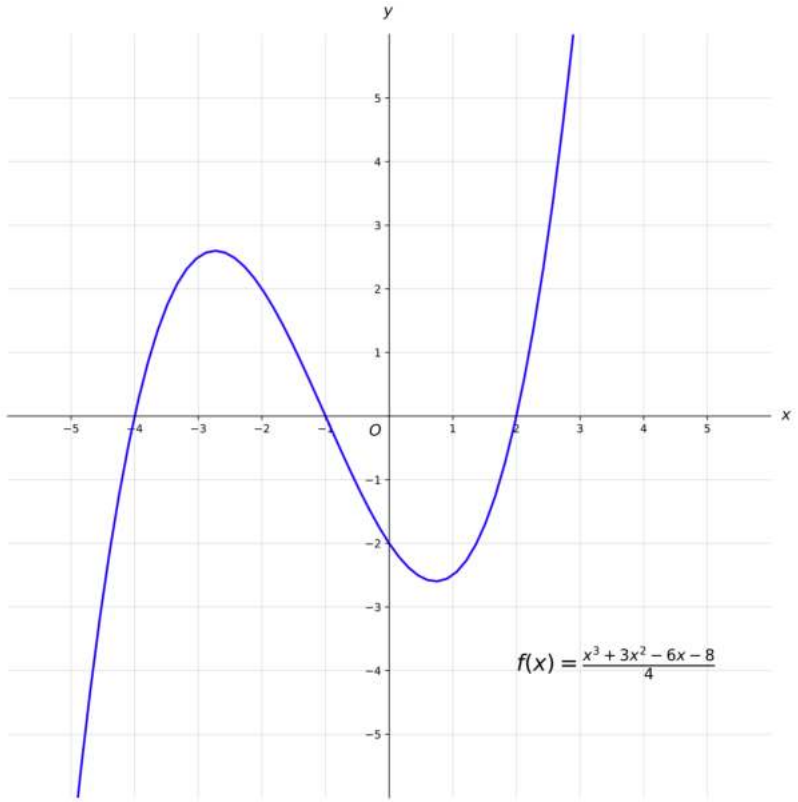
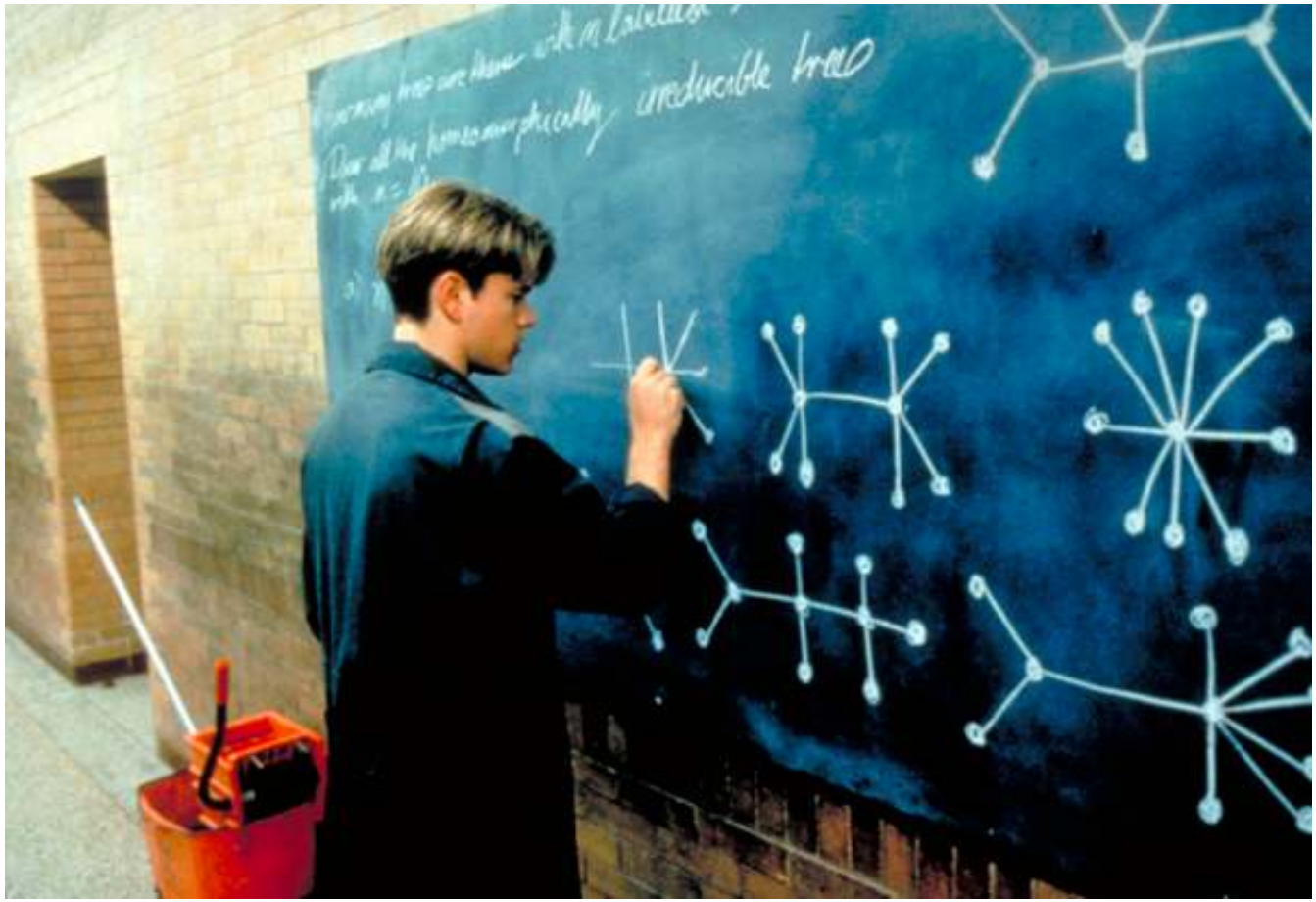
A very brief incursion into GNNs…

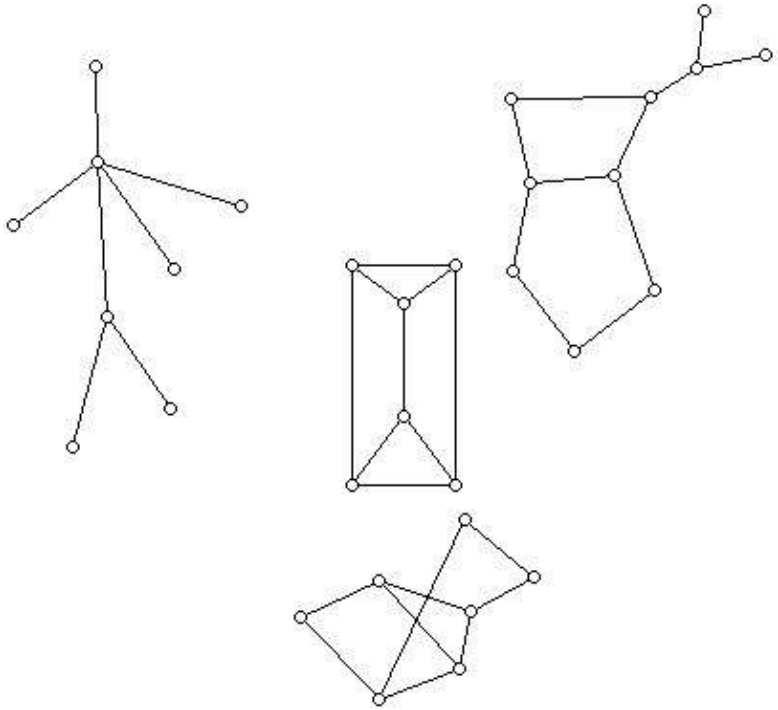
What is a graph?

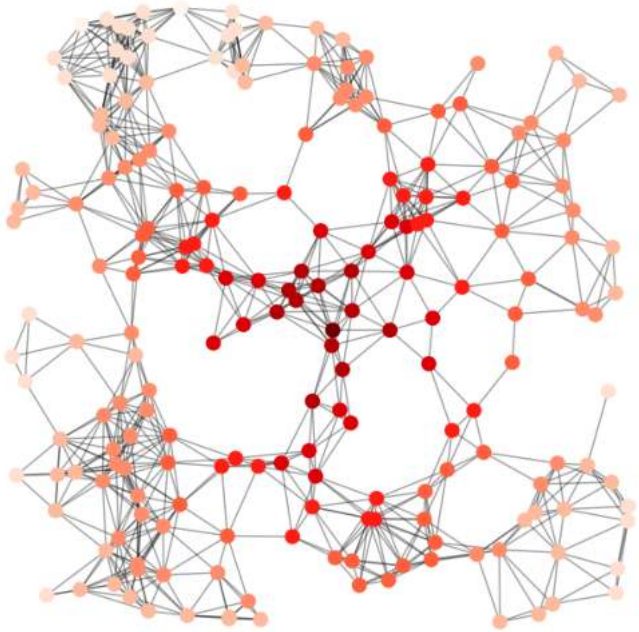

y
x
O
$f(x) = \frac{x^3 + 3x^2 - 6x - 8}{4}$

?


irreducible trees

What is a graph?

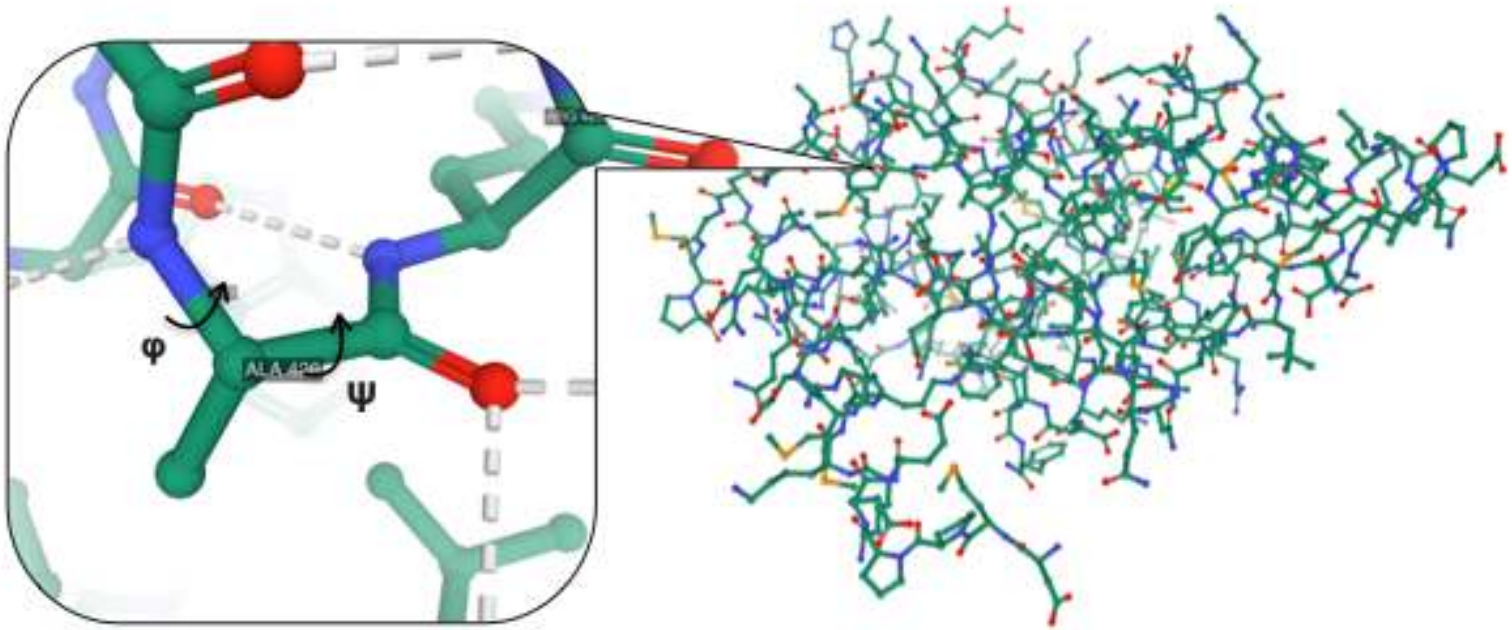




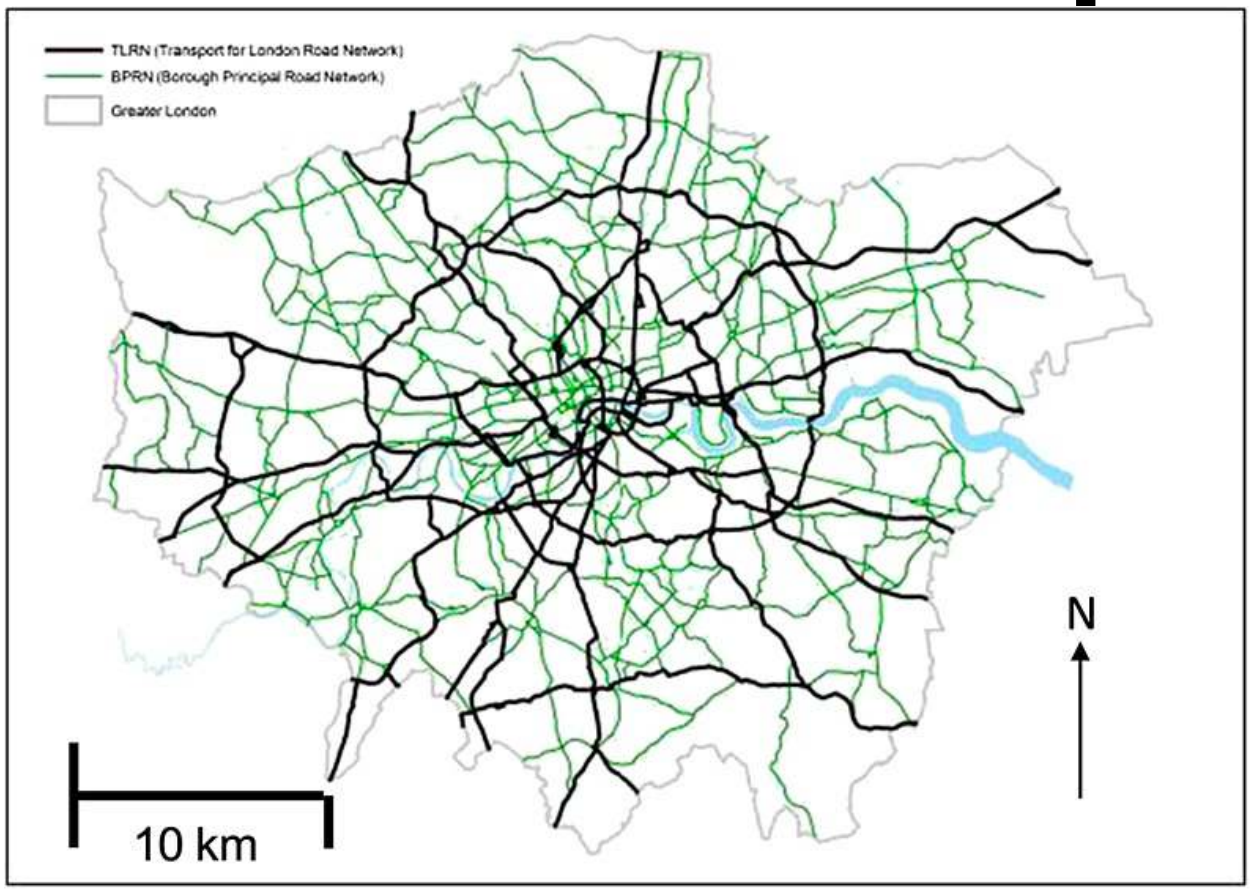
!!!

Graphs in the wild

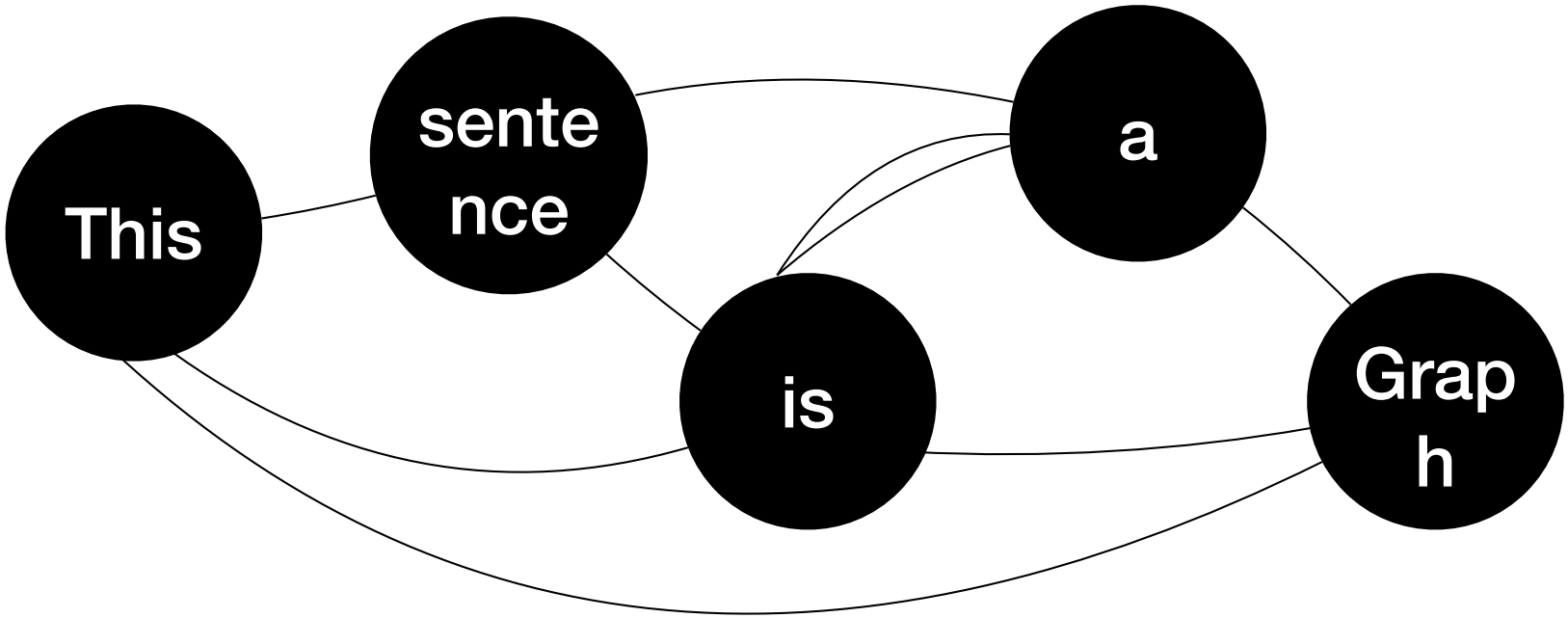
Molecules are Graphs



Traffic is a Graph

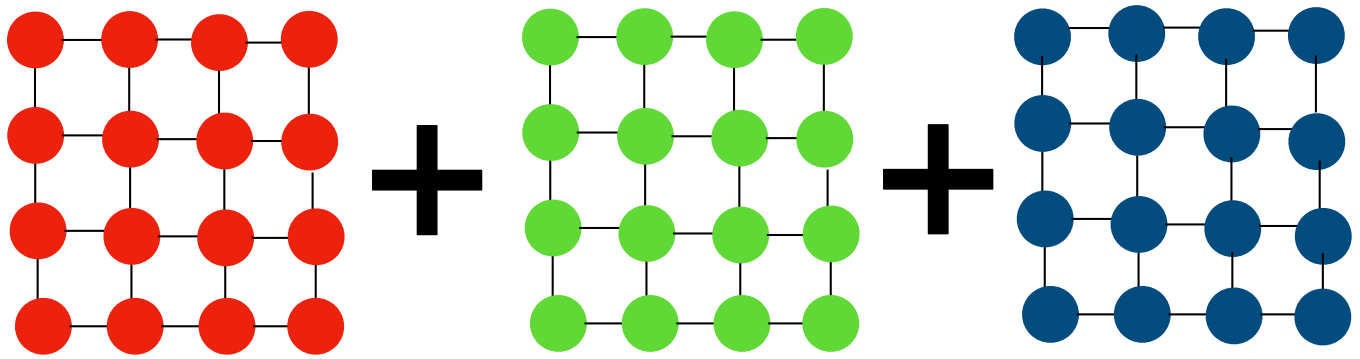


This sentence is a Graph



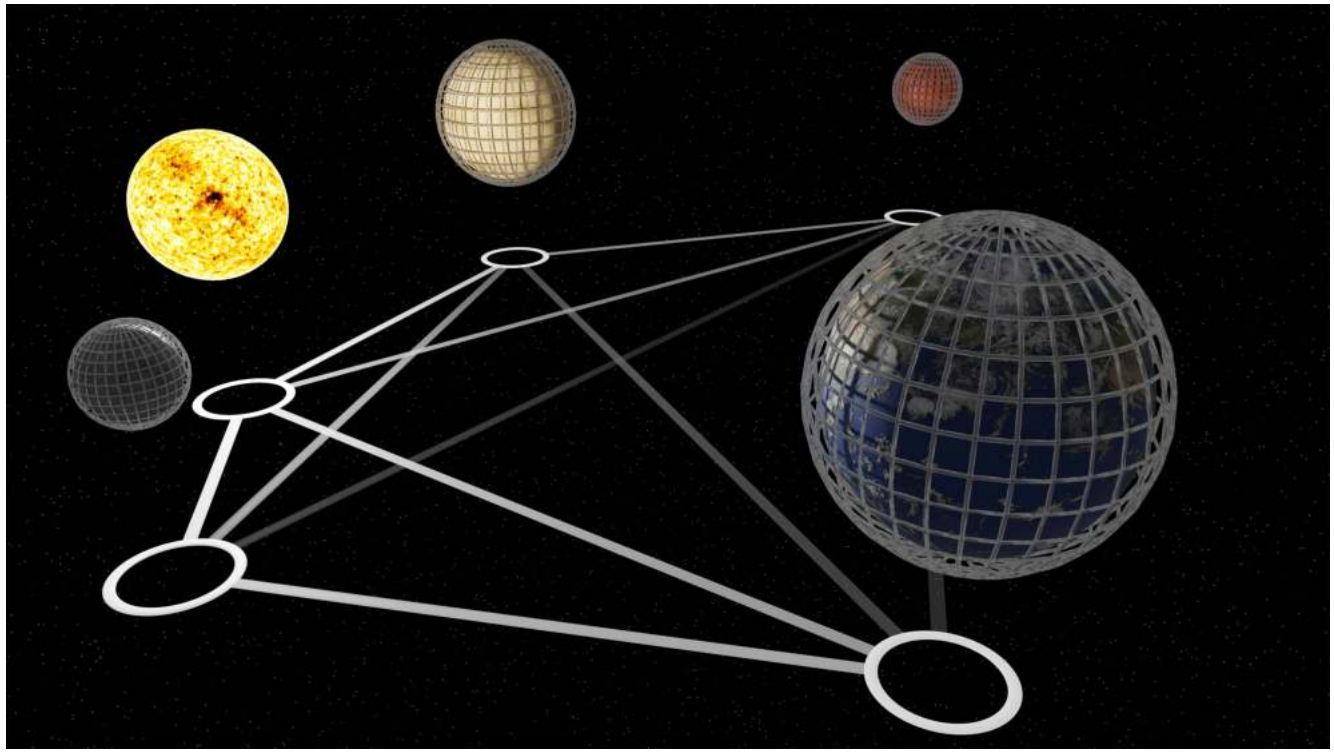
Also, transformers are a special case of GNNs!!

