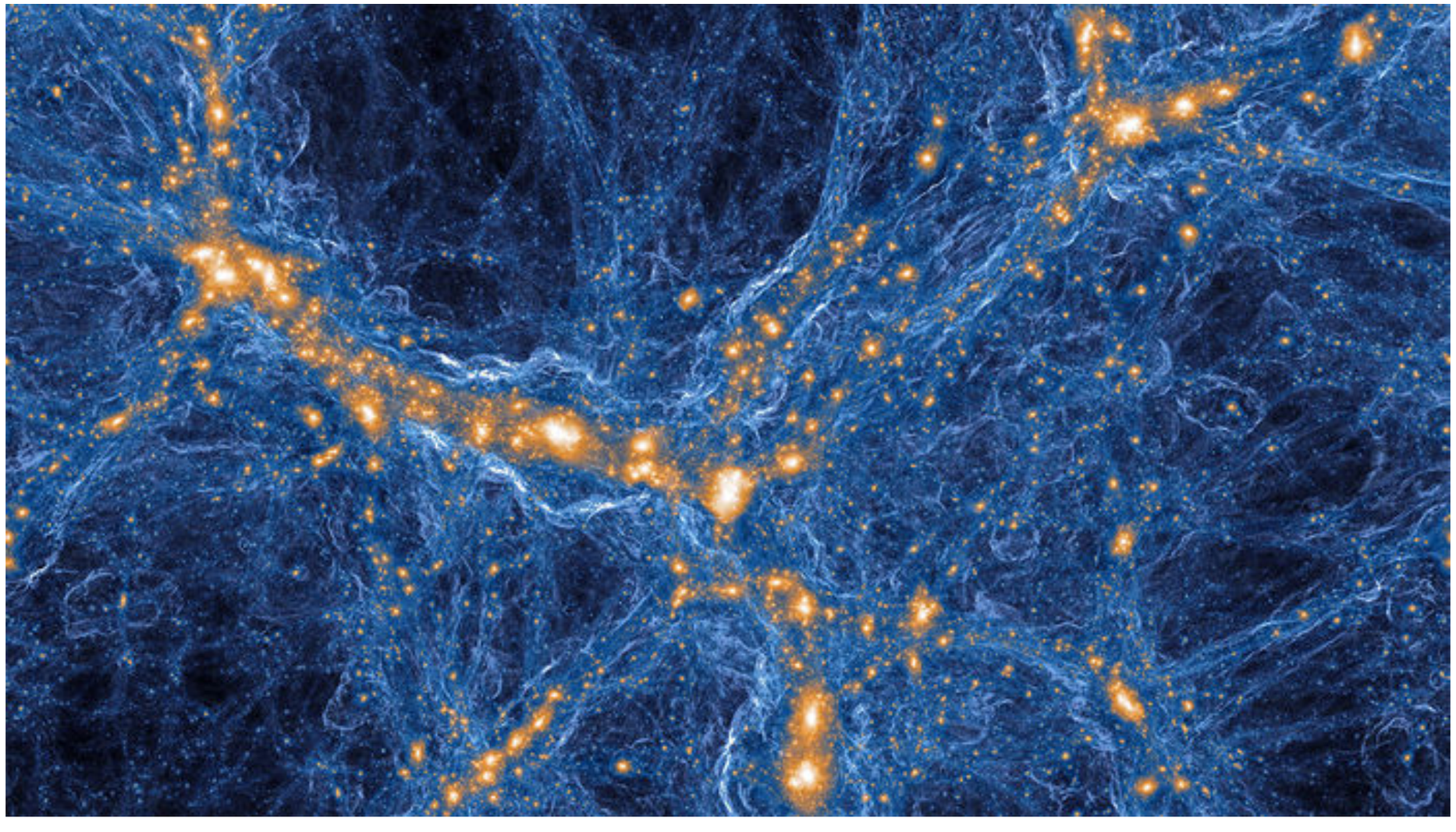
Analysing the cosmic large scale structure with higher order statistics and machine learning

Cristiano Sabiu (Yonsei University)

With Xiao-Dong Li, Shuyang Pan, Miaoxin Liu, Jaime Forero-Romero, Ben Hoyle, Juhan Kim ++



Machine Learning in Astronomy Seminar Series
Western Sydney University - 28/04/2020

Contents

- ★ **Background Cosmology**
- ★ **Big Data Analytics and Graph Databases**
- ★ **Convolutional Neural Networks (CNN)**
- ★ **Conclusions**

Background

* The goal of modern cosmology is to **understand the physics** that governs our Universe on the largest scales

* Figure out the **constituents of the Universe**

$$R_{\mu\nu} - \frac{1}{2}g_{\mu\nu}R - \Lambda g_{\mu\nu} = 8\pi G T_{\mu\nu}$$

Background

* The goal of modern cosmology is to **understand the physics** that governs our Universe on the largest scales

* Figure out the **constituents of the Universe**

$$R_{\mu\nu} - \frac{1}{2} g_{\mu\nu} R - \Lambda g_{\mu\nu} = 8\pi G T_{\mu\nu}$$

The game we play…

1) We start with Einstein's GR
2) Plug in a homogeneous/isotropic metric
3) Plug in energy/matter components
4) Obtain evolution equations for:
   - **Expansion of the Universe**
   - **Growth of density perturbations**

Background

* What causes cosmic acceleration?
  - Vacuum energy or Scalar field(s)?
  - Or something more strange*?!
    * (if that's not strange enough)

* What is Dark Matter?
  - Is it Self-interacting?
  - Is it Decaying?

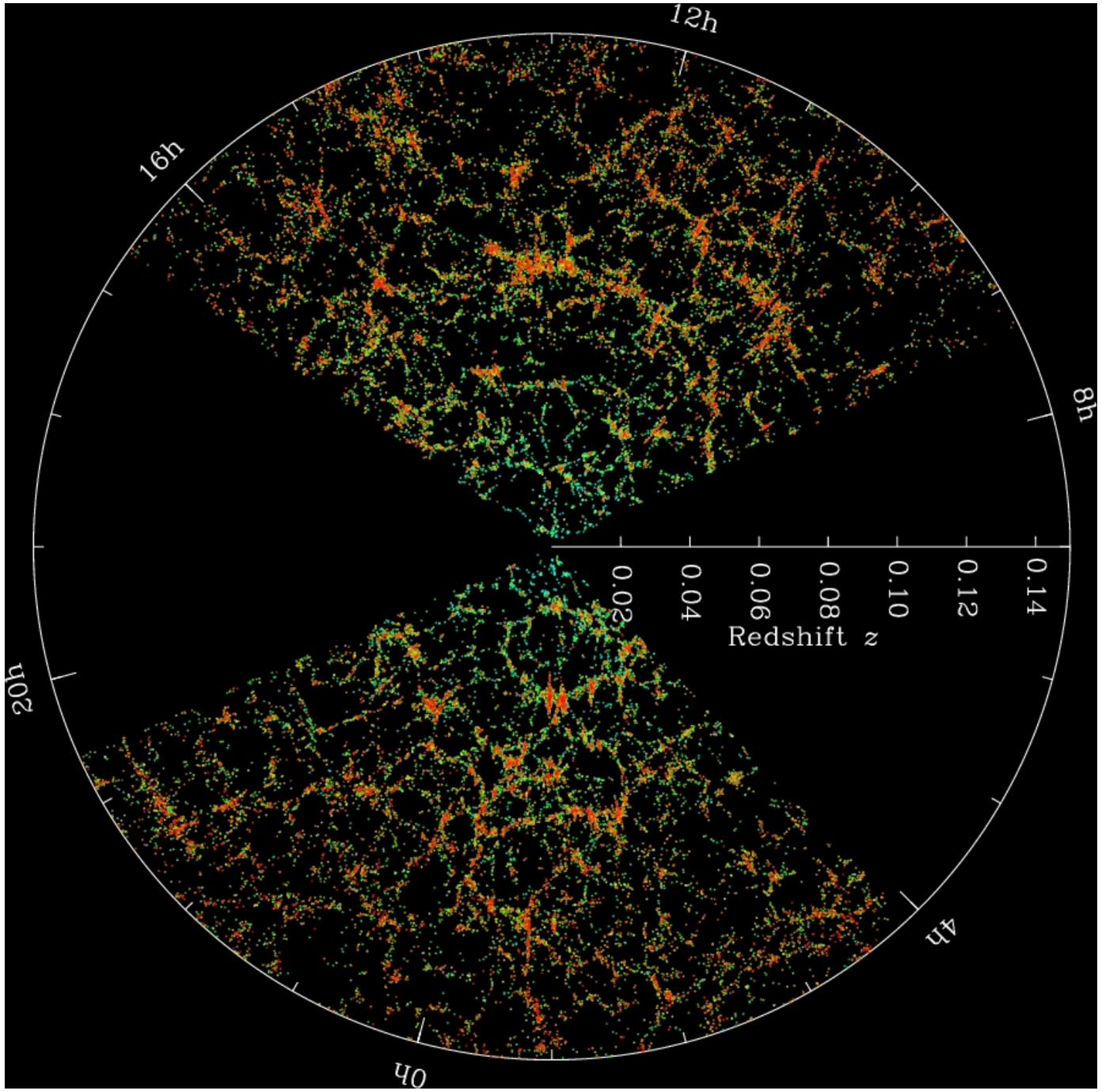
Each of the above possibilities could effect
- Universe expansion
- Clustering of matter

Background

We can now map large volumes of the Universe in 3D using galaxies as tracers of the underlying matter potential

But then what do we do with all these galaxy positions?

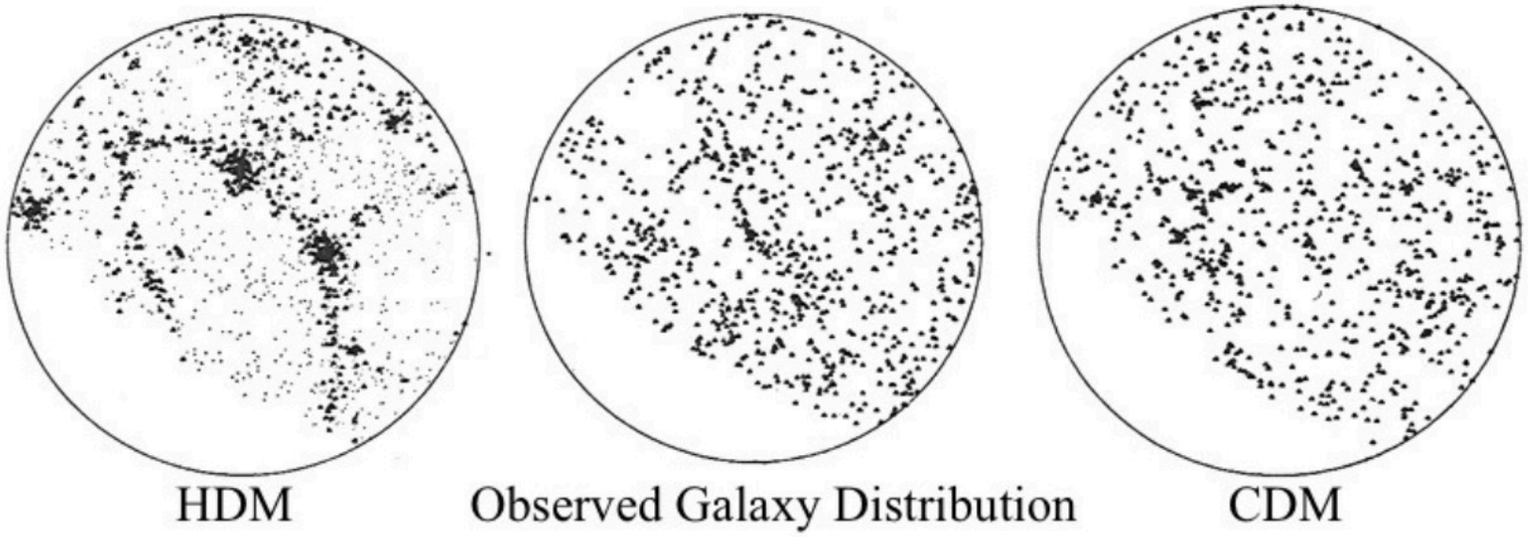
SDSS



Background

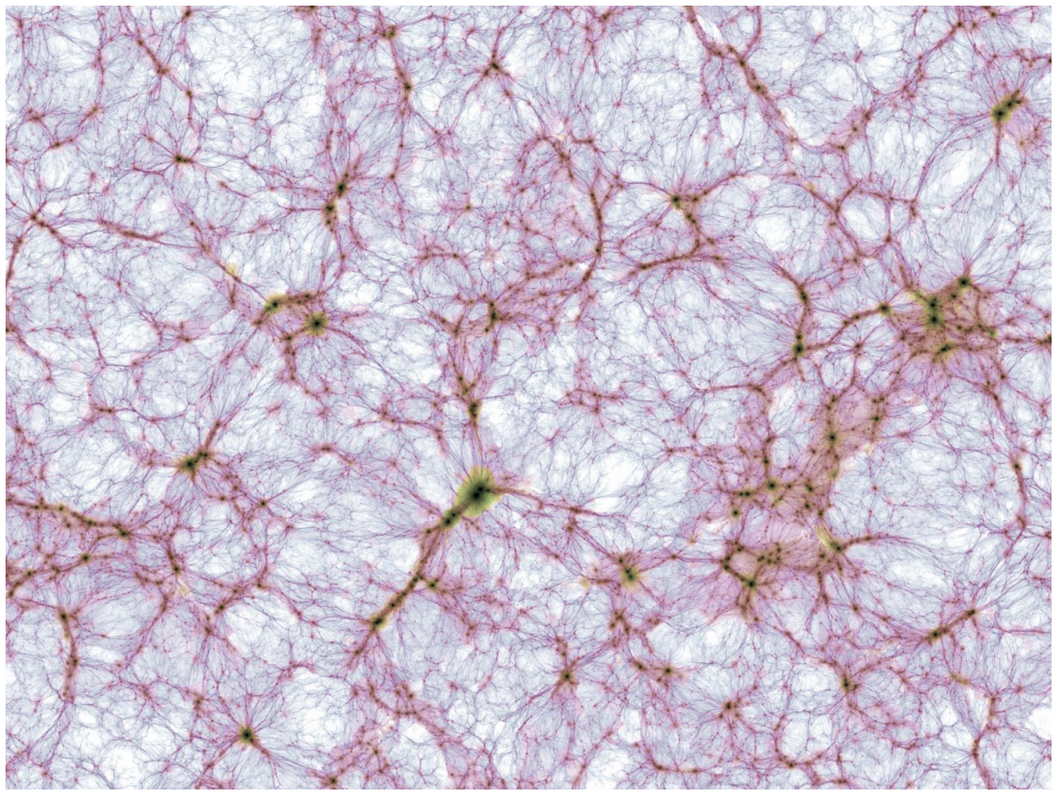
N-body Simulations

1980s
~1000 particles



Credit: S. D. M. White, 1986

Today
~10,000,000,000 particles

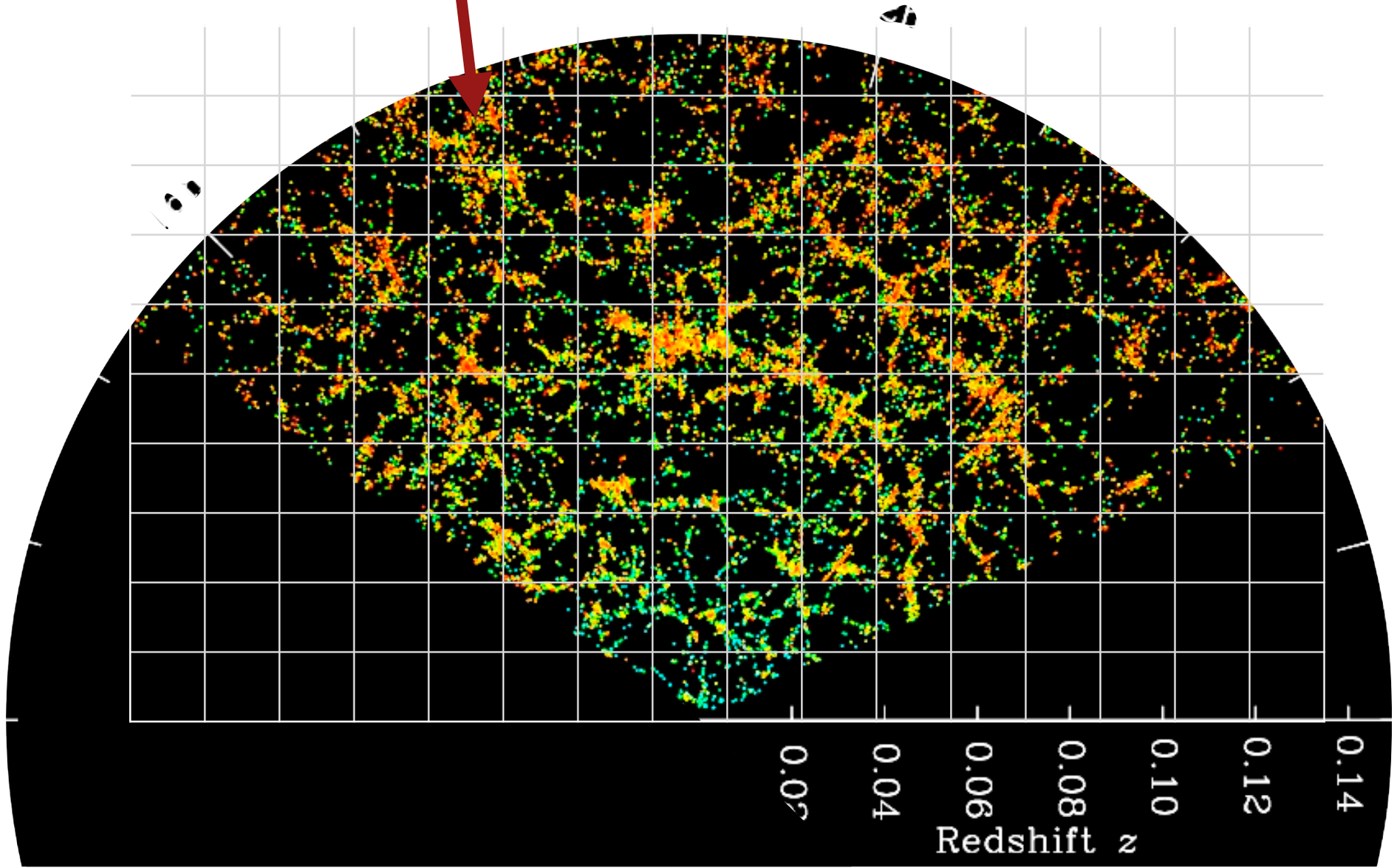


Illustris TNG Simulation

Background

Count galaxies in cells and compute:

$$\xi(r) = \langle \delta(x)\delta(x+r) \rangle_x$$



SDSS

Background

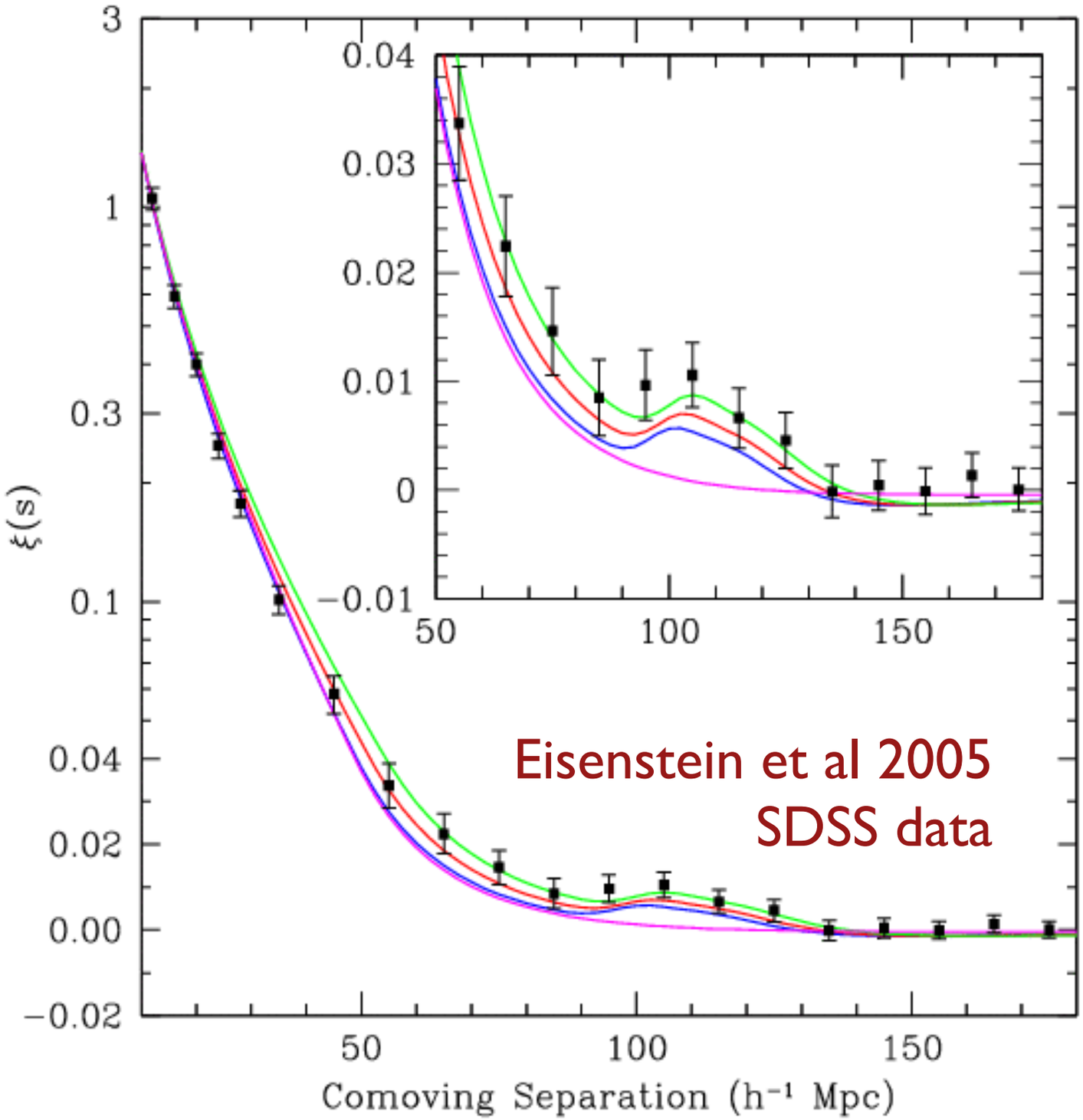
The measured 2-point correlation function (2PCF)

Large clustering strength at small separations

Almost zero on scales > 200Mpc/h

We can compare observation and theory/ simulations with the 2PCF

Recently some groups have used ML…



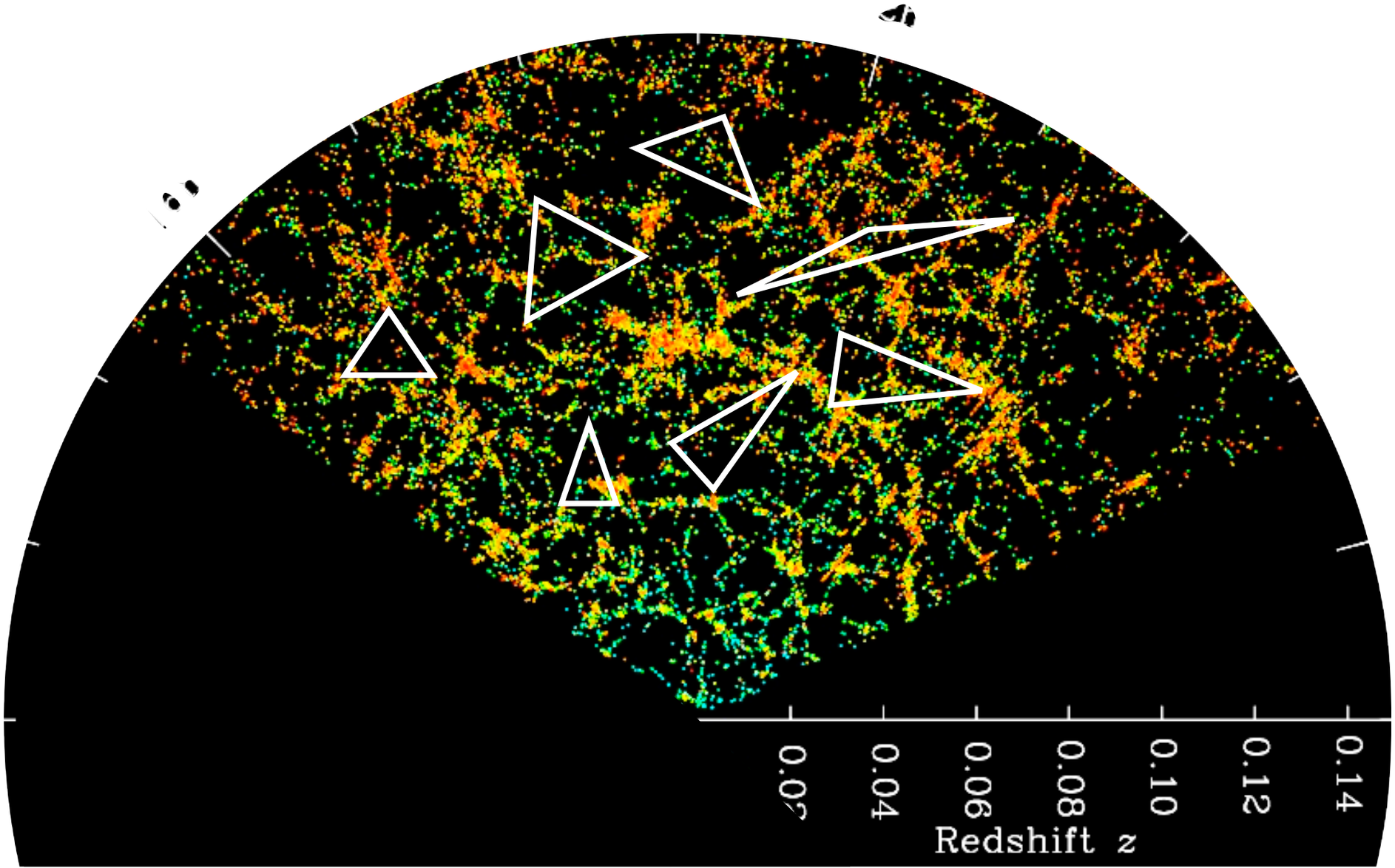
Background

We can try to measure higher order statistics

$$\zeta(r_1, r_2, r_3) = \left\langle \delta_1 \delta_2 \delta_3 \right\rangle$$

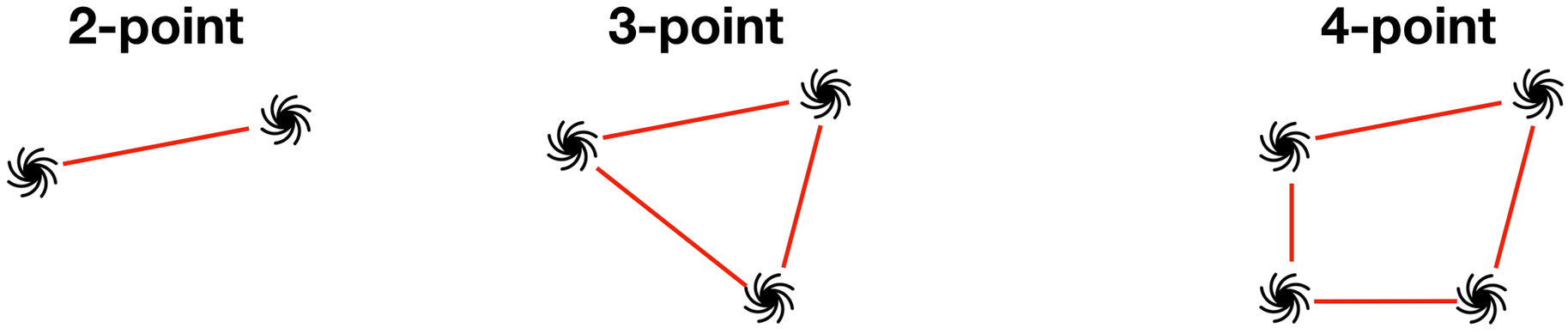
Naively:

- **Counting Pairs requires N^2 calculations**
- **Counting Triplets requires N^3 calculations**



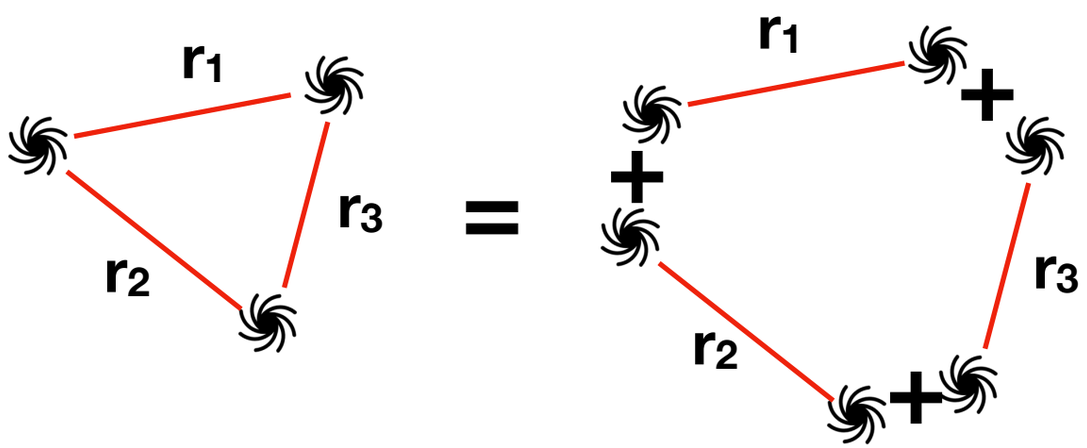
SDSS

Graph Database solution for Galaxy Clustering statistics



- **The computational expensive part of measuring higher order statistics is in making sure the distances between data points satisfy our specific criteria i.e. r_1, r_2, r_3, etc**

- **However we notice that all higher order statistics are just complex combinations 2-point statistics**



So rather than treat data positions as the important feature, rather treat each data pair (relationship) as the main information!

Graph Databases

A graph within graph databases is based on graph theory.

- **<u>Nodes</u> represent entities or instances such as people, businesses, accounts, or any other item to be tracked. They are roughly the equivalent of the record, relation or row in a relational database, or MySQL item**

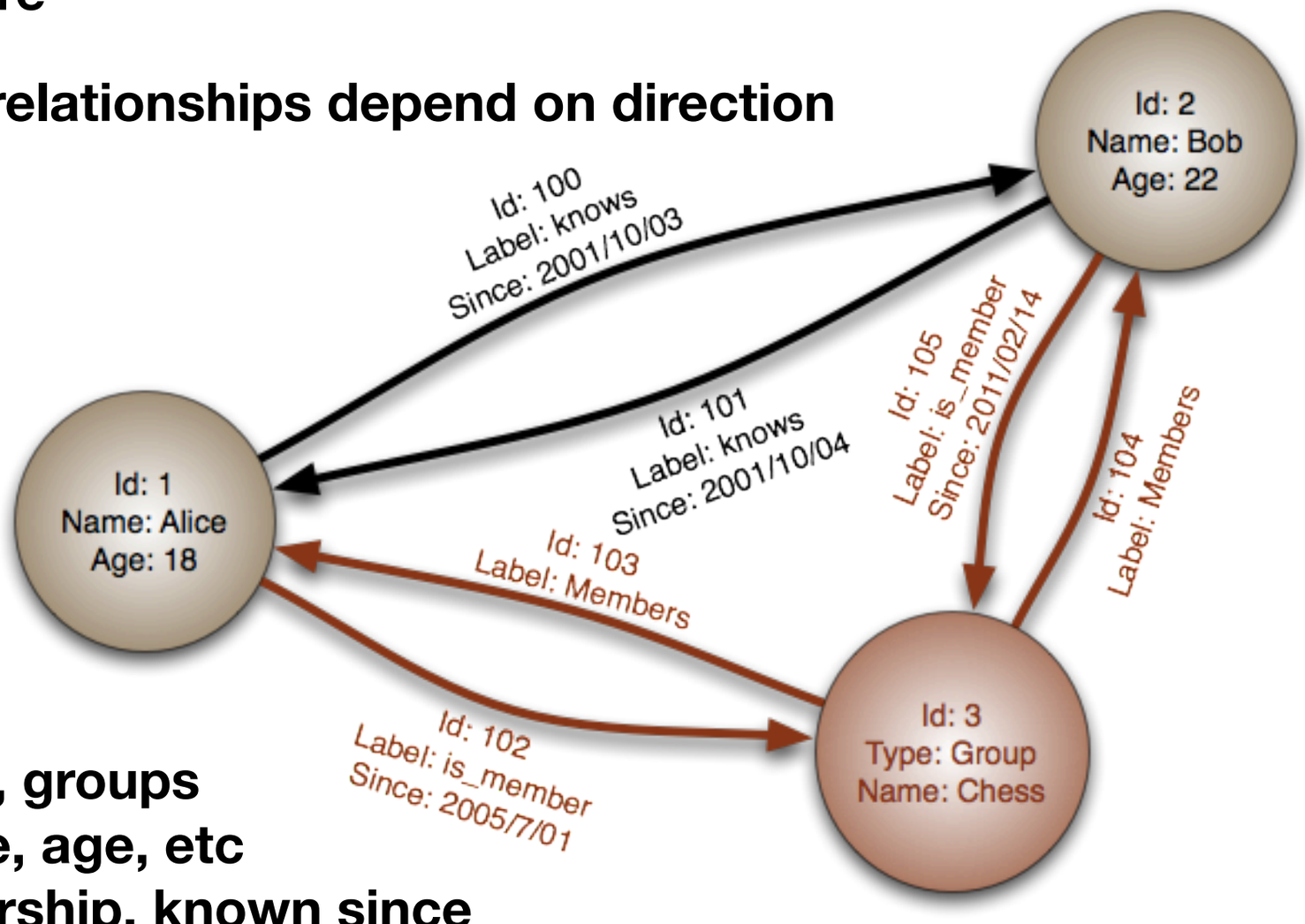
- **<u>Relationships</u>, are the lines that connect nodes to other nodes; representing the relationship between them. Meaningful patterns emerge when examining the connections and interconnections of nodes, properties and relationships. Relationships are the key concept in graph databases, representing an abstraction that is not directly implemented in a relational model like SQL**

- **<u>Properties</u> are germane information to nodes. For example, if *Wikipedia* were one of the nodes, it might be tied to properties such as website, reference material, or words that starts with the letter *w*, depending on which aspects of Wikipedia are germane to a given database.**

Graph Databases

In 2009 Facebook gave up using their MySQL storage and moved to a graph structure

A directed graph: relationships depend on direction



Nodes: people, places, groups
Node properties: name, age, etc
Relationships: membership, known since

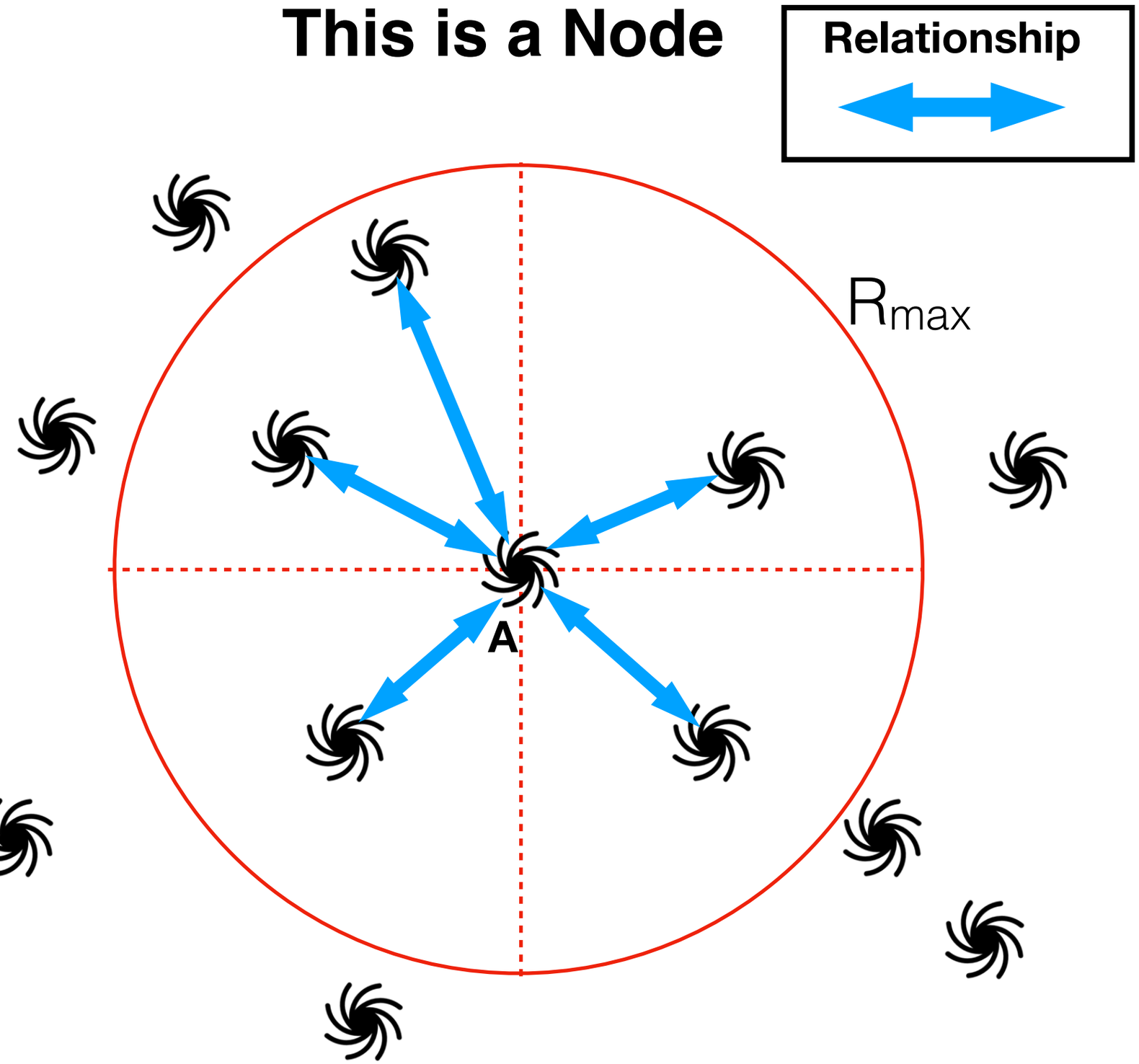
Graph for our purposes will be much simpler!

Graph Database solution for Galaxy Clustering statistics

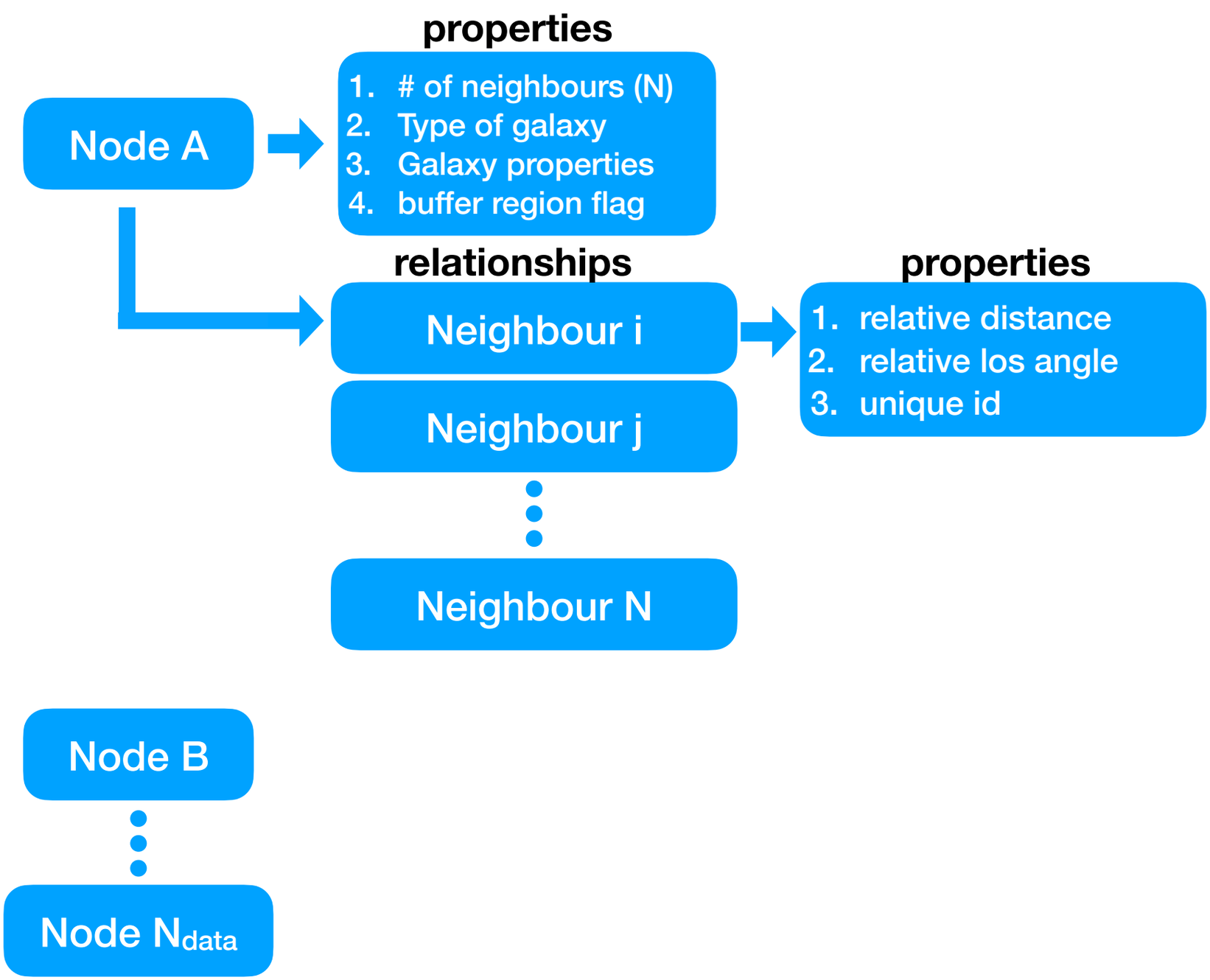
Each galaxy (or random) point is a **node** which may have **relationships** to there nodes

For our purposes the relationship information is the distance to neighbours within R_{max}

Neighbour list can be obtained quickly using a **kd-tree** algorithm

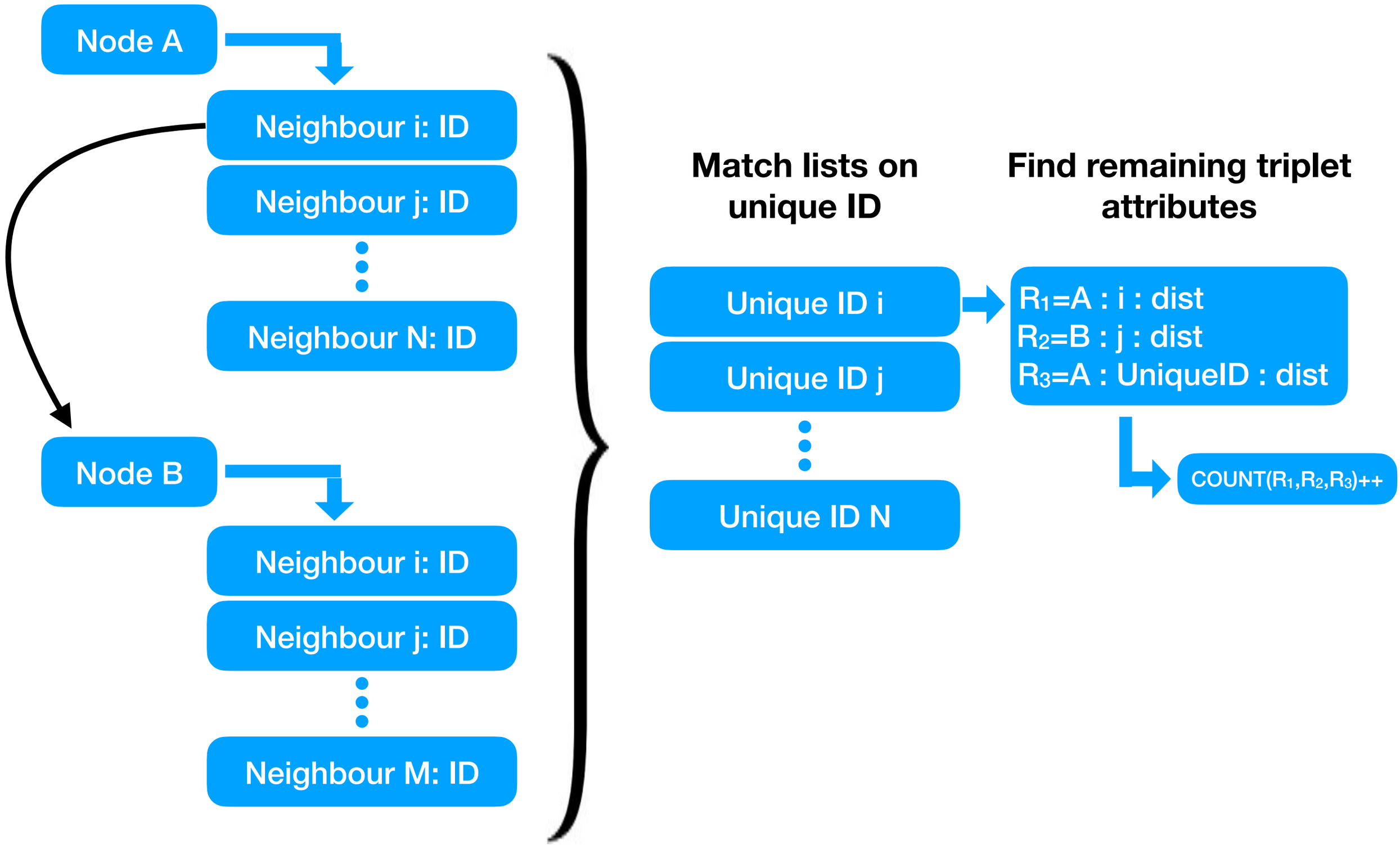


Graph Database solution for Galaxy Clustering statistics



Graph Database solution for Galaxy Clustering statistics

Query Graph: 3PCF



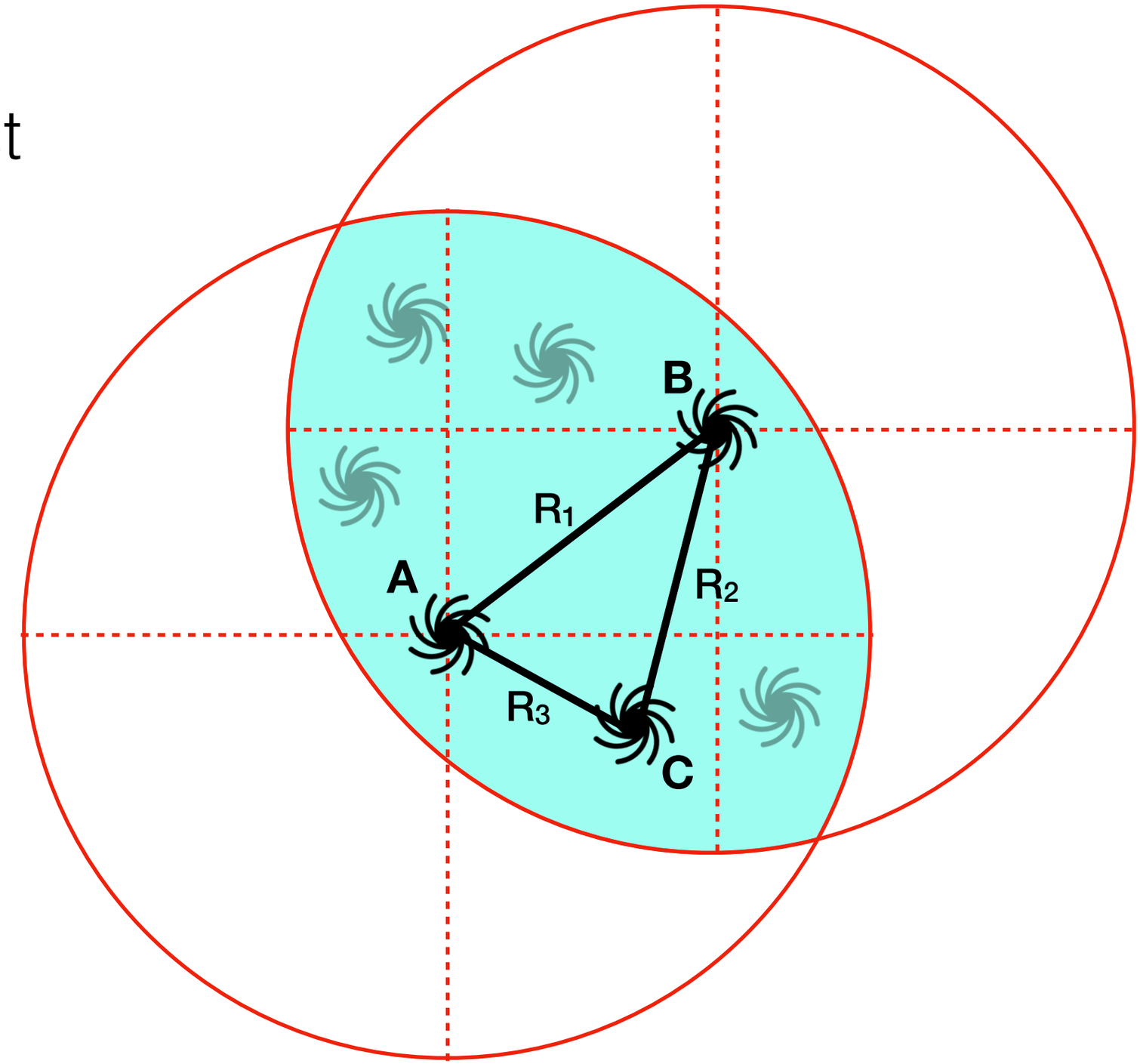
Graph Database solution for Galaxy Clustering statistics

Query Graph: 3PCF

The intersection of two sets (**A**,**B**) contains a list of points, **C**, that will complete the triangle A,B,C such that,
$0<R_1,R_2,R_3<R_{max}$

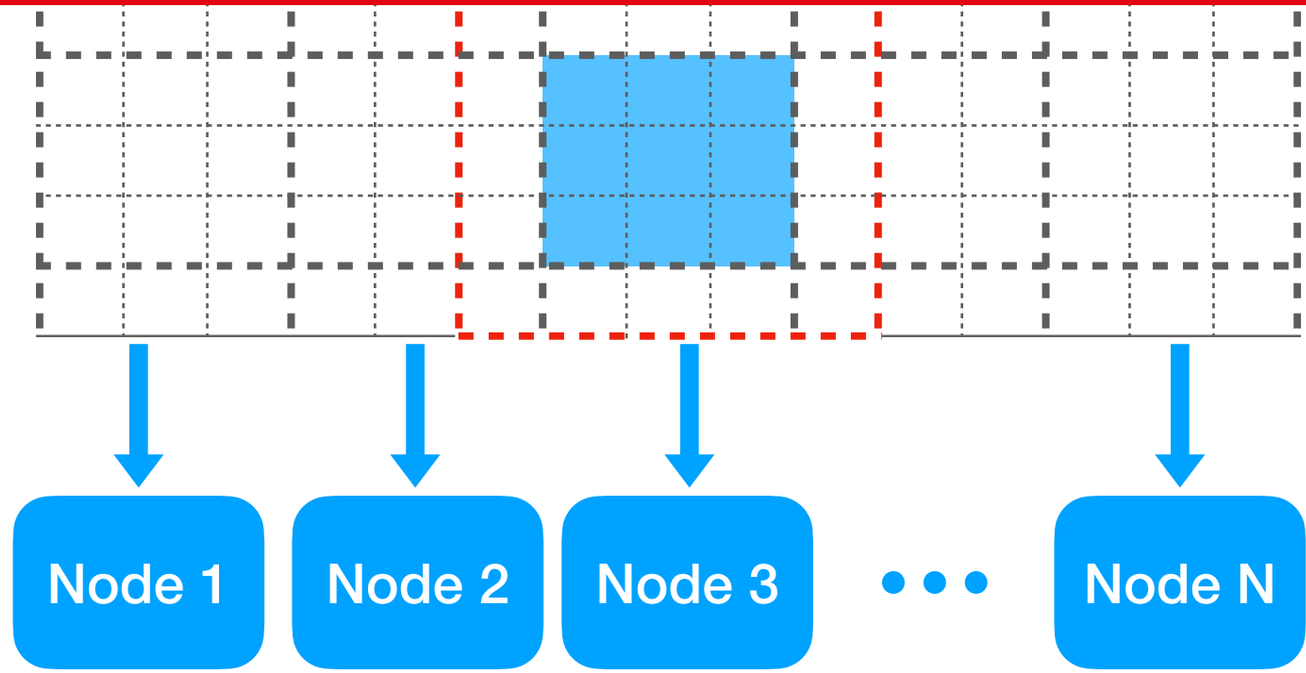
The intersection must be computed ~ N^2 times!

