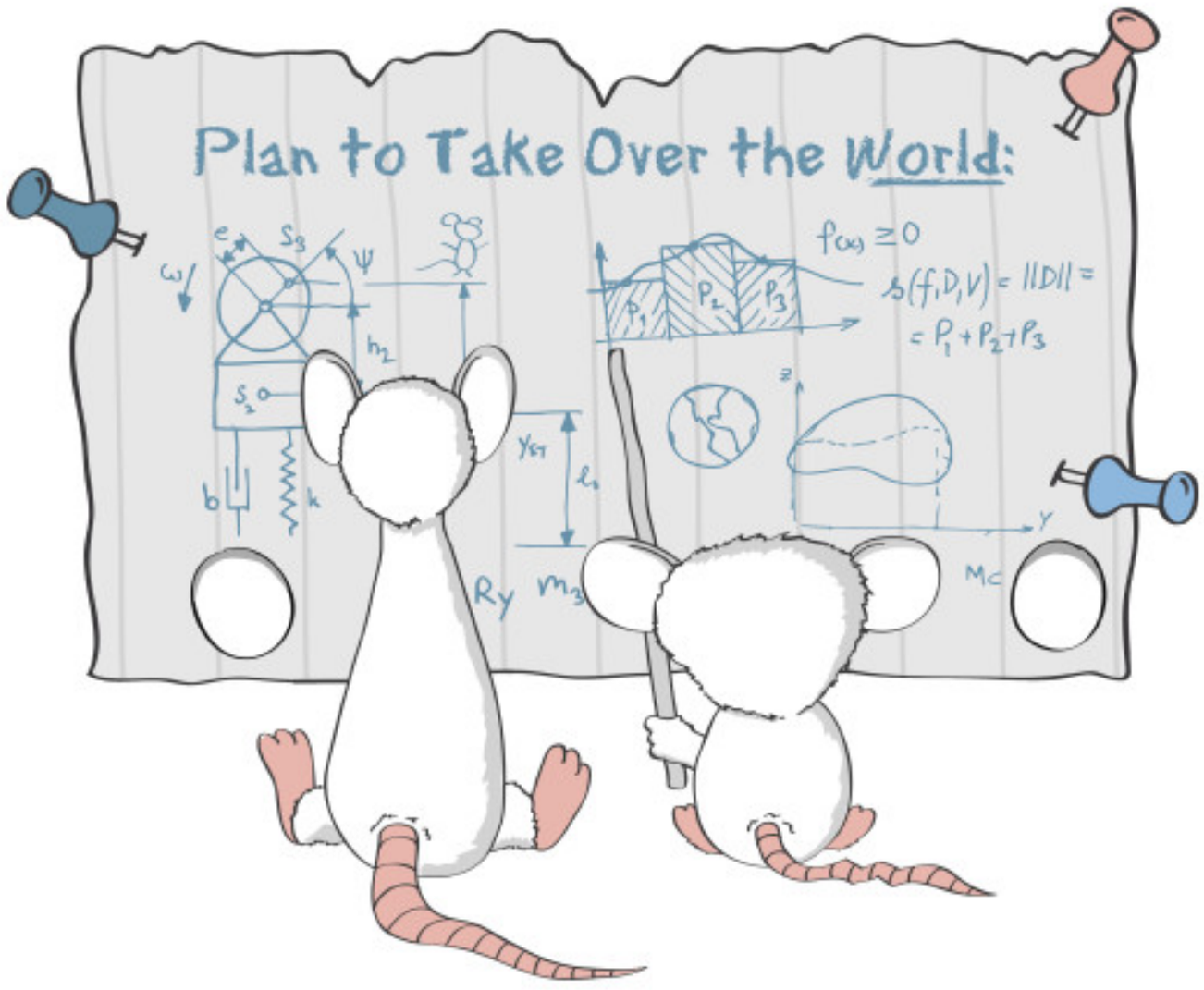


Neural Networks

DIAS ML course by Martin Topinka

https://github.com/toastmaker/ml-dias


PINKA MARTIN
09090027
9.09.1976
3.01.2019
2:39:32
Sn 1
kV 70
mAs 4.0
dGy*cm2 1.6600