Pictures are Graphs

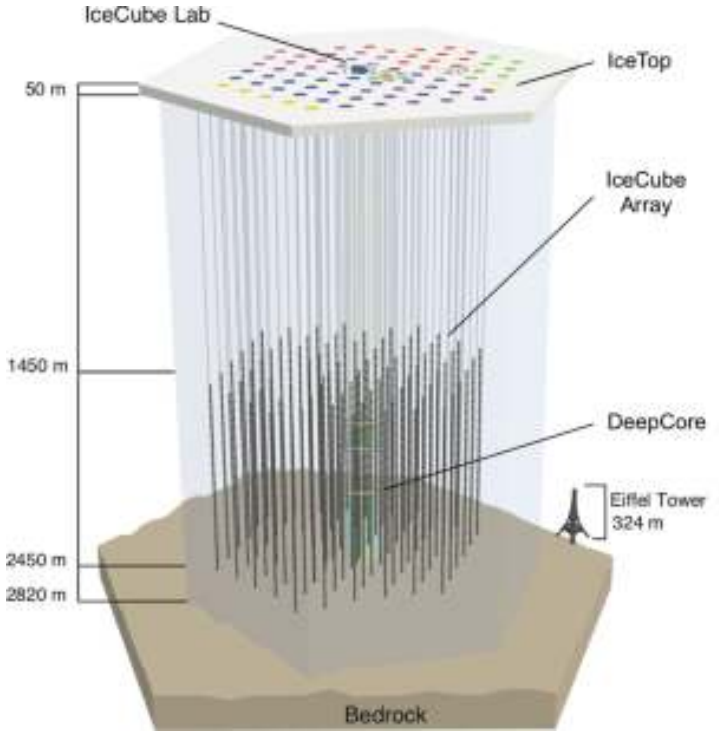


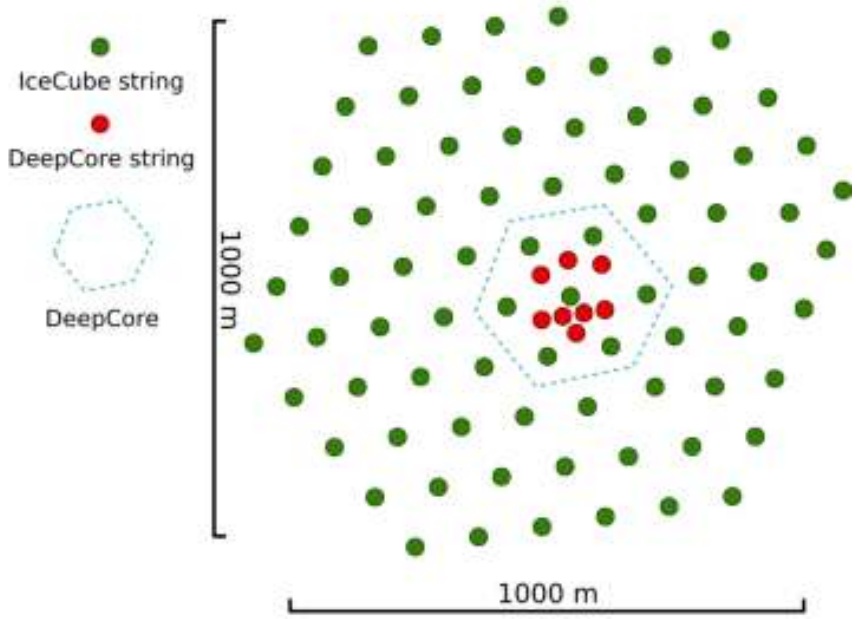
The Solar System is a Graph

Or at least a natural abstraction

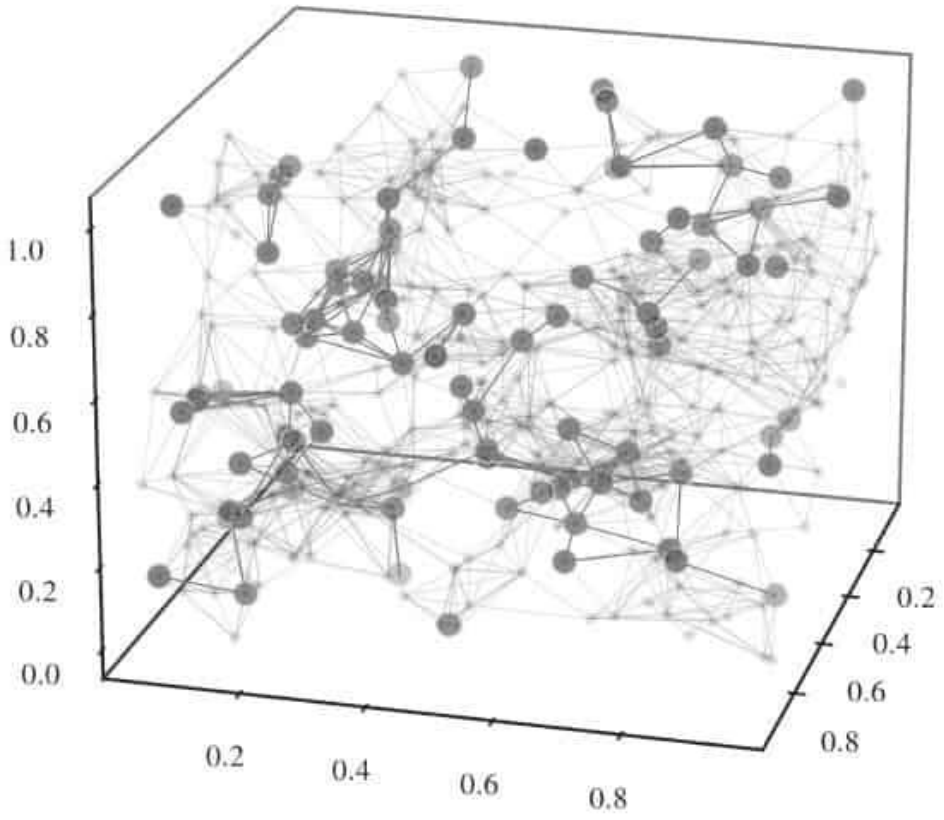


Neutrino Observatories are Graphs

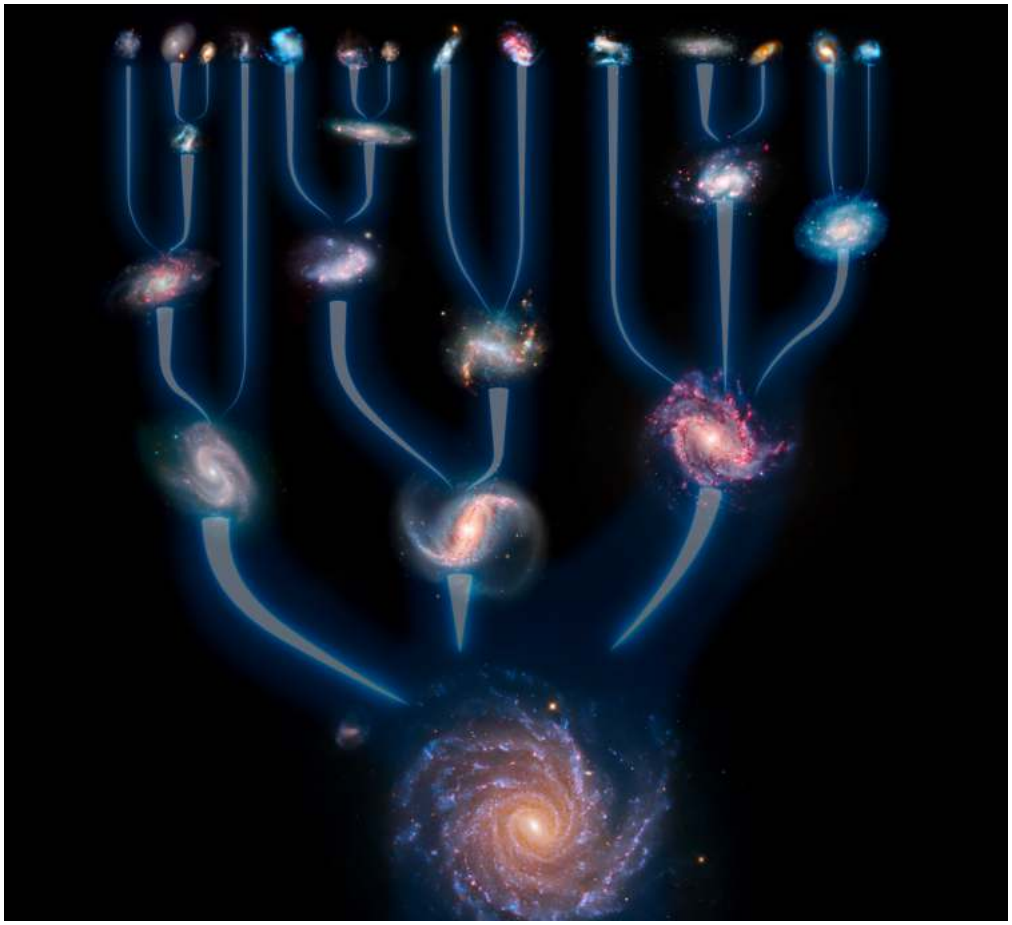


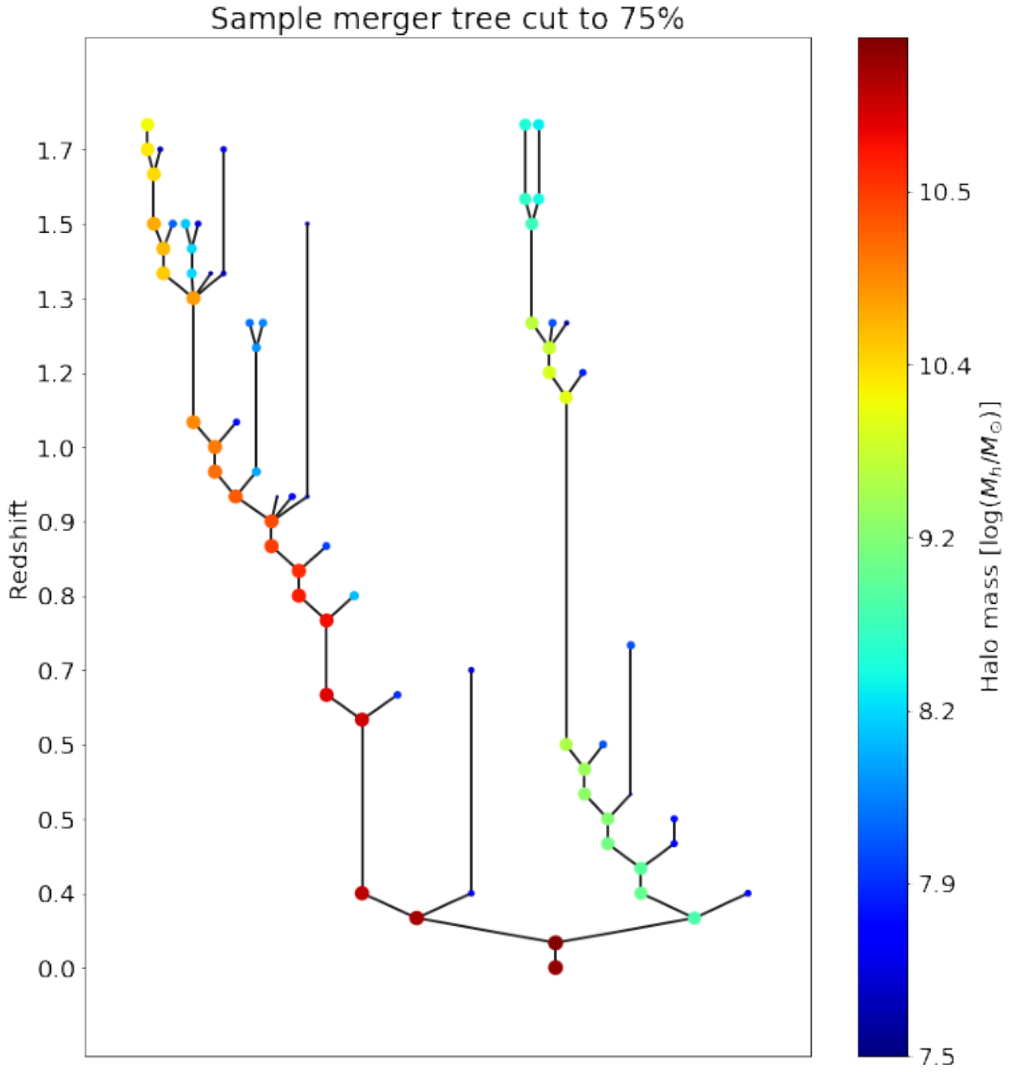


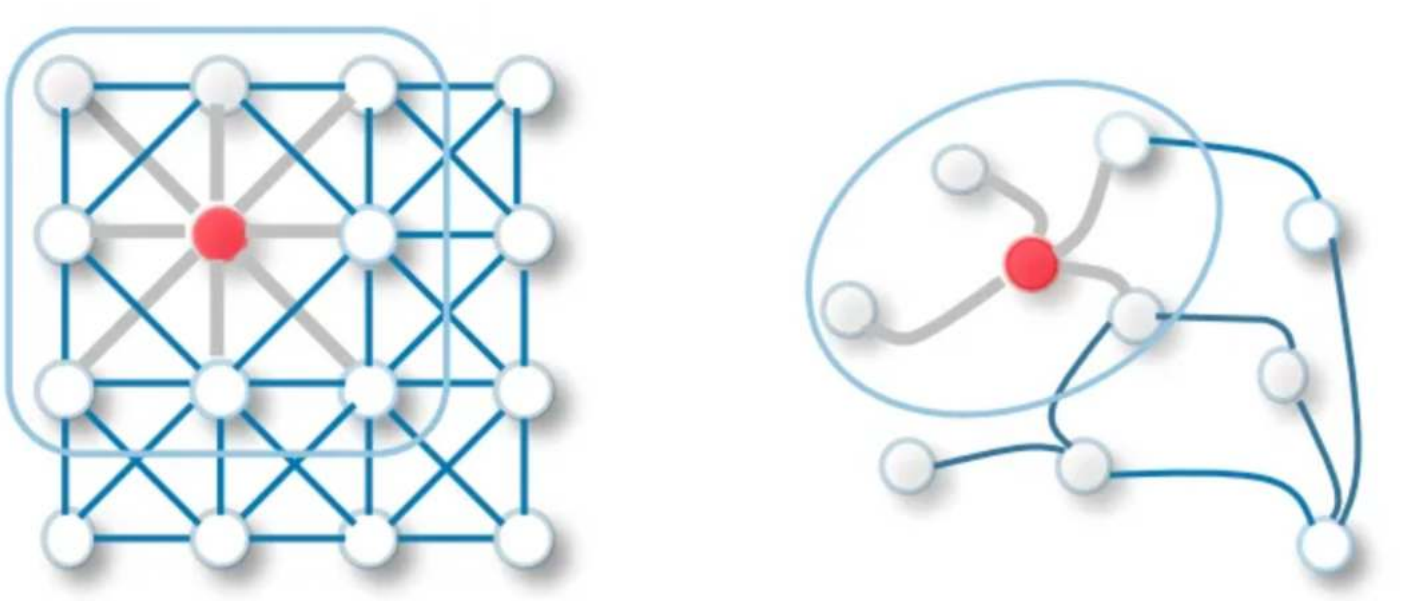
Large Scale Structure is a Graph



Galaxy Merger trees are Graphs




Sample merger tree cut to 75%
Redshift
1.7
1.5
1.3
1.2
1.0
0.9
0.8
0.7
0.5
0.5
0.4
0.0
Halo mass [log($M_h/M_\odot$)]
10.5
10.4
9.2
8.2
7.9
7.5

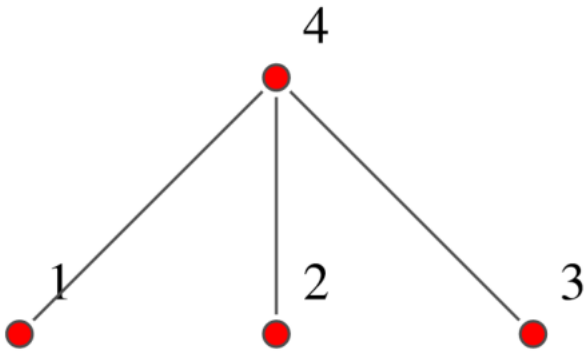


The main difference compared to images
is in the definition of neighbours

The harder question is: What is not a graph?

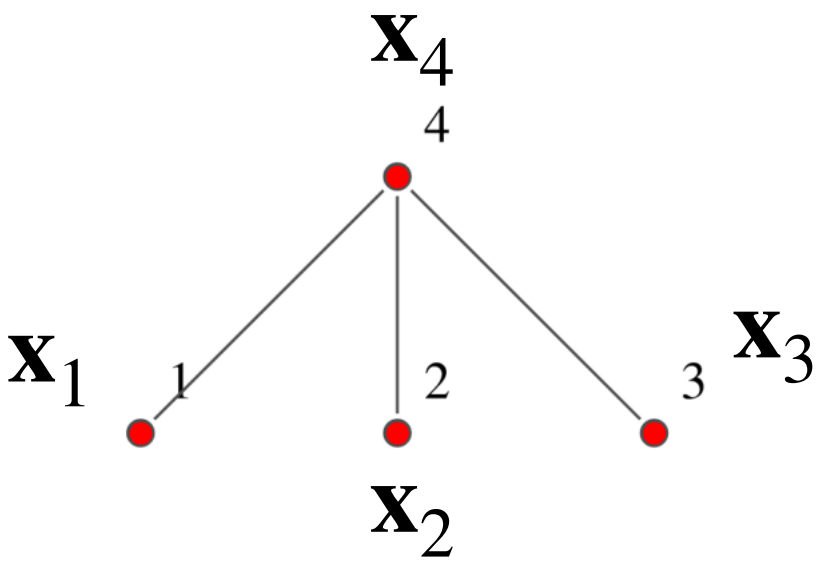
What makes up a graph?