For 2 **ordered lists**, the intersection can be computed very quickly!



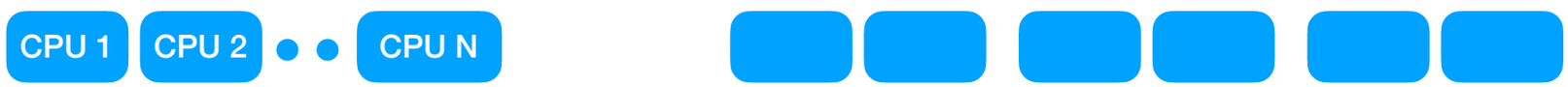
Graph Database solution for Galaxy Clustering statistics

Domain decomposition is precomputed



Spawn *N* MPI processes

OpenMP threading

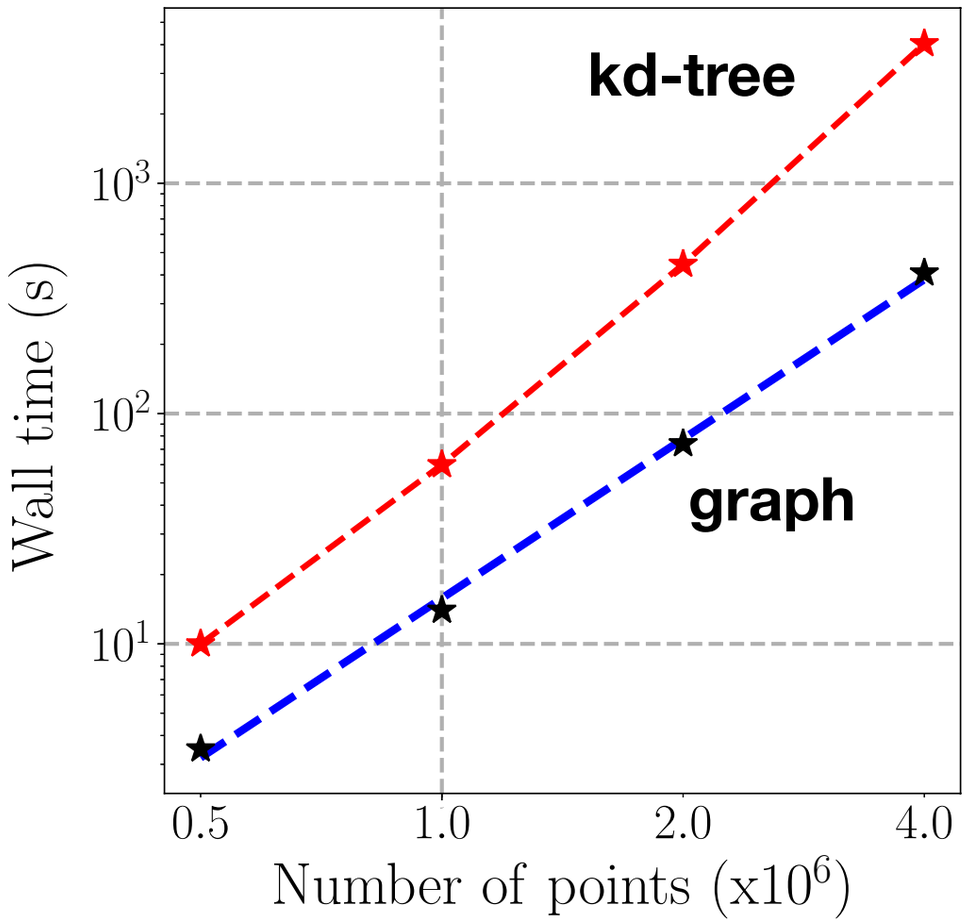


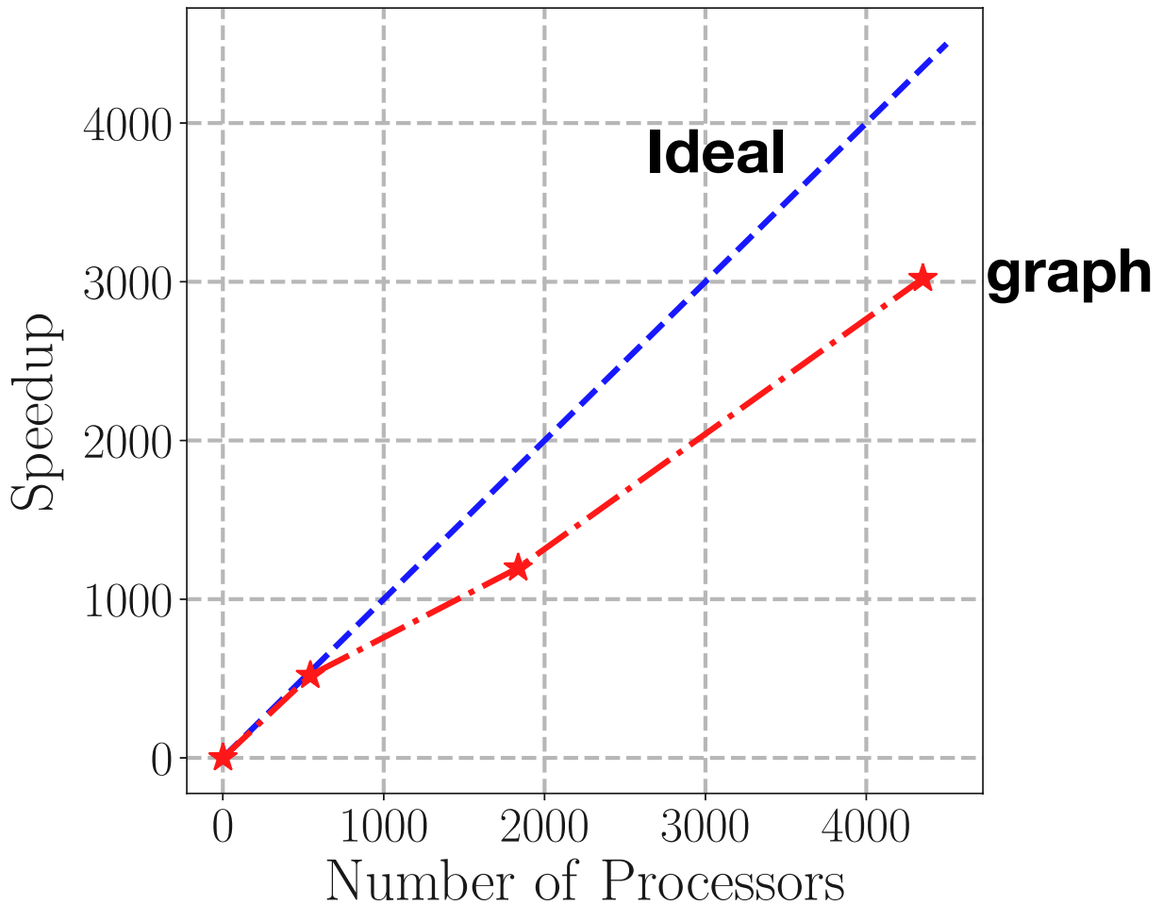
Reduce OMP arrays for each node/domain

Collect MPI arrays and output

Graph Database solution for Galaxy Clustering statistics

Benchmarking



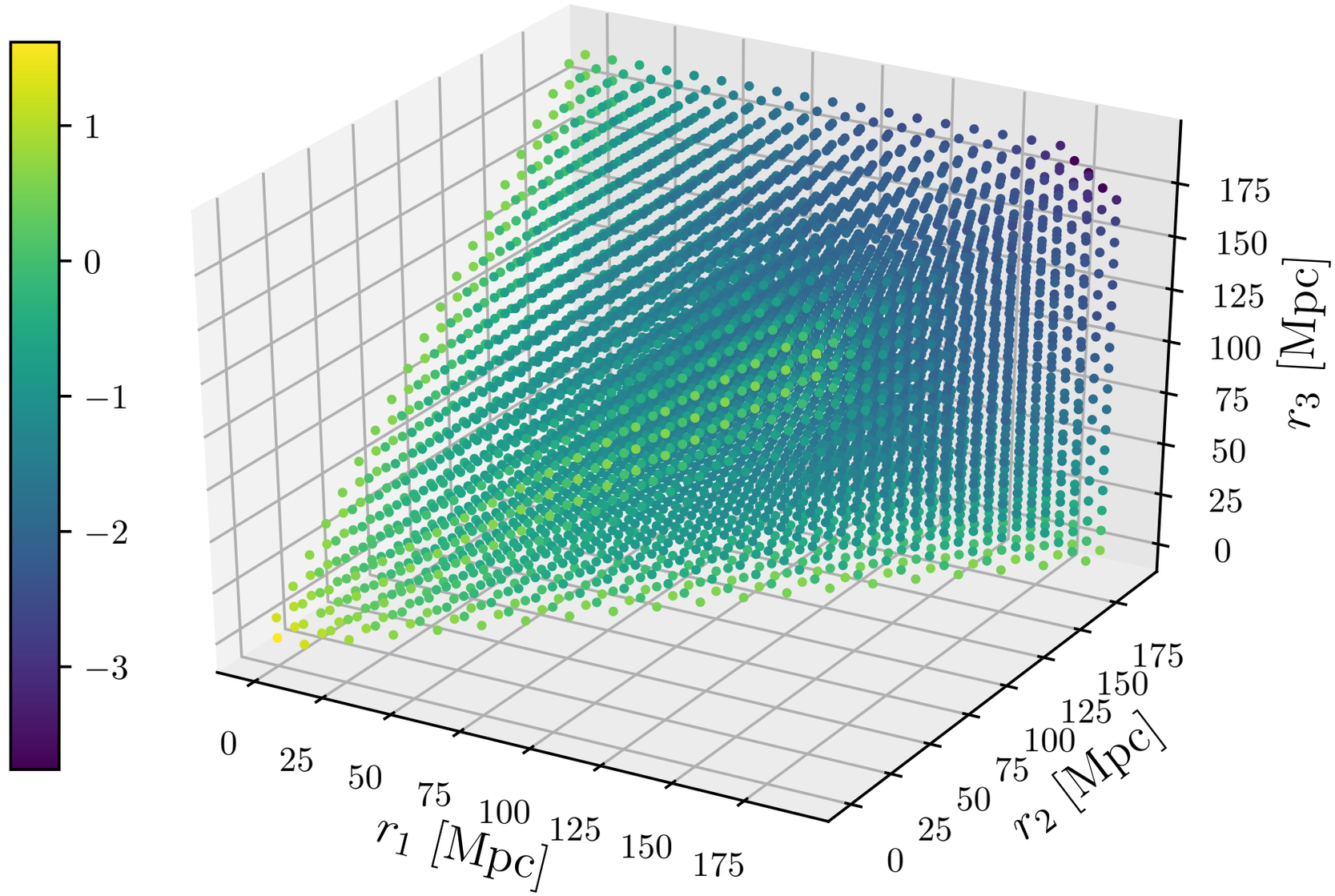


★C. Sabiu, B. Hoyle, J. Kim, X-D Li

★https://arxiv.org/abs/1901.00296

Graph Database solution for Galaxy Clustering statistics

Taking the SDSS CMASS DR12 galaxies (~1M)



★C. Sabiu, B. Hoyle, J. Kim, X-D Li

★https://arxiv.org/abs/1901.00296

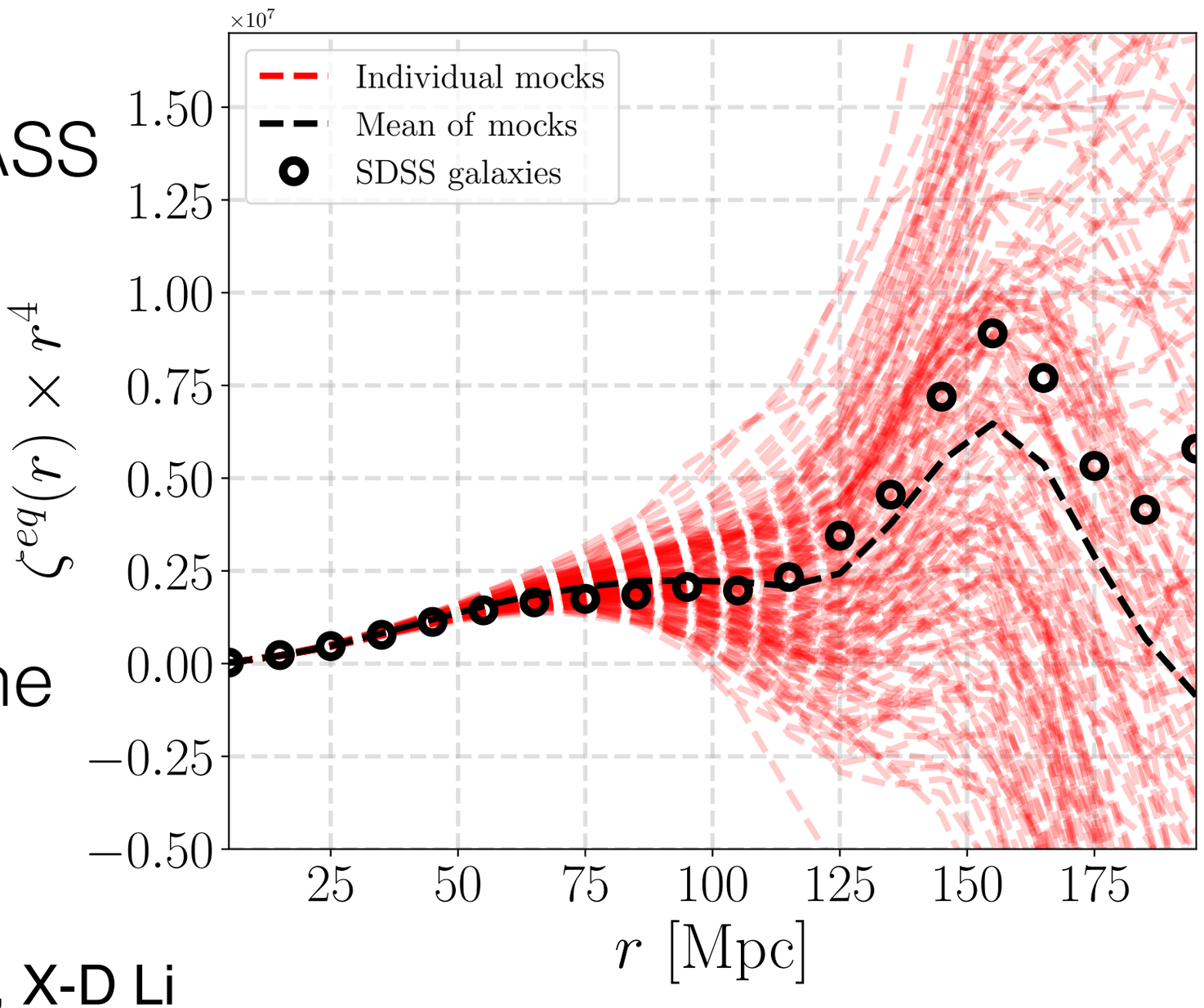
Graph Database solution for Galaxy Clustering statistics

BAO in the 3-point function

Taking the SDSS CMASS DR12 galaxies (~1M)

We look at equilateral triangles

We see evidence of the BAO peak

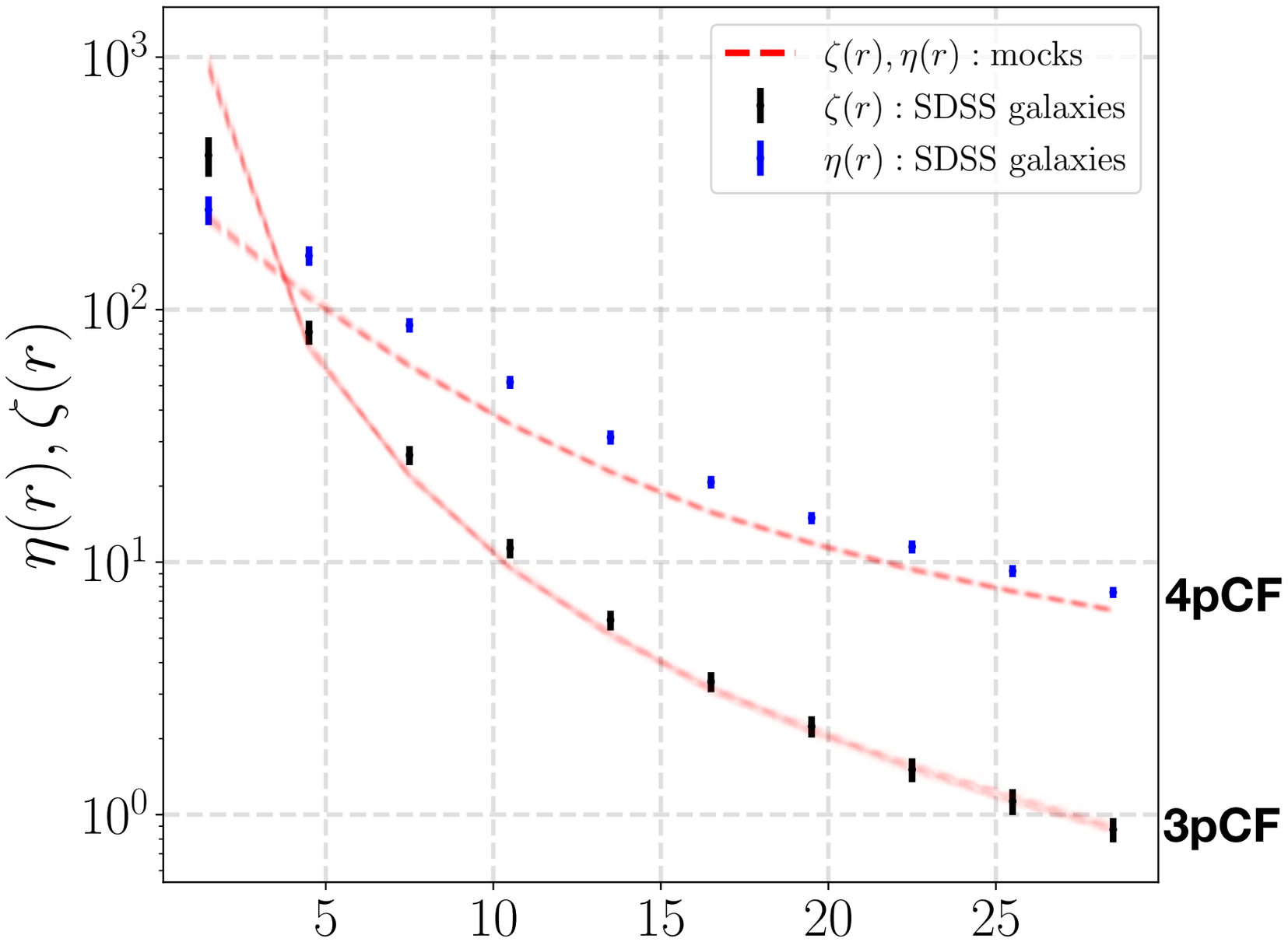


Isotropic, equilateral 3PCF

★C. Sabiu, B. Hoyle, J. Kim, X-D Li
★https://arxiv.org/abs/1901.00296

Graph Database solution for Galaxy Clustering statistics

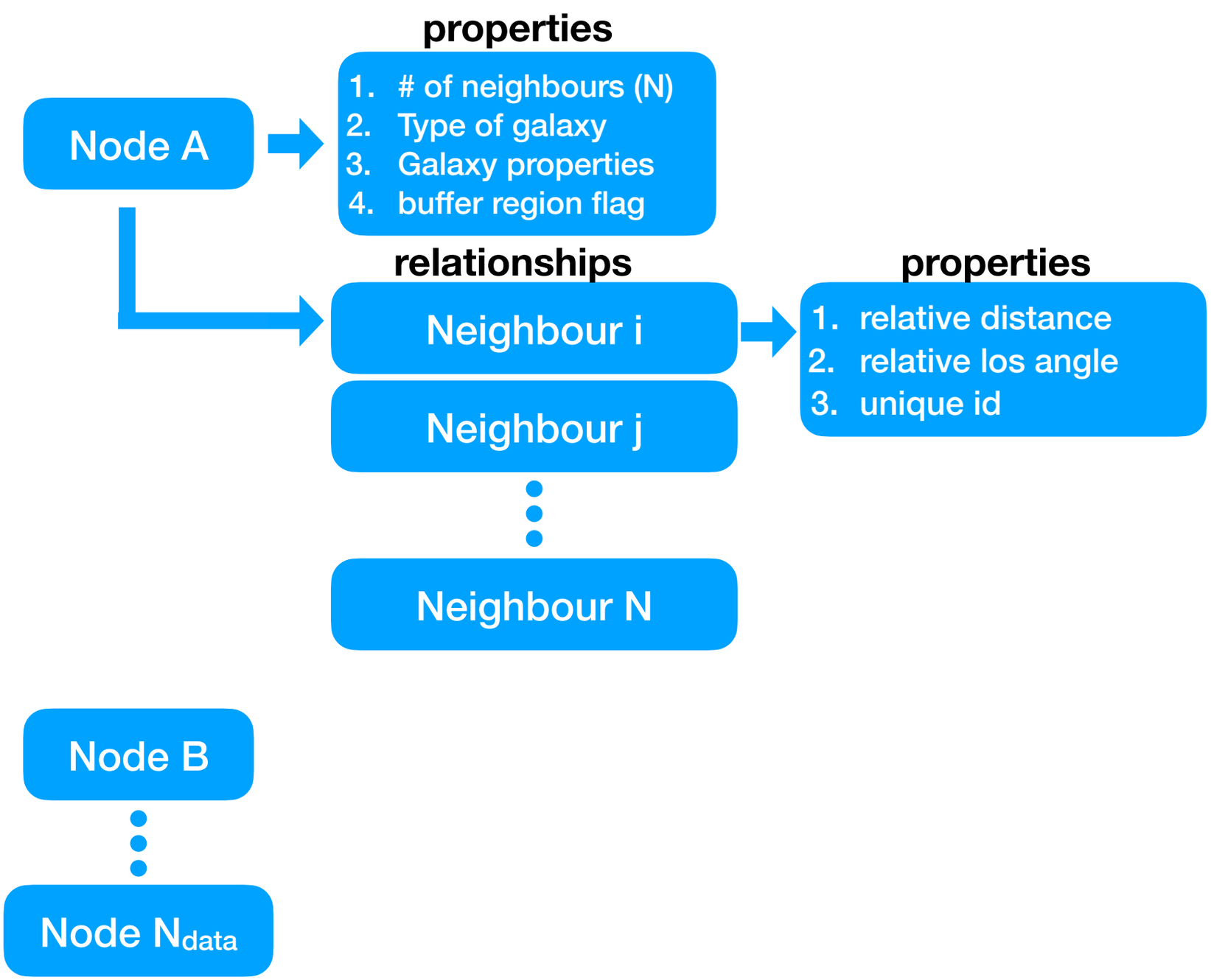
4-point correlation function



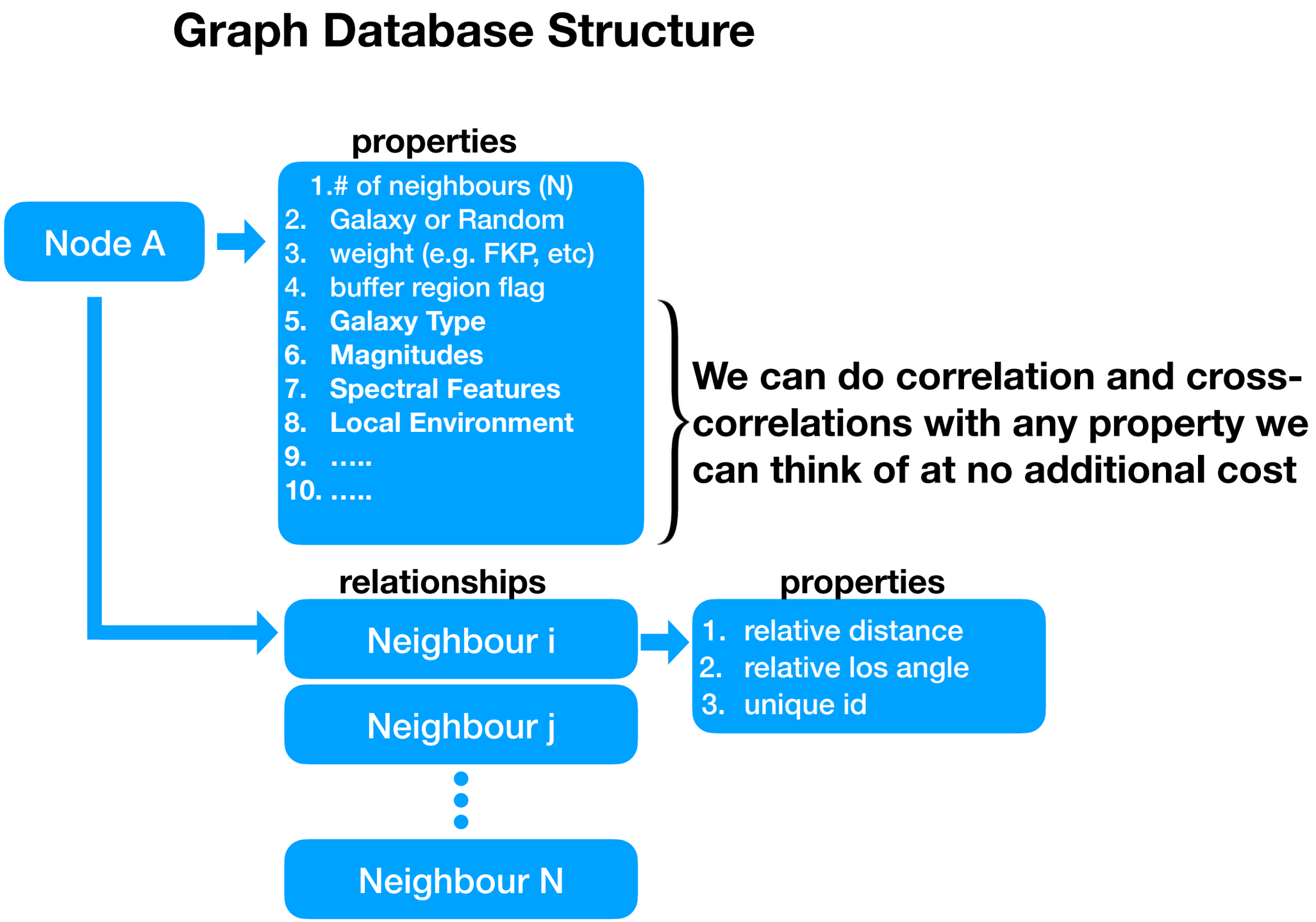
★C. Sabiu, B. Hoyle, J. Kim, X-D Li

★https://arxiv.org/abs/1901.00296

Graph Database for exploring new quantities in galaxy evolution, cosmology



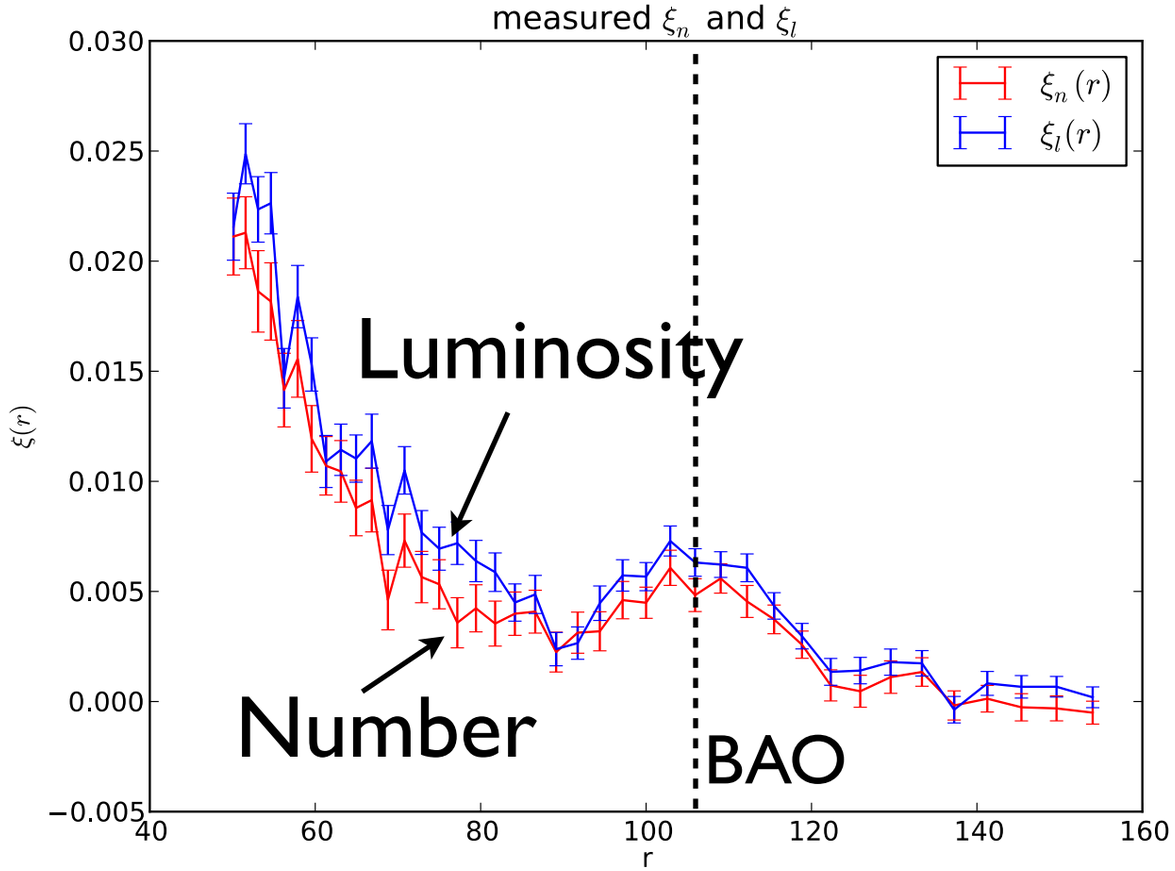
Graph Database for exploring new quantities in galaxy evolution, cosmology

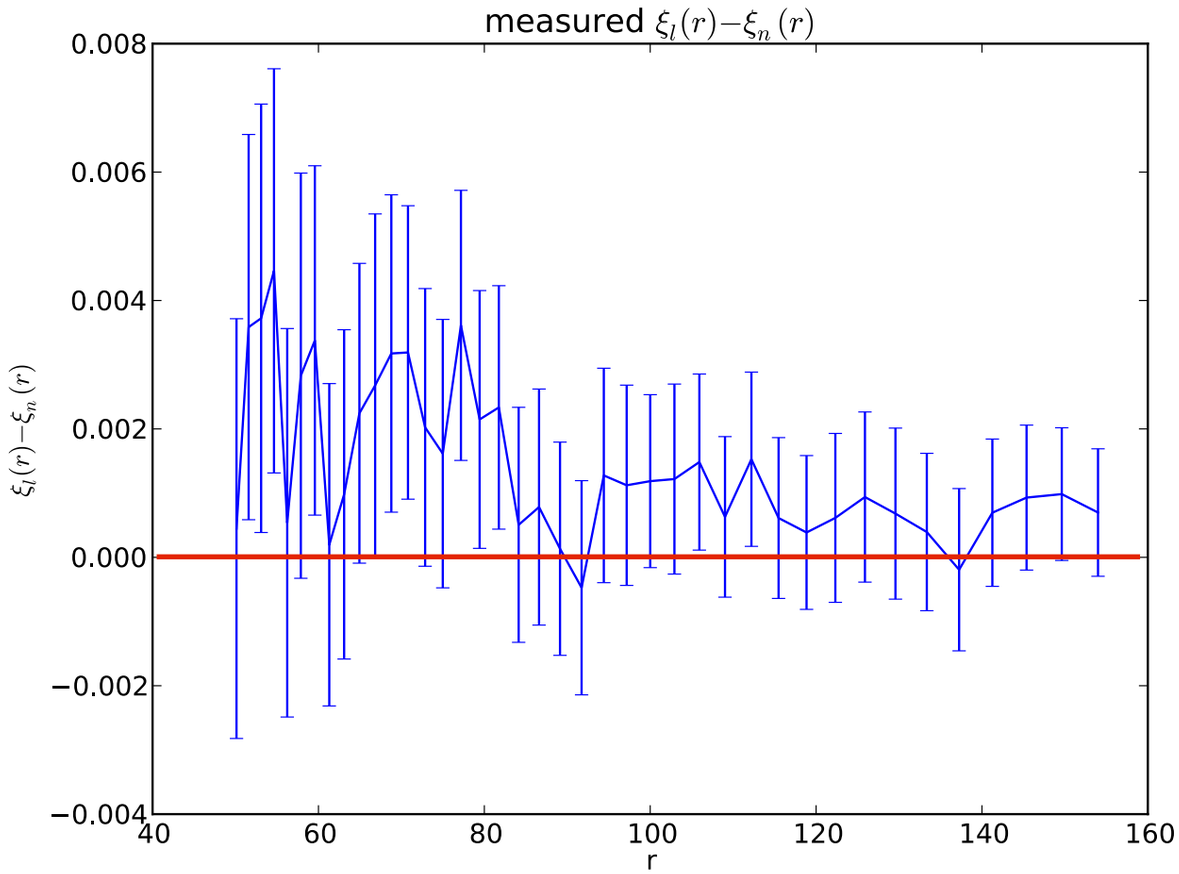


Graph Database for exploring new quantities in galaxy evolution, cosmology

Example: lets look at counting data pairs while weighting by galaxy luminosity

- **Query the database for all relationships where $r_{min}<r<r_{max}$**
- **Count that pair weighted by its absolute luminosity**
- **Compare with unclustered random points**
- **Compute the usual Landy-Szalay estimator for the 2pCF**
- **But what does it mean? What information does it contain?**





Graph Database for exploring new quantities in galaxy evolution, cosmology

Following the theoretical work of Barkana & Loeb 2010

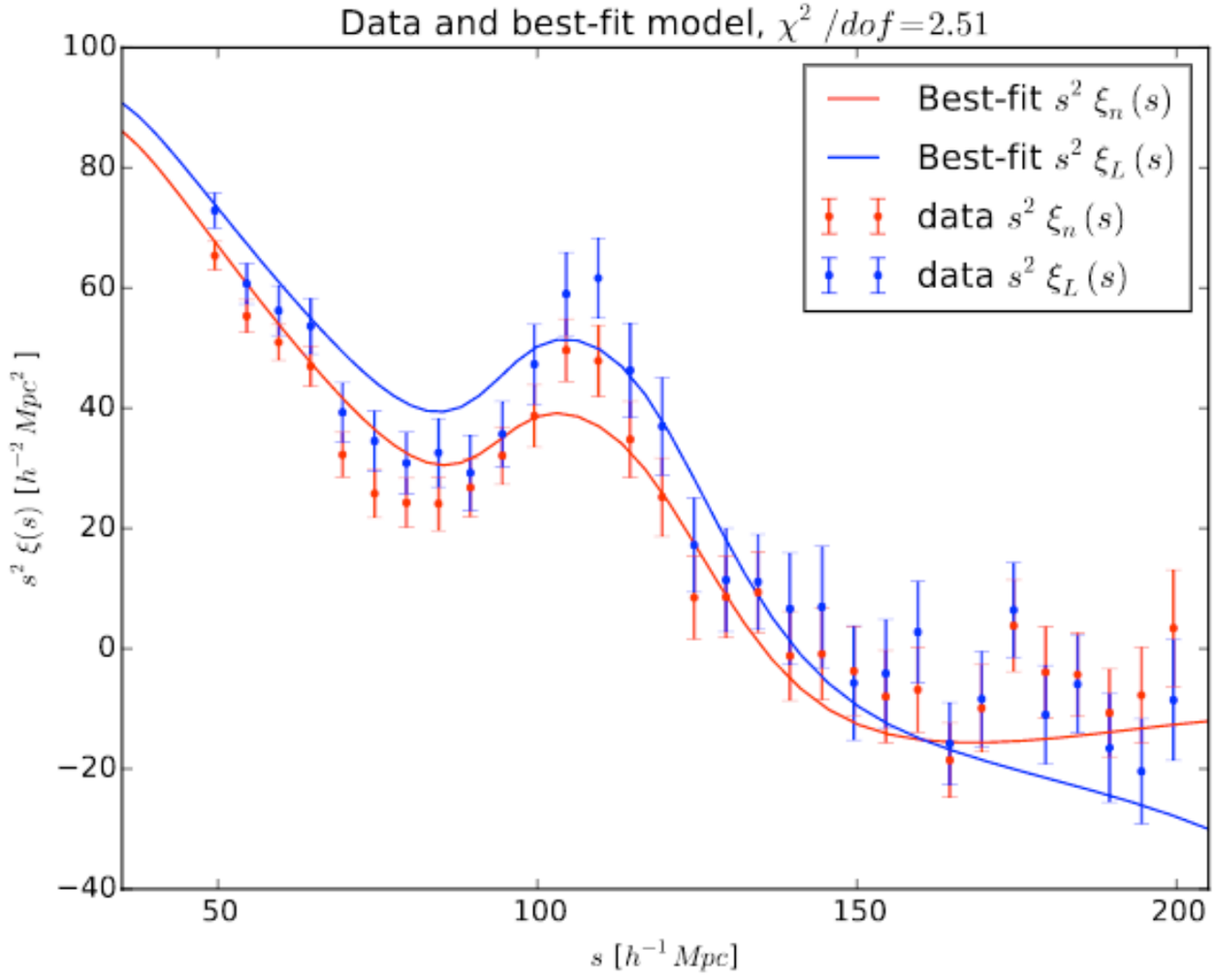
We develop a model for the luminosity weighted correlation function of galaxies:

$$\xi_L = B_{L,t}^2 \xi_{\rm tot} + 2B_{L,t}B_{L,\Delta}\xi_{\rm add} + B_{L,\Delta}^2 B_{\rm CIP}\hat{\xi}_{\rm CIP},$$

This equation has dependence on:

- **A linear bias with dark matter**
- **large scale clustering of baryons, potentially a new quantity to consider in galaxy evolution**
- **Compensated Isocurvature Perturbations (CIP) between baryons and dark matter in the early universe**

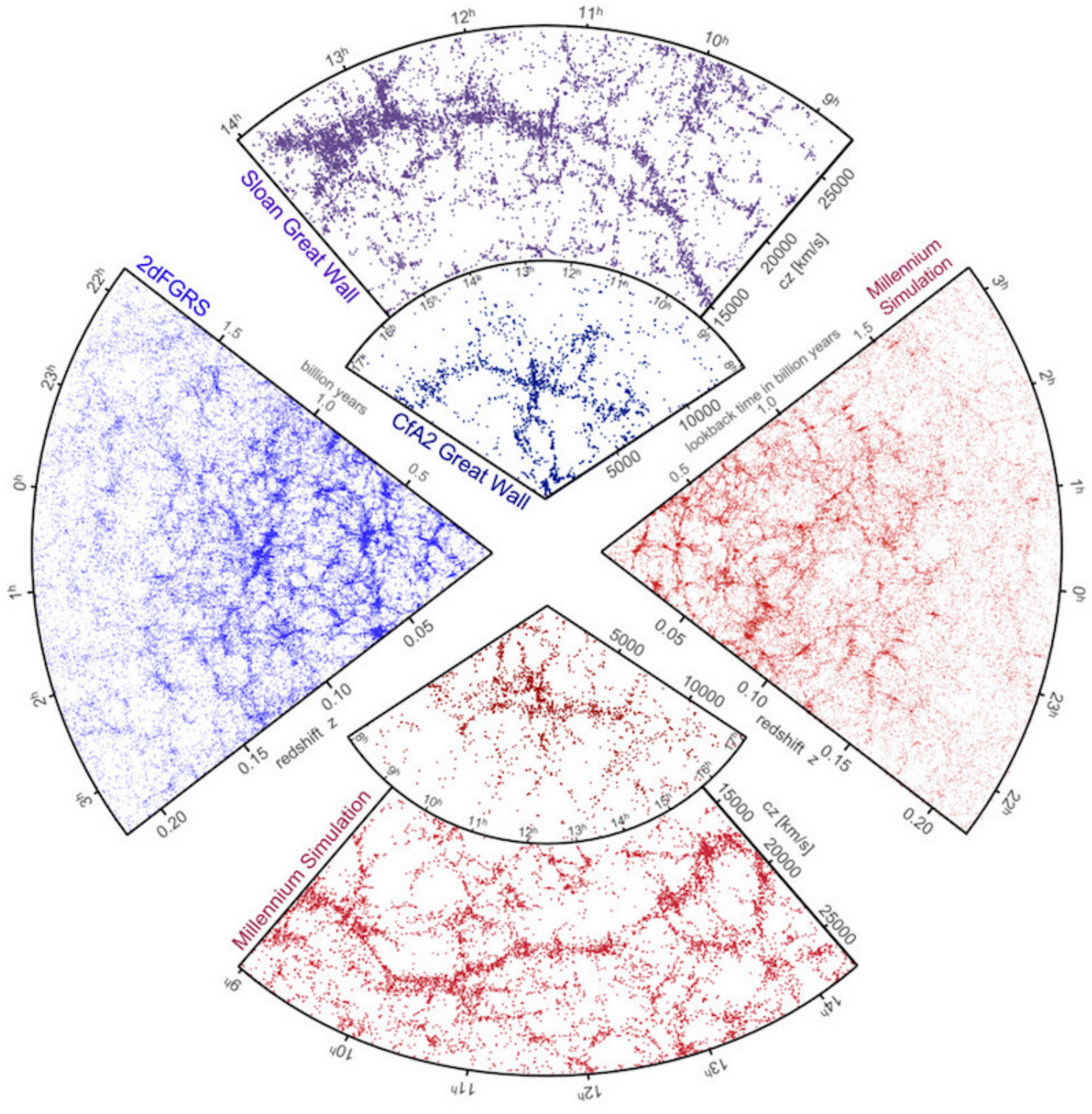
Looking at spatial cross-correlations with different quantities unlocks new physical interpretation of the data



M. T. Soumagnac, R. Barkana, C. G. Sabiu, A. Loeb, et.al **PRL 2016**
M. T. Soumagnac, C. G. Sabiu, R. Barkana, and J. Yoo **MNRAS 2019**

Machine Learning Approach

Use machine learning to match observation with simulation (model)

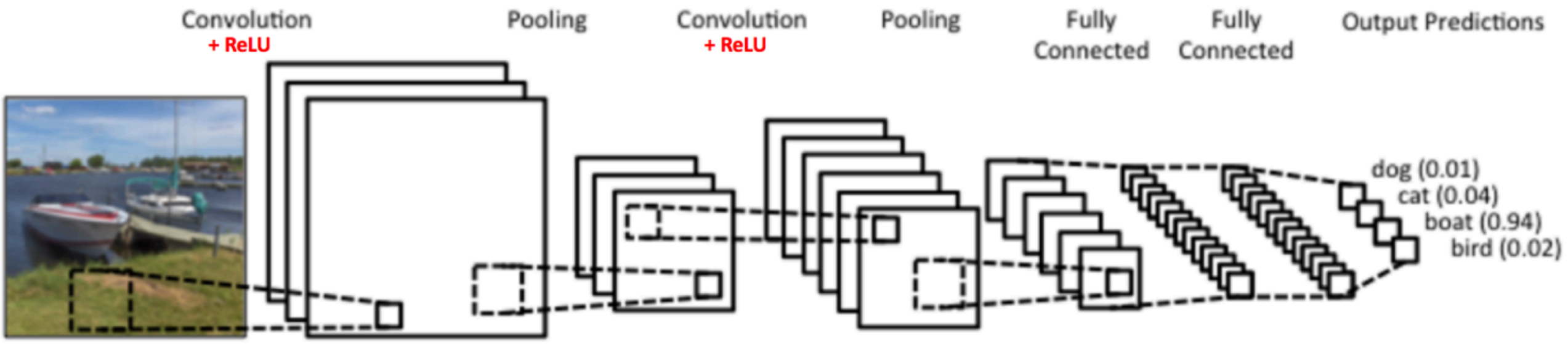


Convolutional Neural Networks

A convolutional neural network for visual classification

From a number of inputs obtain a meaningful output

By training CNN on a known set of data (labelled)



We will break down each step one by one…

Convolutional Neural Networks

1. Convolution Layer

Here are the three elements that enter into the convolution operation:

- Input image
- Feature detector
- Feature map

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

0	0	1
1	0	0
0	1	1

Feature Detector

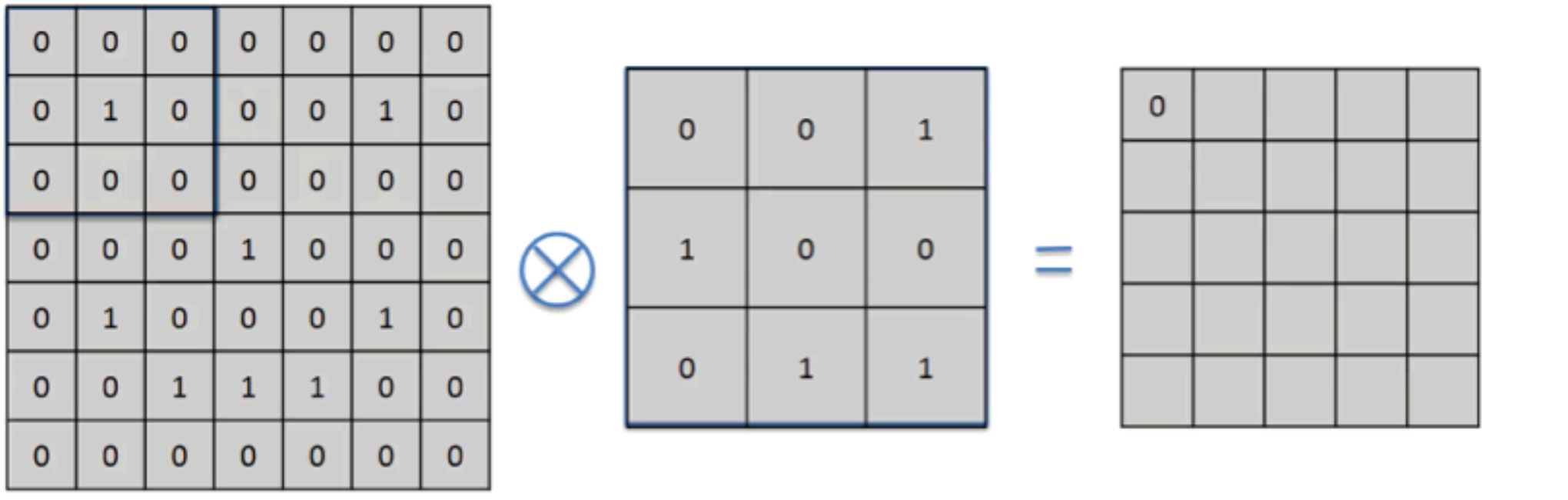
The feature detector is often referred to as a “kernel” or a “filter,” which might sound more familiar to you from other areas of physics.

