - val_acc
: 0.9795
Epoch 5/8
60000/60000 [==============================] - 3s 48us/step - loss: 0.0702 - acc: 0.9788 - val_loss: 0.0622 - val_acc
: 0.9797
Epoch 6/8
60000/60000 [==============================] - 3s 48us/step - loss: 0.0605 - acc: 0.9822 - val_loss: 0.0509 - val_acc
: 0.9828
Epoch 7/8
60000/60000 [==============================] - 3s 48us/step - loss: 0.0540 - acc: 0.9837 - val_loss: 0.0562 - val_acc
: 0.9826
Epoch 8/8
60000/60000 [==============================] - 3s 47us/step - loss: 0.0486 - acc: 0.9850 - val_loss: 0.0528 - val_acc
: 0.9829
Test loss: 0.05276332234479487
Test accuracy: 0.9829
```



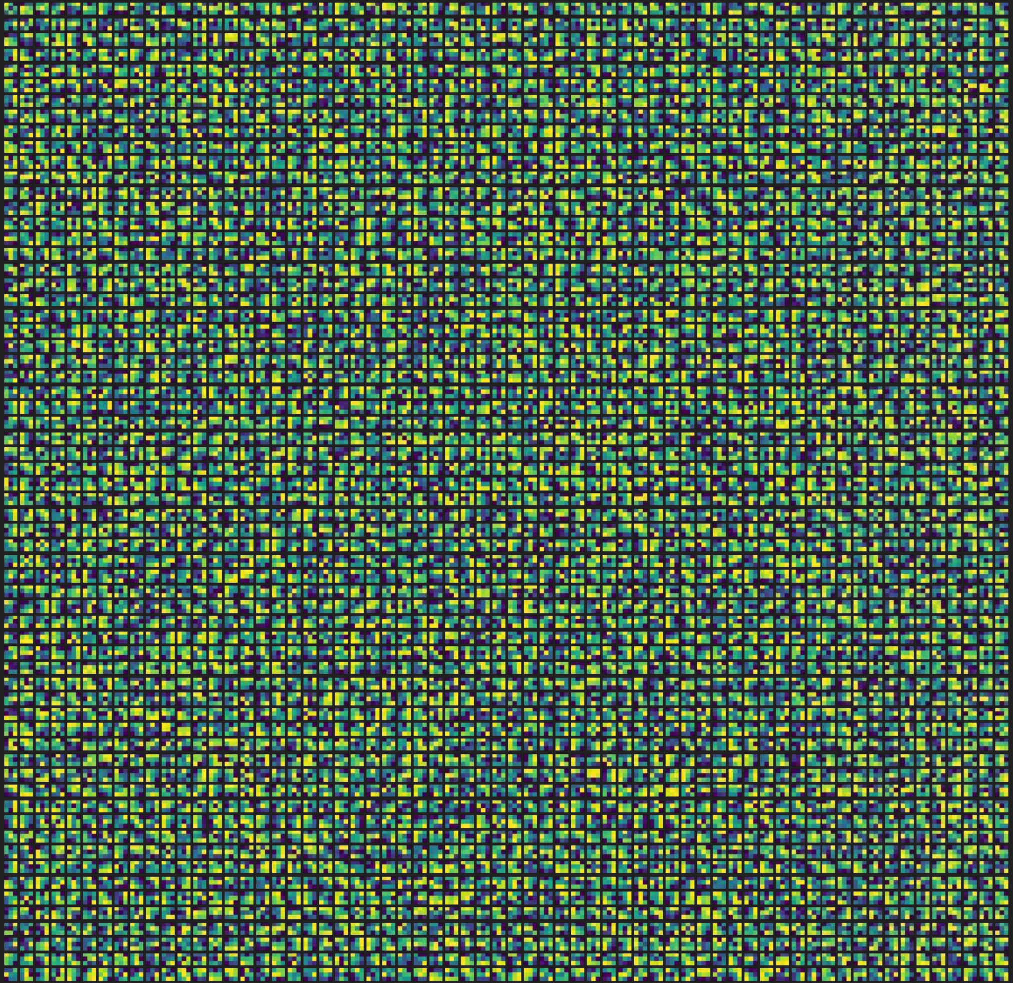


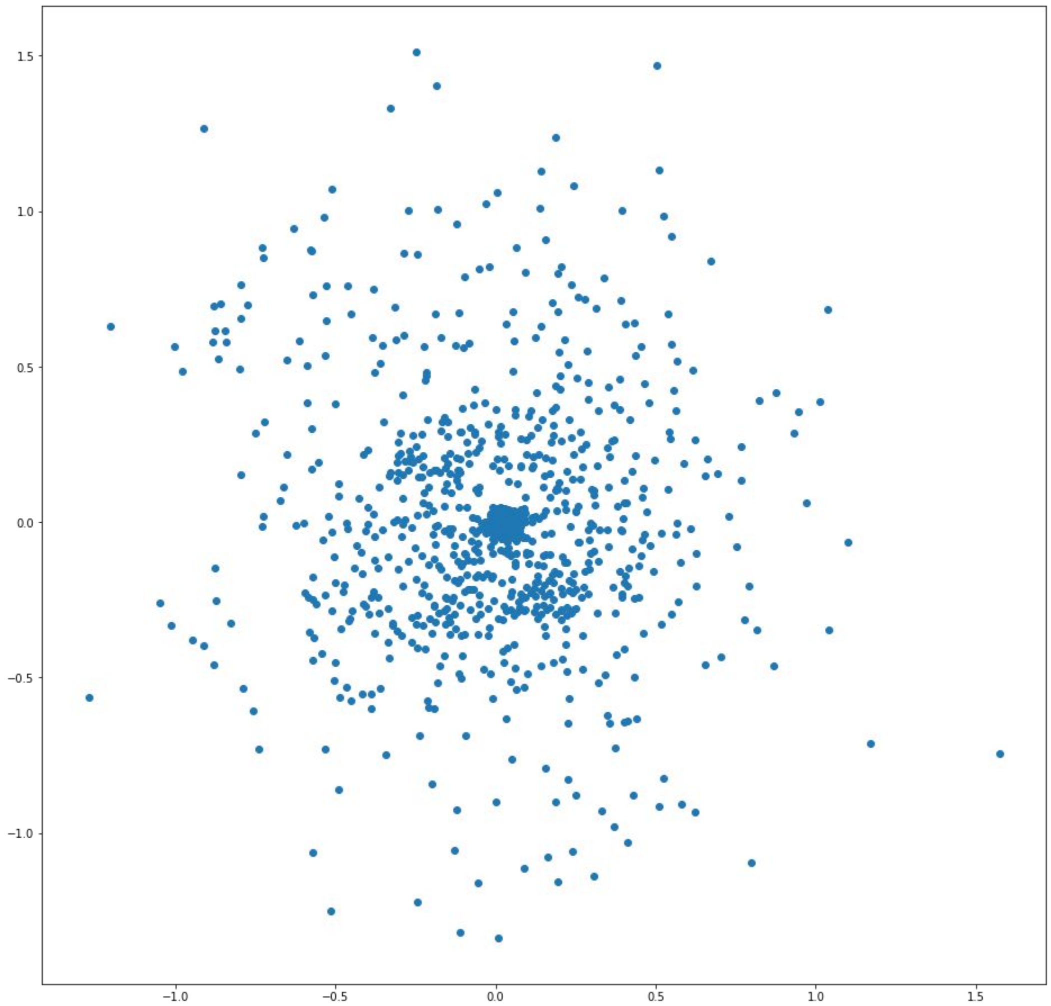

HOW MANY NODES?

Ok

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 4096, 26, 26)      40960
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 2048, 13, 26)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 12, 25)        131088
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 6, 25)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1200)              0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                12010
=================================================================