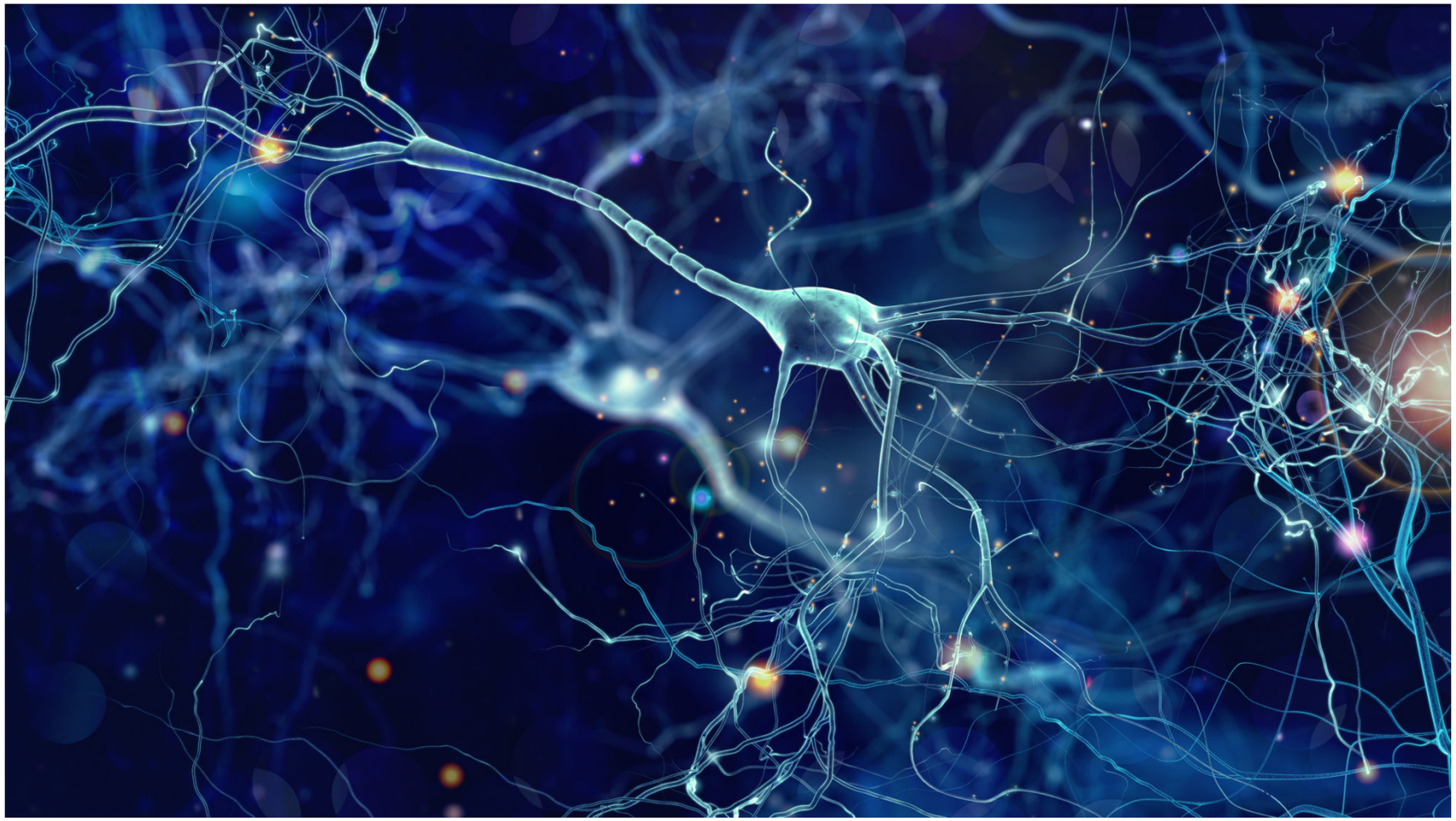
Outline

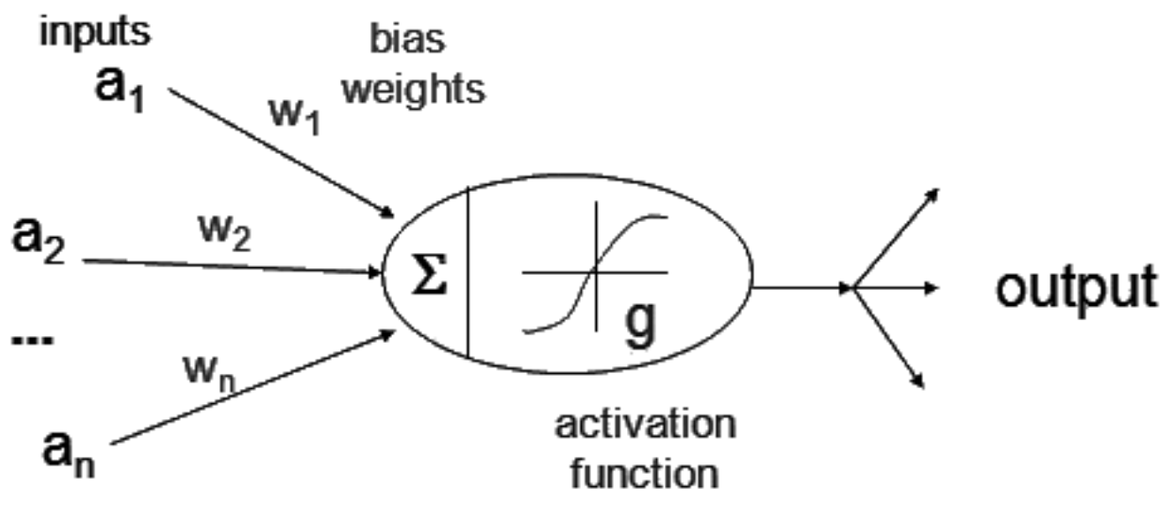
- Introduction to NN: the idea behind, SGD, vanishing gradient, fighting over-fitting)
- Convolutional Neural Network
- Encoder-decoders
- GANs
- TensorFlow/Keras
- Hands-on session: NN star/quasar classification, ConvNN source finding/denoising
- Not included: Reinforcement Learning