Convolutional Neural Networks

Convolution Layer

Here are the three elements that enter into the convolution operation:

- Input image
- Feature detector
- Feature map



Input Image

Feature Detector

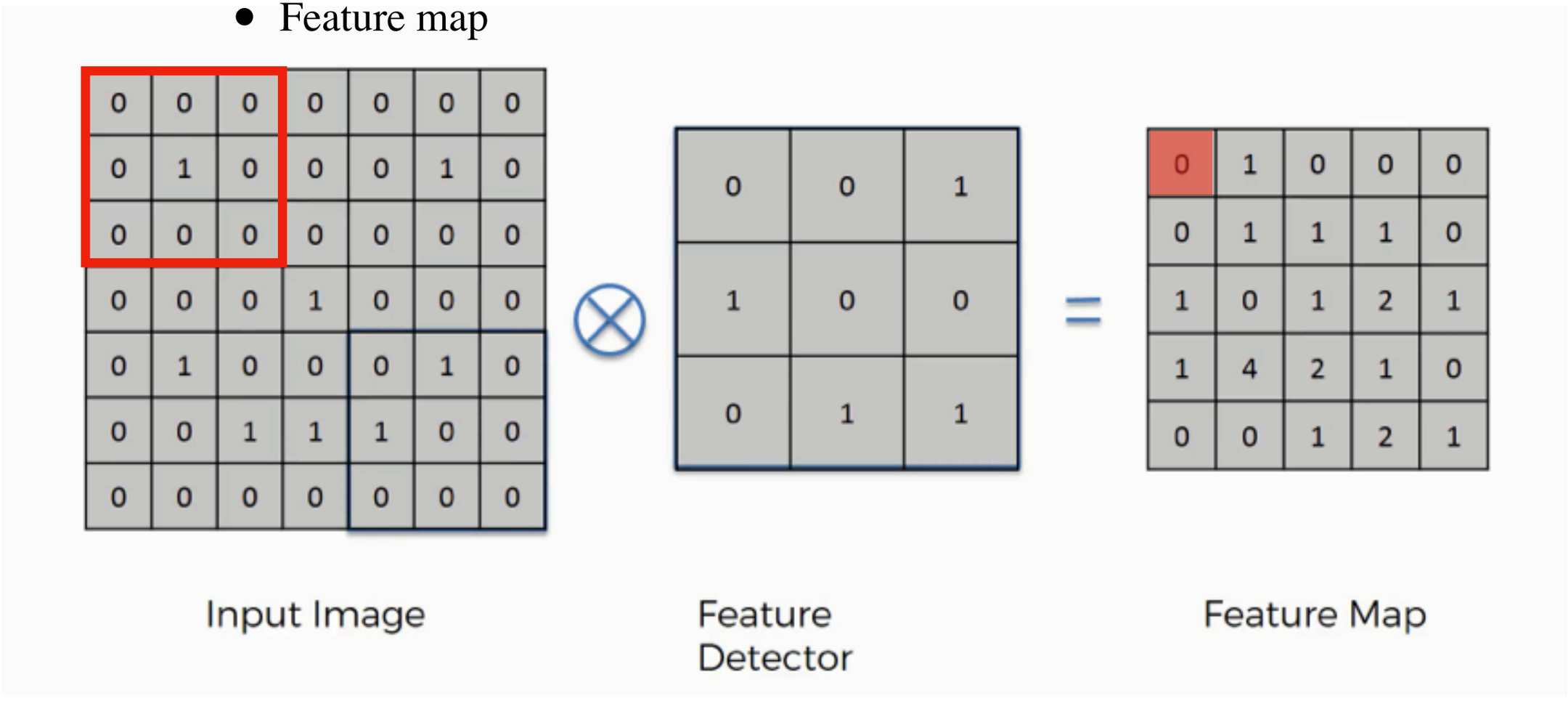
Feature Map

Convolutional Neural Networks

Convolution Layer

Here are the three elements that enter into the convolution operation:

- Input image
- Feature detector
- Feature map

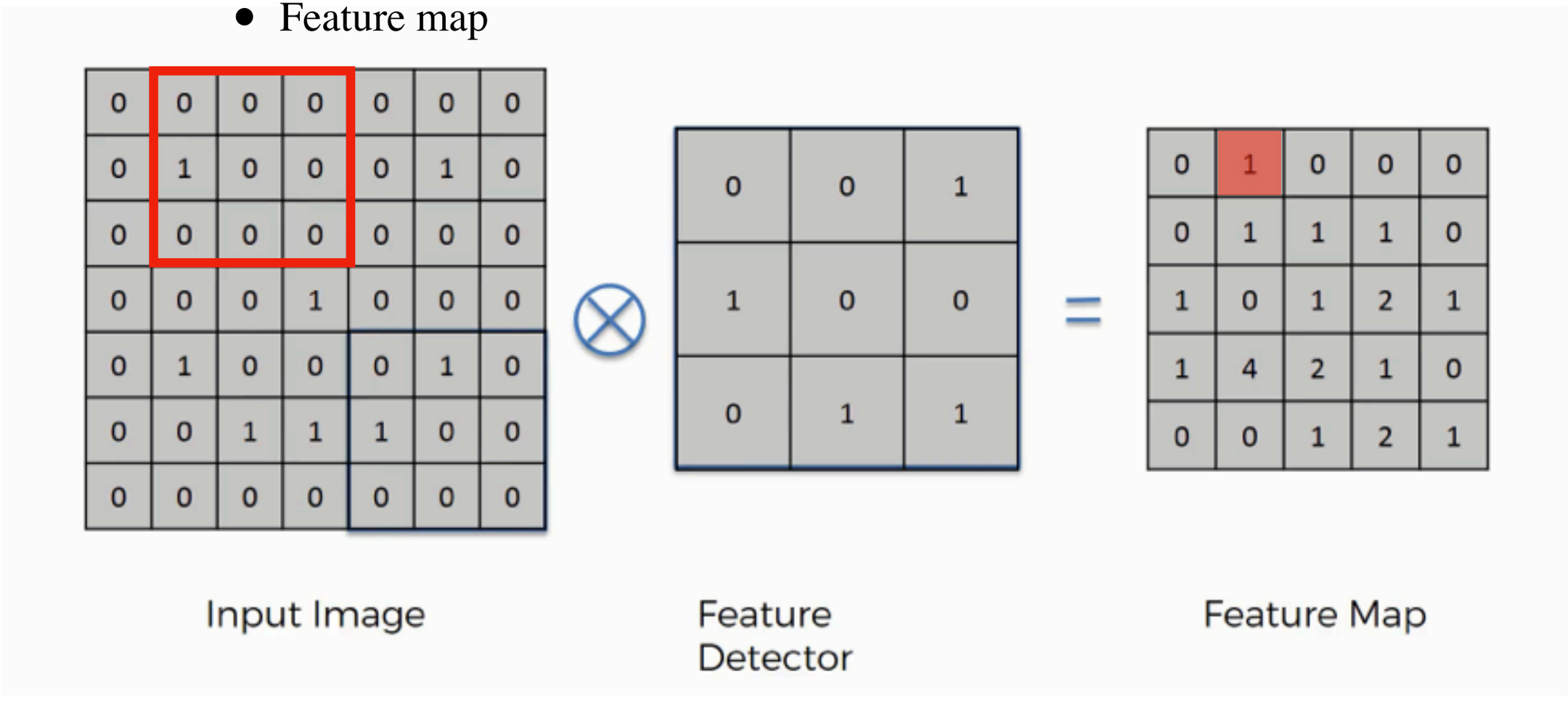


Convolutional Neural Networks

Convolution Layer

Here are the three elements that enter into the convolution operation:

- Input image
- Feature detector
- Feature map

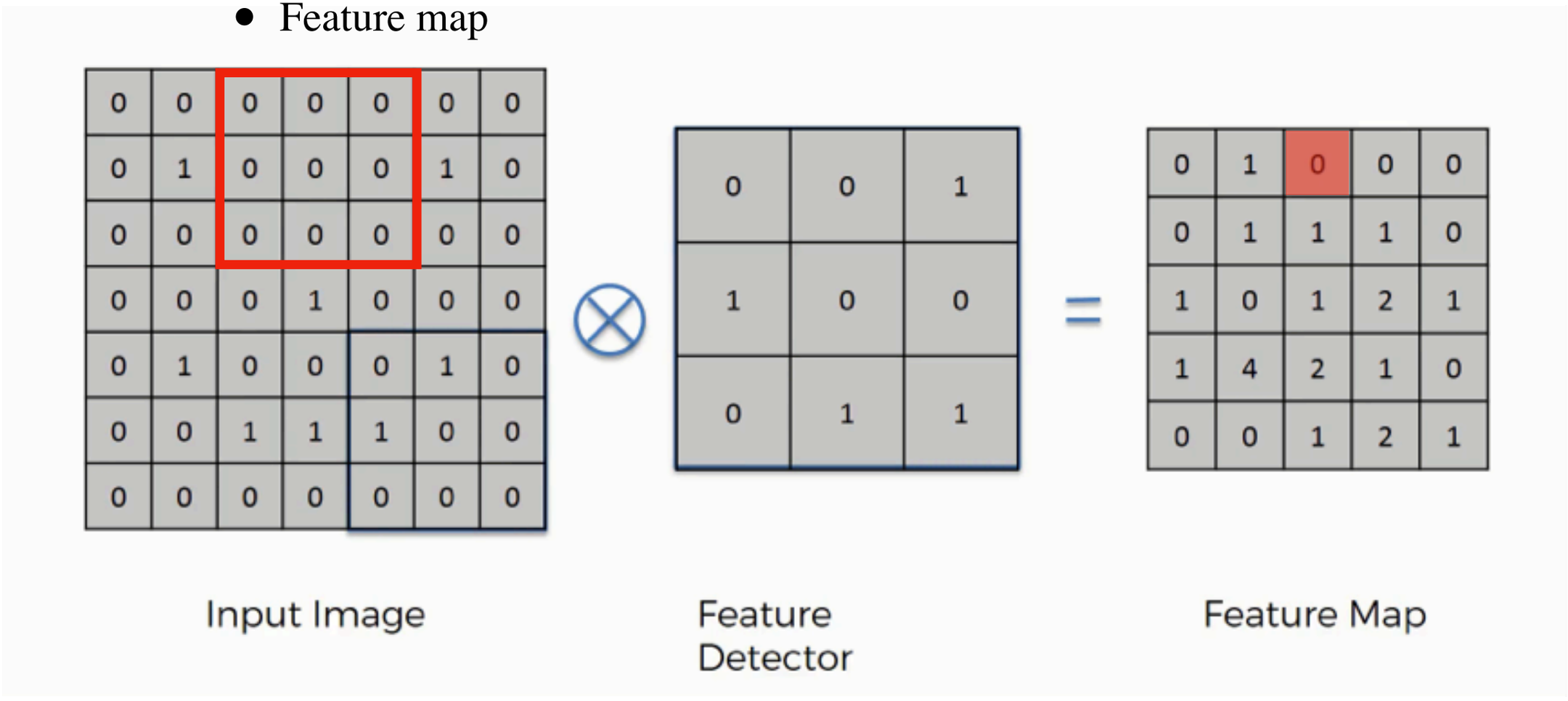


Convolutional Neural Networks

Convolution Layer

Here are the three elements that enter into the convolution operation:

- Input image
- Feature detector
- Feature map

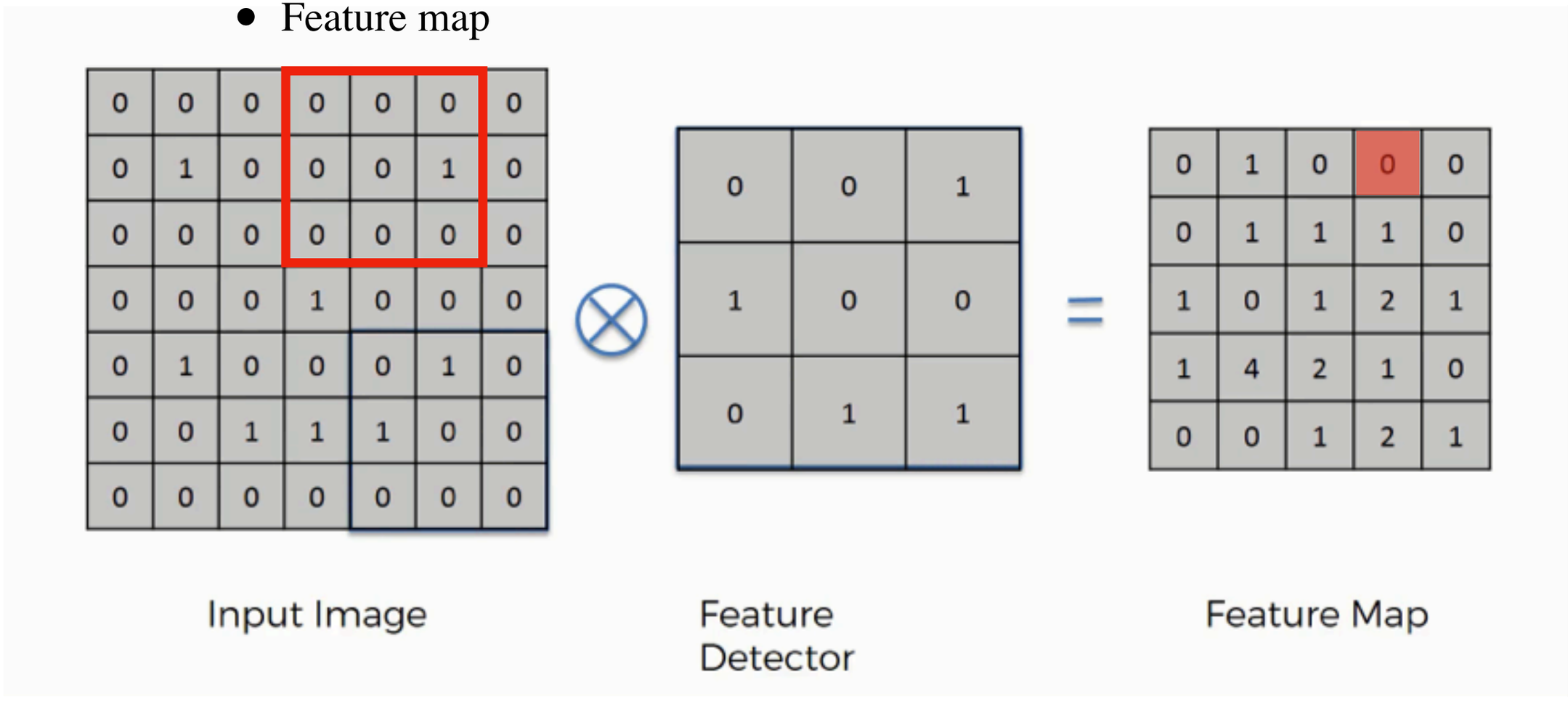


Convolutional Neural Networks

Convolution Layer

Here are the three elements that enter into the convolution operation:

- Input image
- Feature detector
- Feature map

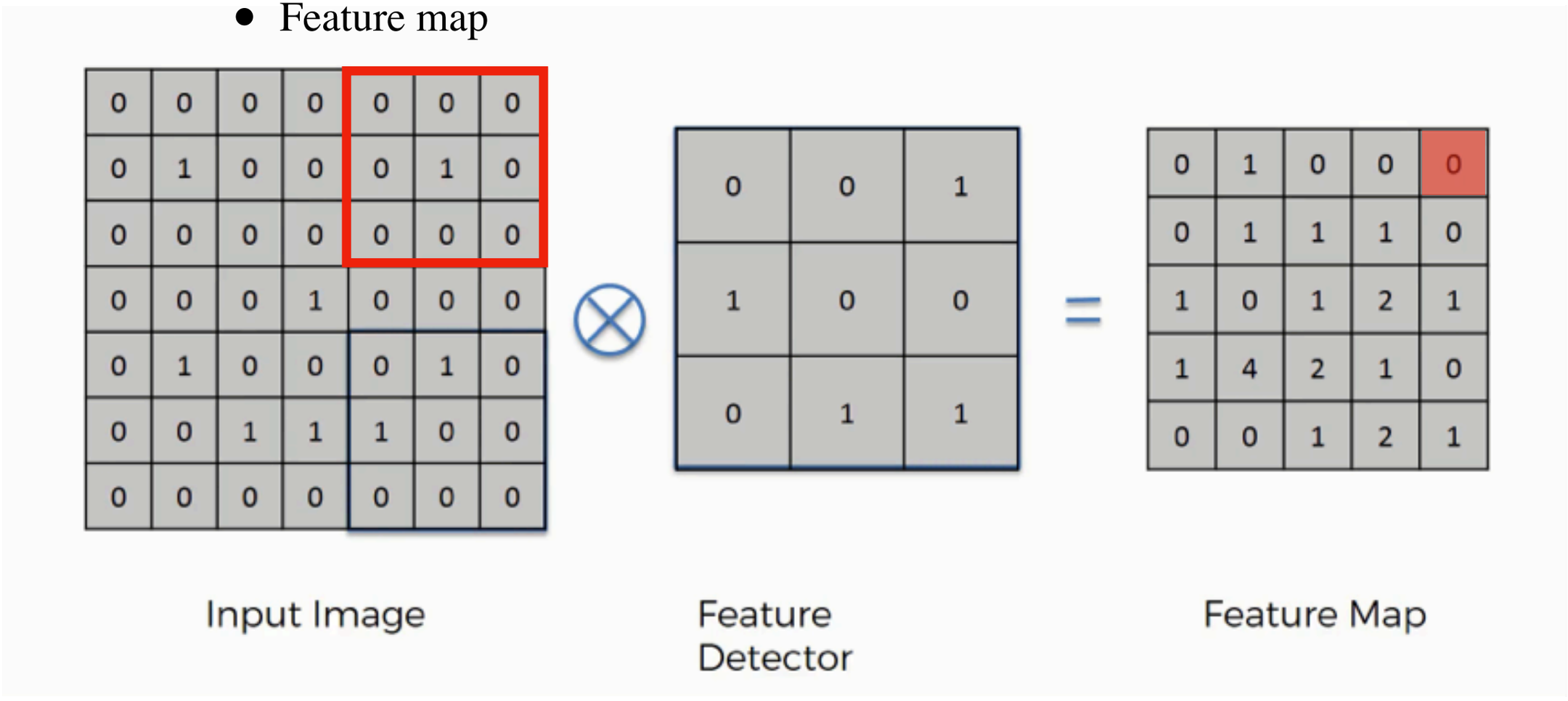


Convolutional Neural Networks

Convolution Layer

Here are the three elements that enter into the convolution operation:

- Input image
- Feature detector
- Feature map

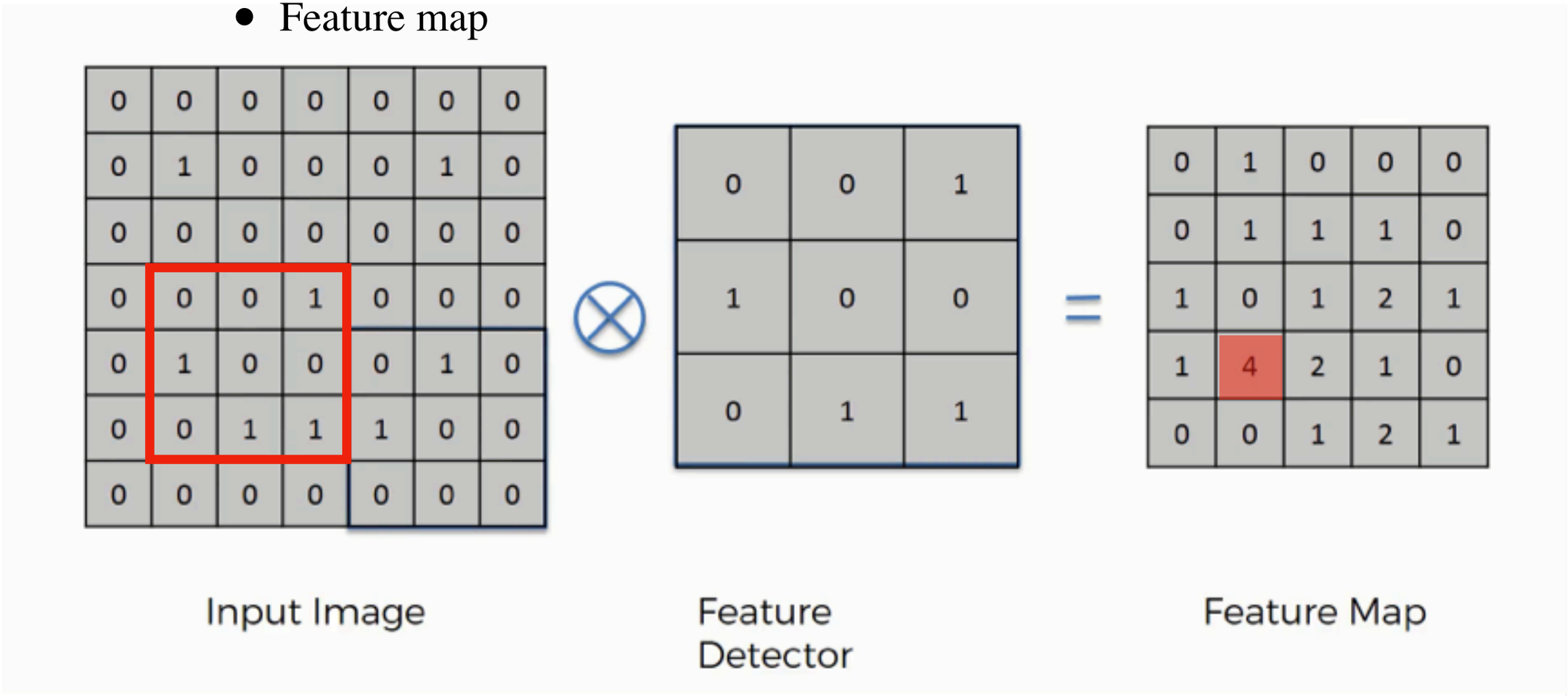


Convolutional Neural Networks

Convolution Layer

Here are the three elements that enter into the convolution operation:

- Input image
- Feature detector
- Feature map

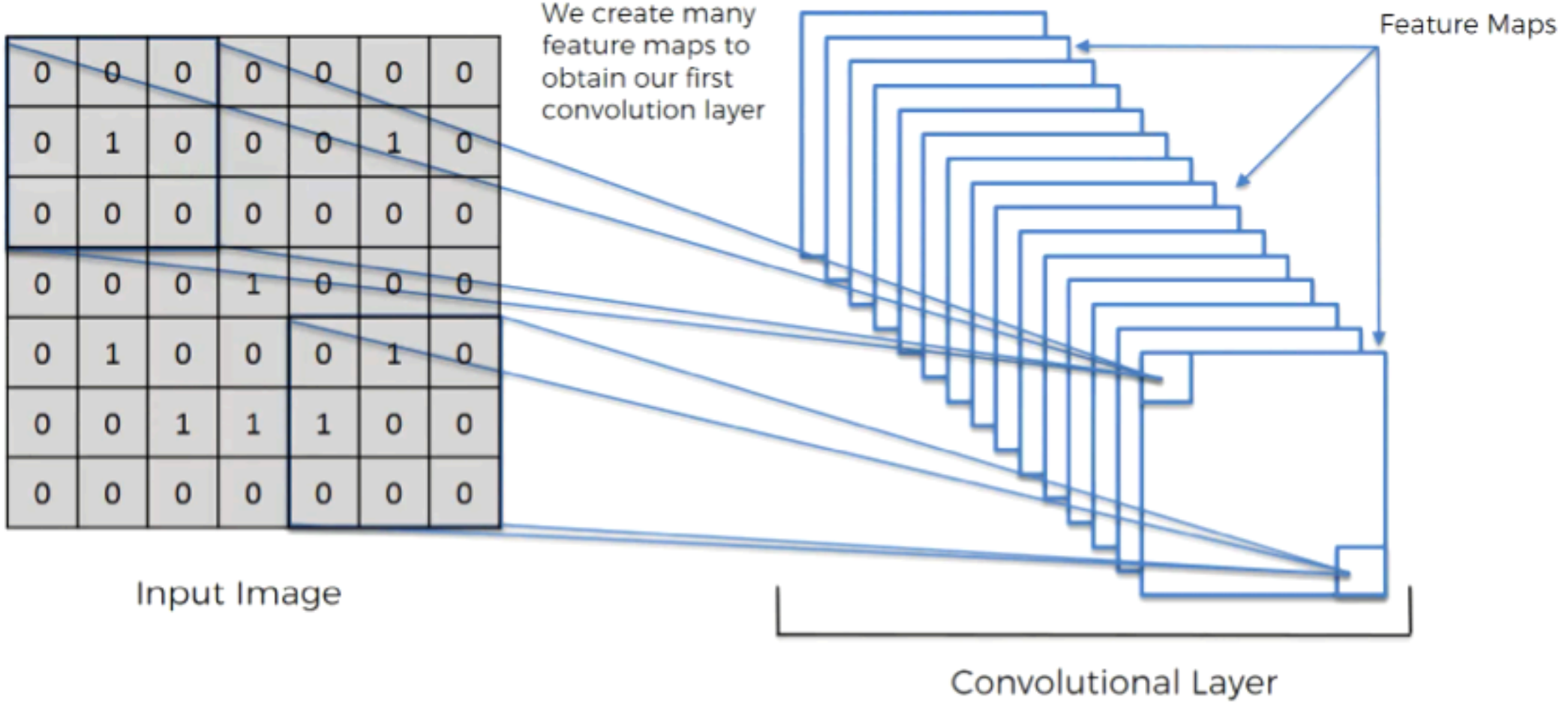


Convolutional Neural Networks

Convolution Layer

In reality, convolutional neural networks develop multiple feature detectors and use them to develop several feature maps which are referred to as convolutional layers

Through training, the network determines what features it finds important in order for it to be able to scan images and categorize them more accurately.



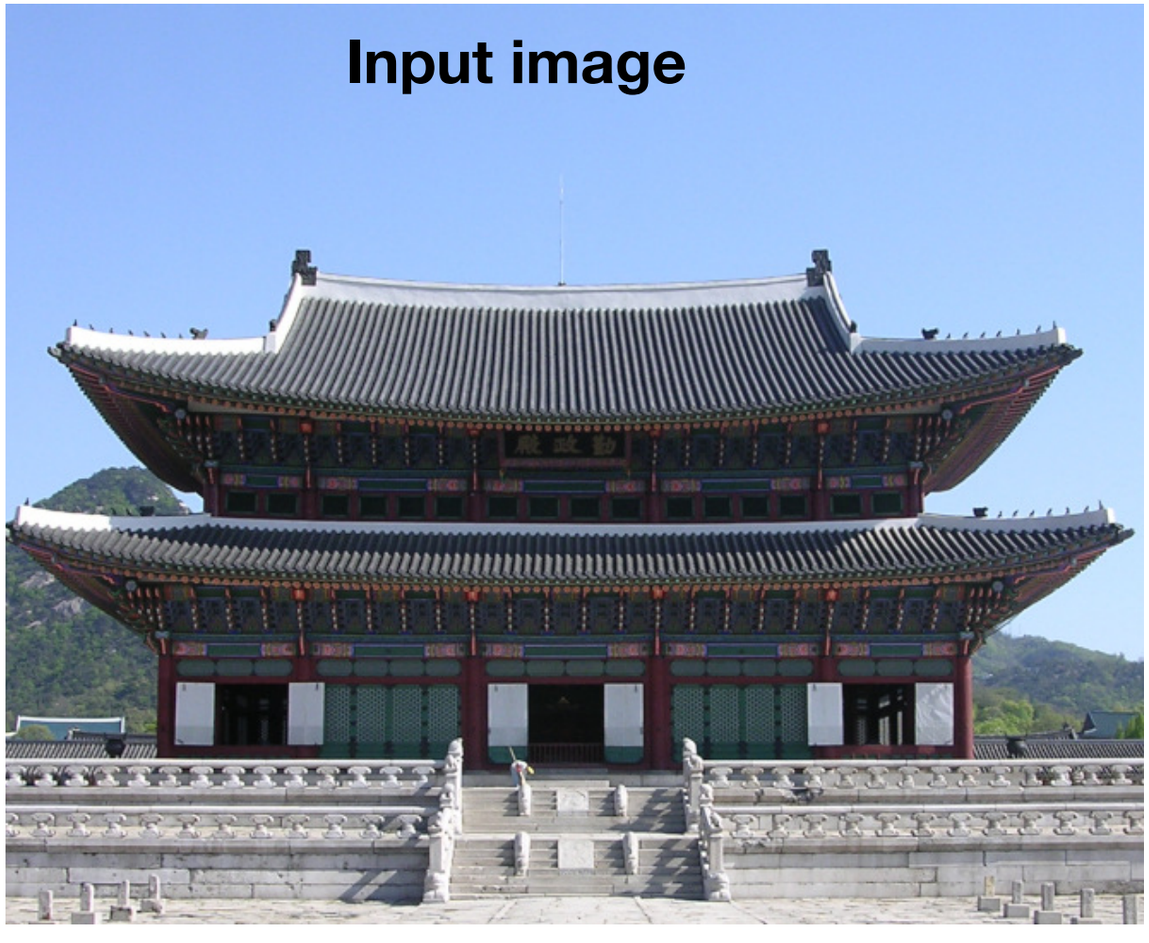
Convolutional Neural Networks

1. Convolution Layer

These filters have a use in image processing and that will help give us a more intuitive feeling about them

Sharpen

0	-1	0
-1	5	-1
0	-1	0



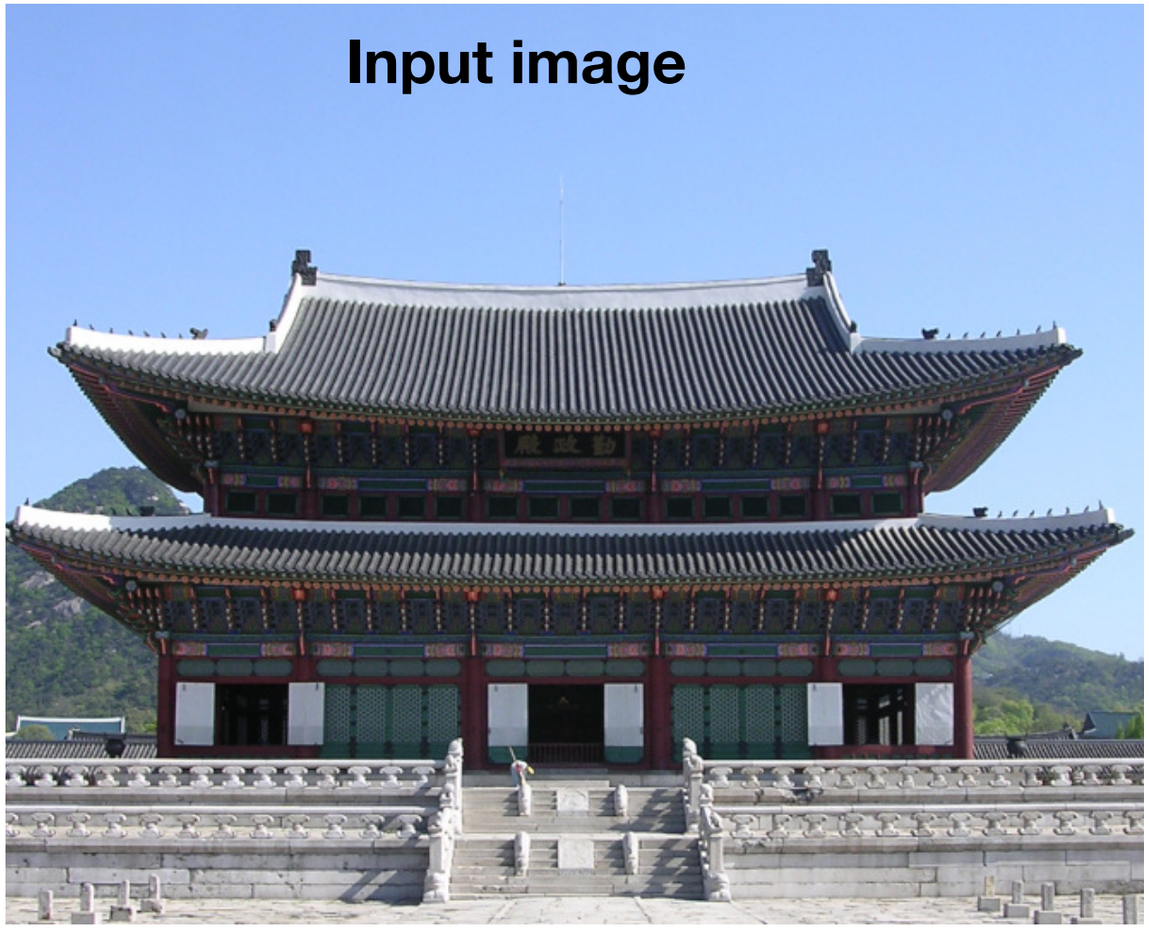
Convolutional Neural Networks

1. Convolution Layer

These filters have a use in image processing and that will help give us a more intuitive feeling about them

Blur

1	1	1
1	1	1
1	1	1



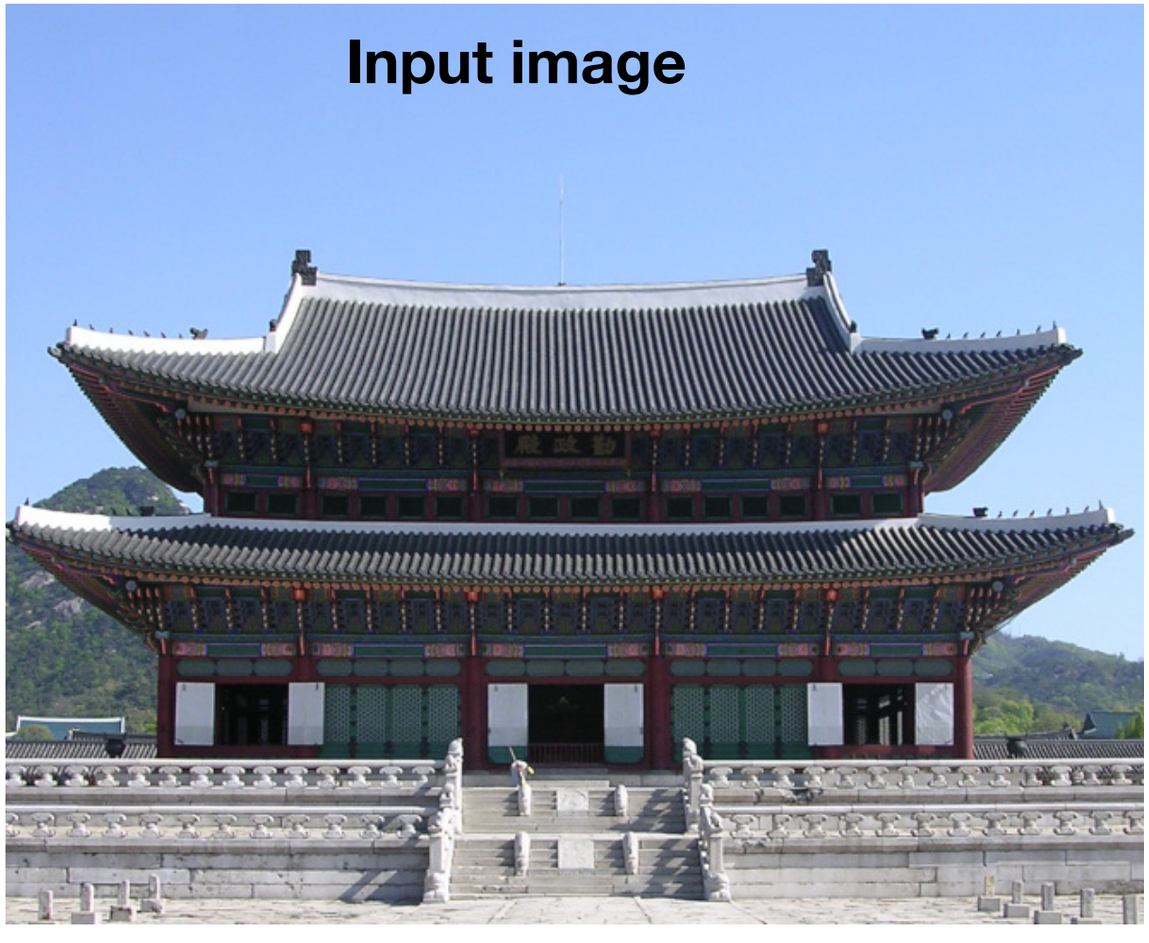
Convolutional Neural Networks

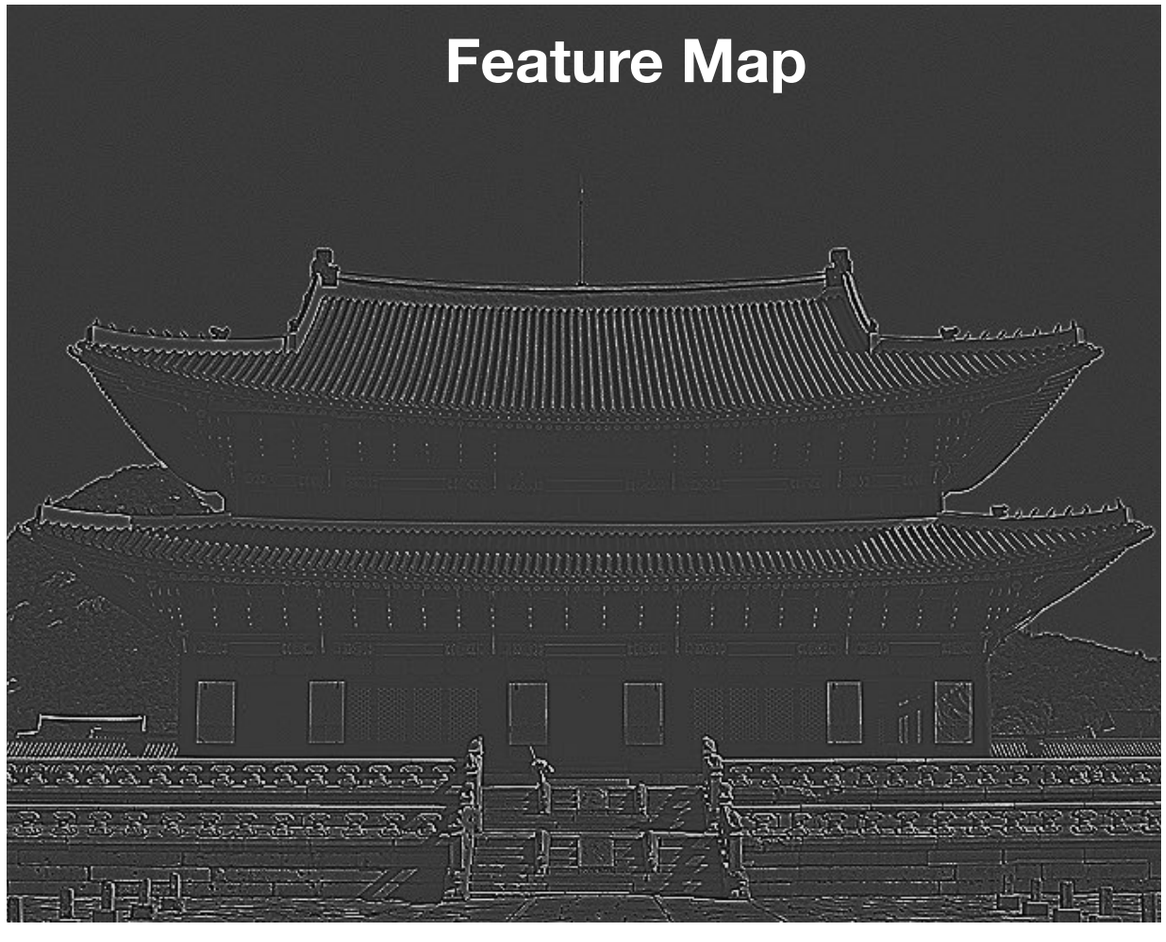
1. Convolution Layer

These filters have a use in image processing and that will help give us a more intuitive feeling about them

Edge Detect

0	1	0
1	-4	1
0	1	0





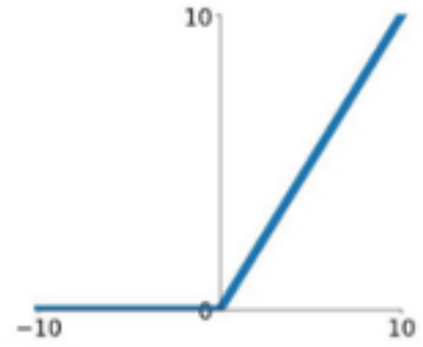
Convolutional Neural Networks

2. Activation Function (eg ReLU)

Rectified Linear Unit (ReLU) is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero. The purpose of ReLU is to introduce non-linearity in our ConvNet, since most of the real-world data we would want our ConvNet to learn would be non-linear

ReLU

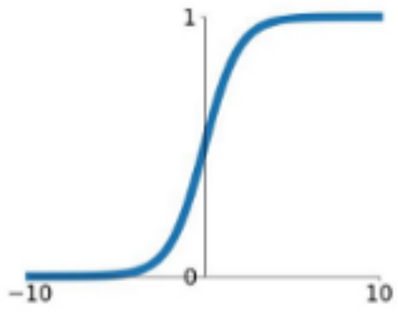
$$\max(0, x)$$



Others…

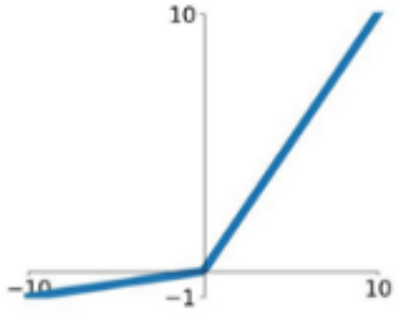
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



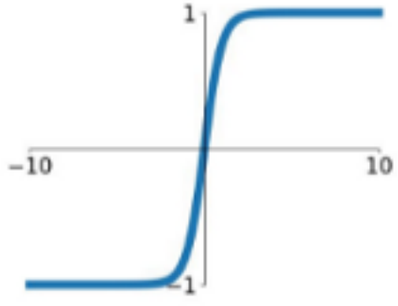
Leaky ReLU

$$\max(0.1x, x)$$



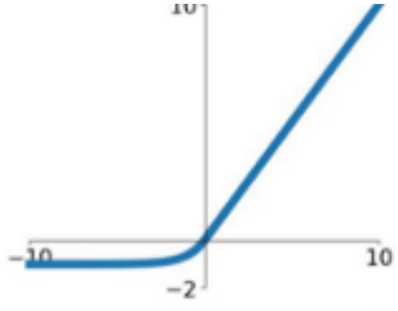
tanh

$$\tanh(x)$$



ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Convolutional Neural Networks

2. ReLU

Rectified Linear Unit (ReLU) is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero. The purpose of ReLU is to introduce non-linearity in our ConvNet, since most of the real-world data we would want our ConvNet to learn would be non-linear

Original

Feature detected image

Convolutional Neural Networks

2. ReLU

Rectified Linear Unit (ReLU) is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero. The purpose of ReLU is to introduce non-linearity in our ConvNet, since most of the real-world data we would want our ConvNet to learn would be non-linear

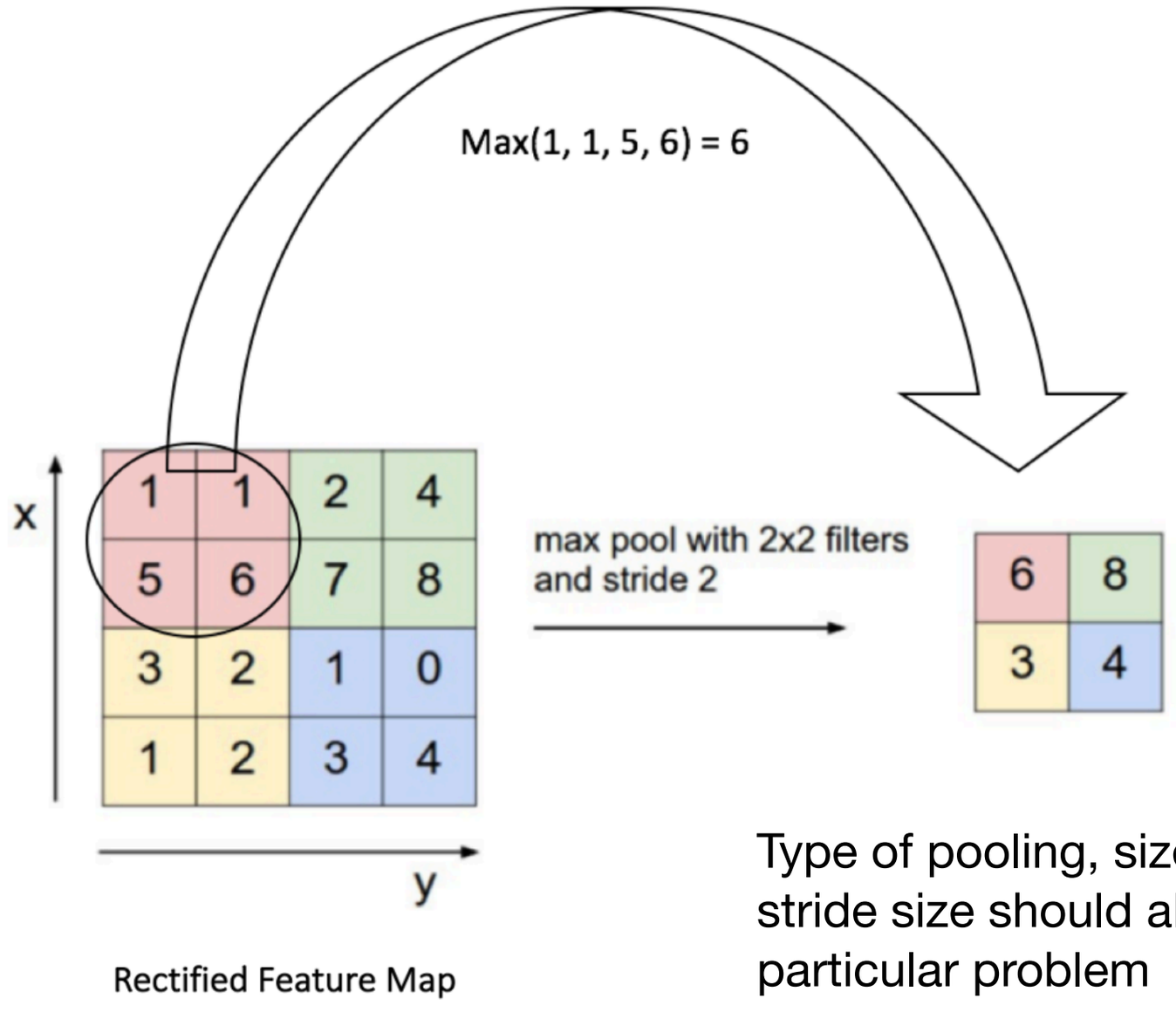
Feature detected image

Activated Feature Map

Convolutional Neural Networks

3. Pooling

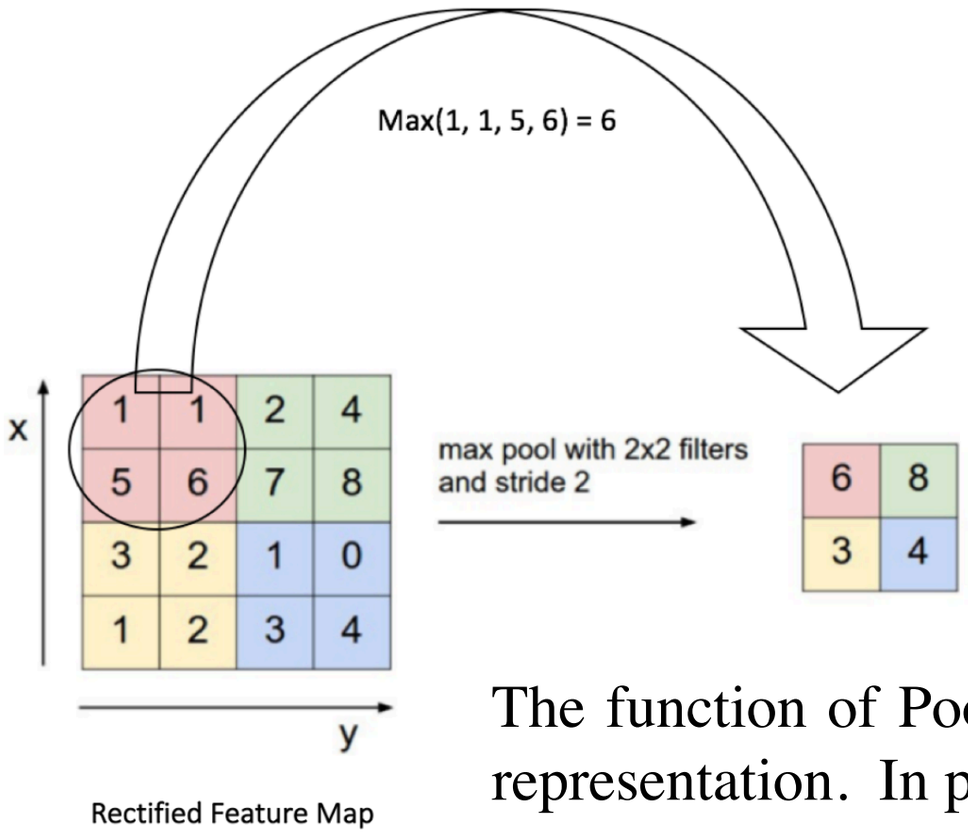
Spatial Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: Max, Average, Sum etc.



Type of pooling, size of filter window and stride size should all be tested for your particular problem

Convolutional Neural Networks

3. Pooling



The function of Pooling is to progressively reduce the spatial size of the input representation. In particular, pooling

- makes the input representations (feature dimension) smaller and more manageable
- reduces the number of parameters and computations in the network, therefore, controlling overfitting
- makes the network invariant to small transformations, distortions and translations in the input image.
- helps us arrive at an almost scale invariant representation of our image (the exact term is “equivariant”). This is very powerful since we can detect objects in an image no matter where they are located

Convolutional Neural Networks

4. Flatten

1	1	0
4	2	1
0	2	1

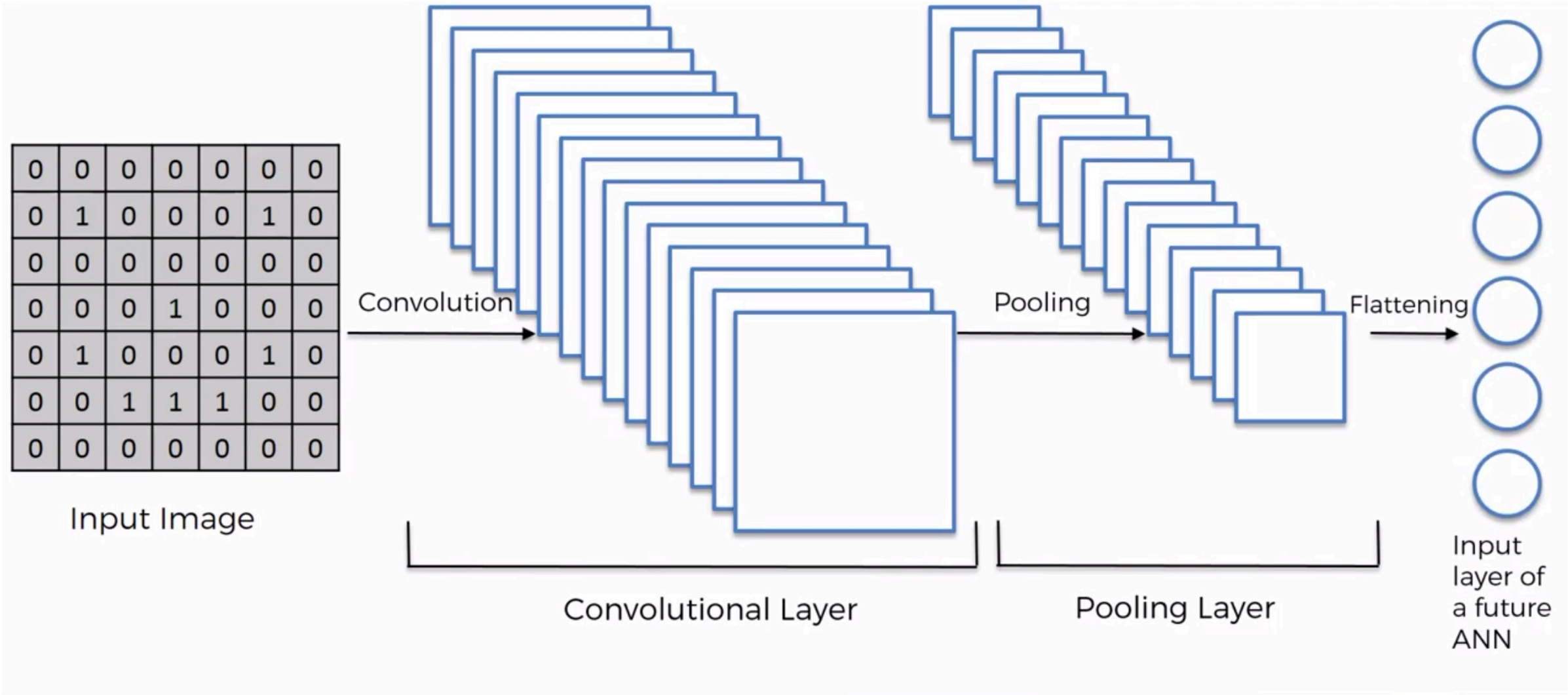
Pooled Feature Map

Flattening

1
1
0
4
2
1
0
2
1

Convolutional Neural Networks

4. Flatten

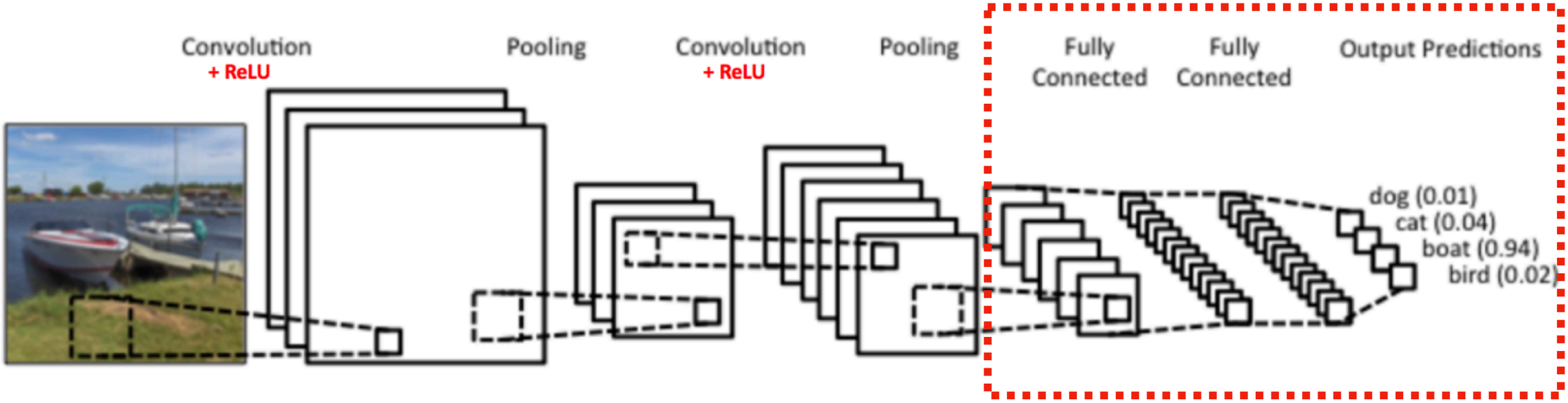


Convolutional Neural Networks

5. Fully connected layer(s)

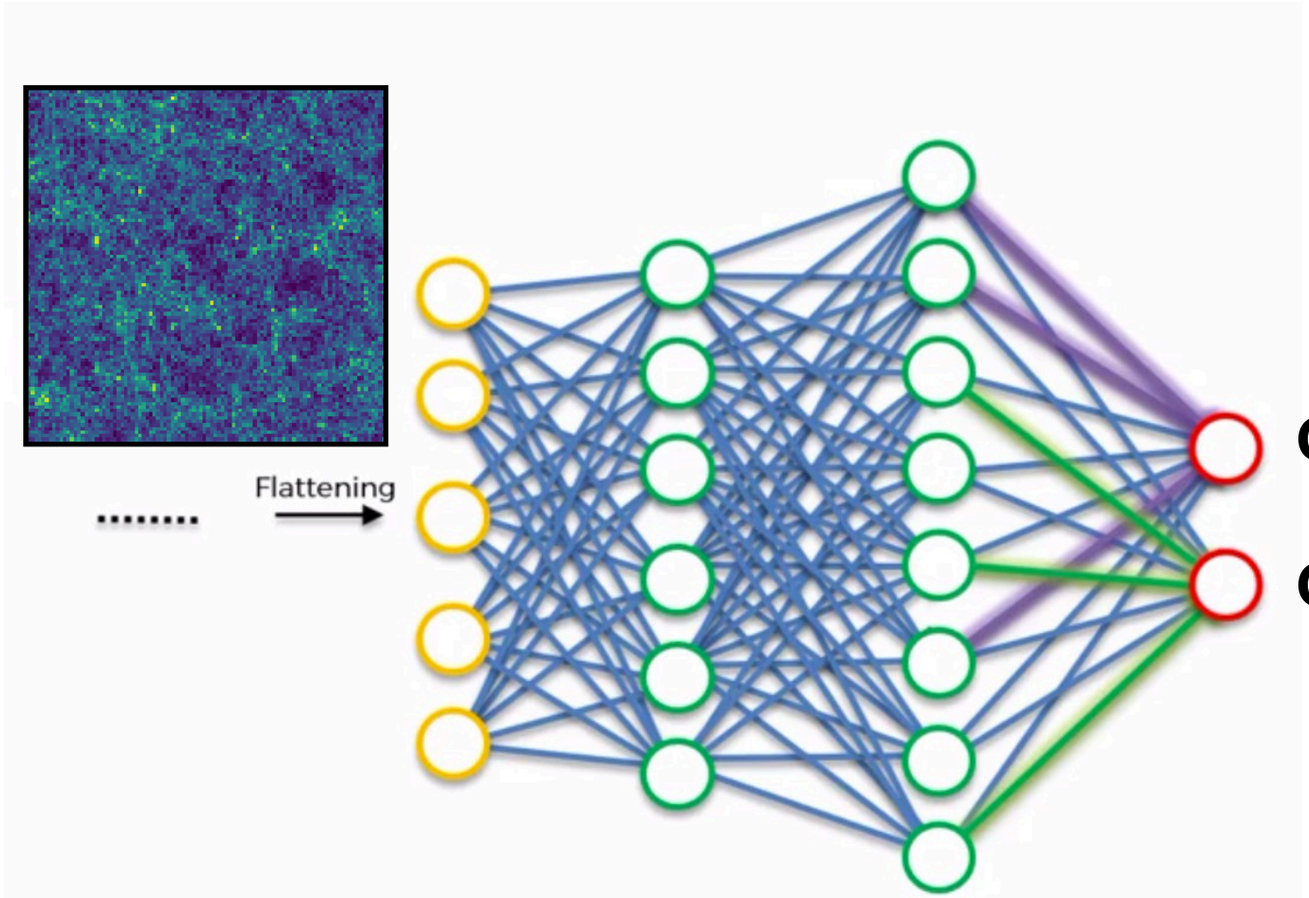
The Fully Connected layer is a traditional Multi Layer Network that uses a softmax activation function in the output layer. The term “Fully Connected” implies that every neuron in the previous layer is connected to every neuron on the next layer.

