ML & Particle Physics

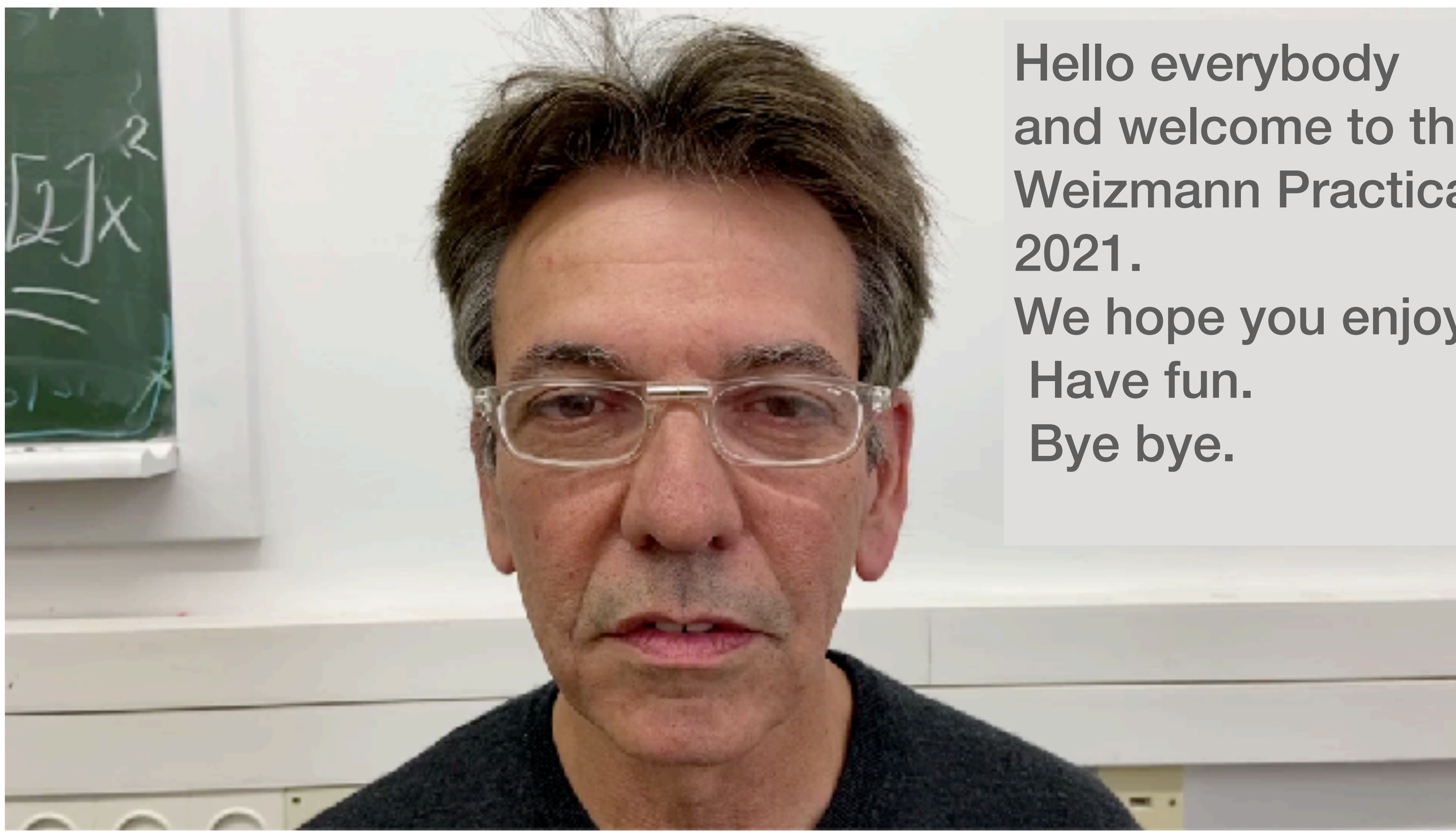
Eilam Gross

ASP 2021

Outline

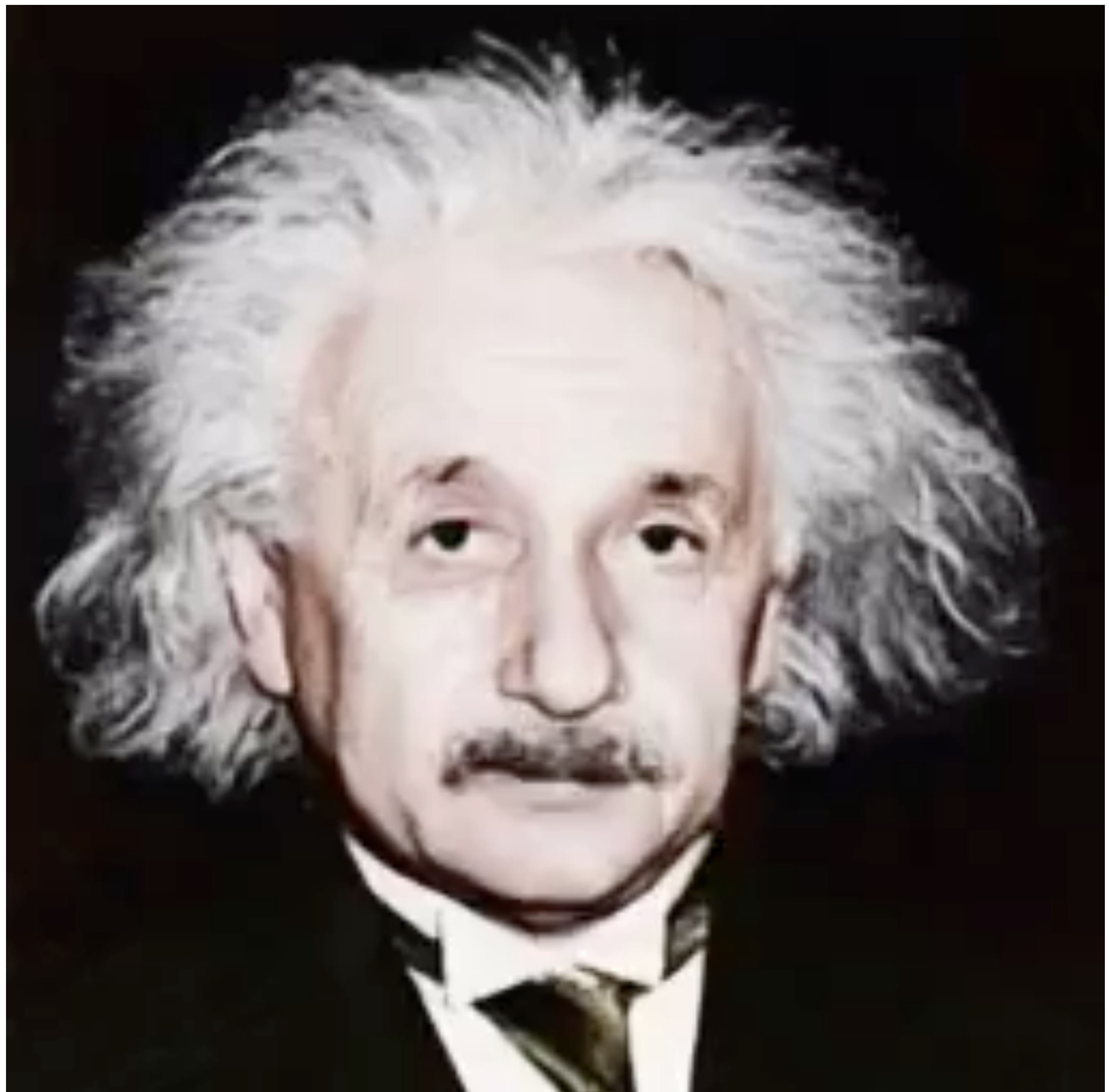
- What is a Neural Net
- Supervised Learning
- What is CNN
- Unsupervised Learning and Auto Encoder
  The role of Hidden Layers
- Graph Neural Net
- Some Applications

Practical Deep Learning 2021

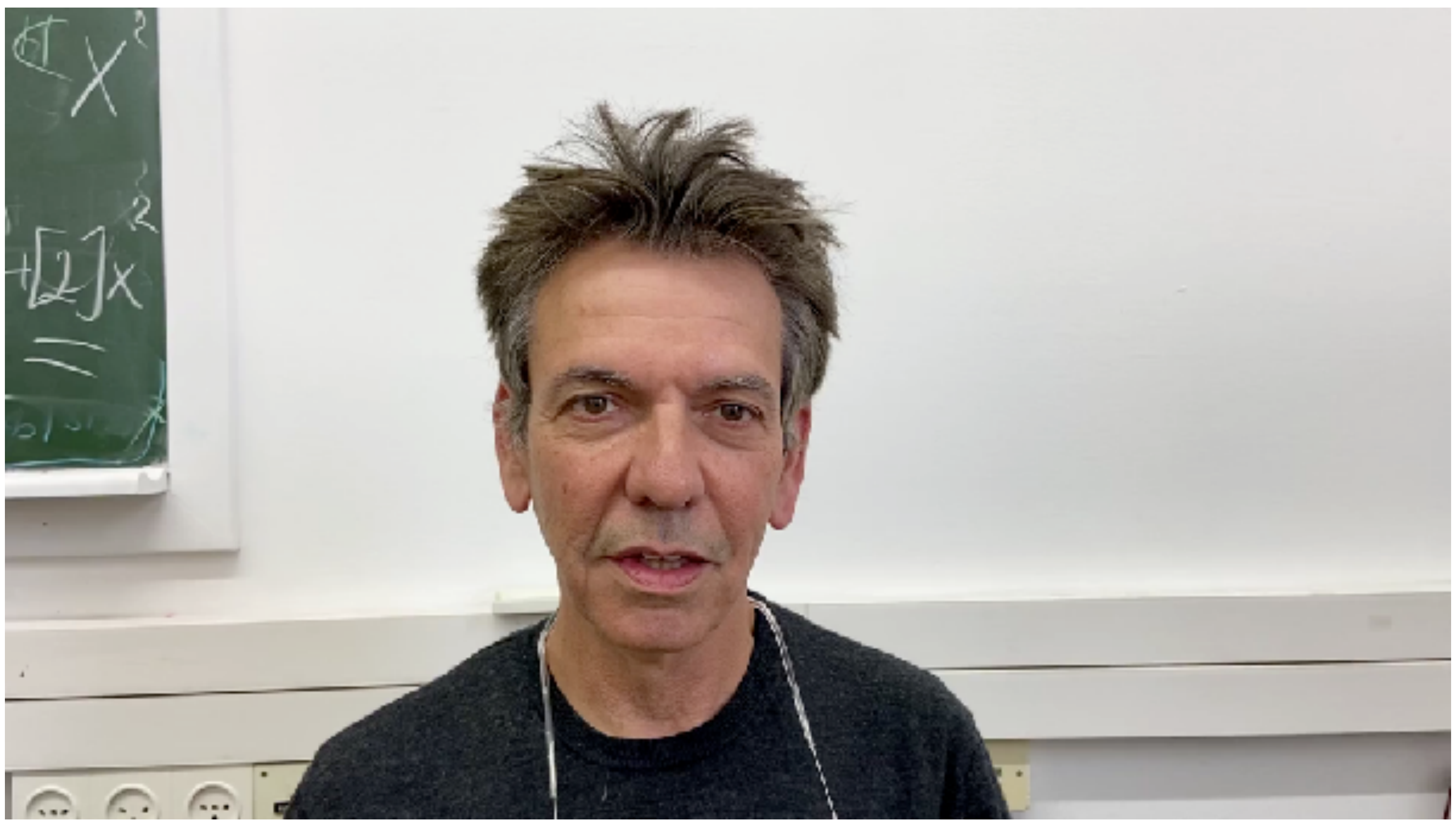


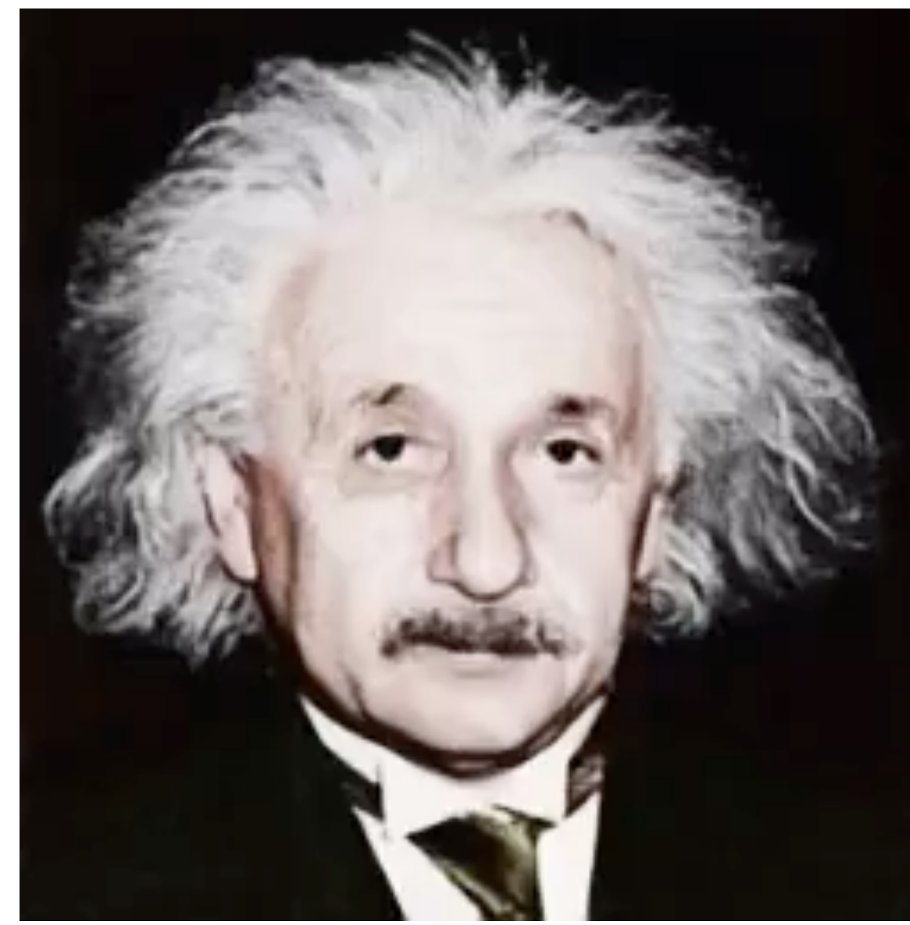
Hello everybody
and welcome to the
Weizmann Practical Deep Learning course
2021.
We hope you enjoy the course.
Have fun.
Bye bye.

Practical Deep Learning 2021



Practical Deep Learning 2021





- What is intelligence?
- What is artificial intelligence?
- What is Deep learning

From Intelligence to Deep Learning

- **What is intelligence?**
  - Intelligence is the ability to process information so it can be used to infer on decisions for the future
- **What is artificial intelligence?**
  - Use the computer to mimic human intelligence
- **What is Machine learning**
  - Learning from experience without being explicitly programmed
- **What is Deep Learning?**
  - Using NN to automatically extract patterns from DATA and use it for inferring and make decisions

Why DL is a Boom Now?

- ML started in the 1950s
- Deep Convolutional Nets since the 1990s
- Autonomous driving is now a multi billion dollars business… TESLA is already there… Why only now?
    - Big DATA - Cheap Storage, easy access, and lots of Big Data
    - GPUs are changing the face of the computer hardware (Parallelizable tasks)
    - Sophisticated software/firmware tools for Deep Learning Models implementation

Machine Learning Example

- Map inputs (such as images) to targets (such as labels: Cat, Dog, Woman)

- BUT let your mind flies by

Layer transforms DATA into a useful representation

Its all about finding appropriate representations

This is done via training rather than programming

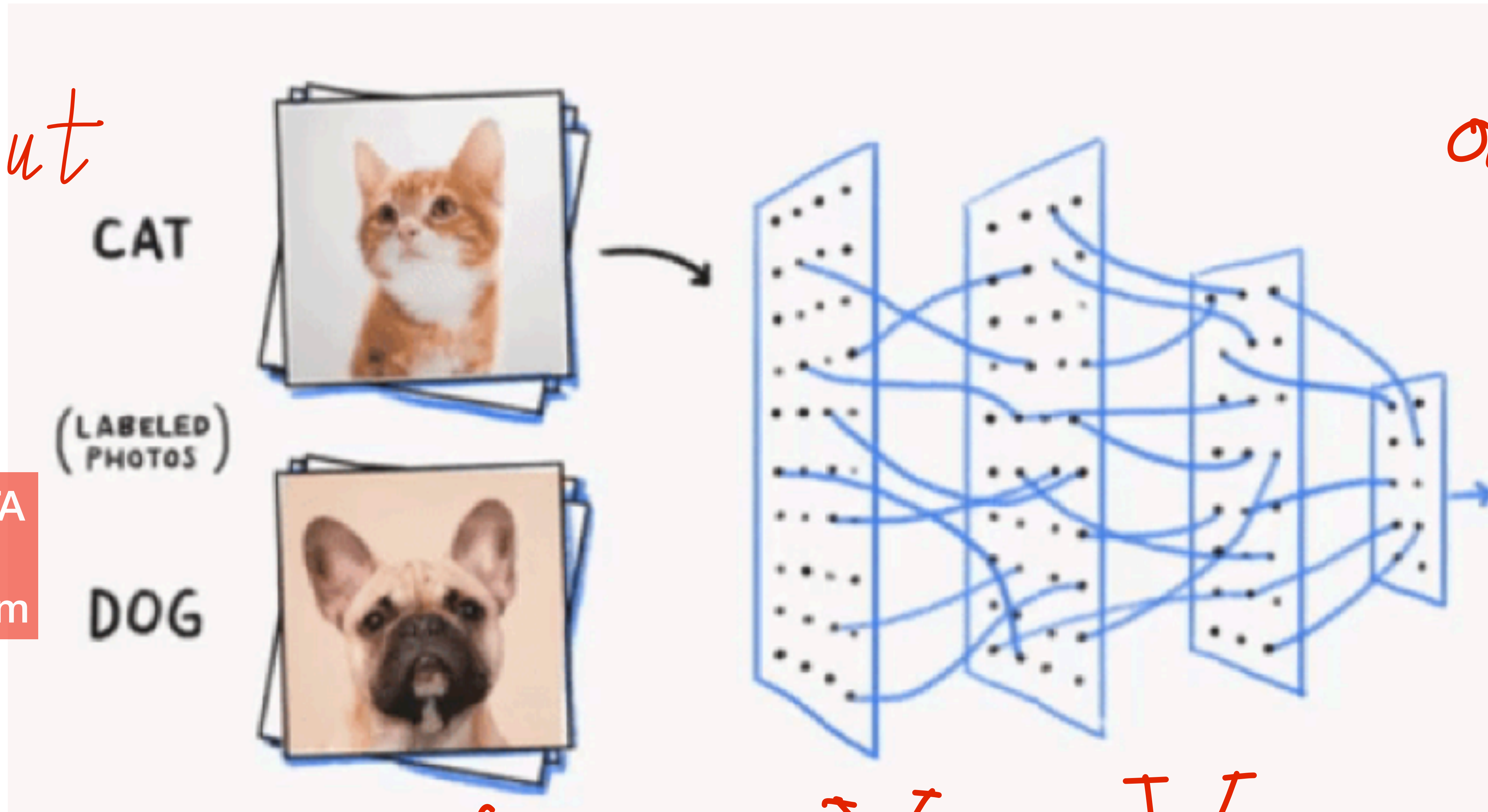
Transformation is controlled by weights

Example: Classification Dog vs Cat

input

X

The DATA we can learn from



output

Y

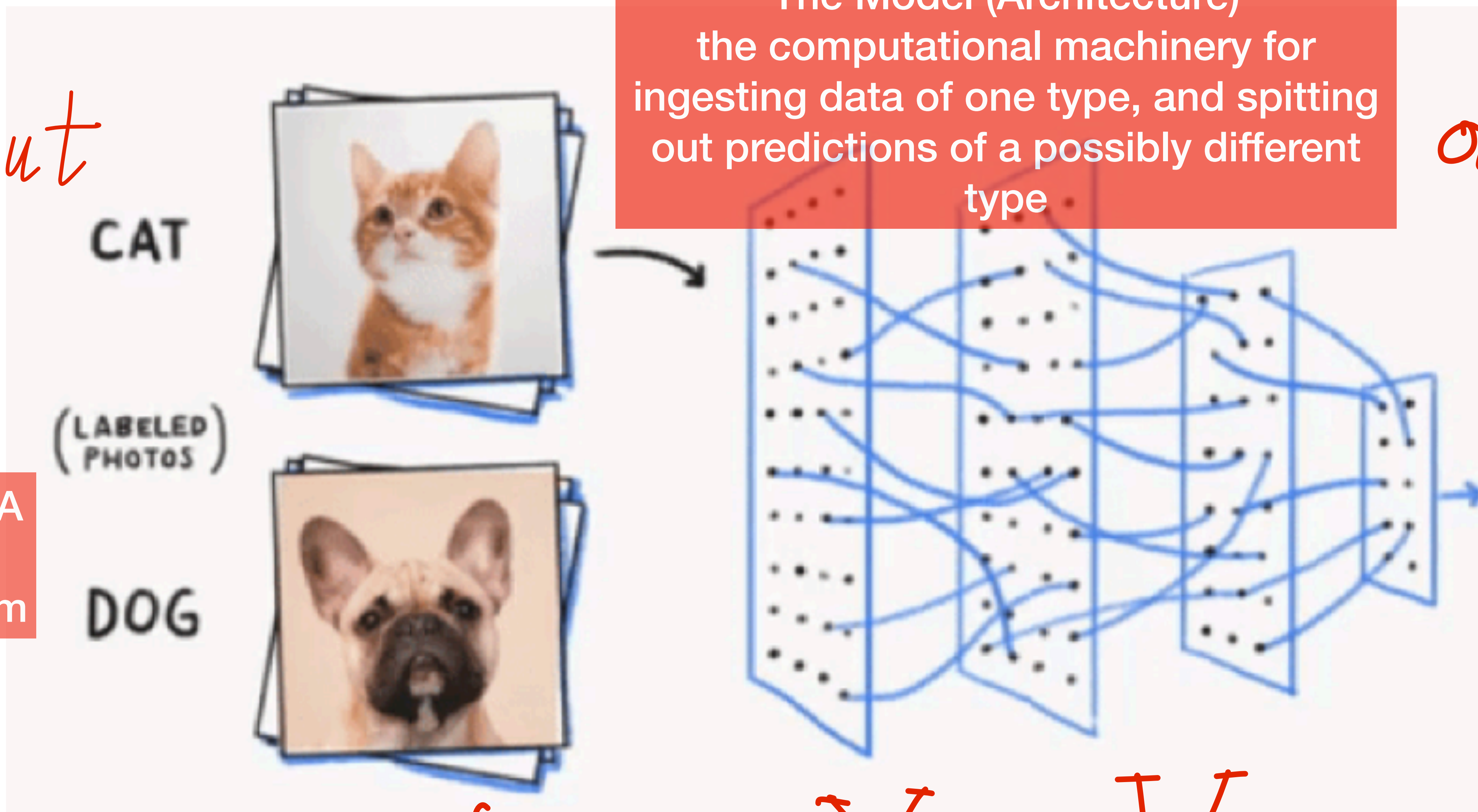
$$f_{(w)} : X \to Y$$

Example: Classification Dog vs Cat

input

X

The DATA we can learn from



The Model (Architecture)
the computational machinery for ingesting data of one type, and spitting out predictions of a possibly different type

output

Y

$f(w): X \to Y$

Example: Classification Dog vs Cat

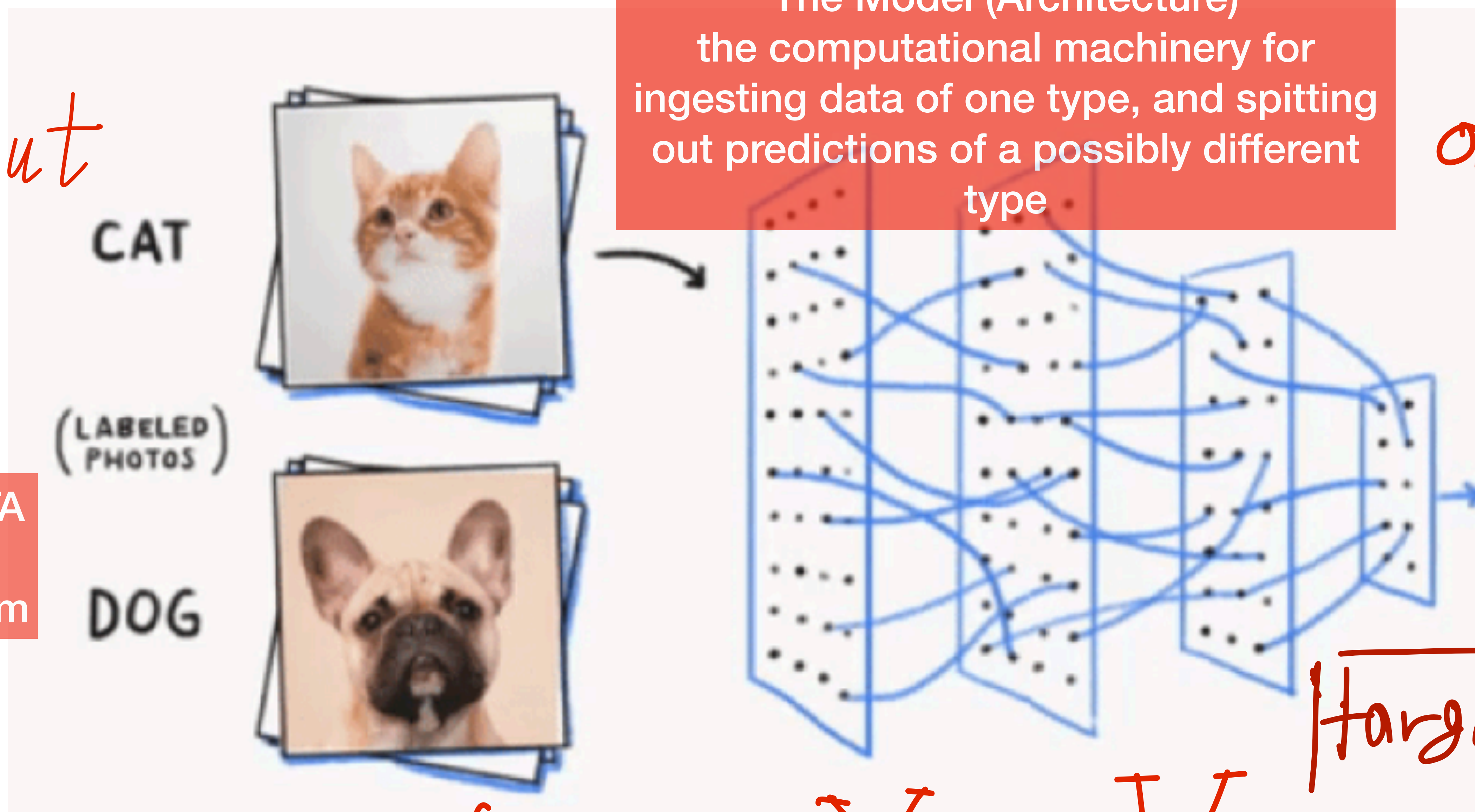
The Model (Architecture)
the computational machinery for ingesting data of one type, and spitting out predictions of a possibly different type

input

output

X

Y



The DATA we can learn from

$f(w): X \to Y$

target $y \in Y$

prediction $\hat{y}$

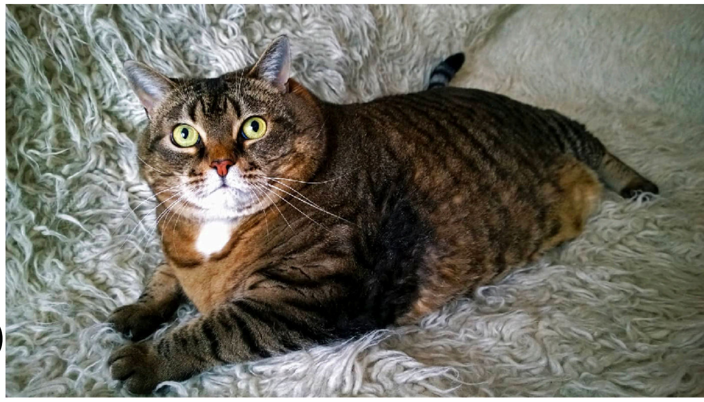
Loss $= f(y, \hat{y})$

e.g. $(y-\hat{y})^2$

The Loss Function:
minimize error rate, i.e., the fraction of instances on which our predictions disagree with the ground truth

Artificial Intelligence Philosophy

Artificial Intelligence



- The Task: Recognise a Cat

- Easy for people to perform. hard to describ formally

- Recognition of an object, or a spoken word, is many times intuitive, almost automatic, but hard to describe “how do I do it”.