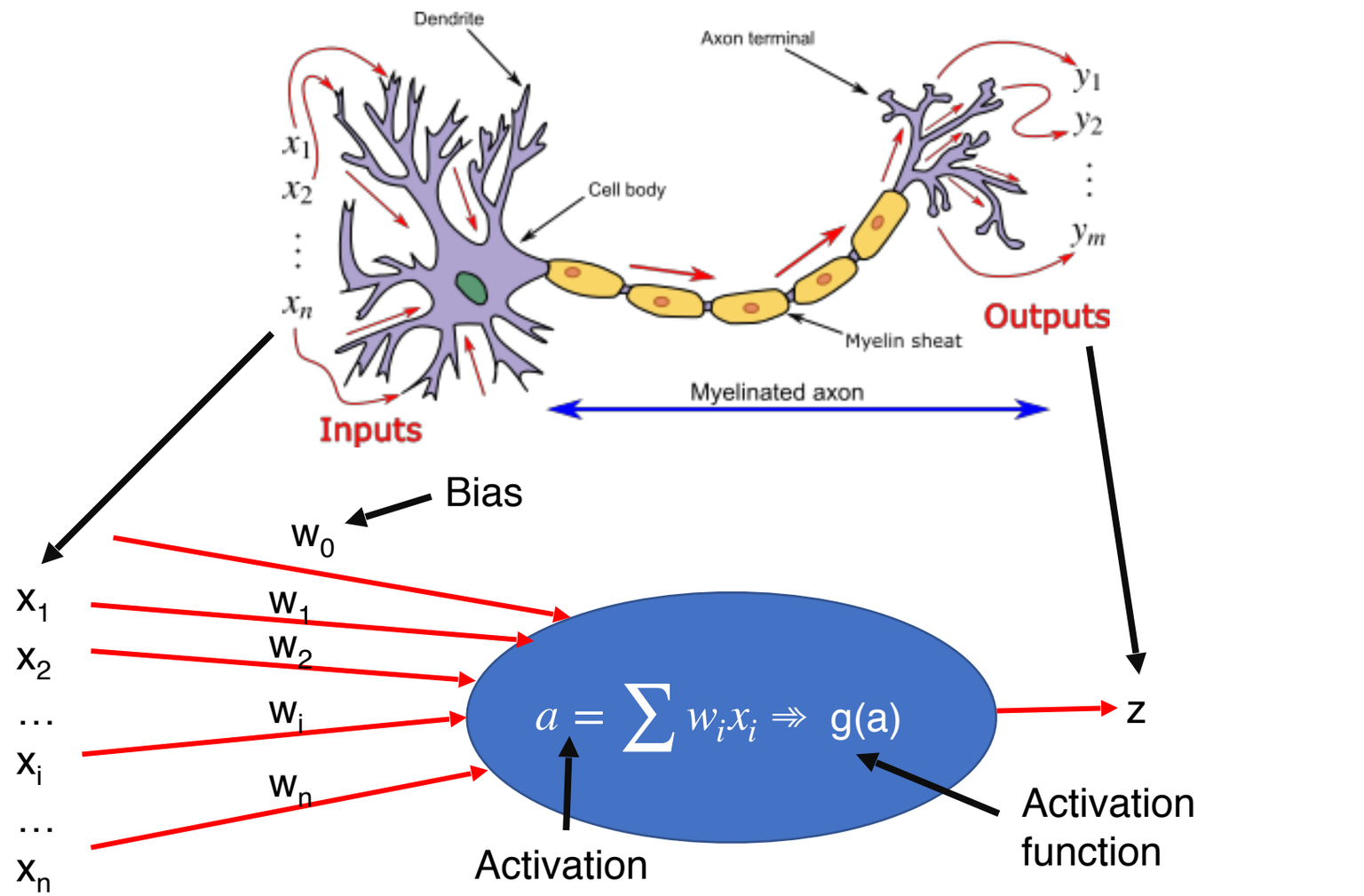
“Deep learning (neural networks) is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it….”

– a ML expert

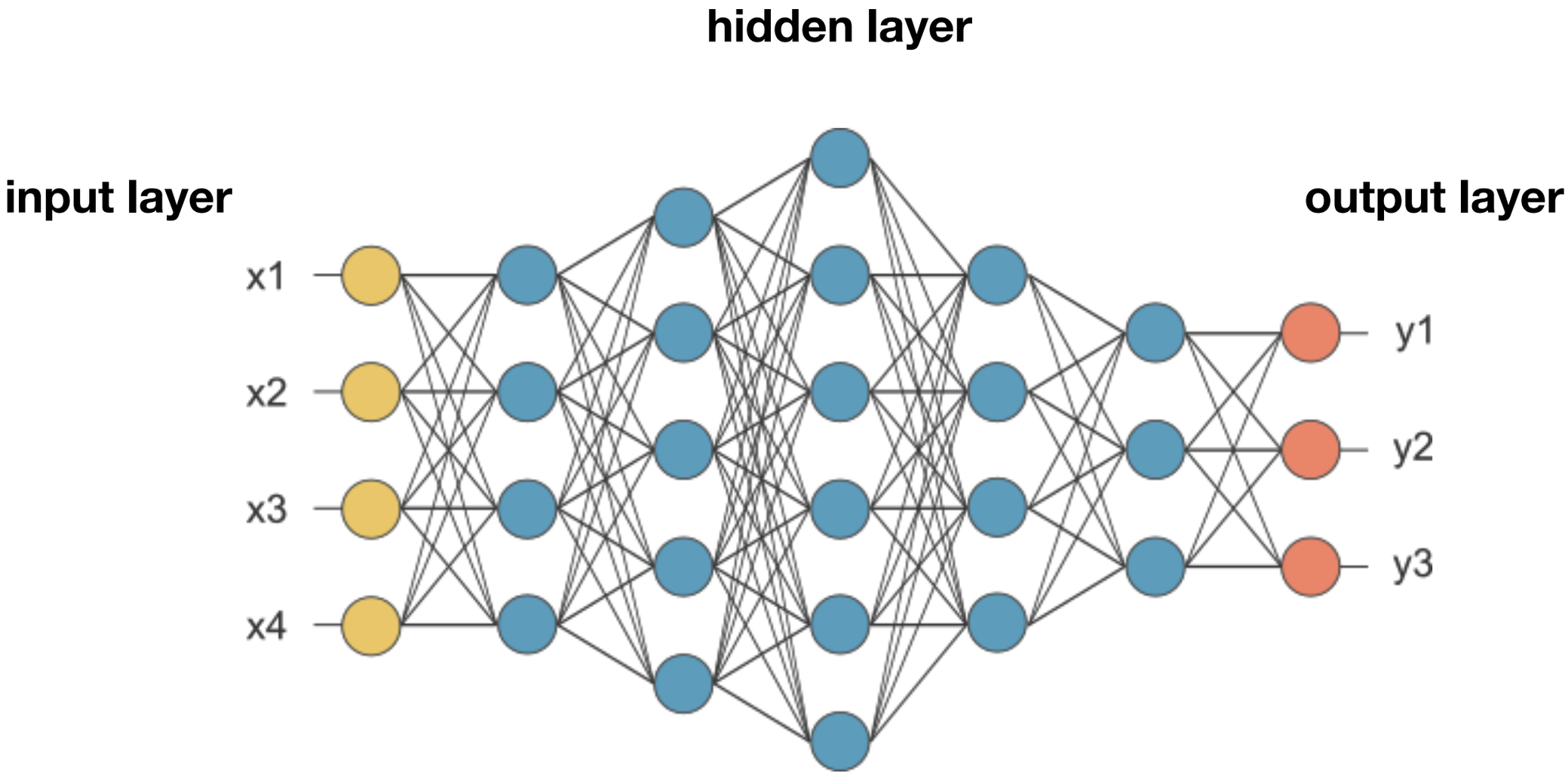
Buzz words: machine learning, deep learning, AI, neural network, Bayesian analysis, gravitational waves, exoplanets, dark matter…