Total params: 184,058
Trainable params: 184,058
Non-trainable params: 0
_________________________________________________________________
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/16
60000/60000 [==============================] - 78s 1ms/step - loss: 0.2617 - acc: 0.9183 - val_loss: 0.0865 -
val_acc: 0.9728
Epoch 2/16
60000/60000 [==============================] - 77s 1ms/step - loss: 0.0858 - acc: 0.9737 - val_loss: 0.0637 -
val_acc: 0.9806
Epoch 3/16
60000/60000 [==============================] - 77s 1ms/step - loss: 0.0650 - acc: 0.9804 - val_loss: 0.0517 -
val_acc: 0.9843
Epoch 4/16
60000/60000 [==============================] - 77s 1ms/step - loss: 0.0547 - acc: 0.9827 - val_loss: 0.0522 -
val_acc: 0.9826
Epoch 5/16
60000/60000 [==============================] - 77s 1ms/step - loss: 0.0481 - acc: 0.9846 - val_loss: 0.0658 -
val_acc: 0.9801
Epoch 6/16
60000/60000 [==============================] - 77s 1ms/step - loss: 0.0436 - acc: 0.9860 - val_loss: 0.0460 -
val_acc: 0.9845
Epoch 7/16
60000/60000 [==============================] - 77s 1ms/step - loss: 0.0376 - acc: 0.9882 - val_loss: 0.0539 -
val_acc: 0.9841
Epoch 8/16
60000/60000 [==============================] - 77s 1ms/step - loss: 0.0347 - acc: 0.9891 - val_loss: 0.0480 -