- Let computers do what we do, learn from experience

Hierarchy of Concepts

- Chess is a very simple DATA set of objects and rules, yet the play is **conceptually** extremely complicated and difficult…
- Let the computer understand the world as a hierarchy of concepts, each defined in terms of its relation to a simpler concept
- Concepts are built on top of each other; complicated concepts are built on top of simpler ones… —> DEEP LEARNING approach to AI

DATA representations

- Much of our knowledge is subjective and intuitive
- Computers need to capture this knowledge in order to make intelligent decisions
- The capability to acquire knowledge by extracting patterns from raw DATA is what Machine Learning is all about

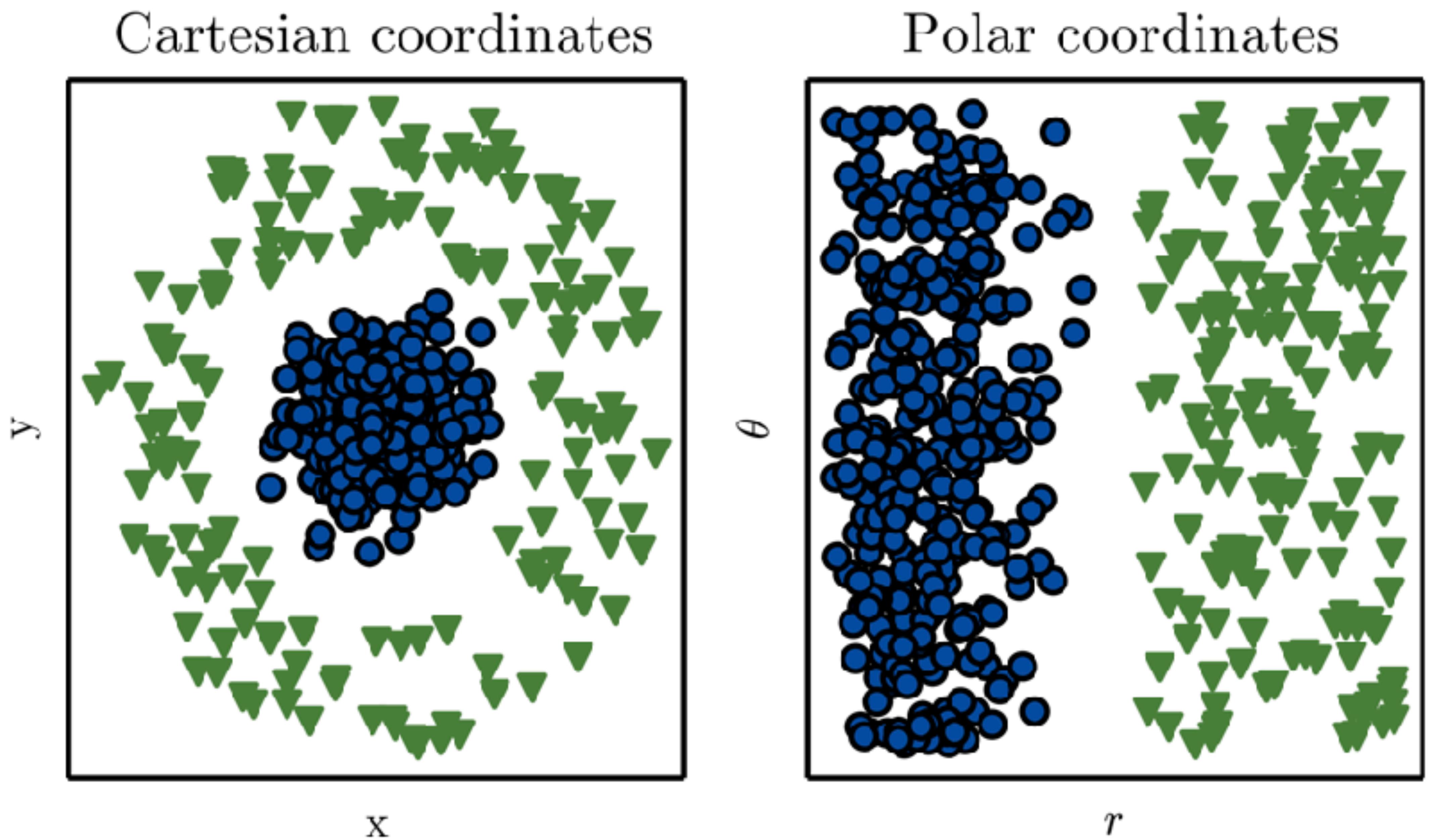

label: DOG


label: CAT

- The performance of a ML algorithm depends heavily on the **representation** of the DATA they are given.
- Each piece of information given in the representation is called a **feature**.

DATA representations

- Example:



- Choosing the right set of features can make a huge difference in solving a task

DATA representations

- Sometimes its difficult to know which features should be extracted….

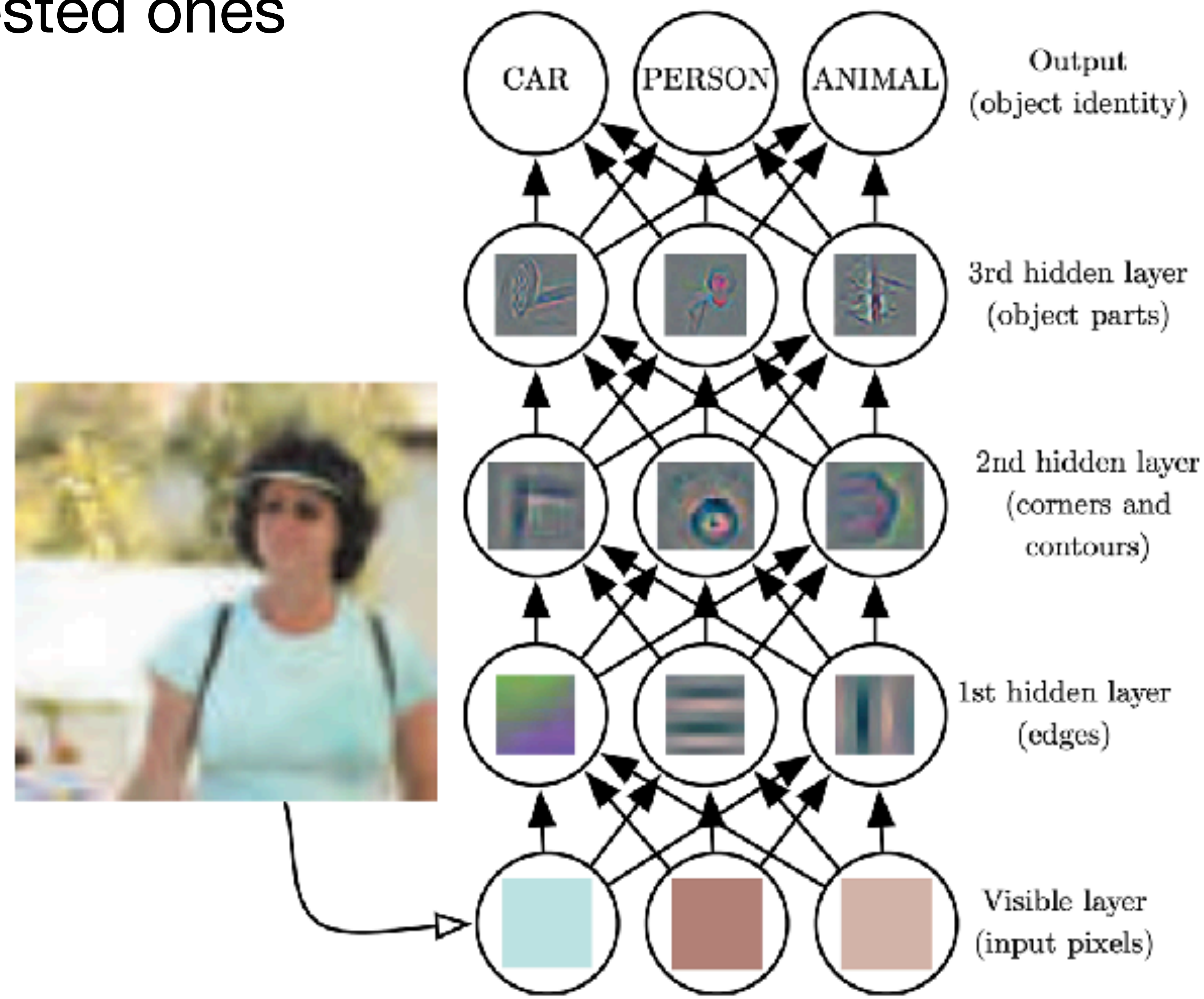
- One solution is **representation learning,** learn the mapping from a representation to a representation… even to itself

- **DEEP LEARNING** is about expressing representations in terms of simpler ones… The computer builds complex concepts out of simpler concepts

- An **Autoencoder encodes** a representation of the DATA to a different representation , while the **decoder**, converts it back to the original representation

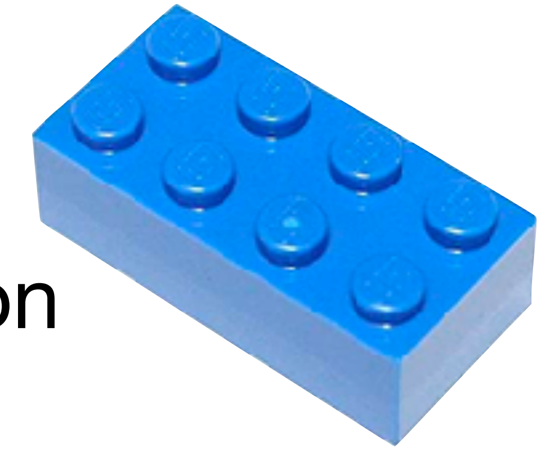
Deep "representation" Learning

- Break the complex presentation into a series of simpler nested ones

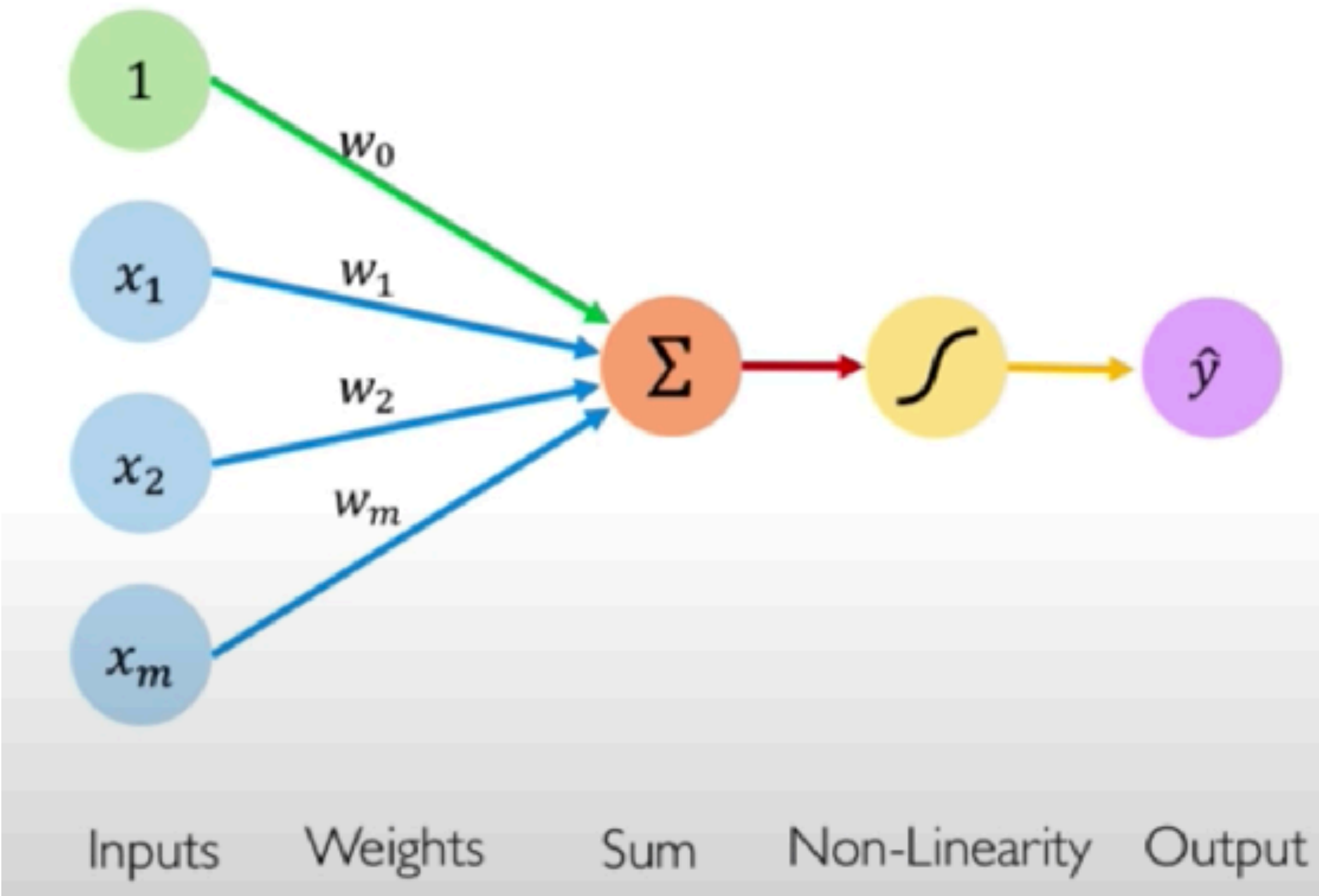


Based on Zeiler and Fergus, 2014

Perceptron



- The basic unit of Deep Learning is the Perceptron



http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L1.pdf

output

BIAS TERM

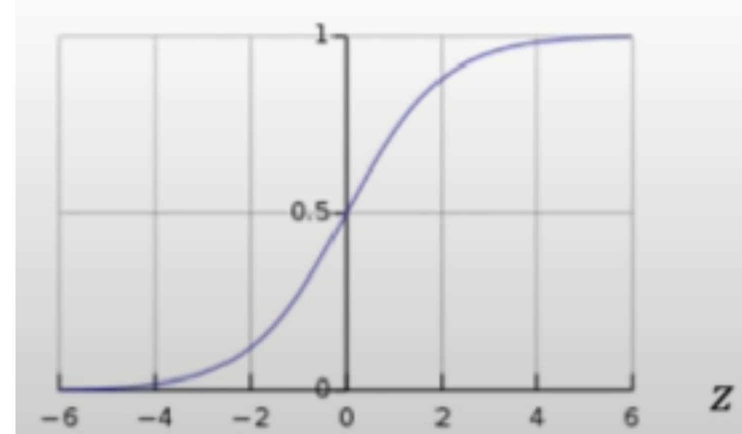
LINEAR COMB OF INPUTS

$$\hat{y} = g\left(w_0 + \Sigma_{i=1}^{m} w_i x_i\right)$$

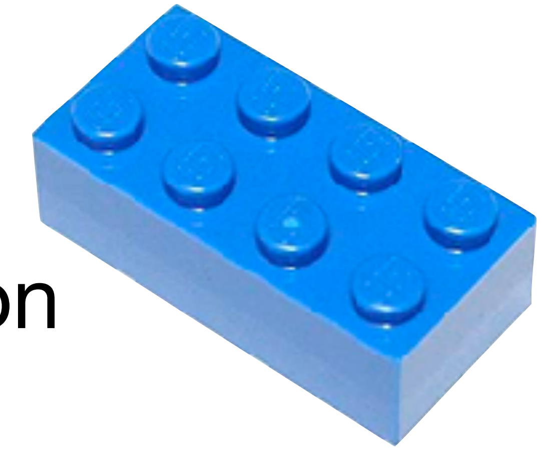
NON LINEAR ACTIVATION FUNCTION

Example: sigmoid function

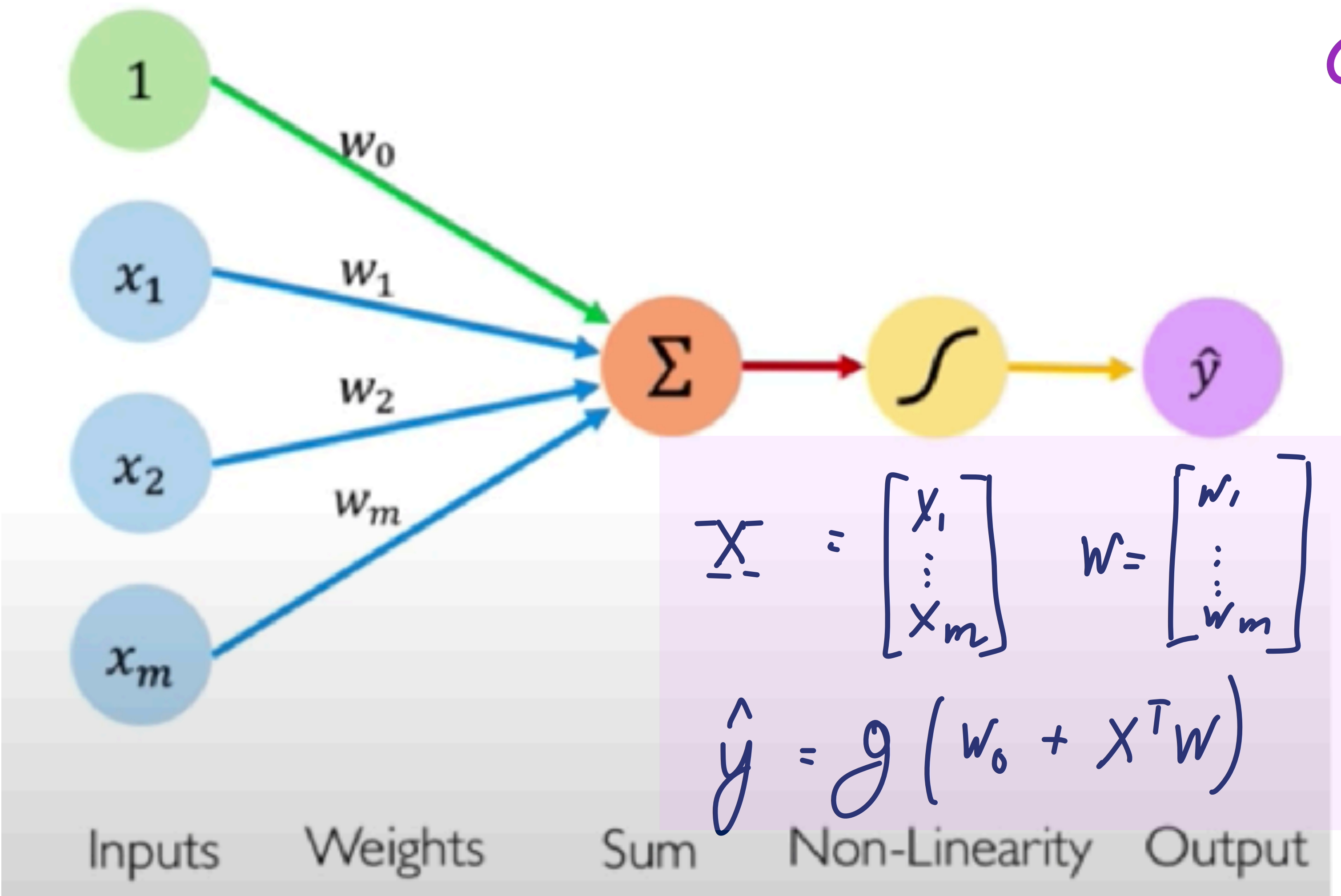
$$g(z) = \sigma(z) = \frac{1}{1 + e^{-z}}$$



Perceptron



- The basic unit of Deep Learning is the Perceptron



$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \quad W = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}$$

$$\hat{y} = g\left(w_0 + X^T W\right)$$

http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L1.pdf

output

BIAS TERM

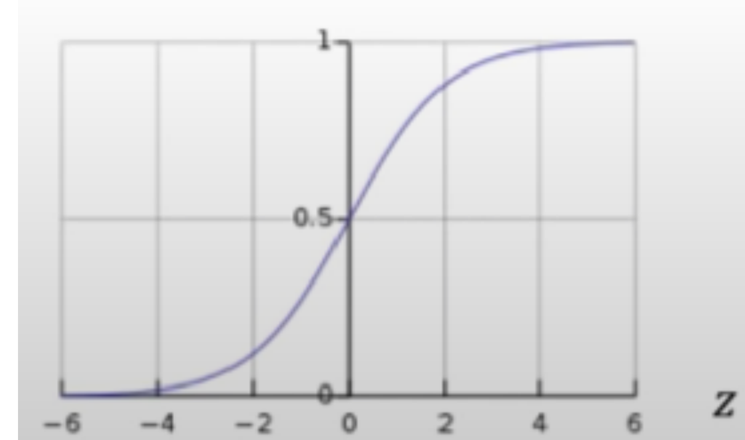
LINEAR COMB OF INPUTS

$$\hat{y} = g\left(w_0 + \Sigma_{i=1}^{m} w_i x_i\right)$$

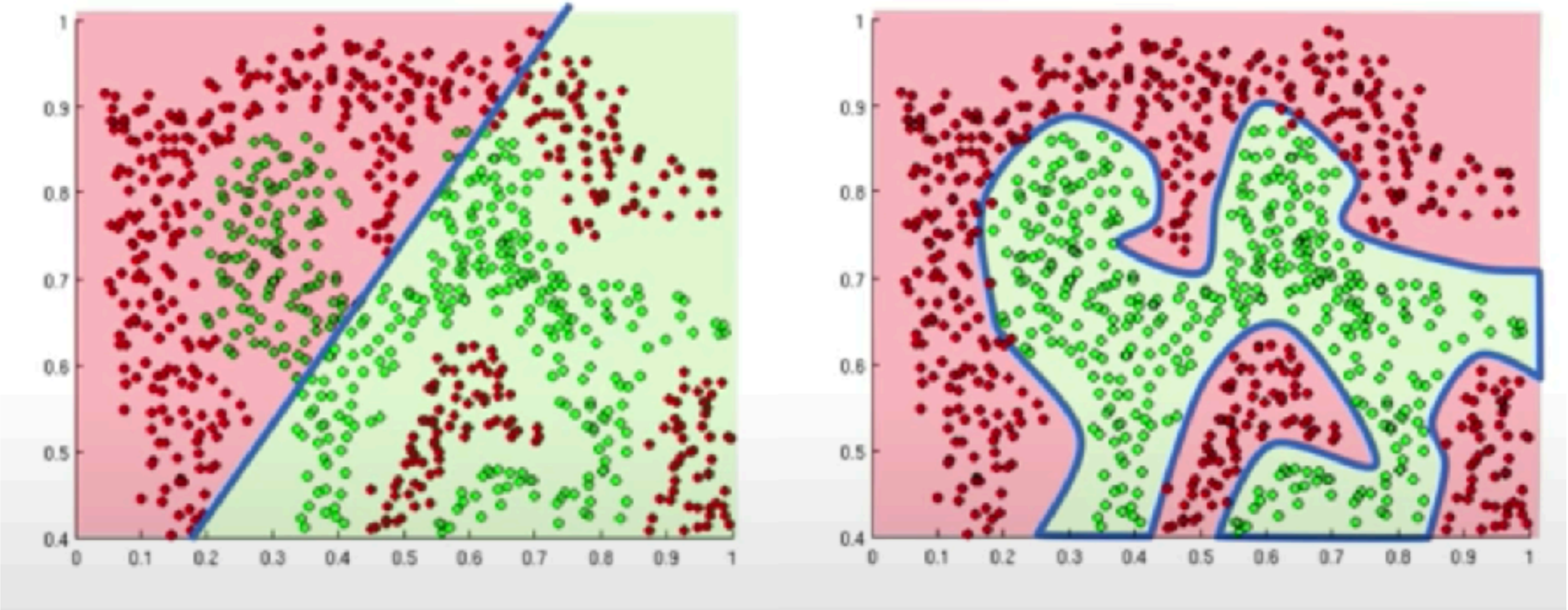
NON LINEAR ACTIVATION FUNCTION

Example: sigmoid function

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



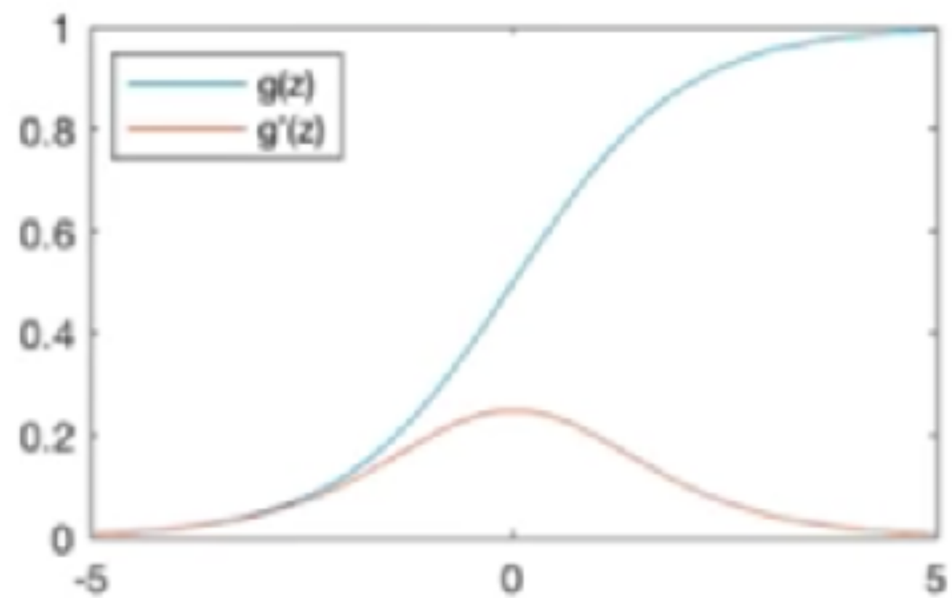
Why non-linearity?



http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L1.pdf

- Linear —> Linear decision boundary
- Non-linear—>Non-linear (complex) decision boundary

Non-Linear Activation Functions

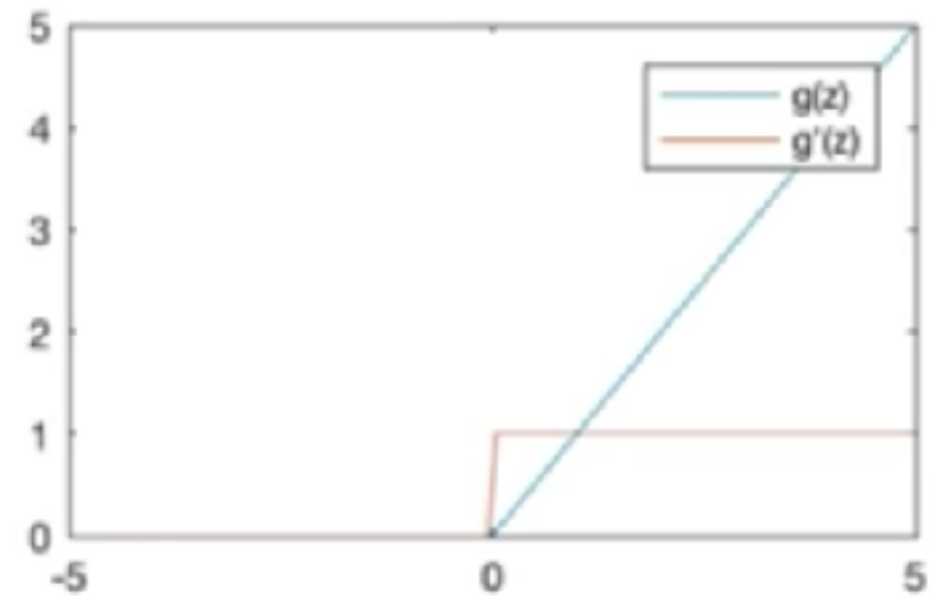


$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L1.pdf

Sigmoid Saturates and kill Gradients



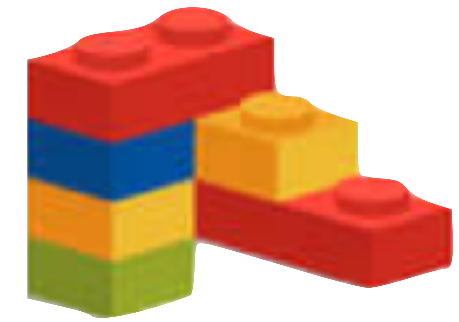
$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

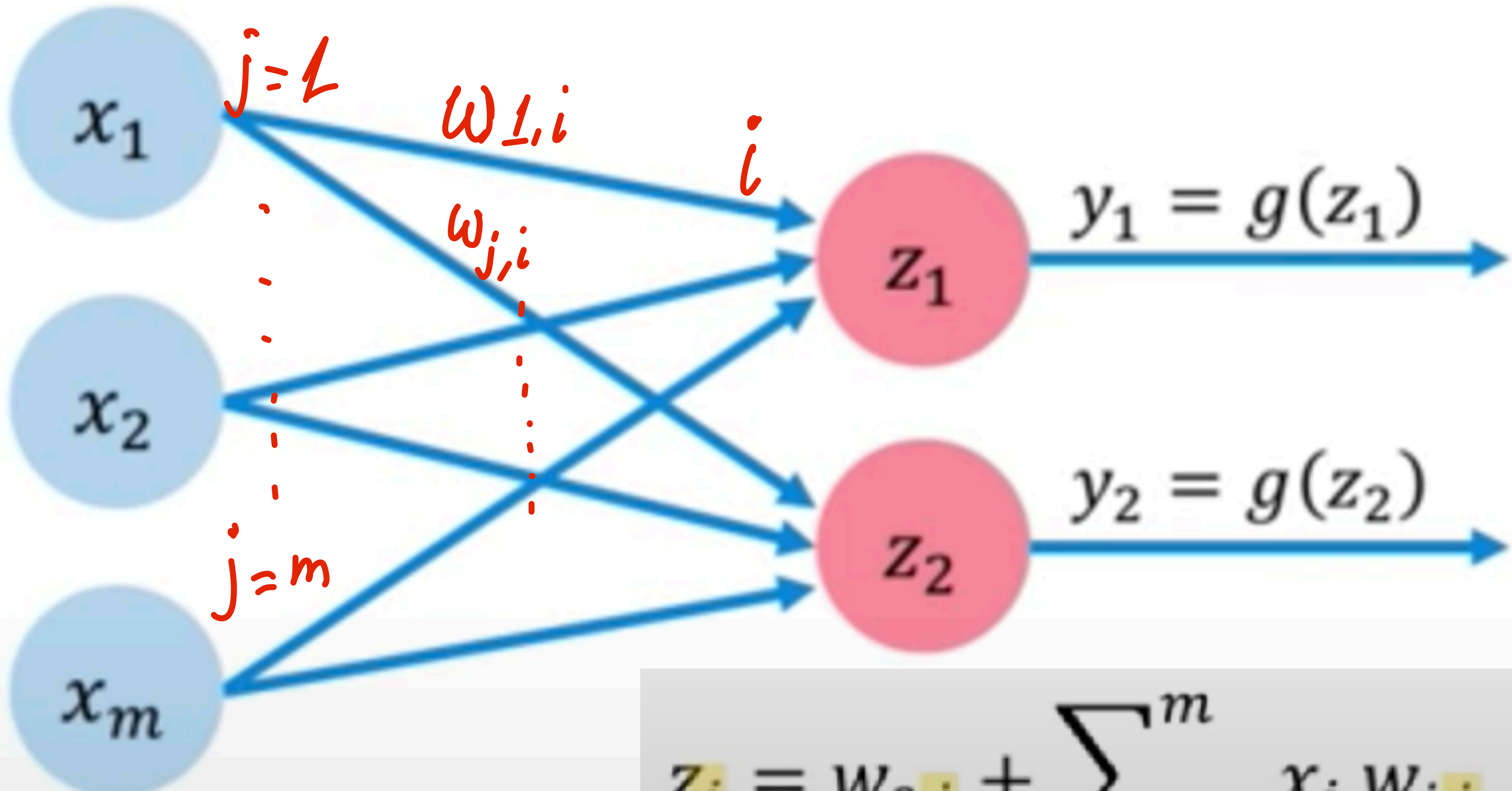
ReLu- Rectified Linear units

Many times you use ReLu **for all Hidden Layers and** Sigmoid **in the final layer as to output a probability (a score between 0 and 1)**