Nodes and node features



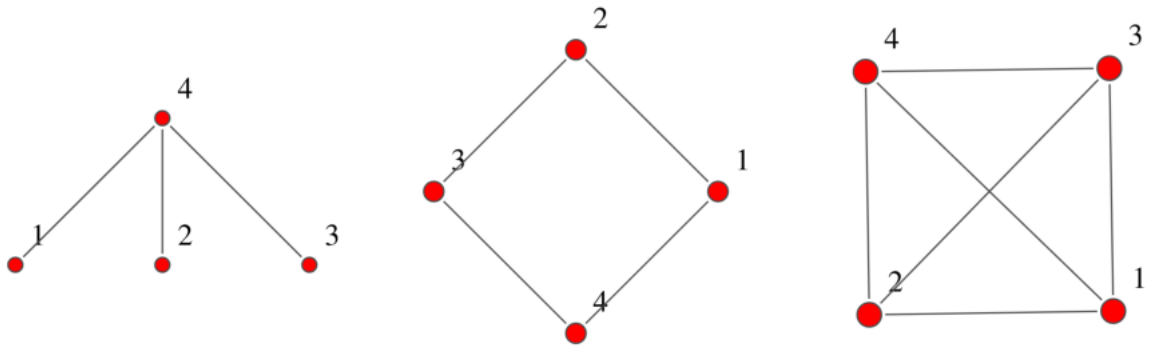
What makes up a graph?

Nodes and node features



What makes up a graph?

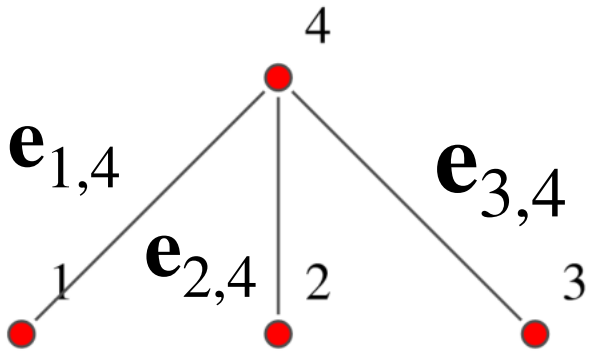
Nodes and node features



$$\mathbf{X} = \left[\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\right]^\top \rightarrow \quad \mathbf{X} = \left[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{|\mathcal{V}|}\right]^\top \rightarrow \quad \mathbf{X} = \left\{\mathbf{x}_i \mid i \in N_v\right\} \in \mathbb{R}^{N_v \times d_v}$$

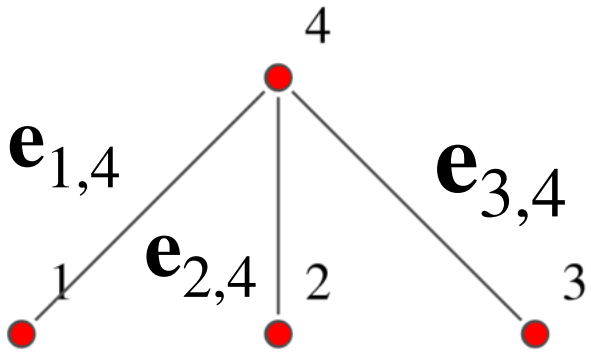
What makes up a graph?

Edges and edge features



What makes up a graph?

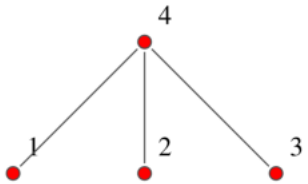
Edges and edge features

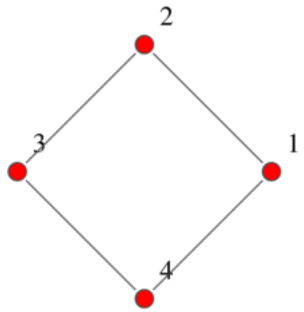


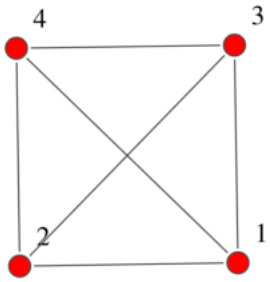
$$\mathbf{E} = \left[\mathbf{e}_{12}, \mathbf{e}_{13}, \mathbf{e}_{14}\right]^\top \rightarrow \quad \mathbf{E} = \left[\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{|\mathscr{E}|}\right]^\top \rightarrow \quad \mathbf{E} = \left\{\mathbf{e}_i \mid i \in N_e\right\} \in \mathbb{R}^{N_e \times d_e}$$

What makes up a graph?

The Adjacency Matrix







$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

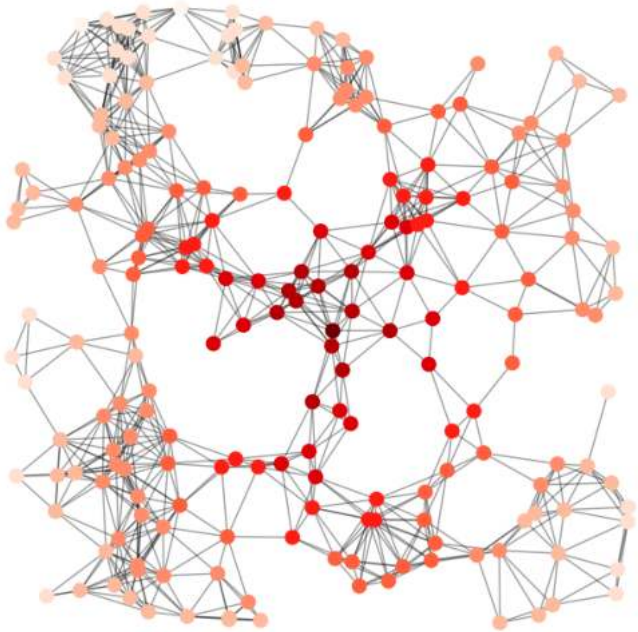
Dense and Sparse Adjacency Matrices

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Sending nodes || Receiving nodes

→ [[0,1,2,3,3,3], [3,3,3,0,1,2]]

Global Features



Could be physical properties e.g.

$$\mathbf{U} = [\Omega_m, \sigma_8]$$

Statistical properties,
e.g. manually inserted

$$\mathbf{U} = \left[\mu_{\mathbf{x}}, \sigma_{\mathbf{x}}, max(\mathbf{x}), min(\mathbf{x}) \ldots P_n(\mathbf{x})\right]$$

Or completely learnable

Summarizing…

- Graph defined as $G(V, E, U)$ with G being the graph, V being the set of nodes, E being the set of edges and U being the global feature vector
- Adjacency matrix (A) encodes the structure of the graph and is a binary matrix

Graph Neural Networks

And why they work so well

- Adaptability
- Physical Principles

- Almost all physical systems have structure - very little of which naturally exists on a Euclidean grid

Locality constraints

The edges of a graph naturally define the concept of a local **neighborhood**

$$\mathcal{N}_u = \{v \mid (u, v) \in \mathbf{E} \vee (v, u) \in \mathbf{E}\}$$

Which allows us to allow us to define local functions, which work on a given node and its neighborhood.

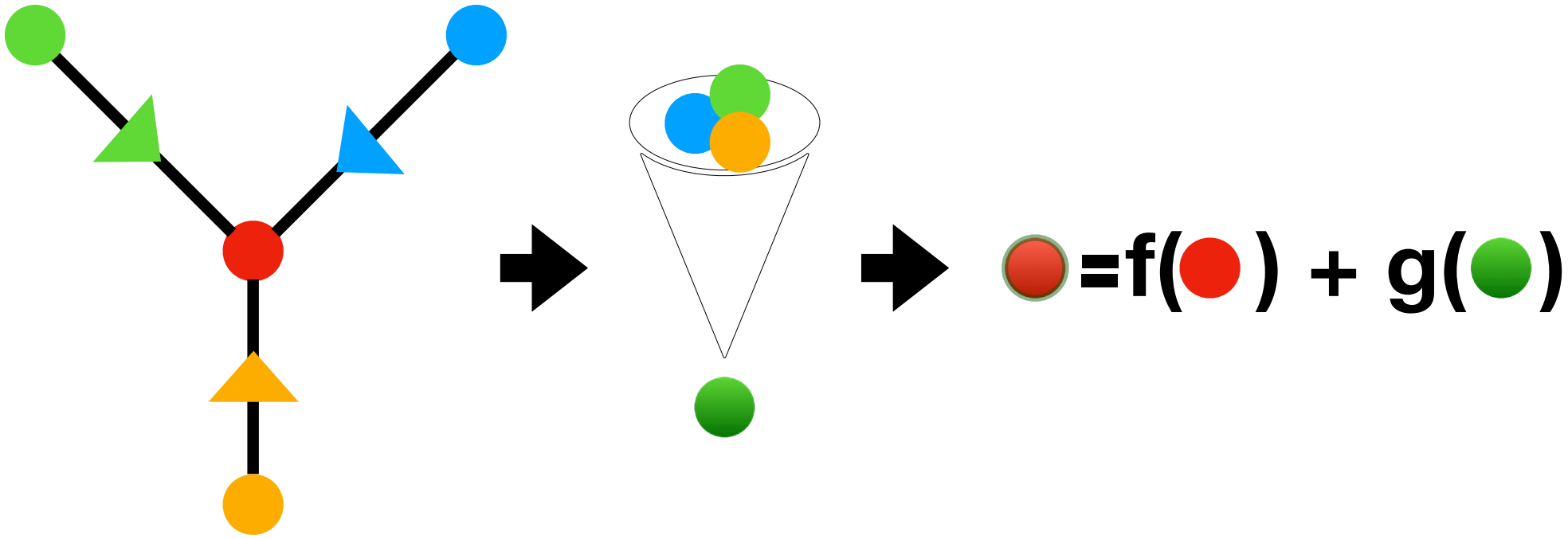
$$\phi\left(\mathbf{x}_u, \mathbf{X}_{\mathcal{N}_u}\right)$$

Graph Neural Network Layers

Increasing complexity/ expressivity/ training time

- Convolutional Layer $\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} c_{vu}\psi\left(\mathbf{x}_v\right)\right)$
- Attentional Layer $\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} a(\mathbf{x}_u, \mathbf{x}_v, \mathbf{e}_{uv})\psi\left(\mathbf{x}_v\right)\right)$

Looks complicated, but!



Looks complicated, but!

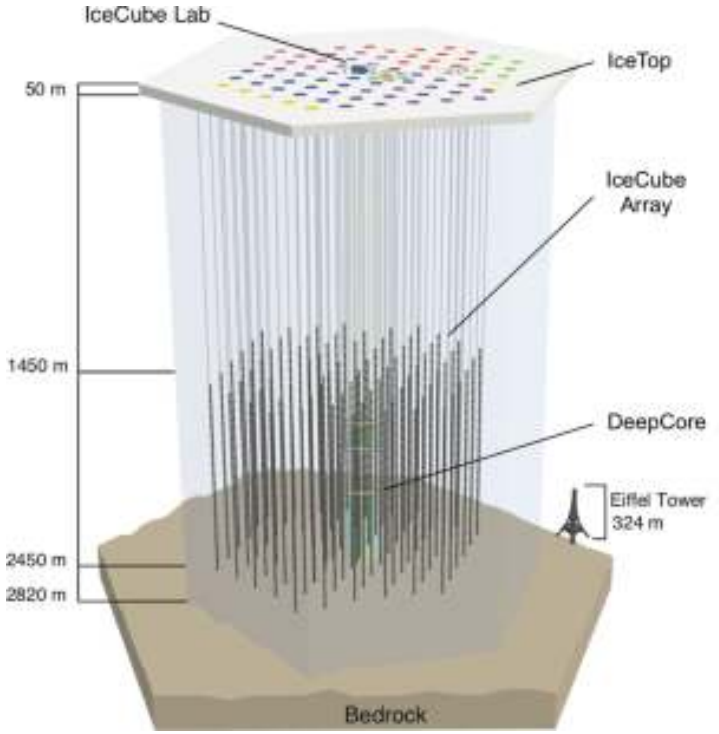
Essentially, these are all variants of:

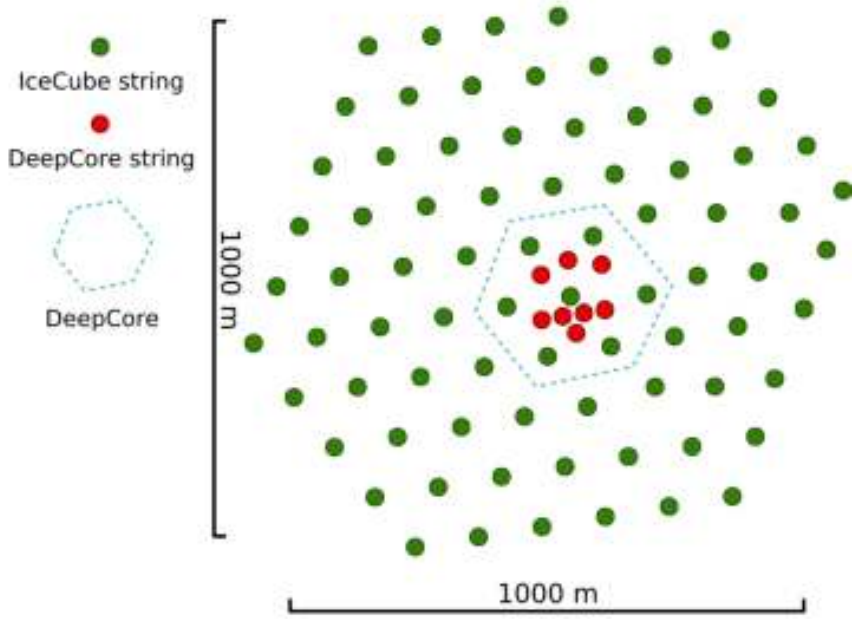
1. Define a node function f and a neighborhood function g (f and g can be the same function)
2. Act with f on the node and with g on the neighborhood

Graphs are good representations of physical systems

In modern physics … a central theme will be a Geometric Principle: The laws of physics must all be expressible as geometric (coordinate-independent and reference frame-independent) relationships between geometric objects (scalars, vectors, tensors, ...) that represent physical entities.

How would you fit this on a grid?





Physics on Graphs

- A natural abstraction of physical systems and inherently local -> easier physical interpretations/separability
- Embeds inductive biases easily by restructuring the graph -> more efficient learning and no need to learn things we already know
- Can embed permutational, rotational, translational and reflectional symmetries

Must-have symmetry

Given some permutation matrix **P**, we want some properties:

For scalar output functions, **invariance** when acting with with **P**

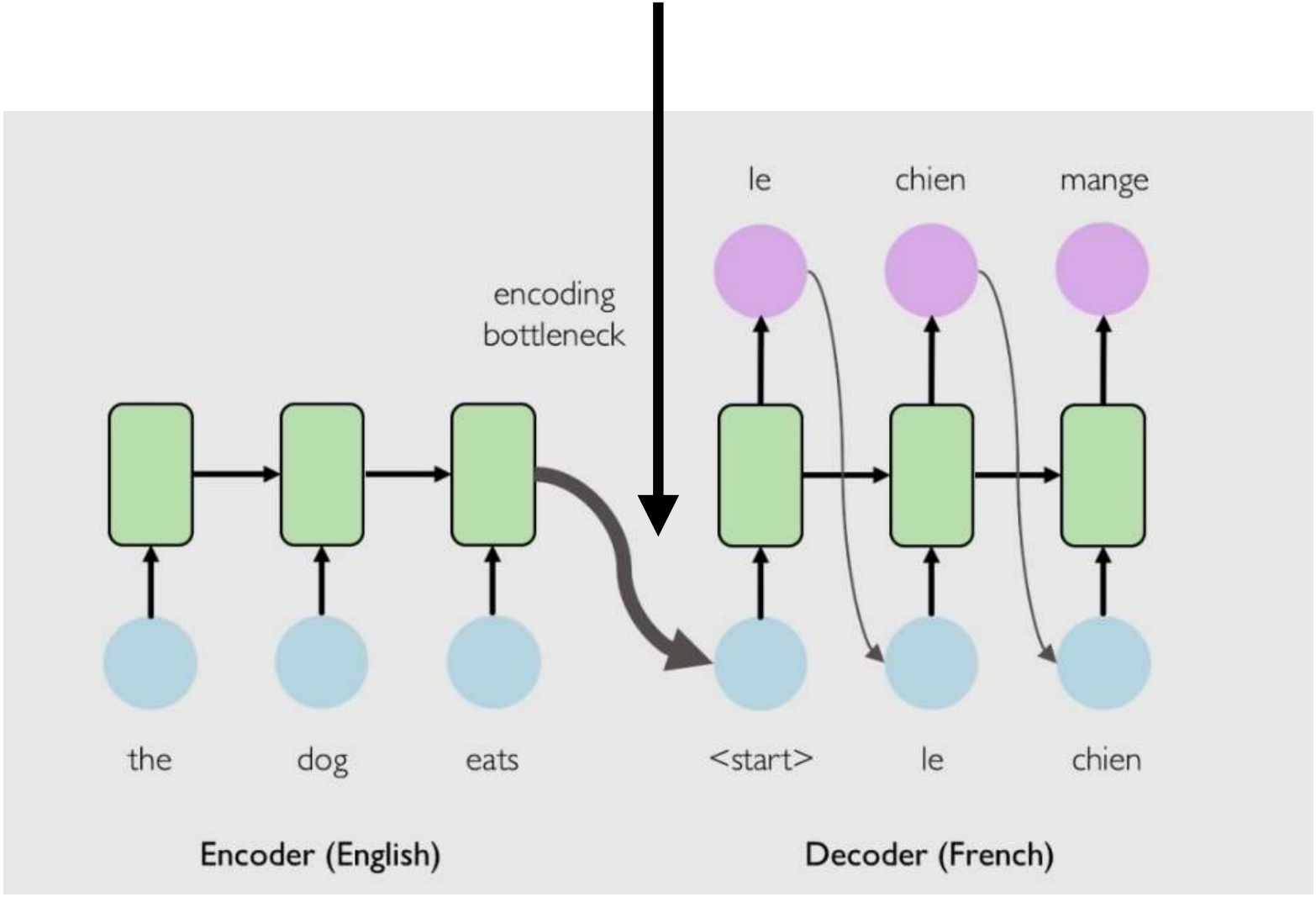
$$f\left(\mathbf{PX}, \mathbf{PAP}^{\top}\right) = f(\mathbf{X}, \mathbf{A}) \text{ (Invariance)}$$