val_acc: 0.9854
Epoch 9/16
60000/60000 [==============================] - 76s 1ms/step - loss: 0.0310 - acc: 0.9900 - val_loss: 0.0512 -
val_acc: 0.9850
Epoch 10/16
60000/60000 [==============================] - 76s 1ms/step - loss: 0.0296 - acc: 0.9903 - val_loss: 0.0621 -
val_acc: 0.9814
Epoch 11/16
60000/60000 [==============================] - 76s 1ms/step - loss: 0.0264 - acc: 0.9913 - val_loss: 0.0484 -
val_acc: 0.9857
Epoch 12/16
60000/60000 [==============================] - 75s 1ms/step - loss: 0.0247 - acc: 0.9916 - val_loss: 0.0486 -
val_acc: 0.9863
Epoch 13/16
60000/60000 [==============================] - 75s 1ms/step - loss: 0.0229 - acc: 0.9923 - val_loss: 0.0623 -
val_acc: 0.9821
Epoch 14/16
60000/60000 [==============================] - 75s 1ms/step - loss: 0.0200 - acc: 0.9935 - val_loss: 0.0592 -
val_acc: 0.9846
Epoch 15/16
60000/60000 [==============================] - 75s 1ms/step - loss: 0.0200 - acc: 0.9933 - val_loss: 0.0719 -
val_acc: 0.9824
Epoch 16/16
60000/60000 [==============================] - 75s 1ms/step - loss: 0.0176 - acc: 0.9944 - val_loss: 0.0634 -
val_acc: 0.9839
Test loss: 0.06344677277751035
Test accuracy: 0.9839
```