The output from the convolutional and pooling layers represent high-level features of the input image. The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset.



Convolutional Neural Networks

5. Fully connected layer(s)



A Standard Multi Layer Neural Network

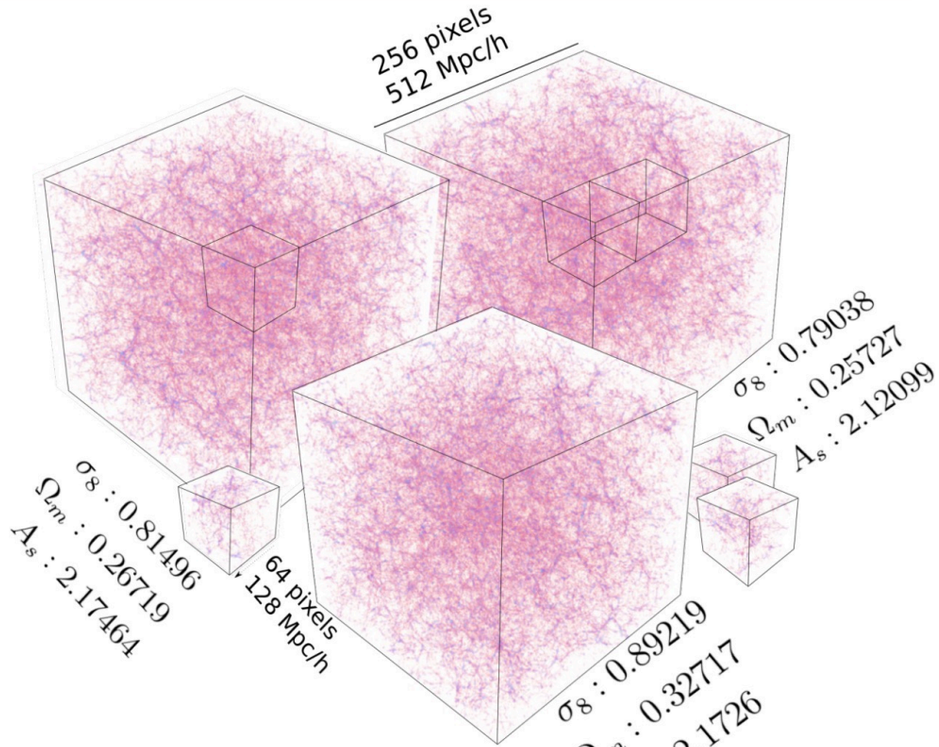
Cosmological parameters
Or
Cosmological Models

Deep Learning the Large Scale Structure

Ok lets apply this to cosmology!

Deep Learning the Large Scale Structure

Ravanbakhsh et al. (2017), Mathuriya et al. (2018) showed that convolutional neural networks can be trained to predict cosmological parameters from the visual shape of the large scale structure, i.e. the filaments, clusters and voids of the cosmic density field.

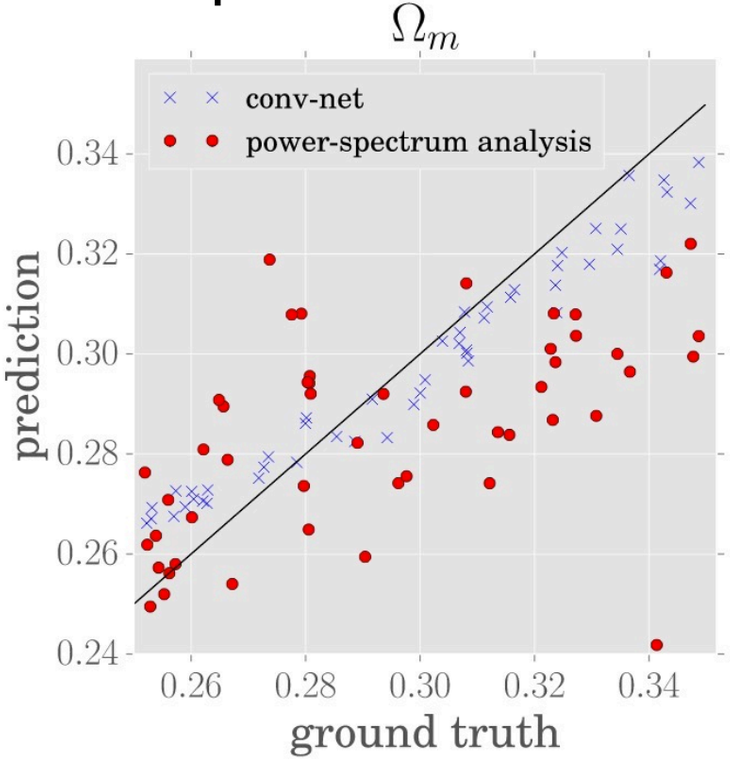


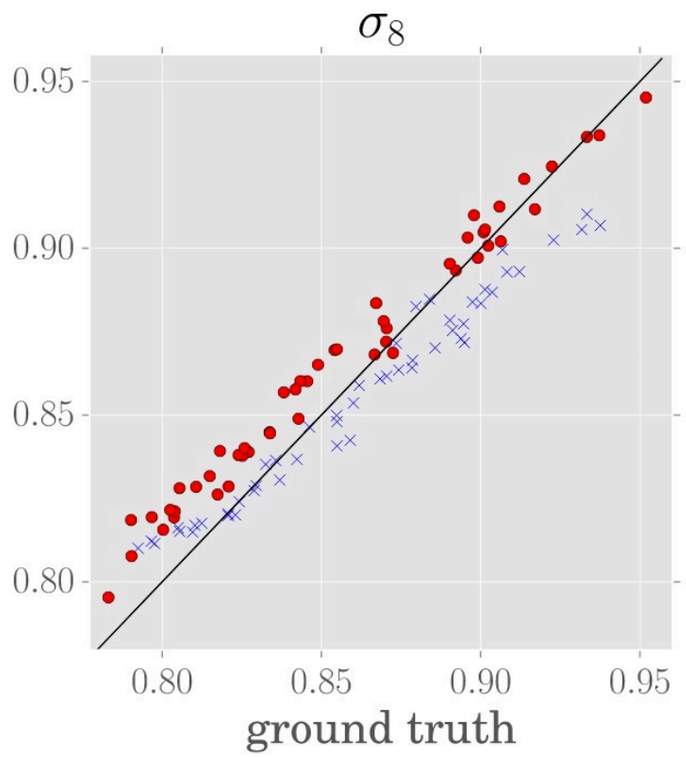
500 COmoving Lagrangian Acceleration(COLA) simulations

512Mpc box with 512^3 dark matter particles

Output at z=0

Limitations:
Cubic volume at z=0
Real space
Dark Matter particles
Fixed Architecture





Deep Learning the Large Scale Structure

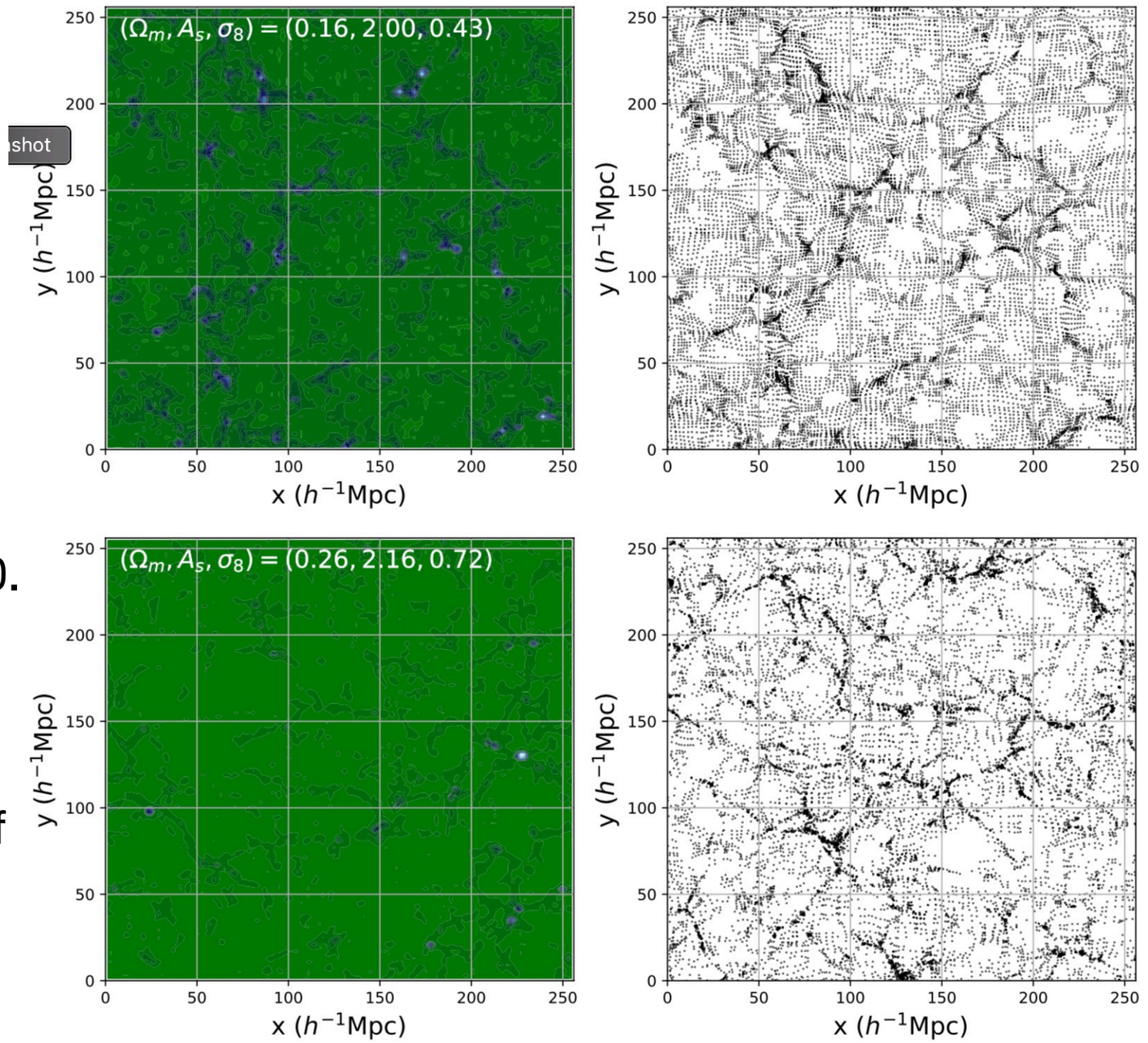
In a grid of 31x15 parameter combinations

$$0.16 < \Omega_M < 0.46$$
$$0.4 < \sigma_8 < 1.1$$

We run COLA DM simulations with with 128^3 particles, in a 256 Mpc box, using timesteps 40 output at z=0.

We grid the data onto 2Mpc voxels.

The input of the whole network is a 32^3-voxel (i.e. (64Mpc)3) subcube of the density field.

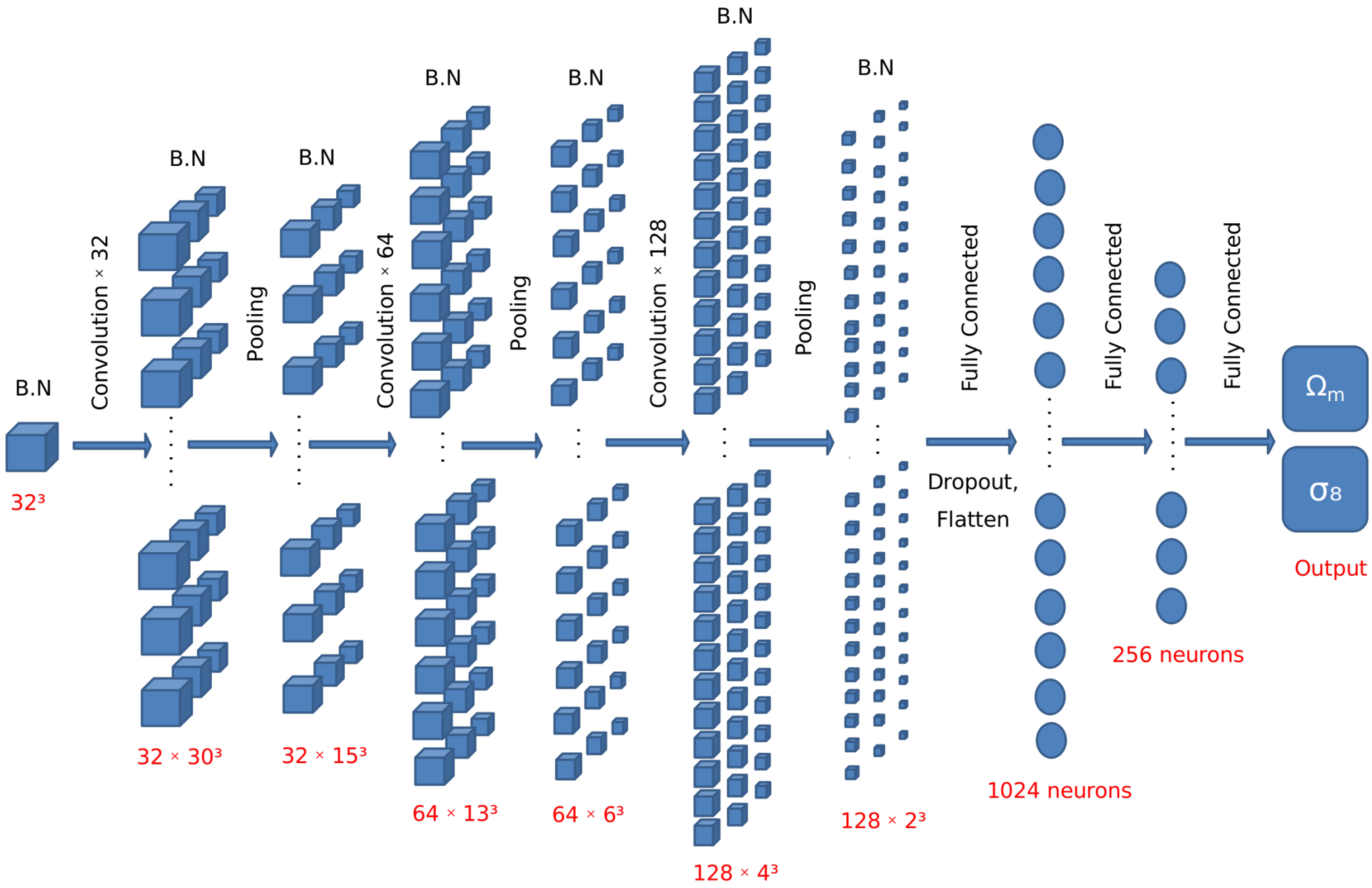


arXiv:1908.10590

Pan, Liu, Forero-Romero, Sabiu, Li, Miao, (led by) Xiao-Dong Li

Deep Learning the Large Scale Structure

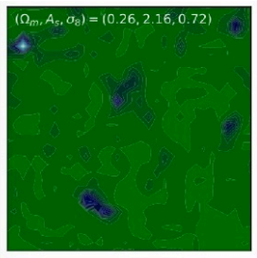
Default Architecture

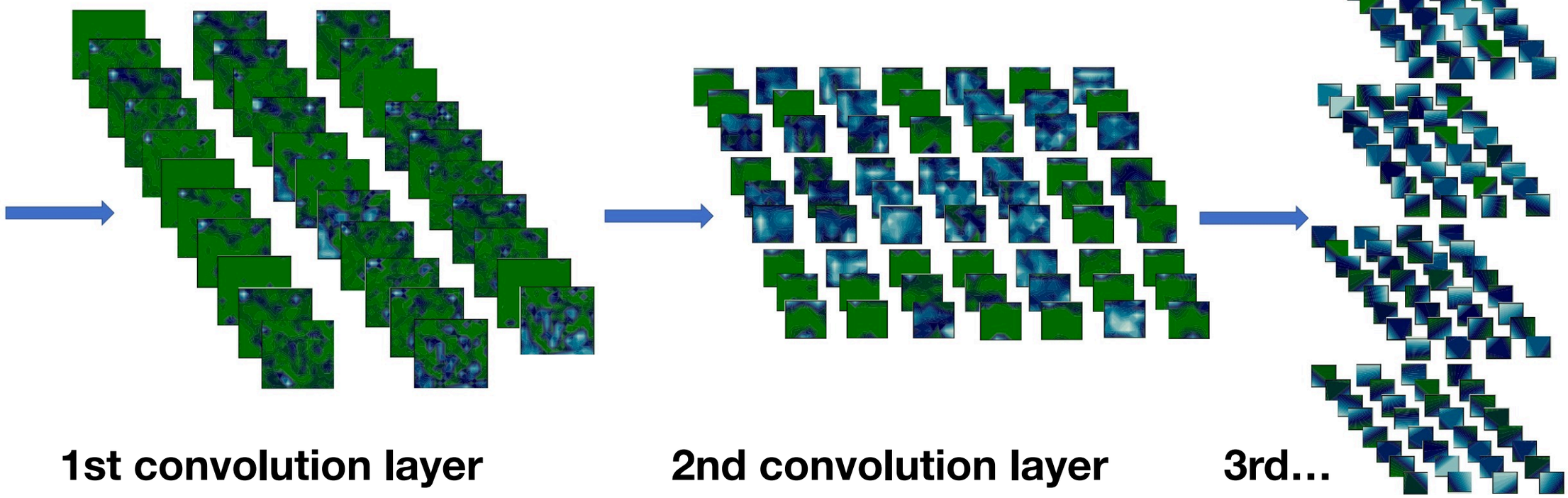


Deep Learning the Large Scale Structure

Inspecting the CNN

Om = 0.26



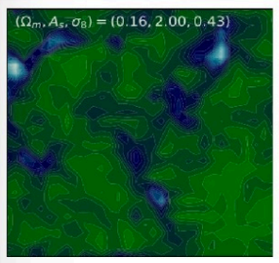


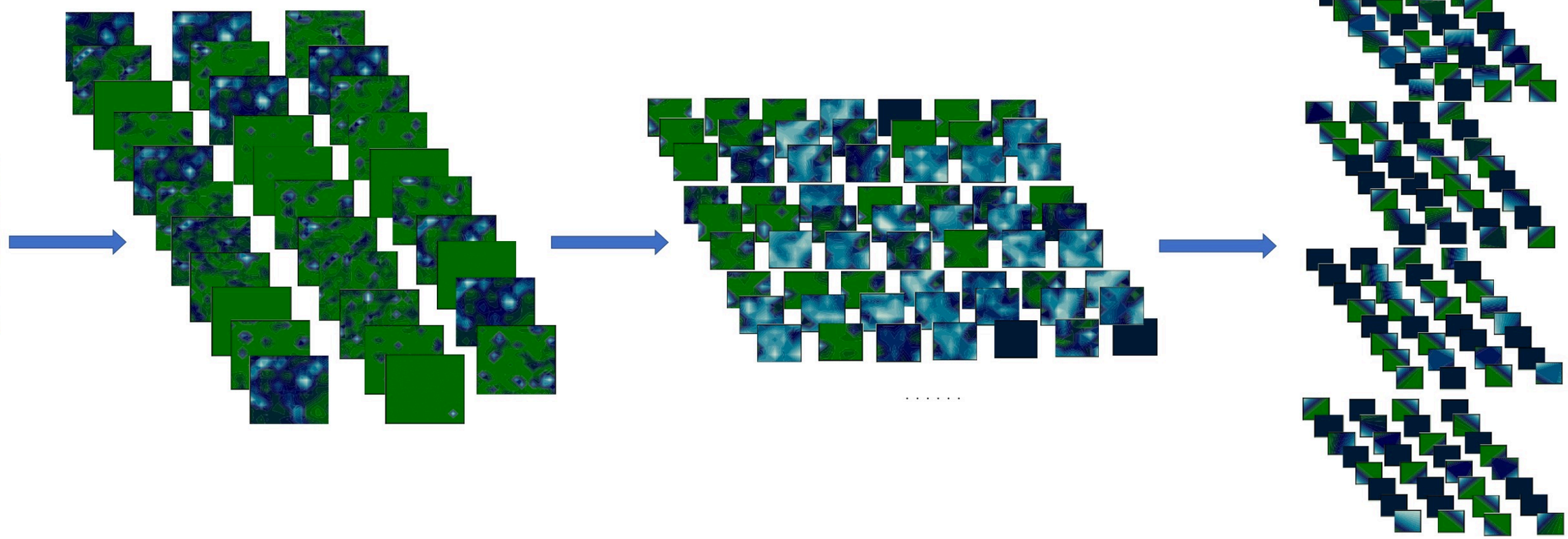
1st convolution layer

2nd convolution layer

3rd…

Om = 0.16





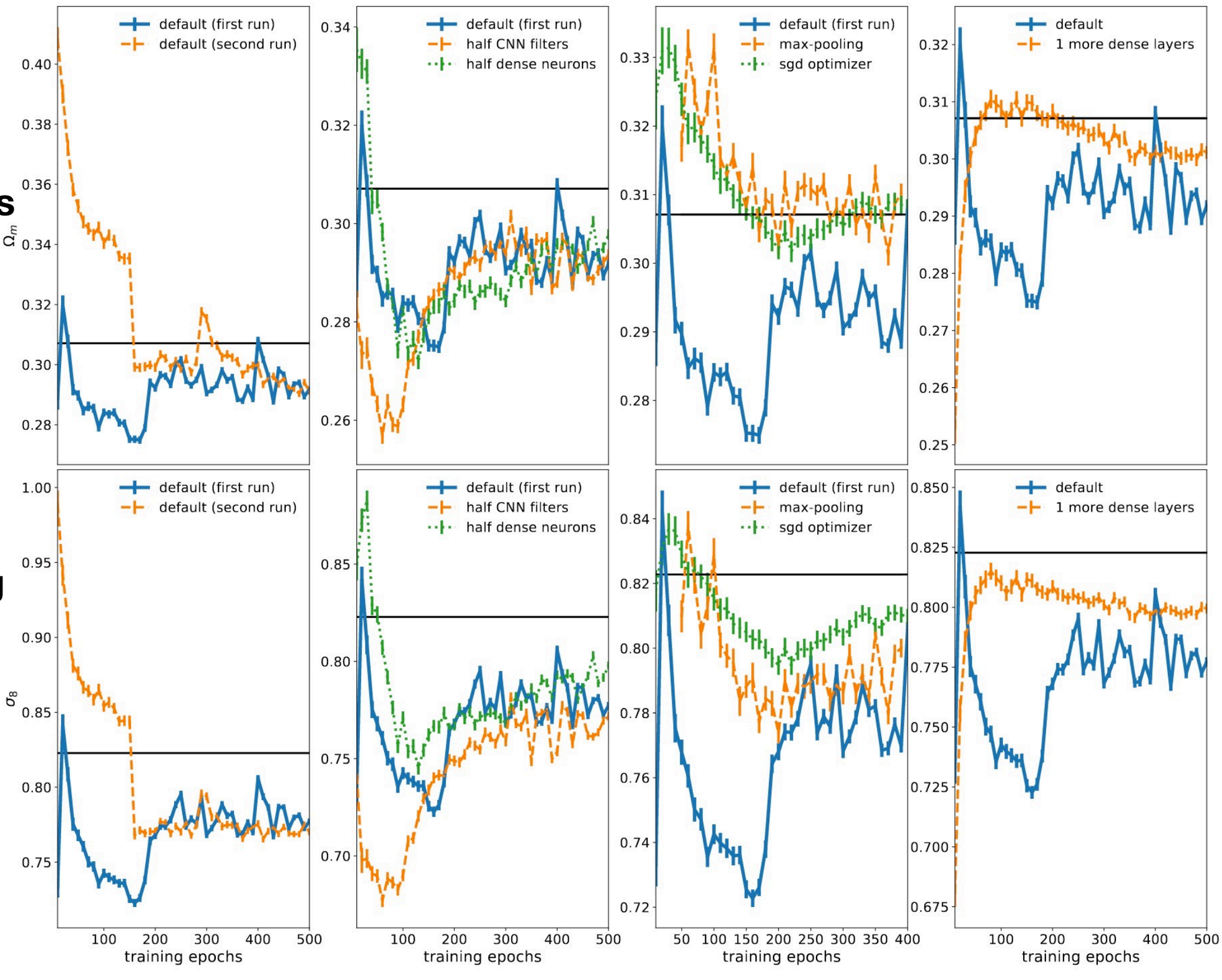
Deep Learning the Large Scale Structure

Learning Curves

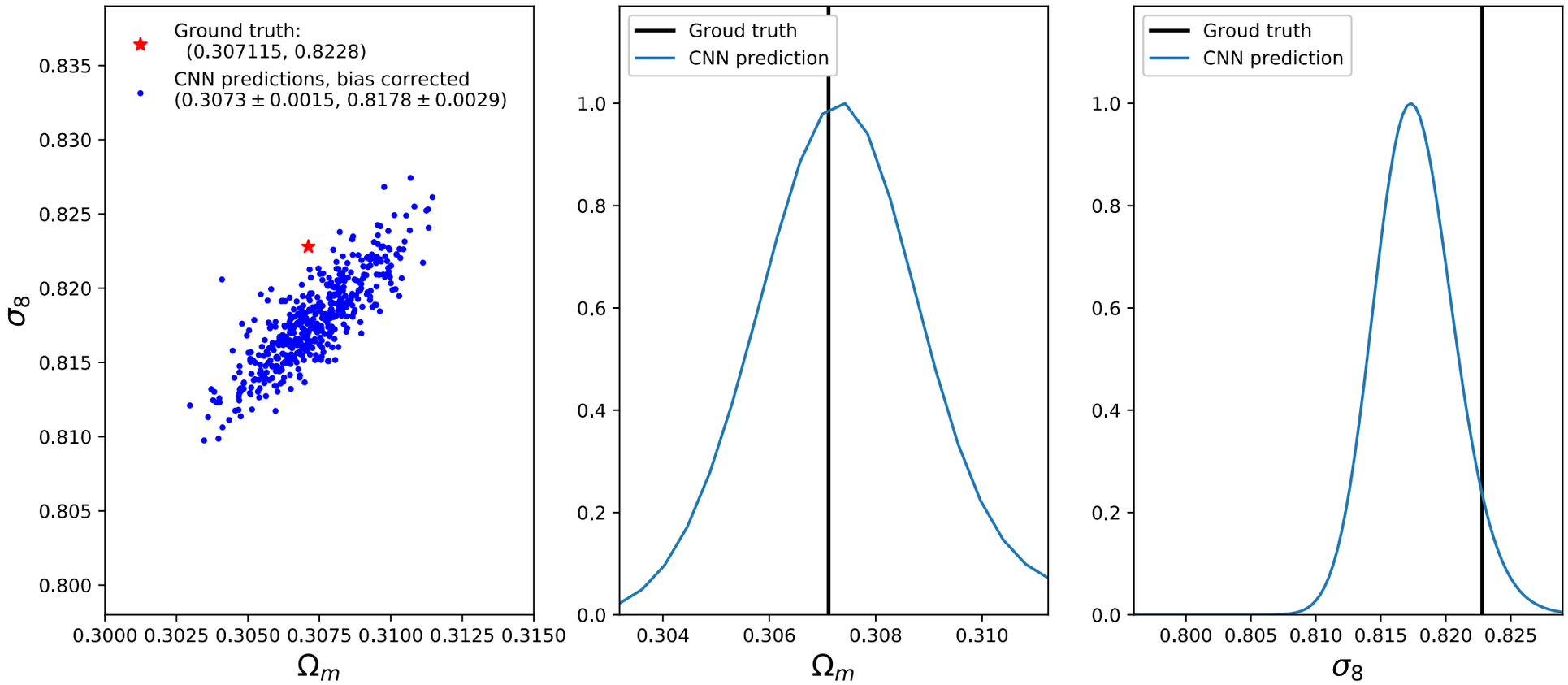
Varying:
- **# of CNN filters**
- **# of dense neurons**
- **# of neuron layers**
- **Optimiser**
- **Pooling type**

Clear advantage in max-pooling over the average pooling

Clear advantage of the sgd optimiser over the default ‘Adam’ optimiser.

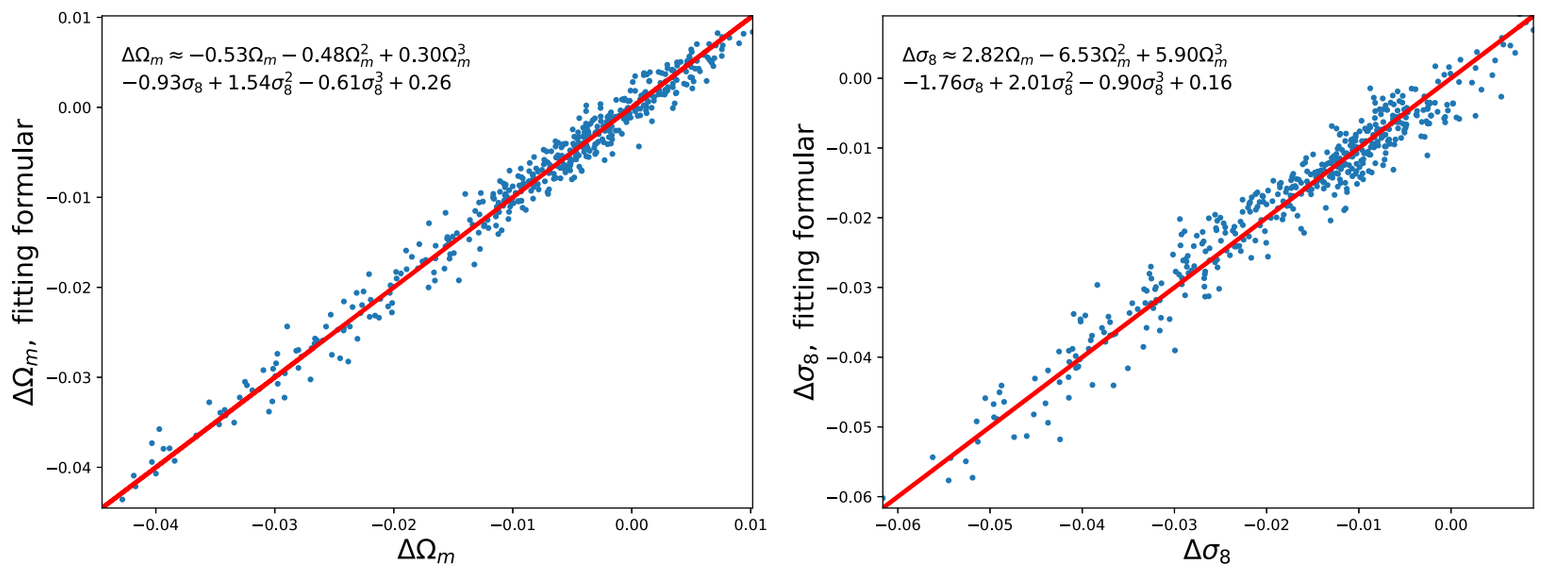


Deep Learning the Large Scale Structure



We found a significant bias in sigma8, this was found in previous works although not really highlighted

It can be easily corrected…



Pan et al. arXiv:1908.10590

Deep Learning the Large Scale Structure

CNN adds more information than 2nd order statistics

	Method	Training sample	Relative error of (Ω_m, σ_8)[a]
Ravanbakhsh et al. (Ravanbakhsh et al. 2017)	CNN	450 simulations $(512\ h^{-1}\mathrm{Mpc})^3$, 512^3 particles	(0.028, 0.012)
Ravanbakhsh et al. (Ravanbakhsh et al. 2017)	Power spectrum	450 simulations $(512\ h^{-1}\mathrm{Mpc})^3$, 512^3 particles	(0.072, 0.013)
This work	CNN	465 simulations $(256h^{-1}\mathrm{Mpc})^3$, 128^3 particles	(0.0048, 0.0053)
This work	2pcf, $s \in (0, 130)h^{-1}\mathrm{Mpc}$	465 simulations $(256h^{-1}\mathrm{Mpc})^3$, 128^3 particles	(0.017, 0.012)
This work	2pcf, $s \in (10, 130)h^{-1}\mathrm{Mpc}$	465 simulations $(256h^{-1}\mathrm{Mpc})^3$, 128^3 particles	(0.1, 0.06)

Pan et al. arXiv:1908.10590

Deep Learning the Large Scale Structure

In collaboration with the Korean Astronomy Machine Learning Group:
David Parkinson, Sungwook E. Hong, Srivatsan Sridhar, Shi Feng, Sangnam Park, Inku Park, Dongsu Bak, Jacobo Asorey Barreiro, Benjamin L'Hullier, James Jee, Arman Shafieloo, Hanwool Koo, Ryan Keeley +++

- ★ **Initiated a project in Korea to further develop this approach**
- ★ **By considering redshift evolution of the density field we aim to constrain expansion dynamics**
- ★ **We will consider more realistic tracers such as mock galaxy or weak lensing convergence fields**
- ★ **We will add systematics such as redshift error, photo-z etc, angular completeness, etc**
- ★ **How to mitigate nonlinear effects and other small scale physics that are difficult to model - clipping, field transforms, etc**

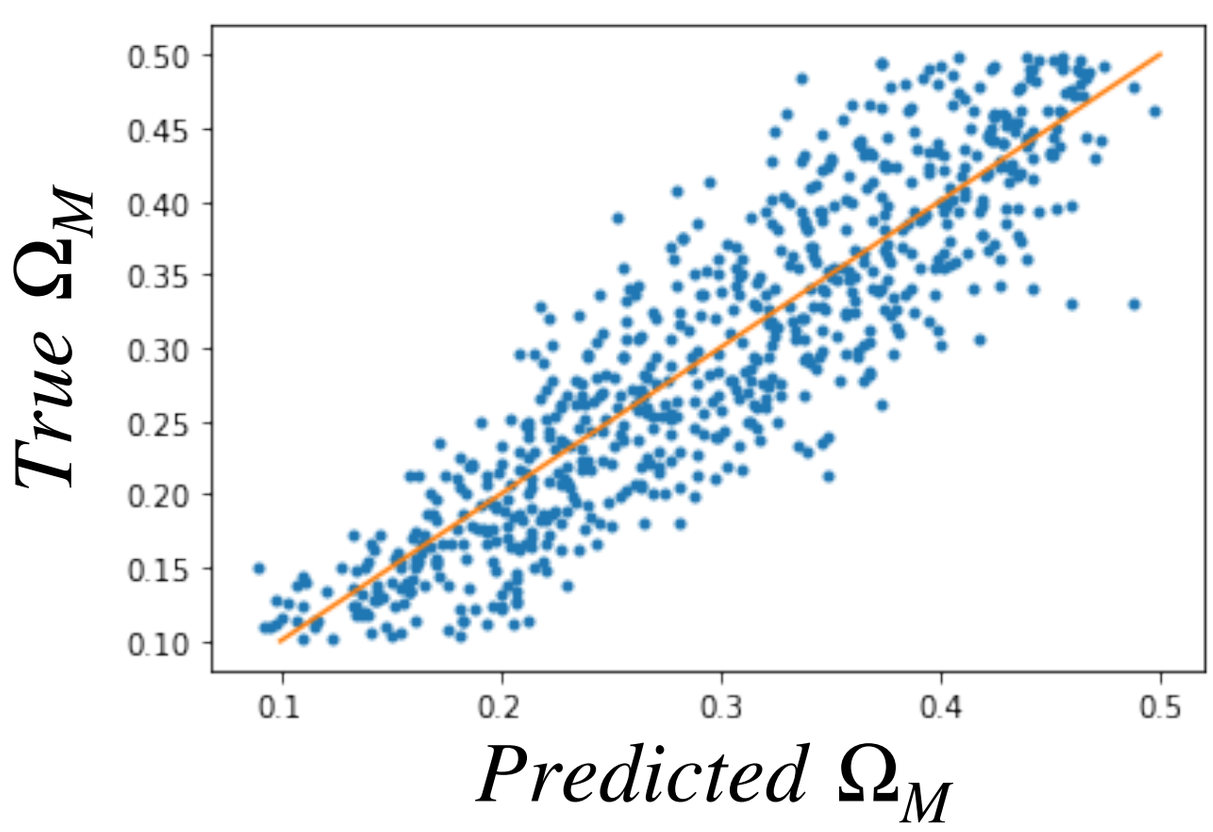
Information Content from CNN

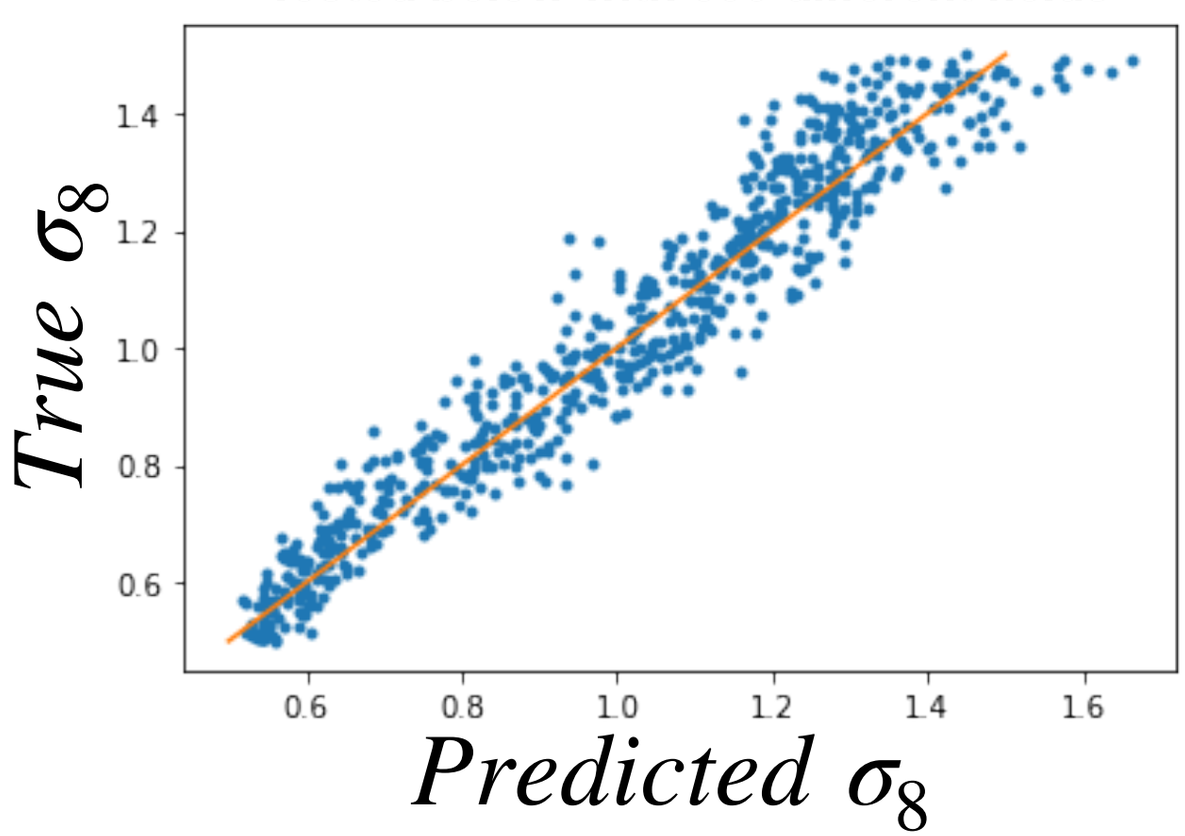
With Stephen Appleby

But first lets step back and look at Gaussian fields…
We know that their information is confined to ≤2nd order statistics
We can compute the errors on parameters from
1) monte carlo realisations
2) Fisher matrix

CNN Trained with 6,000 gaussian fields with linear cosmological power spectra

Tested below with 600 different fields





Information Content from CNN

With Stephen Appleby

But first lets step back and look at Gaussian fields…
We know that their information is confined to ≤2nd order statistics
We can compute the errors on parameters from

1) **monte carlo realisations**
2) **Fisher matrix**
3) **<u>Via the CNN directly?</u>**

We can write a custom loss function,

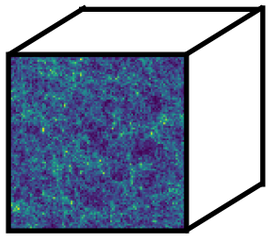
$$L = \frac{1}{2}\left(\ln\left(|\Sigma|\right) + \left(\theta_p - \theta_t\right)^T \Sigma^{-1} \left(\theta_p - \theta_t\right)\right),$$

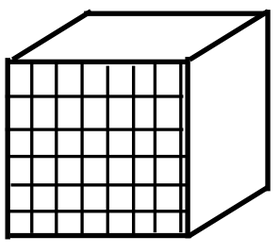
$$L = \begin{bmatrix} a & 0 \\ b & c \end{bmatrix} \qquad \Sigma^{-1} = LL^T,$$

Where the parameters a,b,c are learned by the network during training.
The network can learn the covariance matrix directly without an input ‘label’

Information Content from CNN

Cosmic Gaussian field

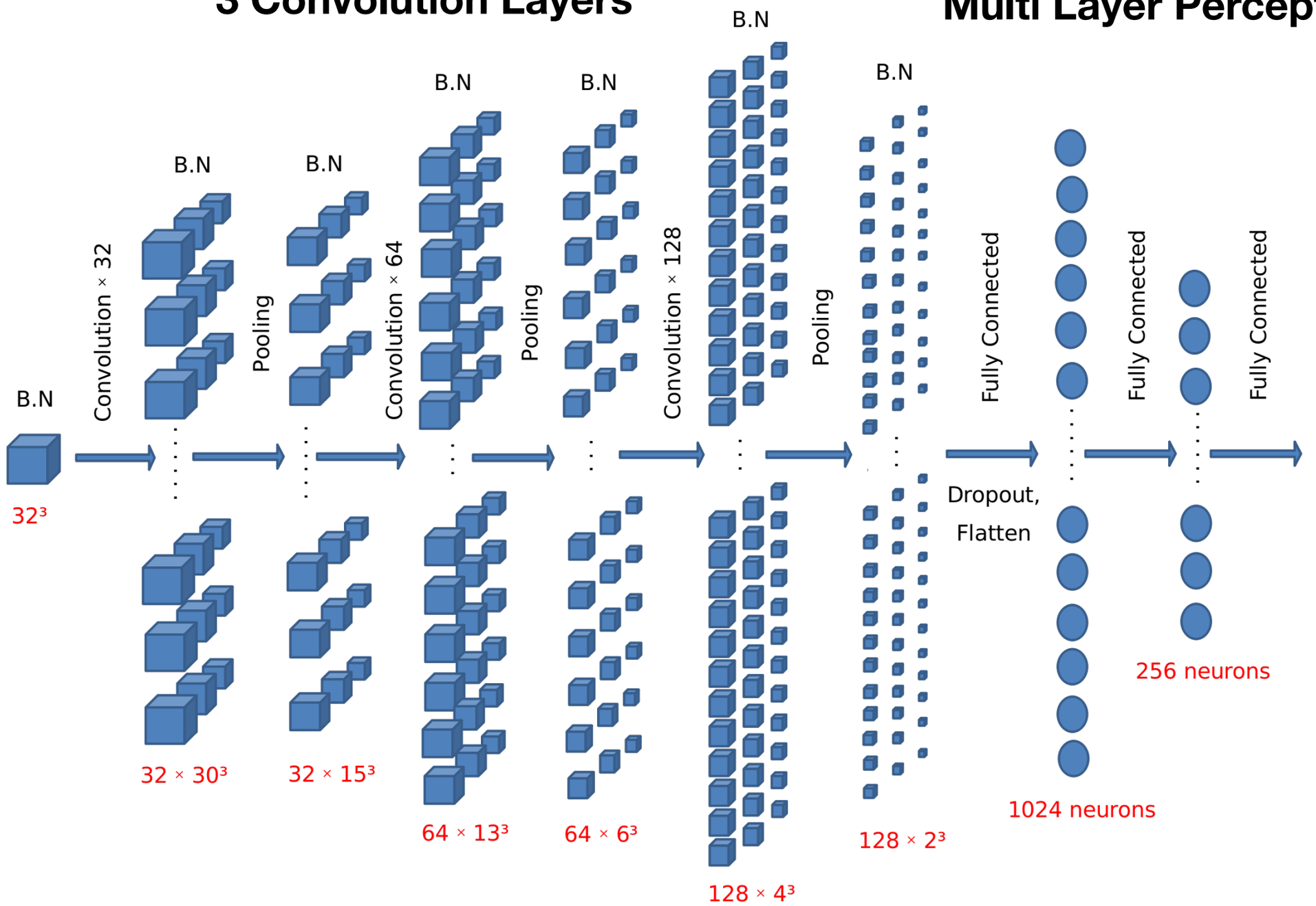




Neural Network Architecture

3 Convolution Layers

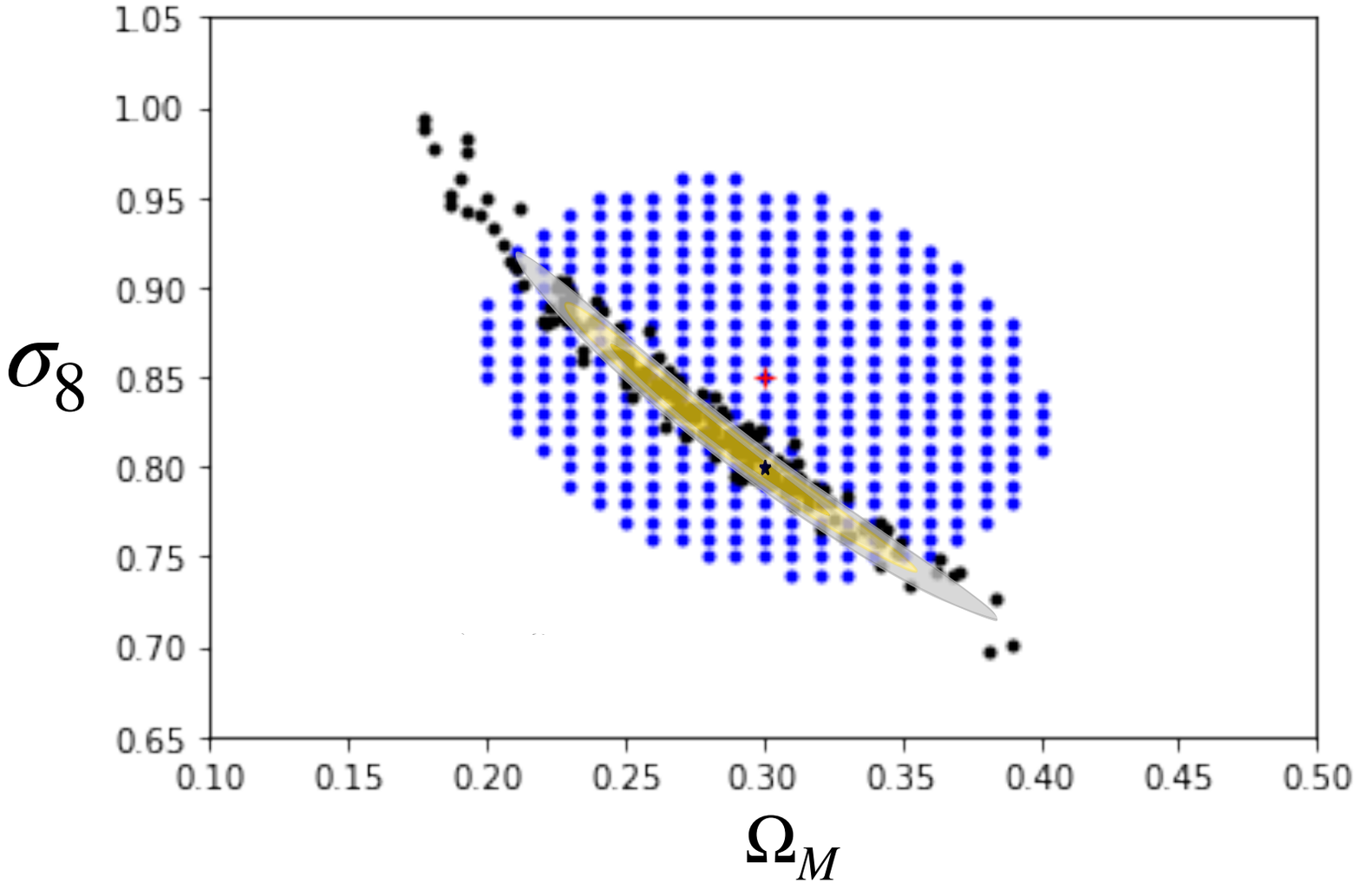
Multi Layer Perceptron



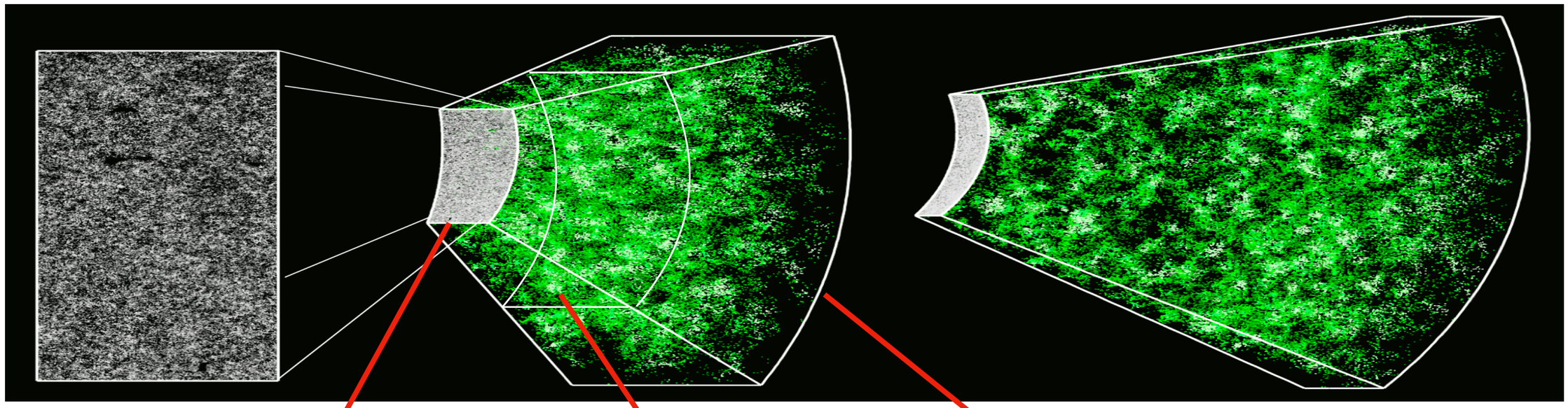
Information Content from CNN

Stepping back lets looks at Gaussian fields.
We know that their information is confined to ≤2nd order statistics
We can compute the errors on parameters from

1) **monte carlo realisations (black points)**
2) **Fisher matrix (yellow)**
3) **Via the CNN directly (blue)**