Multi-out Perceptron



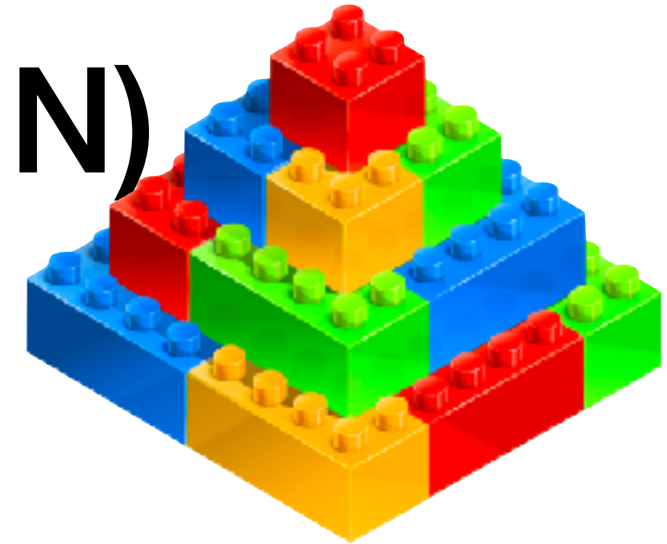
- Since all inputs connected to all outputs, we call It DENSED layer

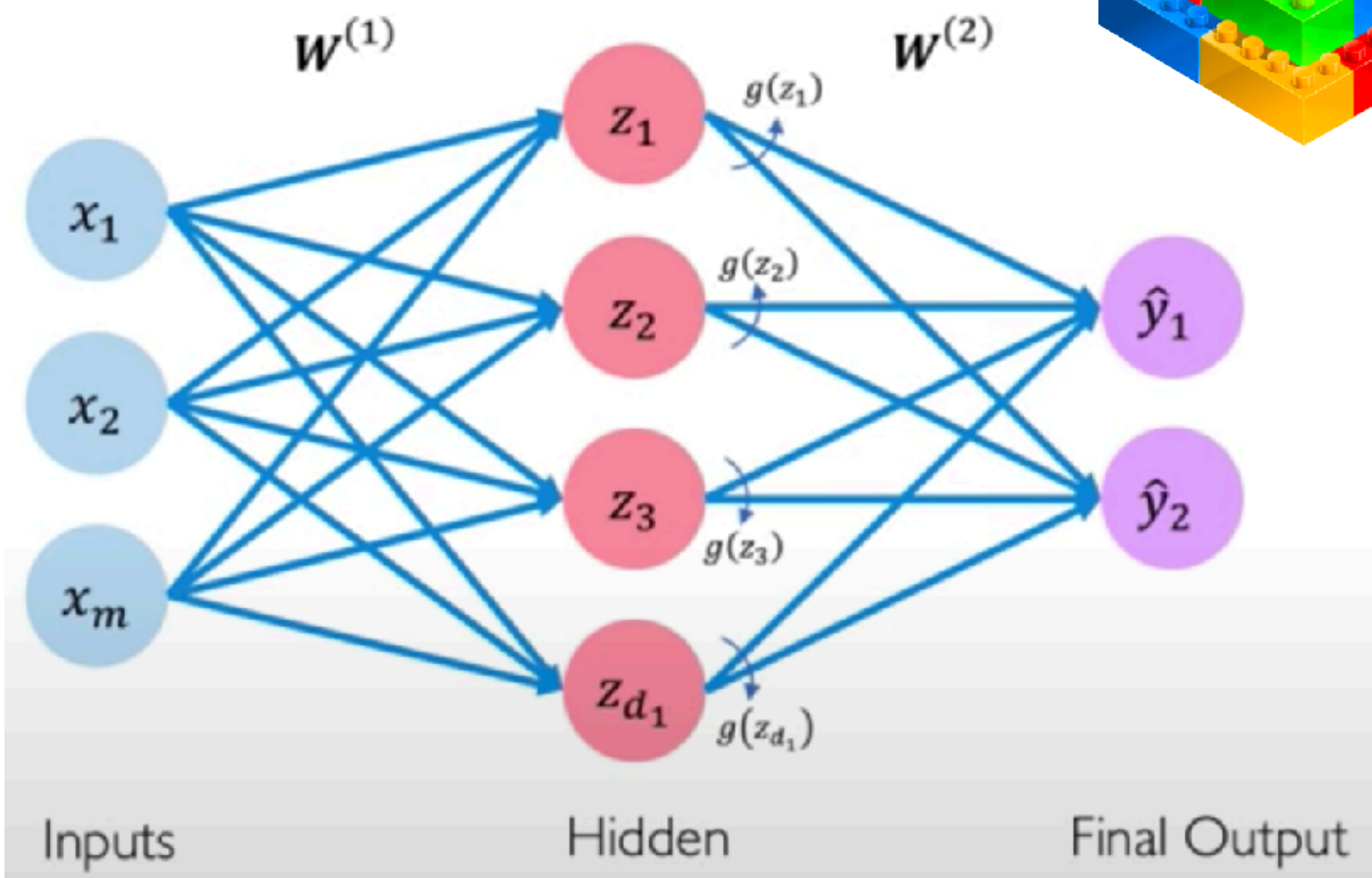


$$z_i = w_{0,i} + \sum_{j=1}^{m} x_j \, w_{j,i}$$

$$Z = W_0 + X^T W$$

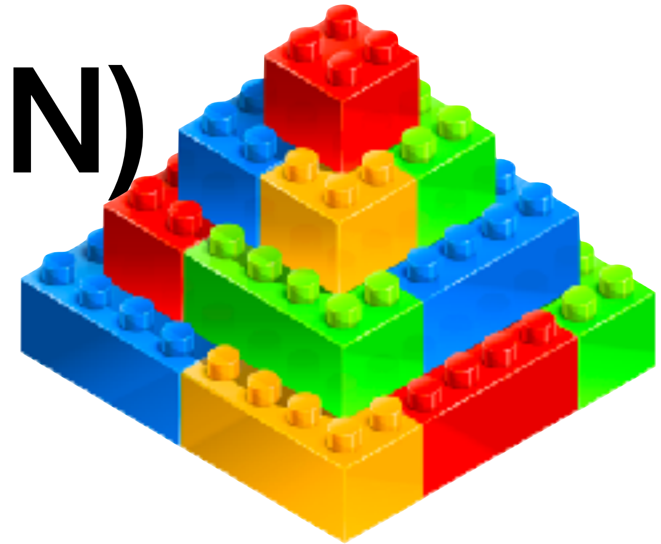
Single Layer Perceptron (NN)



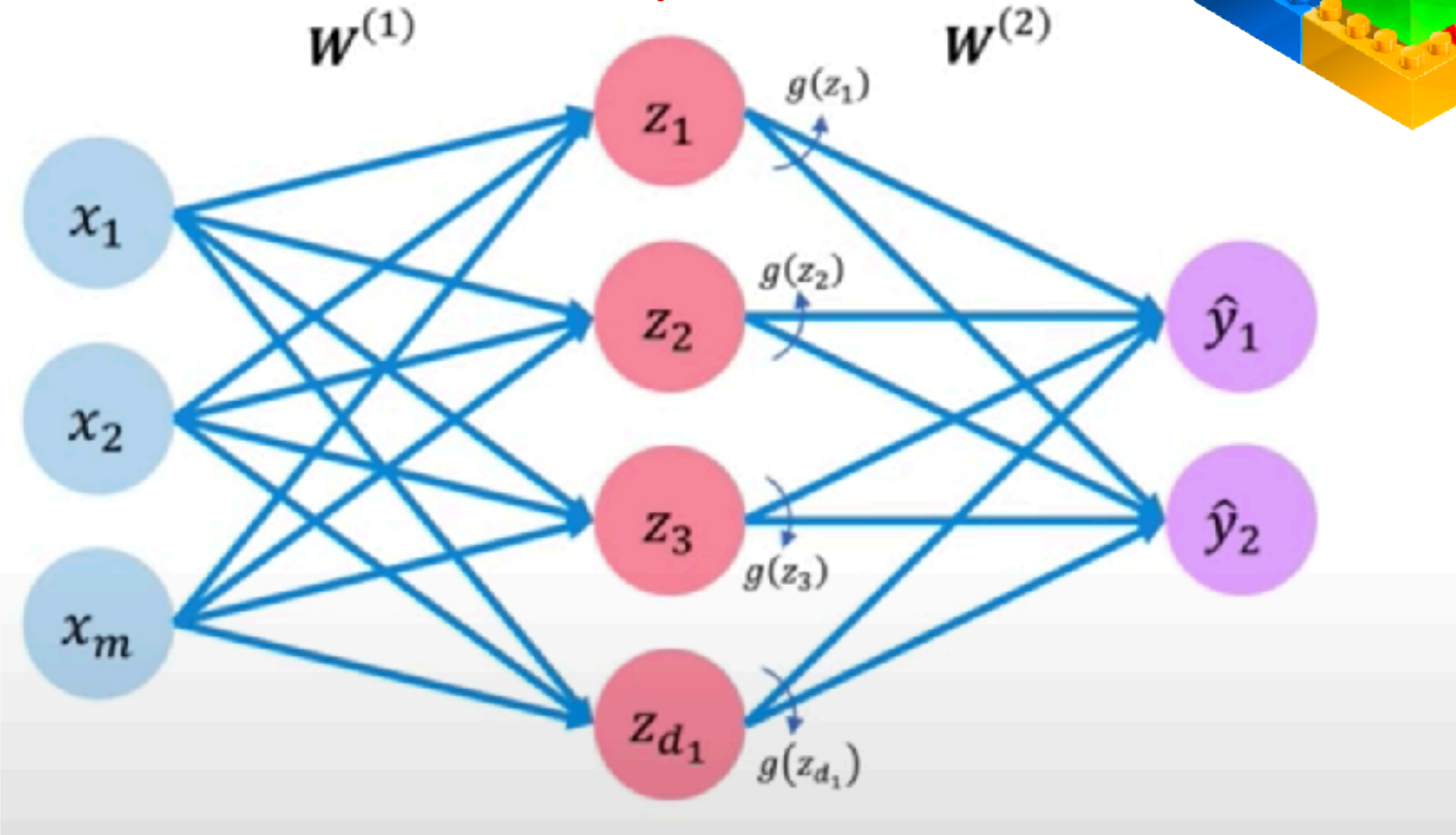


$$z_i = w_{0,i}^{(1)} + \sum_{j=1}^{m} x_j \, w_{j,i}^{(1)} \qquad \hat{y}_i = g\left(w_{0,i}^{(2)} + \sum_{j=1}^{d_1} z_j \, w_{j,i}^{(2)} \right)$$

Single Layer Perceptron (NN)

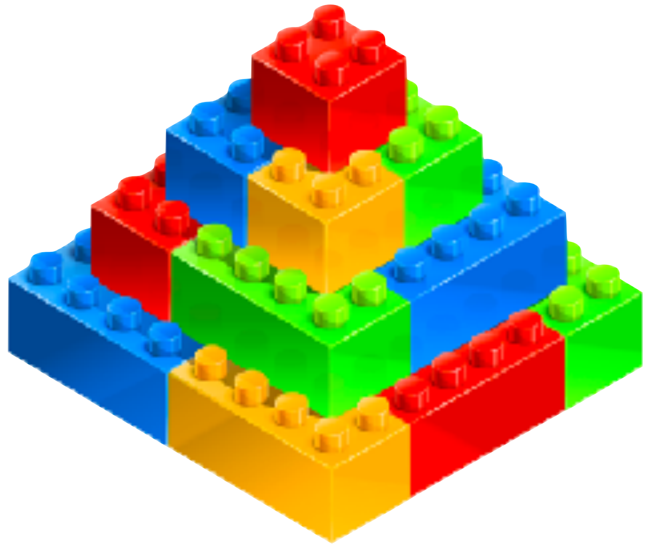


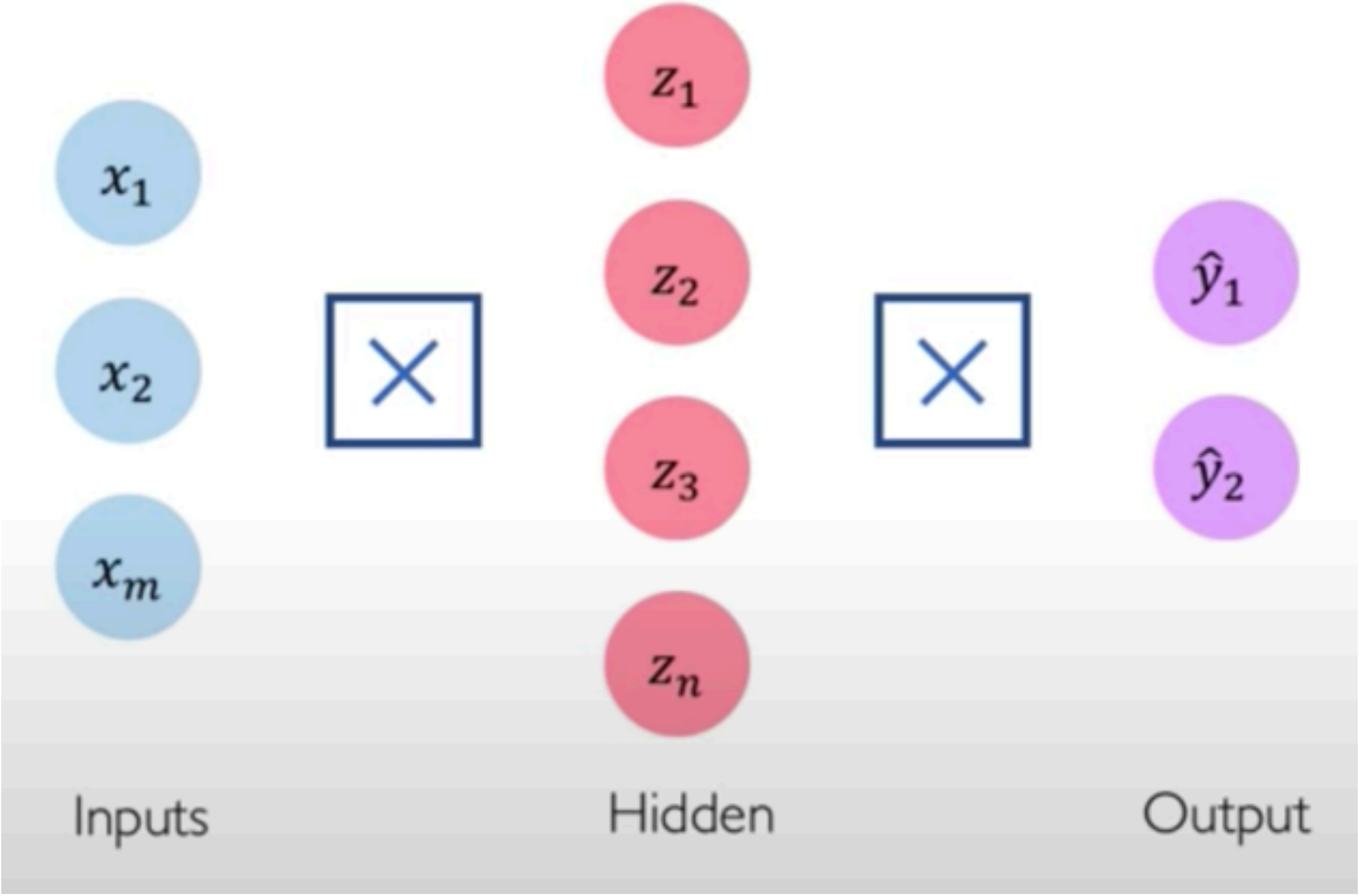
layer 1



$$Z^{(1)} = w_0^{(1)} + X^T W^{(1)}$$

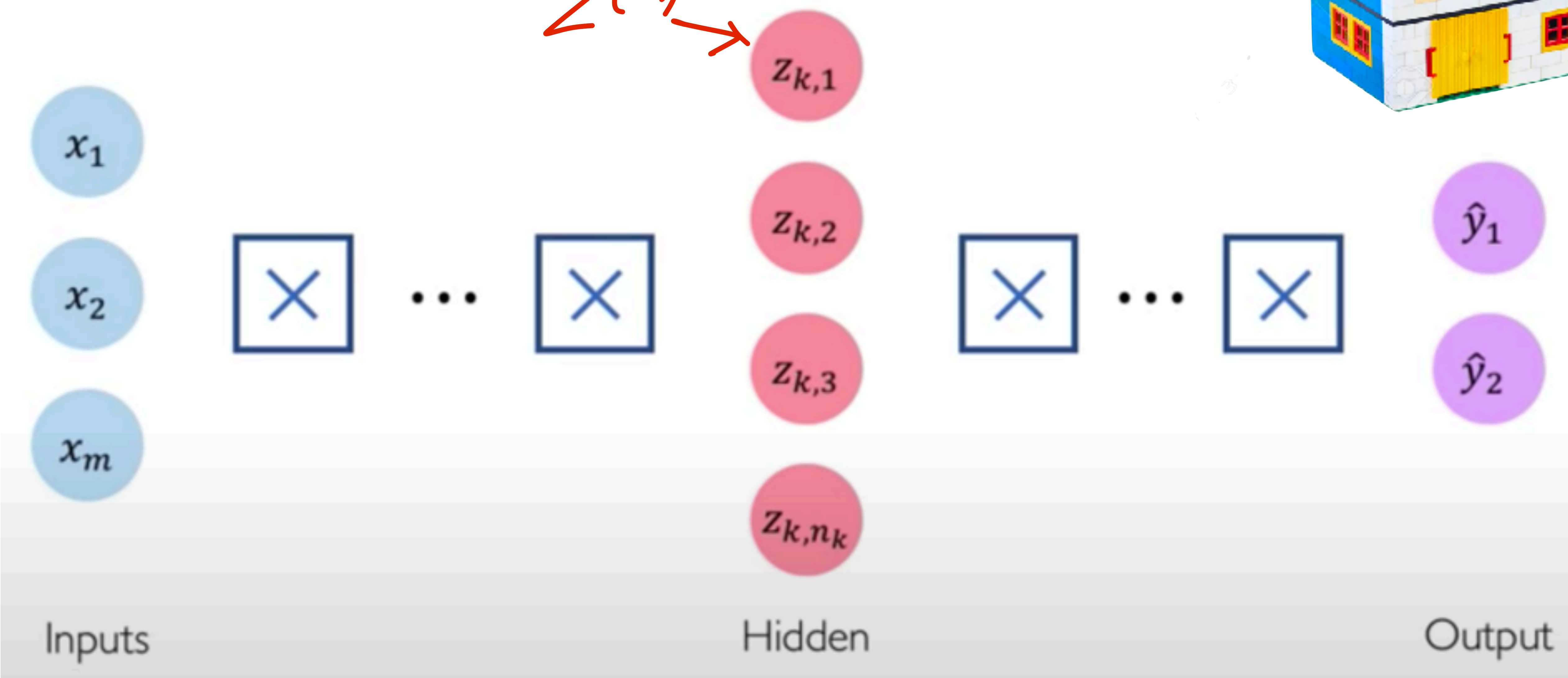
Simplified Drawing





Multi Layer Perceptron (MLP)

$Z^{(k)}$ layer k



$$Z^{(1)} = W_0^{(1)} + X^T W^{(1)}$$

$$Z^{(k)} = W_0^{(k)} + g\left(Z^{(k-1)}\right) \cdot W^{(k)}$$

Summary: Layers & Weights

- Layers extract representation of the DATA fed into them, which are supposed to be more meaningful for decoding the DATA

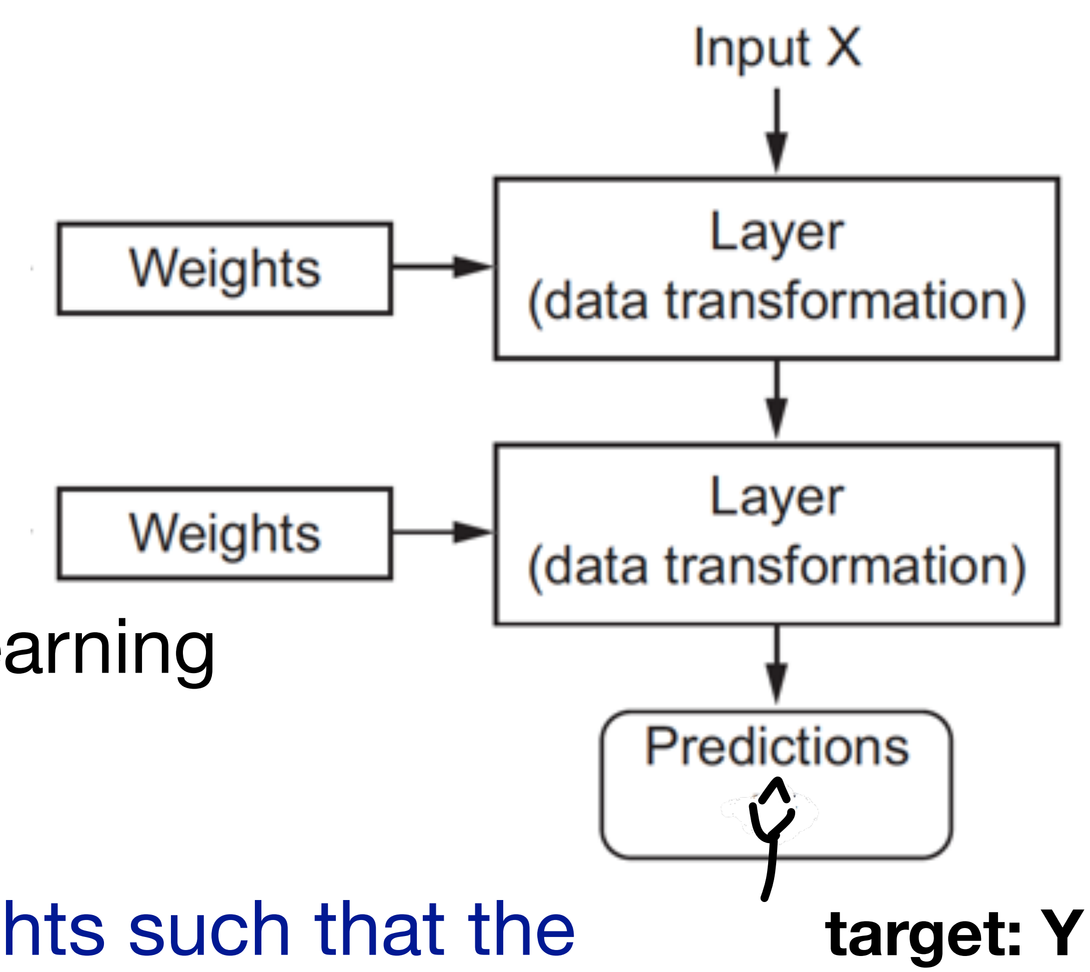
- Some DATA goes in, and comes out in a more useful form

- You can 🤔 of layers as “filters”

- Weights control what the layer is doing to it’s input DATA

- The depth of the model is the number of layers contribute to the learning process

- GOAL: Find right values for the weights such that the network maps the input to its right target



Loss Optimization

- Find the NN weights that give the minimus loss

$$\boldsymbol{W}^* = \underset{\boldsymbol{W}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\left(f\left(x^{(i)}; \boldsymbol{W}\right), y^{(i)}\right)$$

$$\boldsymbol{W}^* = \underset{\boldsymbol{W}}{\operatorname{argmin}}\, J(\boldsymbol{W})$$

$$W = \{ W^{(0)}, W^{(1)}, \ldots \}$$

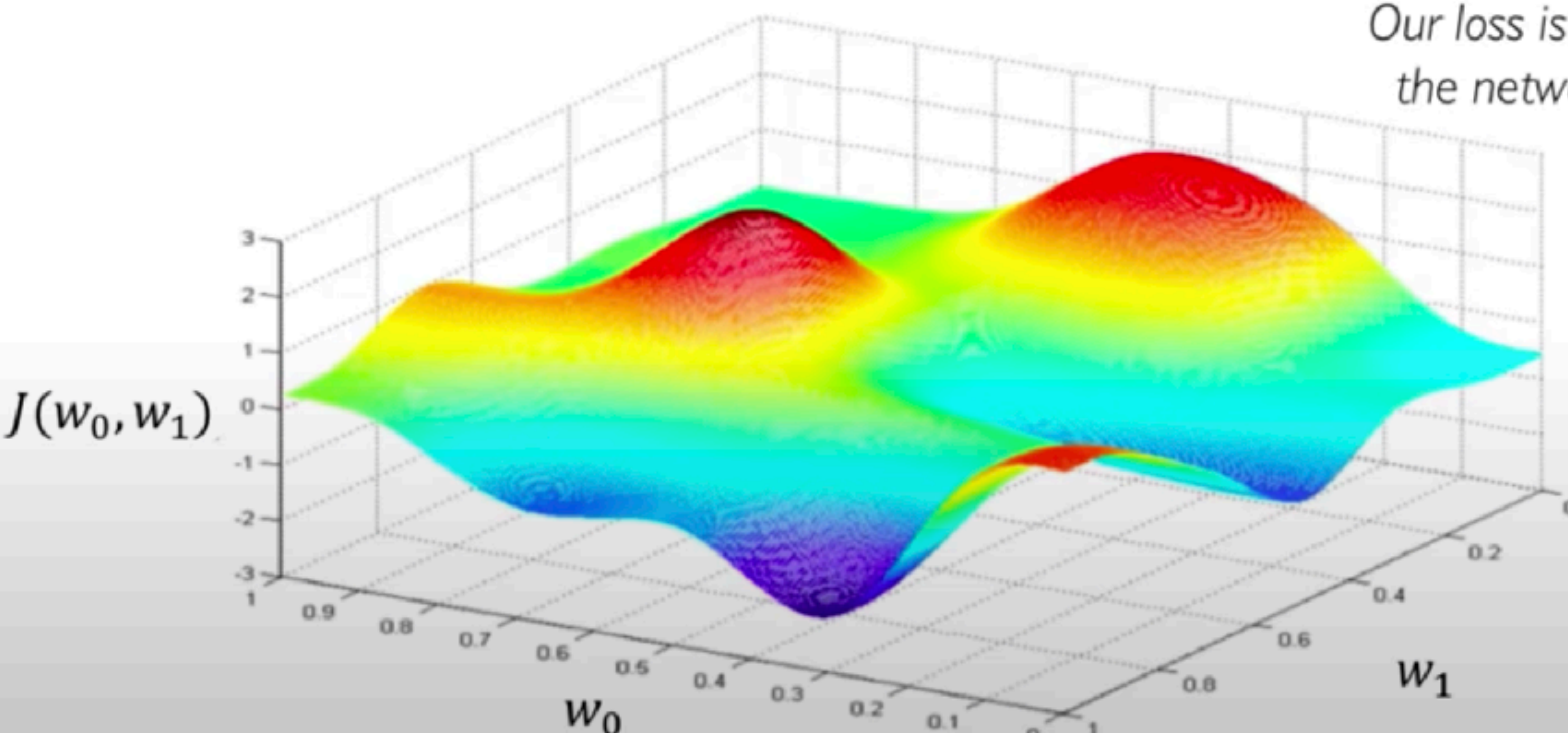
Loss Optimization

- The Loss is a function of the NN weights

for simplicity assume 2D W

$$W^* = \underset{W}{\operatorname{argmin}} J(W)$$

Remember:
Our loss is a function of the network weights!



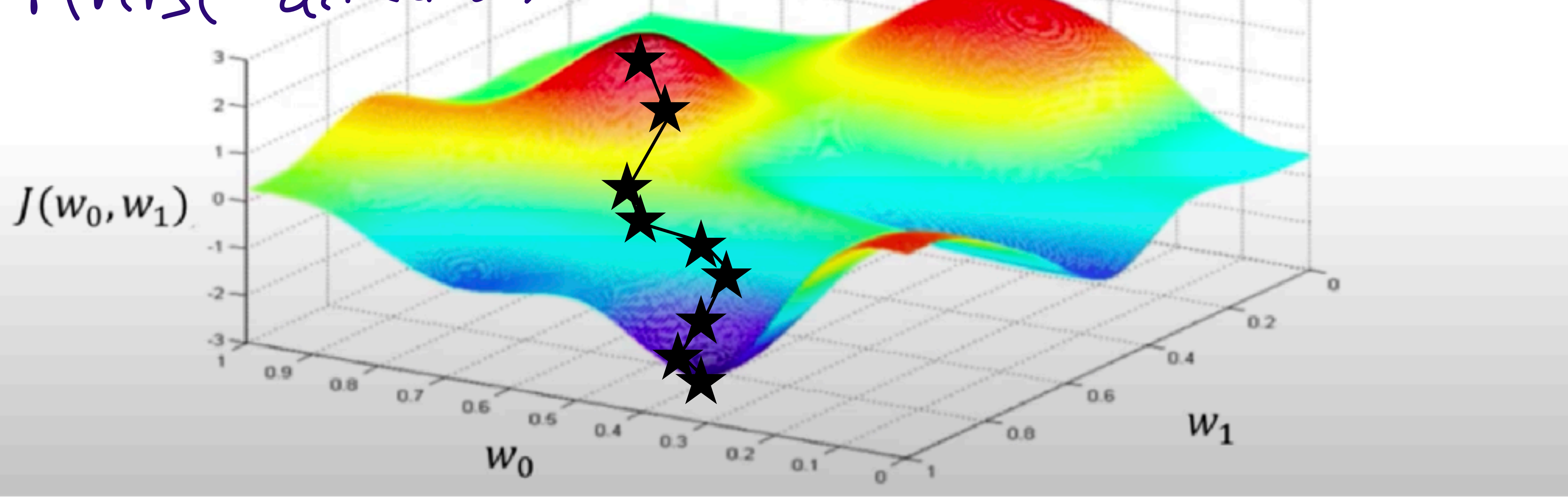
Loss Optimization: Gradient Descent

- The Loss is a function of the NN weights

$$\boldsymbol{W}^* = \underset{\boldsymbol{W}}{\operatorname{argmin}}\, J(\boldsymbol{W})$$

Remember:
Our loss is a function of the network weights!