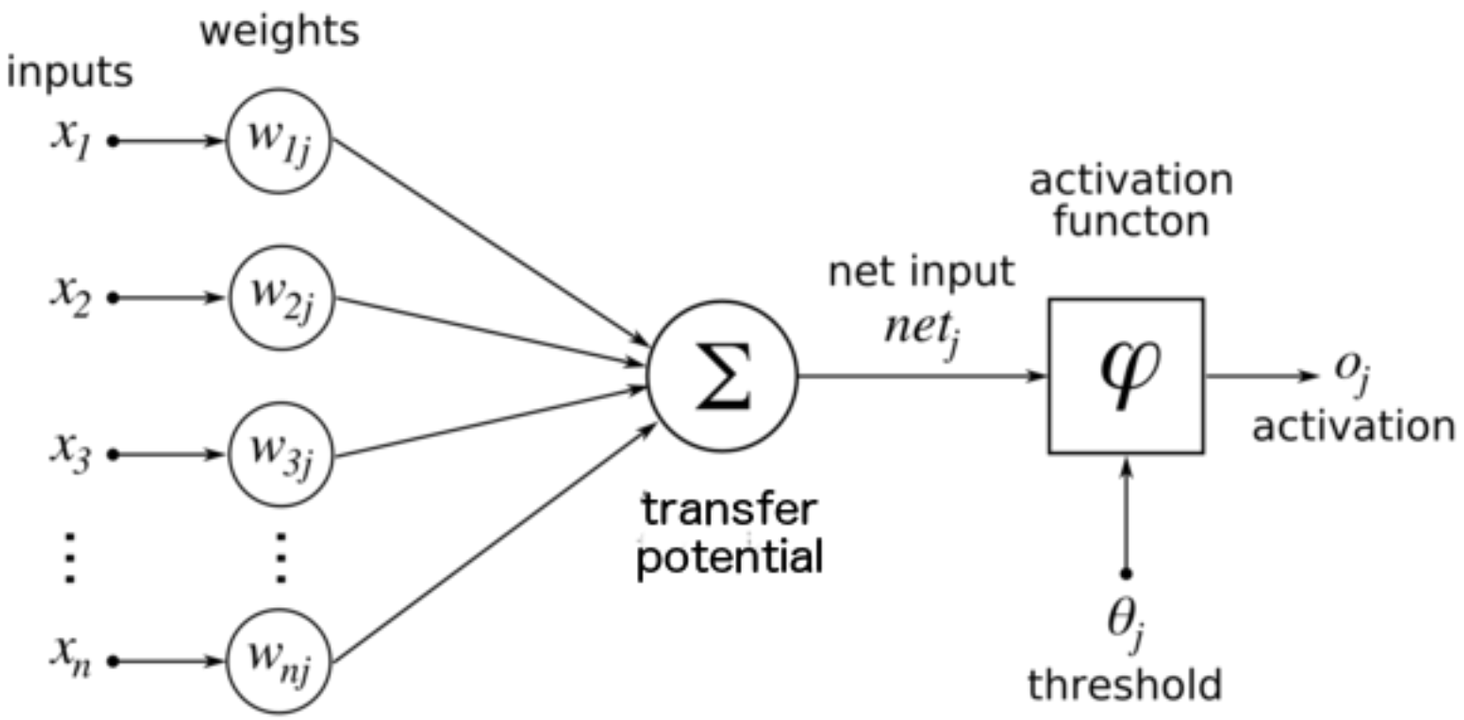



inputs
a_1
a_2
...
a_n
w_1
w_2
w_n
bias
weights
Σ
g
activation
function
output



g = identity or sigmoid
Linear/logistic regression

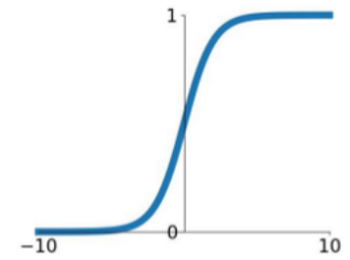

hidden layer
input layer
output layer
x1
x2
x3
x4
y1
y2
y3



Activation Functions

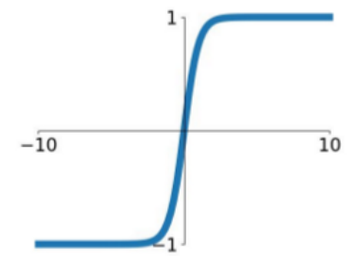
Sigmoid

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



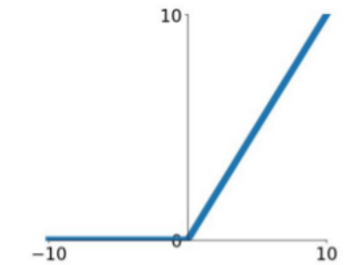
tanh

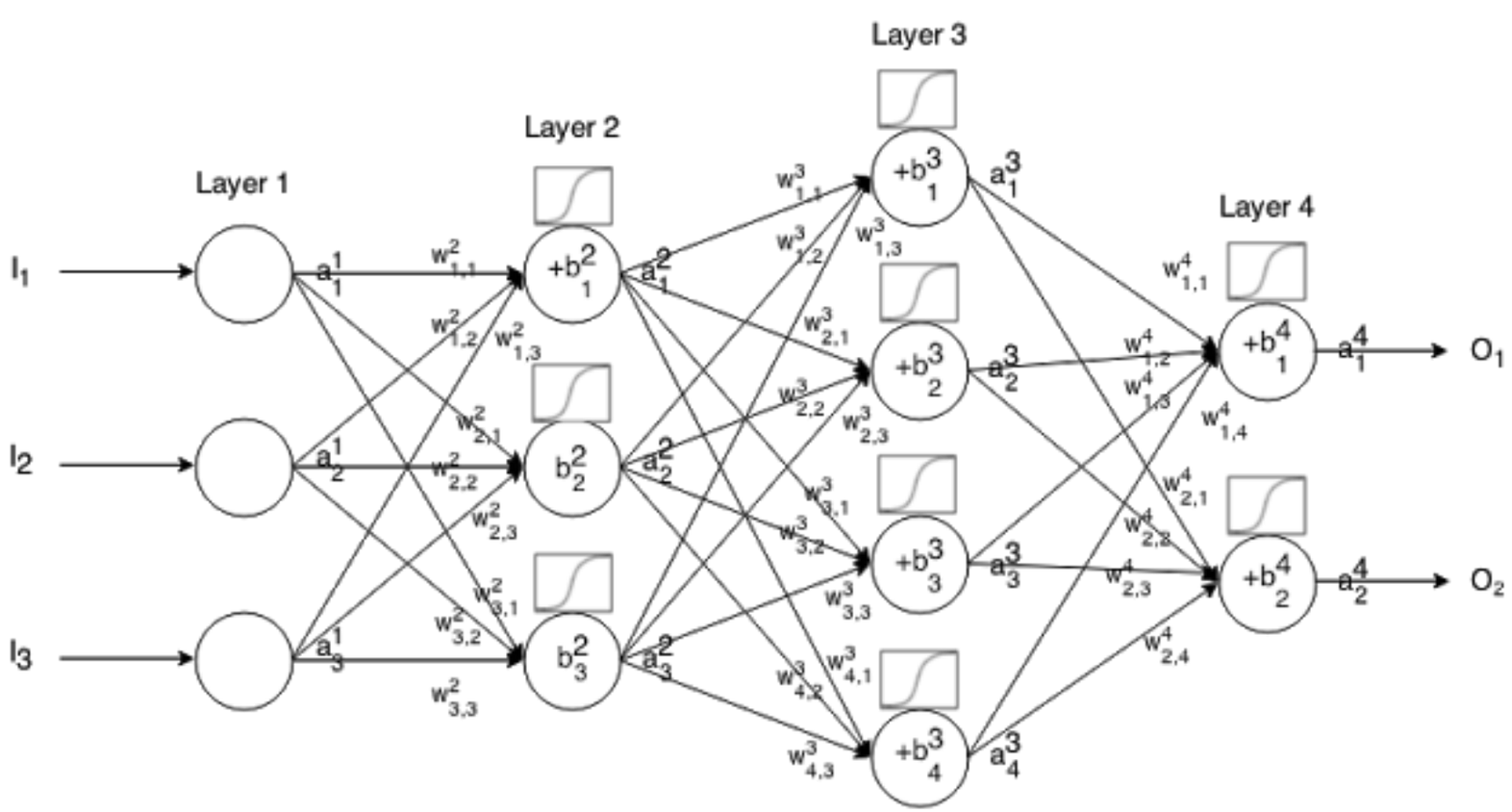
$$\tanh(x)$$

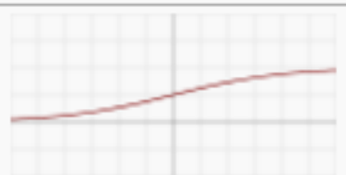
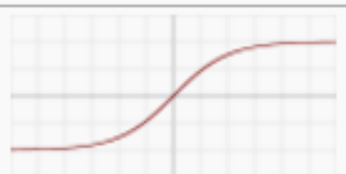