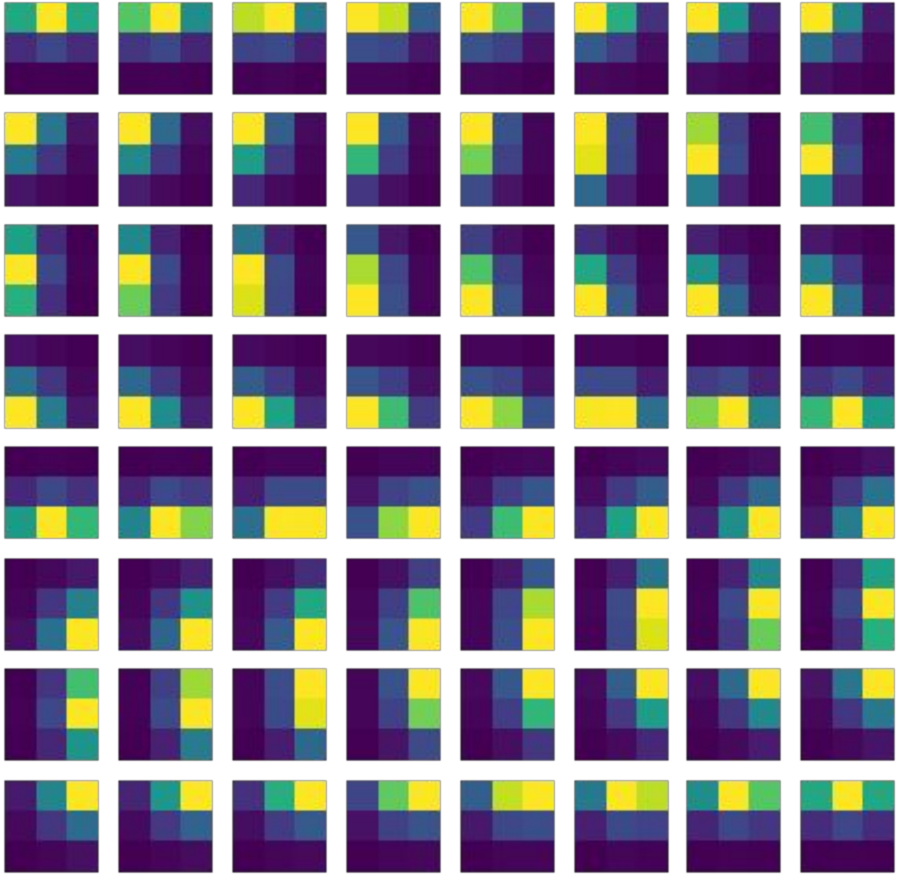


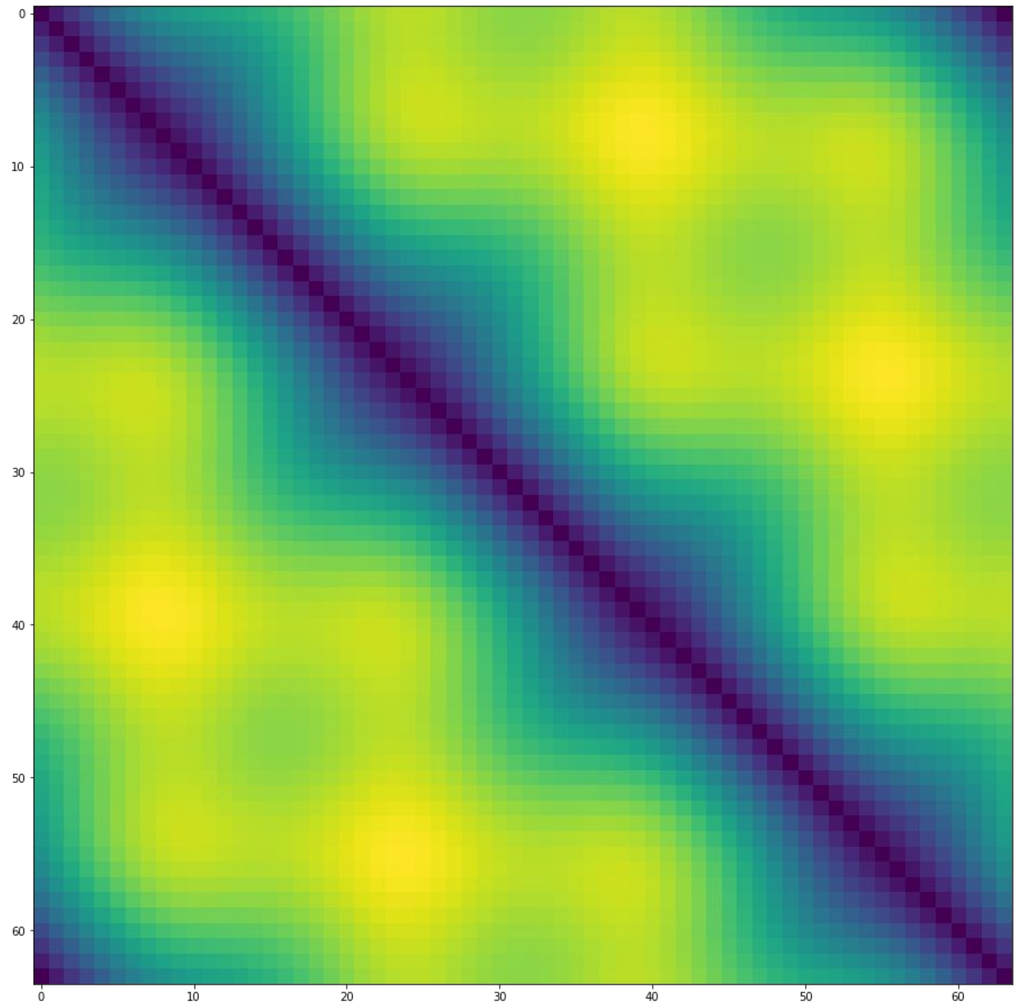