For vector output functions, **equivariance** when acting with with **P**

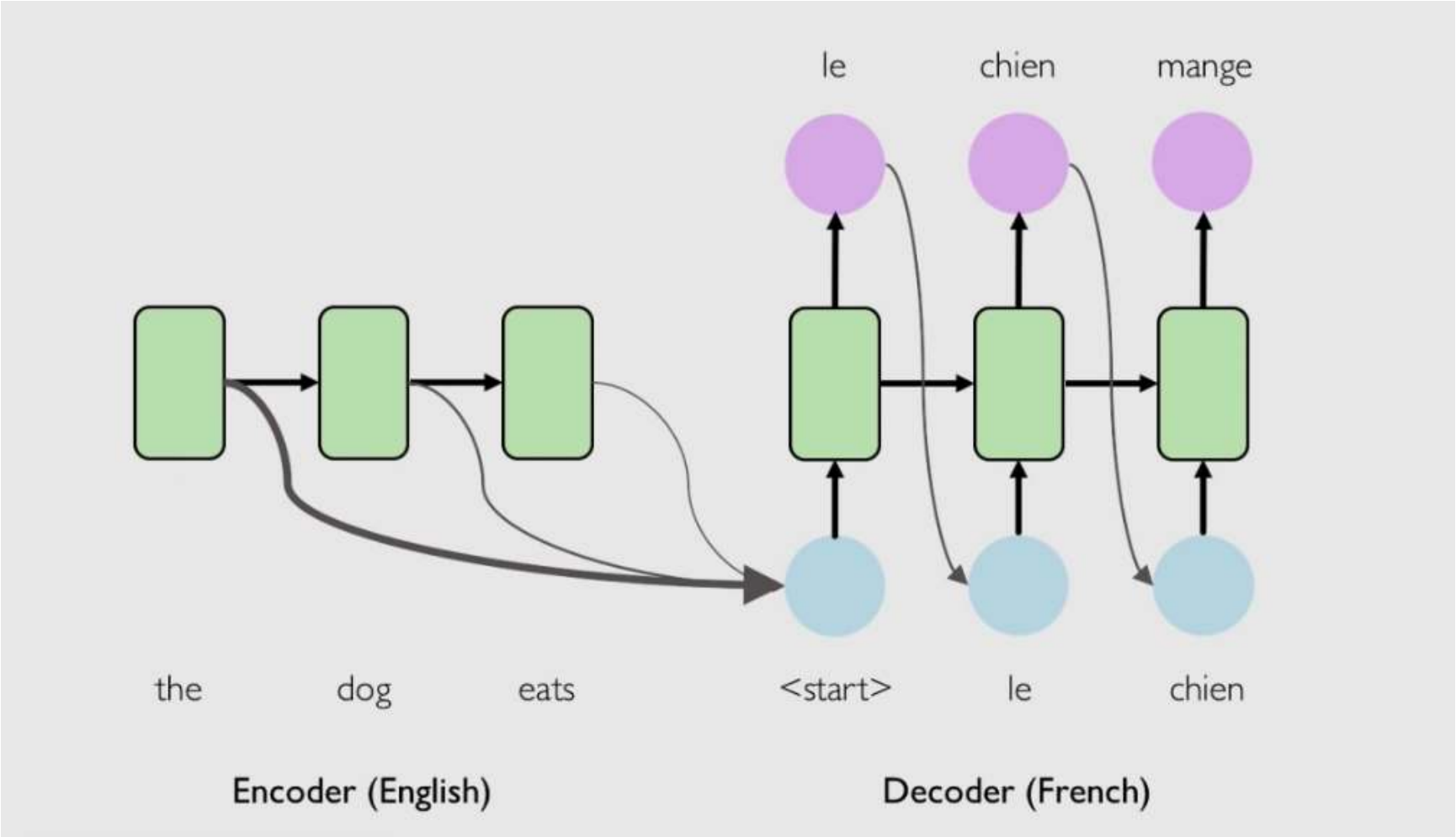
$$\mathbf{F}\left(\mathbf{PX}, \mathbf{PAP}^{\top}\right) = \mathbf{PF}(\mathbf{X}, \mathbf{A}) \text{ (Equivariance)}$$

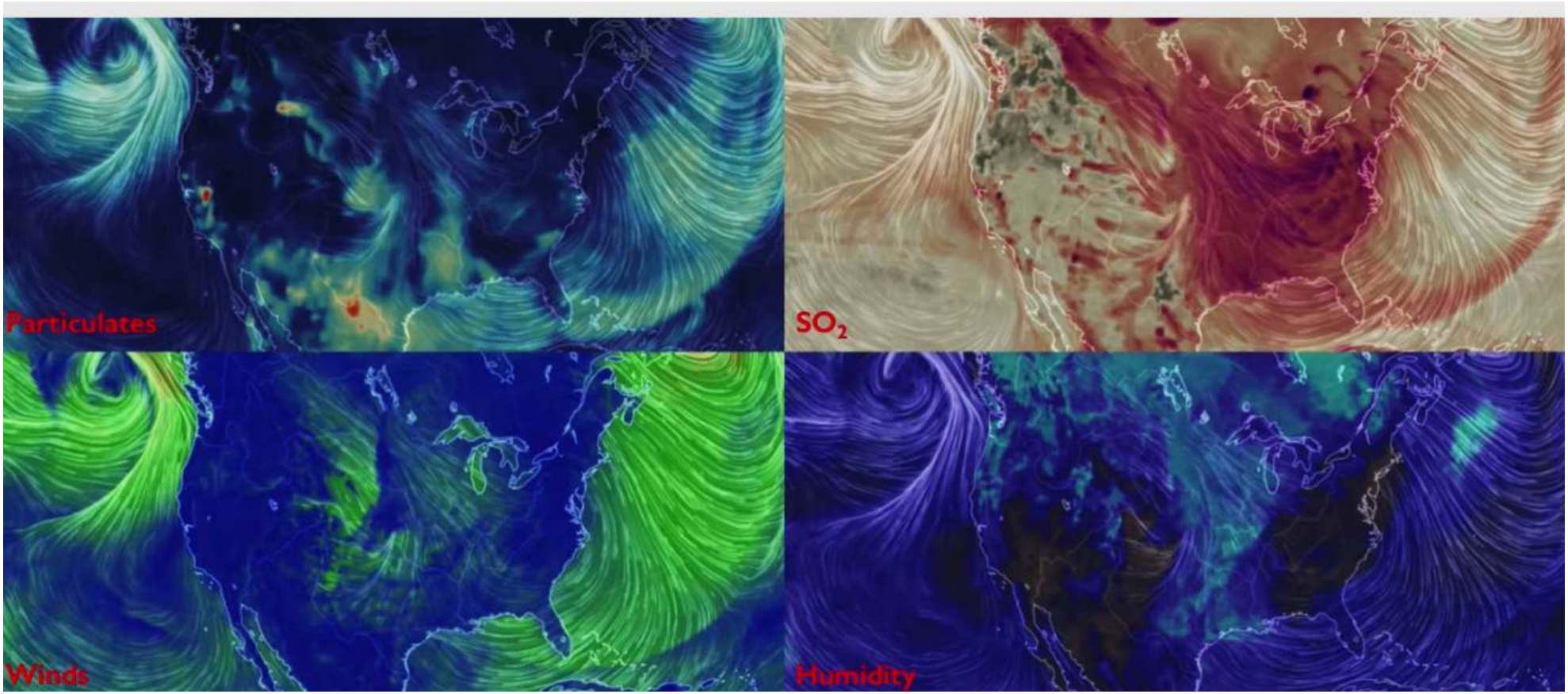
The Unreasonable Efficiency of GNNs for Physics

The entire content is encoded into a single vector
(Large information bottleneck)



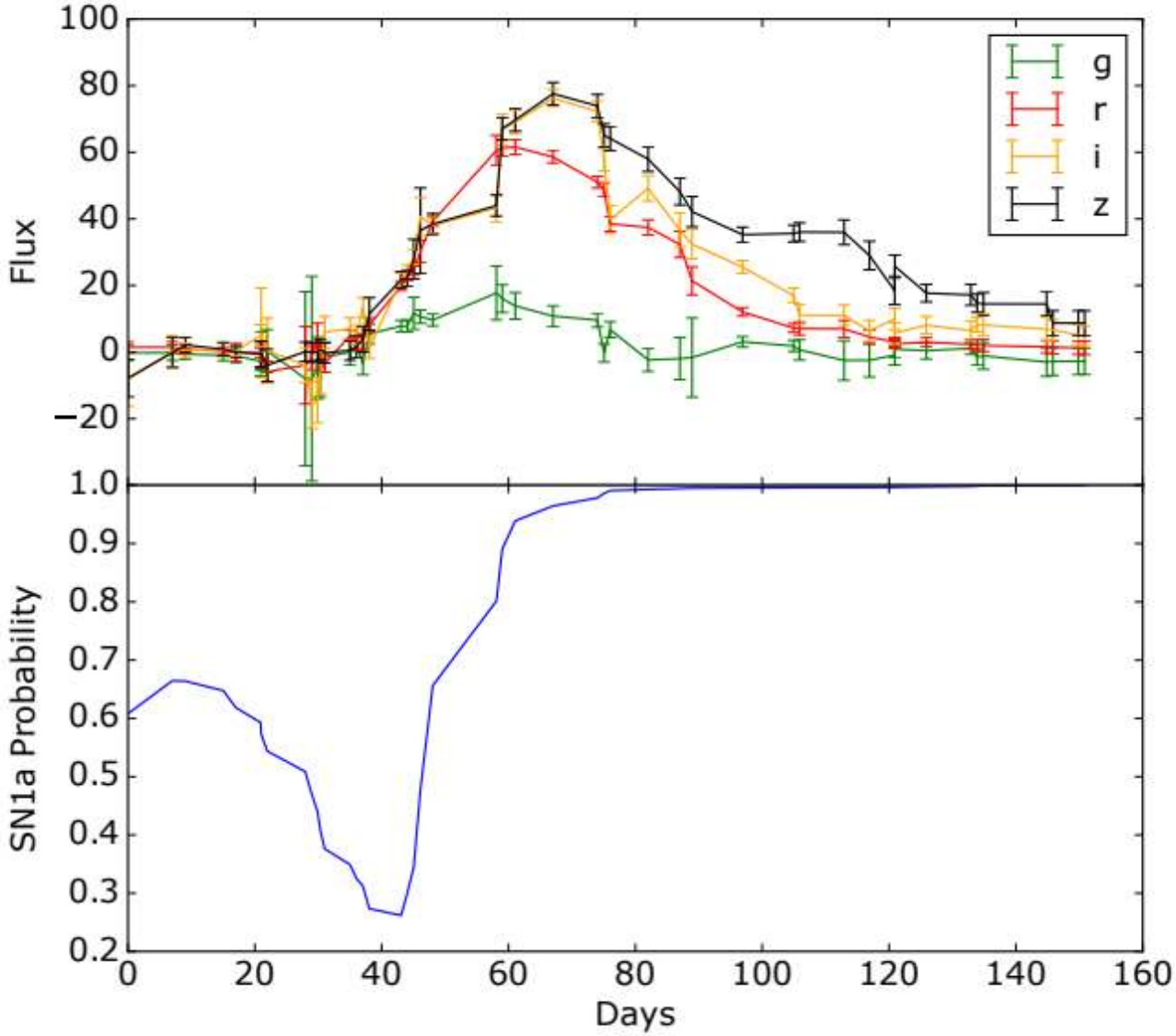
ALL YOU NEED IS ATTENTION: ATTENTION MECHANISMS





ENVIRONMENTAL PREDICTION

SN light curve classification with RNNs



Charnock+17

PART V: A VERY BRIEF INTRODUCTION TO DEEP UNSUPERVISED LEARNING

*elements taken from D. Kirkby lectures at KSPA

GENERATIVE MODELLING AND DENSITY ESTIMATION

TYPES OF MACHINE LEARNING

SUPERVISED

LEARNS A MAPPING FROM X [FEATURES] TO Y [LABELS]

$$P(Y|X)$$

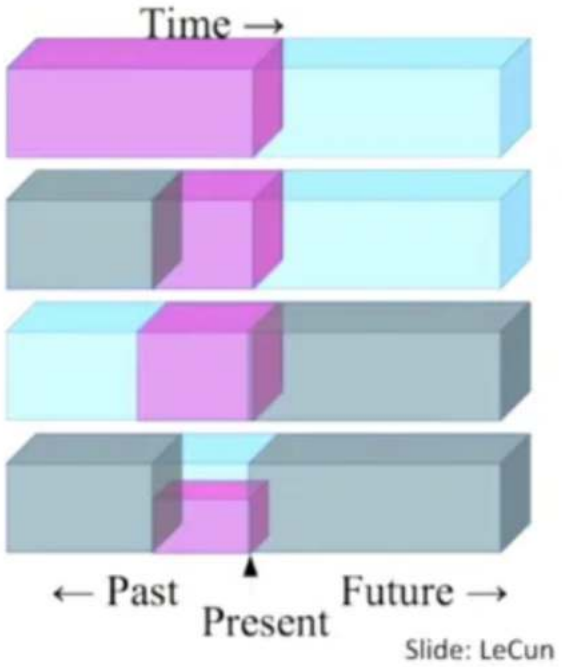
UNSUPERVISED AND SELF-SUPERVISED

NO LABELS - DISCOVER PATTERNS

$$P(X) \qquad P(\hat{Y}|X)$$

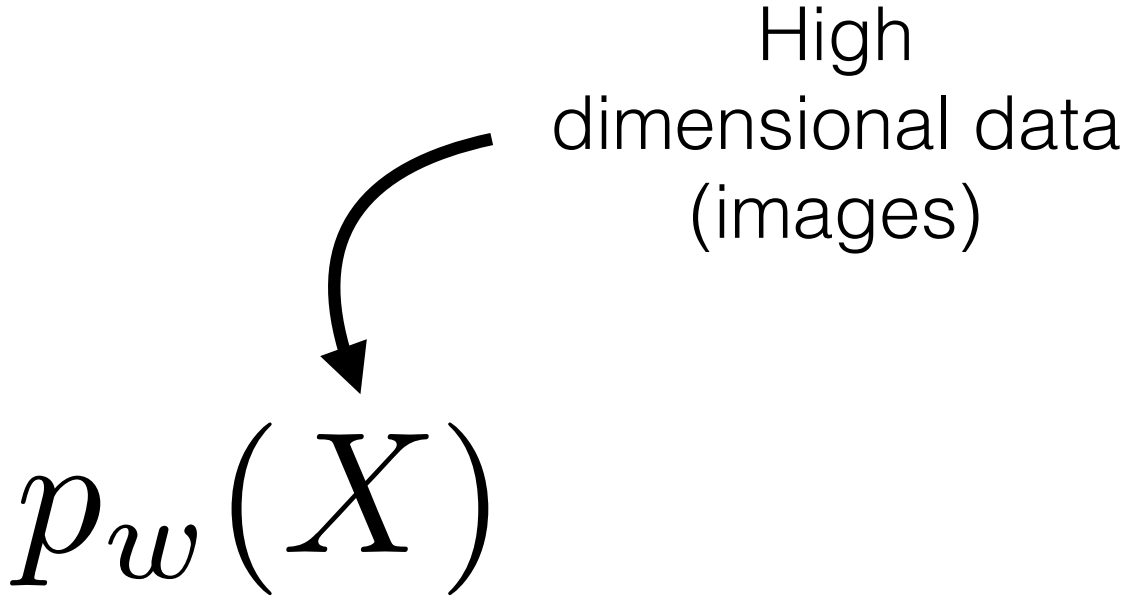
Self-Supervised Learning vs. Unsupervised

- Predict any part of the input from any other part.
- Predict the future from the past.
- Predict the future from the recent past.
- Predict the past from the present.
- Predict the top from the bottom.
- Predict the occluded from the visible
- Pretend there is a part of the input you don't know and predict that.

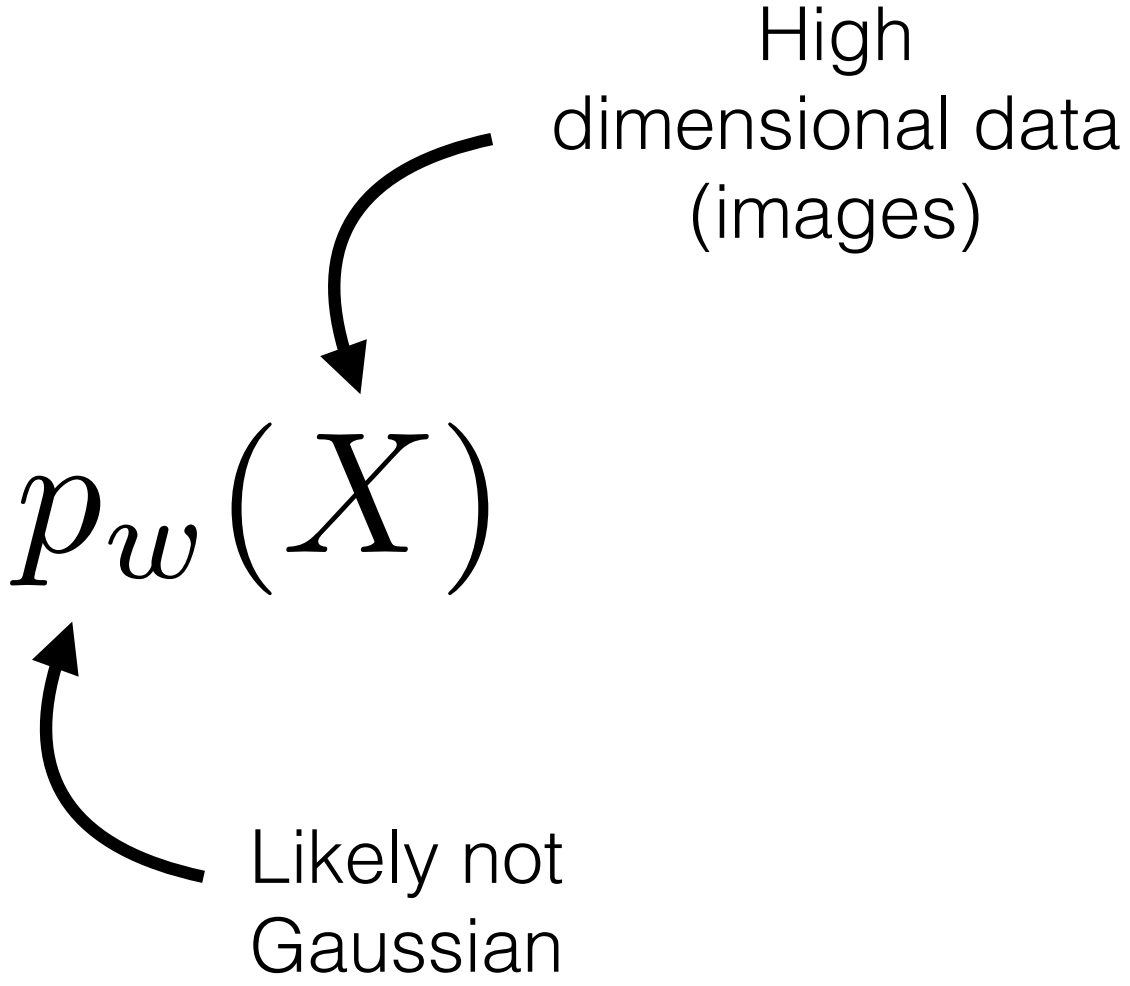


a self-supervised learning system creates a pretext task , which transforms the problem into a "supervised" problem: predict parts of its inputs based on the other parts of its inputs

Estimate a probability density function of an arbitrarily high dimensional data



Estimate a probability density function of an arbitrarily high dimensional data



Estimate a probability density function of an arbitrarily high dimensional data

High dimensional data (images)

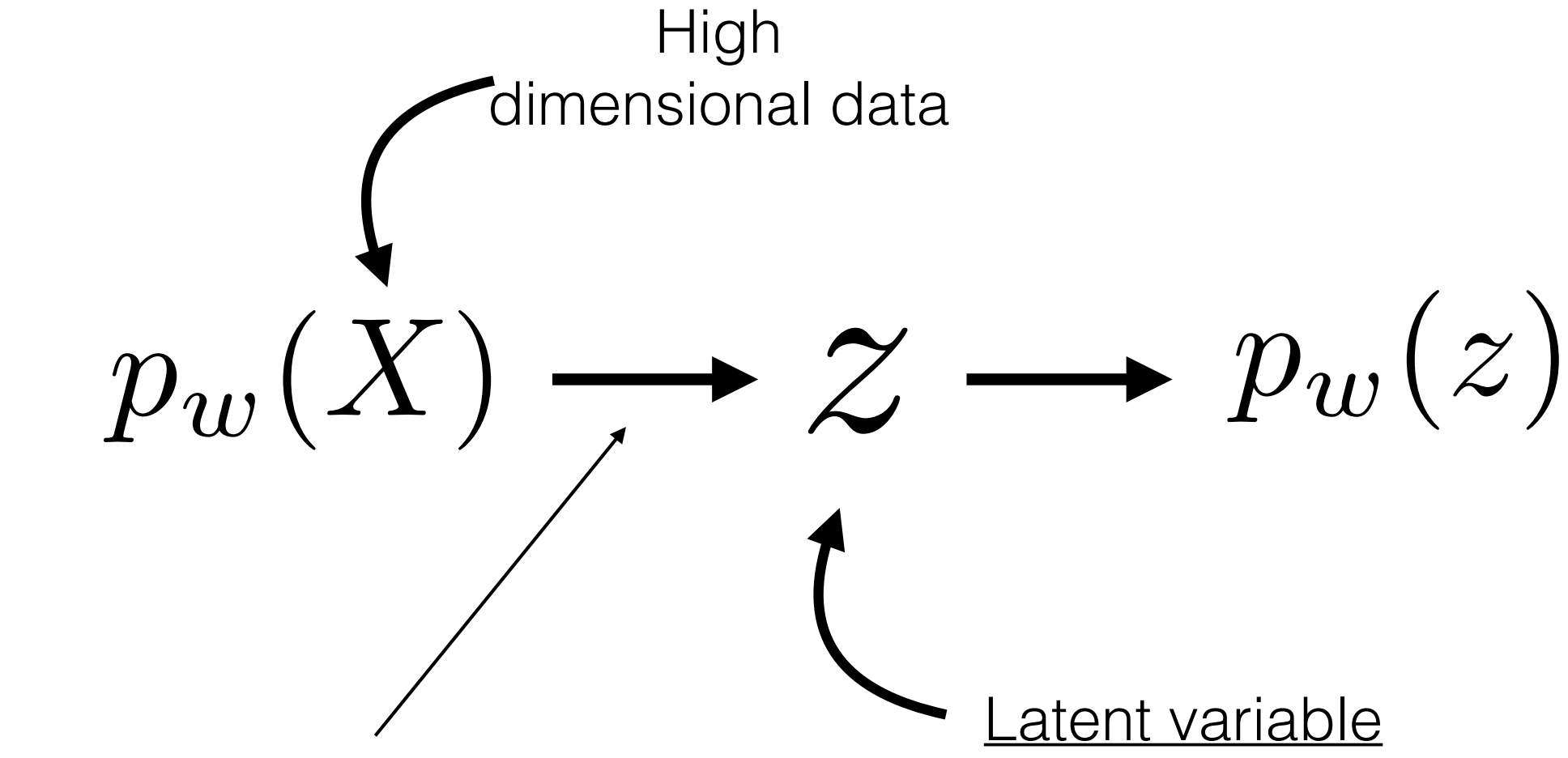
$$p_w(X)$$

Likely not Gaussian

Why?