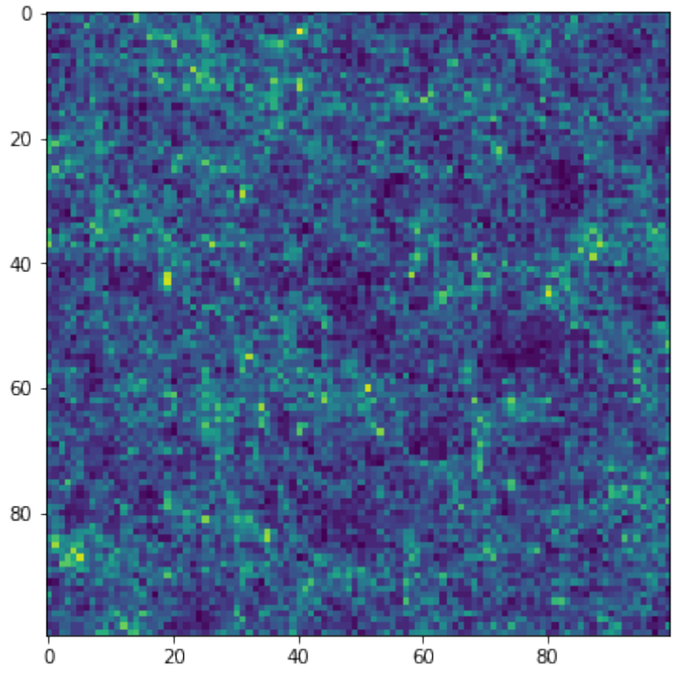


Deep Learning the Large Scale Structure

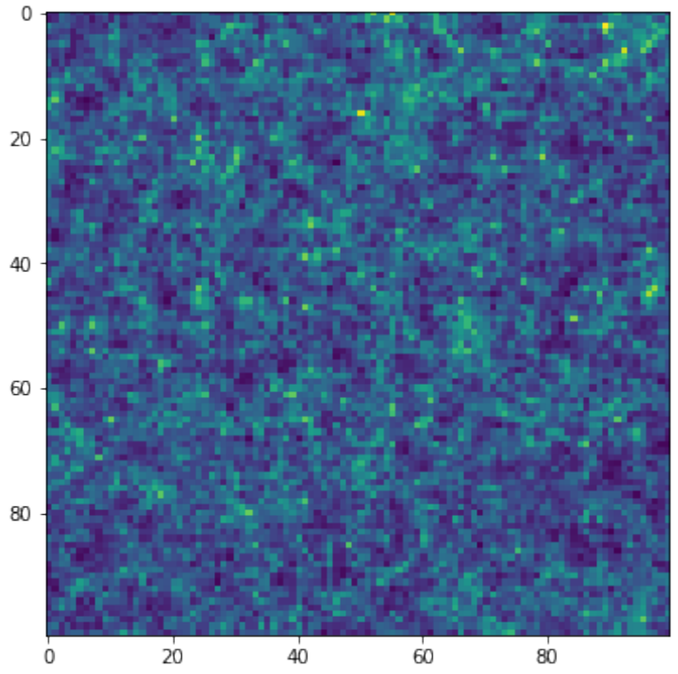


100x100sq deg

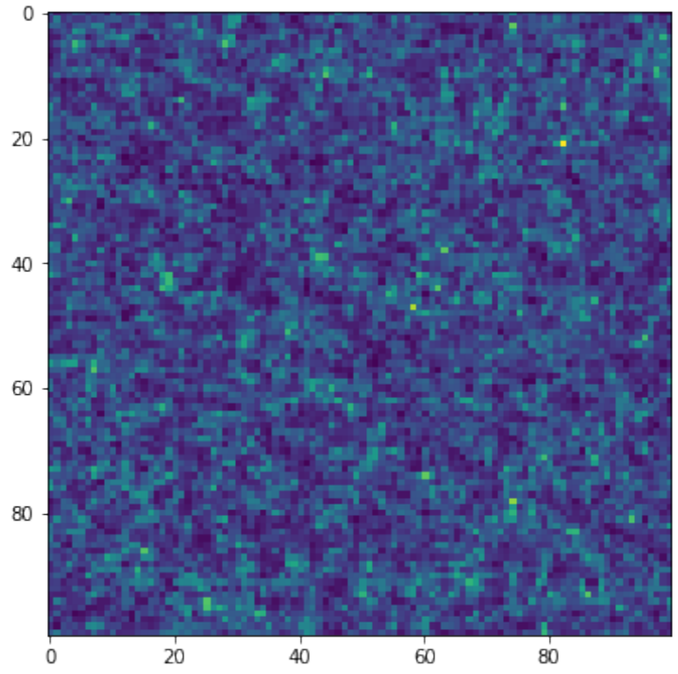
z=0



z=1



z=2

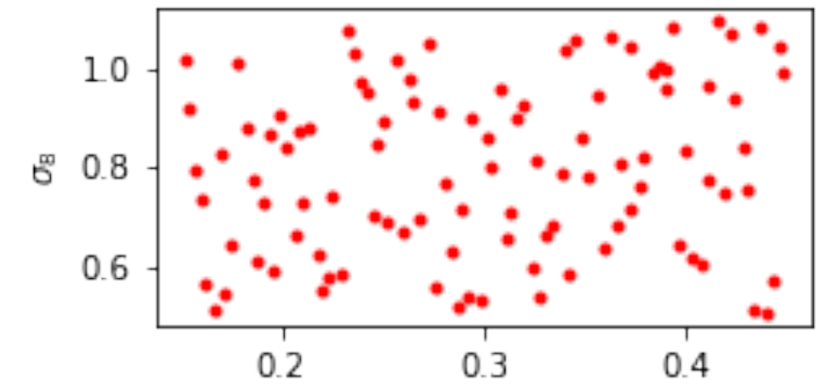


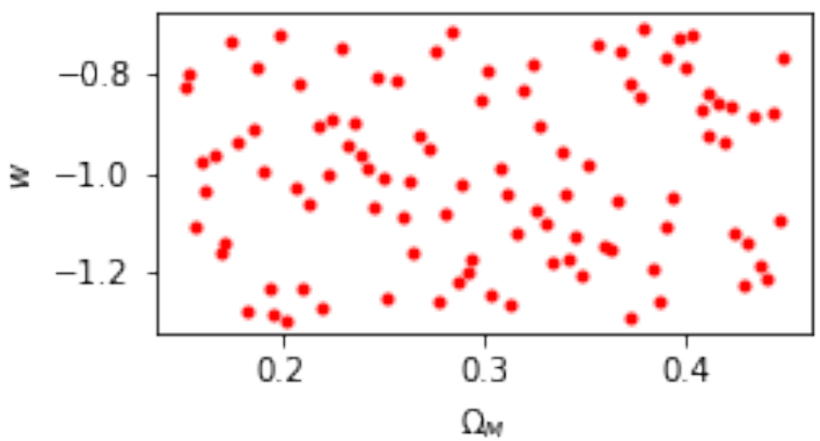
Deep Learning the Large Scale Structure

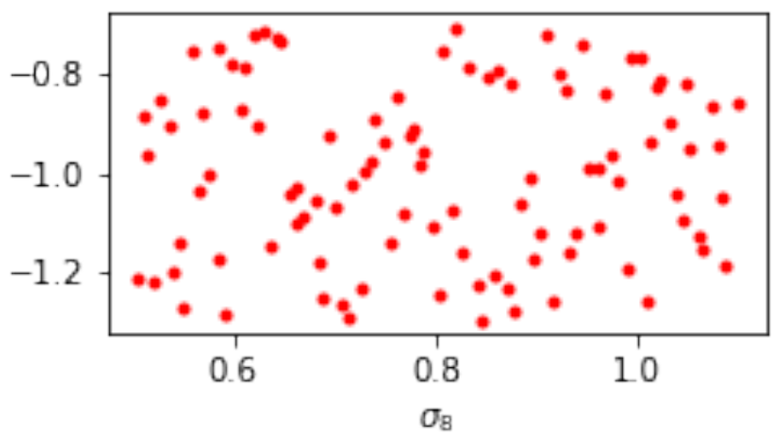
Fast approx N-body method - Pinocchio Code

2 Gpc/h box, 1024^3 particles
FoF halos
Light-cone output 0<z<2
6 redshift slices

So far 100 simulations run
Exploring Om, s8 and w
Parameters sampled from a orthogonal Latin Hypercube

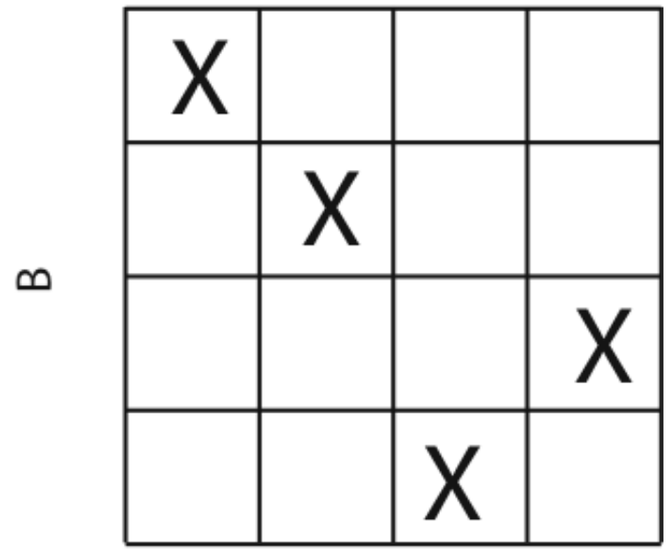






Training Data

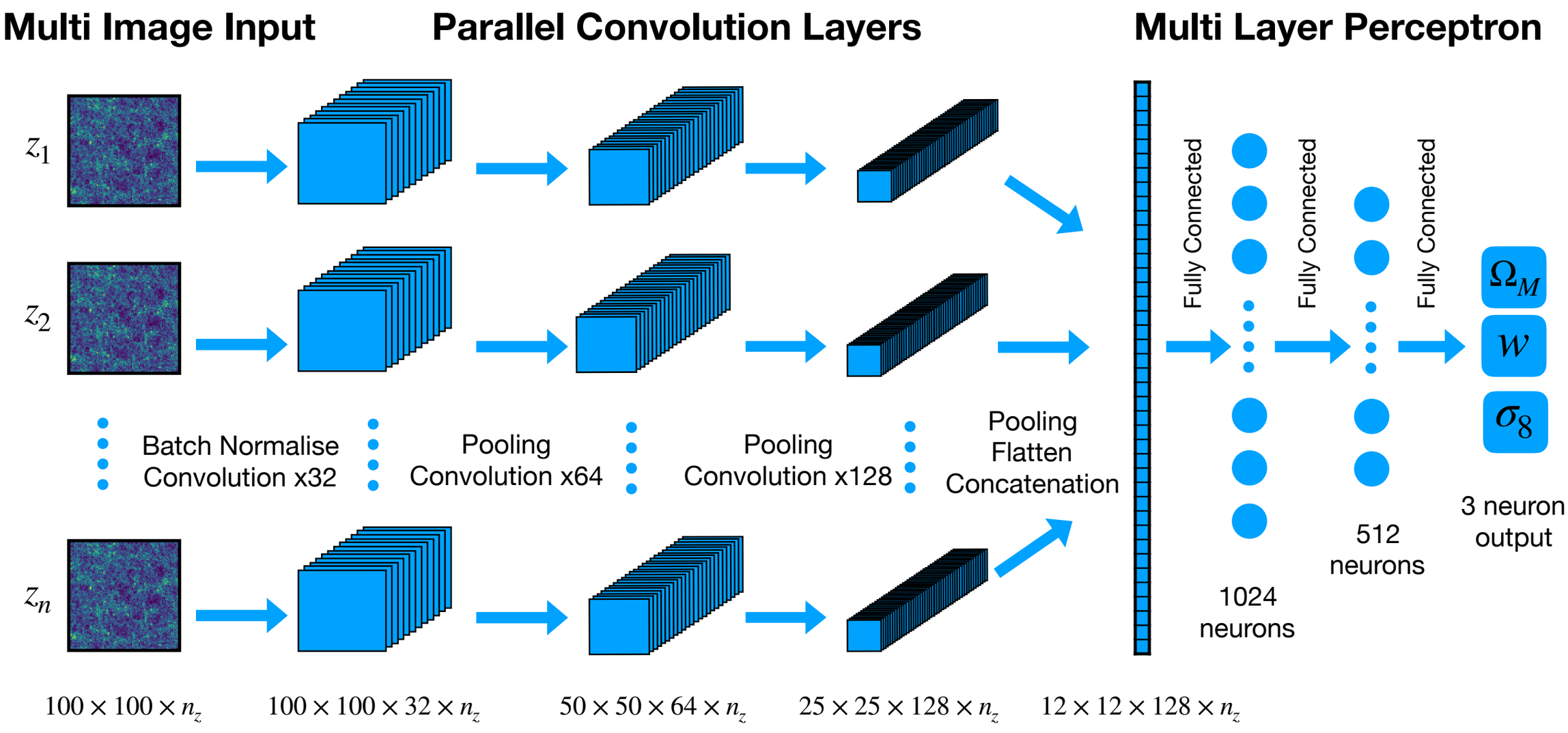
II



Chess: Rook on each row and file without threatening each other

Deep Learning the Large Scale Structure

Neural Network Architecture

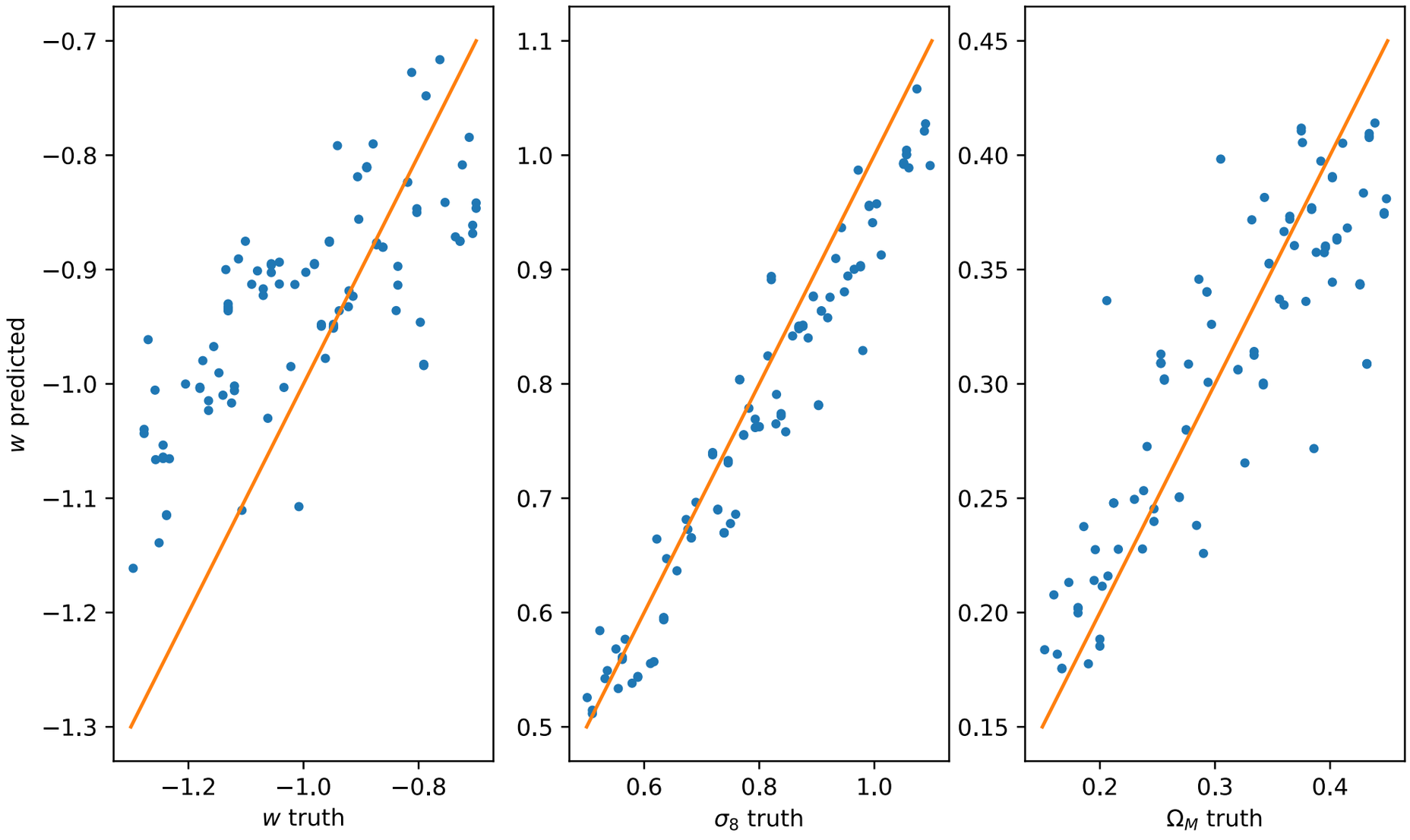


Deep Learning the Large Scale Structure

Preliminary Results

Promising results

- **More training data**
- **Experiment different architecture and hyper parameters**
- **Test redshift space distortions / photo-z errors / etc**



Deep Learning the Large Scale Structure

The Korea Institute of Science and Technology Information (KISTI) Supercomputer Center recently had an open call for proposals specifically aimed at Machine Learning and Big Data.

We were successful and awarded 15 million CPU hours on ‘Nurion’ (15th fastest in the world)

It is a Cray CS500 system with 570,020 cores

Proposal:
- run ~1000 simulations
- Create light-cone cones
- Find halos
- Apply a galaxy assignment model or several models (eg HODs)

- Add observational effects, redshift distortions, photo-z, incompleteness, etc

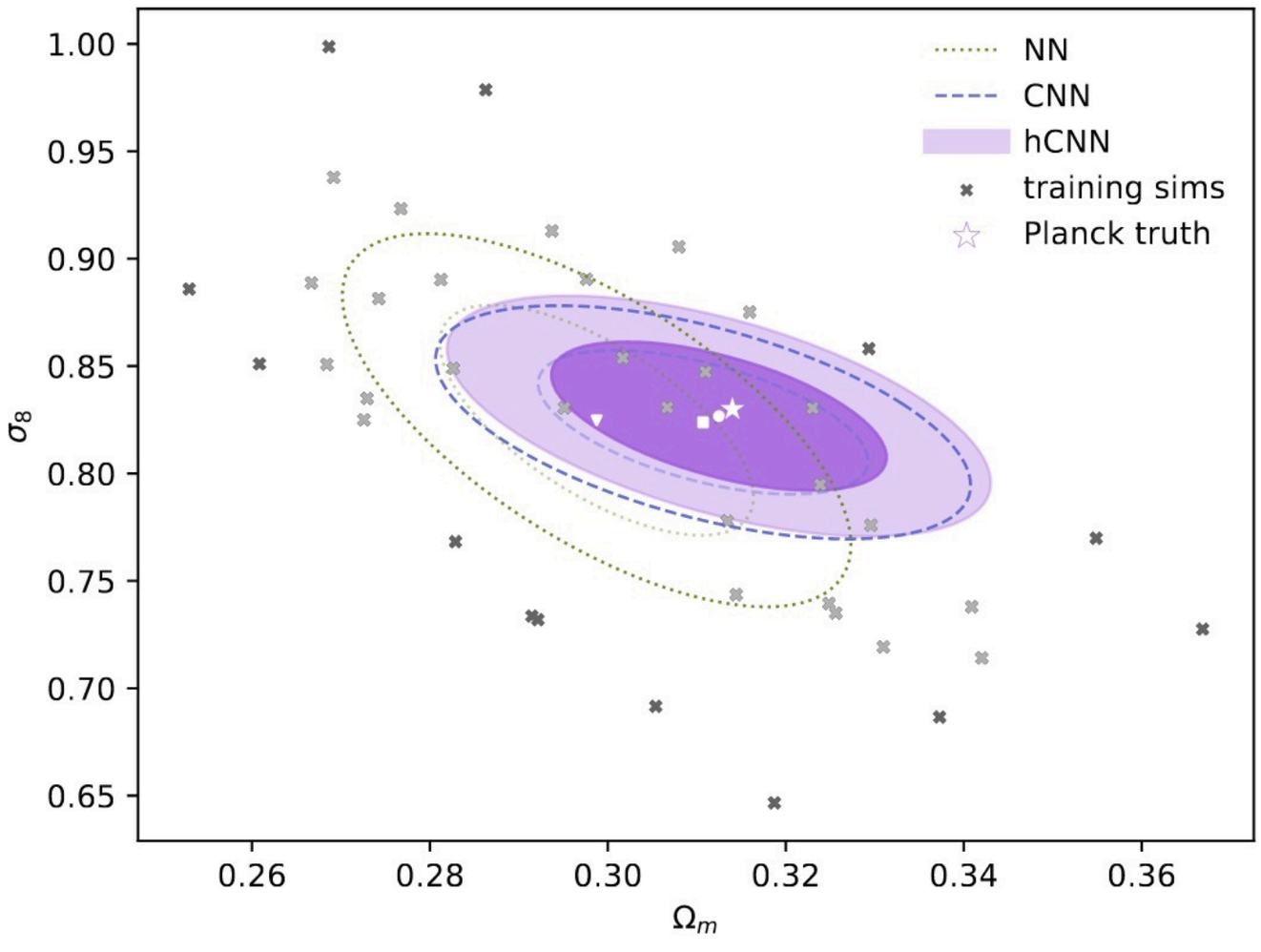
- Train models and constrain parameters within LCDM

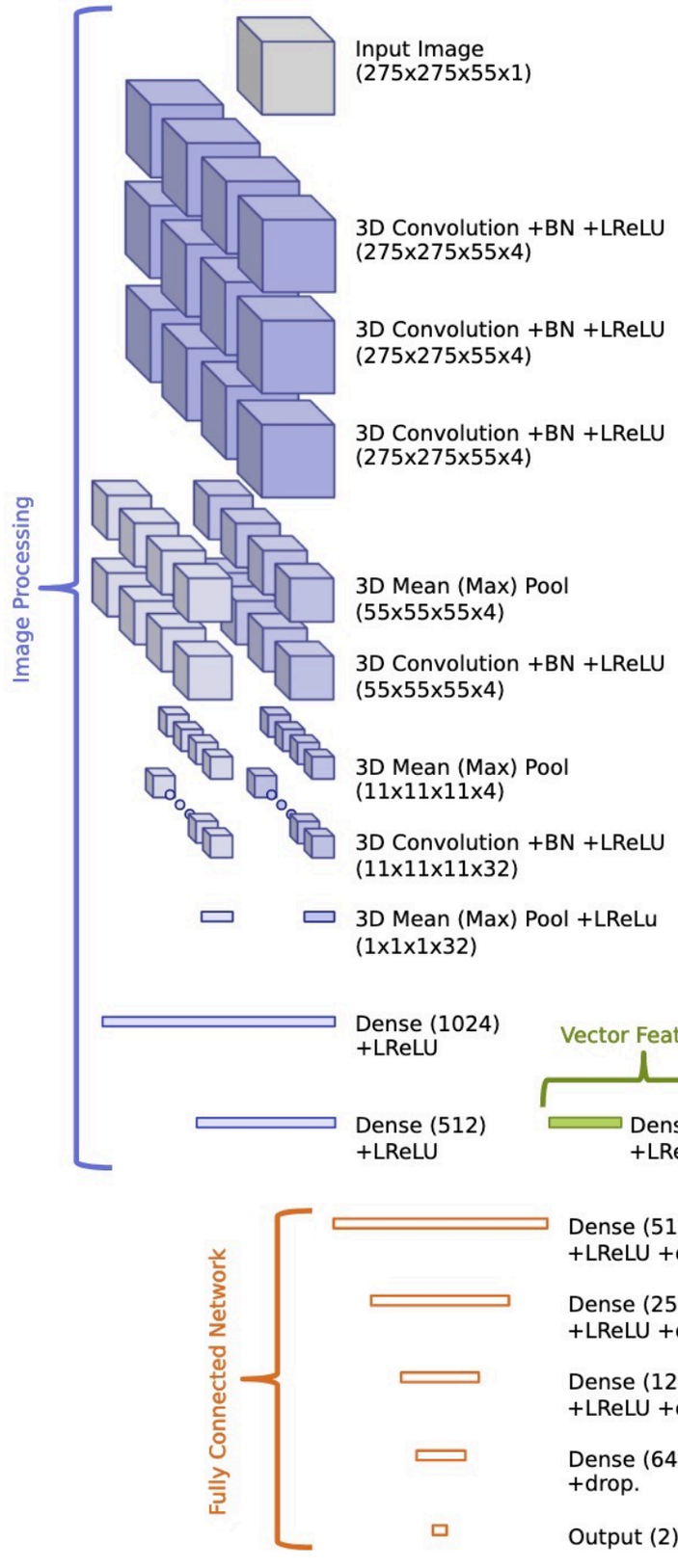
- Apply to observational data, KiDS, SDSS, DLS, early DESI data

Deep Learning the Large Scale Structure

Ntampaka, Eisenstein, et al. (2019), added mock galaxies to the dark matter halo according to an HOD prescription.

The added 2nd order clustering statistics making a hybrid CNN. But it did not add much information





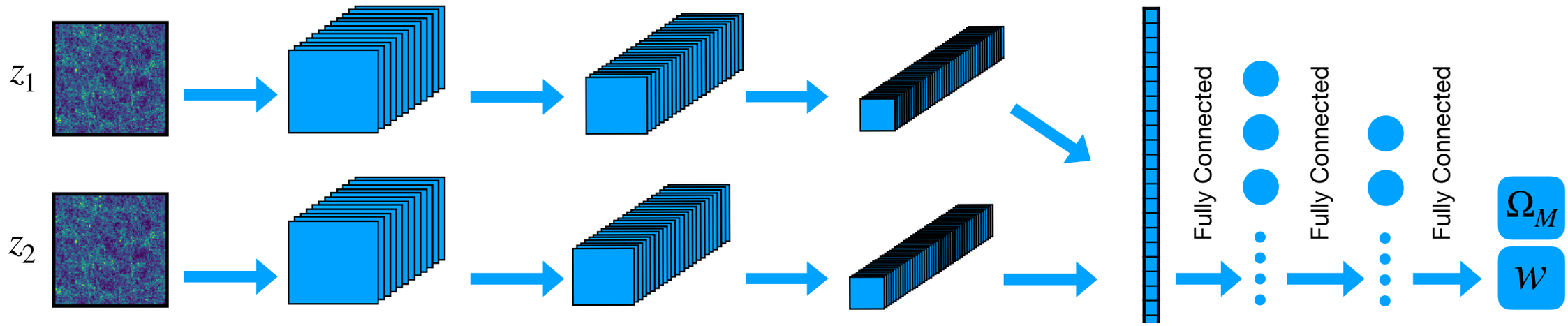
Deep Learning the Large Scale Structure

Hybrid Scheme with Higher order statistics

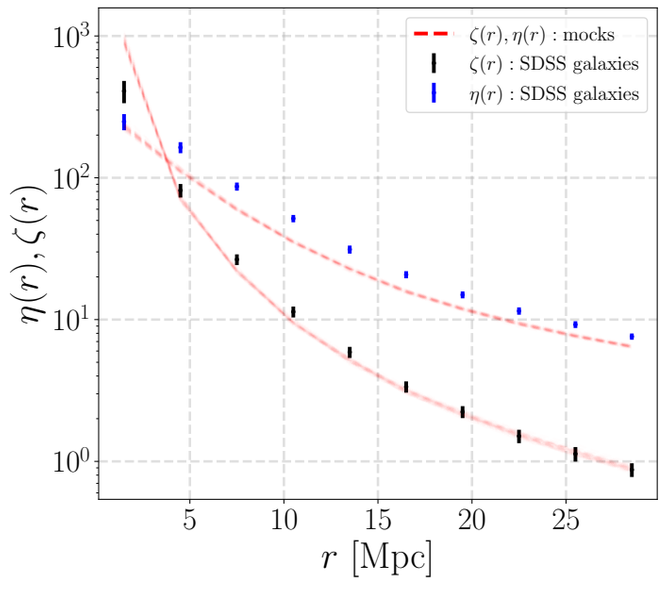
Multi Image Input

Parallel Convolution Layers

Multi Layer Perceptron



v1 out now!



+

3-pCF

4-pCF

★C. Sabiu, B. Hoyle, J. Kim, X-D Li

★https://arxiv.org/abs/1901.00296 ★ **http://bitbucket.org/csabiu/gramsci**

Deep Learning the Large Scale Structure

- ★ **Machine Learning the can do as well and potentially better than usual statistical methods for extracting cosmological information from the distribution of galaxies.**

- ★ **Preliminary results show that CNNs can learn features in the evolving density field to help us constrain parameters including Dark energy and its equation of state, w.**

- ★ **Higher Order Statistics beyond the 3rd order can now be computed in a tractable timeframe using graph databases with GRAMSCI: GRA**ph **M**ade **S**tatistics for **C**osmological **I**nformation **available from: http://bitbucket.org/csabiu/gramsci**

Thanks for your attention!

Extra

Generative Adversarial Networks

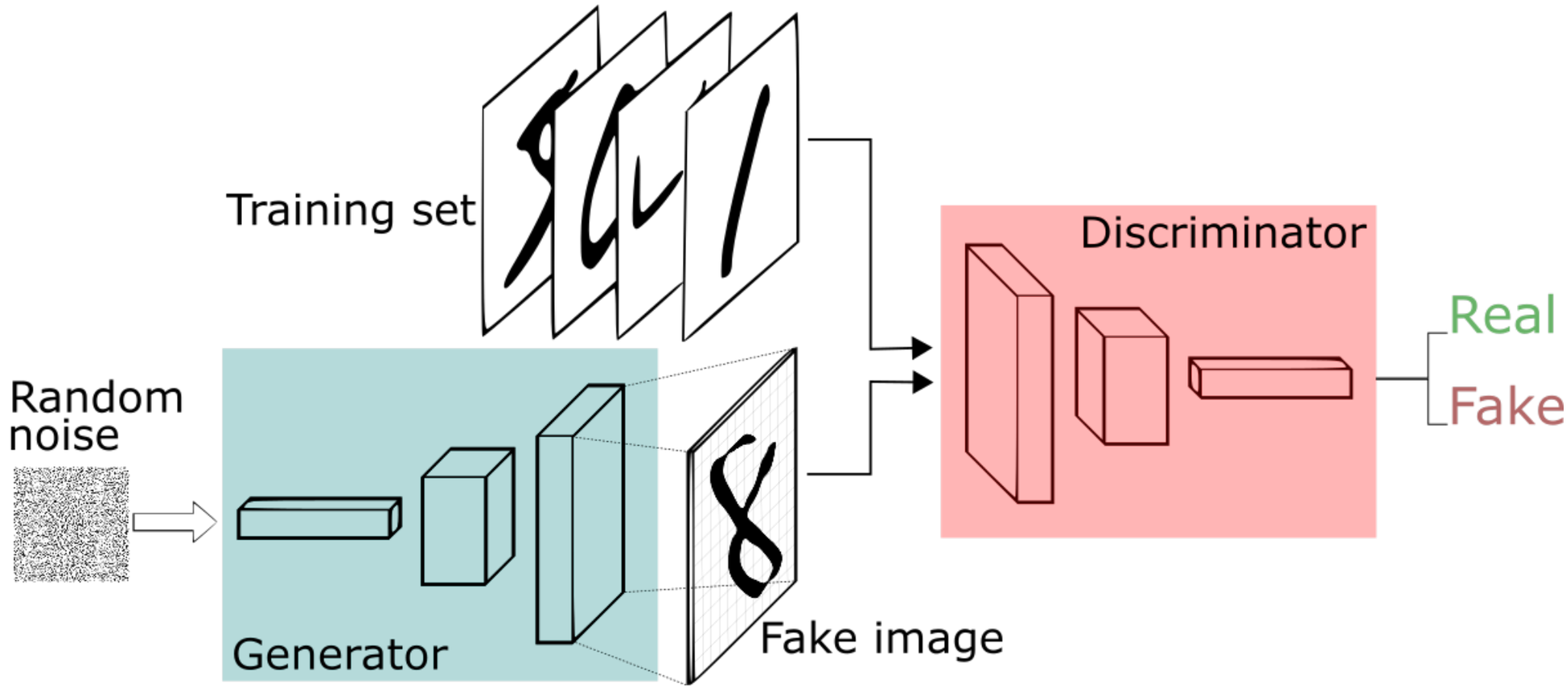


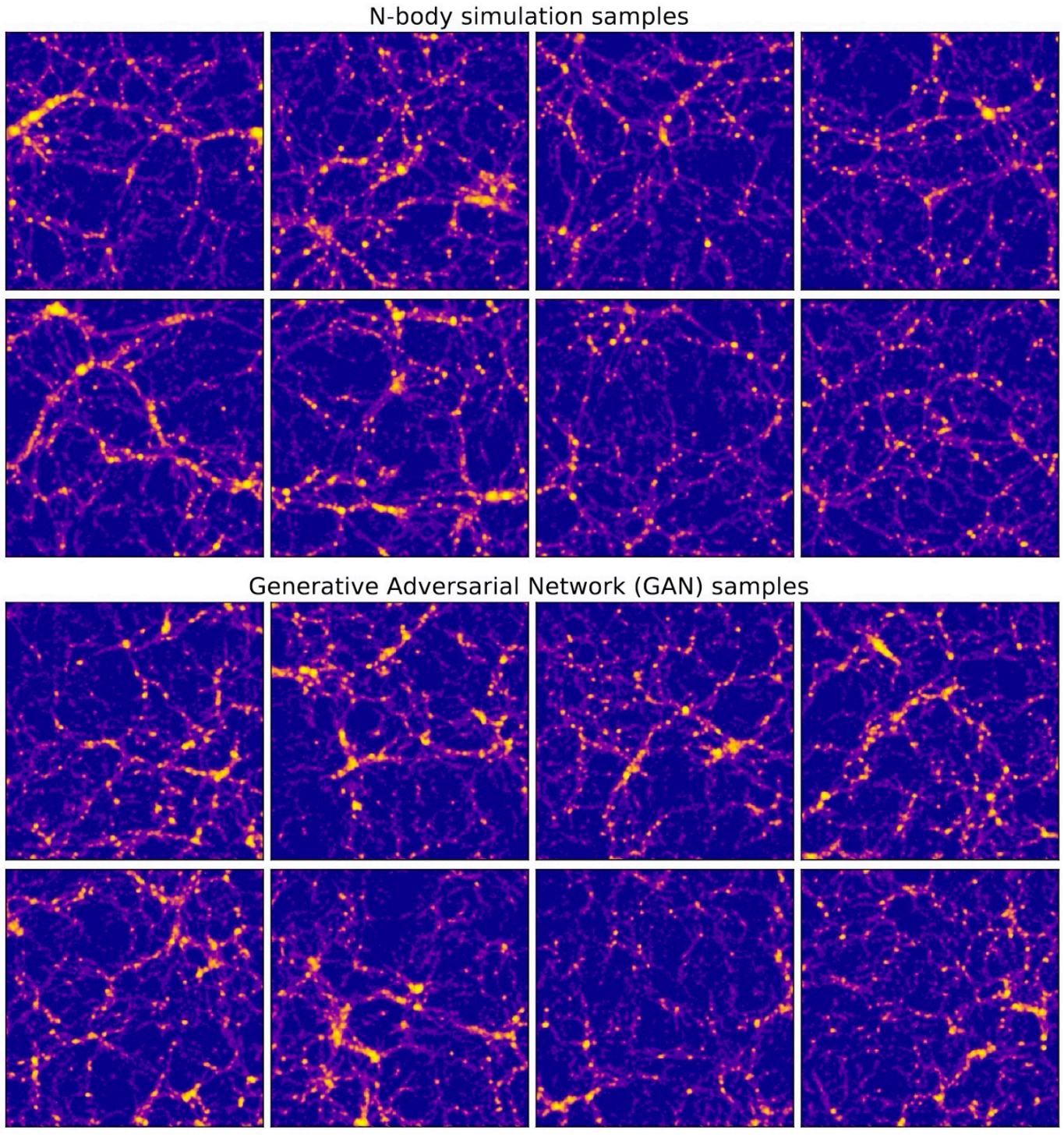
Image credit: Thalles Silva

The basic idea behind GANs consists in pairing-up two neural networks: a generator network G and a discriminator network D. These networks are trained in an adversarial game setting. The discriminator $D : \mathbf{x} \mapsto [0; 1]$ tries to probabilistically classify a sample $\mathbf{x}$ as being real or fake. On the other hand, the generator $G : \mathbf{z} \mapsto \mathbf{x}$ tries to generate samples that look like they were drawn from the true data distribution p_{data}. This generator makes use of a random variable $\mathbf{z}$ drawn from a given prior $p_{\text{prior}}(\mathbf{z})$ which is typically a Gaussian distribution. Formally, the two networks D and G play the following two-player minimax game:

$$\min_G \max_D [V(D, G)]$$

$$V(D, G) := \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\text{prior}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))],$$

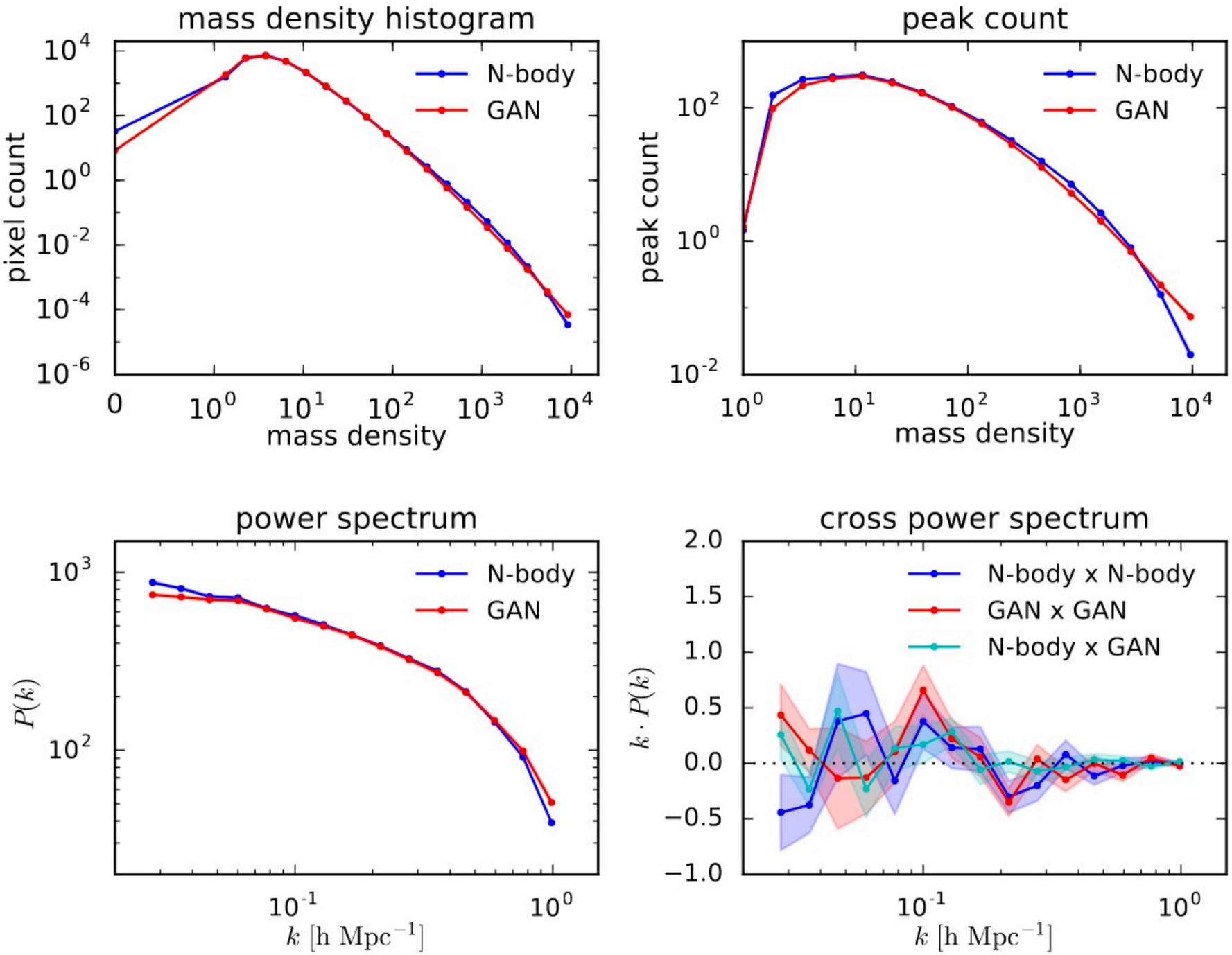
Generative Adversarial Networks



- **Optimising a GAN to reproduce N-body density fields of size 100Mpc**
- **Visually they match very well**
- **How about the statistics?**

Rodríguez et al., 2019 arXiv:1801.09070

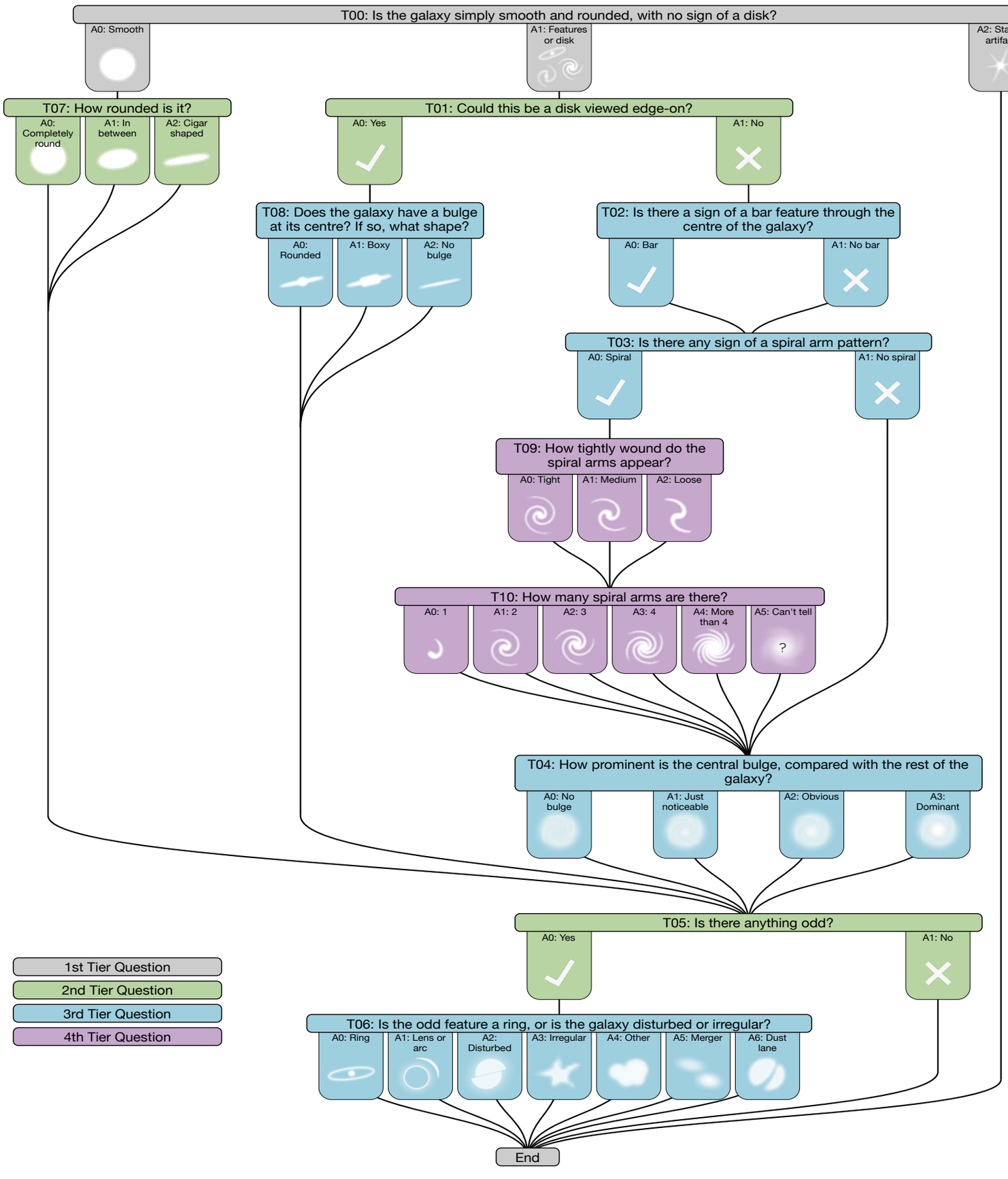
Generative Adversarial Networks



Rodríguez et al., 2019 arXiv:1801.09070

Galaxy Zoo

http://zoo1.galaxyzoo.org/



Members of the public classified galaxies with their eyes (and brains)

The crowdsourced science projected generated a lot of public interest.

1 Million galaxies from SDSS were given multiple classifications

It became possible to do large scale statistics with galaxy morphology!

But the next generation of survey LSST will observe ~37 billion stars and galaxies.

We need a lot more people or use ML!

Python Example

https://github.com/csabiu/

Pull requests Issues Marketplace Explore

ProTip! Updating your profile with your name, location, and a profile picture helps other GitHub users get to know you.

Edit profile

Overview Repositories 3 Projects 0 Stars 0 Followers 1 Following 0

Popular repositories

Customize your pins

kstat

Fast KD-tree MPI correlation statistics package - Author: Cristiano Sabiu

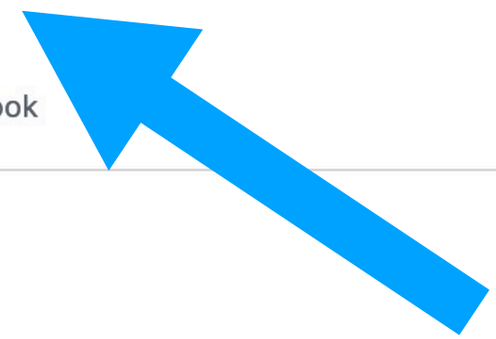
Fortran

Gramsci

GRAph Made Statistics for Cosmological Information

ML_tutorial

Jupyter Notebook



Python Example

csabiu / ML_tutorial

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Security | Insights | Settings

No description, website, or topics provided.