Compute direction of maximum ascent (gradient)
reverse direction, more on



Gradient Descent

- Algorithm
  - Initialize weights randomly
  - Loop until convergence:
    - Compute Gradient $\frac{\partial J(W)}{\partial W}$
    - Take a step η (the learning rate - how fast you want to achieve the goal)
    - Update weights in the opposite direction
      $W \rightarrow W - \eta \frac{\partial J(W)}{\partial W}$
    - Return weights

BACKPROPAGATION

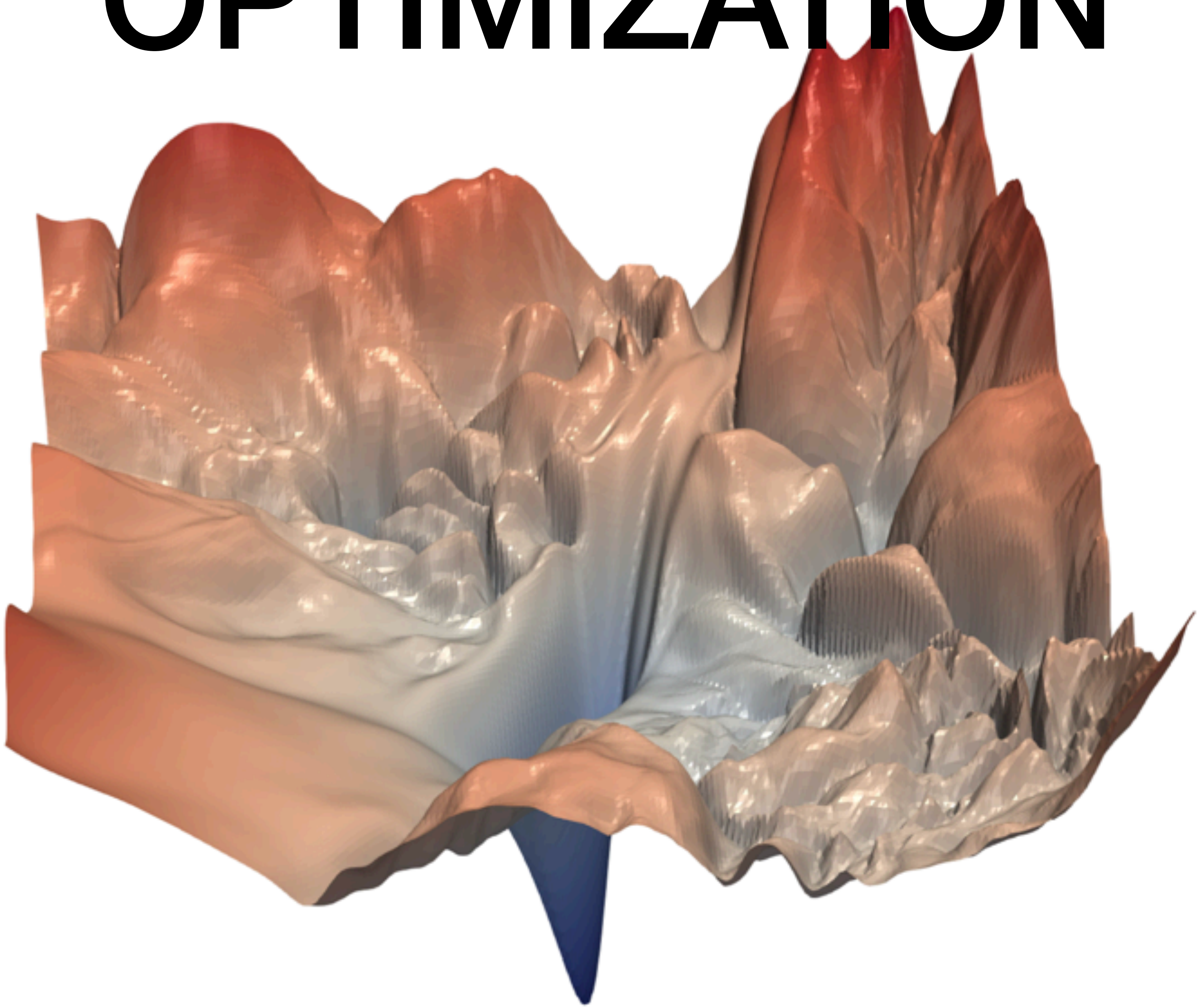
- **Backpropagation** is the algorithm that computes the gradient of a loss function with respect to the weights of the NN in an efficient way

- It is essential to do so in an efficient way in order to cope with Multi Layer Networks.

- Backpropagation is calculating the gradient iterating backwards from the last layer to the network input

- —>next lecture

OPTIMIZATION



https://papers.nips.cc/paper/7875-visualizing-the-loss-landscape-of-neural-nets.pdf

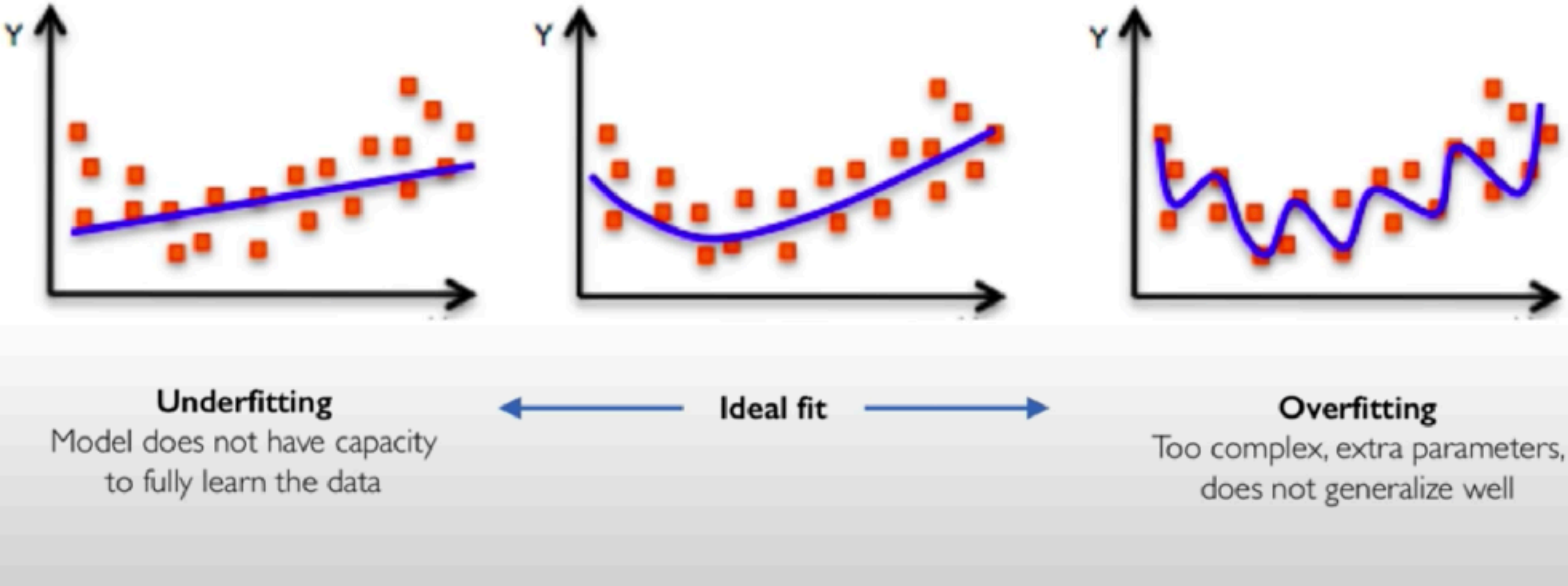
Optimization of NN

Optimization through gradient descent

$$\boldsymbol{W} \leftarrow \boldsymbol{W} - \boxed{\eta} \frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}}$$

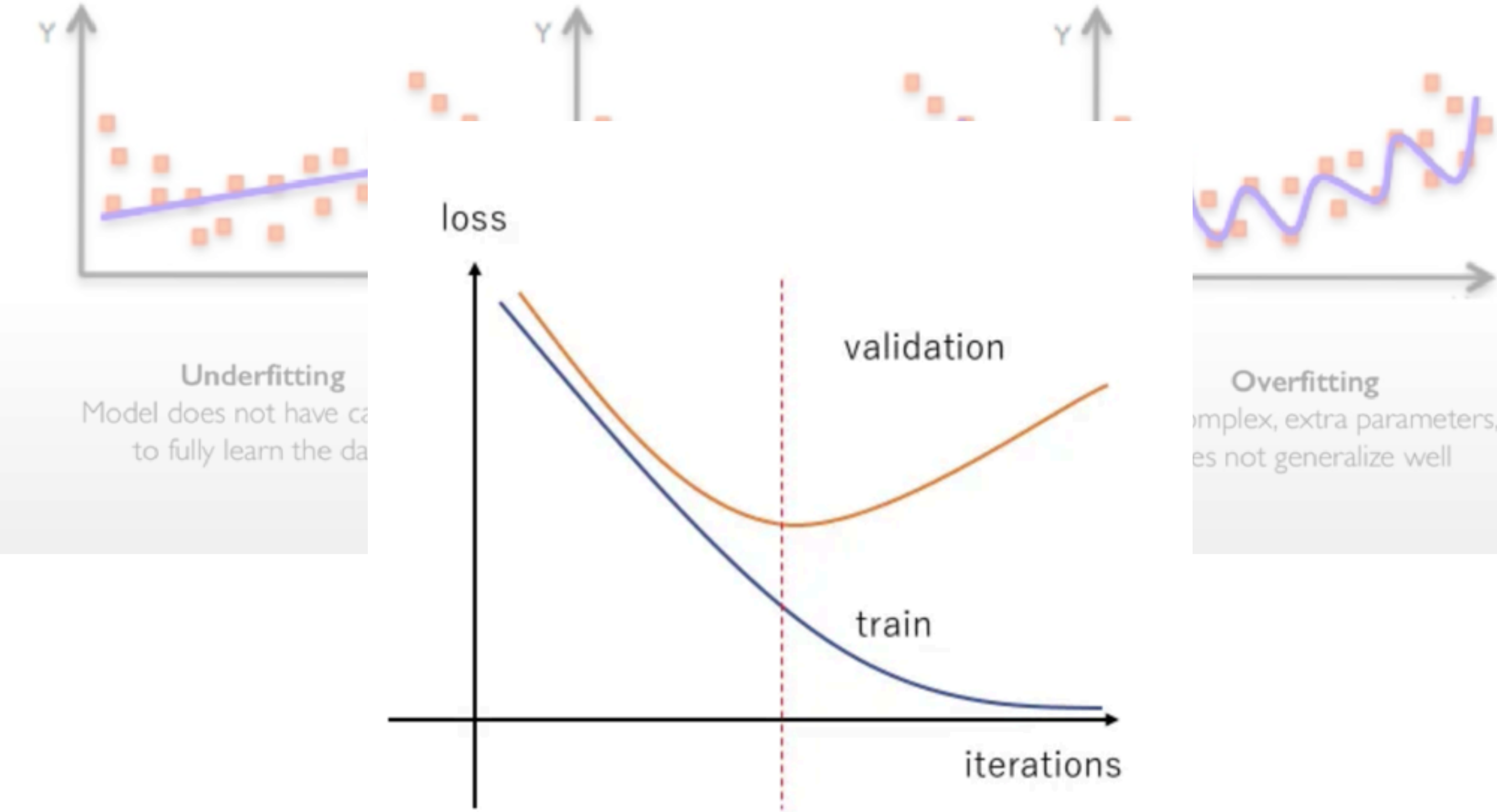
How can we set the learning rate?

Overfitting



Overfitting

- Stop Loss



Training

- We divide our labeled DATA into~80% training and 20% validation (sometimes also keep some DATA for test)

- In each epoch we run on a batch (1000s of samples) and adjust the weights

- Our success is measured by the accuracy of our predictions

- This gap between training accuracy and test accuracy is overfitting: Overfitting is a central issue

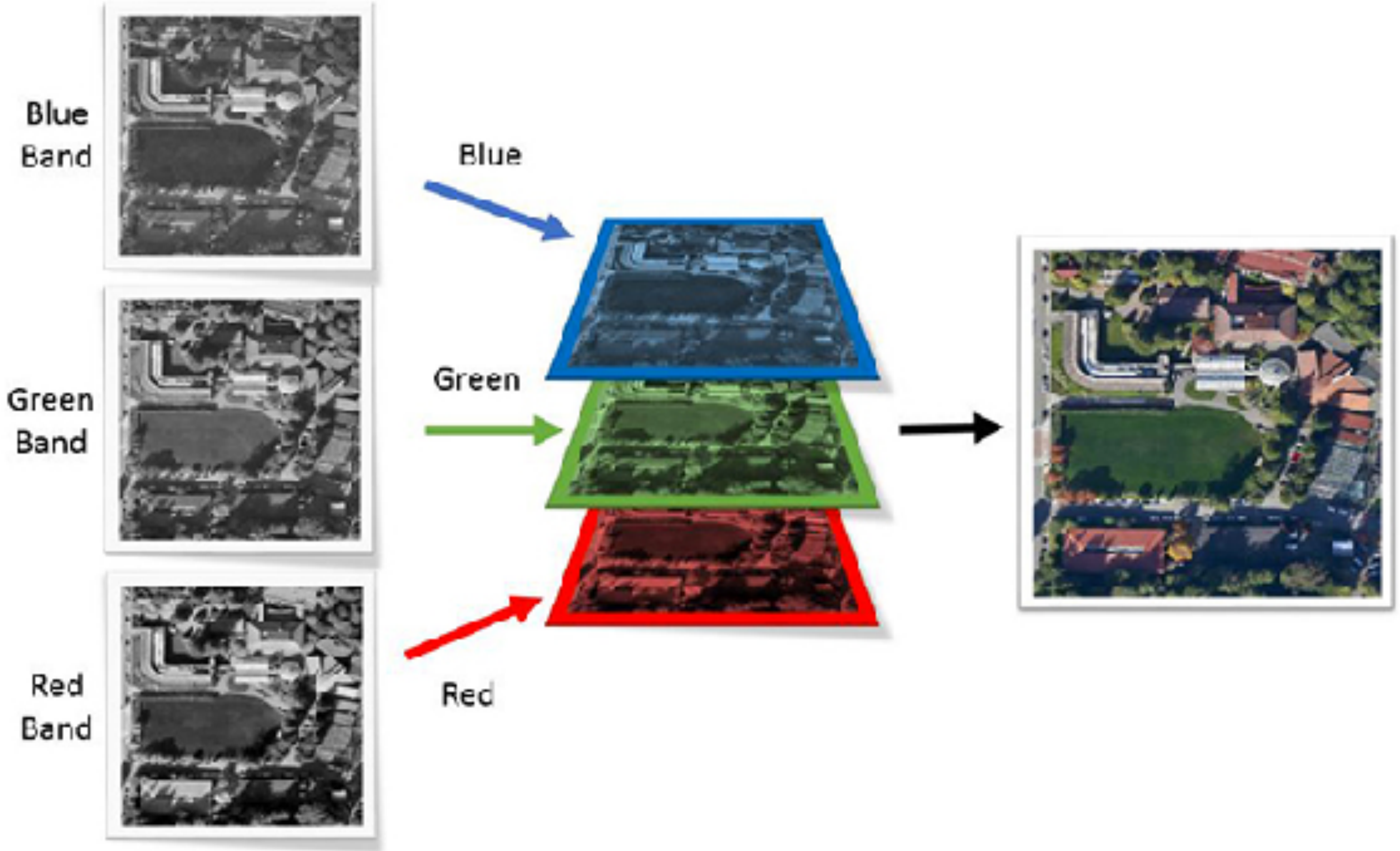
DEEP NN

- A shallow network (a few layers) can do the work but it might require an enormous number of units (neurons) per layer
- Deep Networks seem to suffer from the problem of vanishing gradient as you back propagate deeper and deeper
- Some argue that the most important innovation in deep learning applied to image processing is the CNN (Convolutional NN)

Convolutional NN
CNN

Vision

- Vision is an essential part of our lives
- How does a computer process an image…
- For a computer an image is coded pixels

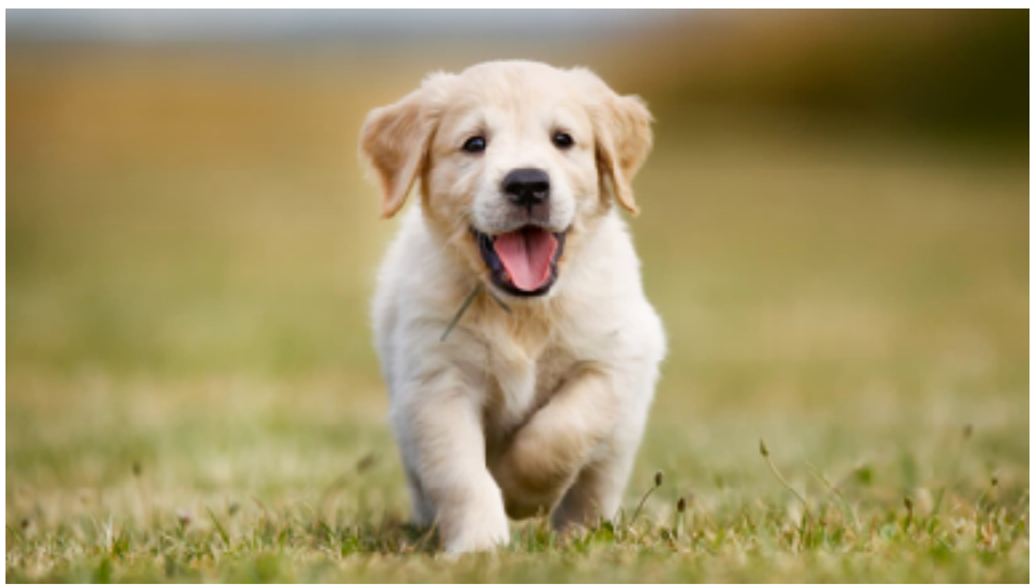


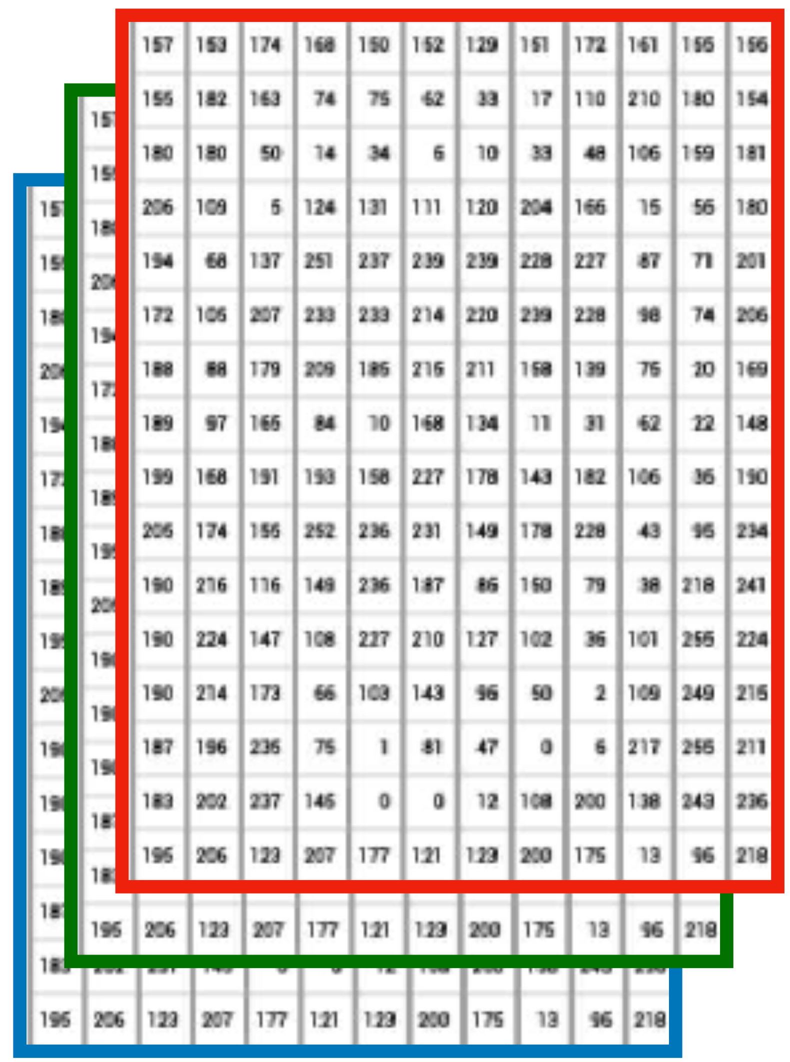
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	106	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
206	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

- An image is a matrix of numbers [0,255]

ref: MIT 6.S191 Introduction to Deep Learning 2021 Lecture 5

Classification





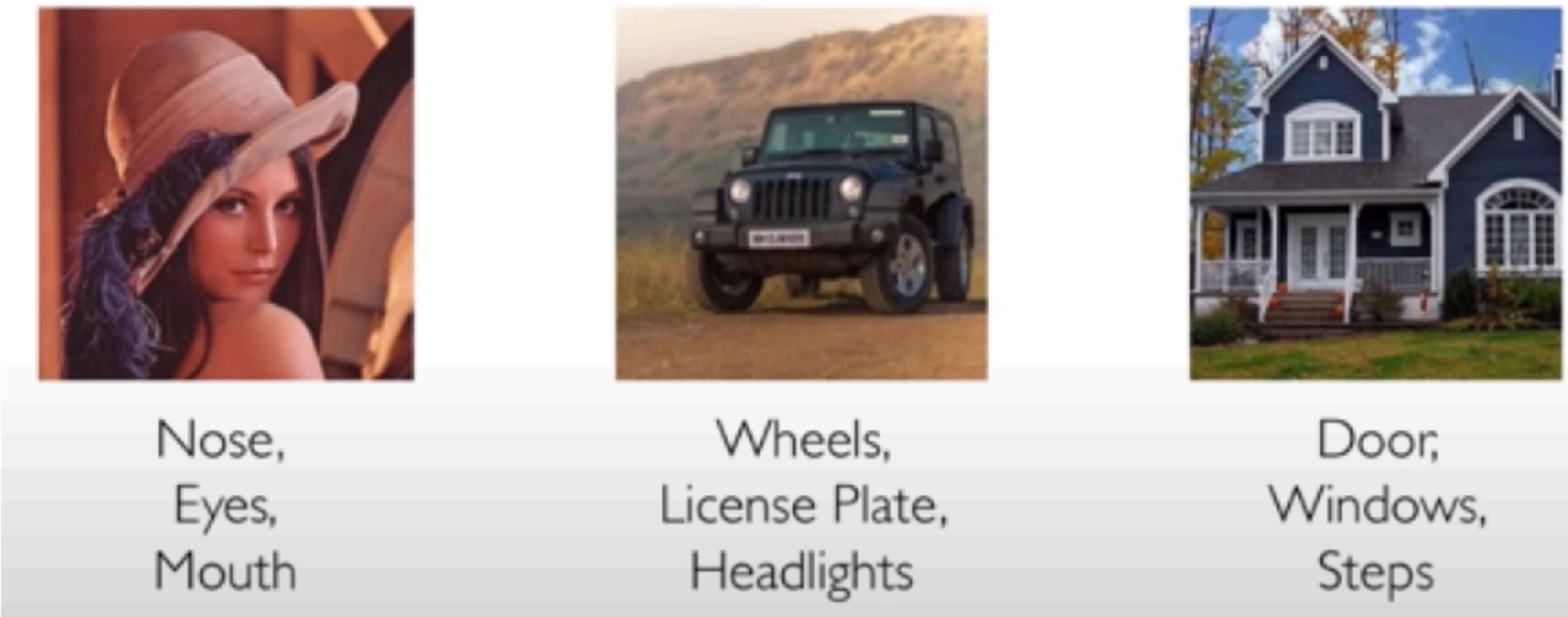
MLP Classification

$$\begin{array}{c} Cat \\ Dog \end{array} \begin{bmatrix} 0.2 \\ 0.8 \end{bmatrix}$$

High Level Features

- How do we do it?

- Detect specific characteristic features

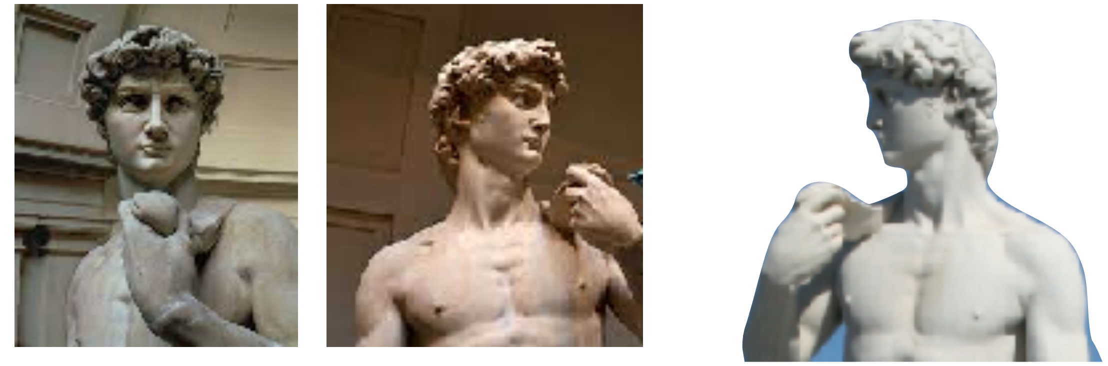


- We could tell the computer what are the features to look for but this is extremely difficult due to variations of images

ref: MIT 6.S191 Introduction to Deep Learning 2021 Lecture 5

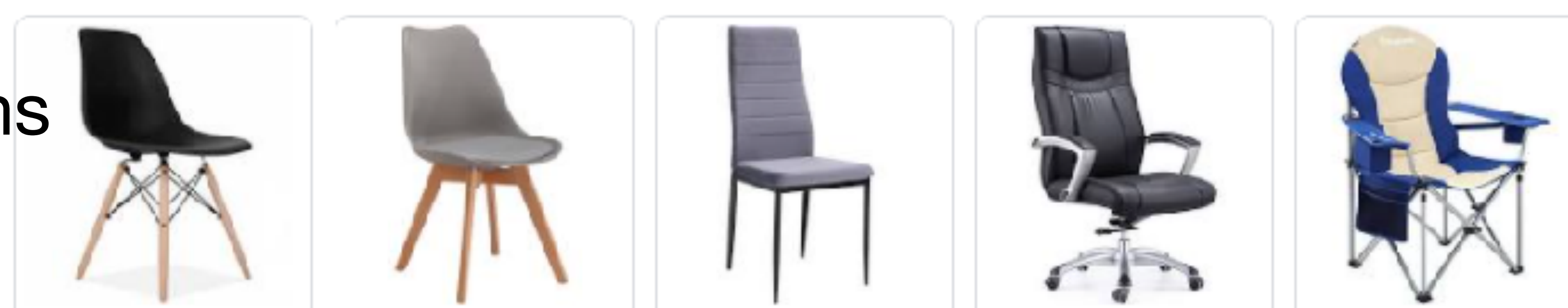
Image Variations

- Different angle, rotations, translations



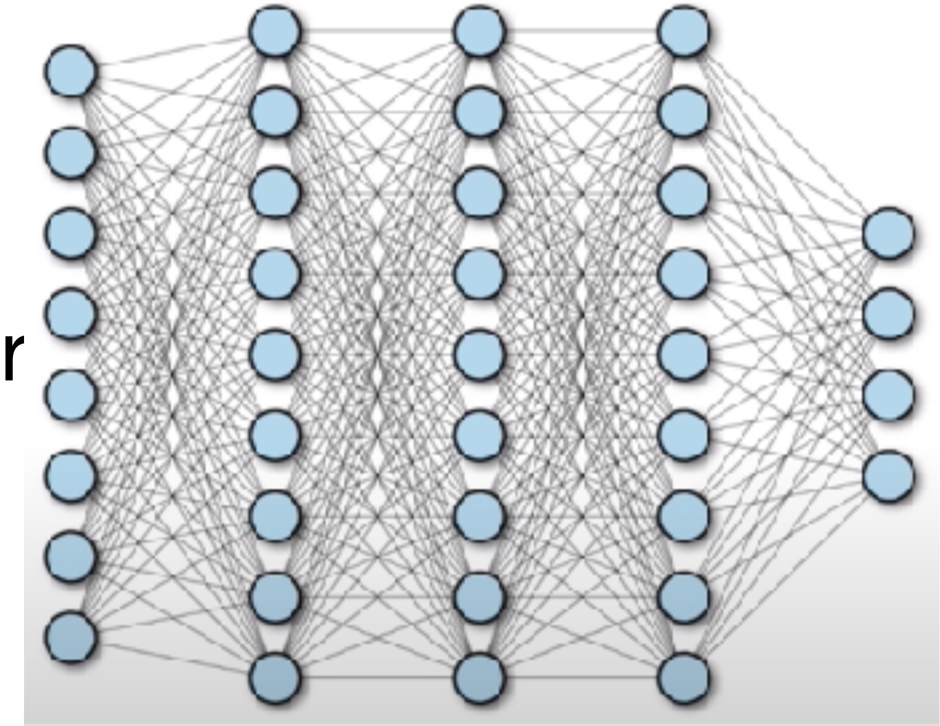
- Scale

- Intra class variations



Learning Features, How?

- We are looking for a way to extract features automatically in an hierarchical fashion.
- We want to learn features directly from DATA, without pre define the features…
- NN allow to do the above….
- But will a fully connected NN sufficient?
- The 2D image will be mapped to a vector
  All spatial information will be LOST!
- How can we use the spatial information and introduce scale and translation invariance?