1. Sampling. how can I interpolate / extrapolate a (small, sparse) dataset to generate new data sampled from the same (unknown) distribution?

2. Likelihood evaluation. what is the probability that a new observation is drawn from the same (unknown) distribution as some reference (small, sparse) dataset?


High
dimensional data
$p_w(X)$
z
$p_w(z)$
Latent variable
Representation learning /
dimensionality reduction

Types of Machine Learning

Approach	**Discriminative**		**Generative**	
Data type	Target Function	Method	Target Function	Method
Labels	$p(y\|x)$	All supervised networks with bottleneck	$p(x\|y)$	Conditional generative models
No Labels	$p(\hat{y}\|x)$	Self-supervised learning	$p(x)$	Generative models, Autoencoders

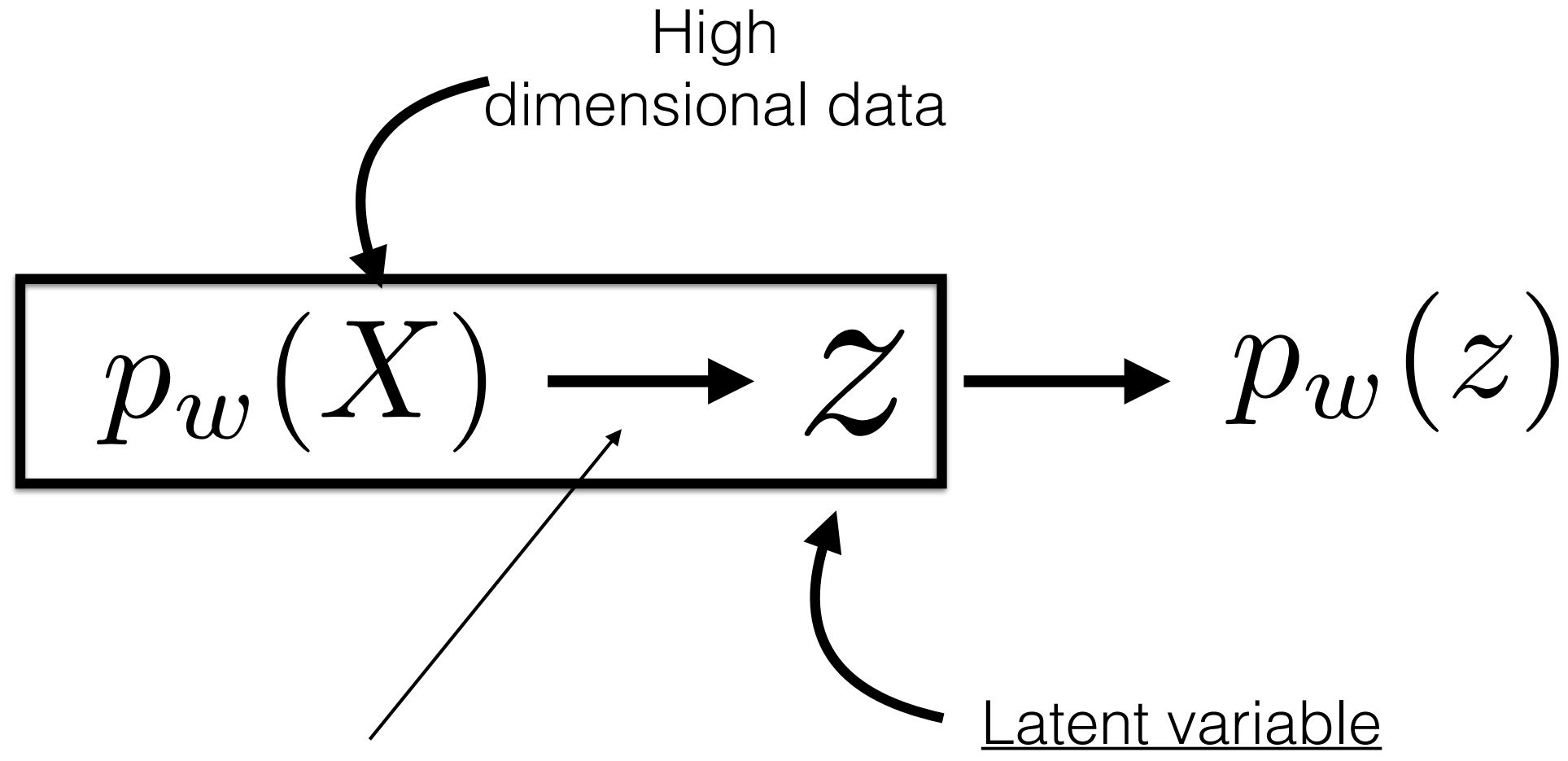
Two major applications:

- **SAMPLING:** HOW CAN I INTERPOLATE / EXTRAPOLATE A (SMALL, SPARSE) DATASET TO GENERATE NEW DATA SAMPLED FROM THE SAME (UNKNOWN) DISTRIBUTION?

- **LIKELIHOOD EVALUATION:** WHAT IS THE PROBABILITY THAT A NEW OBSERVATION IS DRAWN FROM THE SAME (UNKNOWN) DISTRIBUTION AS SOME REFERENCE (SMALL, SPARSE) DATASET?

1. Representation learning

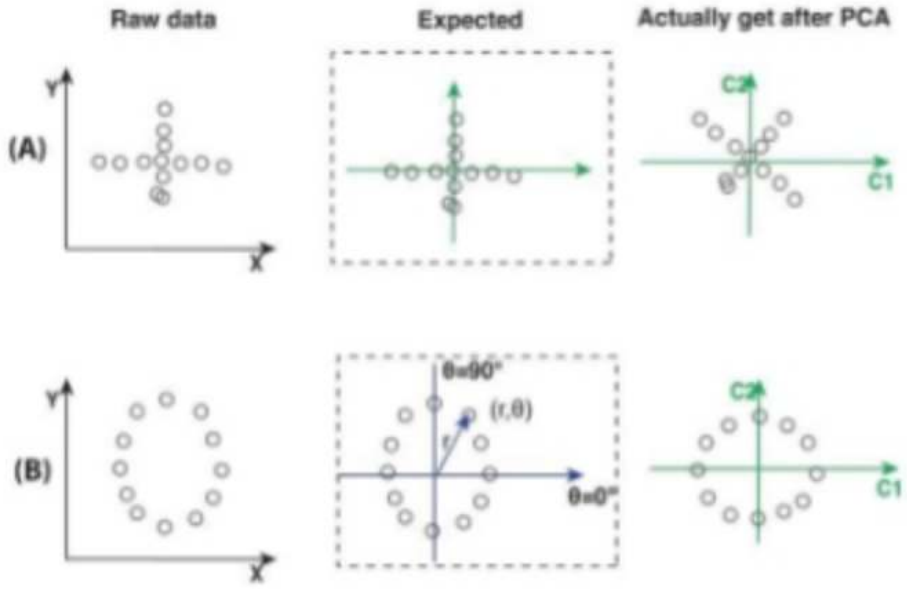
(See Dalya's lecture)


High
dimensional data
$p_w(X)$
z
$p_w(z)$
Latent variable
Representation learning /
dimensionality reduction

LIMITATIONS OF PCA

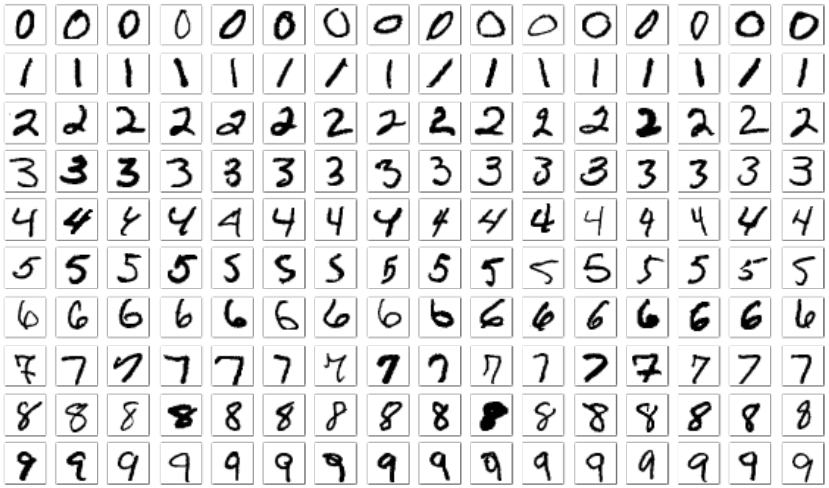
PCA APPLY LINEAR TRANSFORMATIONS

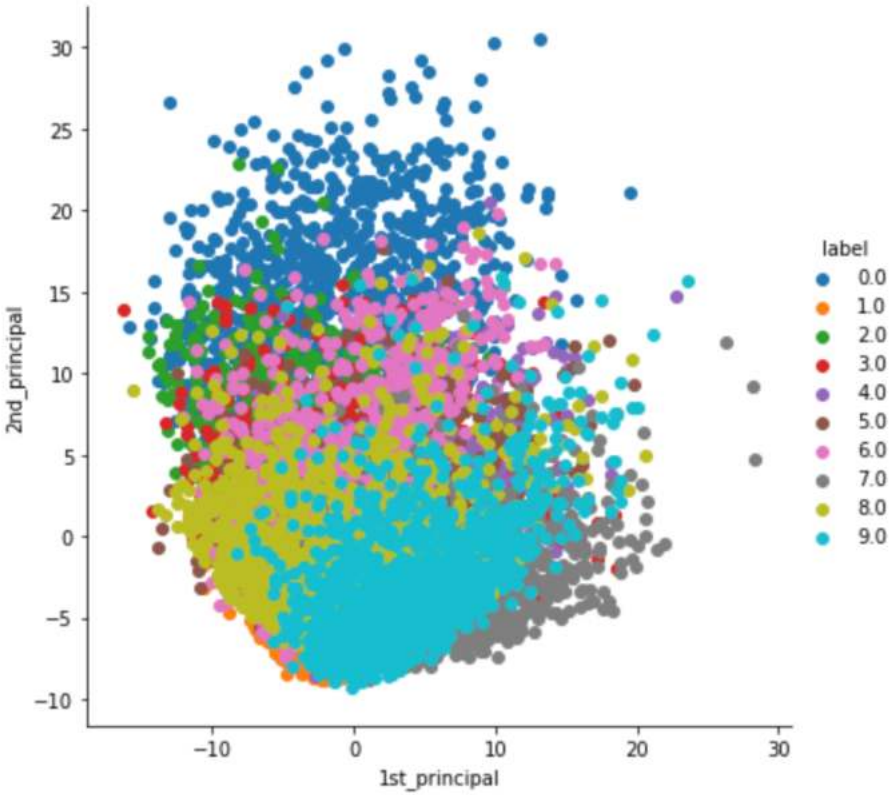
SINCE WE USE THE COVARIANCE MATRIX, IT ASSUMES THAT THE DATA FOLLOWS A **MULTIDIMENSIONAL NORMAL DISTRIBUTION**



LIMITATIONS OF PCA

AND DATA IS NOT ALWAYS GAUSSIAN….

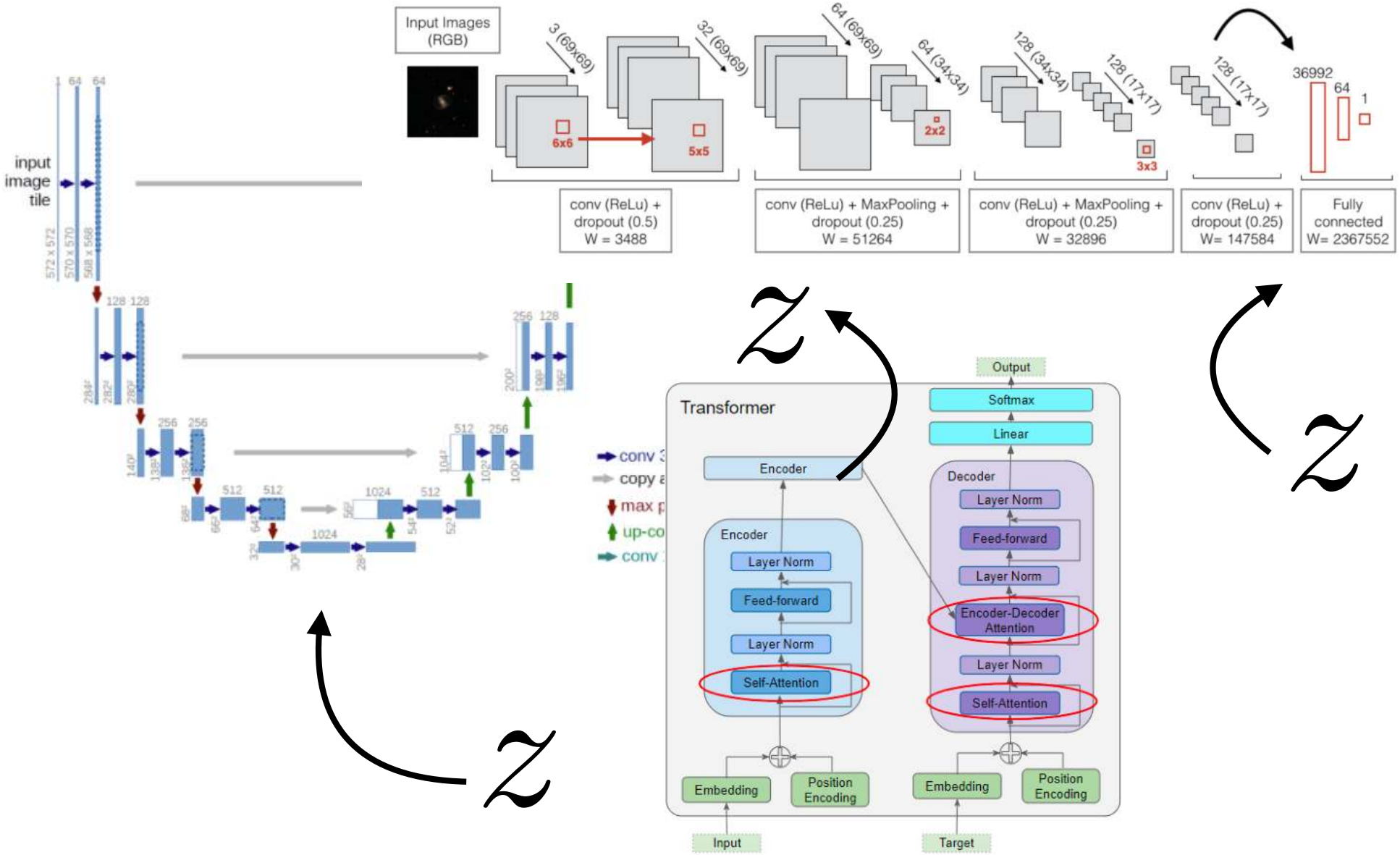




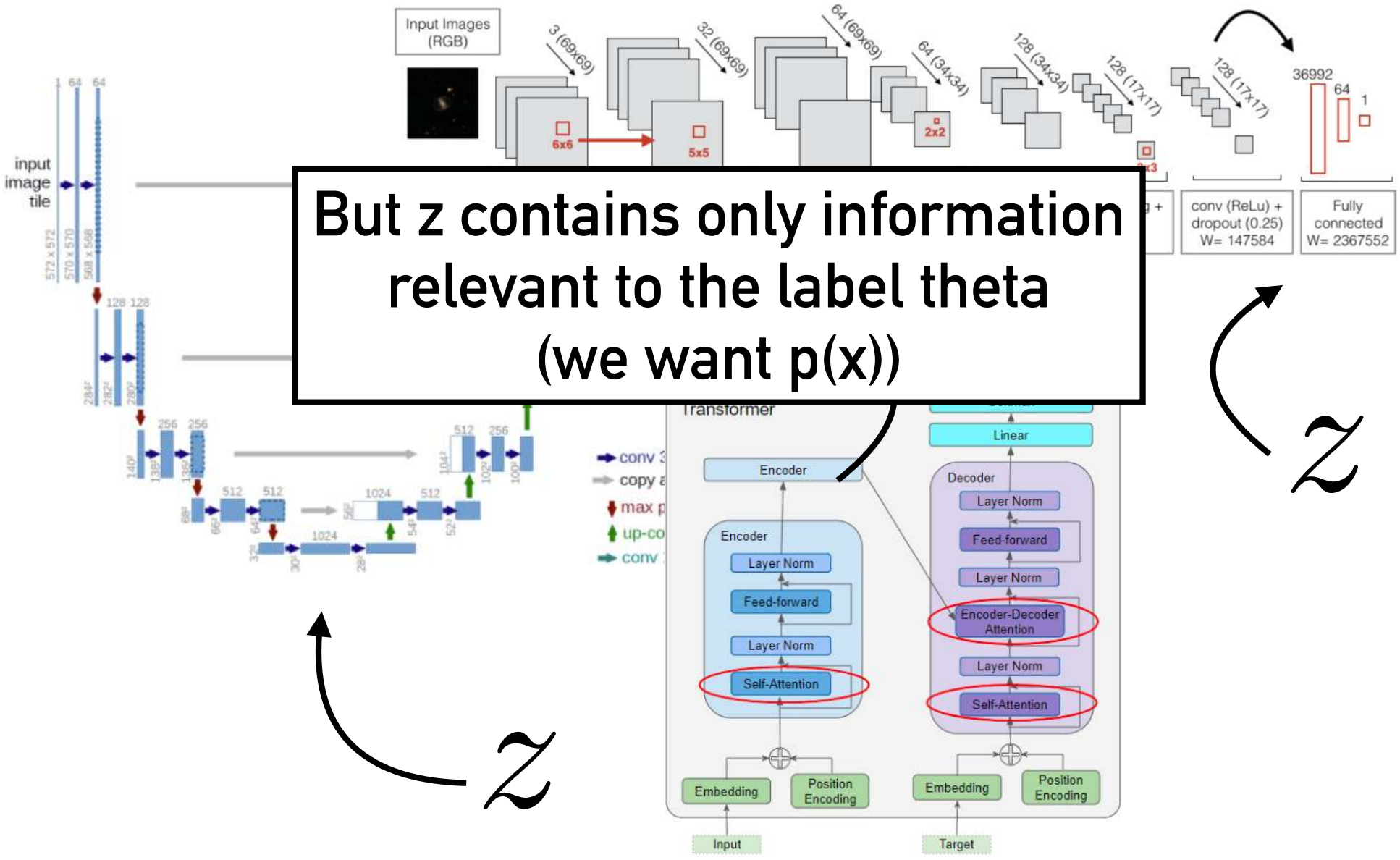
https://medium.com/analytics-vidhya/principal-component-analysis-pca-with-code-on-mnist-dataset-da7de0d07c22

CAN WE GENERALIZE THAT?