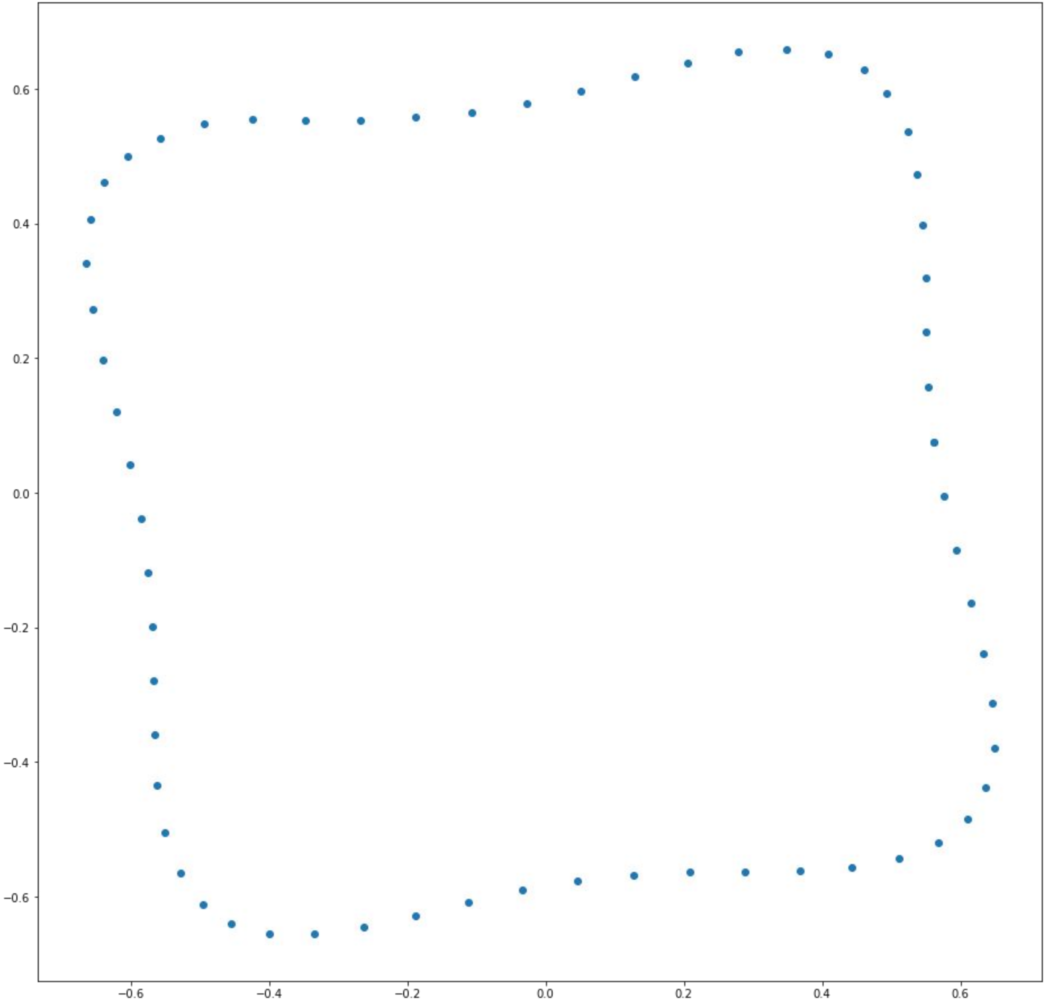
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
1.5

MEMORY ERROR!!!