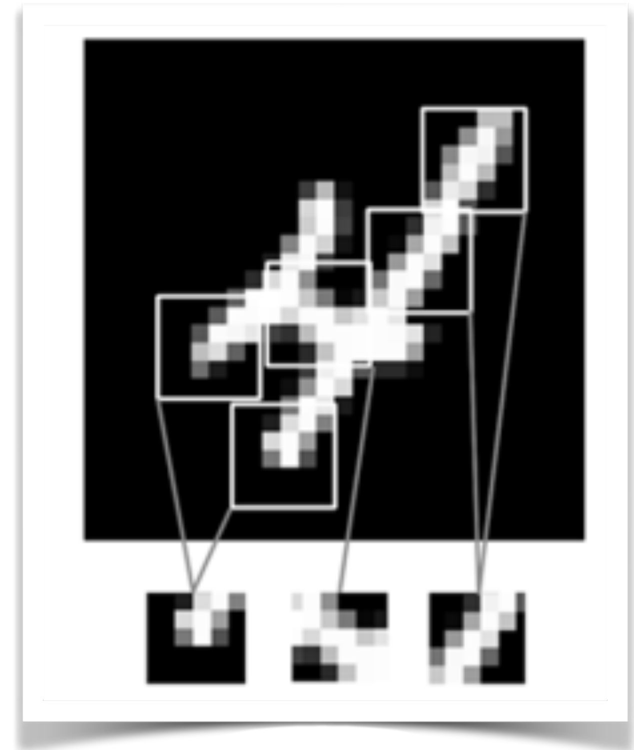


A CNN

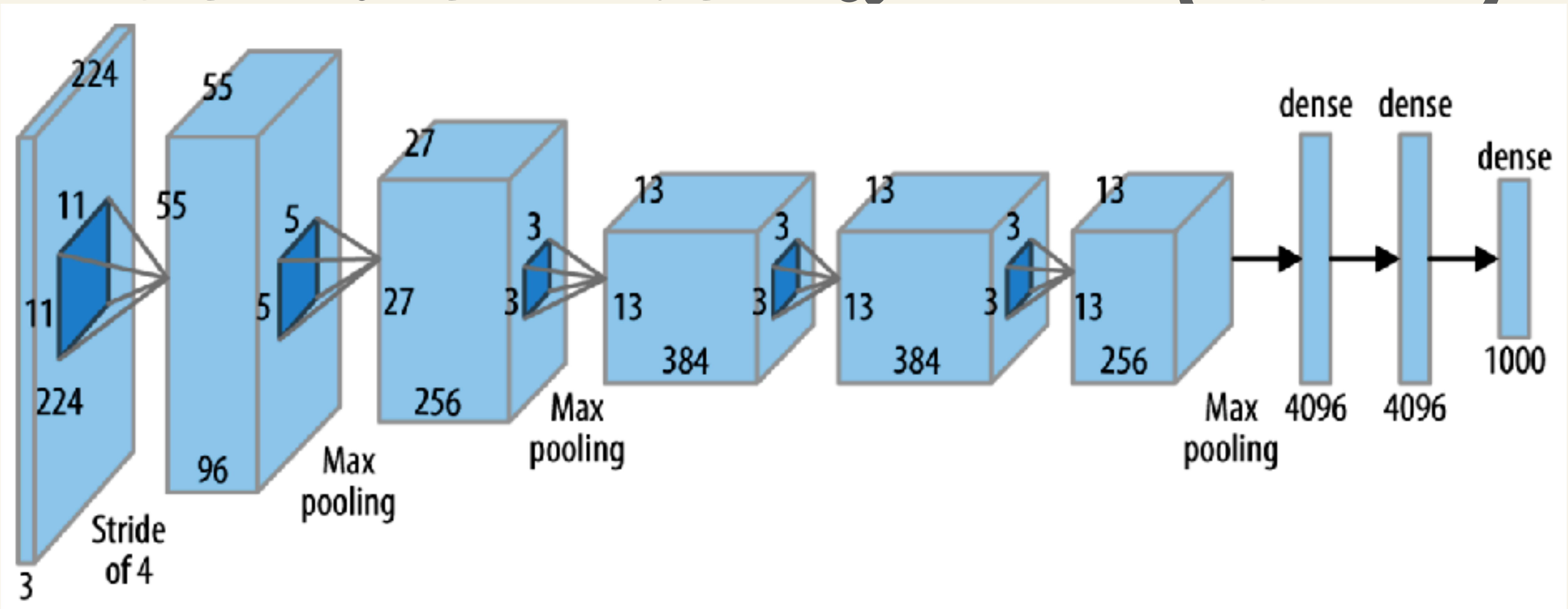
- CNN (Convolutional Neural Network - CONVNET) is made of layers that preserve the spatial characteristics of an image

- Dense Layers (CL) learn global patterns, CNN learn local patterns

- It is translational invariant, highly data efficient on perceptual problems



Convolutional NN (CNN)



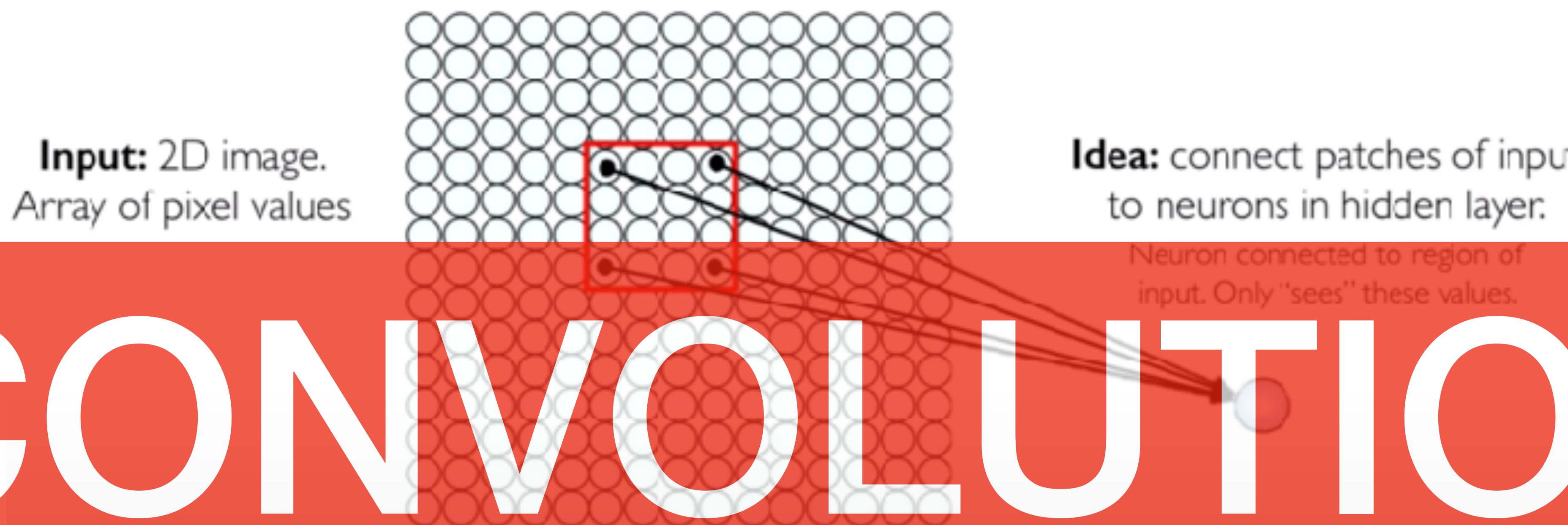
- A Fully Connected dense layers NN learn global patterns, CNN learn local patterns

- CNN (Convolutional Neural Network - CONVNET) is made of layers that preserve the spatial characteristics of an image

- CNN is based on SHARED WEIGHTS, which reduces the dimensionality of the problem and introduces translation invariance, highly data efficient on perceptual problems

Visual Reception Field

- A neuron in the hidden layer sees only a patch of the image:
  Number of weights is reduced, and the spatial relation between pixels is kept.



CONVOLUTION

- We apply a filte … filter size NxN
  carries N^2 weig…
- We apply multip … tures
- The same filter is used via a sliding window all over the image, weights are shared, so it does not care where in the image a feature appears…. (invariance)

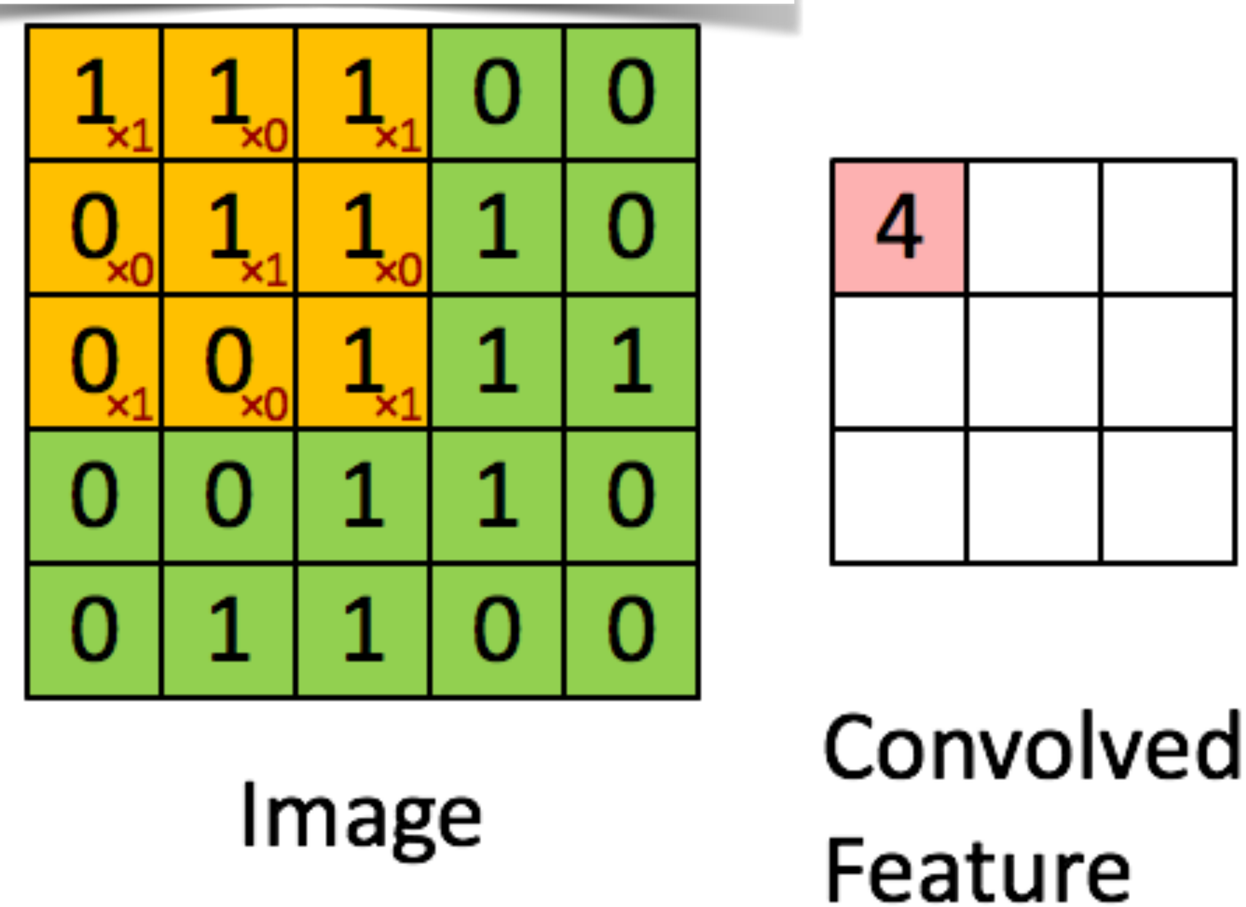
Convolution is able to preserve the spatial relationship between pixels by learning image features in local square areas if the image

Introducing Convolution

Image

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

$$(f * g)(x) = \sum_{u=-\infty}^{+\infty} f(u)g(x-u)$$



Kernel

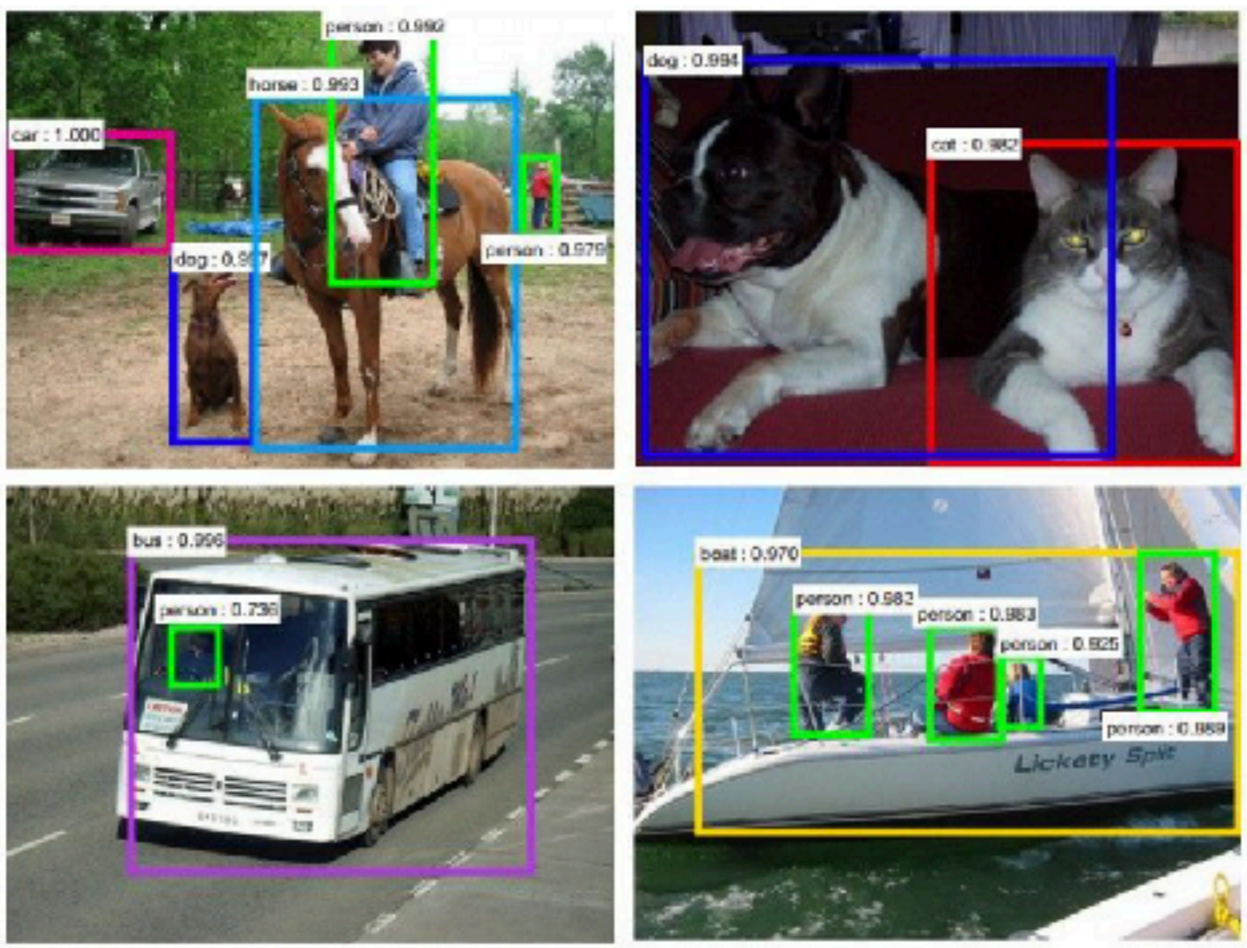
1	0	1
0	1	0
1	0	1

Kernel is a feature detector of the input layer
Kernel is also called a **filter**

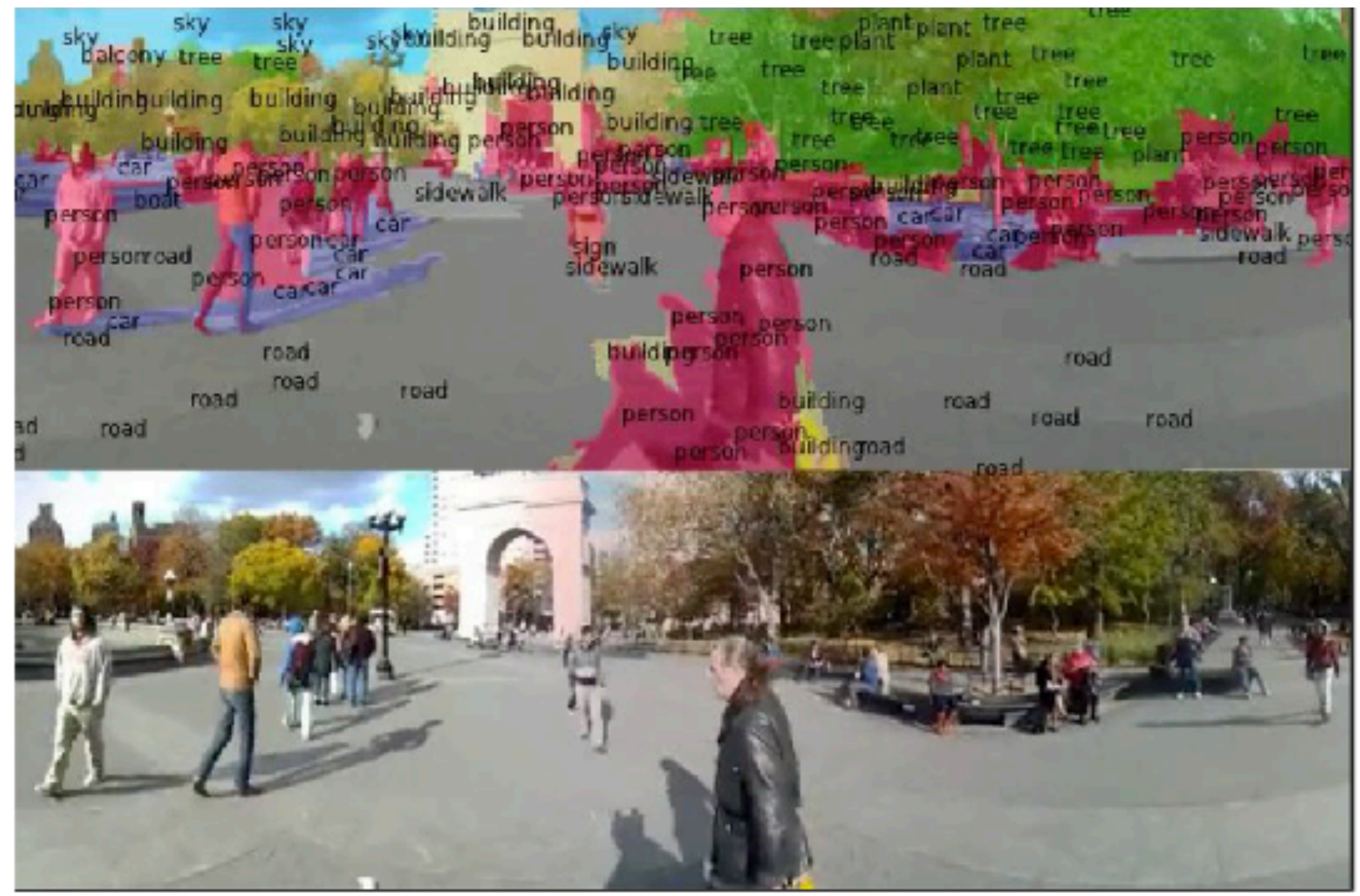
The convolved image is also called
a **Feature Map** or
Activation Map

What are CNNs good for

- Detection (Self Driving Cars) & Segmentation (pixel by pixel probability for objects)



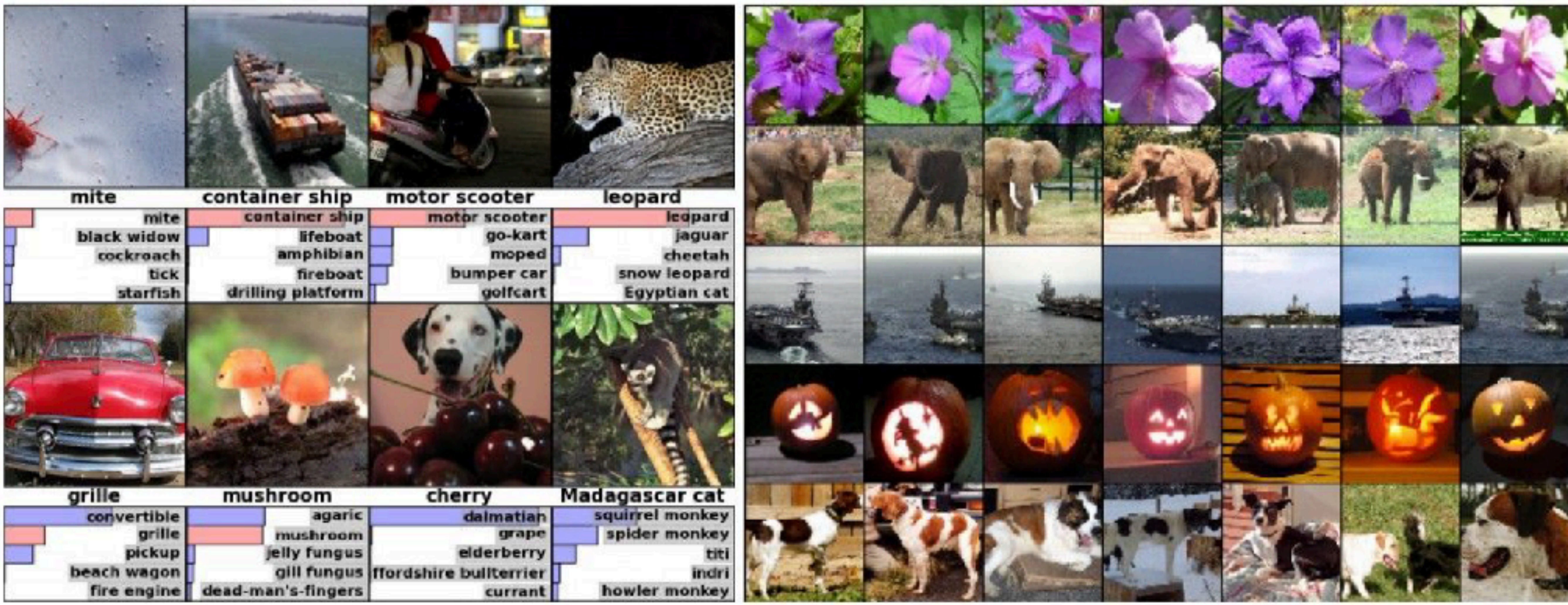
[Faster R-CNN: Ren, He, Girshick, Sun 2015]



[Farabet et al., 2012]

What are CNNs good for

- Classification & Retrieval (Similarity Matching, Google Images)



What are CNNs good for

- Face Recognition

What are CNNs good for

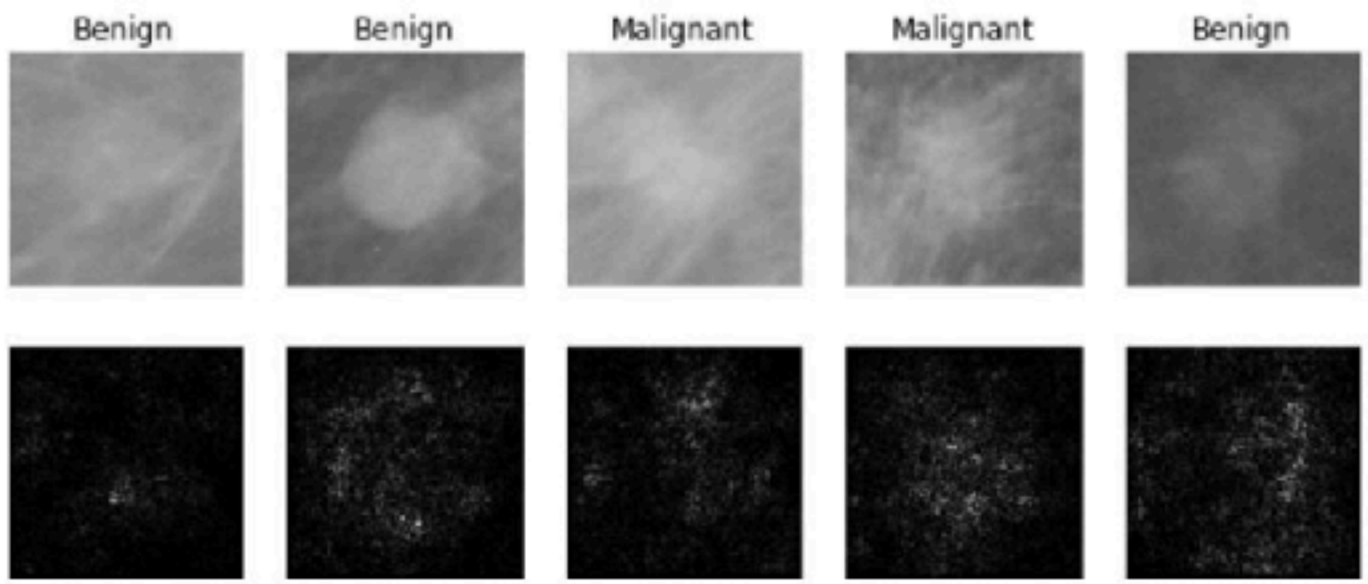
- Pose Recognition

Images are examples of pose estimation, not actually from Toshev & Szegedy 2014. Copyright Lane McIntosh.

[Toshev, Szegedy 2014]

What are CNNs good for

- Medical Images, Street Sign Recognition, Classification of Galaxies



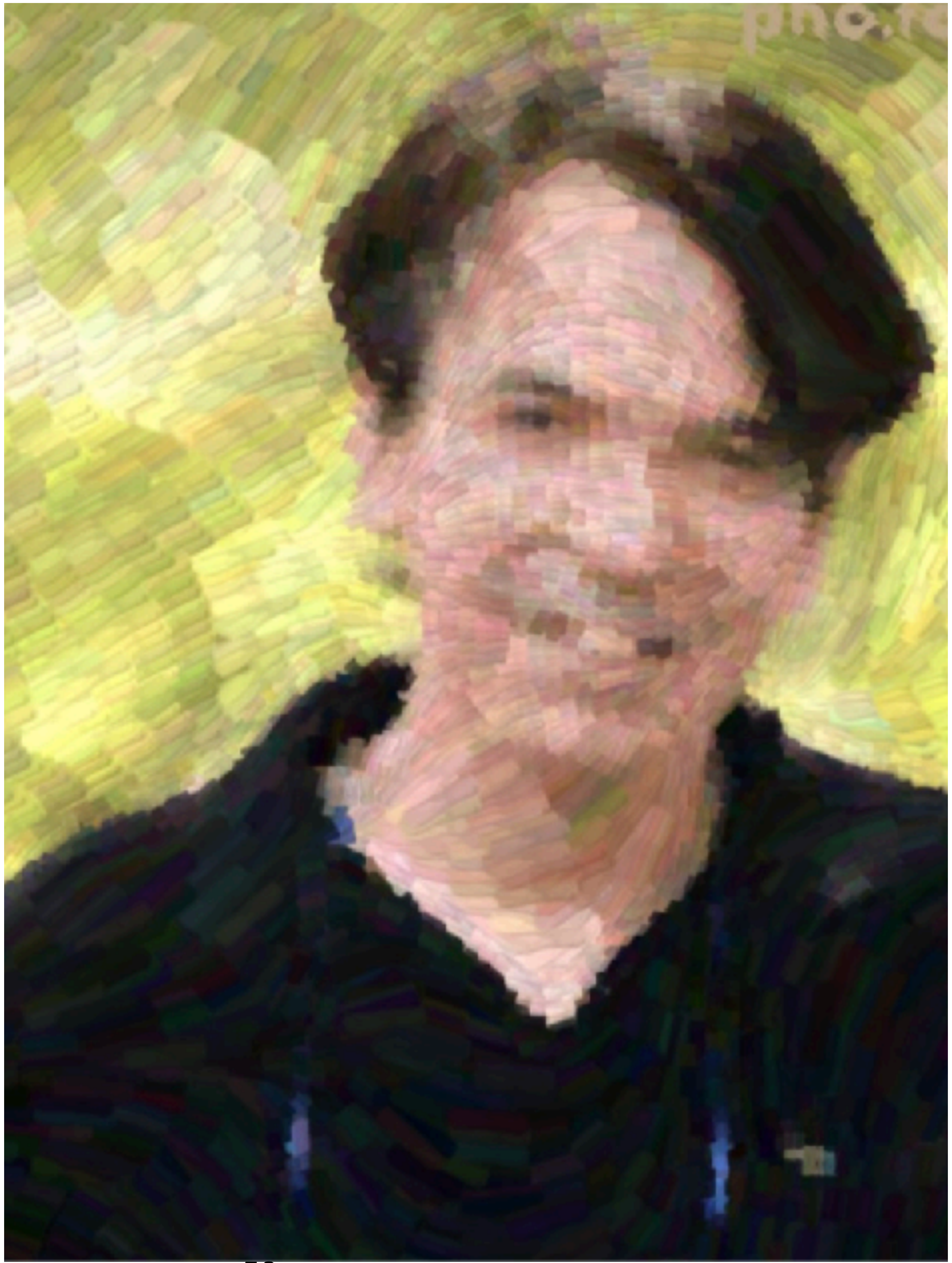
[Levy et al. 2016]

[Dieleman et al. 2014]

[Sermanet et al. 2011]
[Ciresan et al.]