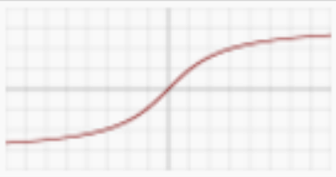
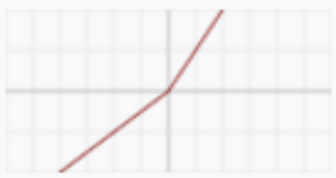
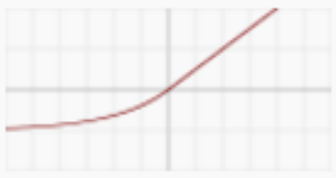
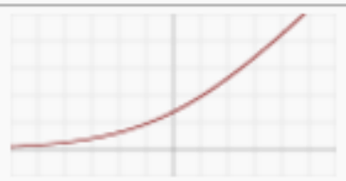


ReLU

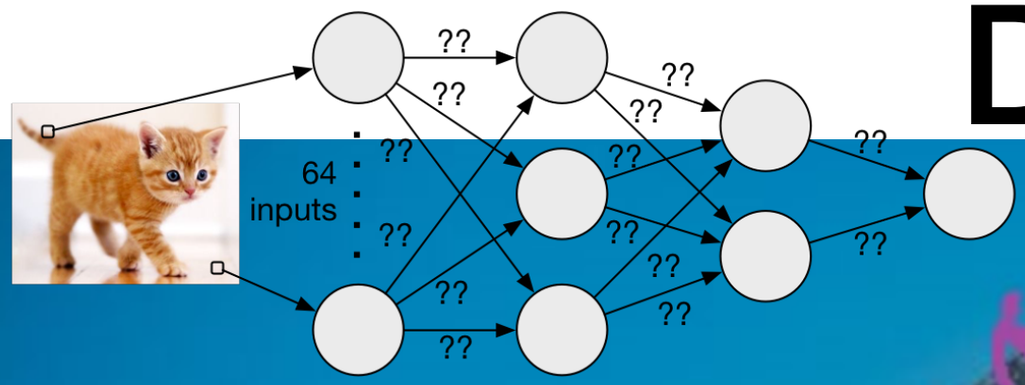
$$\max(0, x)$$



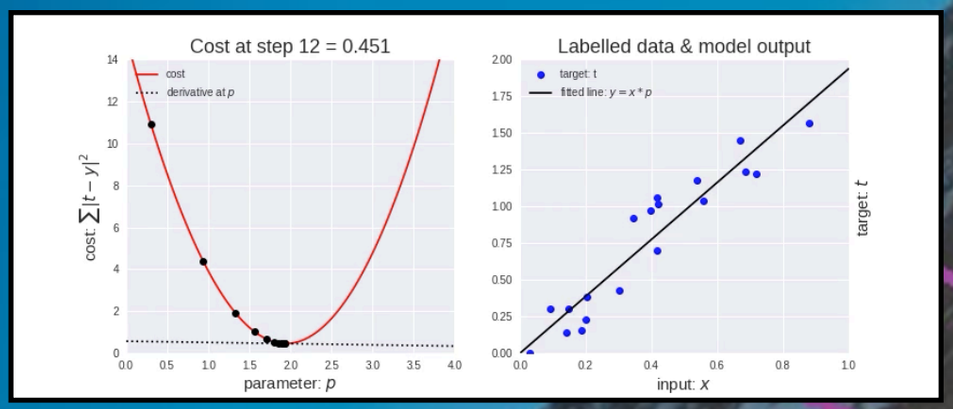


Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) \begin{cases} 0 & \text{for} \quad x \neq 0 \\ ? & \text{for} \quad x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU)[2]		$f(x) = \begin{cases} \alpha x & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)[3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