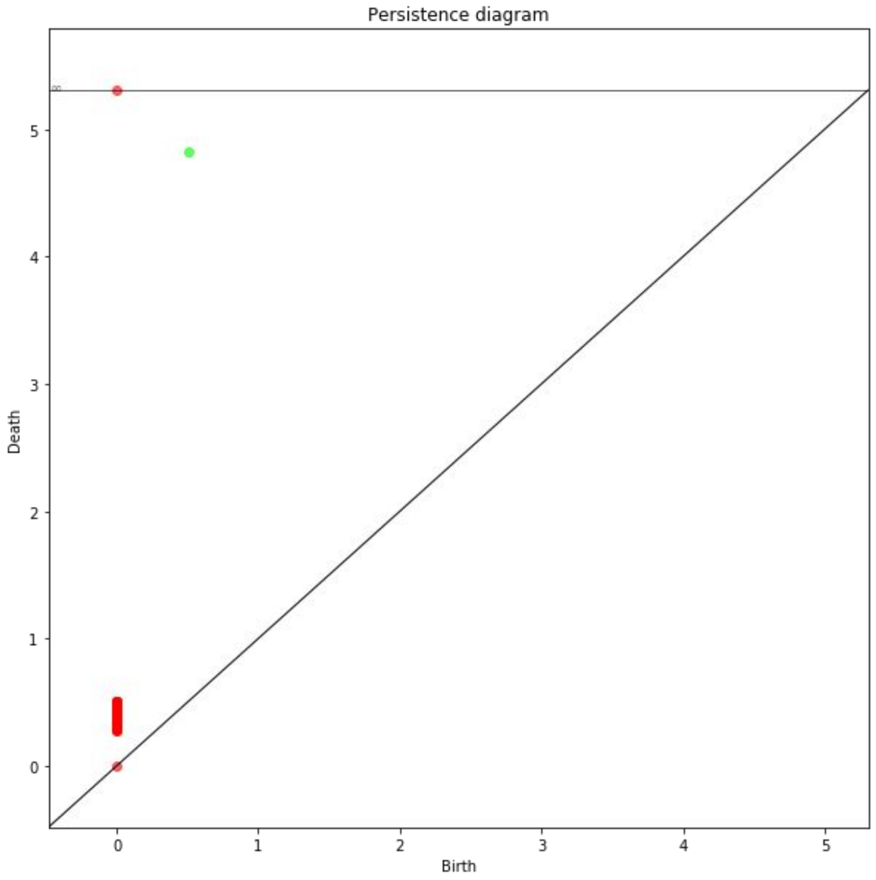
Does it make sense?