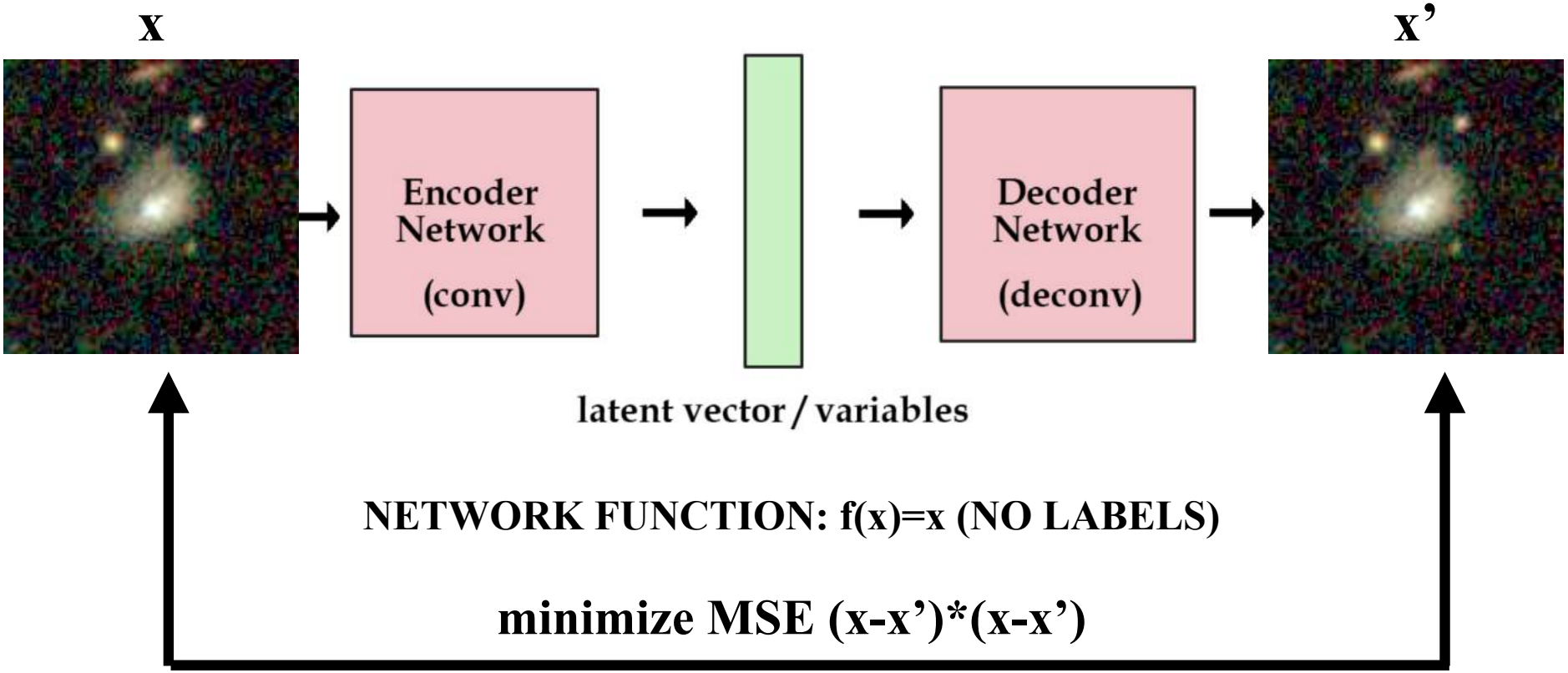
Any supervised NN obtains a non linear representation “z”



Any supervised NN obtains a non linear representation “z”

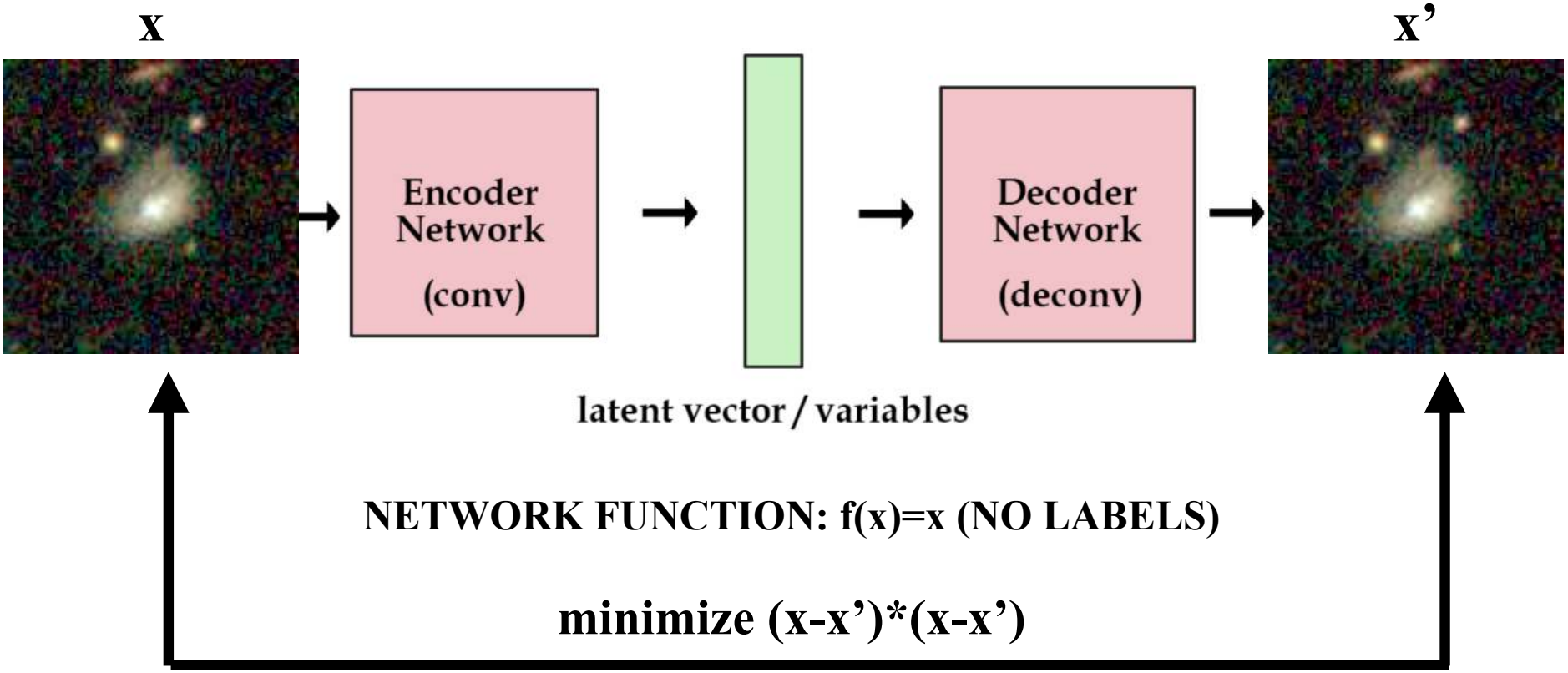


(CONVOLUTIONAL) AUTO-ENCODER



AN AUTO-ENCODER IS SIMPLY ANY NETWORK WITH IDENTICAL INPUT AND OUTPUT

CONVOLUTIONAL AUTO-ENCODER

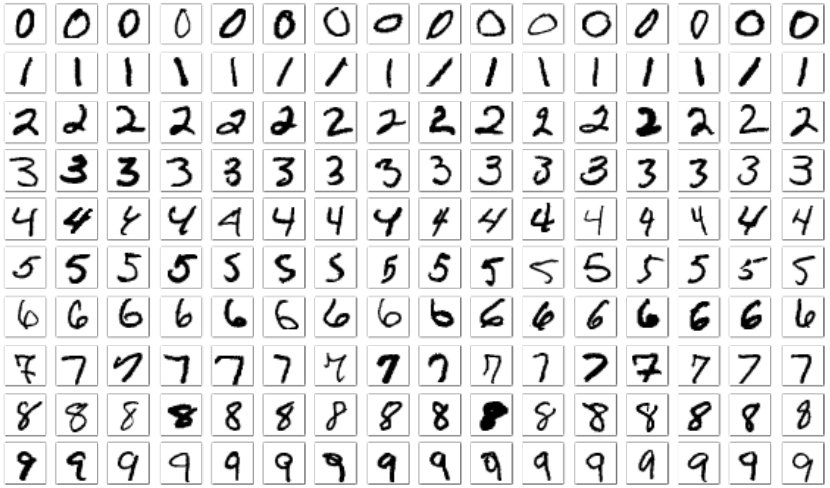


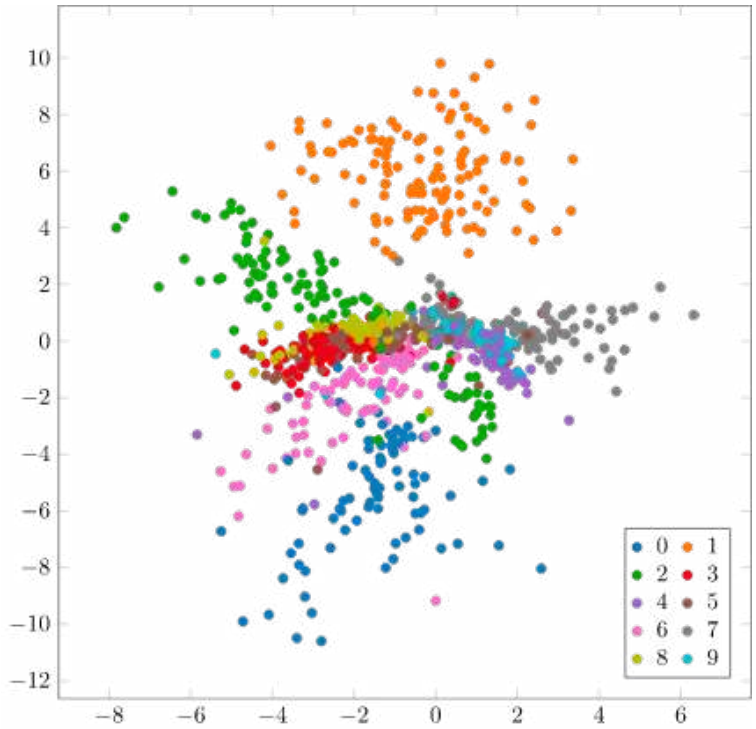
BY REDUCING THE DIMENSIONALITY IN THE LATENT SPACE WE FORCE THE NETWORK TO LEARN A REPRESENTATION OF THE INPUT DATA IN A LOWER DIMENSIONALITY SPACE

* NO NEED TO BE CONVOLUTIONAL - ANY NEURAL NETWORK WITH A BOTTLENECK WILL DO THE JOB

* **QUESTION**: WHAT WOULD HAPPEN IF WE SET AN AUTOENCODER WITH NO ACTIVATION FUNCTIONS?

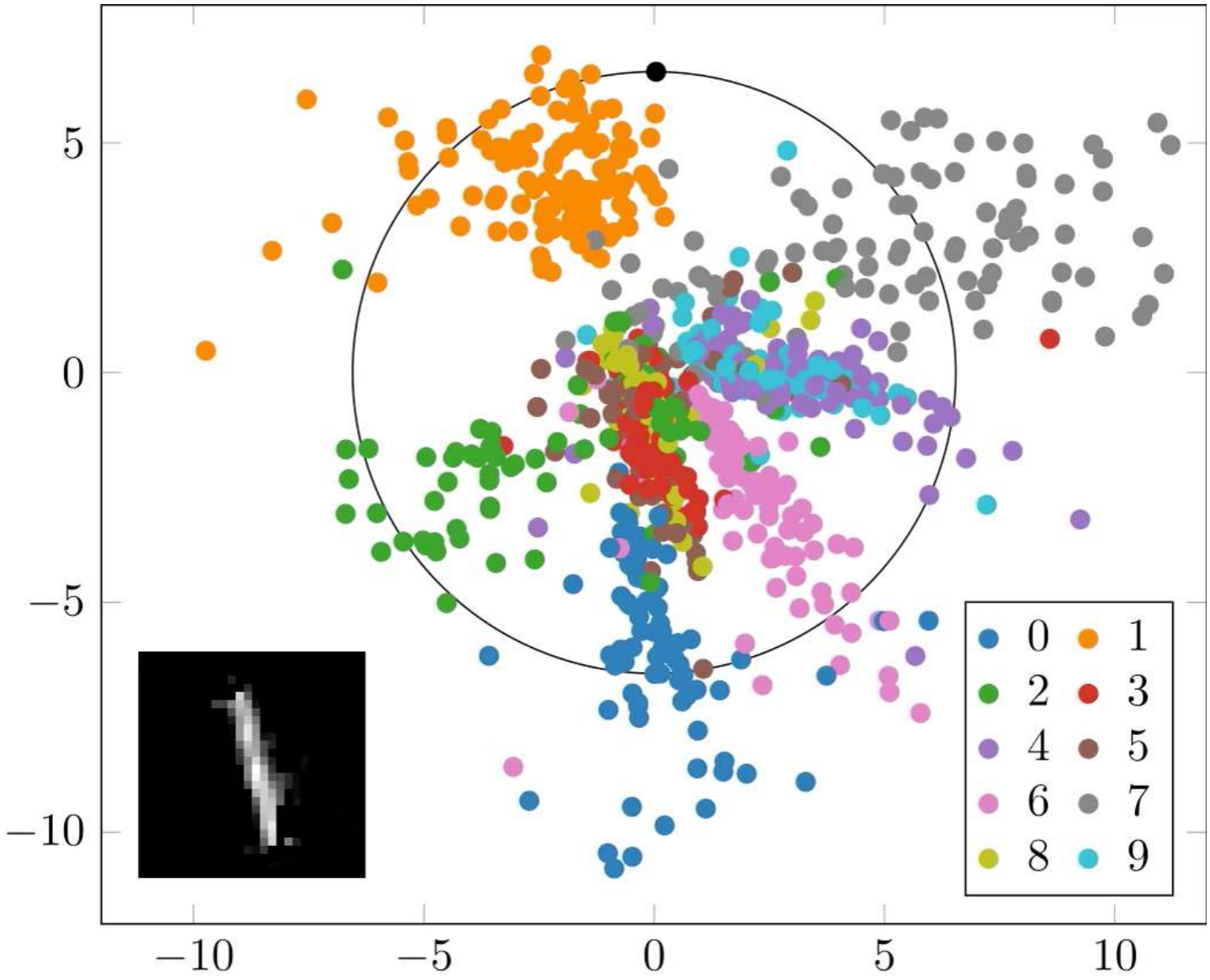
AUTOENCODER REPRESENTATION OF MNIST



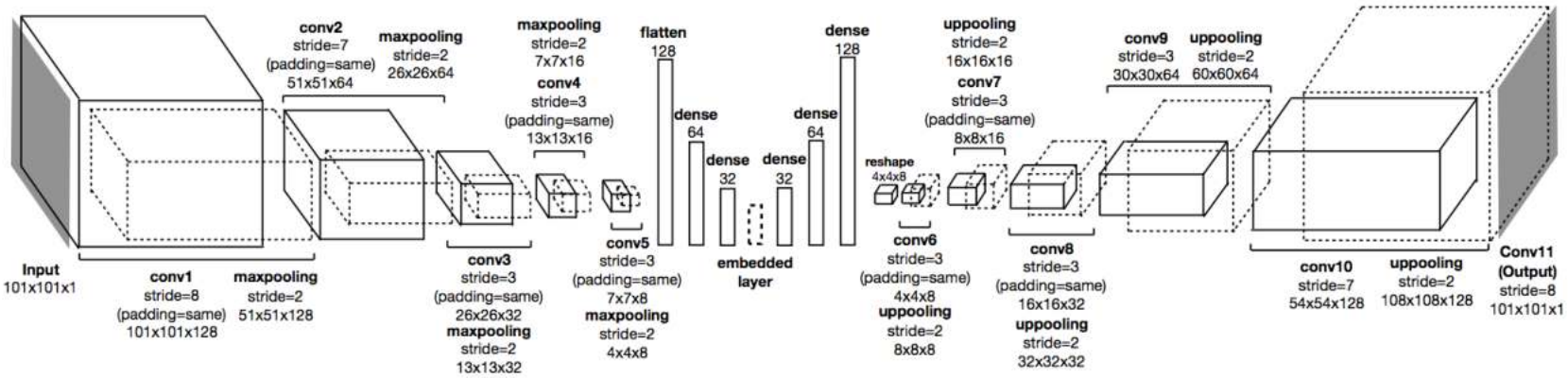


https://gertjanvandenburg.com/blog/autoencoder/

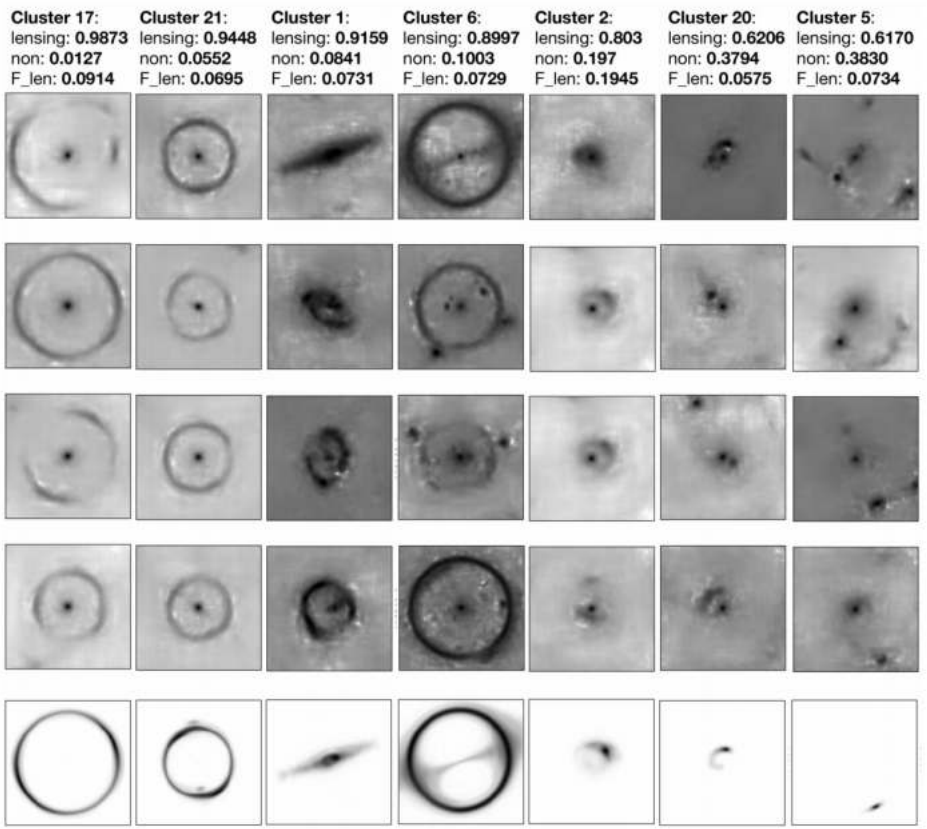
AUTOENCODER REPRESENTATION OF MNIST

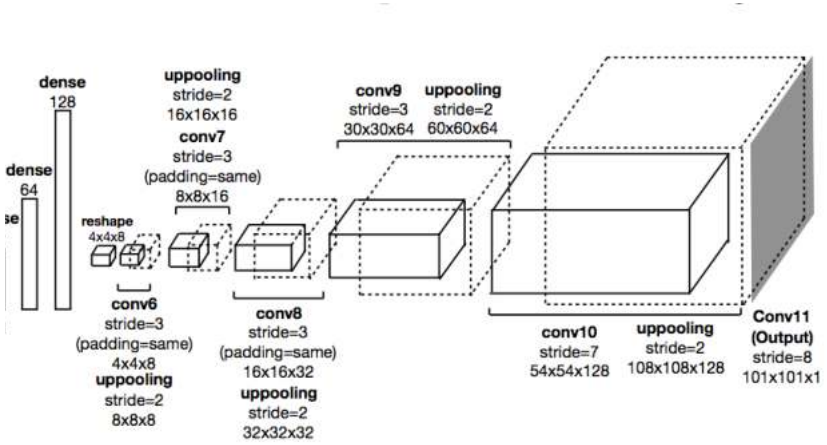


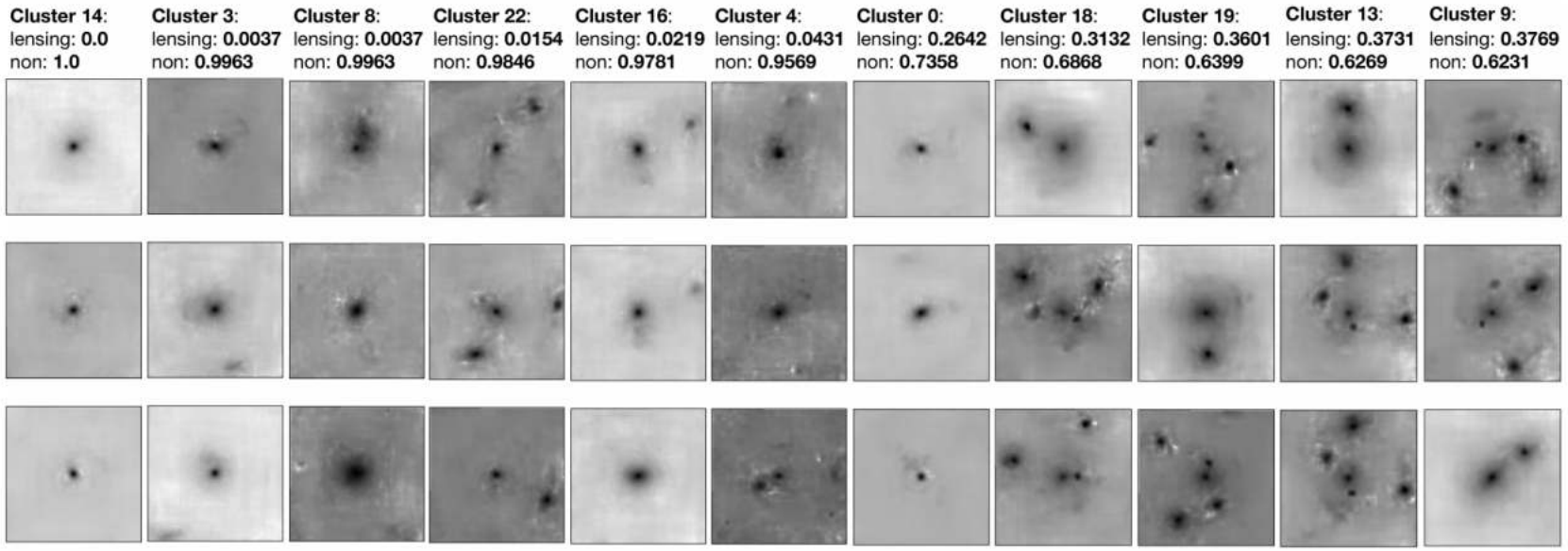
https://gertjanvandenburg.com/blog/autoencoder/