(Stochastic) Gradient Descent



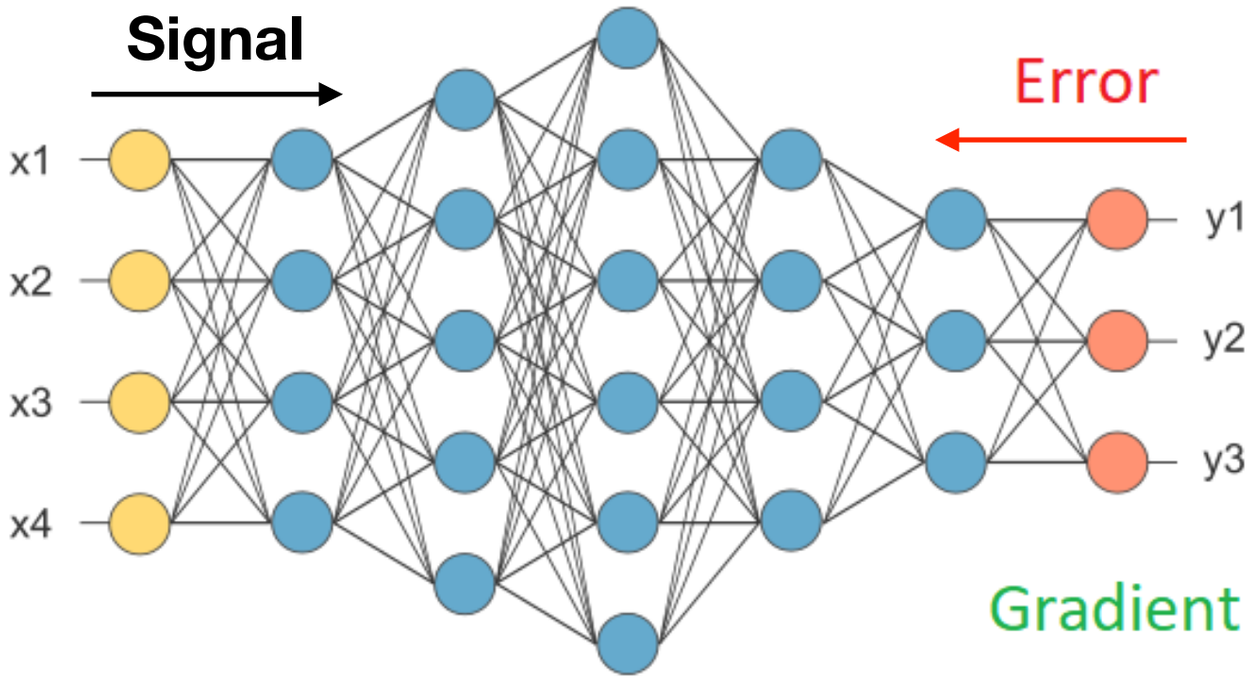
$$w = w - \alpha \nabla_w J(w)$$



Back-propagation

Forward-propagation: get **estimates** during training and **predictions** then

Back-propagation: apply chain rule to gradient of loss function to **adjust weights/biases**