Edit

Manage topics

20 commits | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find File | Clone or download

csabiu Created using Colaboratory — Latest commit f62a647 3 days ago

GZ_classify.ipynb	Created using Colaboratory	3 days ago
class.tar.gz.partaa	Add files via upload	5 days ago
class.tar.gz.partab	Add files via upload	5 days ago
class.tar.gz.partac	Add files via upload	5 days ago
class.tar.gz.partad	Add files via upload	5 days ago

Open GZ_classify.ipynb

Python Example

csabiu / ML_tutorial

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Security | Insights | Settings

Branch: master | ML_tutorial / GZ_classify.ipynb | Find file | Copy path

csabiu Created using Colaboratory — f62a647 — 3 days ago

1 contributor

1090 lines (1090 sloc) | 374 KB | Raw | Blame | History

Open in Colab

Click here

Import the necessary packages

In [1]:
```
# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import os

print(tf.__version__)
```

```
1.13.1
```

Download data sample

Python Example

https://colab.research.google.com/github/csabiu/ML_tutorial/blob/master/GZ_classify.ipynb

Python Example

Import the necessary packages

```
[ ]  # TensorFlow and tf.keras
     import tensorflow as tf
     from tensorflow import keras

     # Helper libraries
     import numpy as np
     import matplotlib.pyplot as plt
     from PIL import Image
     import os

     print(tf.__version__)
```

```
1.13.1
```

Download data sample

```
[ ]  !wget -q https://github.com/csabiu/ML_tutorial/blob/master/class.tar.gz.parta{a,b,c,d,e,f,g,h,i}?raw=true > tmp
     !cat class.tar.gz.parta*true > class.tar.gz
     !gunzip class.tar.gz
     !tar -xvf class.tar > tmp
     !rm class.tar*
     !rm tmp
     !ls
```

```
class  sample_data
```

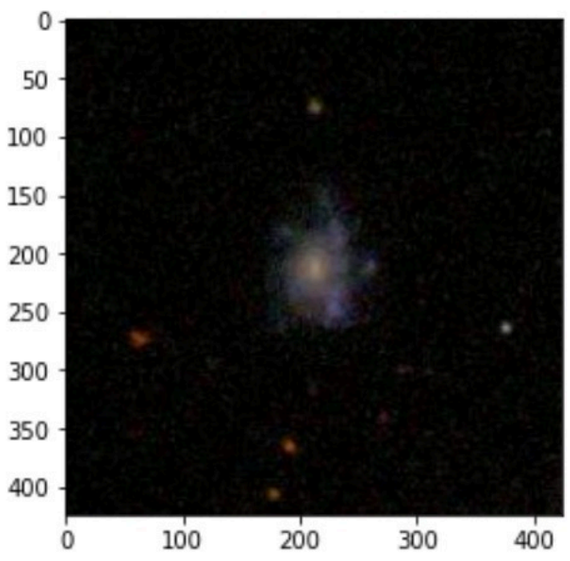
Python Example

Lets look at an image

shot

```
jpgfile = Image. open("class/100134.jpg")
plt.imshow(jpgfile)
print(np.shape(jpgfile))
```

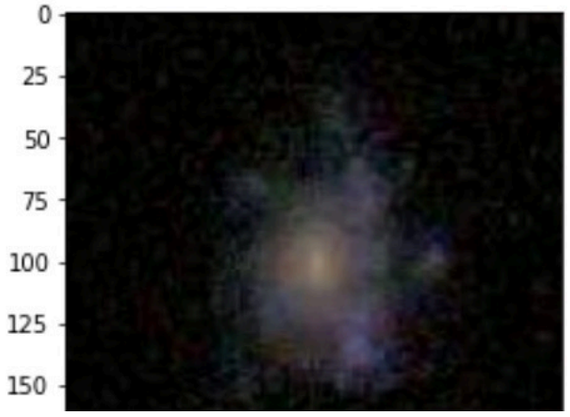
```
(424, 424, 3)
```



Lets crop it

```
[ ]  plt.imshow(jpgfile.crop((112,112,312,312)))
```

```
<matplotlib.image.AxesImage at 0x7f3662676208>
```



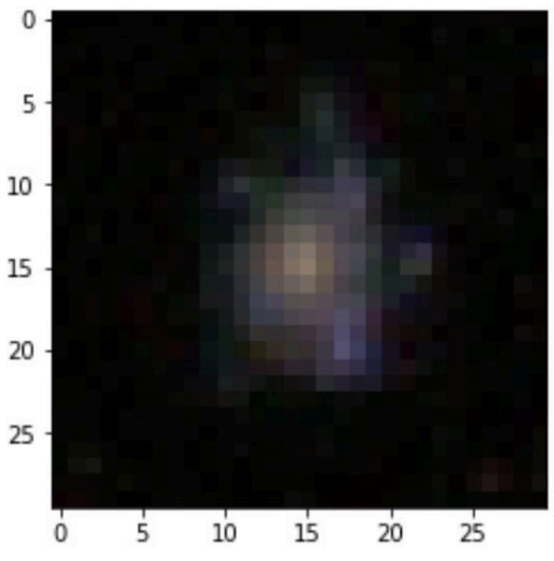
Python Example

And lower the resolution

shot

```
plt.imshow(jpgfile.crop((112,112,312,312)).resize((30,30),Image.ANTIALIAS))
```

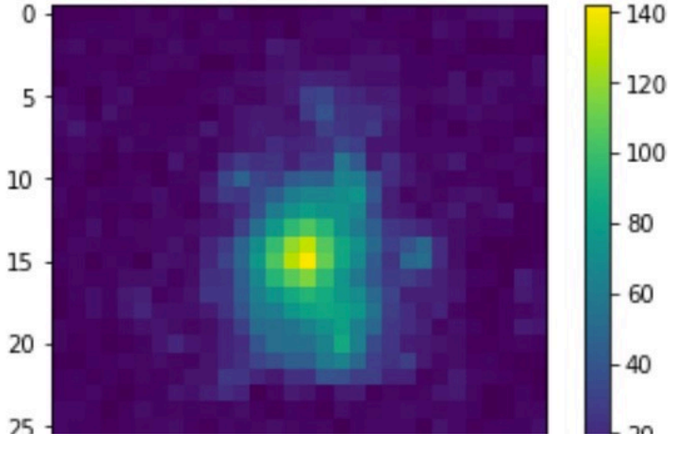
```
<matplotlib.image.AxesImage at 0x7f3660e4eeb8>
```



Split into RGB colors

```
r,g,b=jpgfile.split()
plt.imshow(r.crop((112,112,312,312)).resize((30,30),Image.ANTIALIAS))
plt.colorbar()
```

```
<matplotlib.colorbar.Colorbar at 0x7f3660d58dd8>
```



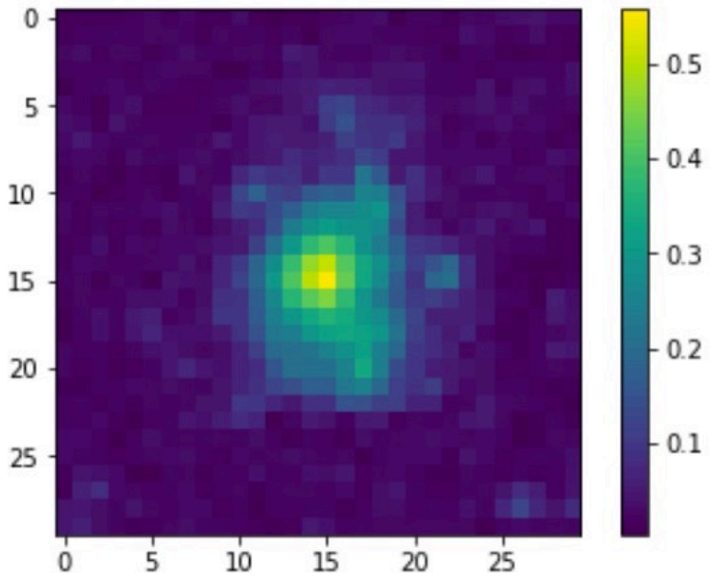
Python Example

Normalise the pixel values to (0,1)

shot

```
im=np.zeros((30,30))
im[:,:]=r.crop((112,112,312,312)).resize((30,30),Image.ANTIALIAS)
im=im/255.
plt.imshow(im)
plt.colorbar()
```

```
<matplotlib.colorbar.Colorbar at 0x7fe7907426a0>
```



Load the labels (truth) data

(Image ID, galaxy type) - 0=smooth, 1=featured

```
data=np.loadtxt("class/truth.txt",dtype='i')
print(np.shape(data))
labels=data[:,1]

print(data[1:10,:]) # print first 10 entries
```

```
(16885, 2)
[[100134      1]
 [100322      1]
```

Python Example

Lets transform all the images and save into an array

shot

```
[ ]  images=np.zeros((16885,30,30,3))
     nn=0

     for i in (data[:,0]):
       filename=str(int(i))+".jpg"
       jpgfile = Image. open("./class/"+filename)
       images[nn,:,:,:]=jpgfile.crop((112,112,312,312)).resize((30,30),Image.ANTIALIAS)
       nn=nn+1

     images=images/255.
     print(np.shape(images))
```

```
(16885, 30, 30, 3)
```

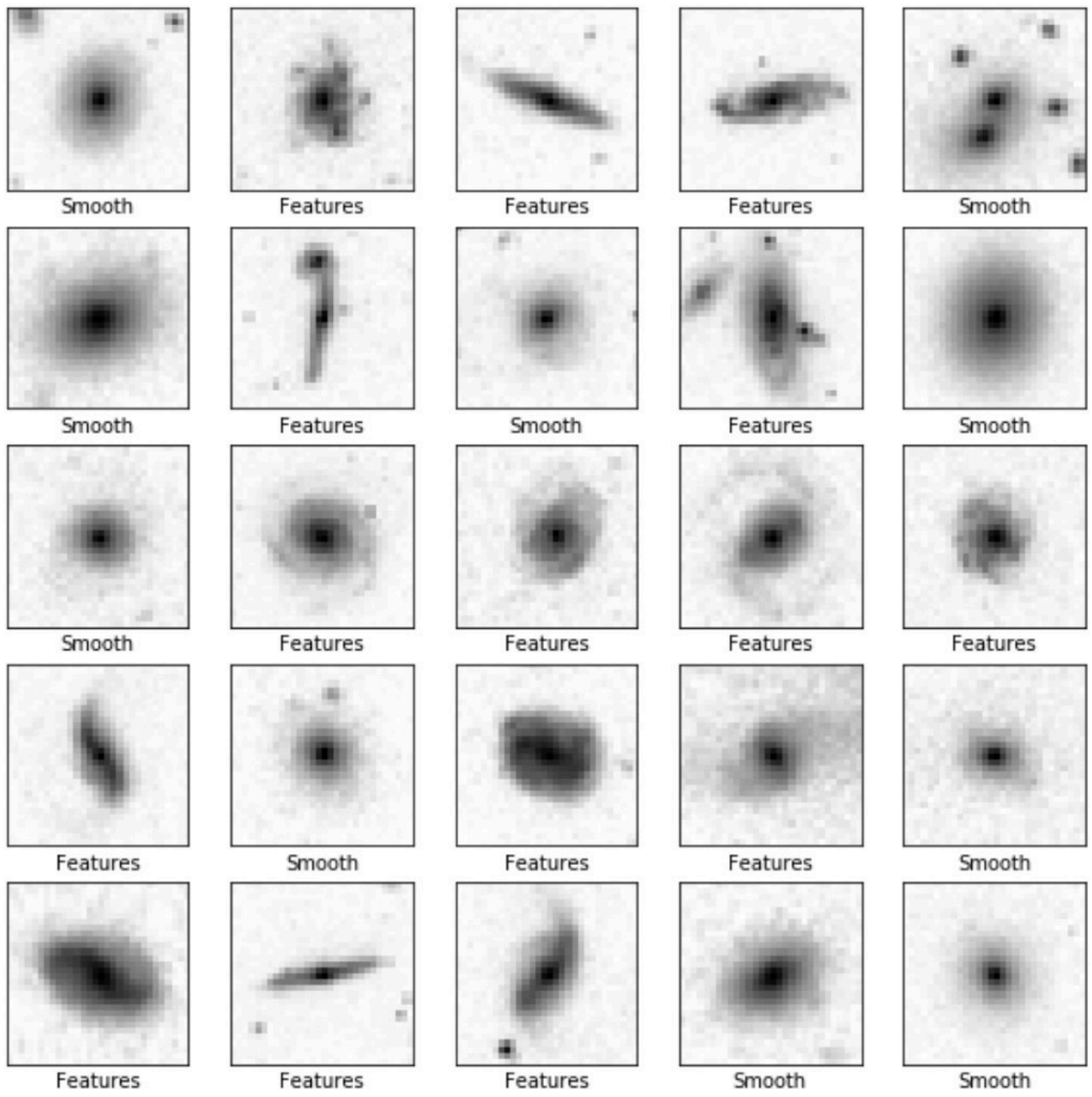
Split images and labels into training and test data

```
[ ]  images=images.mean(axis=3) # average colour -> greyscale
     train_data=images[:13000,:,:]
     test_data=images[13000:,:,:]

     train_label=labels[:13000]
     test_label=labels[13000:]

     print(np.shape(train_data))
     print(np.shape(train_label))
     print(np.shape(test_data))
     print(np.shape(test_label))
```

```
(13000, 30, 30)
(13000,)
(3885, 30, 30)
(3885,)
```

Python Example

```
[ ]  class_names = ['Smooth', 'Features']
     plt.figure(figsize=(10,10))
     for i in range(25):
         plt.subplot(5,5,i+1)
         plt.xticks([])
         plt.yticks([])
         plt.grid(False)
         plt.imshow(train_data[i], cmap=plt.cm.binary)
         plt.xlabel(class_names[int(train_label[i])])
     plt.show()
```



Python Example

Define a simple neural network

shot

```
[ ]  from keras.utils.vis_utils import plot_model
     keras.backend.clear_session()
     model = keras.Sequential([
         keras.layers.Flatten(input_shape=(30, 30)),
         keras.layers.Dense(64, activation=tf.nn.relu),
         keras.layers.Dense(2, activation=tf.nn.softmax)])
     model.summary()

     model.compile(optimizer='adam',
                   loss='sparse_categorical_crossentropy',
                   metrics=['accuracy'])
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/
Instructions for updating:
Colocations handled automatically by placer.
Using TensorFlow backend.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 900)               0
_________________________________________________________________
dense (Dense)                (None, 64)                57664
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 130
=================================================================
Total params: 57,794
Trainable params: 57,794
Non-trainable params: 0
_________________________________________________________________
```

Python Example

```
[ ]  model.fit(train_data[:,:,:], train_label, epochs=7)
```

```
Epoch 1/7
13000/13000 [==============================] - 1s 96us/sample - loss: 0.5173 - acc: 0.7727
Epoch 2/7
13000/13000 [==============================] - 1s 72us/sample - loss: 0.4068 - acc: 0.8407
Epoch 3/7
13000/13000 [==============================] - 1s 73us/sample - loss: 0.3689 - acc: 0.8573
Epoch 4/7
13000/13000 [==============================] - 1s 72us/sample - loss: 0.3510 - acc: 0.8630
Epoch 5/7
13000/13000 [==============================] - 1s 74us/sample - loss: 0.3323 - acc: 0.8722
Epoch 6/7
13000/13000 [==============================] - 1s 74us/sample - loss: 0.3243 - acc: 0.8740
Epoch 7/7
13000/13000 [==============================] - 1s 73us/sample - loss: 0.3088 - acc: 0.8816
<tensorflow.python.keras.callbacks.History at 0x7fe790509080>
```

```
[ ]  test_loss, test_acc = model.evaluate(test_data[:,:,:], test_label)
     print('Test accuracy:', test_acc)
```

```
3885/3885 [==============================] - 0s 46us/sample - loss: 0.3299 - acc: 0.8798
Test accuracy: 0.87979406
```

Python Example

Make predictions on test data from the trained model

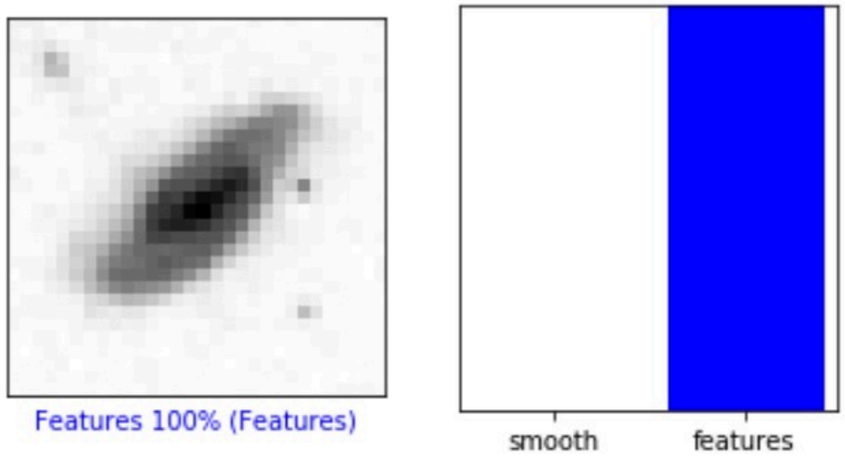
shot

```
[ ]  predictions = (model.predict(test_data[:,:,:]))

     print("First 10 galaxies")
     print("Predicted:",np.argmax(predictions[1:11],axis=1))
     print("Truth:    ",np.int_(test_label[1:11]))
```

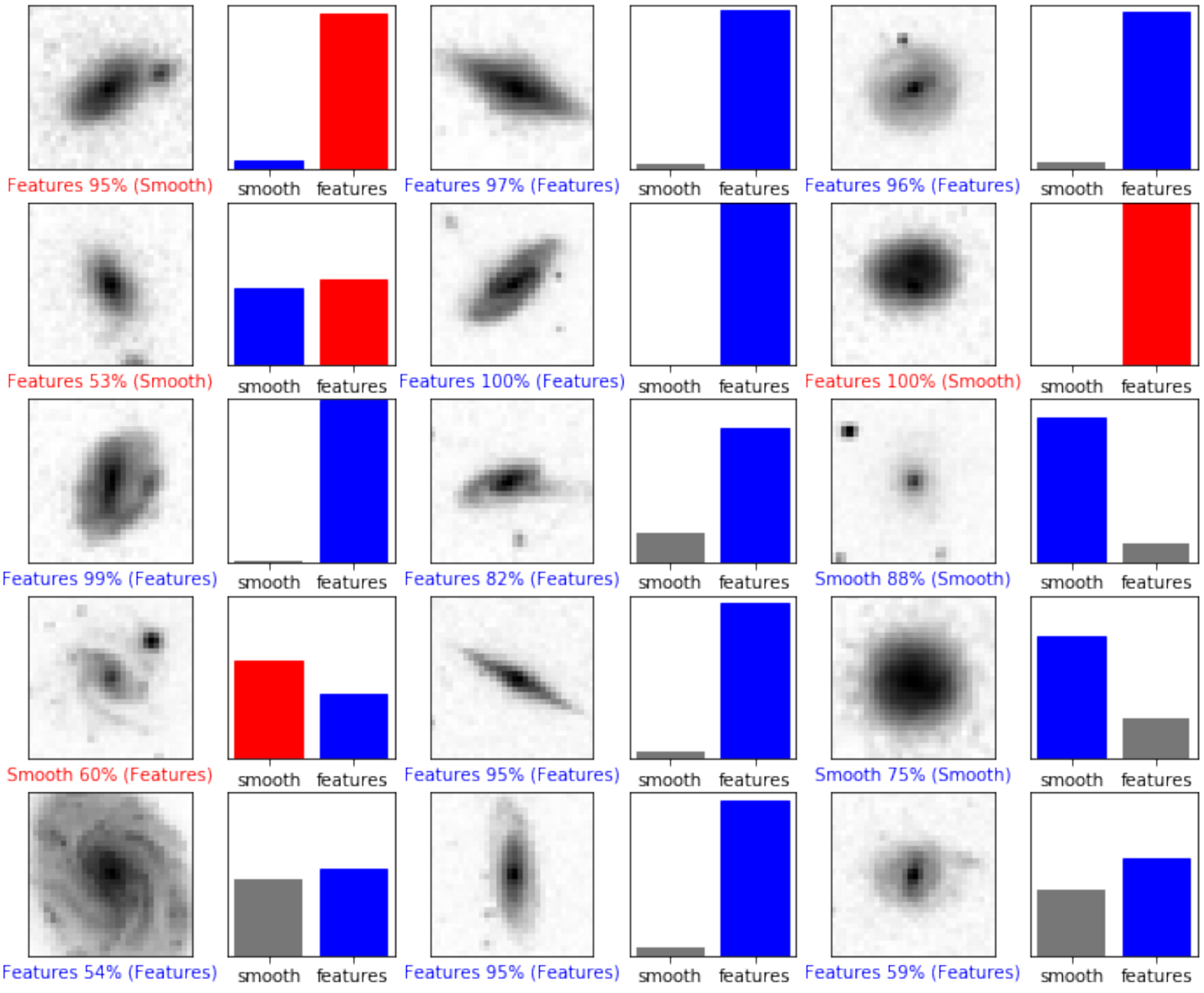
```
First 10 galaxies
Predicted: [1 1 1 1 1 1 1 0 0 1]
Truth:     [1 1 0 1 0 1 1 0 1 1]
```

Lets visialise these classifications

```
[ ]  i = 4
     plt.figure(figsize=(6,3))
     plt.subplot(1,2,1)
     plot_image(i, predictions, np.int_(test_label), test_data)
     f=plt.subplot(1,2,2)
     plot_value_array(i, predictions,  np.int_(test_label))
     f.axes.set_xticklabels(["smooth","features"])
     plt.show()
```

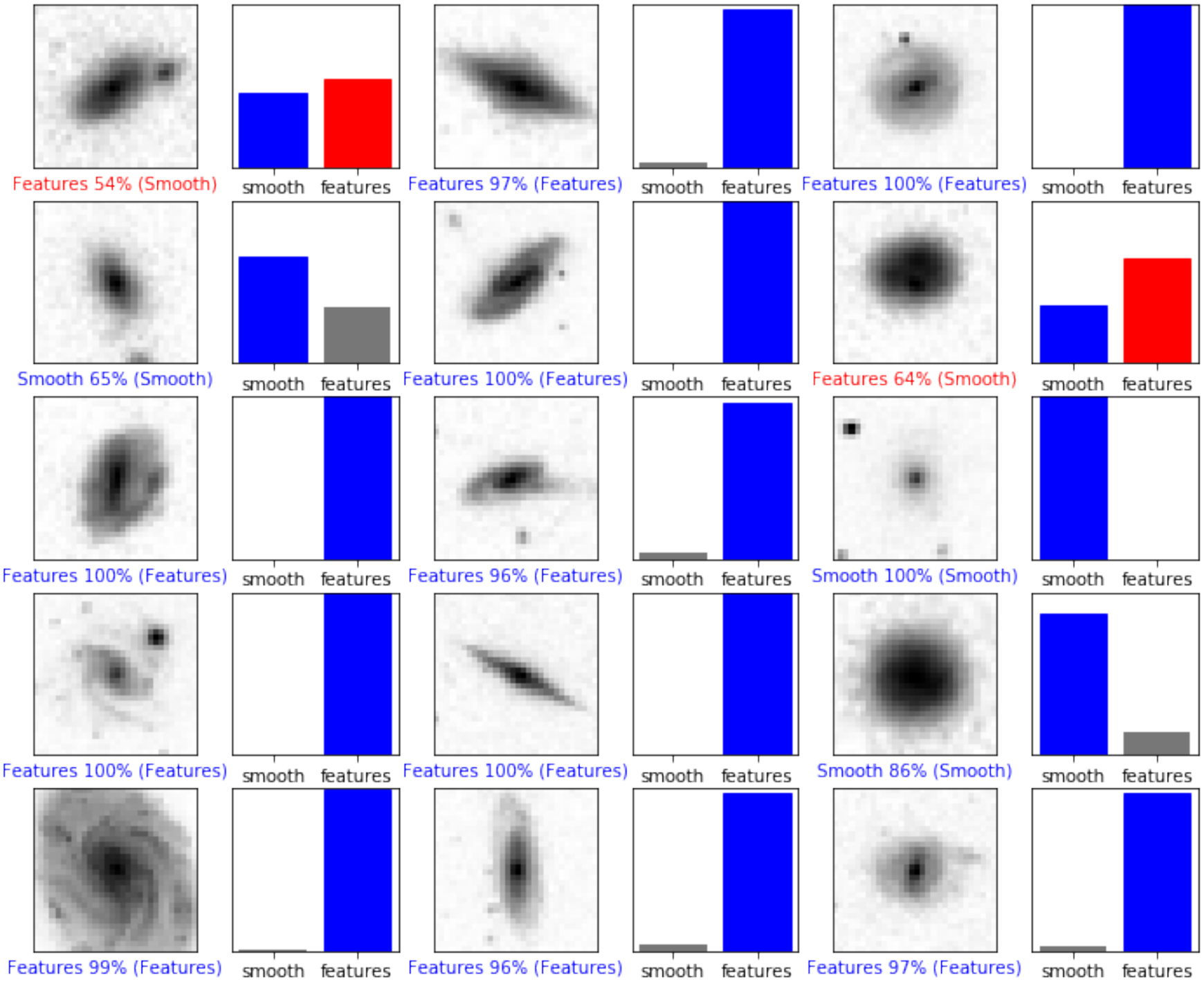


Python Example



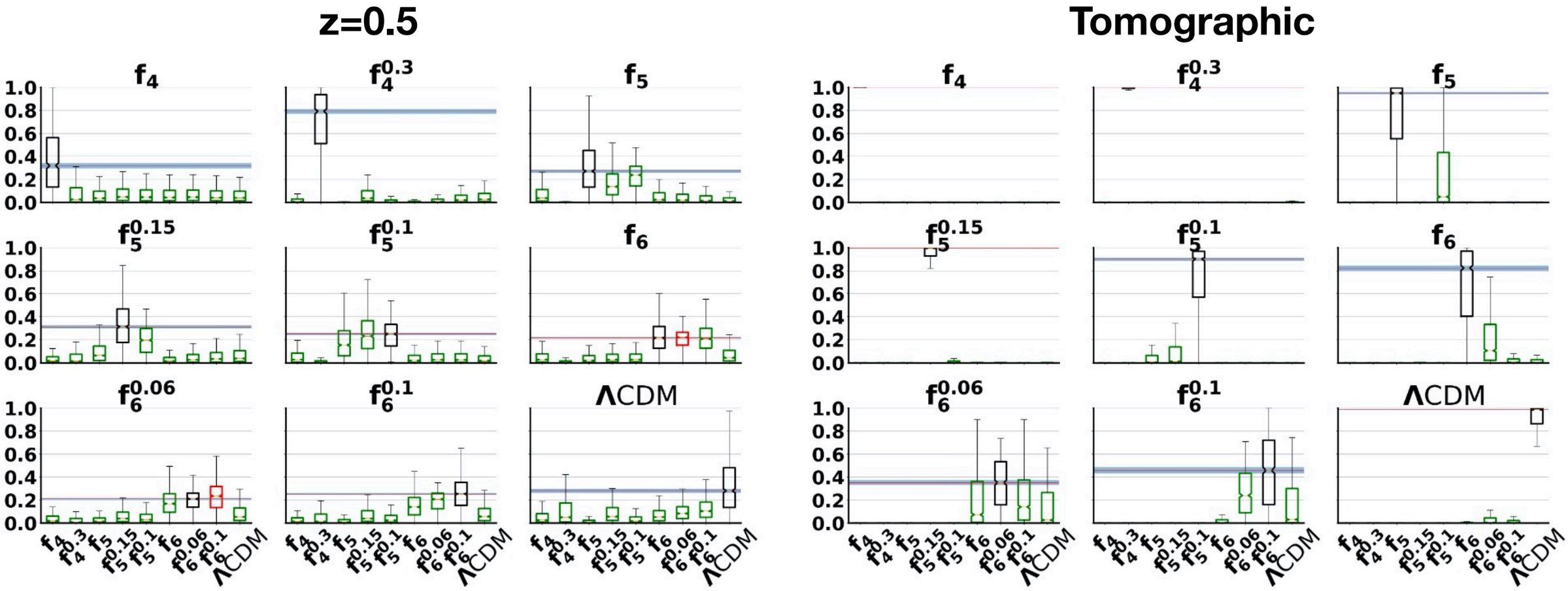
Python Example

Now using a convolutional neural network!



Deep Learning the Large Scale Structure

CNN applied to Convergence maps in fR gravity

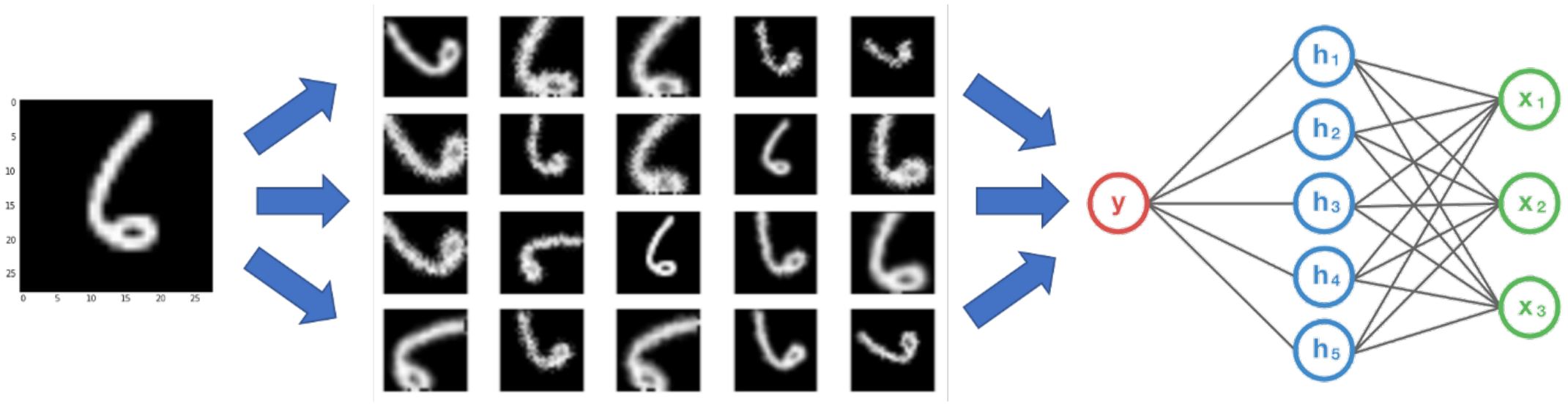


Julian Merten, et al 2019 (arXiv:1810.11027)

Convolutional Neural Networks

Data Augmentation

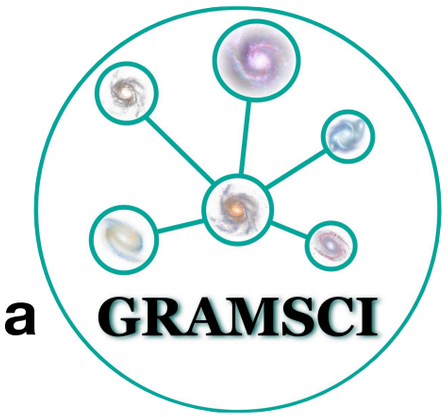
A convolutional neural network that can robustly classify objects even if its placed in different orientations is said to have the property called **invariance**. More specifically, a CNN can be invariant to translation, viewpoint, size or illumination (Or a combination of the above).



Popular augmentations:

- **left/right flip**
- **up/down flip**
- **Random rotation**
- **Zoom scale**
- **Crop**
- **X,Y translation (shift)**

Conclusions

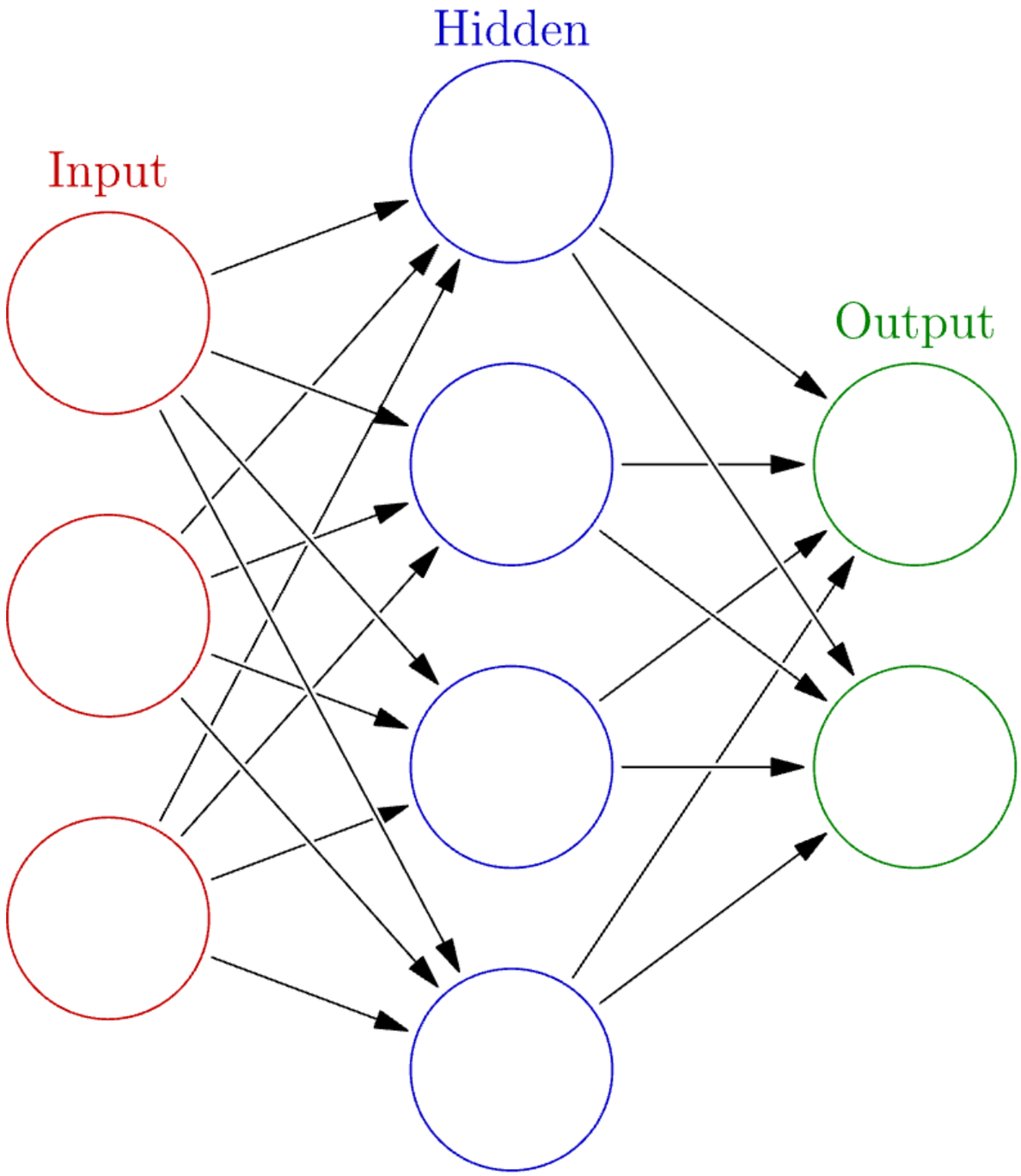


v1 out now!

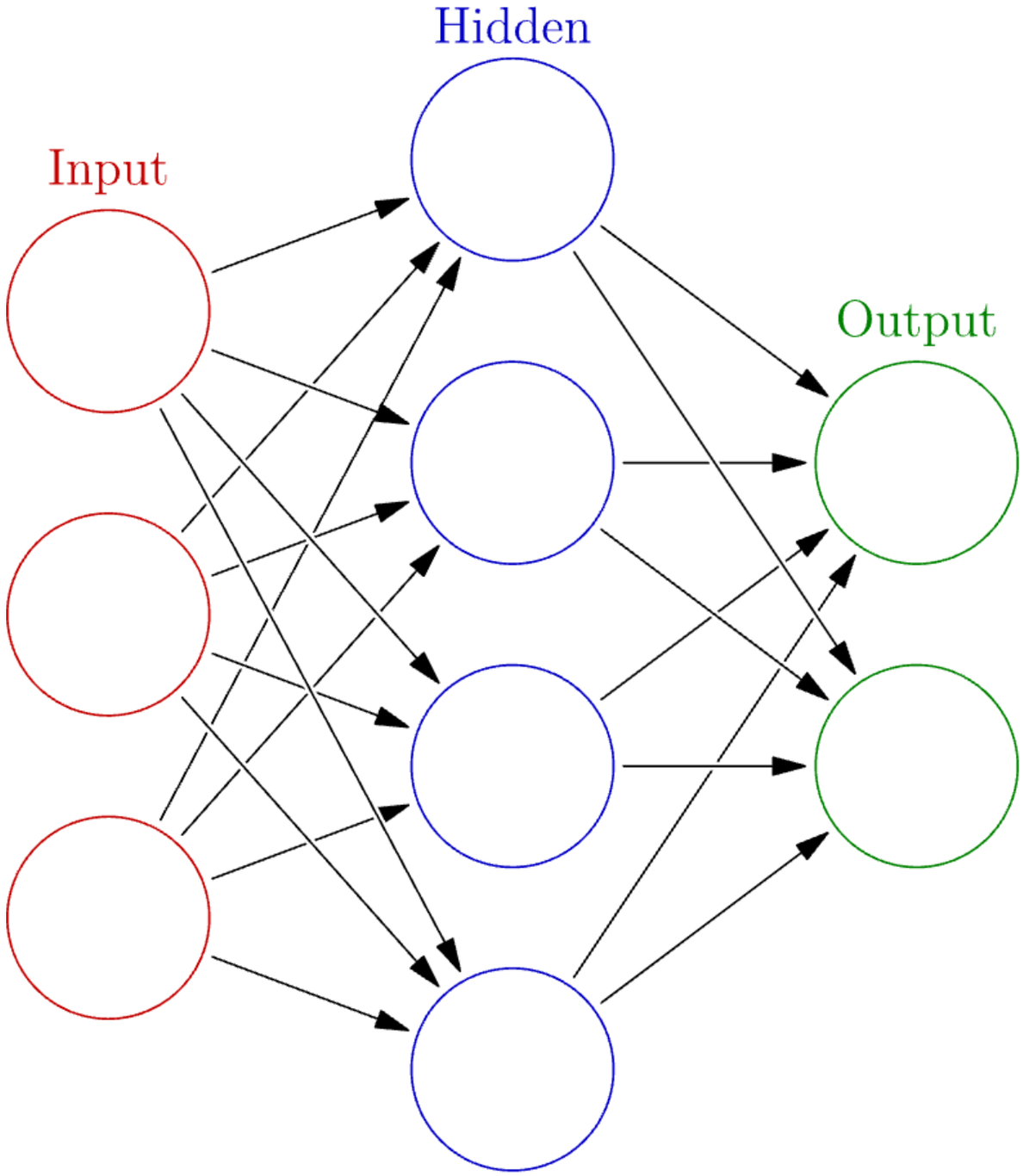
- **We introduce a new clustering algorithm, publicly available under a GNU public release licence**
- **GRA**ph **M**ade **S**tatistics for **C**osmological **I**nformation: **GRAMSCI** **available from: http://bitbucket.org/csabiu/gramsci**
- **GRAMSCI performs much better than purely tree based approaches**
- **We show the performance by measuring all possible 3pCF unto and beyond the BAO scale with current SDSS BOSS data**
- **We make the first measurements of the 4-point function of SDSS galaxies**
- **We show the flexibility of adopting a Big Data Analytic approach. As an example the luminosity-number density cross correlation has the potential to unlock new information in the galaxy data that depends on baryonic physics and compensated isocurvature originating in the very early Universe.**

Thank You 감사합니다

Artificial Neural Networks

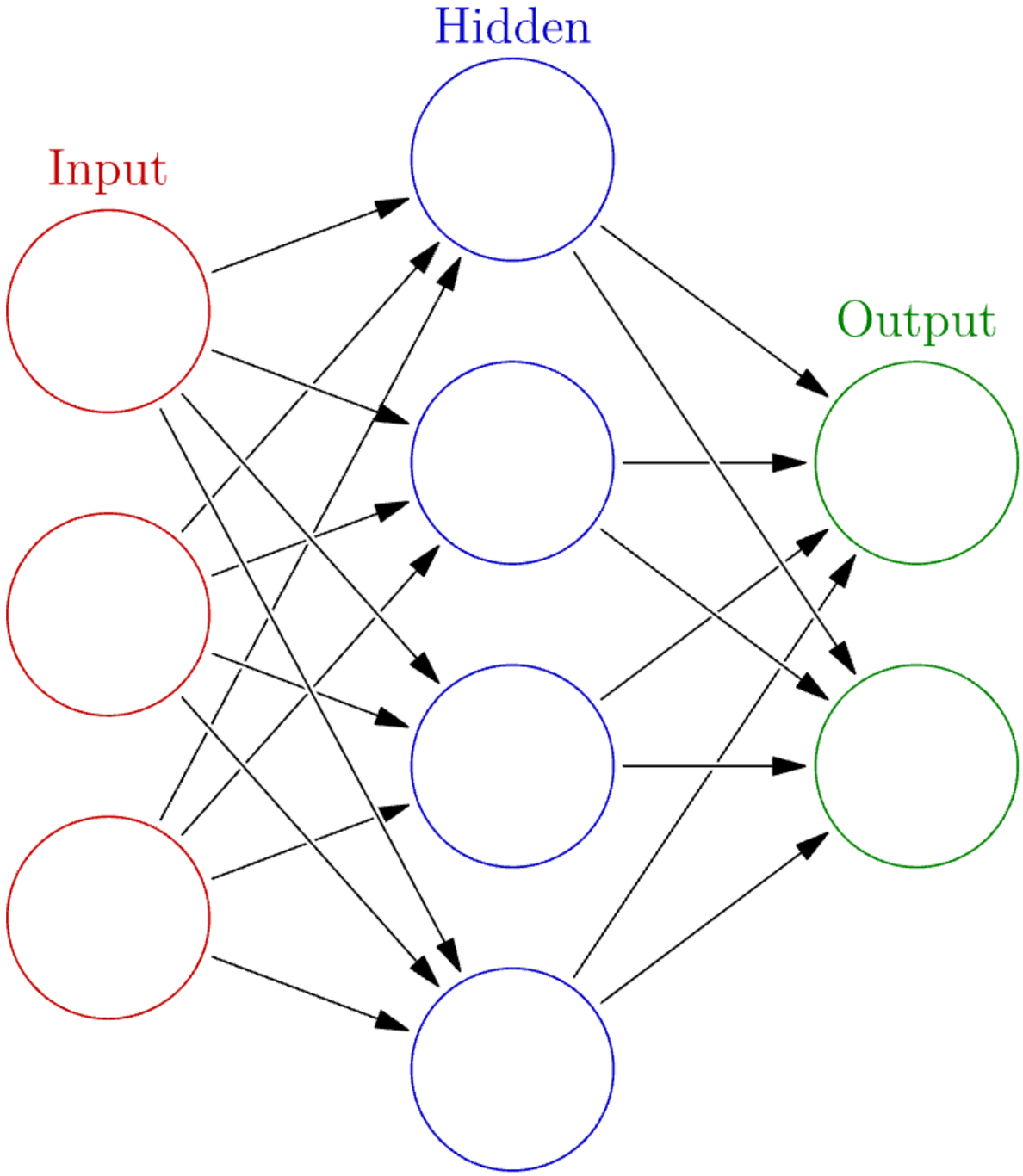


Artificial Neural Networks



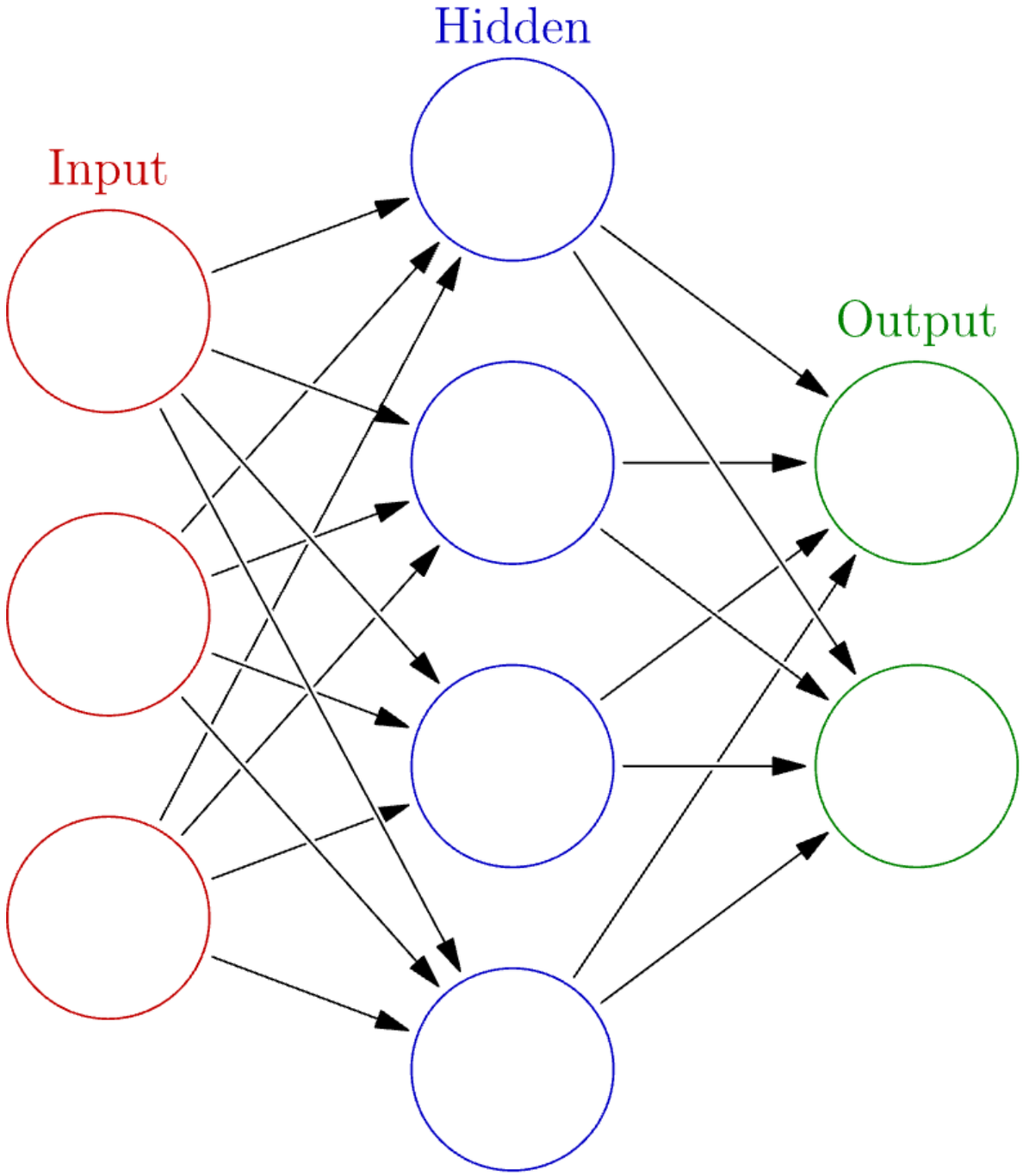
- classification based on a large collection of neural units.

Artificial Neural Networks



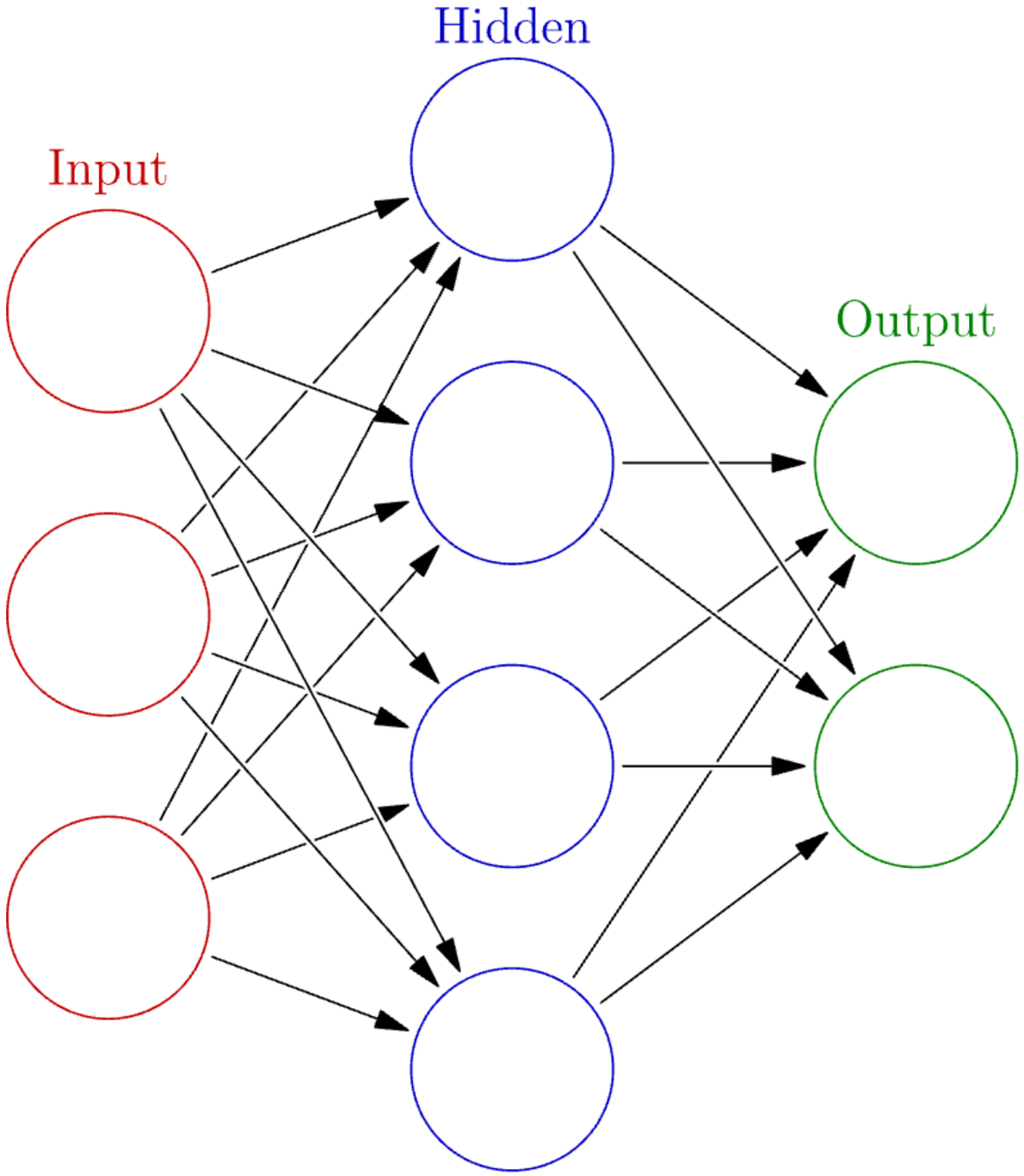
- classification based on a large collection of neural units.
- modeling the way a biological brain works.

Artificial Neural Networks



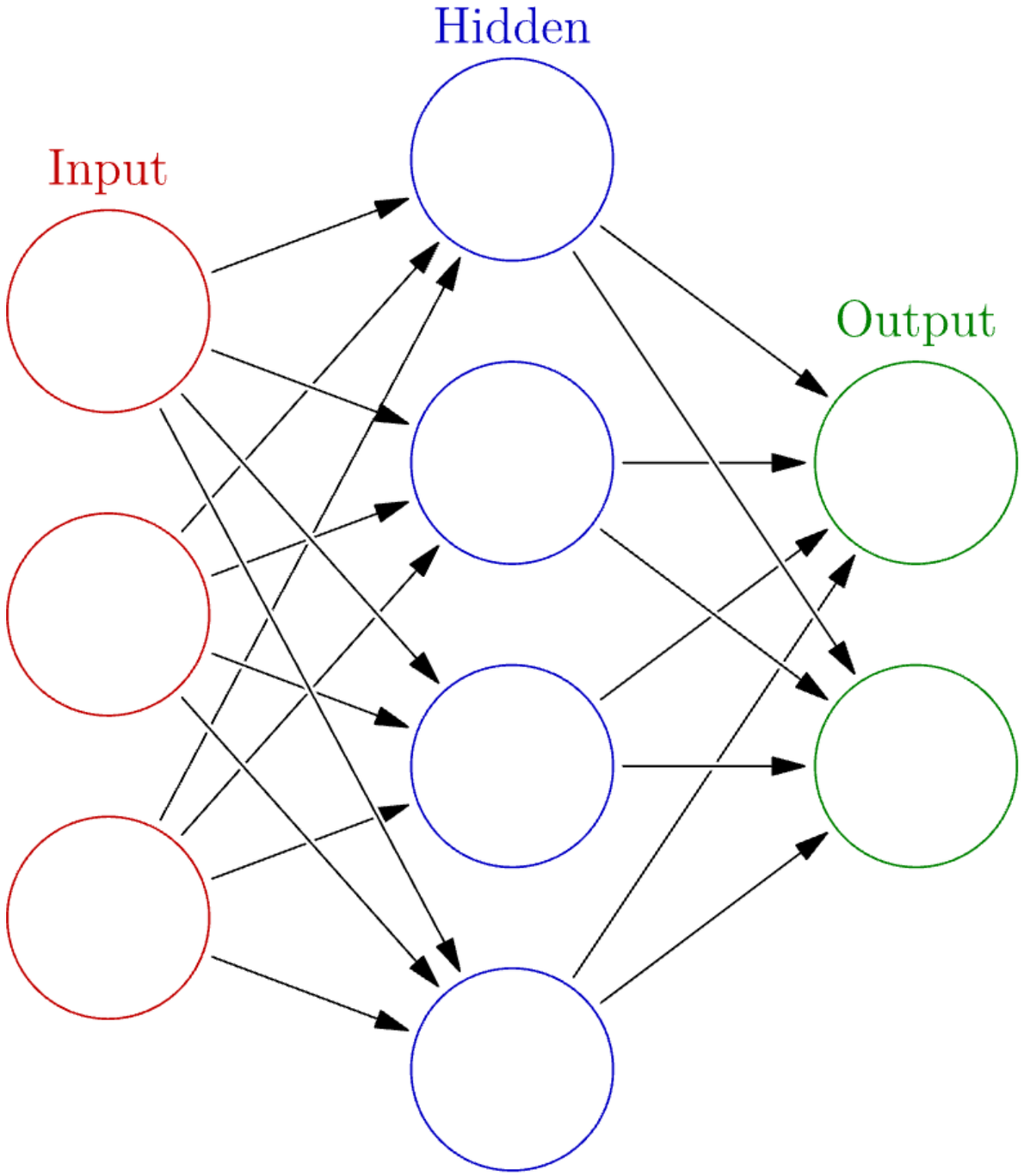
- classification based on a large collection of neural units.
- modeling the way a biological brain works.
- Each neural unit is connected with many others.

Artificial Neural Networks



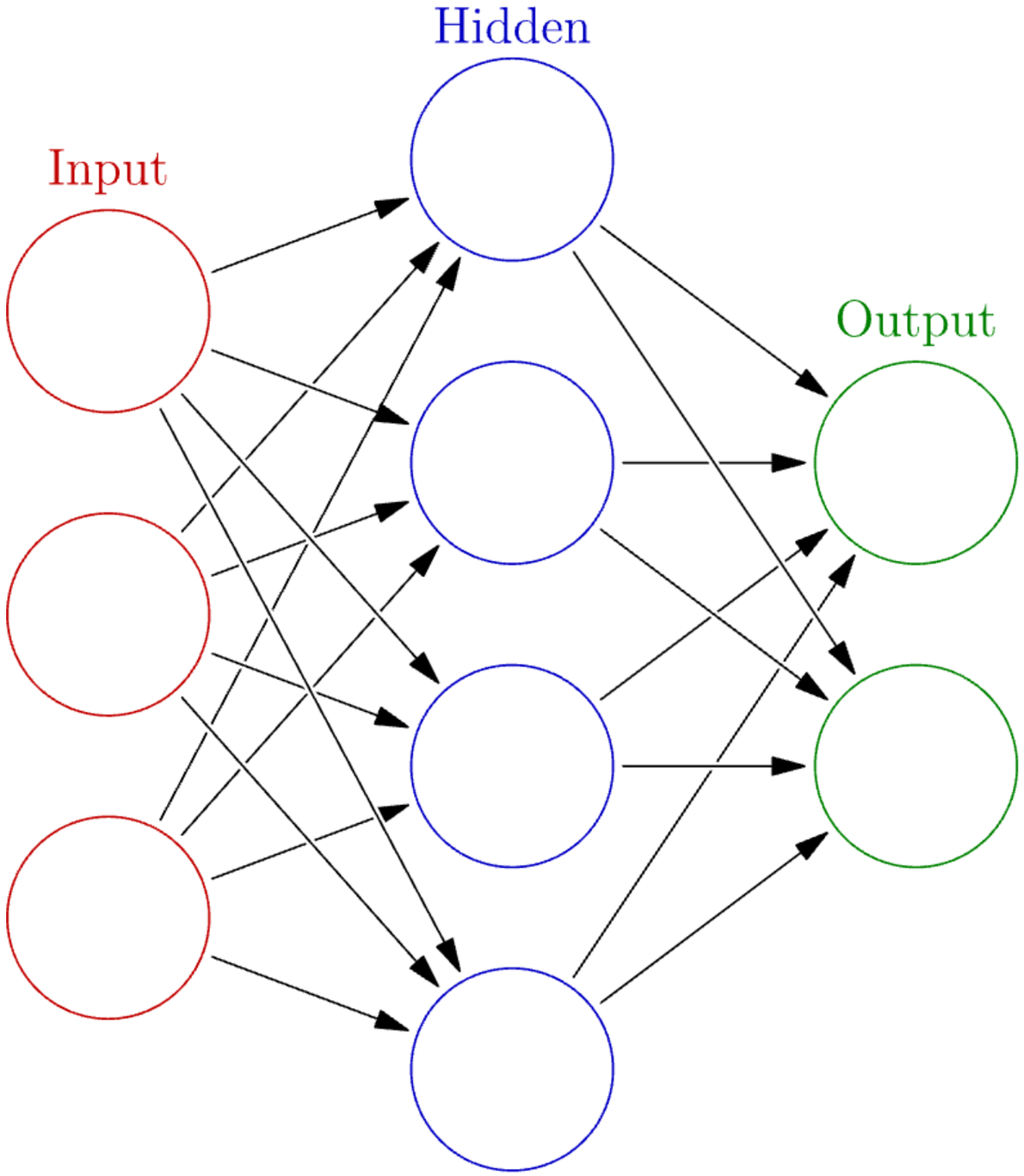
- classification based on a large collection of neural units.
- modeling the way a biological brain works.
- Each neural unit is connected with many others.
- links can be enforcing or inhibitory.