Input
101x101x1
conv1
stride=8
(padding=same)
101x101x128
maxpooling
stride=2
51x51x128
conv2
stride=7
(padding=same)
51x51x64
maxpooling
stride=2
26x26x64
conv3
stride=3
(padding=same)
26x26x32
maxpooling
stride=2
13x13x32
maxpooling
stride=2
7x7x16
conv4
stride=3
(padding=same)
13x13x16
conv5
stride=3
(padding=same)
7x7x8
maxpooling
stride=2
4x4x8
flatten
128
dense
64
dense
32
embedded
layer
dense
32
dense
64
dense
128
reshape
4x4x8
conv6
stride=3
(padding=same)
4x4x8
uppooling
stride=2
8x8x8
uppooling
stride=2
16x16x16
conv7
stride=3
(padding=same)
8x8x16
conv8
stride=3
(padding=same)
16x16x32
uppooling
stride=2
32x32x32
conv9
stride=3
30x30x64
uppooling
stride=2
60x60x64
conv10
stride=7
54x54x128
uppooling
stride=2
108x108x128
Conv11
(Output)
stride=8
101x101x1

Cheng+20


Cluster 17: lensing: 0.9873 non: 0.0127 F_len: 0.0914
Cluster 21: lensing: 0.9448 non: 0.0552 F_len: 0.0695
Cluster 1: lensing: 0.9159 non: 0.0841 F_len: 0.0731
Cluster 6: lensing: 0.8997 non: 0.1003 F_len: 0.0729
Cluster 2: lensing: 0.803 non: 0.197 F_len: 0.1945
Cluster 20: lensing: 0.6206 non: 0.3794 F_len: 0.0575
Cluster 5: lensing: 0.6170 non: 0.3830 F_len: 0.0734


dense
128
dense
64
reshape
4x4x8
conv6
stride=3
(padding=same)
4x4x8
uppooling
stride=2
8x8x8
uppooling
stride=2
16x16x16
conv7
stride=3
(padding=same)
8x8x16
conv8
stride=3
(padding=same)
16x16x32
uppooling
stride=2
32x32x32
conv9
stride=3
30x30x64
uppooling
stride=2
60x60x64
conv10
stride=7
54x54x128
uppooling
stride=2
108x108x128
Conv11
(Output)
stride=8
101x101x1


Cluster 14: lensing: 0.0 non: 1.0
Cluster 3: lensing: 0.0037 non: 0.9963
Cluster 8: lensing: 0.0037 non: 0.9963
Cluster 22: lensing: 0.0154 non: 0.9846
Cluster 16: lensing: 0.0219 non: 0.9781
Cluster 4: lensing: 0.0431 non: 0.9569
Cluster 0: lensing: 0.2642 non: 0.7358
Cluster 18: lensing: 0.3132 non: 0.6868
Cluster 19: lensing: 0.3601 non: 0.6399
Cluster 13: lensing: 0.3731 non: 0.6269
Cluster 9: lensing: 0.3769 non: 0.6231

Cheng+20

SELF-SUPERVISED LEARNING

Another alternative is to invent a target to obtain z

$$p(\hat{y}|X)$$

(Still using a discriminative model, with a general target, and use the latent z as a representation of the data)

SELF-SUPERVISED LEARNING

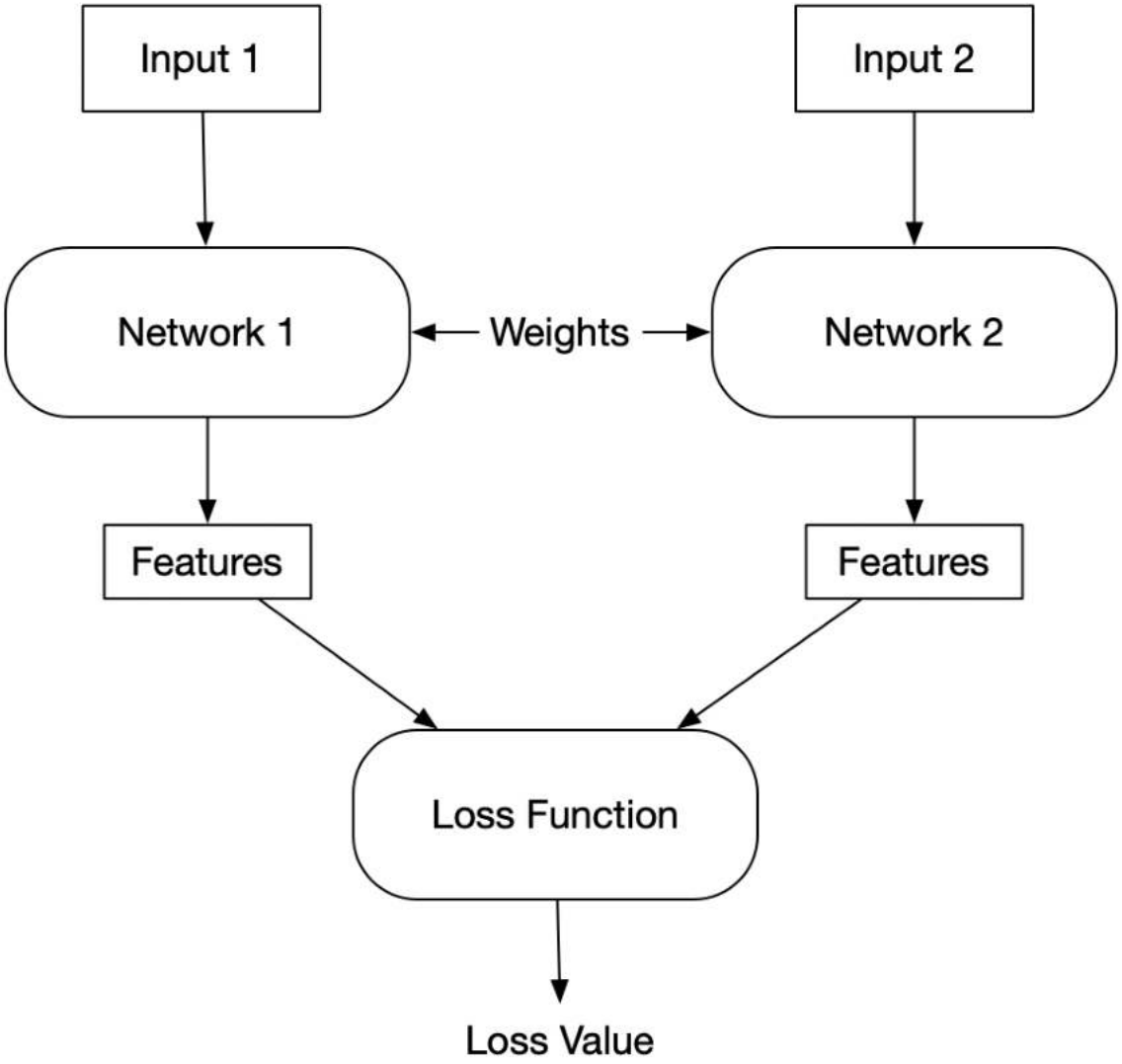
Another alternative is to invent a target to obtain z
(Pretext tasks)

$$p(\hat{y}|X)$$

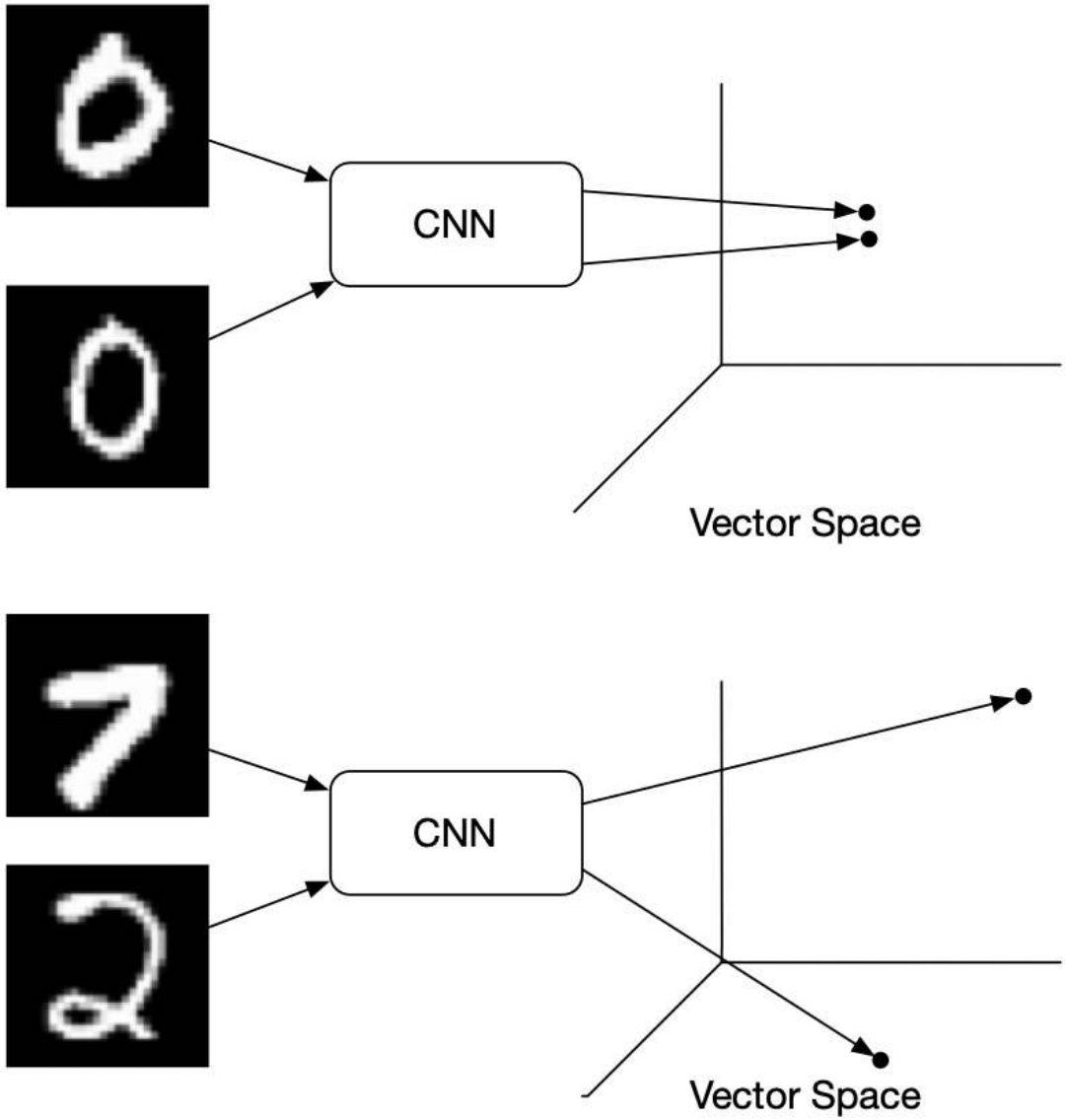
Rotation, zoom etc..

(Still using a discriminative model, with a general target, and use the latent z as a representation of the data)

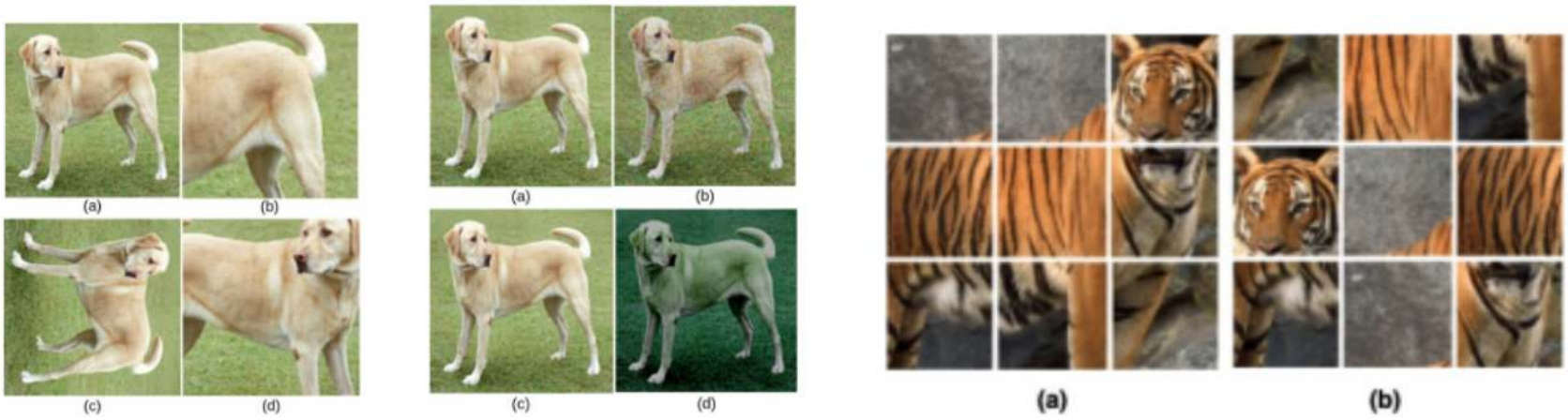
Siamese networks are two networks sharing weights and common loss function



In a supervised setting, similar objects are used to be together by simply using a crossentropy loss



e.g. contrastive learning uses the similarity between augmented version as a pretext task



Color Augmentation

Image Rotation

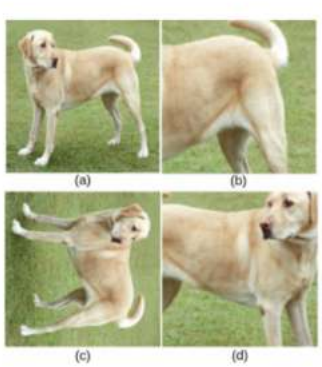
Image Cropping

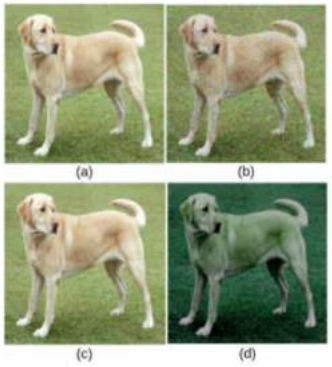
Any geometrical transformation …

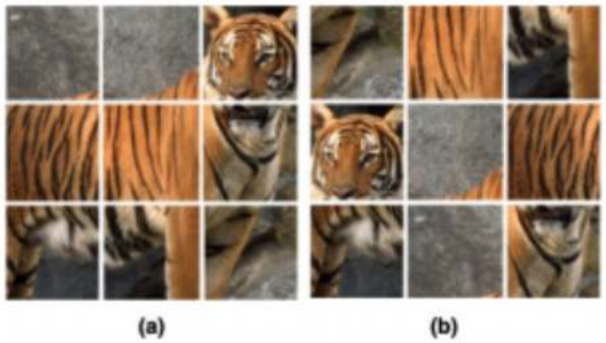
In most cases, labels are not available.

How do you recognise similar objects?

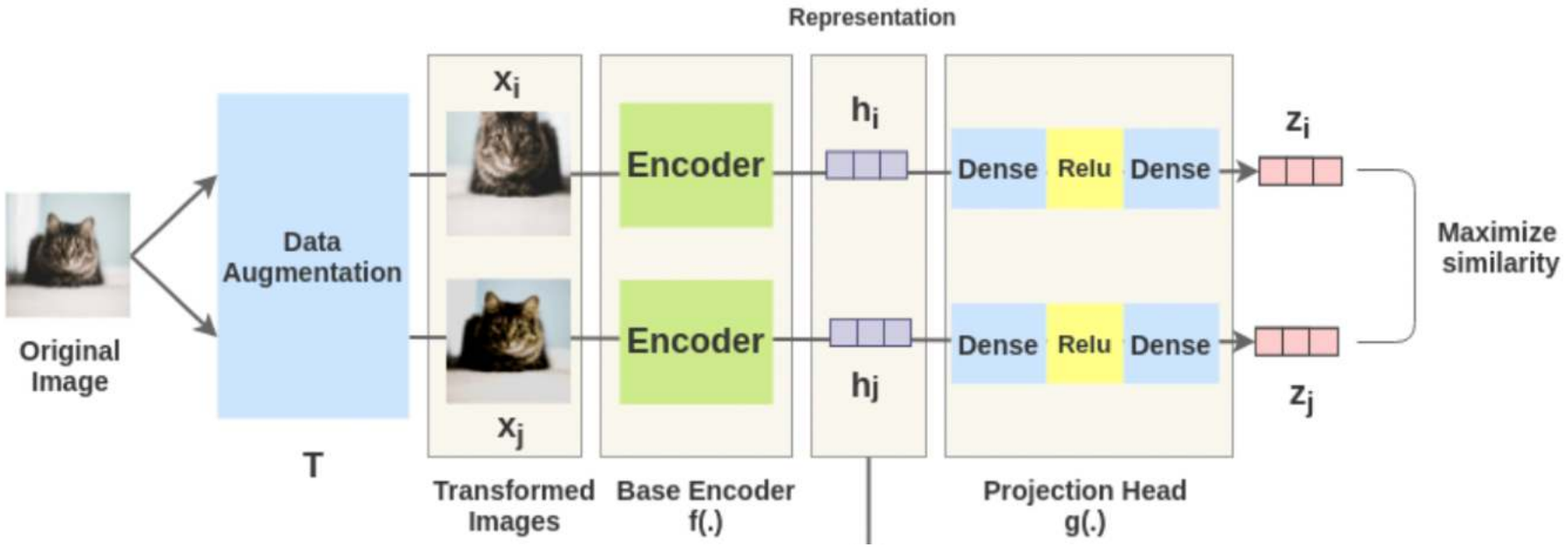
What would be the loss function?