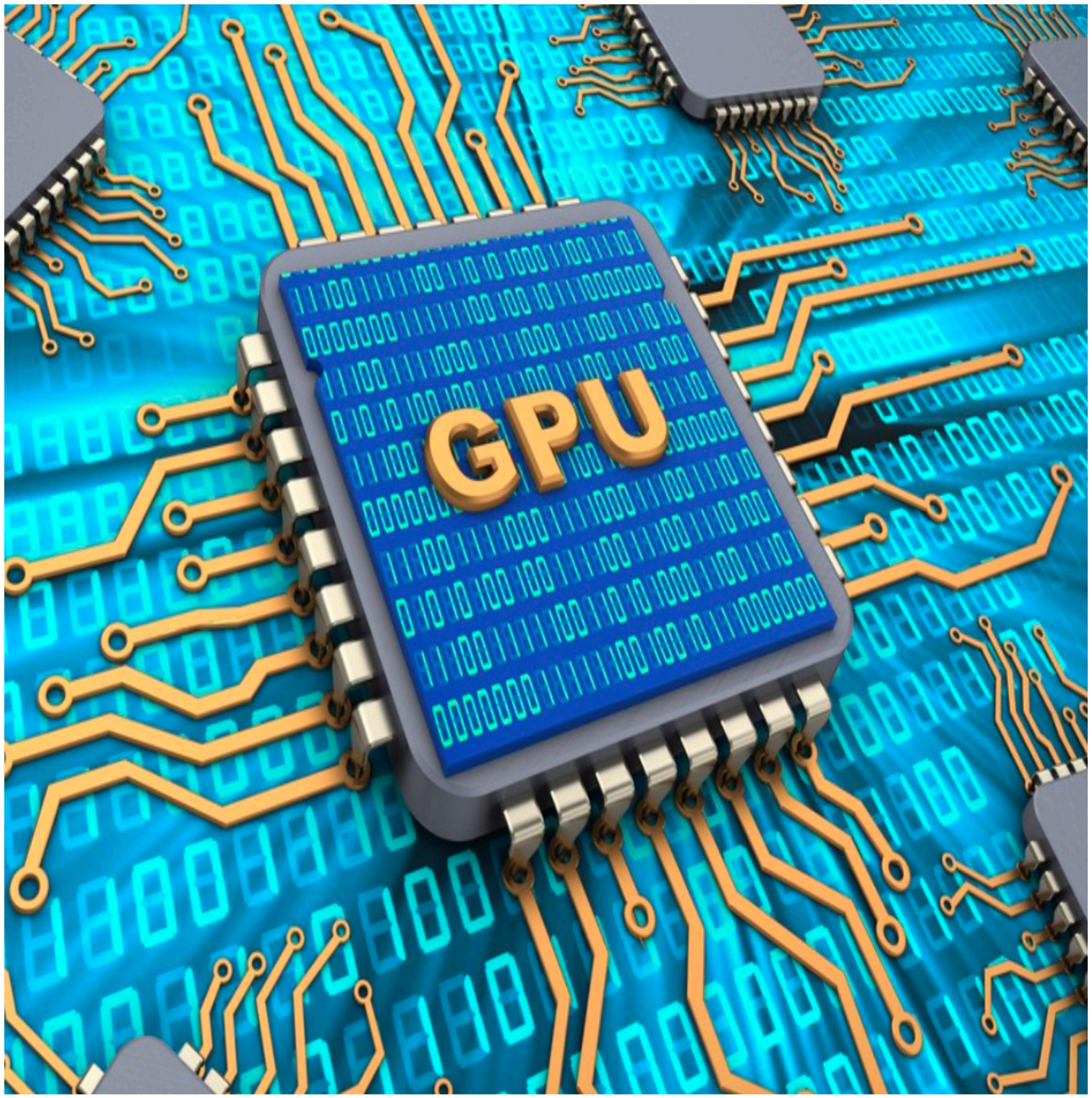
Vanishing Gradient

Exploding Gradient

Remedies against vanishing gradient:

- ReLU
- Re-normalisation

Algorithms + Computational Resources + Available data = Boom of ML (specially deep learning)

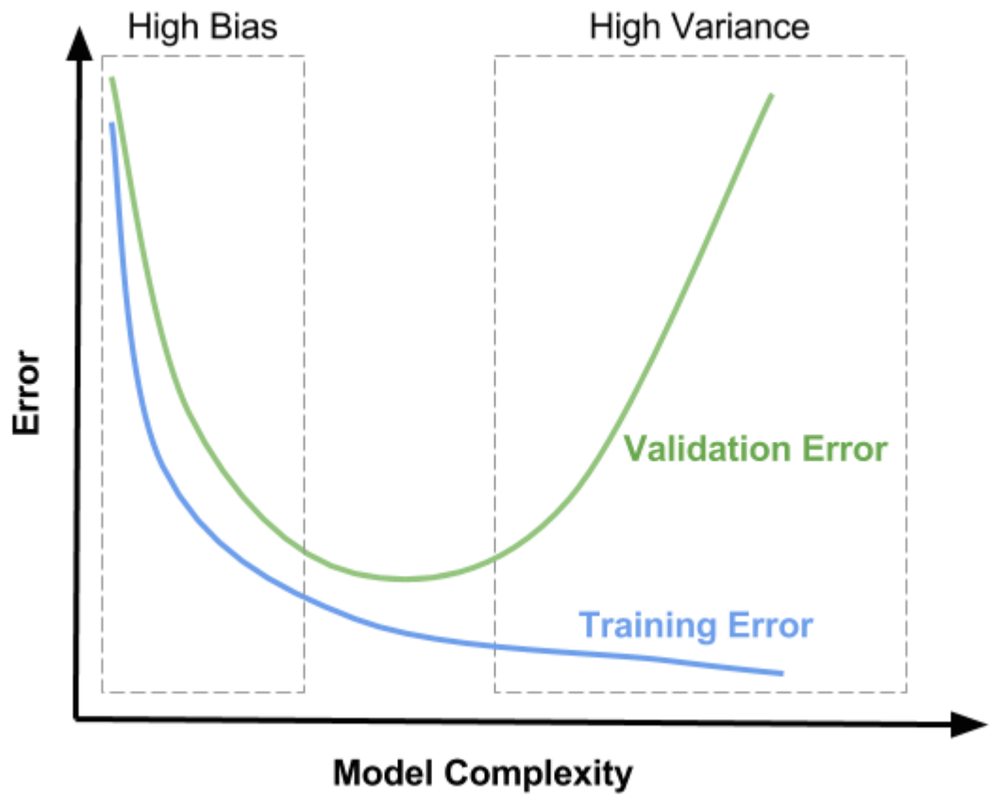


Small Dataset → over-fitting

It’s a cat

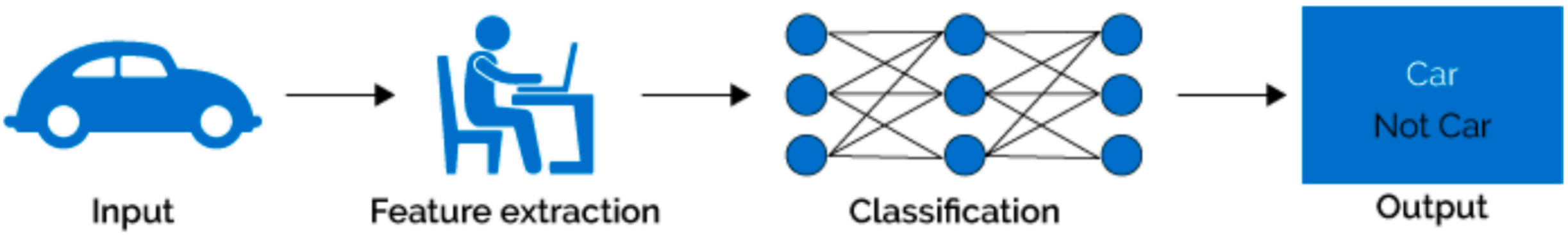
Fighting over-fitting

- Dropout
- Regularisation
- Early stopping
- Simplify the architecture
- More data (even if you have to make it up: translate, rotate, flip, crop, lighten/darken, add noise)