Artificial Neural Networks



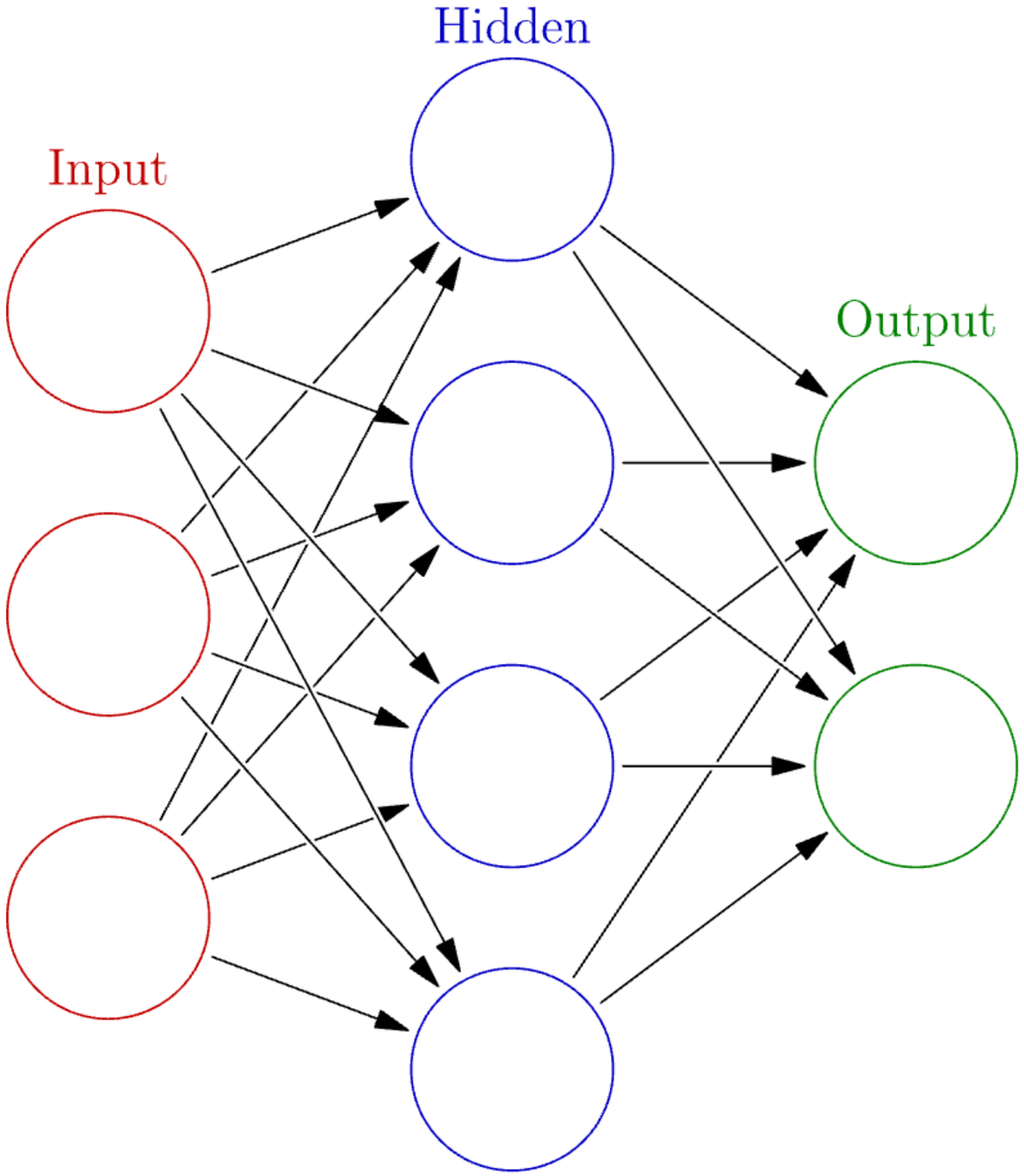
- classification based on a large collection of neural units.
- modeling the way a biological brain works.
- Each neural unit is connected with many others.
- links can be enforcing or inhibitory.
- Each individual neural unit has a summation function.

Artificial Neural Networks



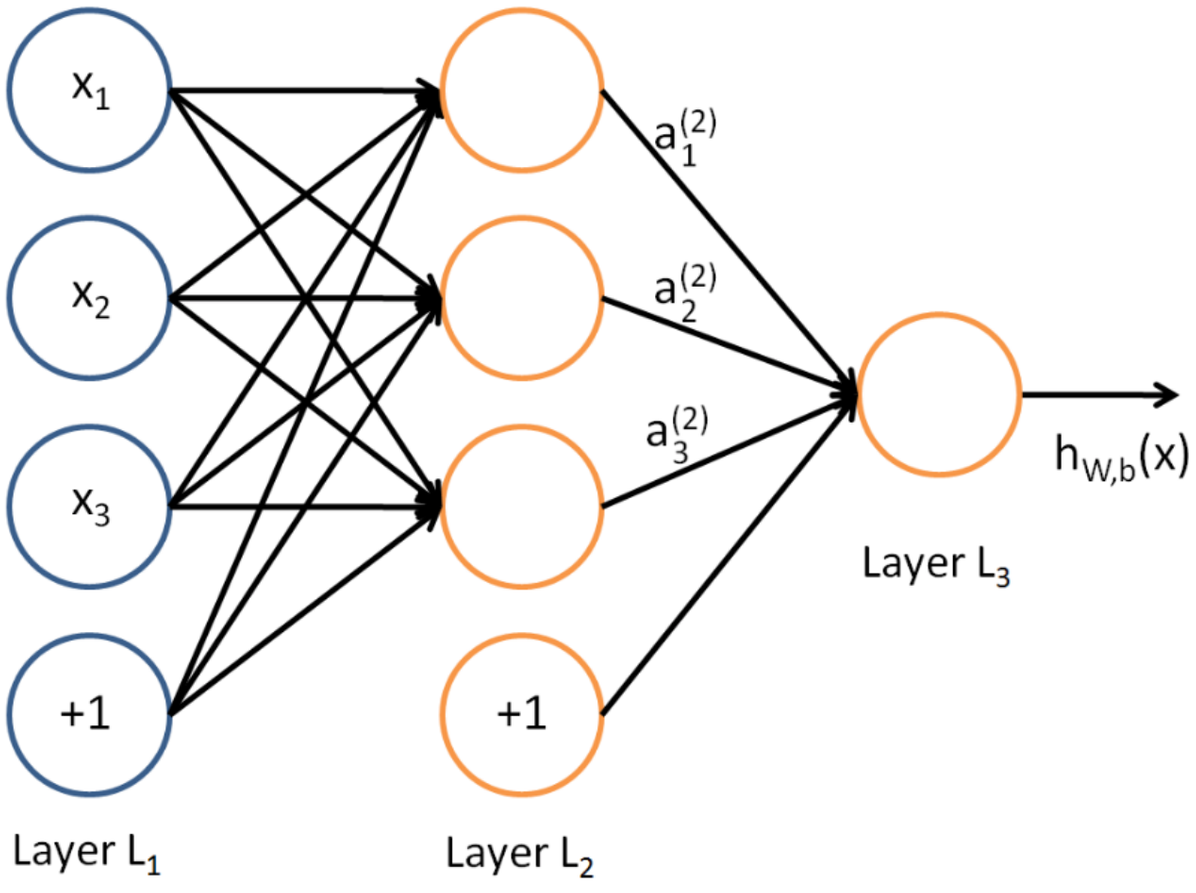
- classification based on a large collection of neural units.
- modeling the way a biological brain works.
- Each neural unit is connected with many others.
- links can be enforcing or inhibitory.
- Each individual neural unit has a summation function.
- These systems are self-learning and trained.

Artificial Neural Networks



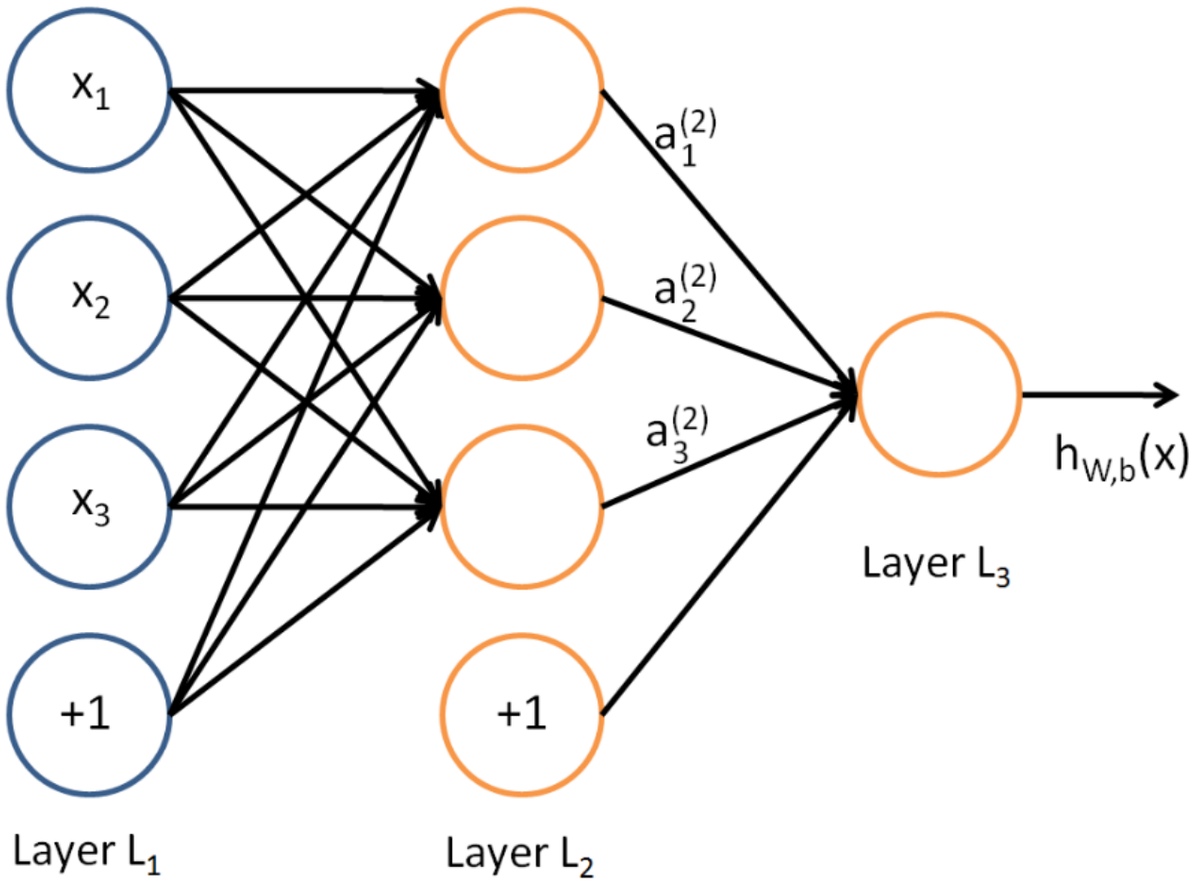
- classification based on a large collection of neural units.
- modeling the way a biological brain works.
- Each neural unit is connected with many others.
- links can be enforcing or inhibitory.
- Each individual neural unit has a summation function.
- These systems are self-learning and trained.
- There can be multiple layers.

Artificial Neural Networks



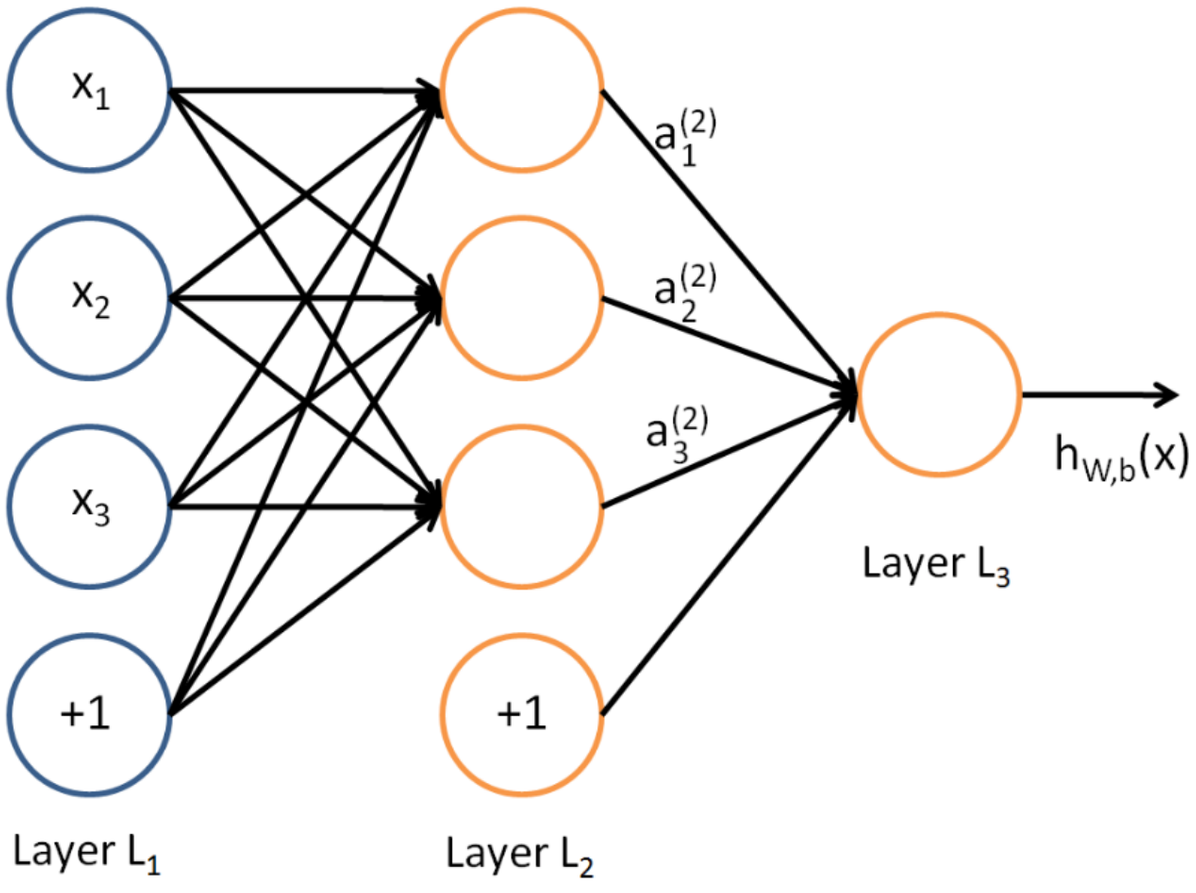
Artificial Neural Networks

- Orange circles represent neurons.

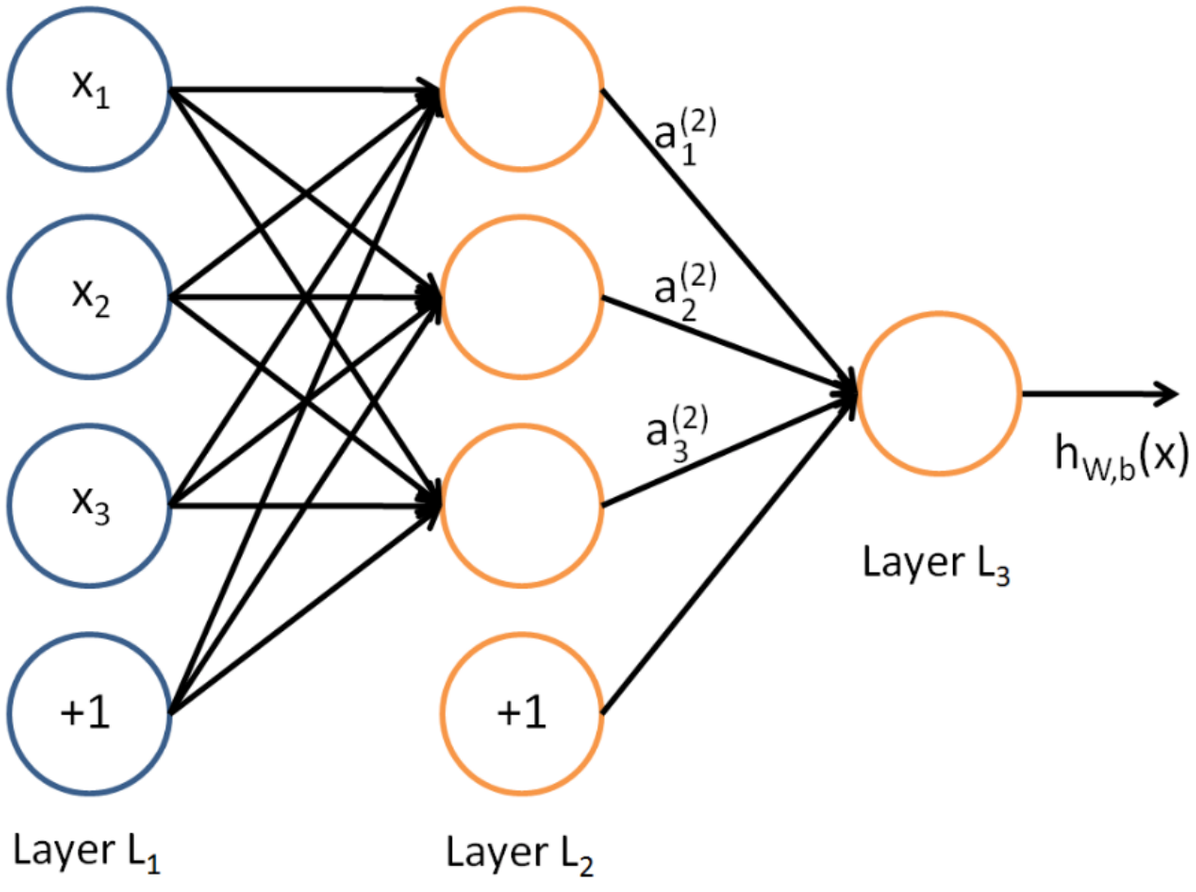


Artificial Neural Networks

- Orange circles represent neurons.
- A neuron can be the input of another.

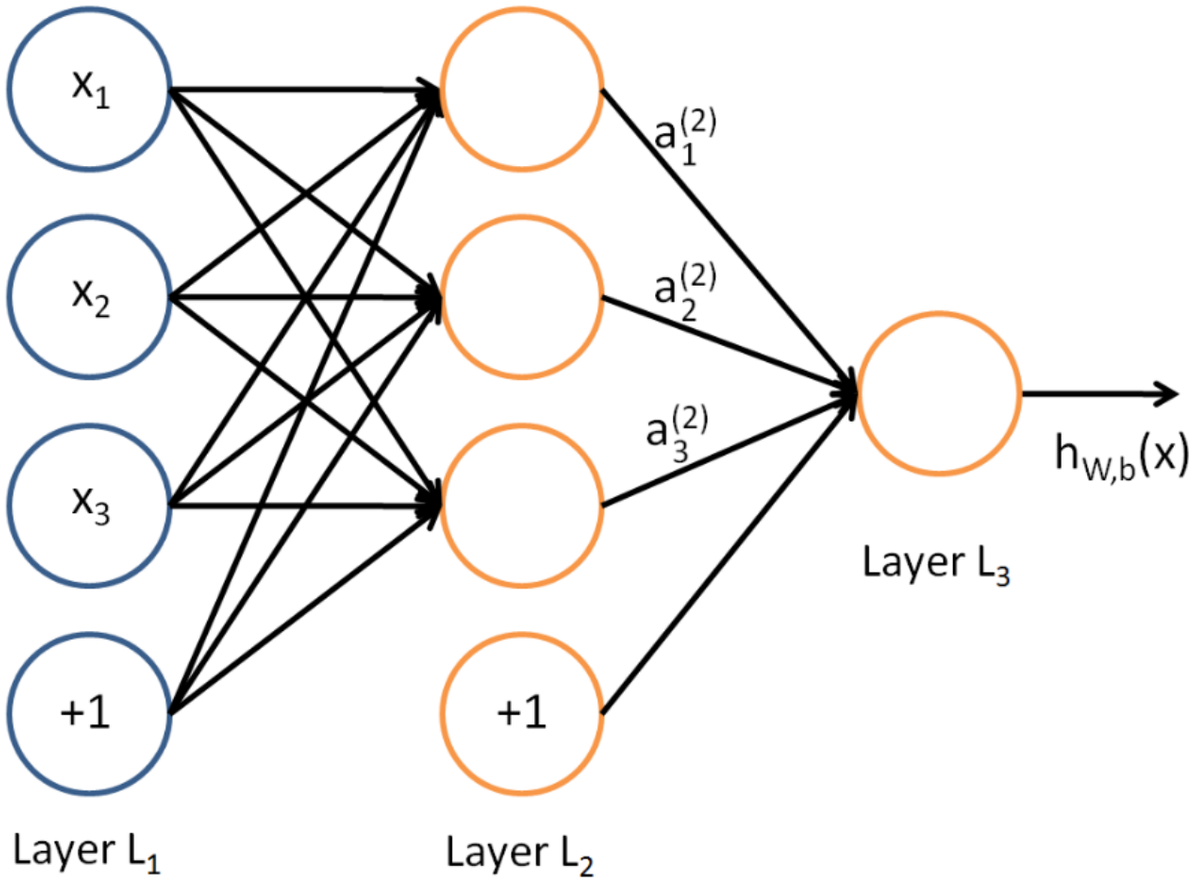


Artificial Neural Networks



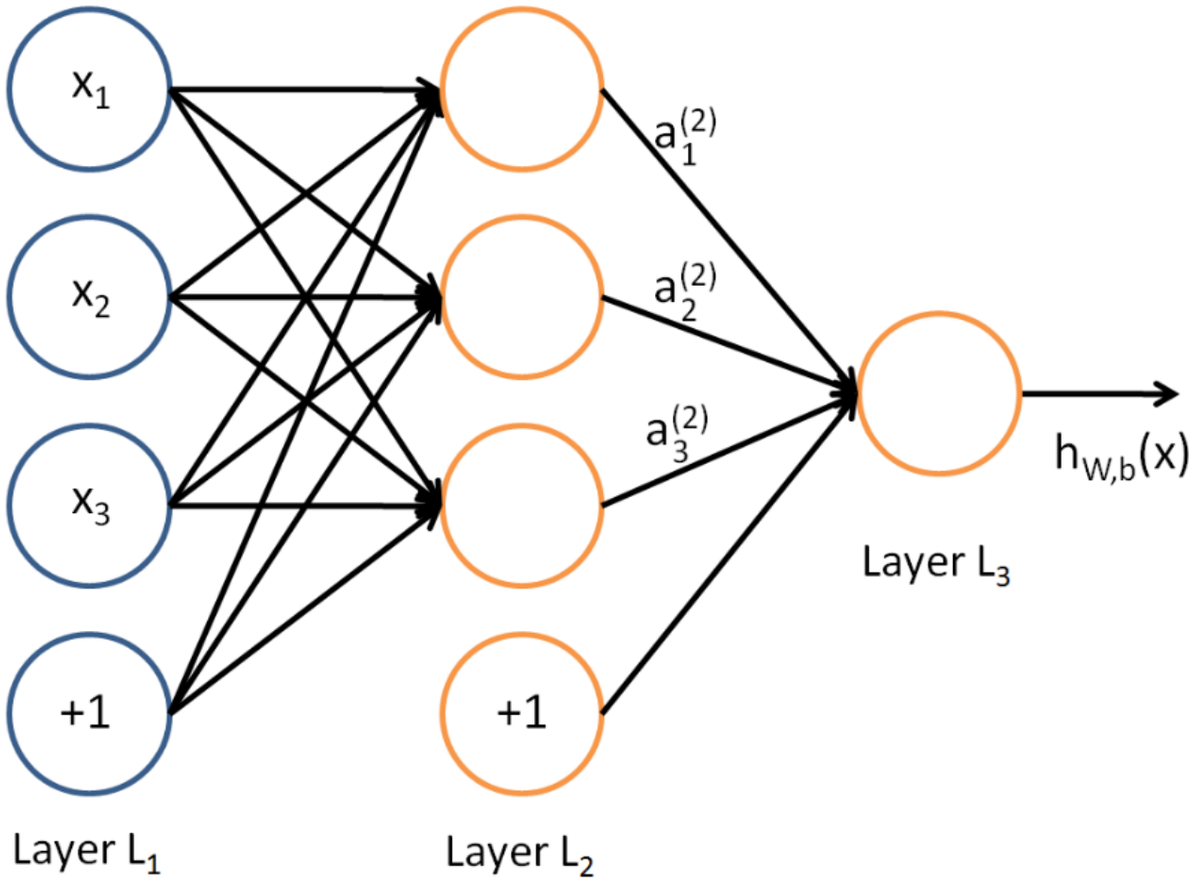
- Orange circles represent neurons.
- A neuron can be the input of another.
- X_1, X_2, and X_3 in L_1 are the inputs.

Artificial Neural Networks



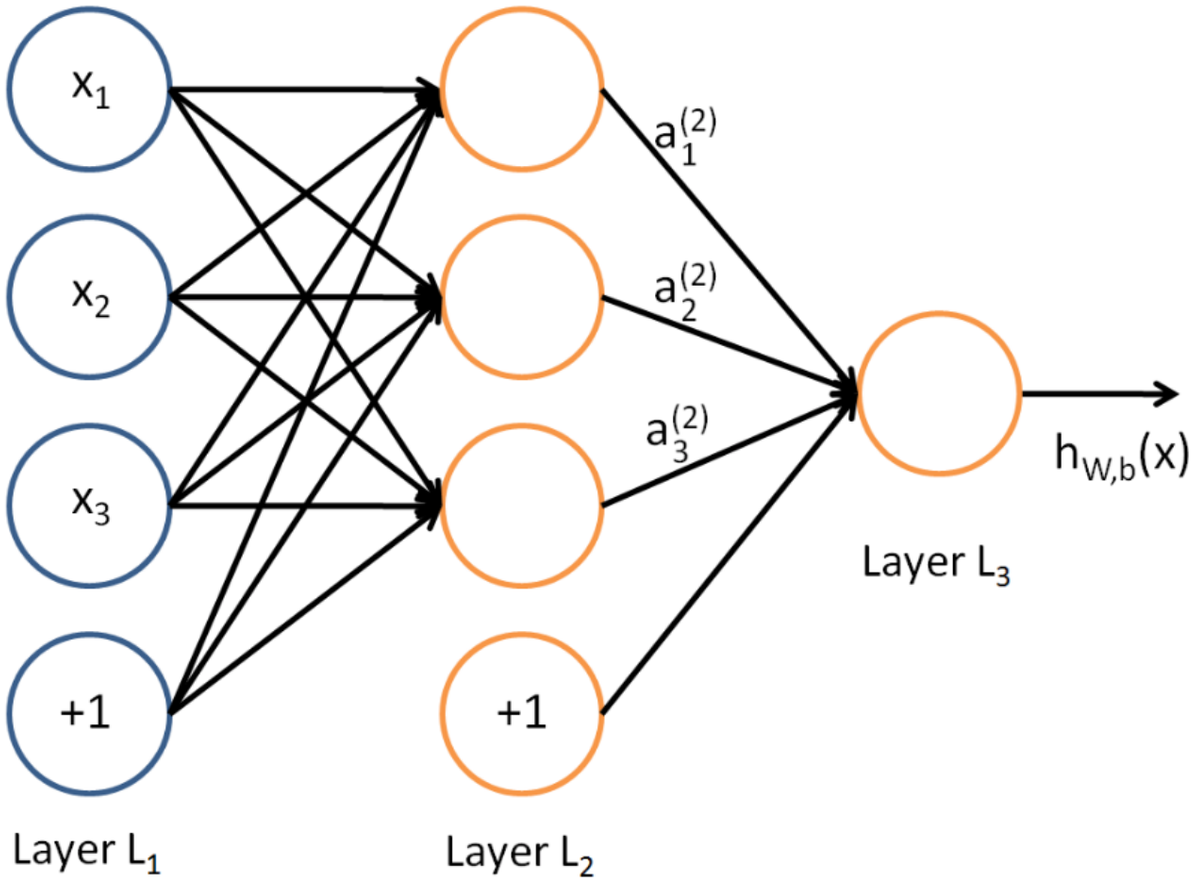
- Orange circles represent neurons.
- A neuron can be the input of another.
- X_1, X_2, and X_3 in L_1 are the inputs.
- “+1” are called bias units.

Artificial Neural Networks



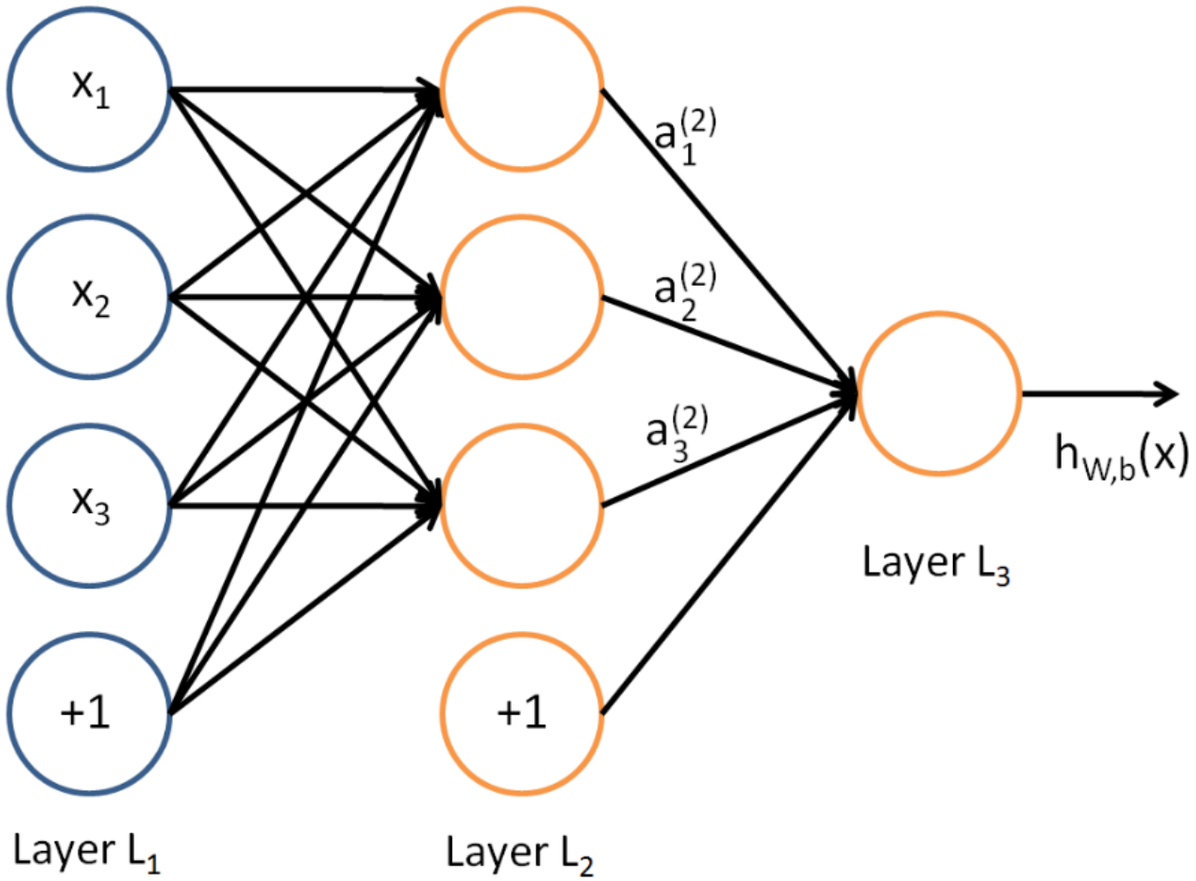
- Orange circles represent neurons.
- A neuron can be the input of another.
- X_1, X_2, and X_3 in L_1 are the inputs.
- "+1" are called bias units.
- L_2 is called a hidden layer.

Artificial Neural Networks



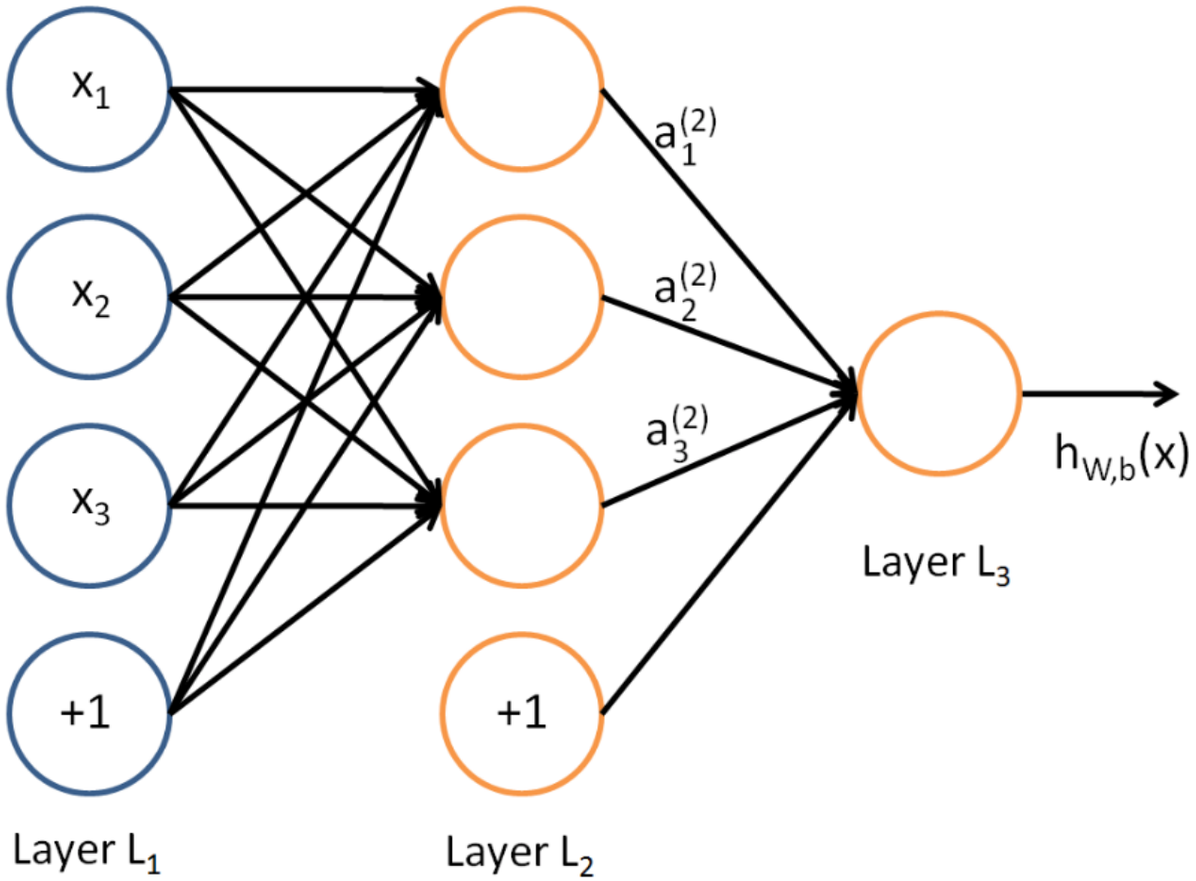
- Orange circles represent neurons.
- A neuron can be the input of another.
- X_1, X_2, and X_3 in L_1 are the inputs.
- "+1" are called bias units.
- L_2 is called a hidden layer.
- L_3 is the output layer.

Artificial Neural Networks



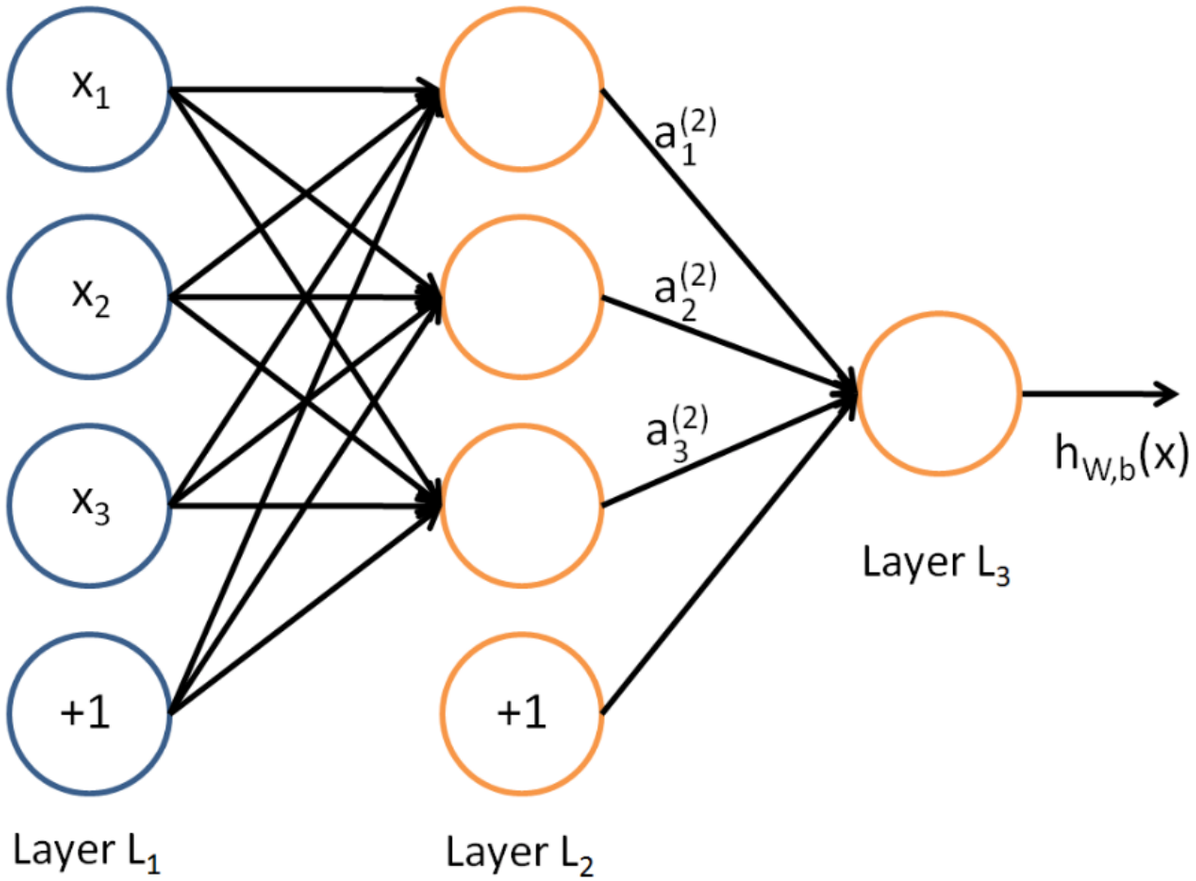
- Orange circles represent neurons.
- A neuron can be the input of another.
- X_1, X_2, and X_3 in L_1 are the inputs.
- "+1" are called bias units.
- L_2 is called a hidden layer.
- L_3 is the output layer.
- This ANN has 3 input units and 1 output unit.

Artificial Neural Networks



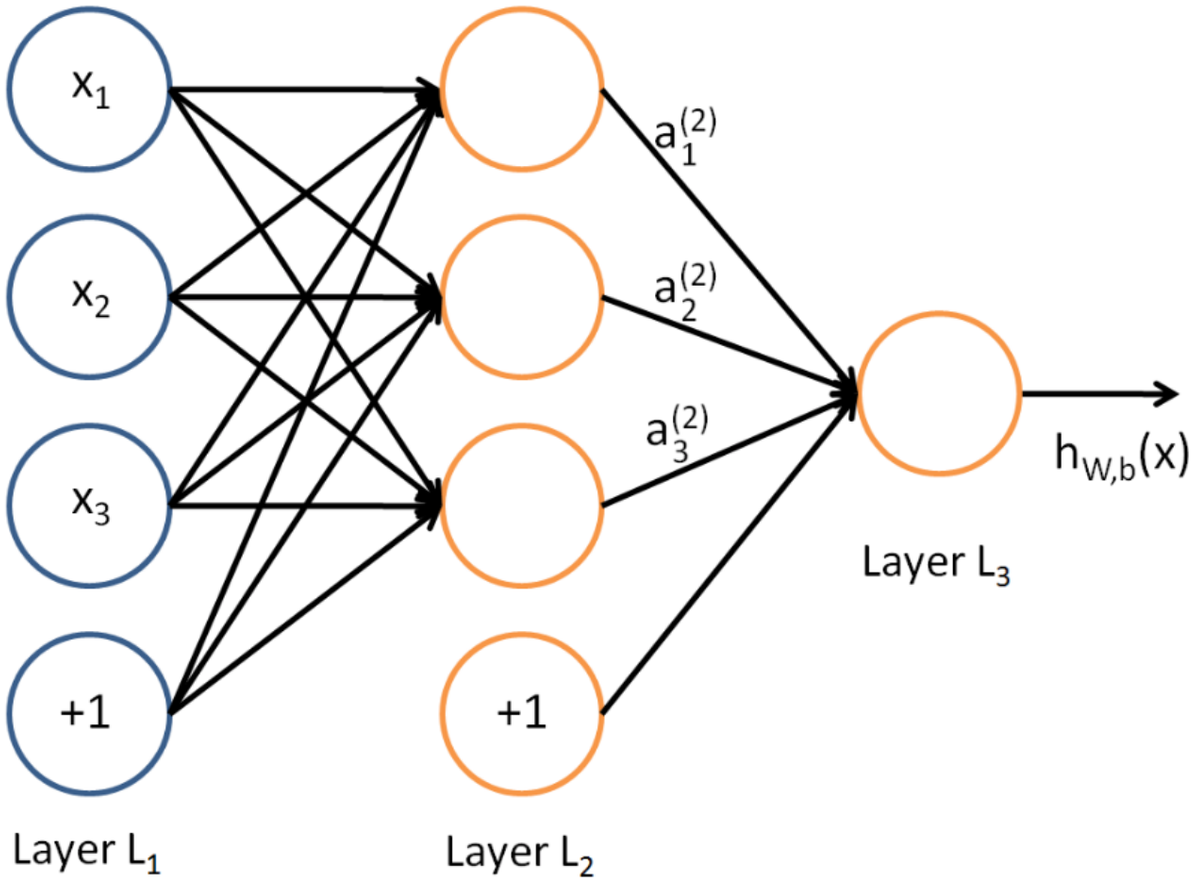
- Orange circles represent neurons.
- A neuron can be the input of another.
- X_1, X_2, and X_3 in L_1 are the inputs.
- "+1" are called bias units.
- L_2 is called a hidden layer.
- L_3 is the output layer.
- This ANN has 3 input units and 1 output unit.
- Each neuron in L_2 is a computational unit and outputs $h=f(W_1x_1+W_2x_2+W_3x_3+b)$

Artificial Neural Networks



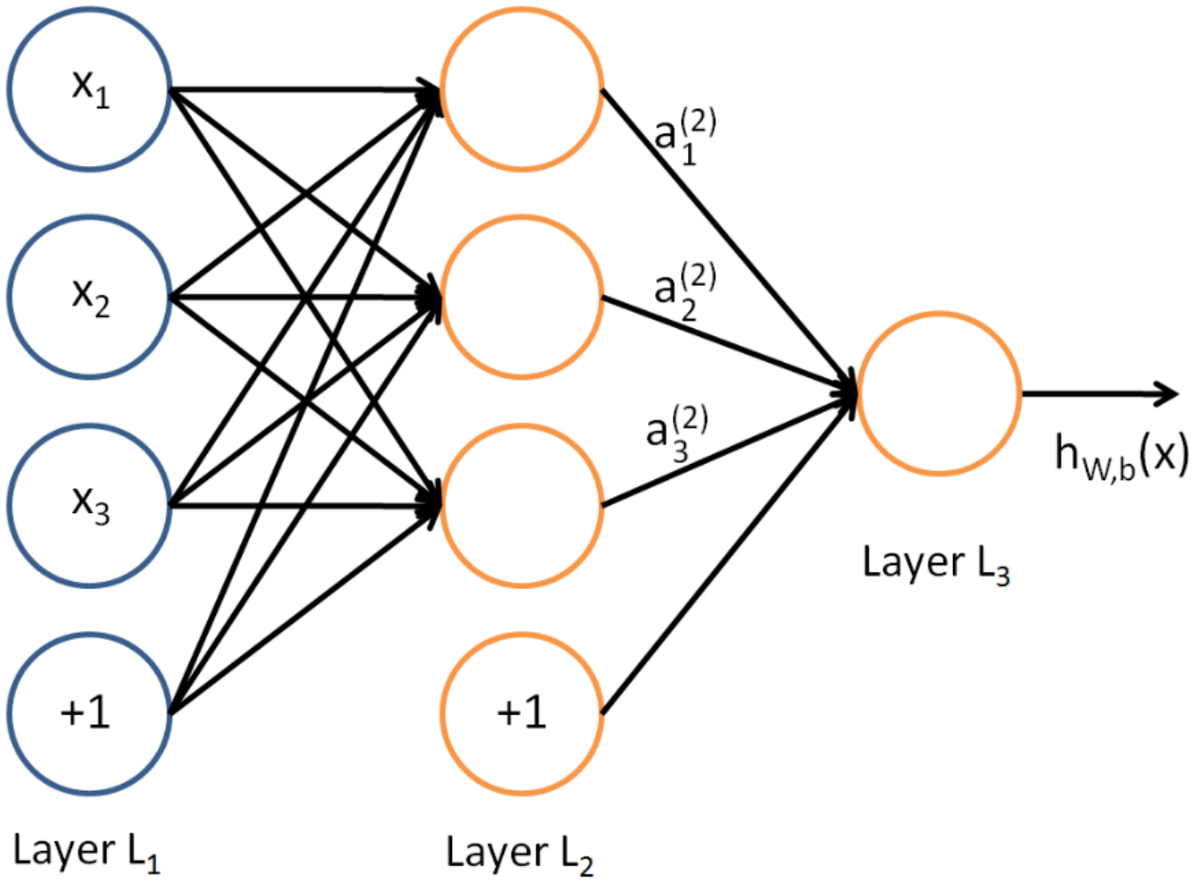
- Orange circles represent neurons.
- A neuron can be the input of another.
- X_1, X_2, and X_3 in L_1 are the inputs.
- "+1" are called bias units.
- L_2 is called a hidden layer.
- L_3 is the output layer.
- This ANN has 3 input units and 1 output unit.
- Each neuron in L_2 is a computational unit and outputs $h=f(W_1x_1+W_2x_2+W_3x_3+b)$
- h is called activation function and we choose $f(z)=1/(1+e^{-z})$

Artificial Neural Networks



- Orange circles represent neurons.
- A neuron can be the input of another.
- X_1, X_2, and X_3 in L_1 are the inputs.
- "+1" are called bias units.
- L_2 is called a hidden layer.
- L_3 is the output layer.
- This ANN has 3 input units and 1 output unit.
- Each neuron in L_2 is a computational unit and outputs $h=f(W_1x_1+W_2x_2+W_3x_3+b)$
- h is called activation function and we choose $f(z)=1/(1+e^{-z})$
- W is the weight and varies path-by-path.

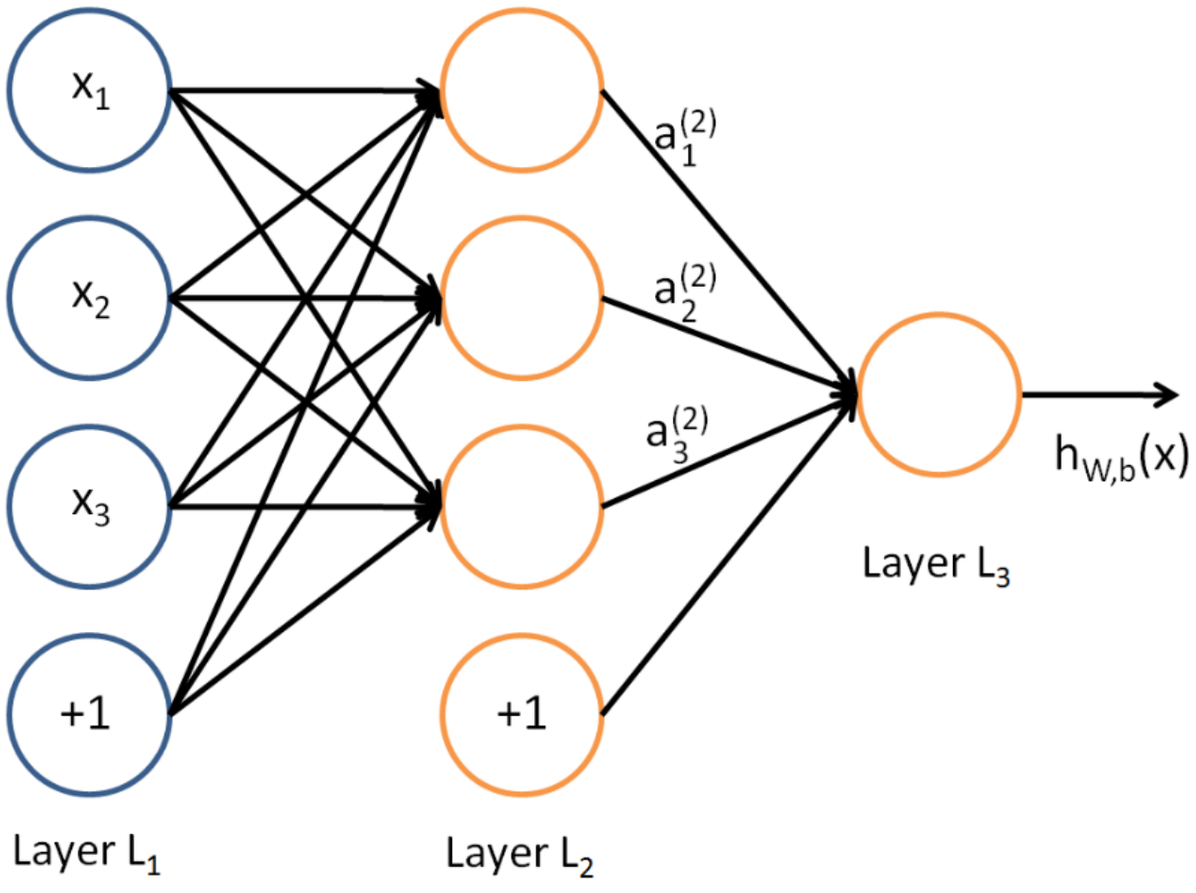
Artificial Neural Networks



- Orange circles represent neurons.
- A neuron can be the input of another.
- X_1, X_2, and X_3 in L_1 are the inputs.
- "+1" are called bias units.
- L_2 is called a hidden layer.
- L_3 is the output layer.
- This ANN has 3 input units and 1 output unit.
- Each neuron in L_2 is a computational unit and outputs $h=f(W_1x_1+W_2x_2+W_3x_3+b)$
- h is called activation function and we choose $f(z)=1/(1+e^{-z})$
- W is the weight and varies path-by-path.

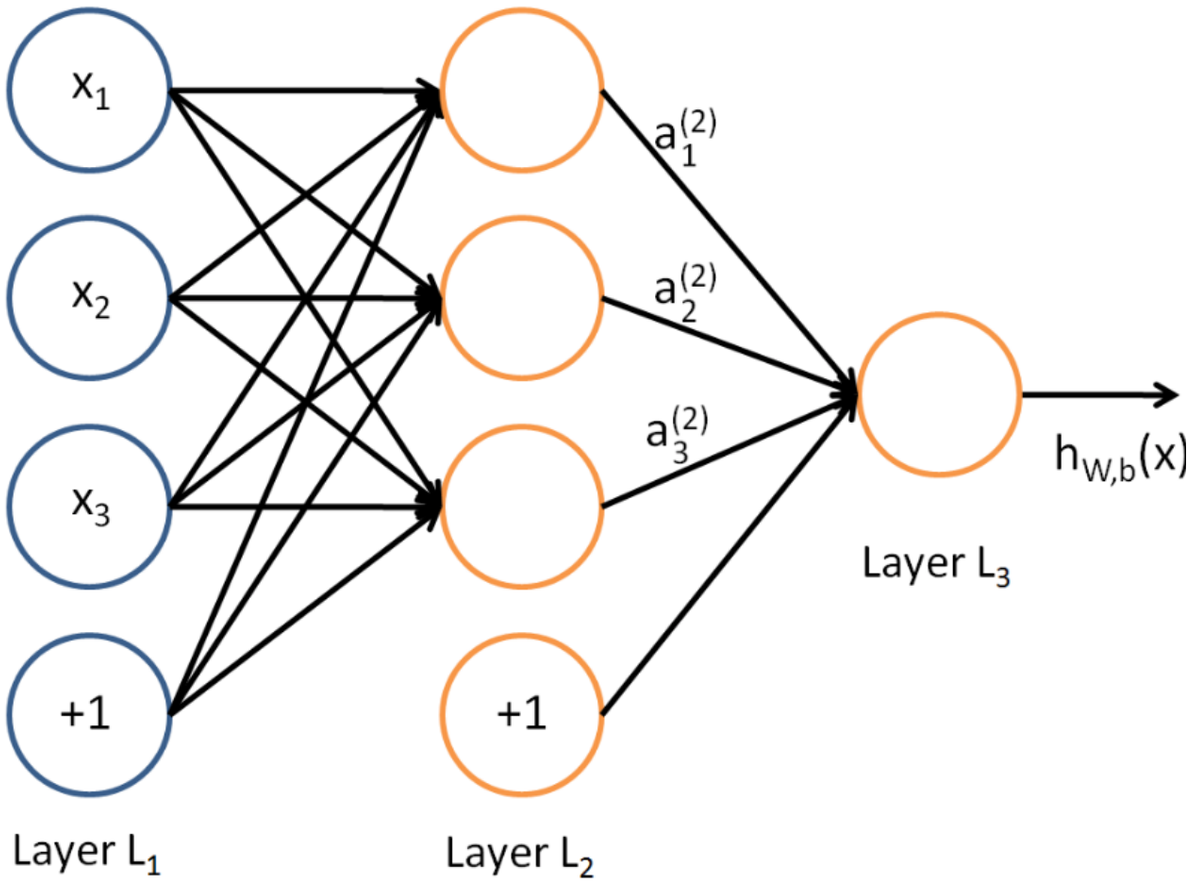
Artificial Neural Networks

We can express these functions by the following equations.