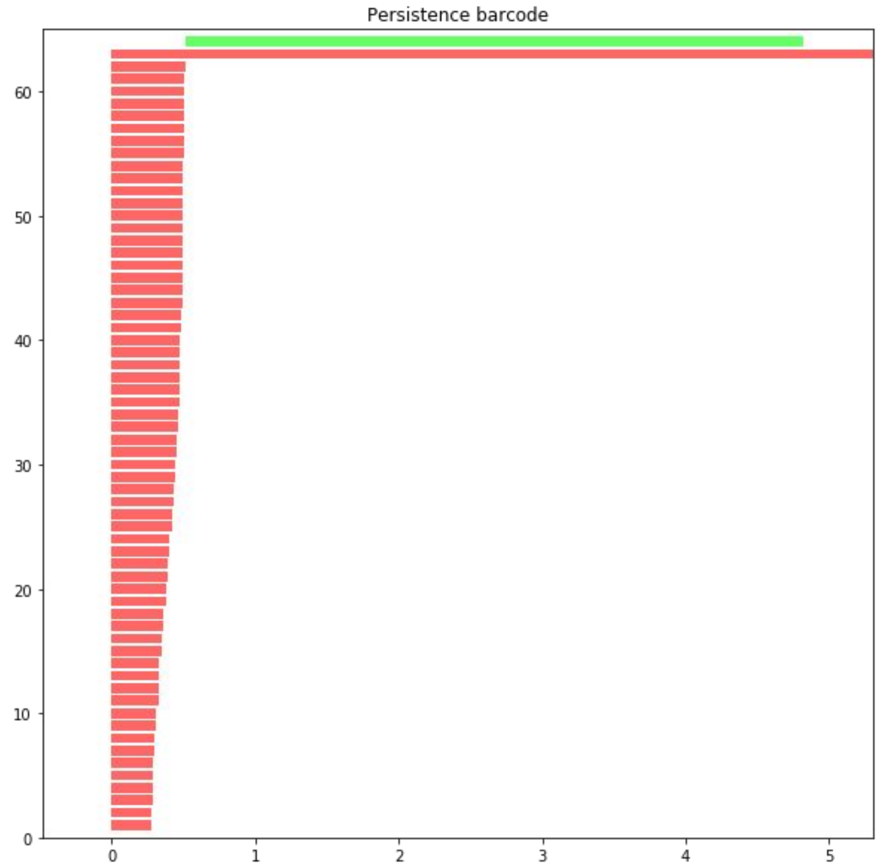

0
10
20
30
40
50
60
0
10
20
30
40
50
60



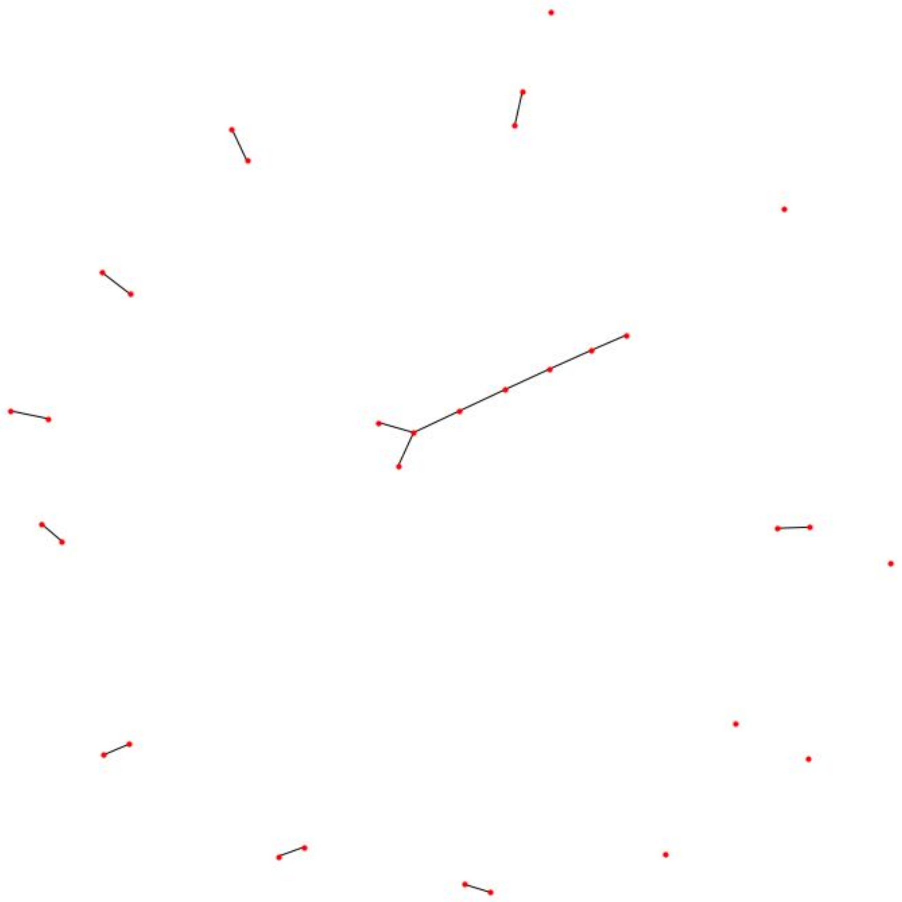
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6