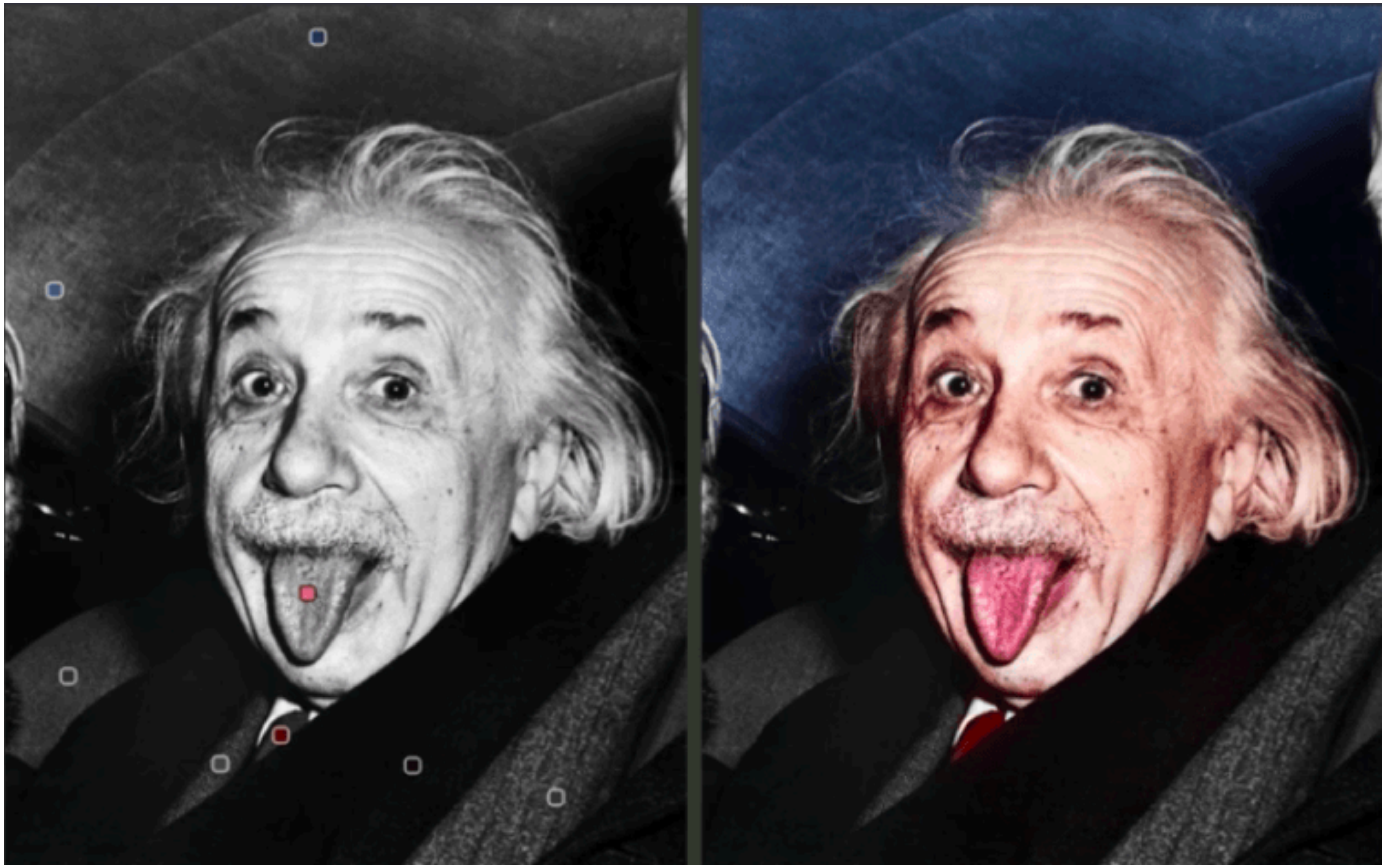
What are CNNs good for

- Rendering Images
  Van Gogh style



What are CNNs good for

- Colorizing BW Images



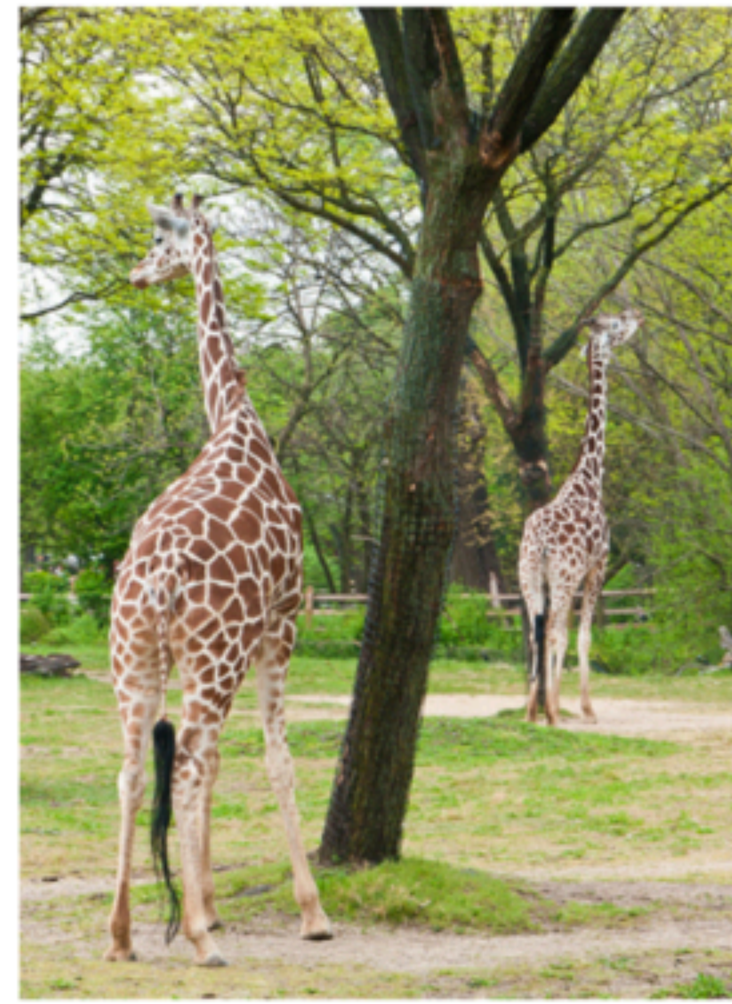
Richard Zhang, Adobe Research

What are CNNs good for

- Let the computer recognize scenes and suggest a relevant caption

a soccer player is kicking a soccer ball

a street sign on a pole in front of a building


a couple of giraffe standing next to each other

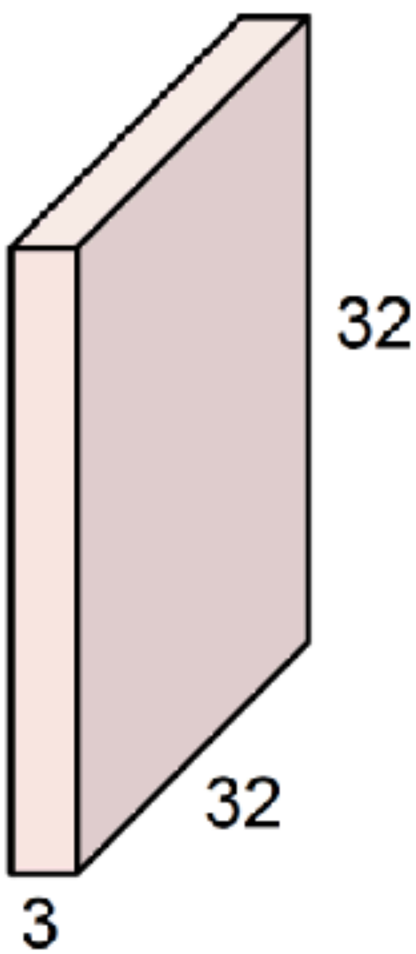
Here we want to preserve spatial structure

Convolution Layer

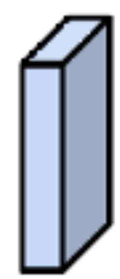
A set of learnable filters that produce activation maps

Filters always extend the full depth of the input volume

32x32x3 image



5x5x3 filter



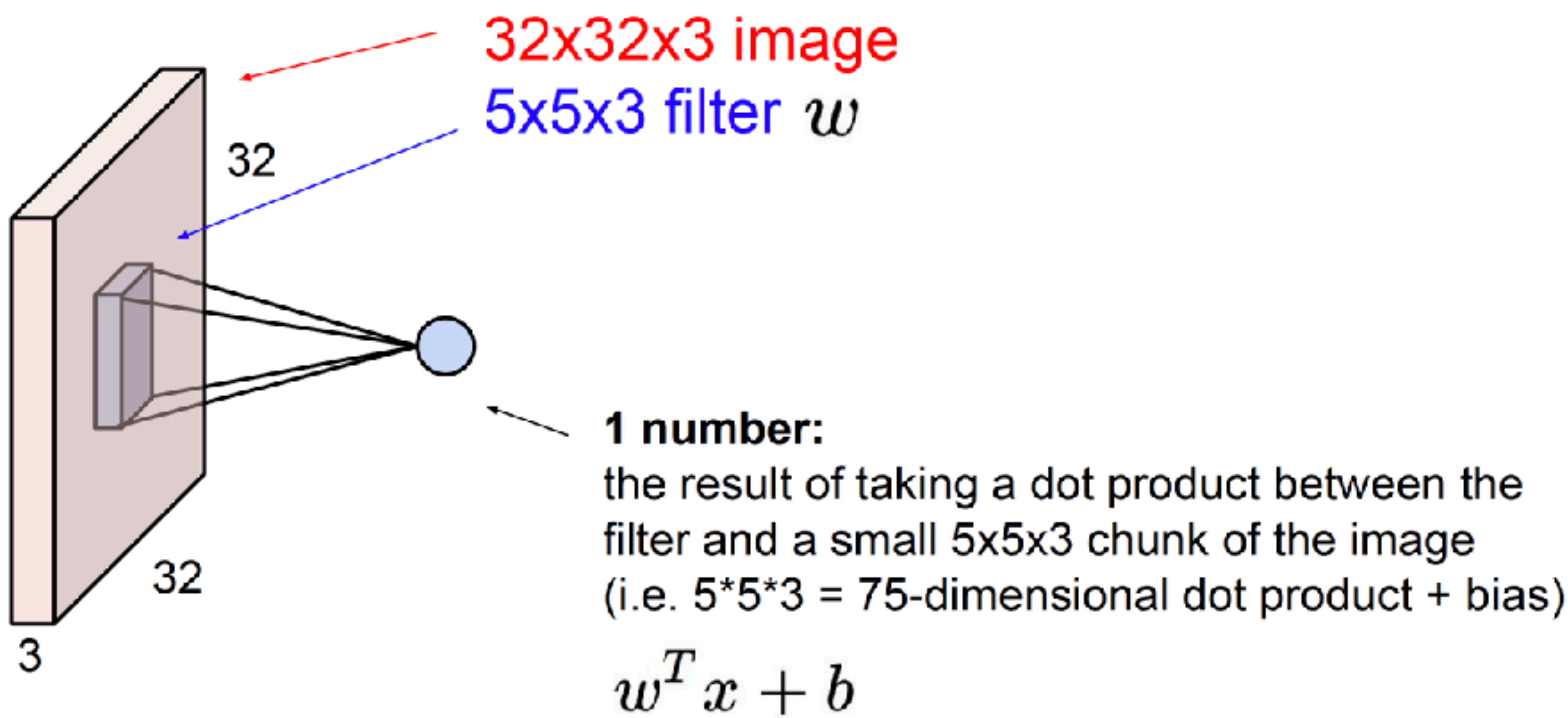
Convolve the filter with the image i.e. "slide over the image spatially, computing dot products"

We can put as many filters as we like
The number of filters define the DEPTH of the resulting layer

CNN

- The **weights are shared**, and a feature map is represented by 5x5x3+1=76 parameters

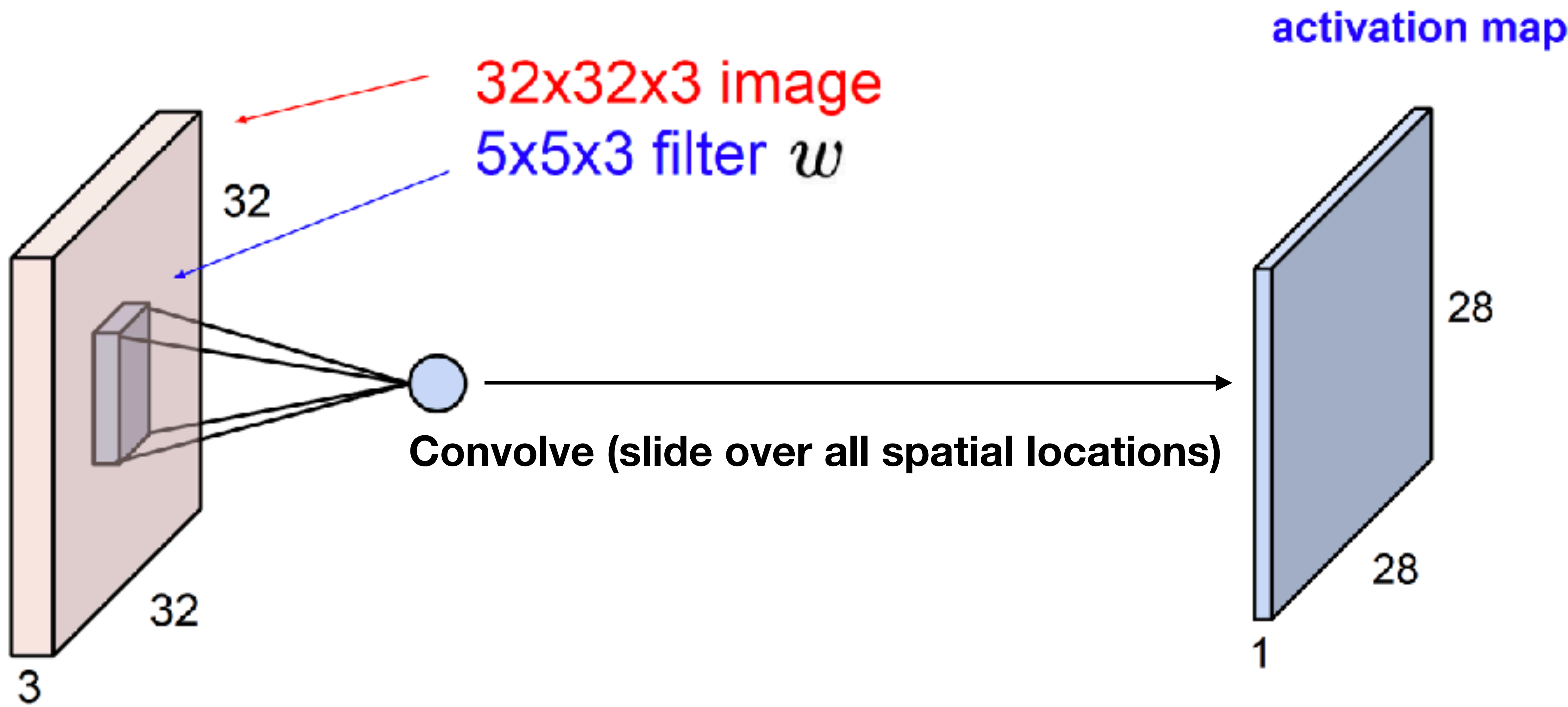
Convolution Layer



CNN

- The **weights are shared**, and a feature map is represented by 5x5x3+1=76 parameters

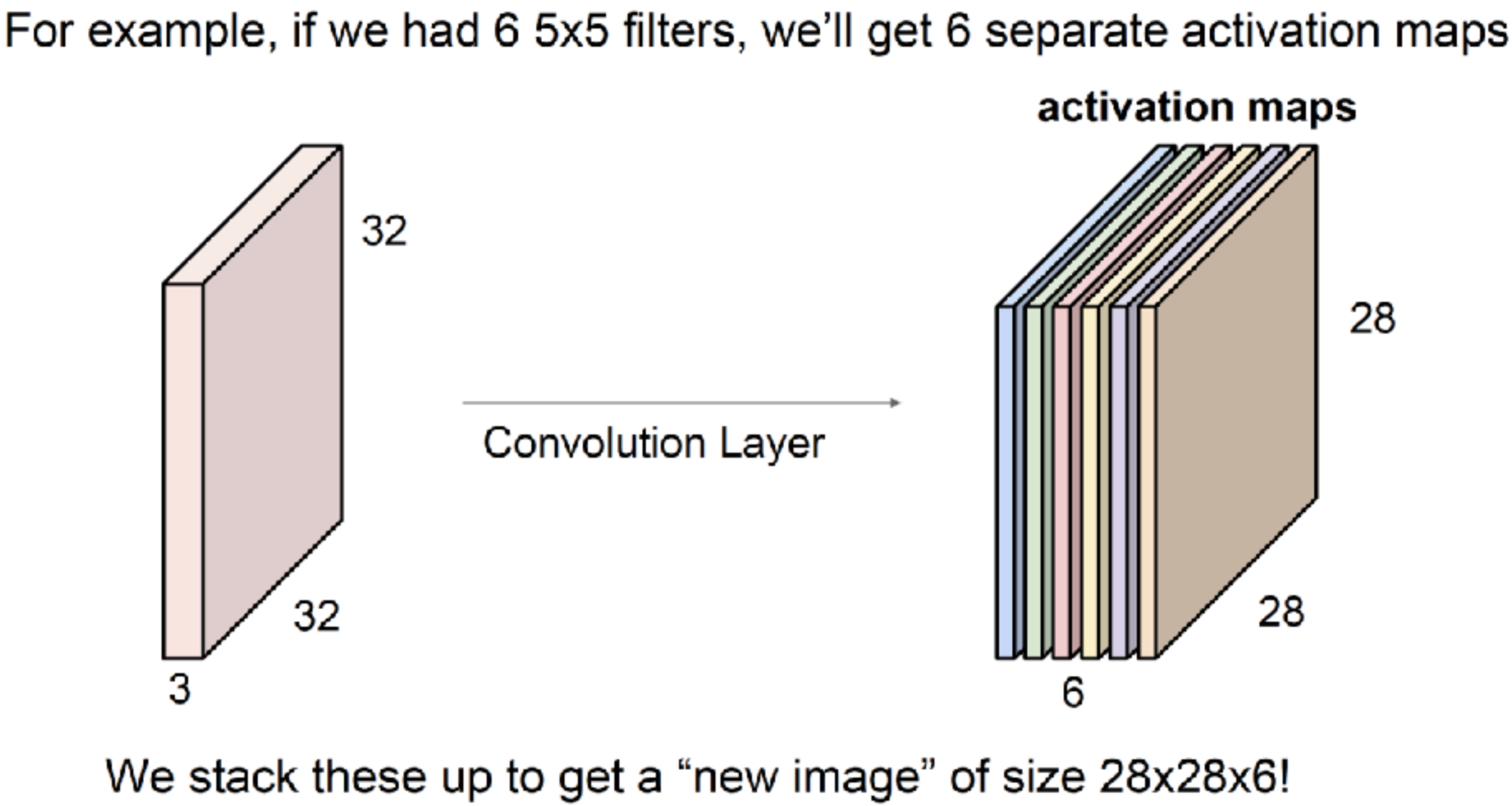
Convolution Layer



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

CNN

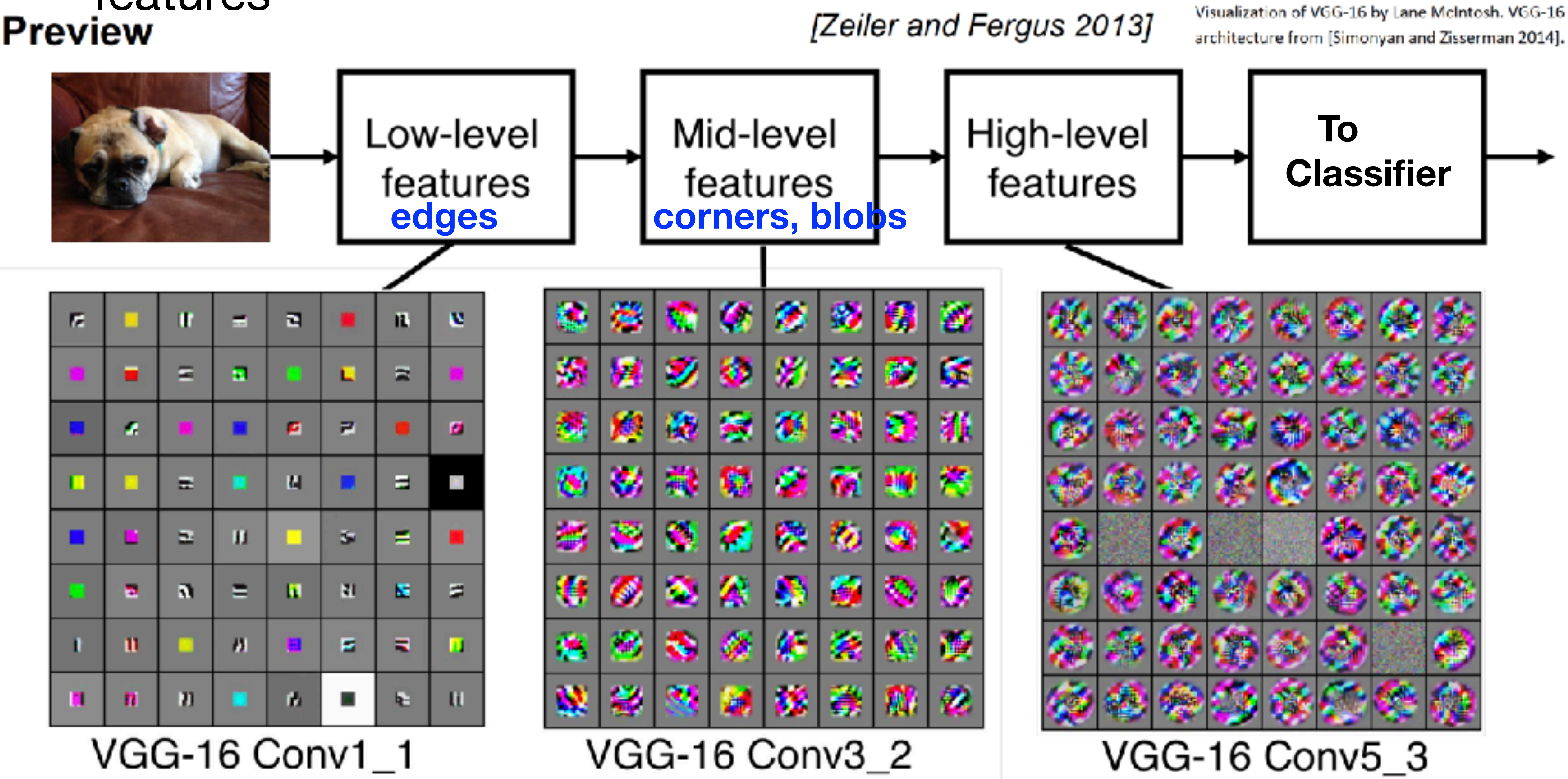
- Stacking activation maps , each learns different features



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

CNN

- As we go deeper in the net we strat to probe higher level features



Unsupervised Learning
Auto Encoders

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x -> y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.

Unsupervised Learning

Data: x
Just data, no labels!

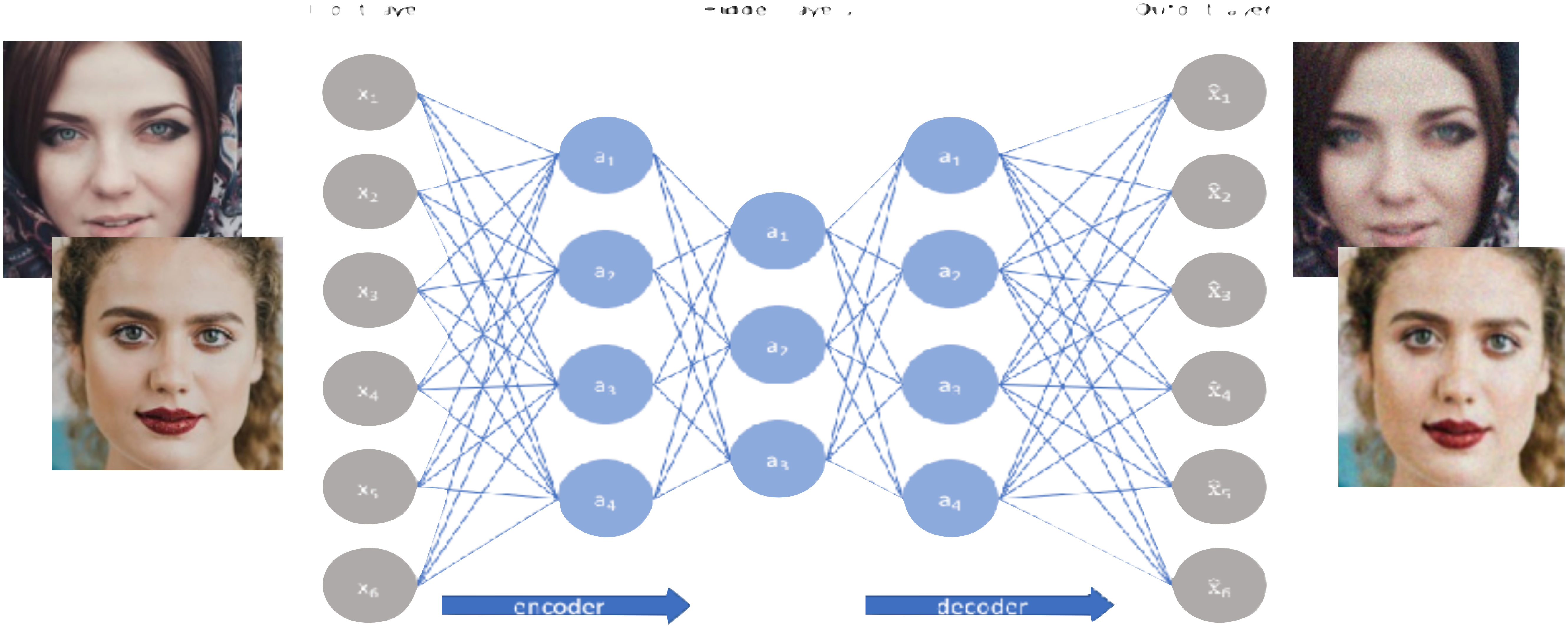
Goal: Learn some underlying hidden *structure* of the data

- No need for annotation

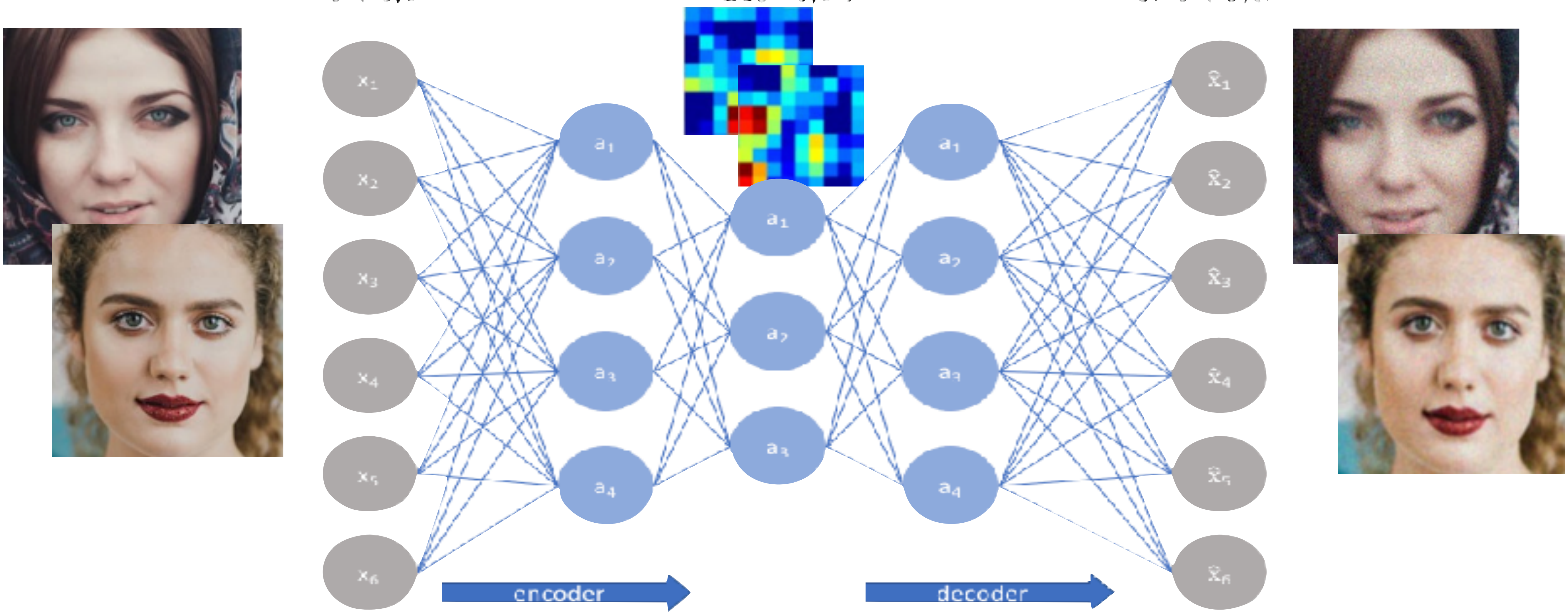
Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

- If we manage to understand the underlying features of our DATA , it’s a huge step towards understanding the visual world around us