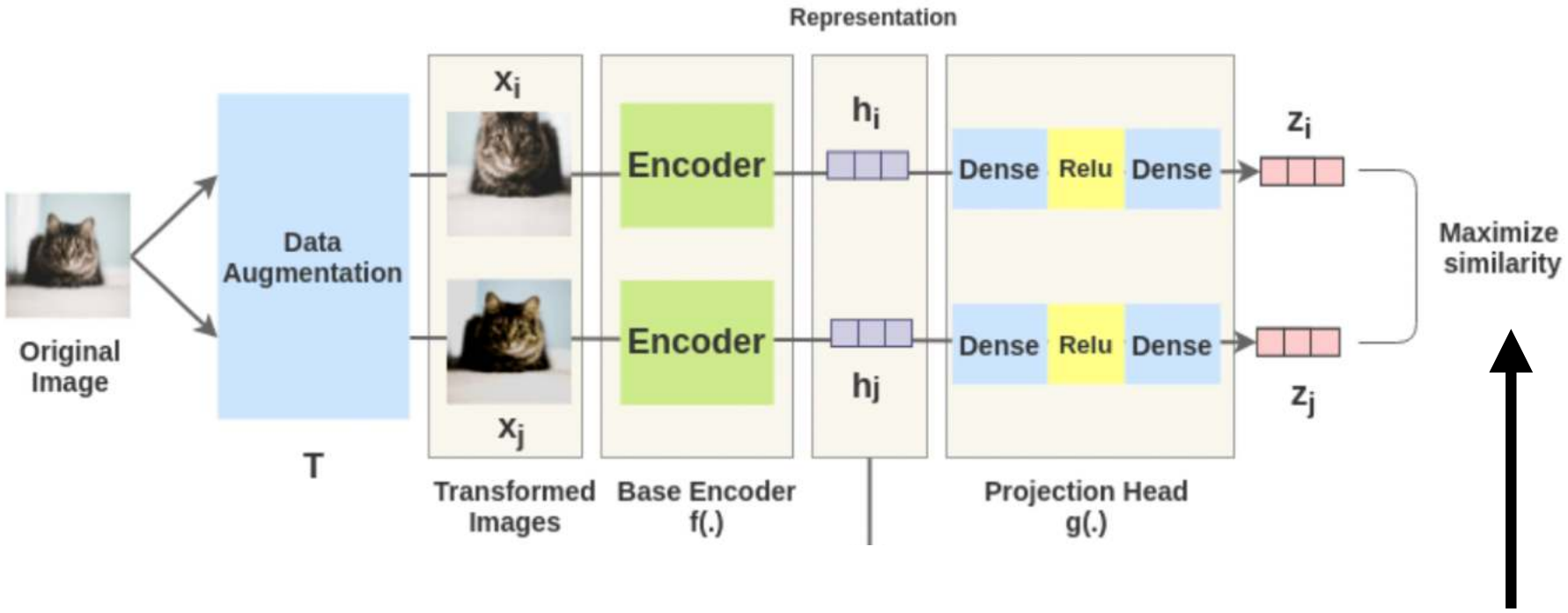




The augmented versions of the images are passed through siamese networks and projected into a latent variable z



The augmented versions of the images are passed through siamese networks and projected into a latent variable z



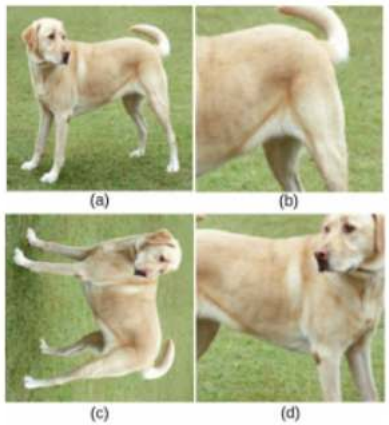
The key is the loss function

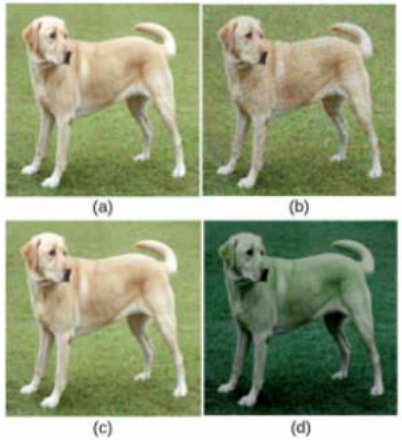
In most cases, labels are not available.

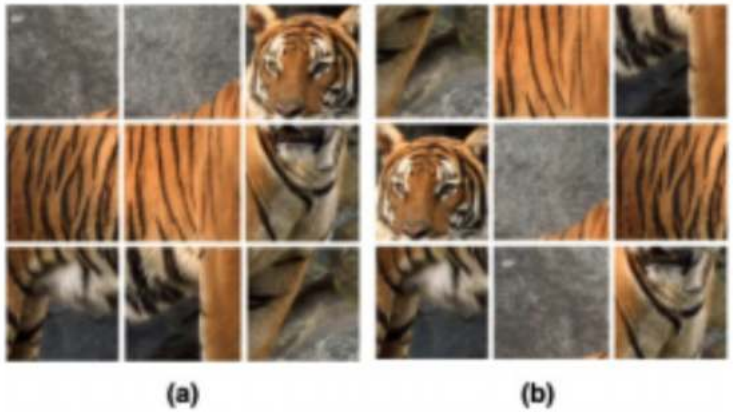
How do you recognise similar objects?

What would be the loss function?

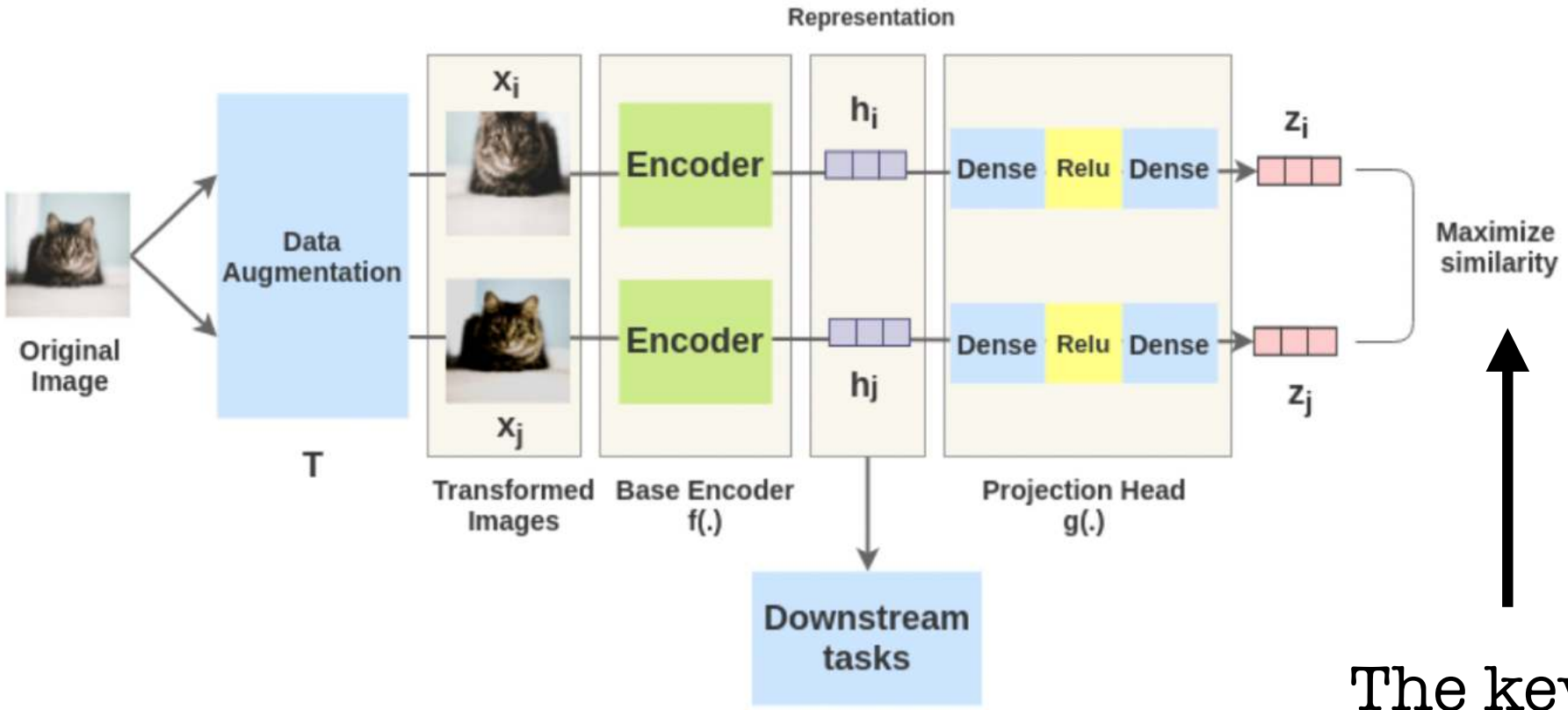
We create“pretext tasks” from a unique image:







The augmented versions of the images are passed through siamese networks and projected into a latent variable z



The key is the loss function

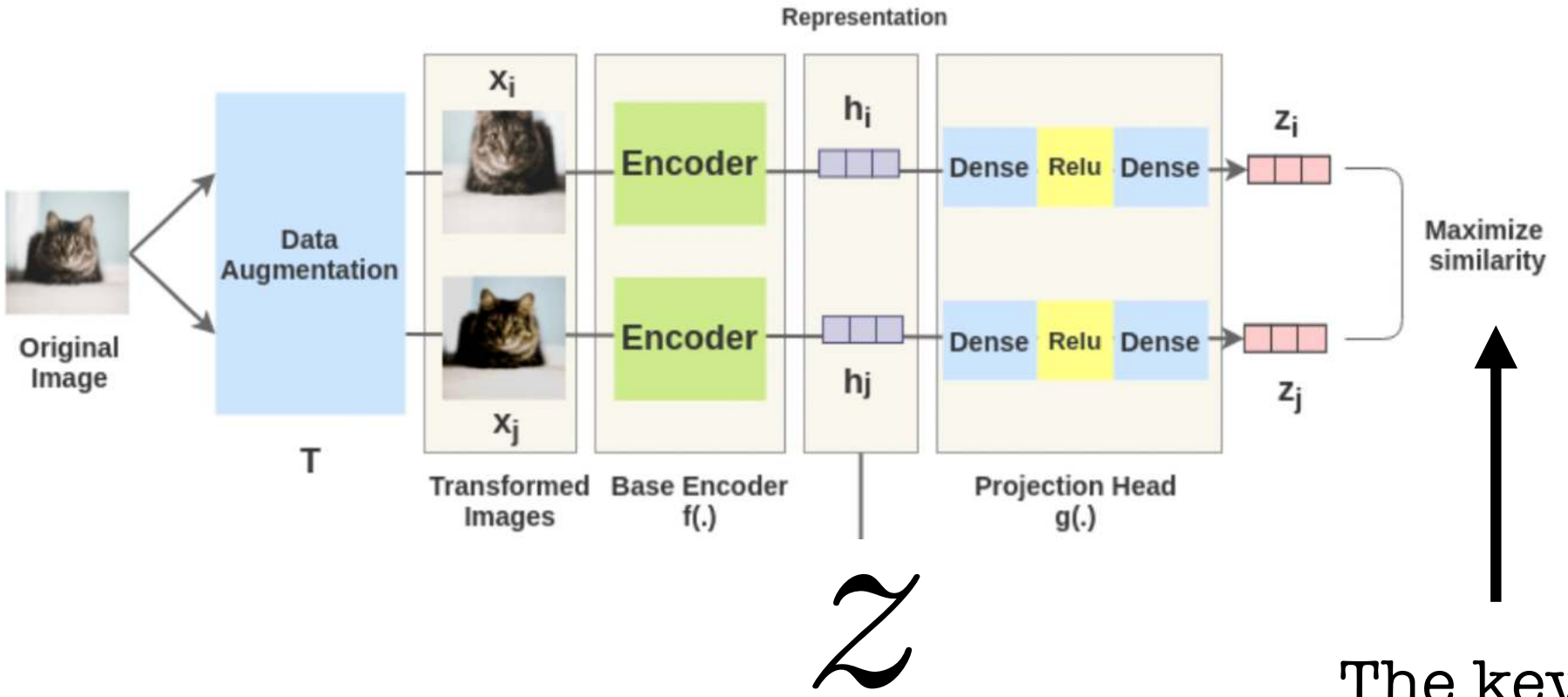
The contrastive loss:

Similarly between
two representations of positive pairs

$$l_{i,j} = -\log \frac{\exp(\langle z_i, z_j \rangle / h)}{\sum_{k=1,k\neq i}^{2N} \exp(\langle z_i, z_k \rangle / h)},$$

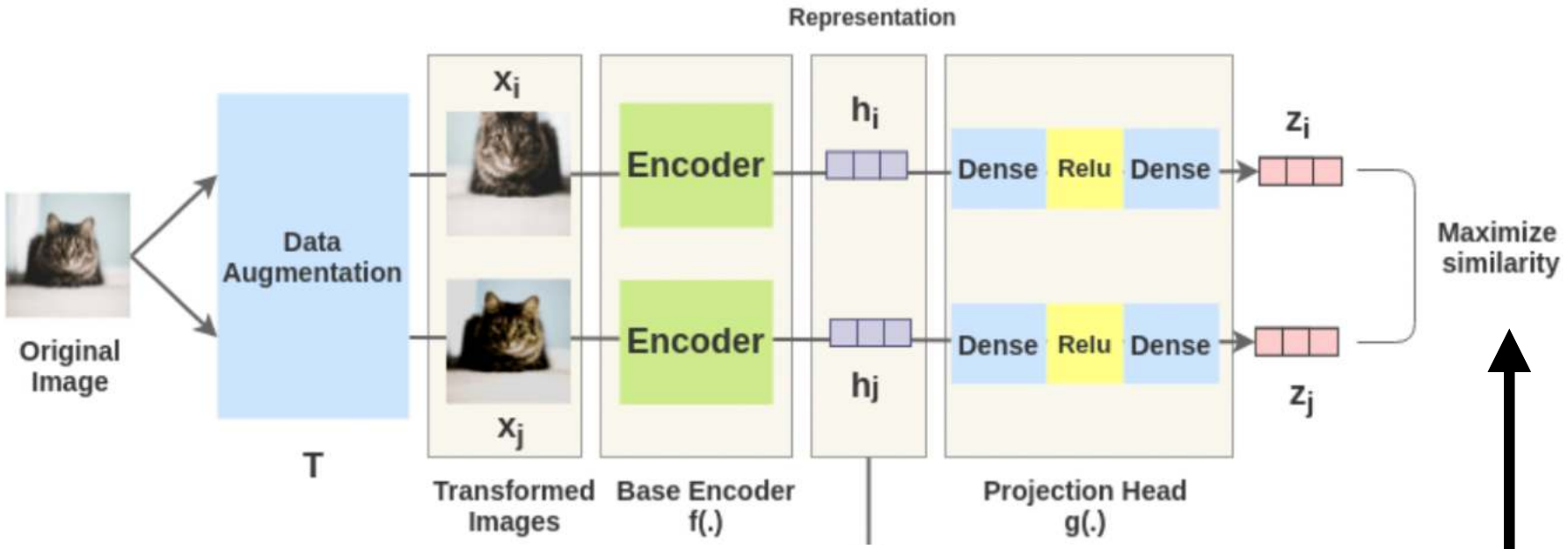
Sum of all similarities between
negative pairs

The augmented versions of the images are passed through siamese networks and projected into a latent variable z



The key is the loss function

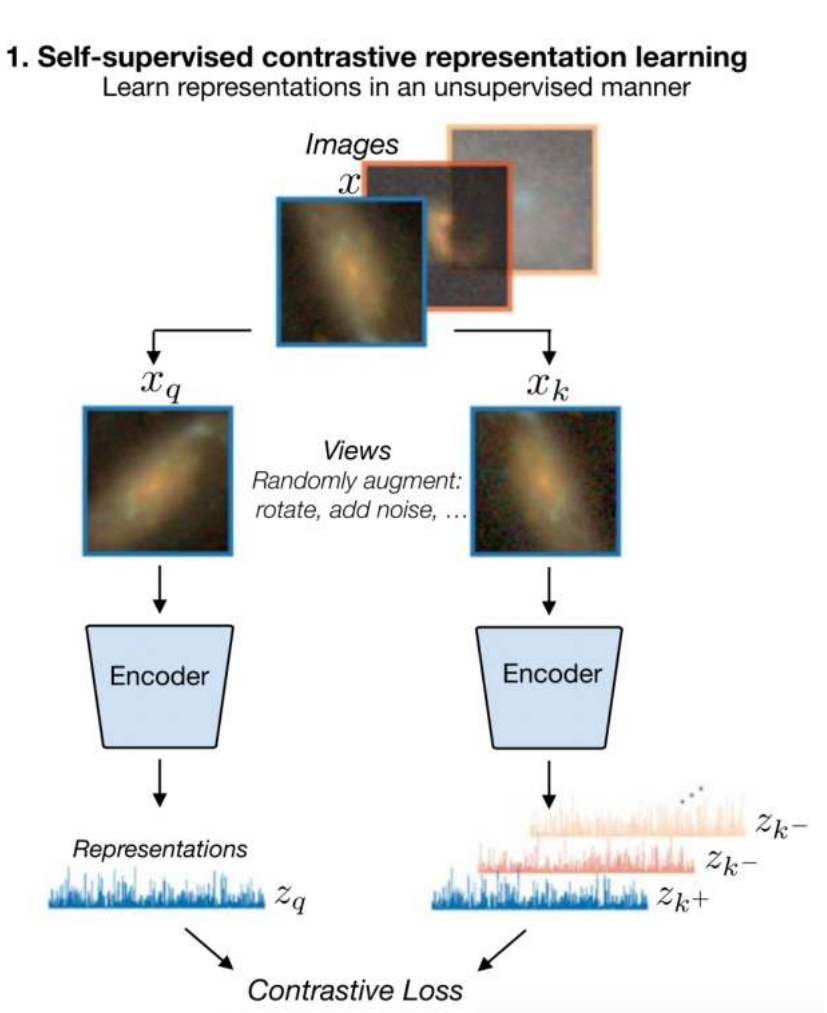
The augmented versions of the images are passed through siamese networks and projected into a latent variable z

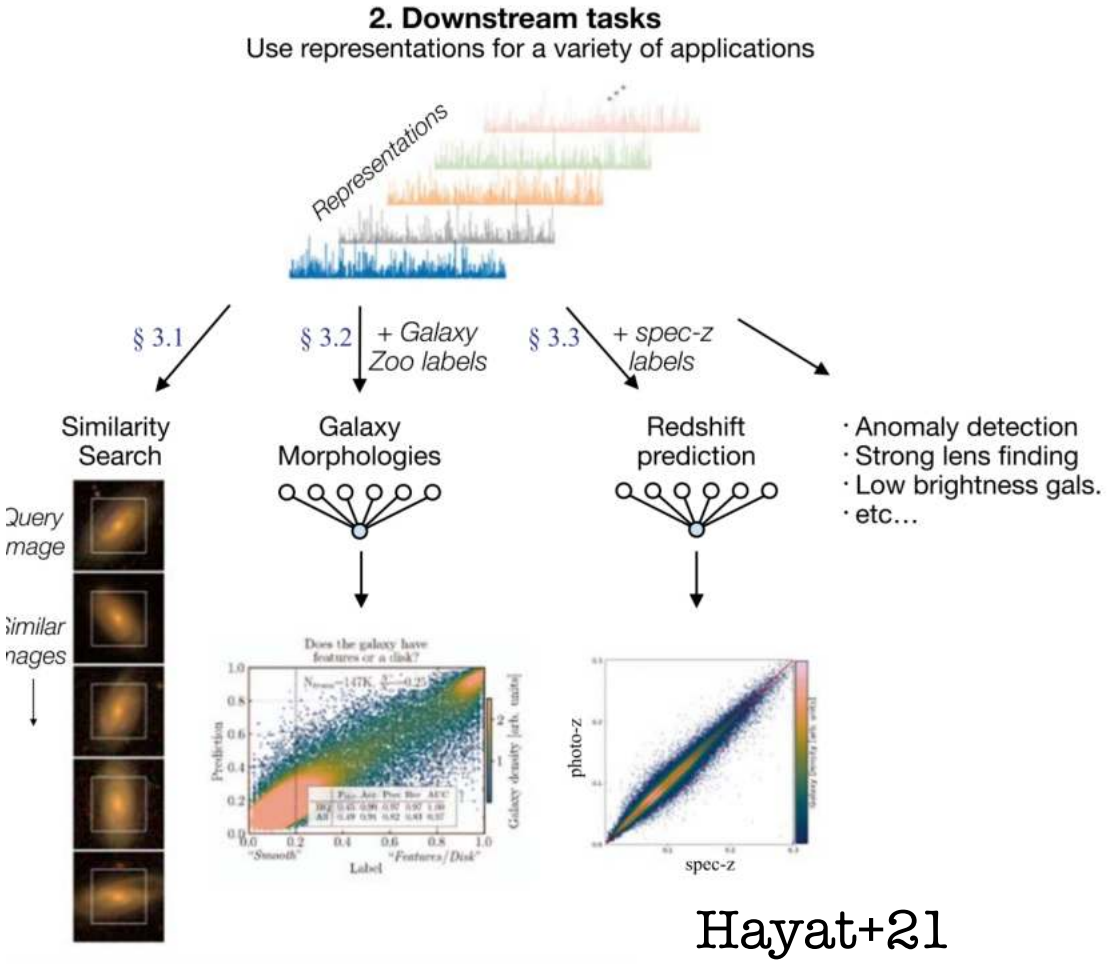


z

Downstream tasks

The key is the loss function





Hayat+21

It turns out that the representations learned are very general and can be used for a variety of “downstream tasks”

(Foundation Models)