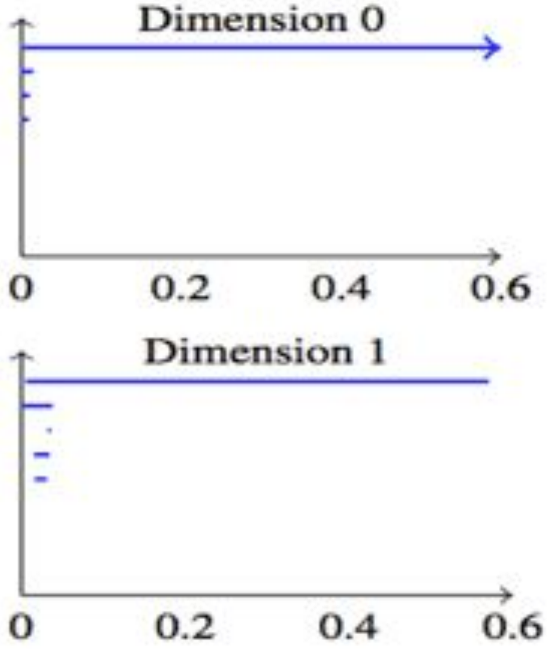


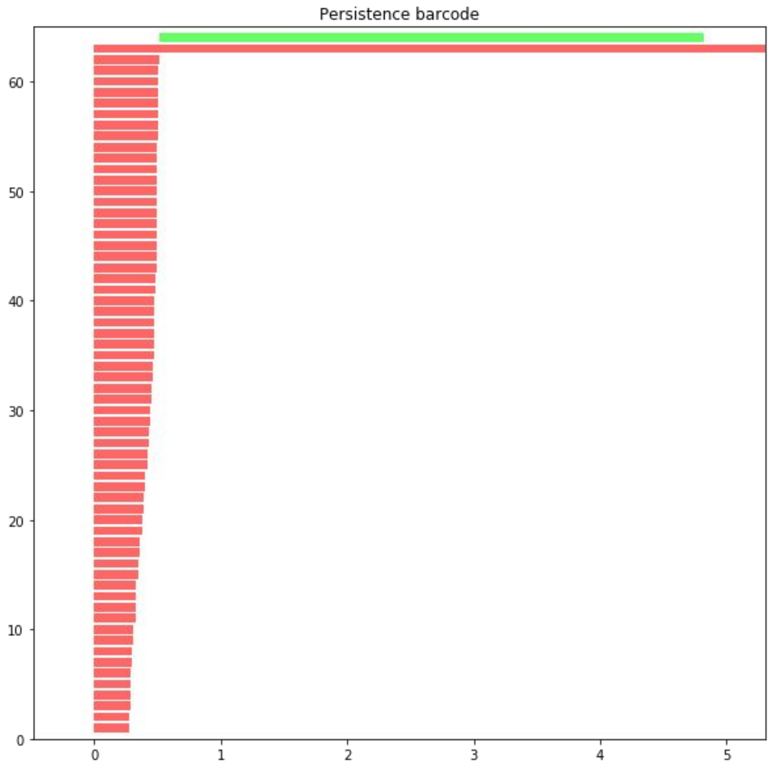
Persistence diagram
Death
Birth
∞
0
1
2
3
4
5



Persistence barcode
0
10
20
30
40
50
60
0
1
2
3
4
5




Dimension 0
0
0.2
0.4
0.6
Dimension 1
0
0.2
0.4
0.6


Persistence barcode
0
10
20
30
40
50
60
0
1
2
3
4
5

Conclusion

Challenges

- Mapper is confusing, too many parameters to tune
- Computations are very memory extensive
- Requires sophisticated preprocessing
- Toolkits are not perfect

Still very promising!

Why TDA?

- No good understanding what is happening inside Neural Networks, despite of abundance of good research done by very smart people
- Intellectually satisfying and intuitive
- Terra incognita

Questions to ask

- How topology changes over layers?
- How topology changes over training?
- Do different nets have the same underlying structures?
- What do this structures mean?
- ...

Further research

- CNNs:
    - Do they have the same structure?
    - What happens when overfit?
    - How topology of learned weights depends on topology of training data
    - ...
- RNNs:

    - What do the cycles mean?

Thank you !

Aleksei Prokopev

aleksei@akaintelligence.com

+82 10 3742 3945