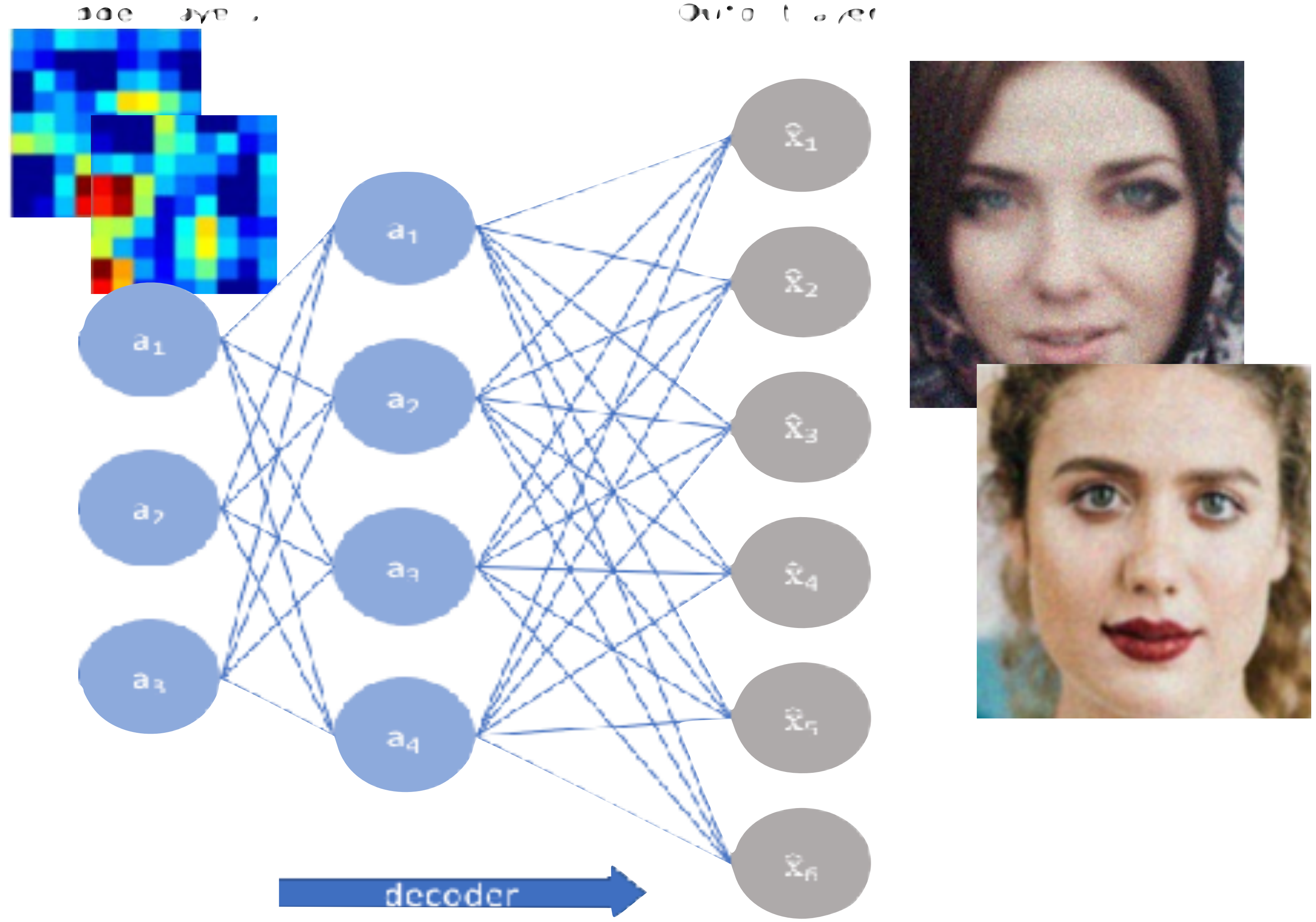
AutoEncoder



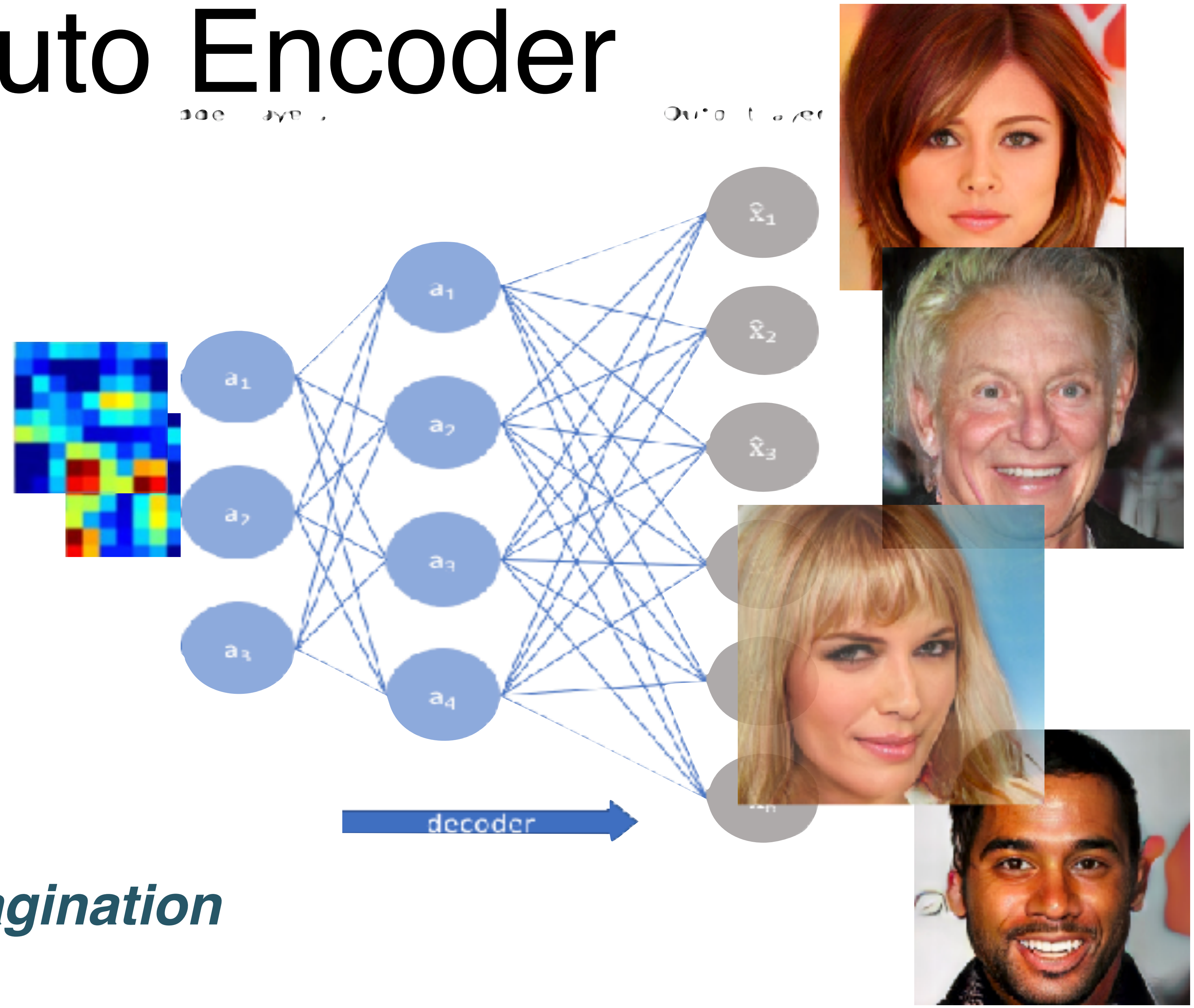
AutoEncoder



AutoEncoder



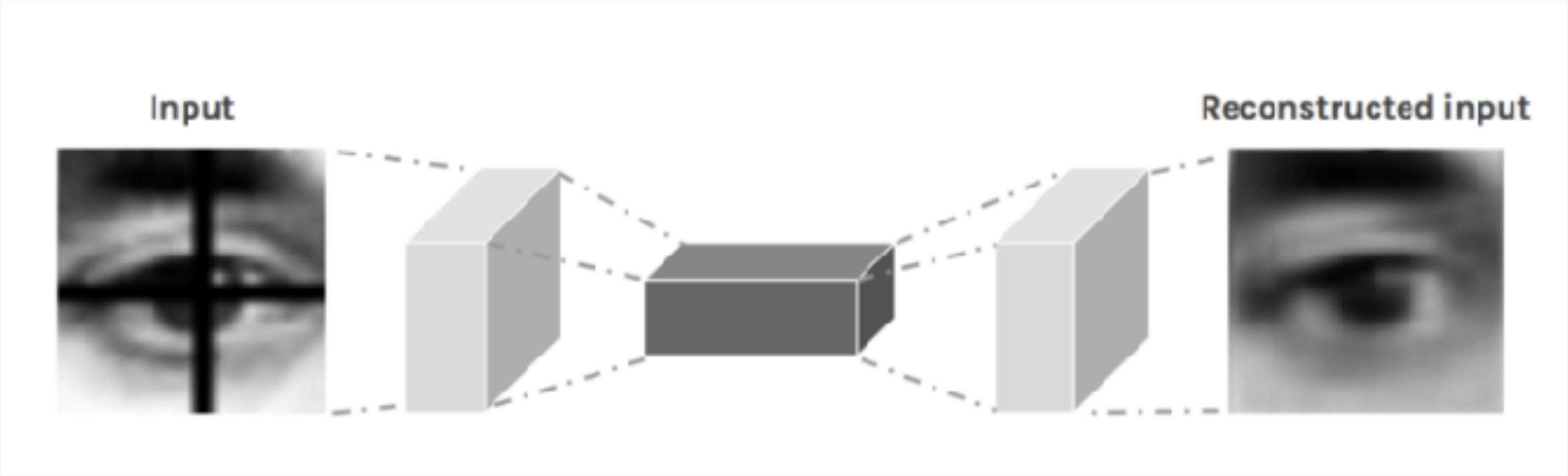
Auto Encoder



Definition of *imagination*

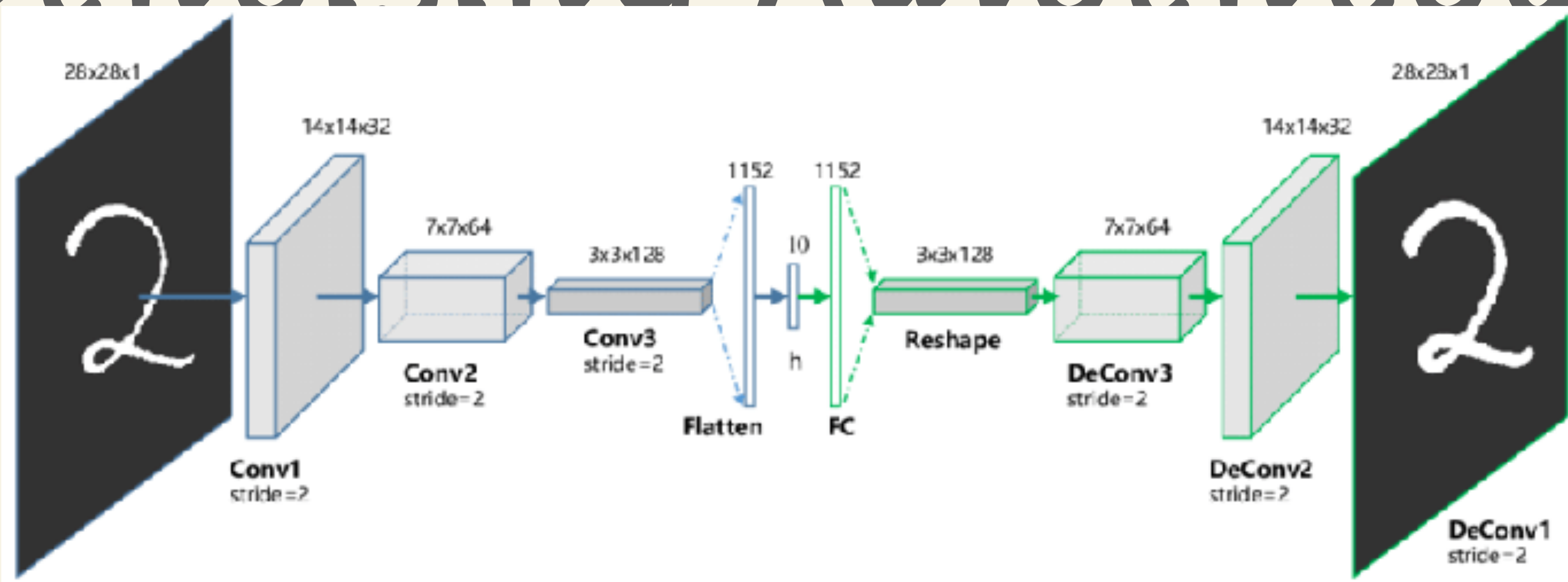
the act or power of forming a mental image of something not present to the senses or never before wholly perceived in reality

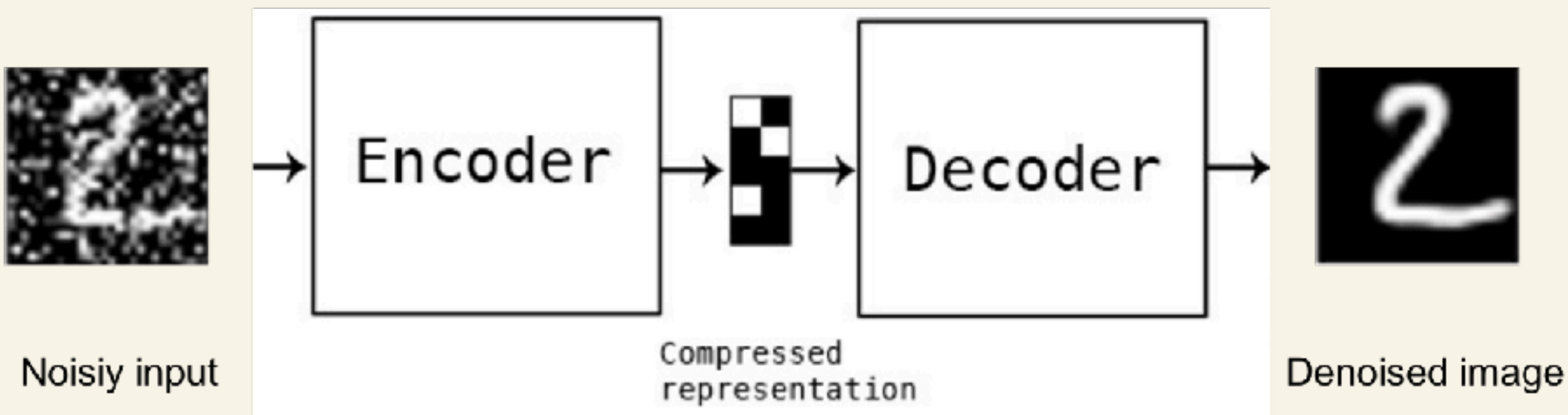
Traditional Autoencoders



https://towardsdatascience.com/autoencoders-introduction-and-implementation-3f40483b0a85

Denoising Autoencoder



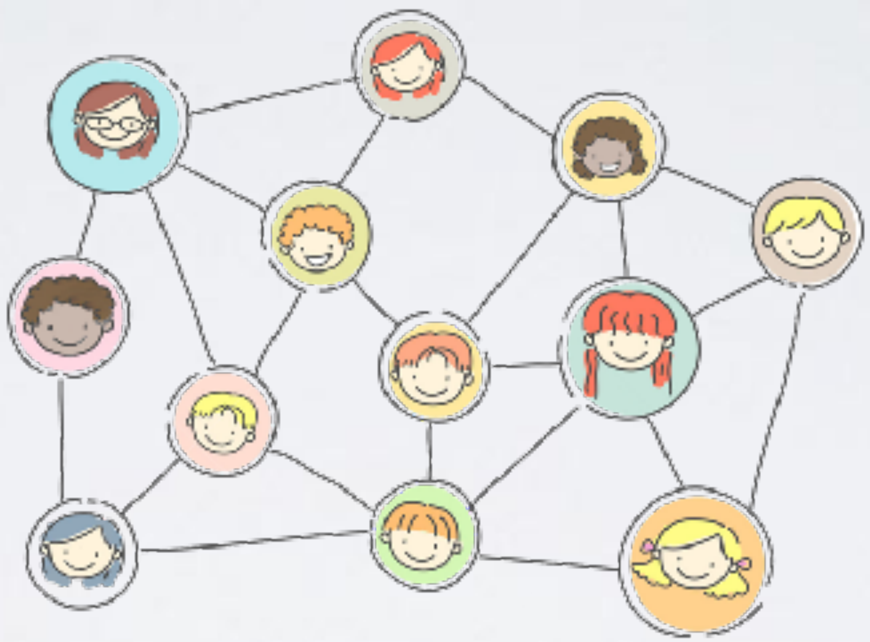


The feature we want to
extract from the image

Graph Neural Nets

Data can often be represented as a **graph**

-



https://edorado93.github.io/2017/12/10/Deep-Dive-Into-Graph-Traversals-227a90c6a261/

Social networks, molecules, planets in a solar system, particles in a gas, road networks, computer networks, covid-19 patients, etc.

a **graph**

It's a data structure. It's made from nodes and edges. Nodes and edges have "features" or "attributes"

The order of the nodes/edges in this table **is arbitrary**

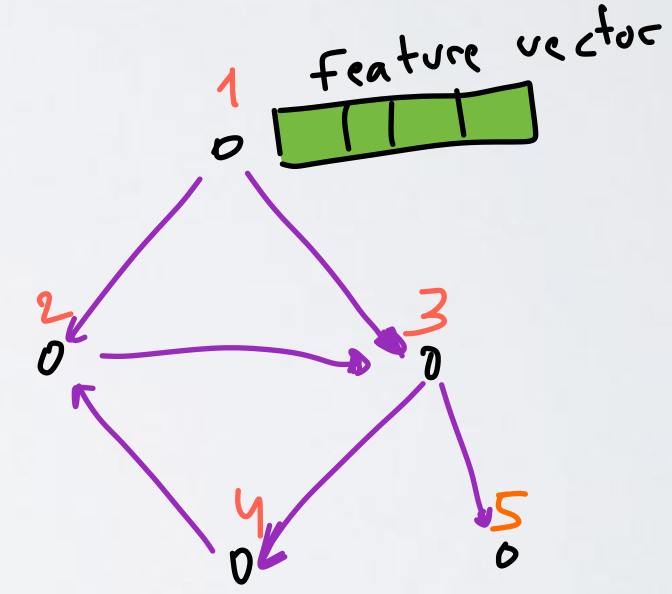
nodes

index	features
1	
2	
3	
4	
5	

edges

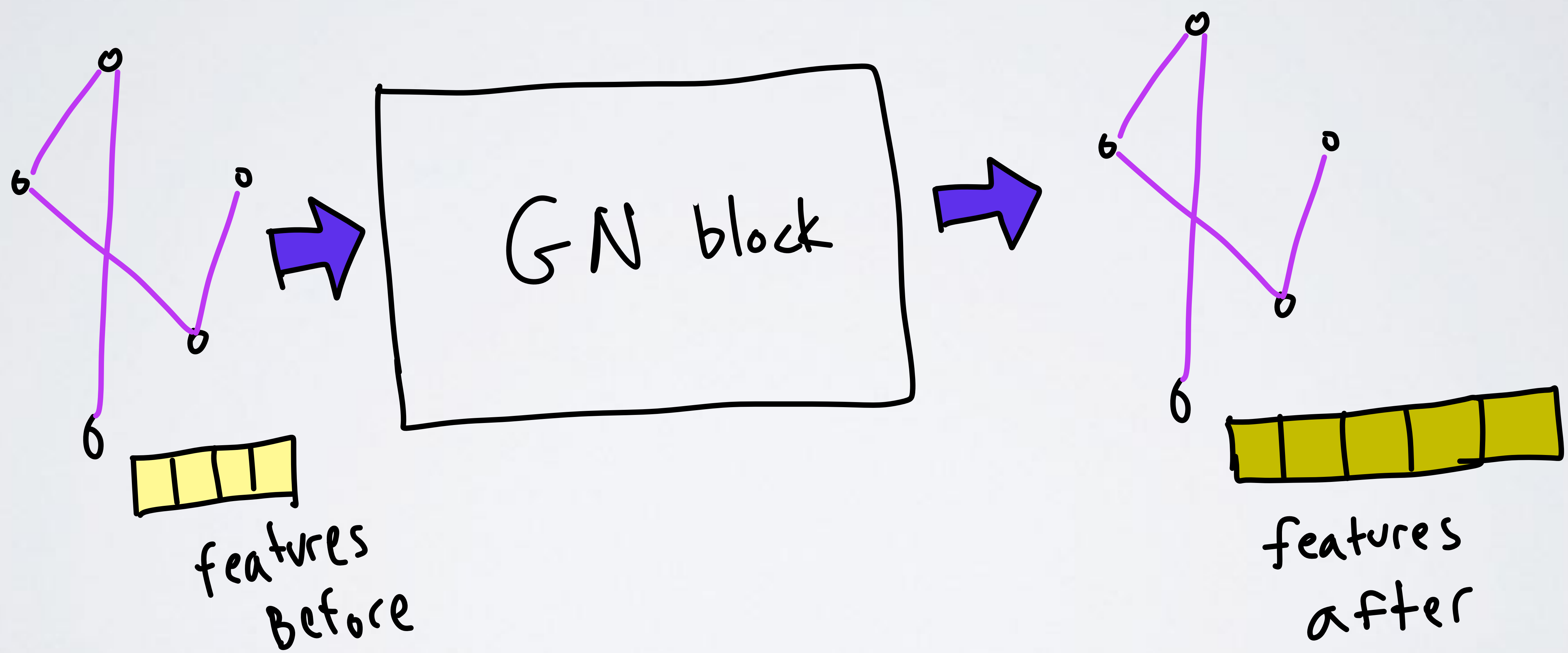
index	start/end	features
1	3 → 4	
2	1 → 2	
3	4 → 2	
4	1 → 3	
5	2 → 3	
6	3 → 5	

its easy for us to visualise

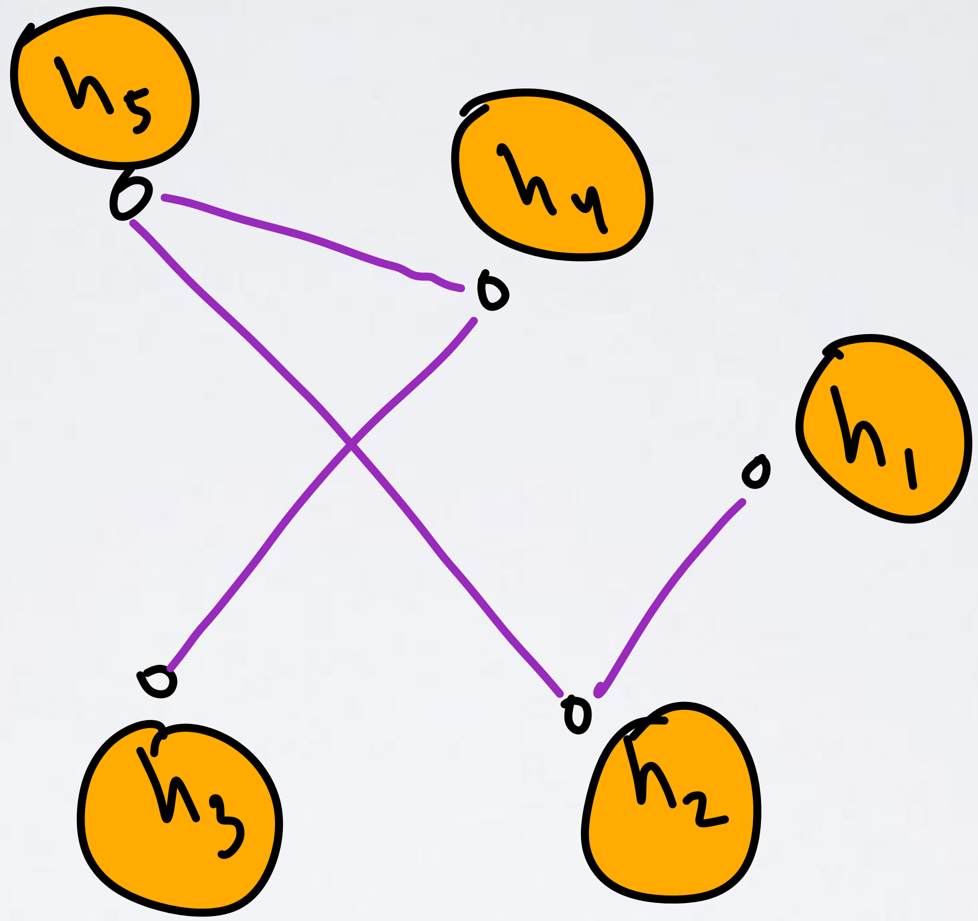


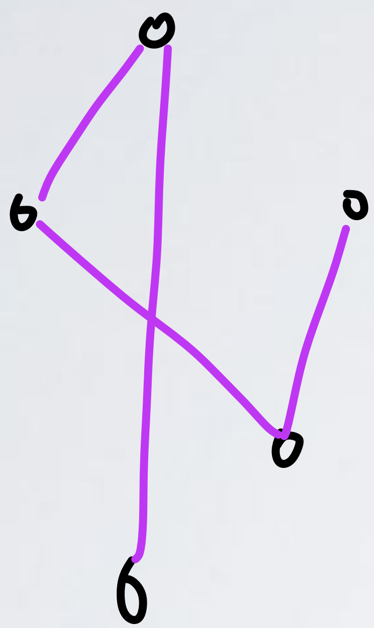
the "GN block"

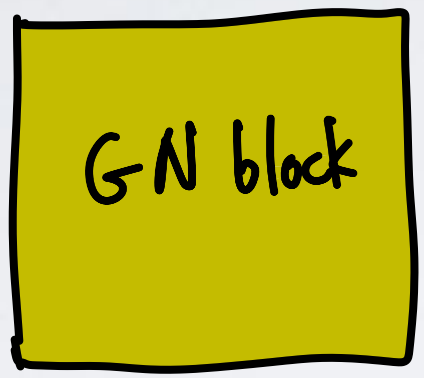
it updates the node/edge/graph features



In the GNN, there is a hidden state on each node - and we want to “update” it so it contains information from the rest of the graph







inside:

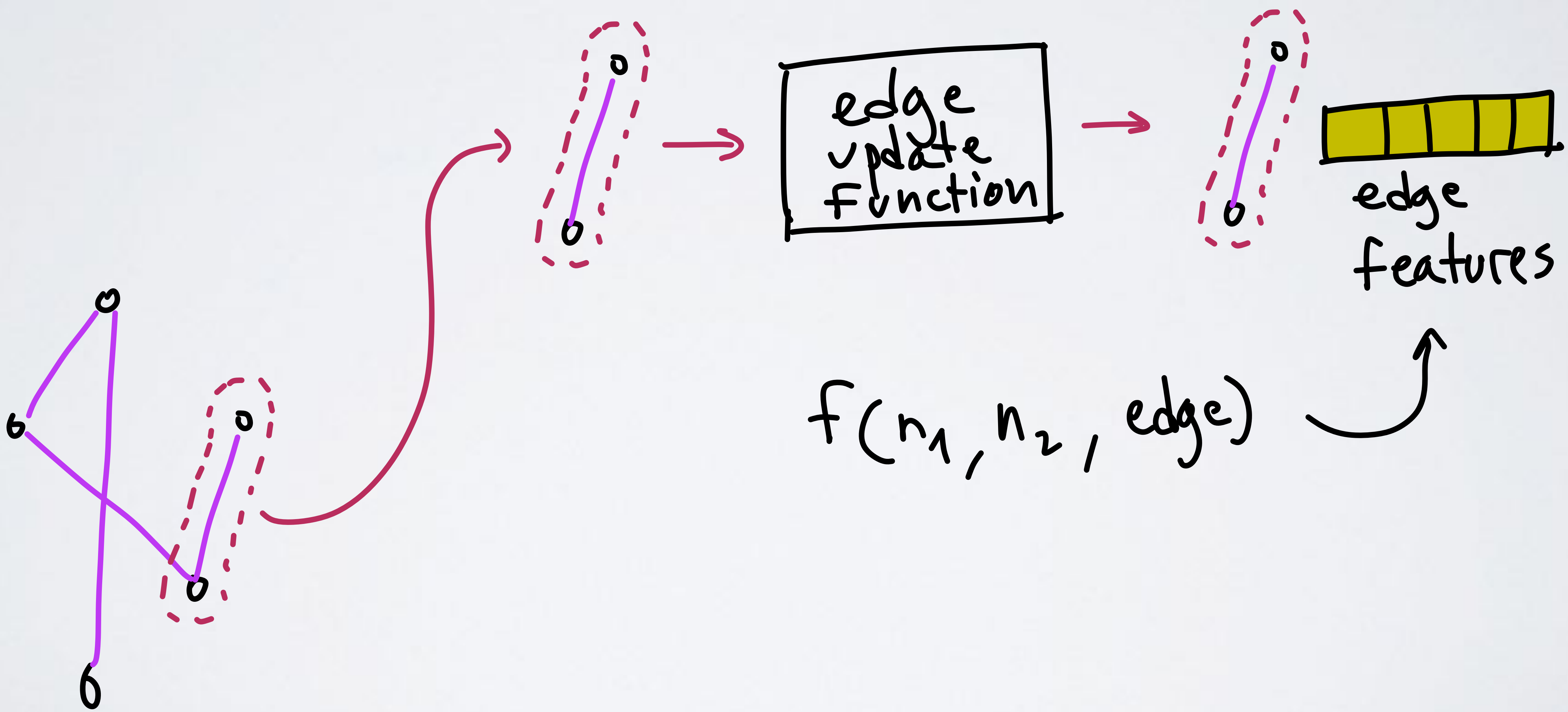
- edge update function
- node update function
- graph update function

These functions can be anything we want - usually involving neural networks

GN block

inside:

- edge update function

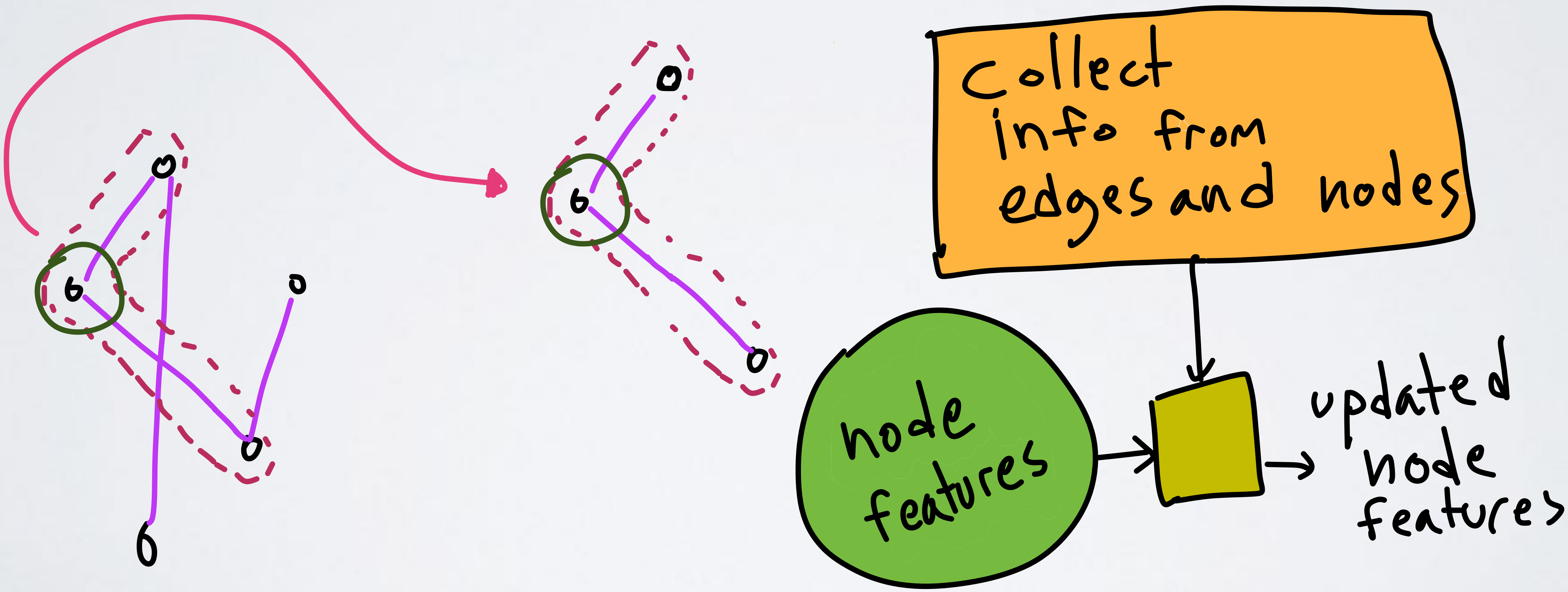


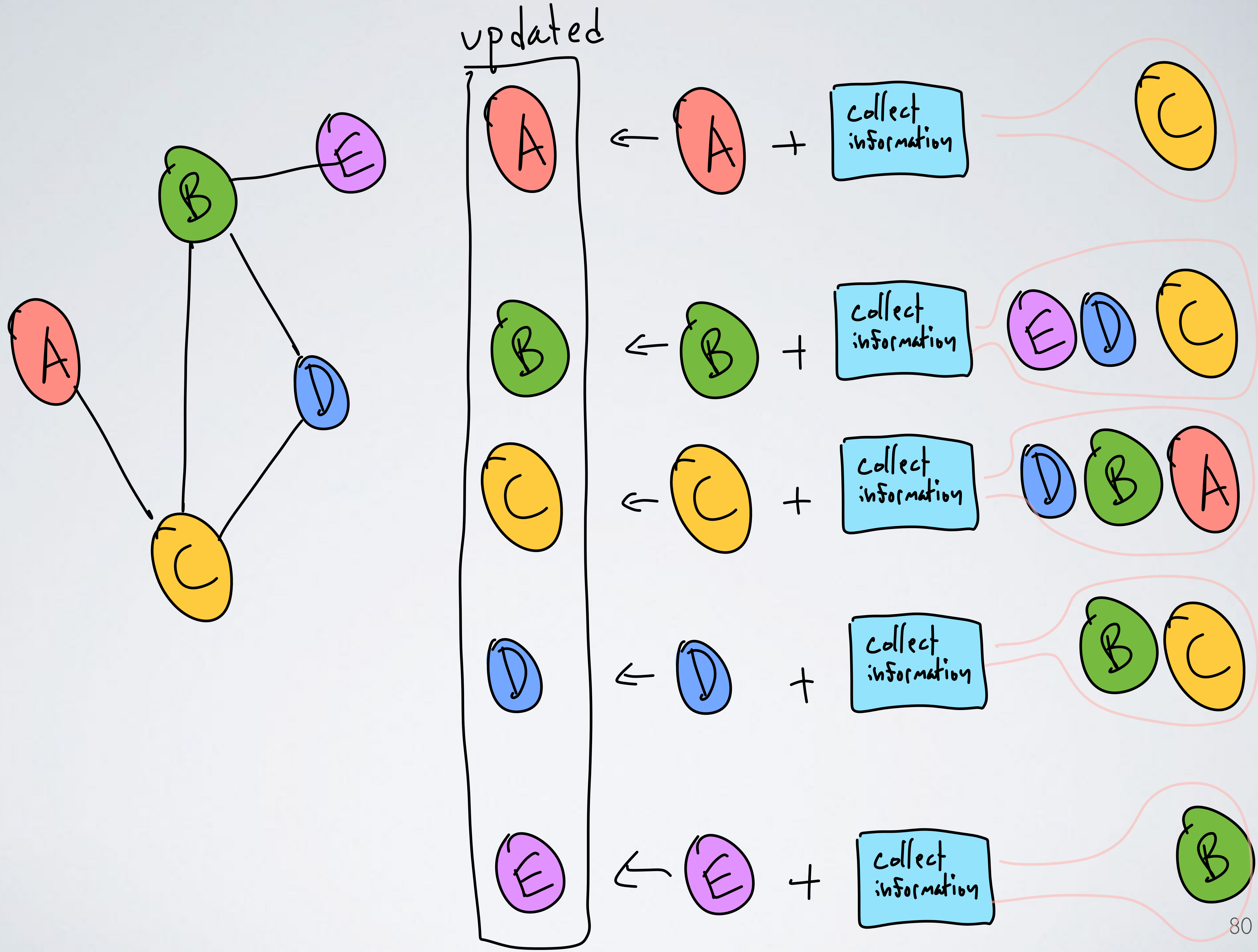
$$f(n_1, n_2, \text{edge})$$

GN block

inside:

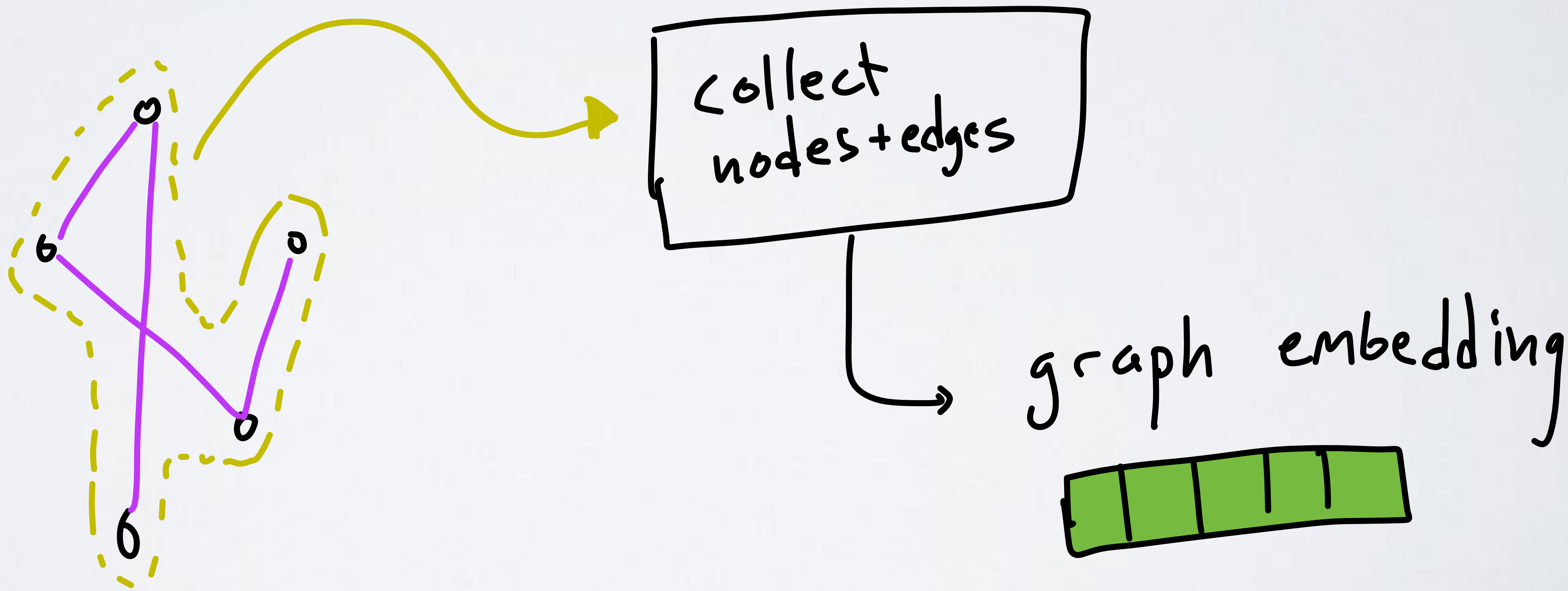
- edge update function
- node update function




updated
A
B
C
D
E
collect information
+

GN block

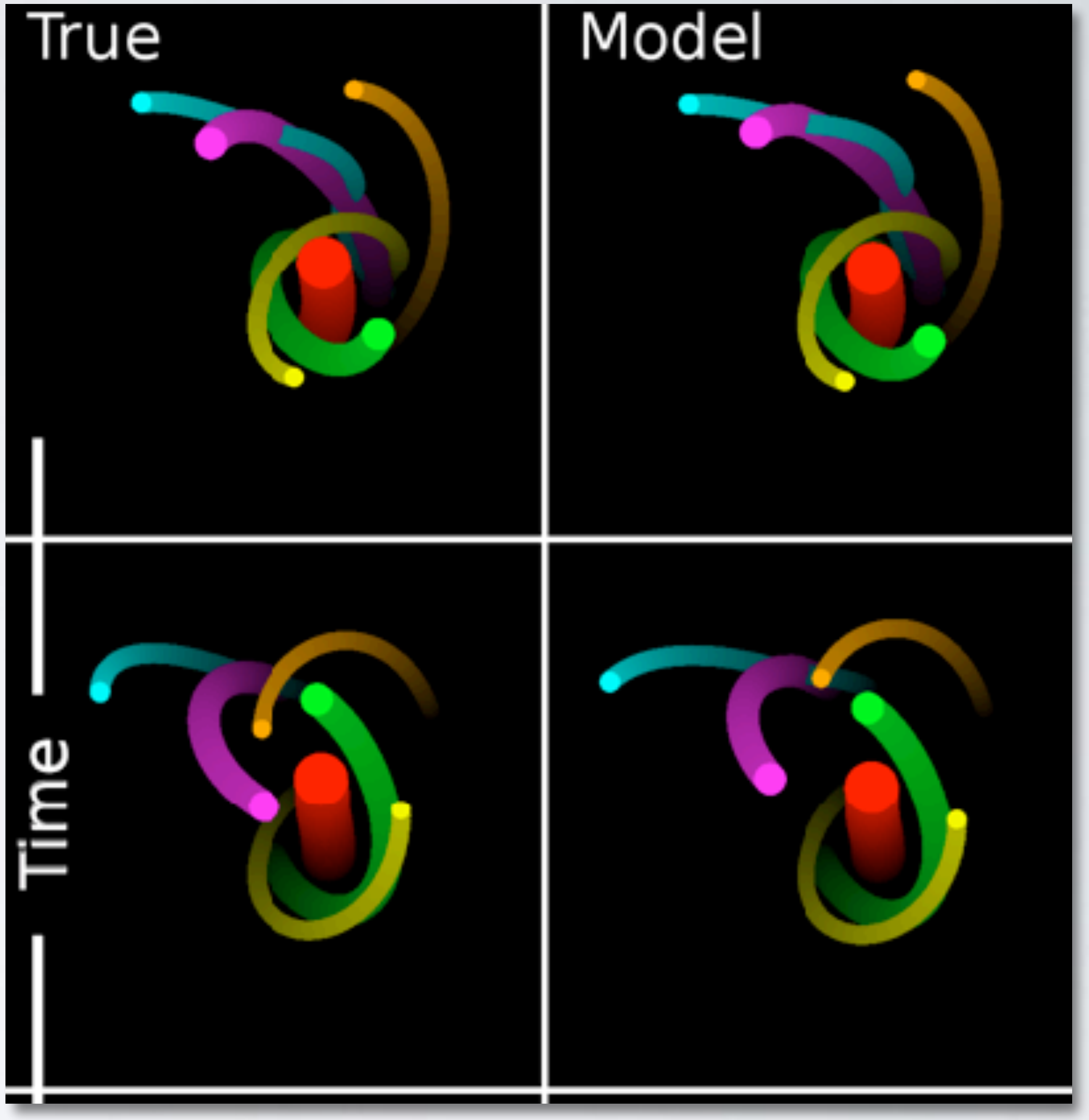
inside:
- edge update function
- node update function
- graph update function



Example 2

Learn the dynamics of physical systems

arXiv:1612.00222



Input

dataset:

Nodes = planets

Features = position in (x,y), velocity vector, mass

Edges = connect every planet to every other planet

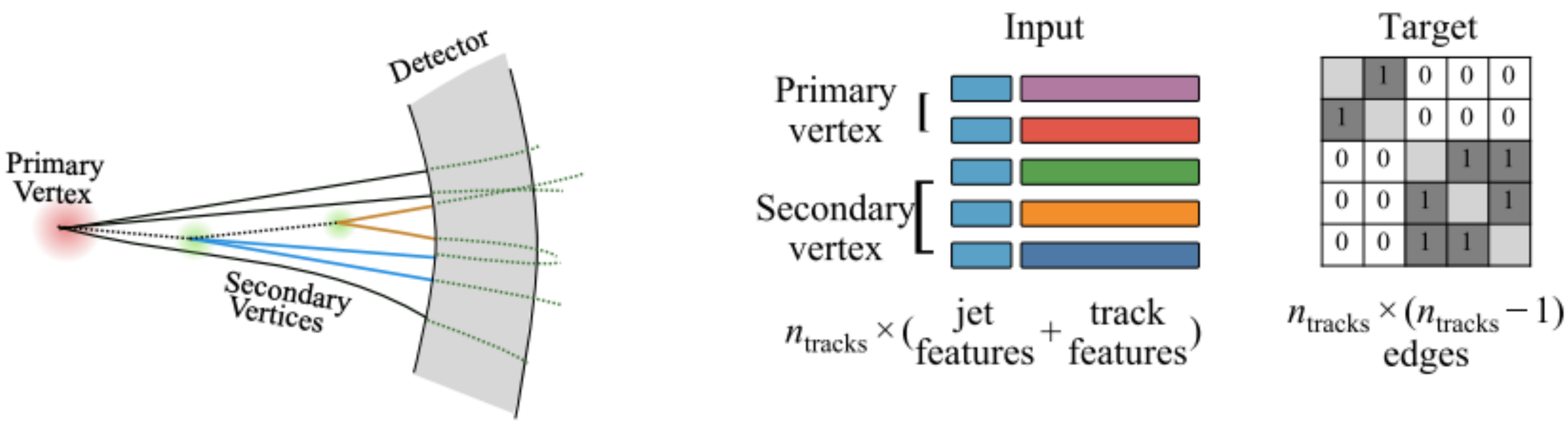
Some Applications

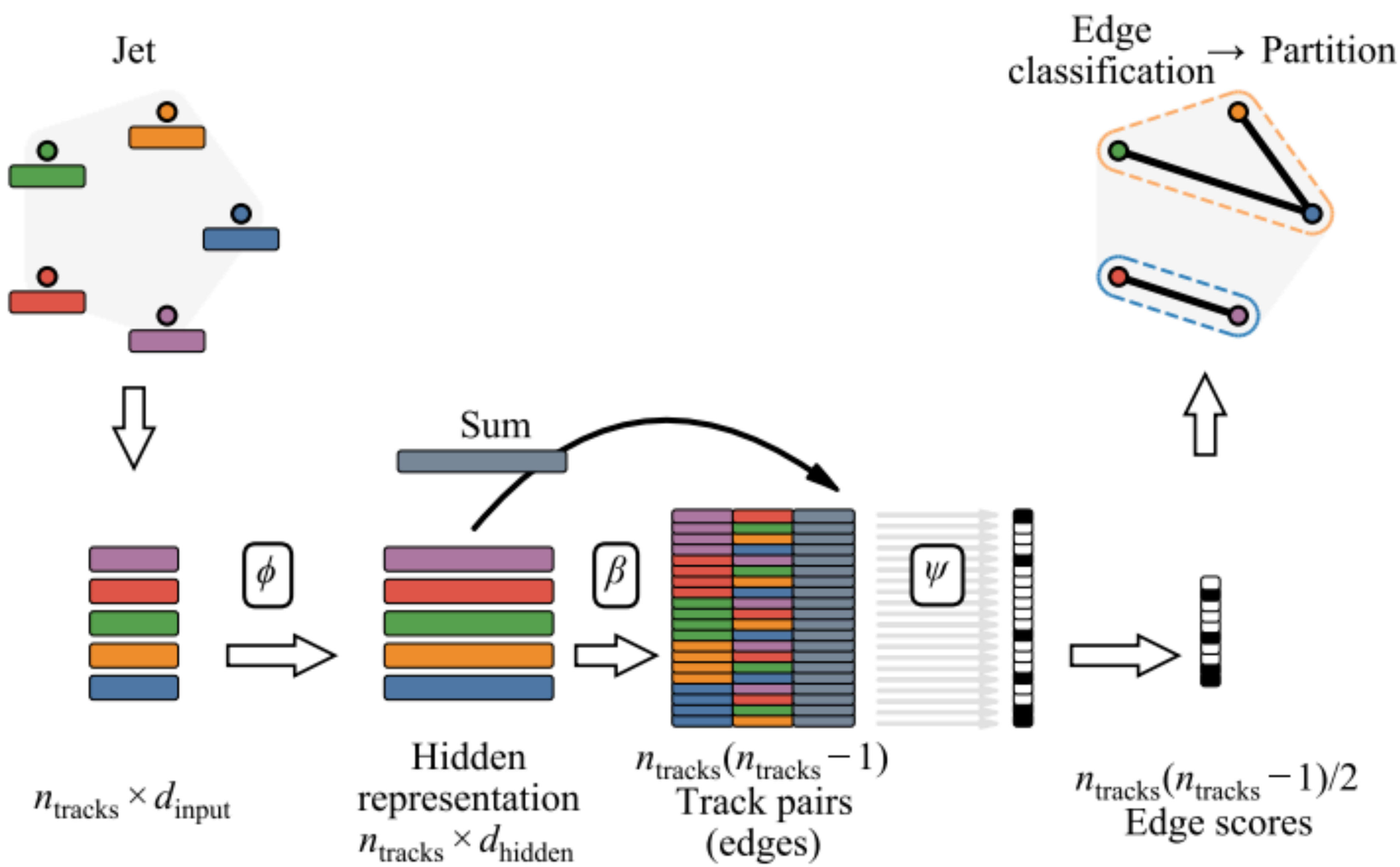
https://arxiv.org/pdf/2008.02831.pdf

Secondary Vertex Finding in Jets with Neural Networks

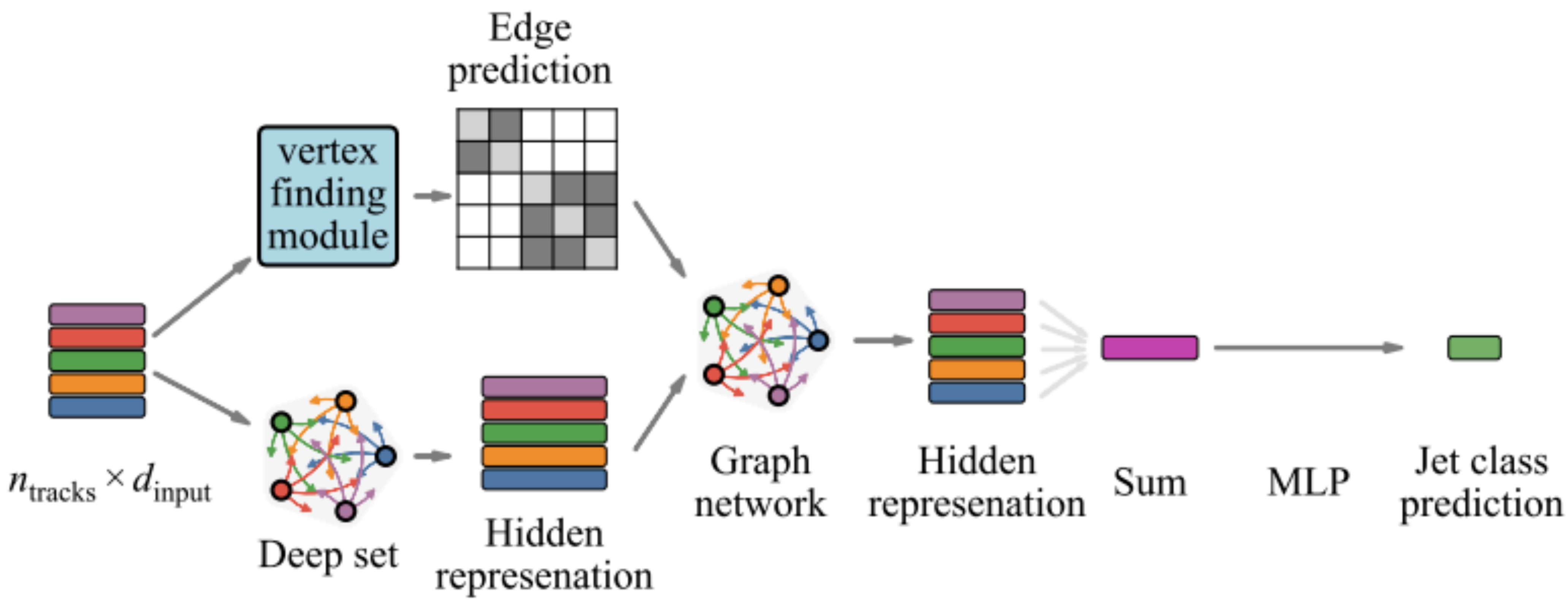
Jonathan Shlomi[1], Sanmay Ganguly[1], Eilam Gross[1], Kyle Cranmer[2] Yaron Lipman[1], Hadar Serviansky[1], Haggai Maron[3], Nimrod Segol[1],

[1]Weizmann Institute Of Science, Israel
[2]NYU
[3]NVIDIA Research





Jet
Edge classification → Partition
Sum
ϕ
β
ψ
$n_{\text{tracks}} \times d_{\text{input}}$
Hidden representation
$n_{\text{tracks}} \times d_{\text{hidden}}$
$n_{\text{tracks}}(n_{\text{tracks}}-1)$
Track pairs (edges)
$n_{\text{tracks}}(n_{\text{tracks}}-1)/2$
Edge scores

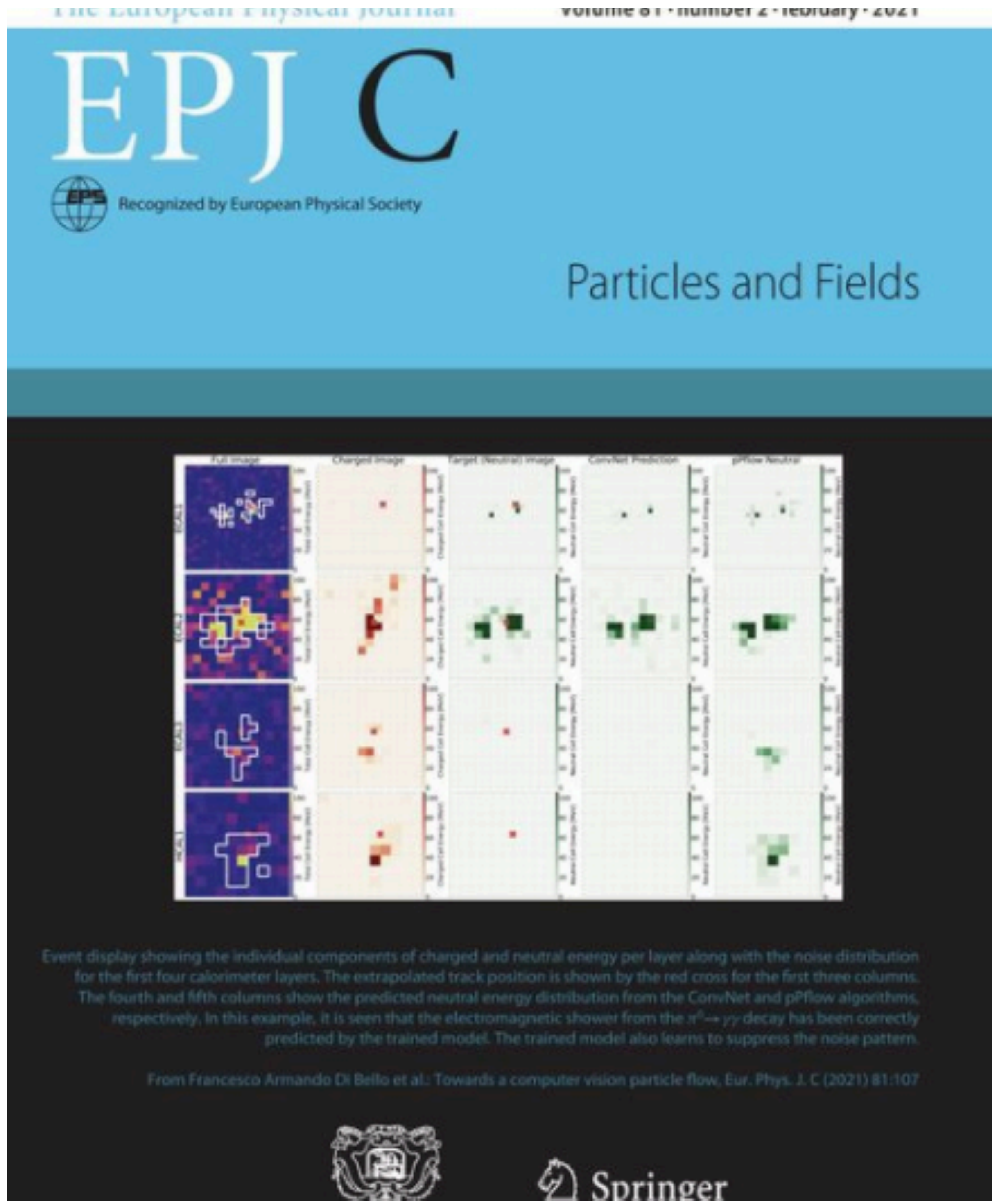


Vertex Finding Module	Accuracy	F1	b jets F1	c jets F1	light jets F1
AVR	0.50	0.49	0.62	0.44	0.40
Baseline	0.57	0.56	0.67	0.40	0.60
Track Pair	0.56	0.57	0.65	**0.48**	0.57
RNN	0.62	0.60	**0.74**	0.37	**0.69**
Set2Graph	**0.63**	**0.62**	0.72	0.44	**0.69**

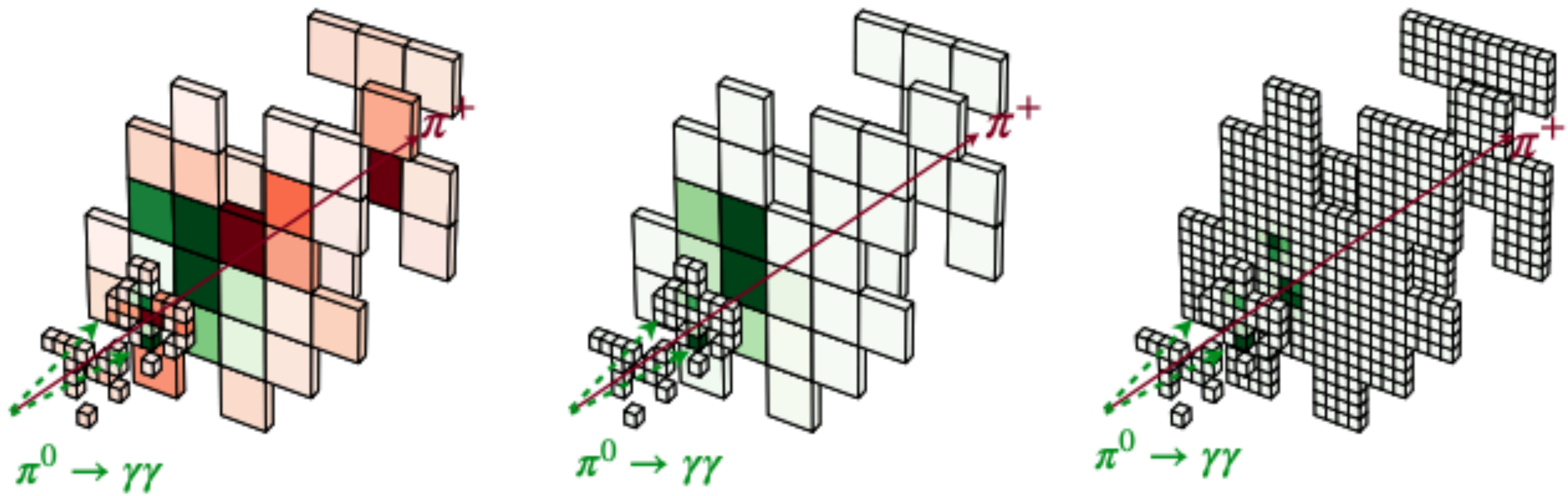
Towards a Computer Vision Particle Flow *

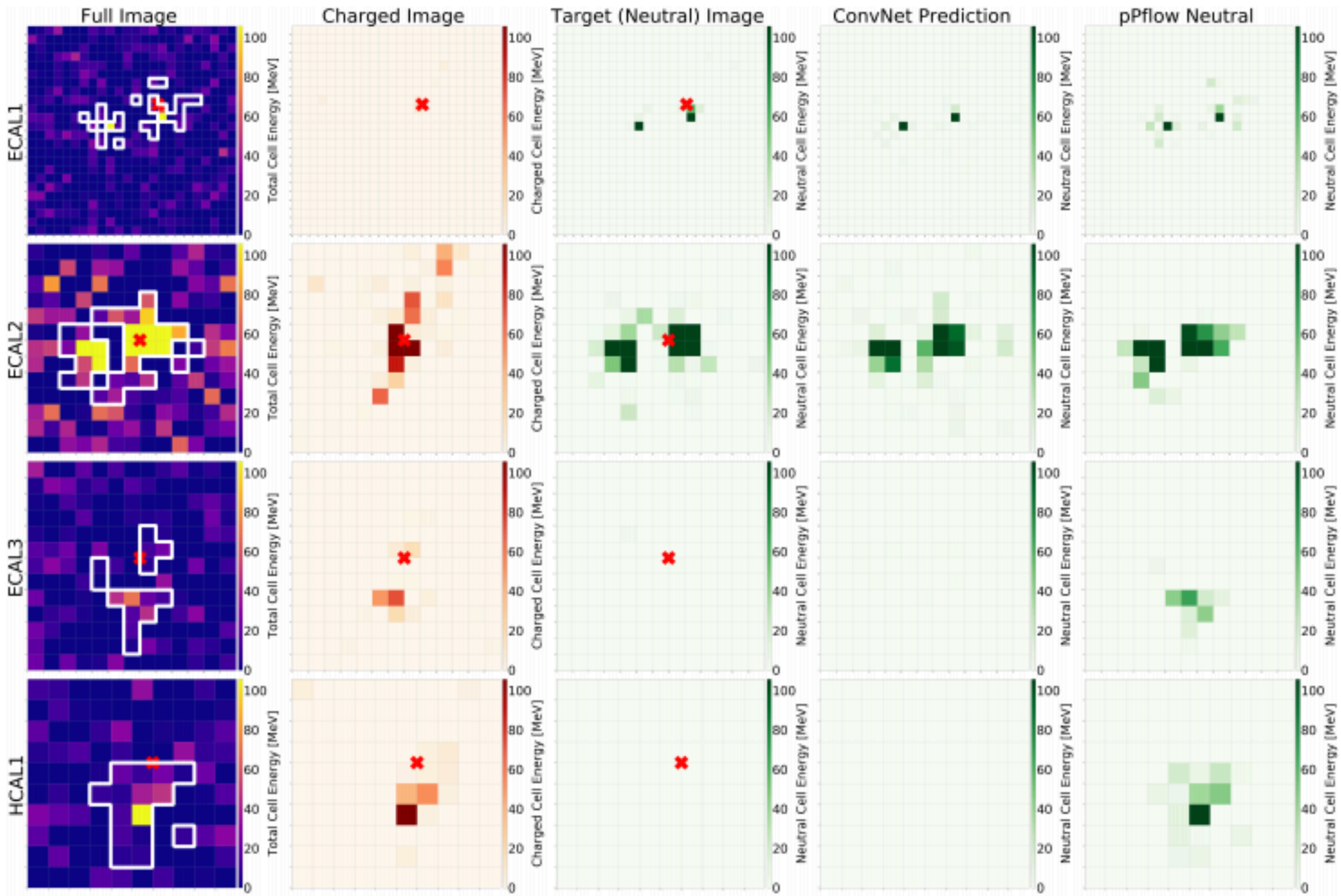
Francesco Armando Di Bello[a,3], **Sanmay Ganguly**[b,1], **Eilam Gross**[1], **Marumi Kado**[3,4], **Michael Pitt**[2], **Lorenzo Santi** [3], **Jonathan Shlomi**[1]

[1]Weizmann Institute of Science, Rehovot 76100, Israel
[2]CERN, CH 1211, Geneva 23, Switzerland
[3]Università di Roma Sapienza, Piazza Aldo Moro, 2, 00185 Roma, Italy e INFN, Italy
[4]Université Paris-Saclay, CNRS/IN2P3, IJCLab, 91405, Orsay, France



Super resolution





Full Image
Charged Image
Target (Neutral) Image
ConvNet Prediction
pPflow Neutral
ECAL1
ECAL2
ECAL3
HCAL1
Total Cell Energy [MeV]
Charged Cell Energy [MeV]
Neutral Cell Energy [MeV]
0
20
40
60
80
100

Summary

- Deep Learning enables the implementation of a complicated or unknown function from DATA X to some Target Y

- It can be supervised (Classification) or Unsupervised (Learning from DATA without labeling)

- Graph Neural Nets enable functions from sets to sets of sparse DATA with different structures

- DL is becoming an analysis tool you cannot do without, so start to take it seriously