Artificial Neural Networks

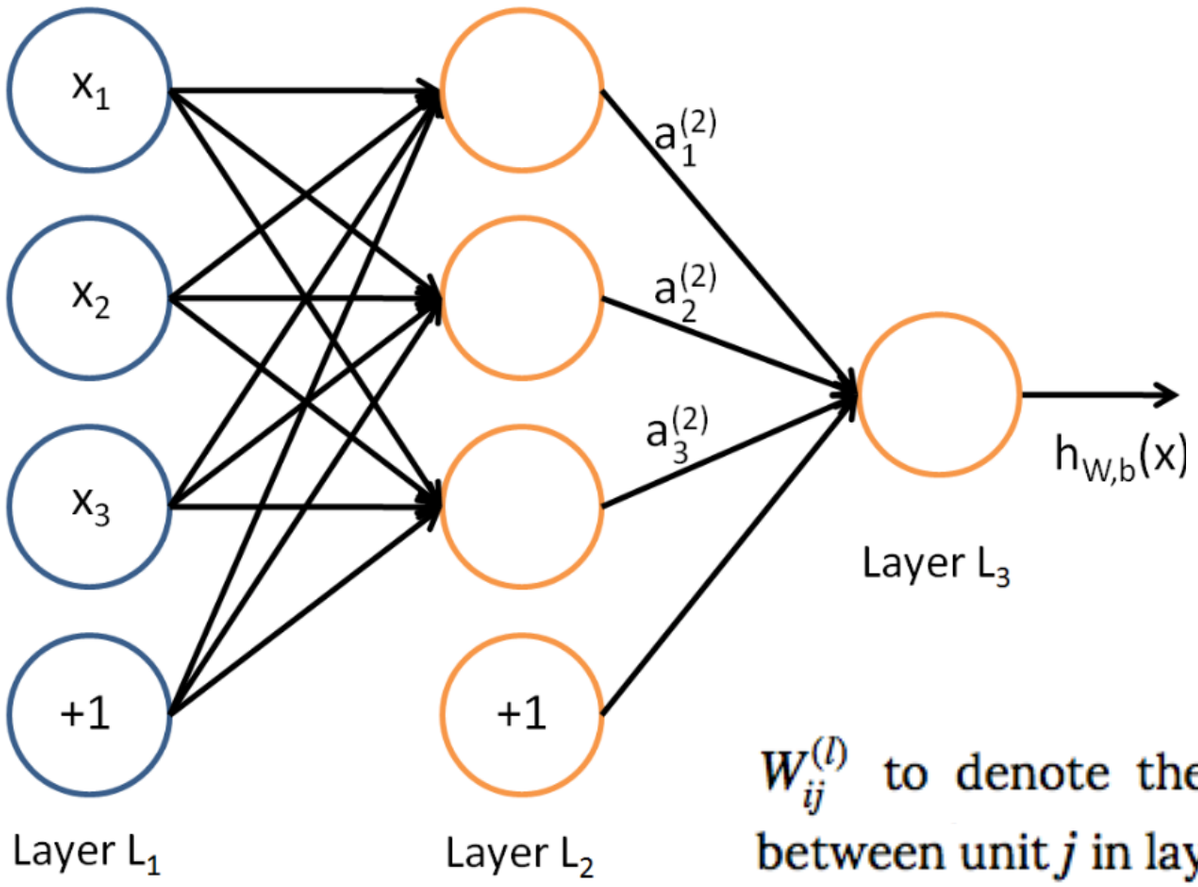
We can express these functions by the following equations.



$$a_1^{(2)} = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)})$$
$$a_2^{(2)} = f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)})$$
$$a_3^{(2)} = f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)})$$
$$h_{W,b}(x) = a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})$$

Artificial Neural Networks

We can express these functions by the following equations.

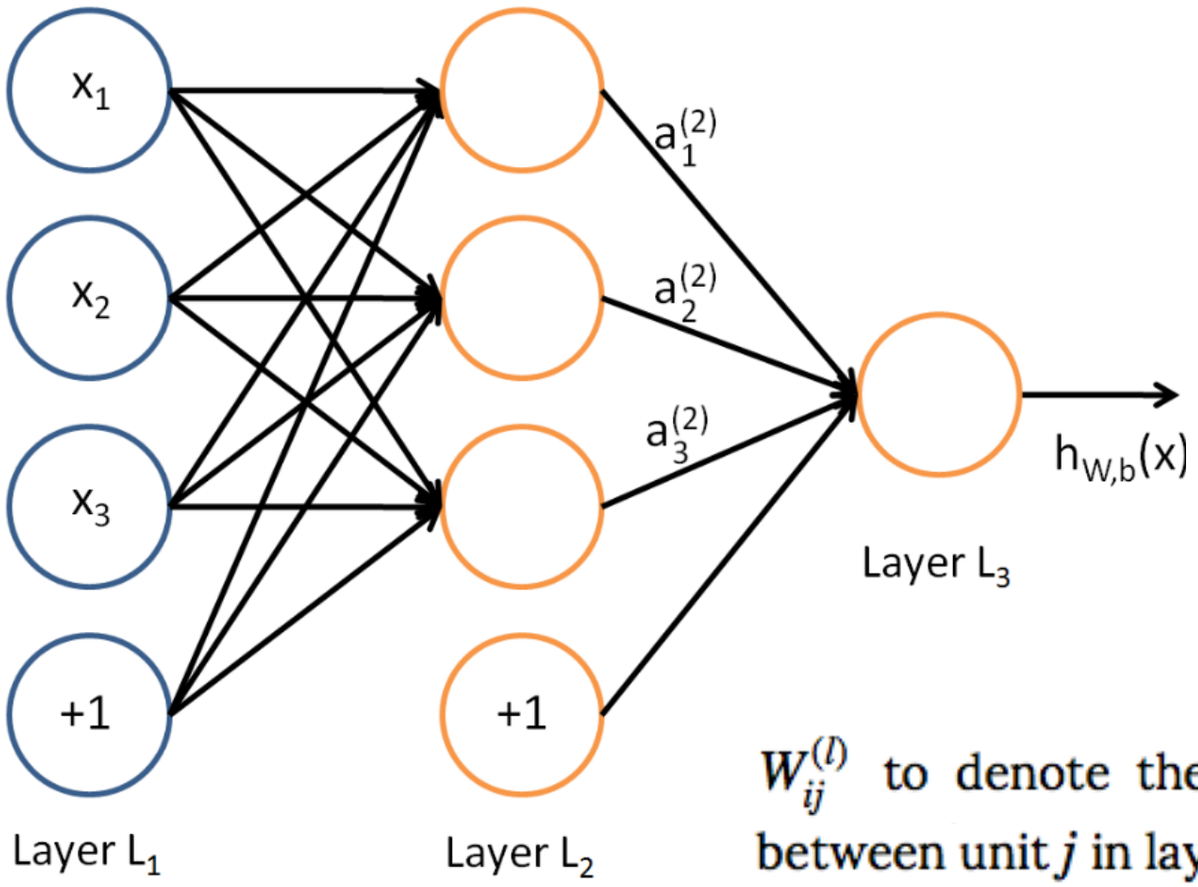


$$a_1^{(2)} = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)})$$
$$a_2^{(2)} = f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)})$$
$$a_3^{(2)} = f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)})$$
$$h_{W,b}(x) = a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})$$

$W_{ij}^{(l)}$ to denote the parameter (or weight) associated with the connection between unit j in layer l, and unit i in layer $l+1$

Artificial Neural Networks

We can express these functions by the following equations.



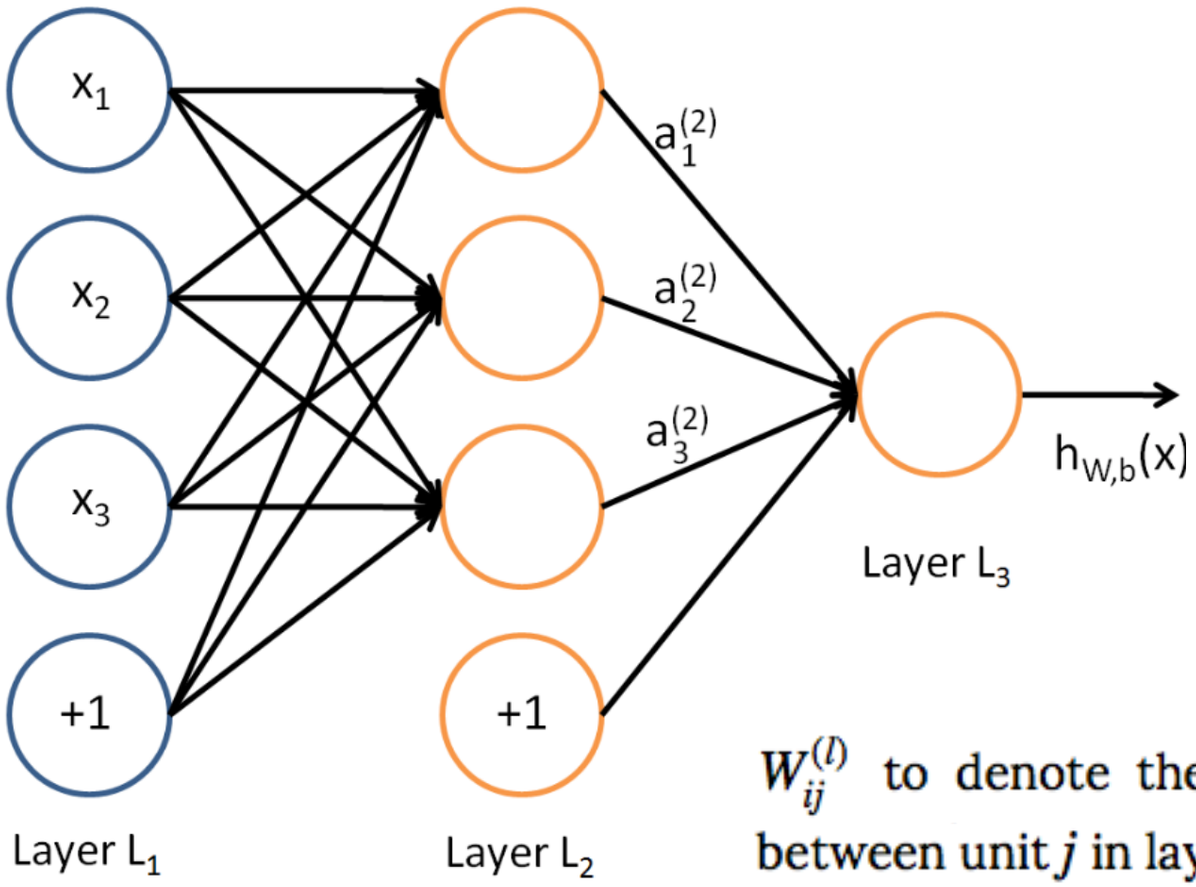
$$a_1^{(2)} = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)})$$
$$a_2^{(2)} = f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)})$$
$$a_3^{(2)} = f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)})$$
$$h_{W,b}(x) = a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})$$

$W_{ij}^{(l)}$ to denote the parameter (or weight) associated with the connection between unit j in layer l, and unit i in layer $l+1$

We will write $a_i^{(l)}$ to denote the **activation** (meaning output value) of unit i in layer l.

Artificial Neural Networks

We can express these functions by the following equations.



$$a_1^{(2)} = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)})$$
$$a_2^{(2)} = f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)})$$
$$a_3^{(2)} = f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)})$$
$$h_{W,b}(x) = a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})$$

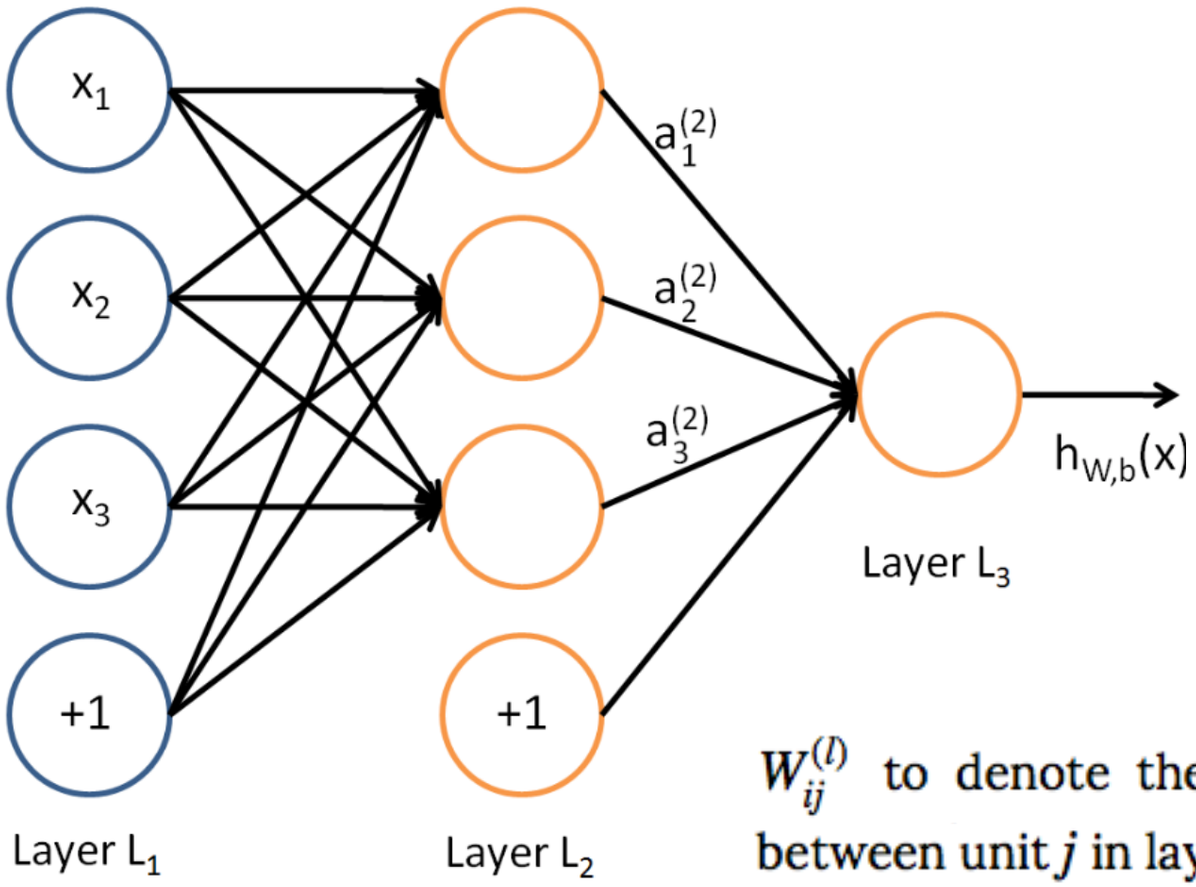
$W_{ij}^{(l)}$ to denote the parameter (or weight) associated with the connection between unit j in layer l, and unit i in layer $l+1$

We will write $a_i^{(l)}$ to denote the **activation** (meaning output value) of unit i in layer l.

For $l = 1$, we also use $a_i^{(1)} = x_i$ to denote the i-th input.

Artificial Neural Networks

We can express these functions by the following equations.



$$a_1^{(2)} = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)})$$
$$a_2^{(2)} = f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)})$$
$$a_3^{(2)} = f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)})$$
$$h_{W,b}(x) = a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})$$

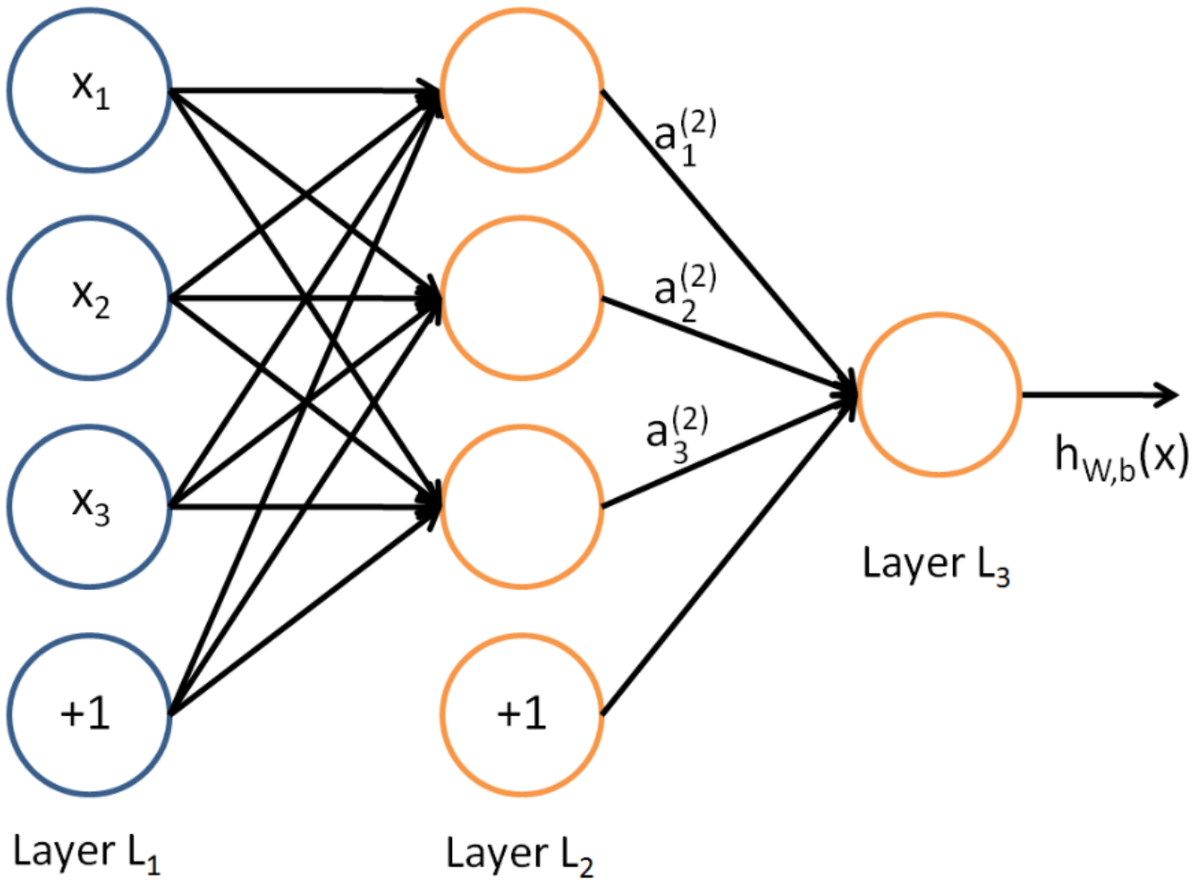
$W_{ij}^{(l)}$ to denote the parameter (or weight) associated with the connection between unit j in layer l, and unit i in layer $l+1$

We will write $a_i^{(l)}$ to denote the **activation** (meaning output value) of unit i in layer l.

For $l = 1$, we also use $a_i^{(1)} = x_i$ to denote the i-th input.

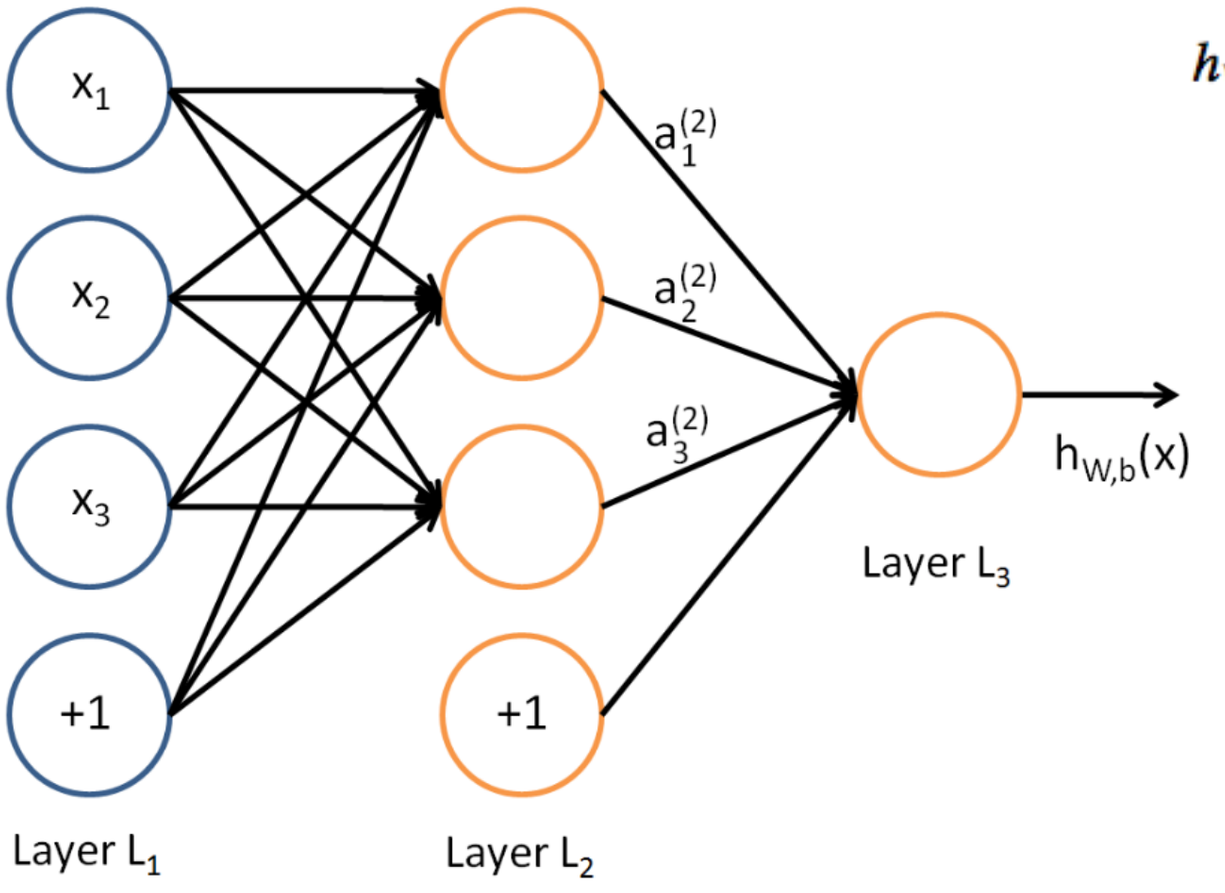
Our neural network has parameters $(W, b) = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})$

Artificial Neural Networks



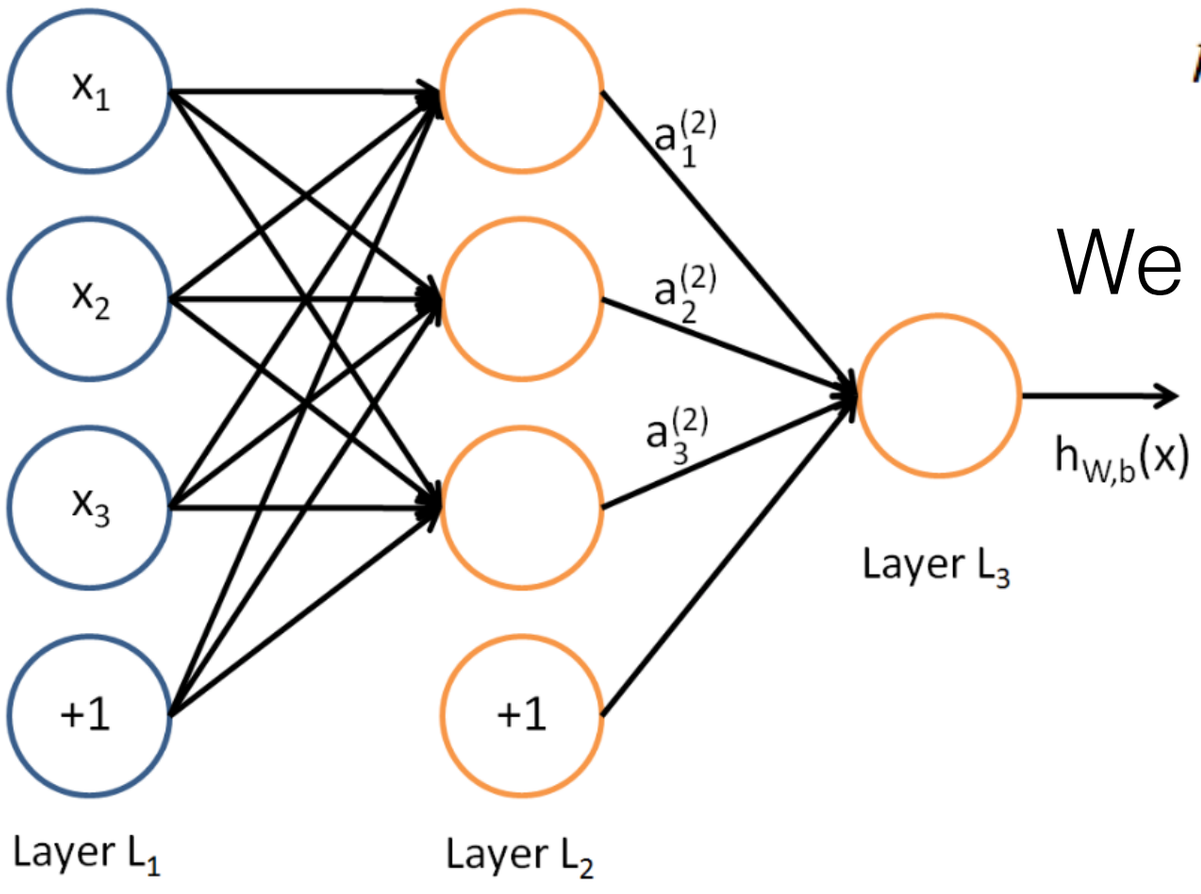
Artificial Neural Networks

$$a_1^{(2)} = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)})$$
$$a_2^{(2)} = f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)})$$
$$a_3^{(2)} = f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)})$$
$$h_{W,b}(x) = a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})$$



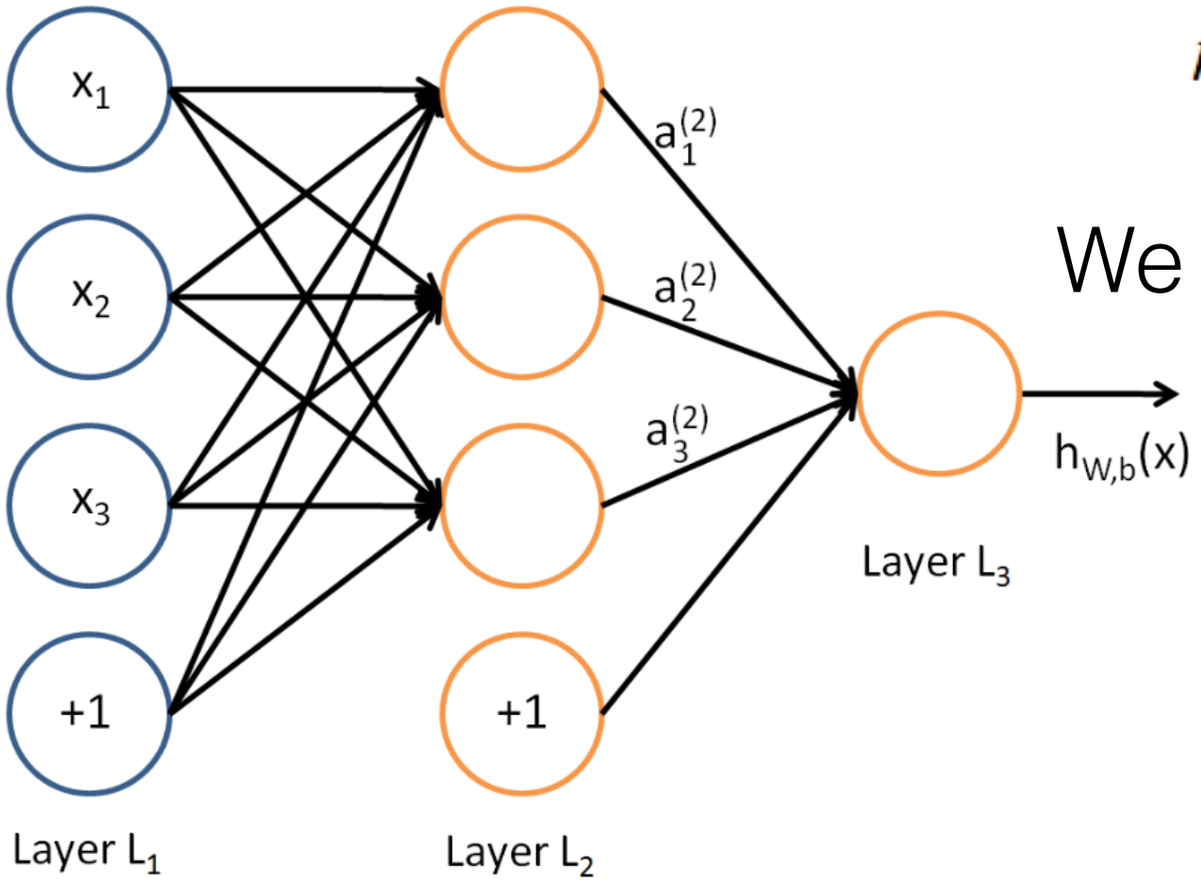
Artificial Neural Networks

$$a_1^{(2)} = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)})$$
$$a_2^{(2)} = f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)})$$
$$a_3^{(2)} = f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)})$$
$$h_{W,b}(x) = a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})$$



We can simplify these equations as

Artificial Neural Networks



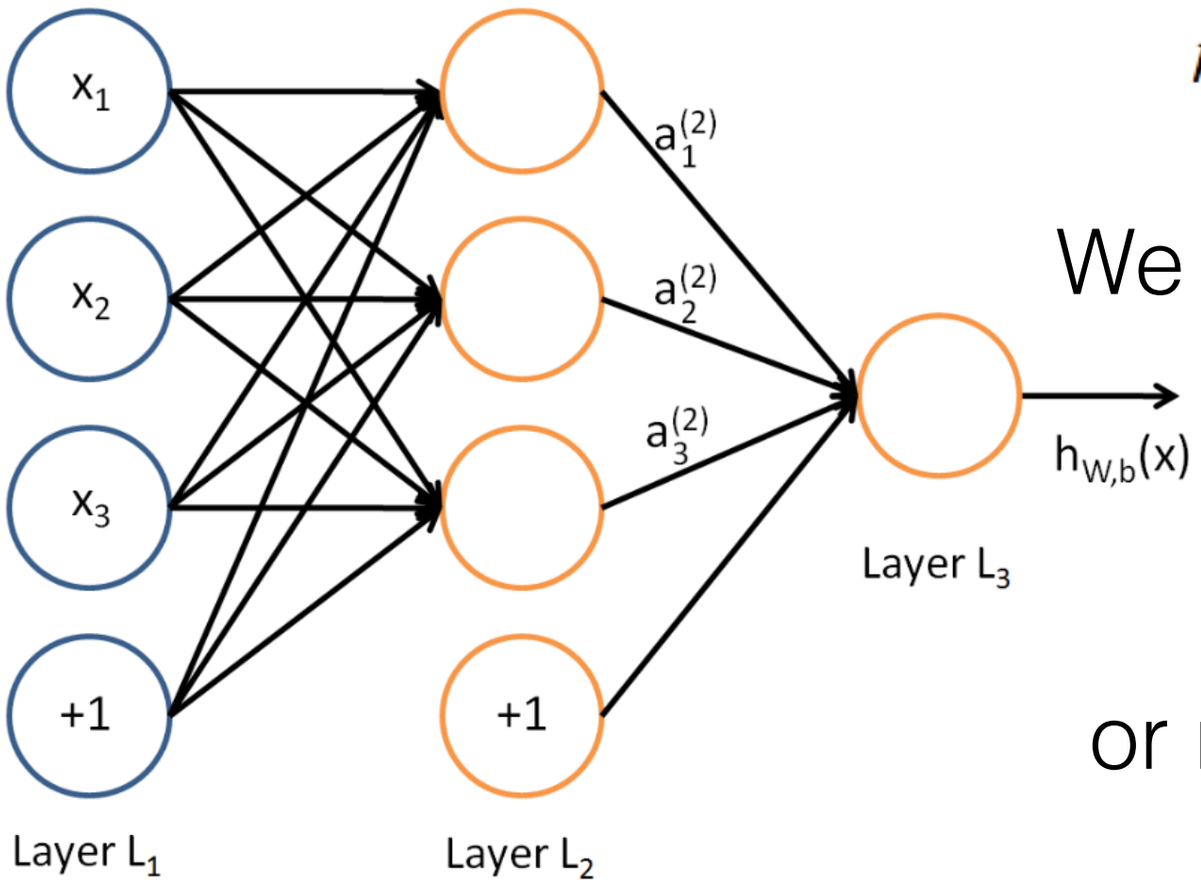
$$a_1^{(2)} = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)})$$
$$a_2^{(2)} = f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)})$$
$$a_3^{(2)} = f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)})$$
$$h_{W,b}(x) = a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})$$

We can simplify these equations as

$$z^{(2)} = W^{(1)}x + b^{(1)}$$
$$a^{(2)} = f(z^{(2)})$$
$$z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$$
$$h_{W,b}(x) = a^{(3)} = f(z^{(3)})$$

Artificial Neural Networks

$$
\begin{aligned}
a_1^{(2)} &= f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)}) \\
a_2^{(2)} &= f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}) \\
a_3^{(2)} &= f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)}) \\
h_{W,b}(x) = a_1^{(3)} &= f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})
\end{aligned}
$$

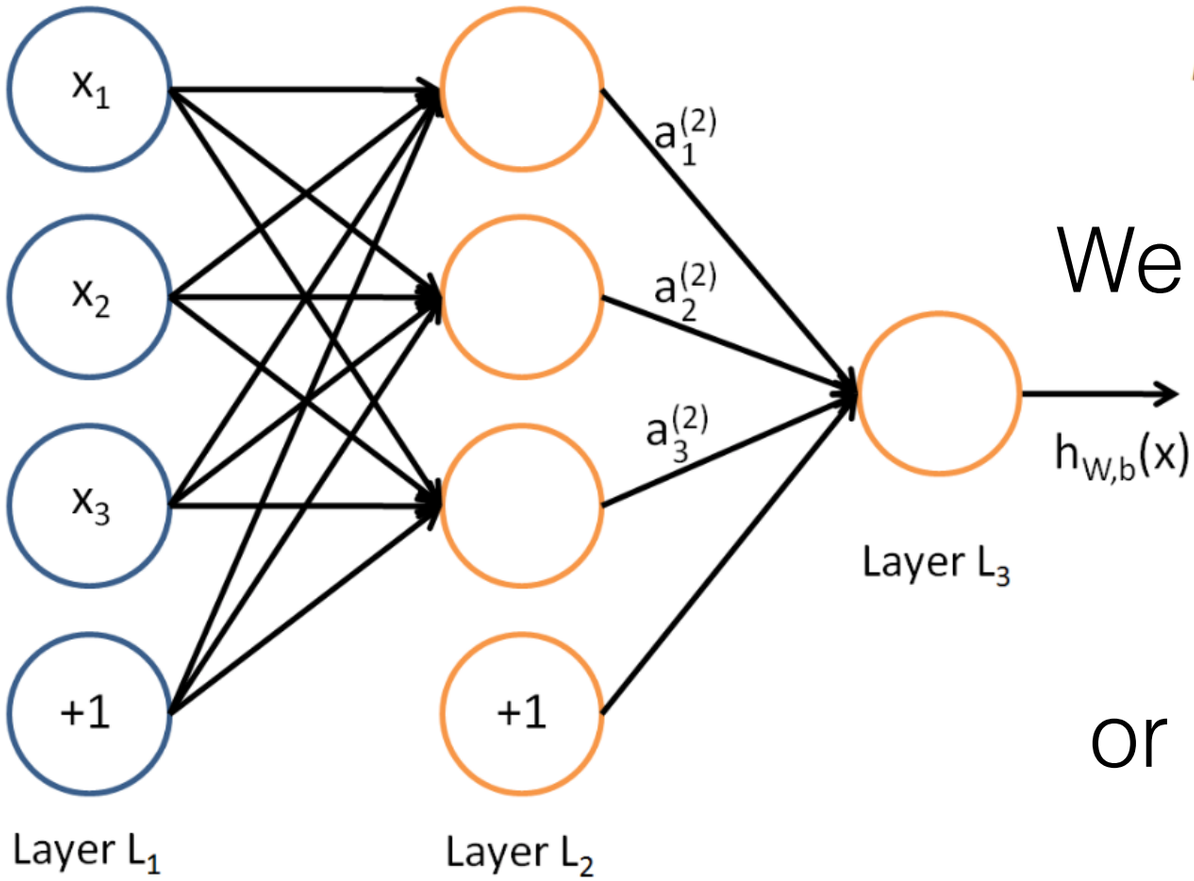


We can simplify these equations as

$$
\begin{aligned}
z^{(2)} &= W^{(1)}x + b^{(1)} \\
a^{(2)} &= f(z^{(2)}) \\
z^{(3)} &= W^{(2)}a^{(2)} + b^{(2)} \\
h_{W,b}(x) &= a^{(3)} = f(z^{(3)})
\end{aligned}
$$

or more generally as

Artificial Neural Networks



$$a_1^{(2)} = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)})$$
$$a_2^{(2)} = f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)})$$
$$a_3^{(2)} = f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)})$$
$$h_{W,b}(x) = a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})$$

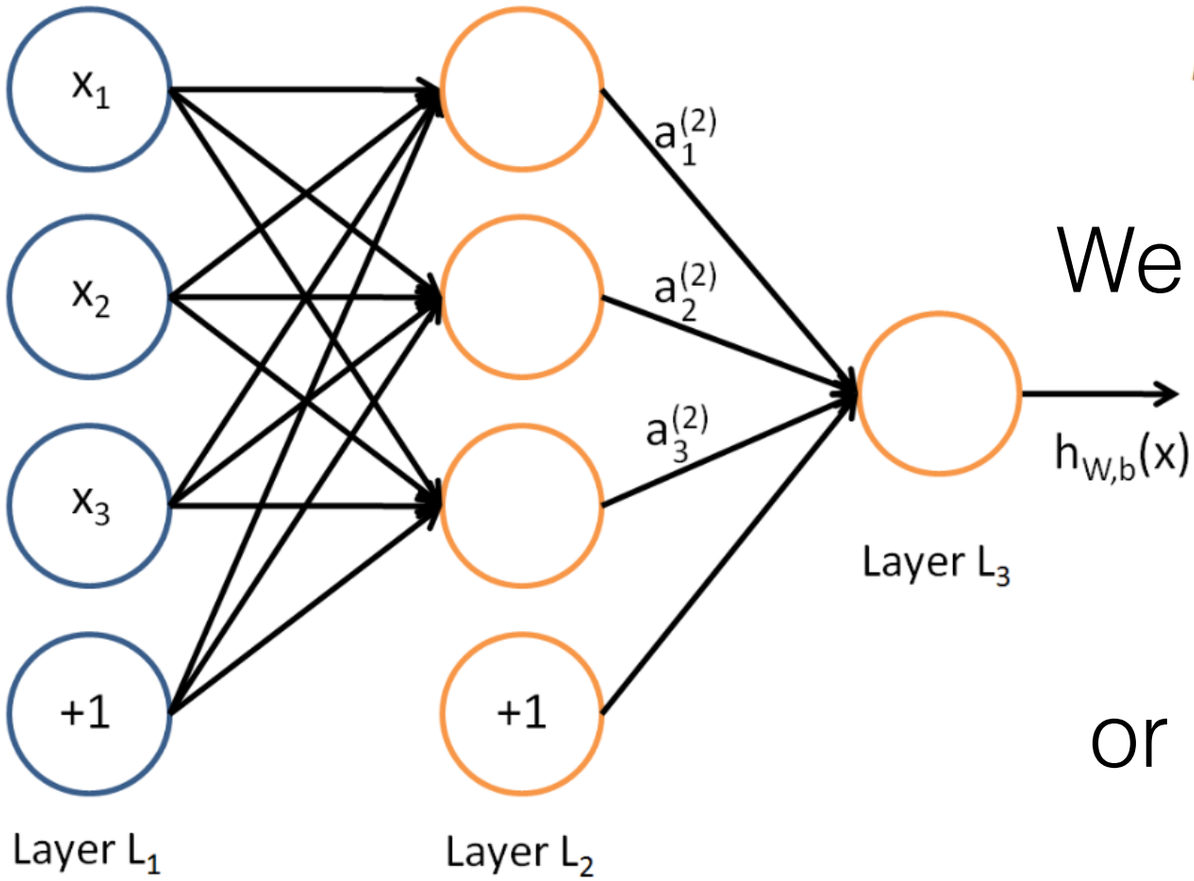
We can simplify these equations as

$$z^{(2)} = W^{(1)}x + b^{(1)}$$
$$a^{(2)} = f(z^{(2)})$$
$$z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$$
$$h_{W,b}(x) = a^{(3)} = f(z^{(3)})$$

or more generally as

$$z^{(l+1)} = W^{(l)}a^{(l)} + b^{(l)}$$
$$a^{(l+1)} = f(z^{(l+1)})$$

Artificial Neural Networks



$$a_1^{(2)} = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)})$$
$$a_2^{(2)} = f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)})$$
$$a_3^{(2)} = f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)})$$
$$h_{W,b}(x) = a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})$$

We can simplify these equations as

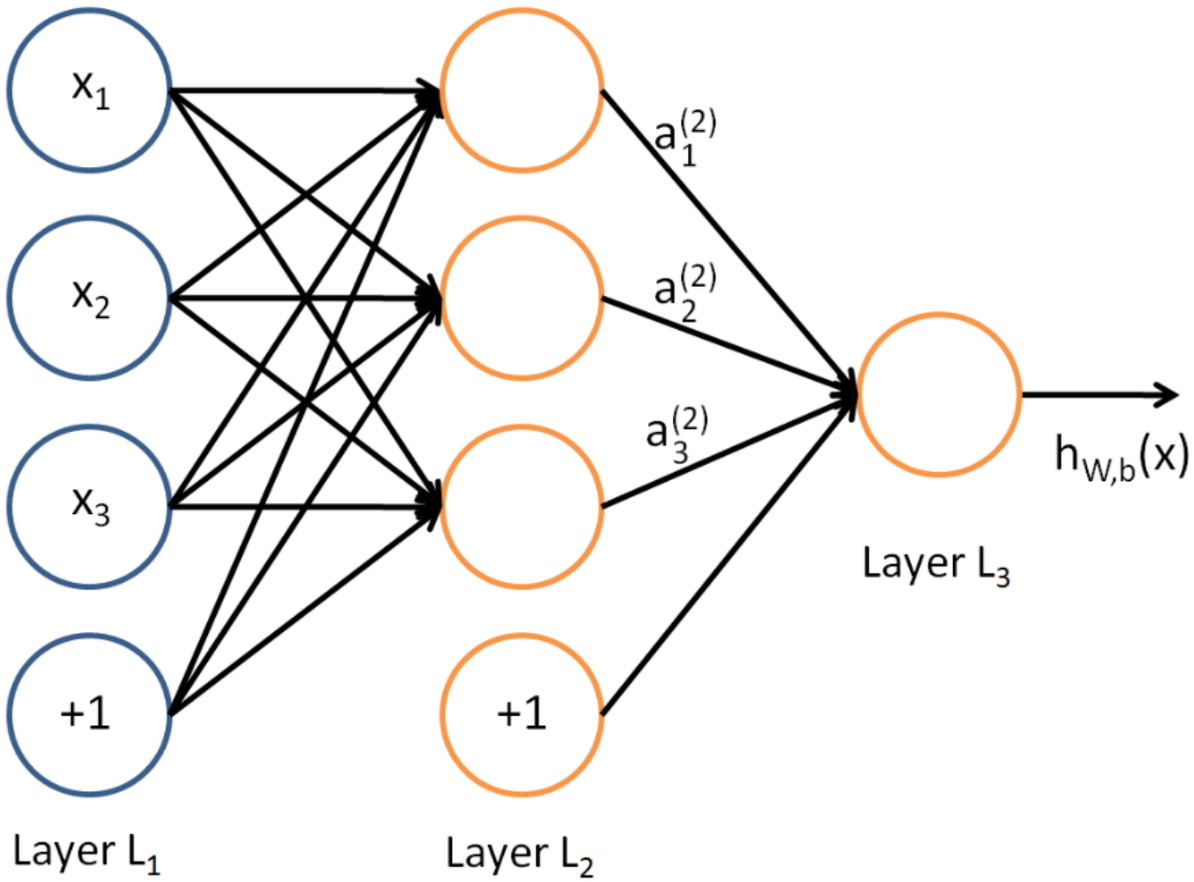
$$z^{(2)} = W^{(1)}x + b^{(1)}$$
$$a^{(2)} = f(z^{(2)})$$
$$z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$$
$$h_{W,b}(x) = a^{(3)} = f(z^{(3)})$$

or more generally as

$$z^{(l+1)} = W^{(l)}a^{(l)} + b^{(l)}$$
$$a^{(l+1)} = f(z^{(l+1)})$$

This is a matrix operation.

Artificial Neural Networks



Input: Set the corresponding activation x_1 for the input layer and initialize weights.

Feedforward: For each l=2,3, …,L compute

$$z^{(l+1)} = W^{(l)} a^{(l)} + b^{(l)}$$
$$a^{(l+1)} = f(z^{(l+1)})$$

Ouput: compute cost function.

$$J(W, b; x, y) = \frac{1}{2} \|h_{W,b}(x) - y\|^2$$

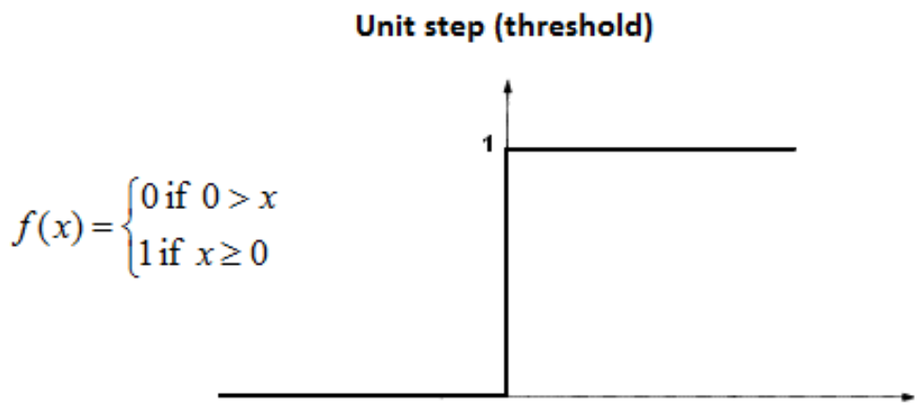
Gradient: compute gradient.

Backpropagate: update the weights. Go to feedforward.

Artificial Neural Networks

Activation Function

- translates the input signals to output signals.
- inspired by biological activation that turns information transfer on/off from one neuron to next.
- the simplest form is binary, meaning either the neuron is firing or not.

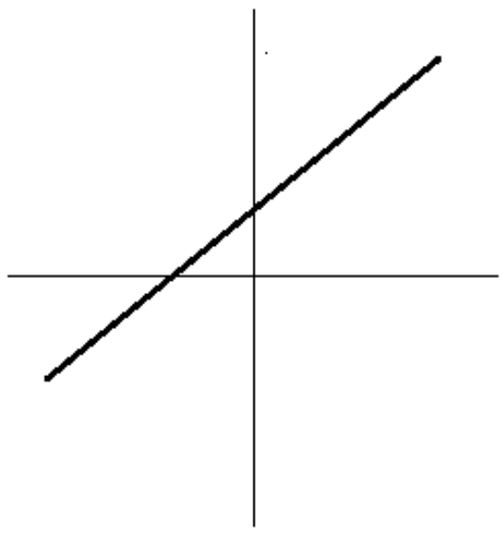


heaviside step function

- obviously, this function transfers only binary information.

Artificial Neural Networks

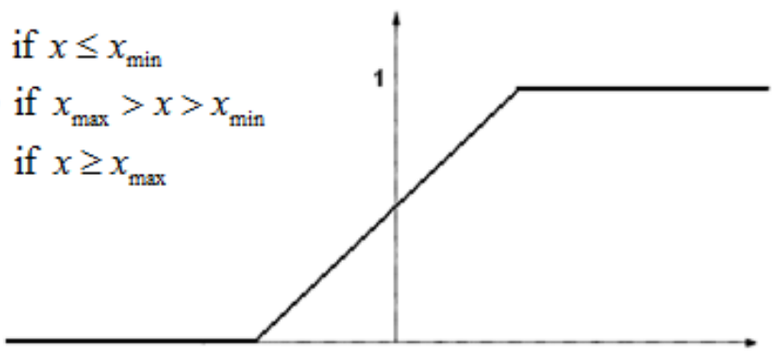
Linear



- a linear activation function transforms the weighted sum inputs of the neuron to an output using a linear response.
- this model has unstable convergence because neuron inputs along favoured paths tend to increase without bound, as this function is not normalizable.

Piecewise Linear

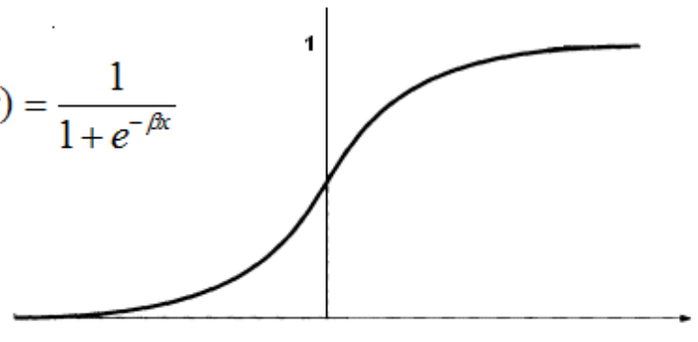
$$f(x) = \begin{cases} 0 & \text{if } x \le x_{min} \\ mx+b & \text{if } x_{max} > x > x_{min} \\ 1 & \text{if } x \ge x_{max} \end{cases}$$



- overcomes the drawback of linear.
- the function is normalizable.

Sigmoid

$$f(x) = \frac{1}{1+e^{-\beta x}}$$



- overcomes the drawback of piecewise linear.
- smooth transition
- better resamples biological response
- popular choice

Artificial Neural Networks

Gradient-Descent Procedure

A method that modifies the weights to reduce the cost function:

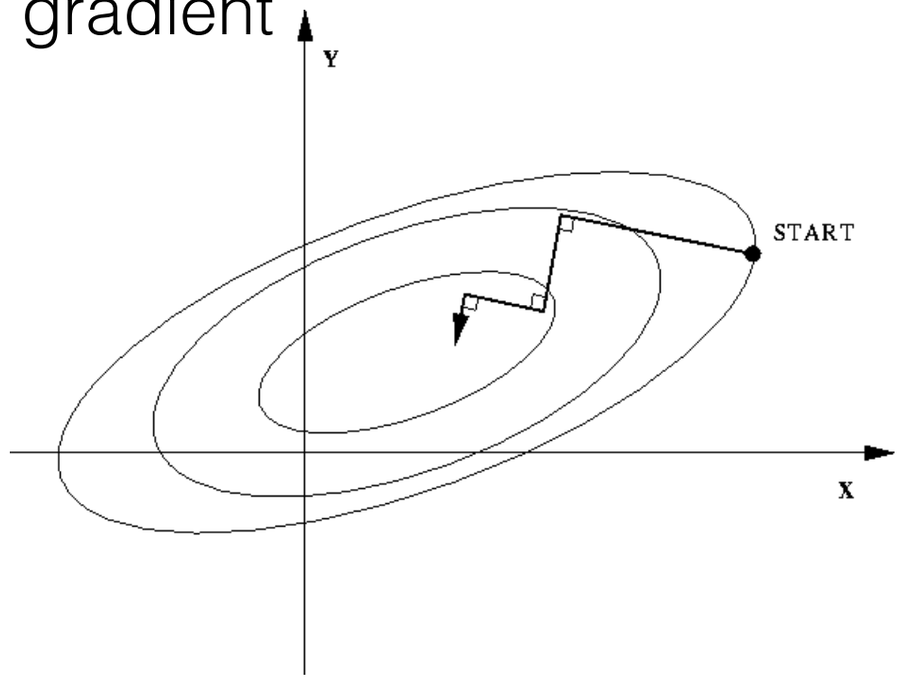
$$J(W, b; x, y) = \frac{1}{2}\|h_{W,b}(x) - y\|^2$$

Algorithm:

step size

$$w_{ij}[n+1] = w_{ij}[n] + \eta g(w_{ij}[n])$$

improved weight current weight gradient



Artificial Neural Networks

Deep Neural Networks

If we add more hidden layers we are doing deep learning

<u>Pros</u>: It can model more complex patterns of the data
<u>Cons</u>: It is prone to overfitting and increased computing time