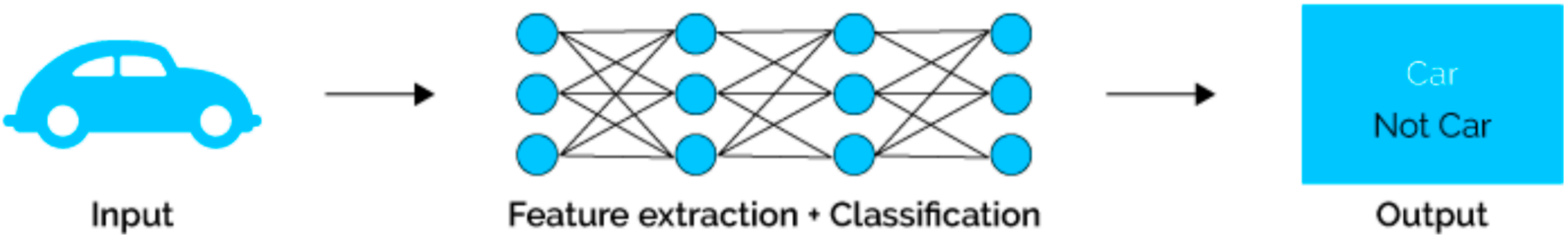


# of weights ~ VC dimension ~ # degrees of freedom

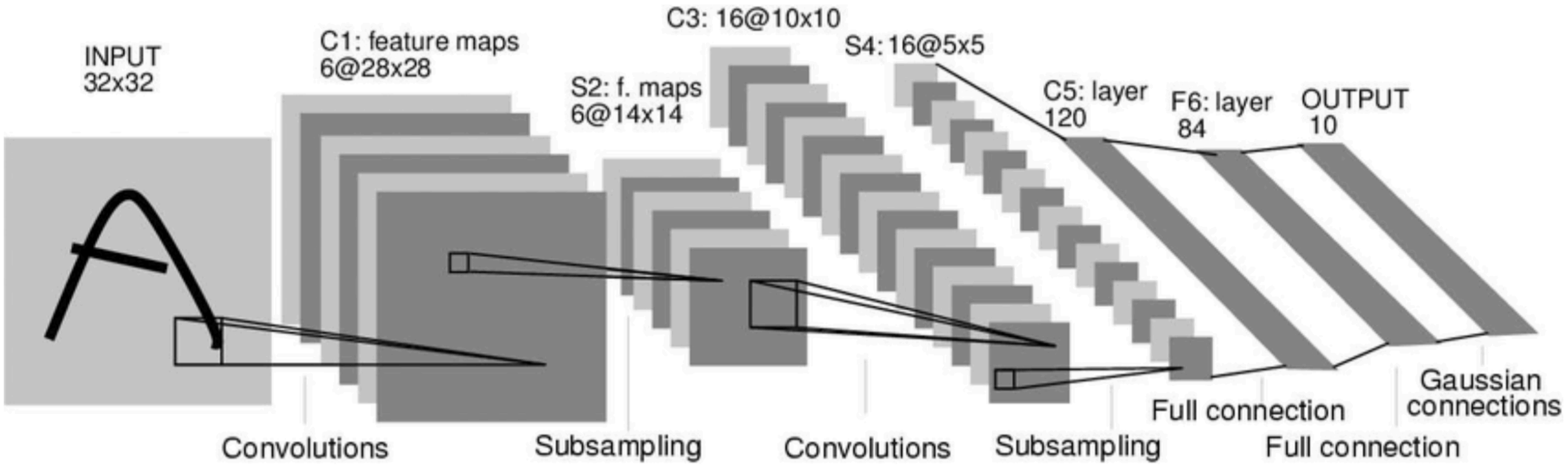
Machine Learning

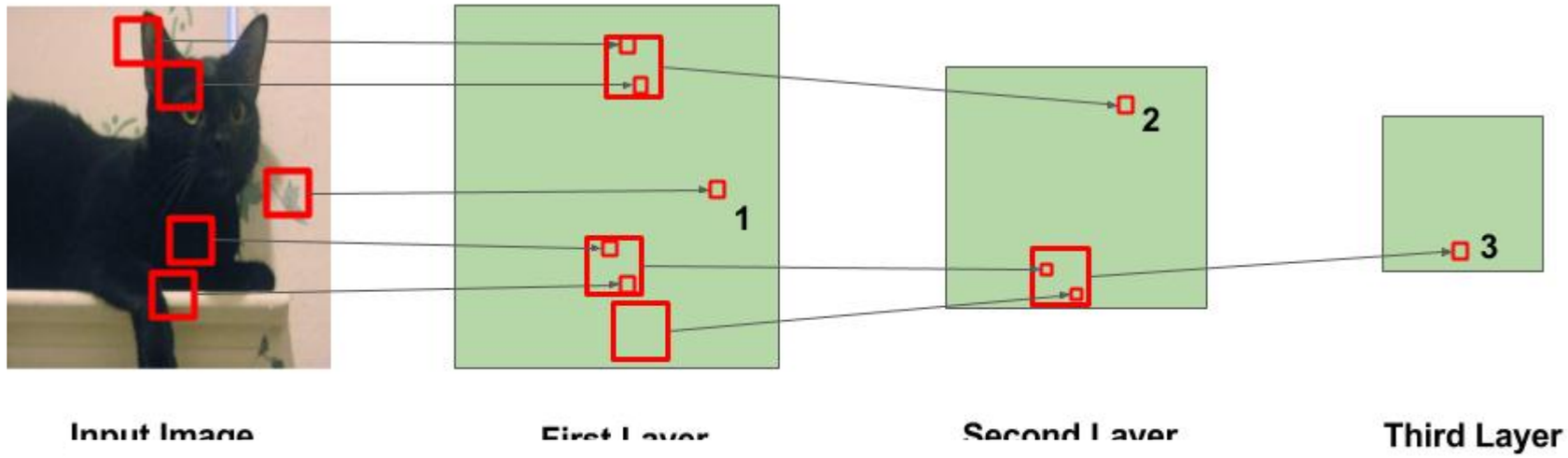


Deep Learning

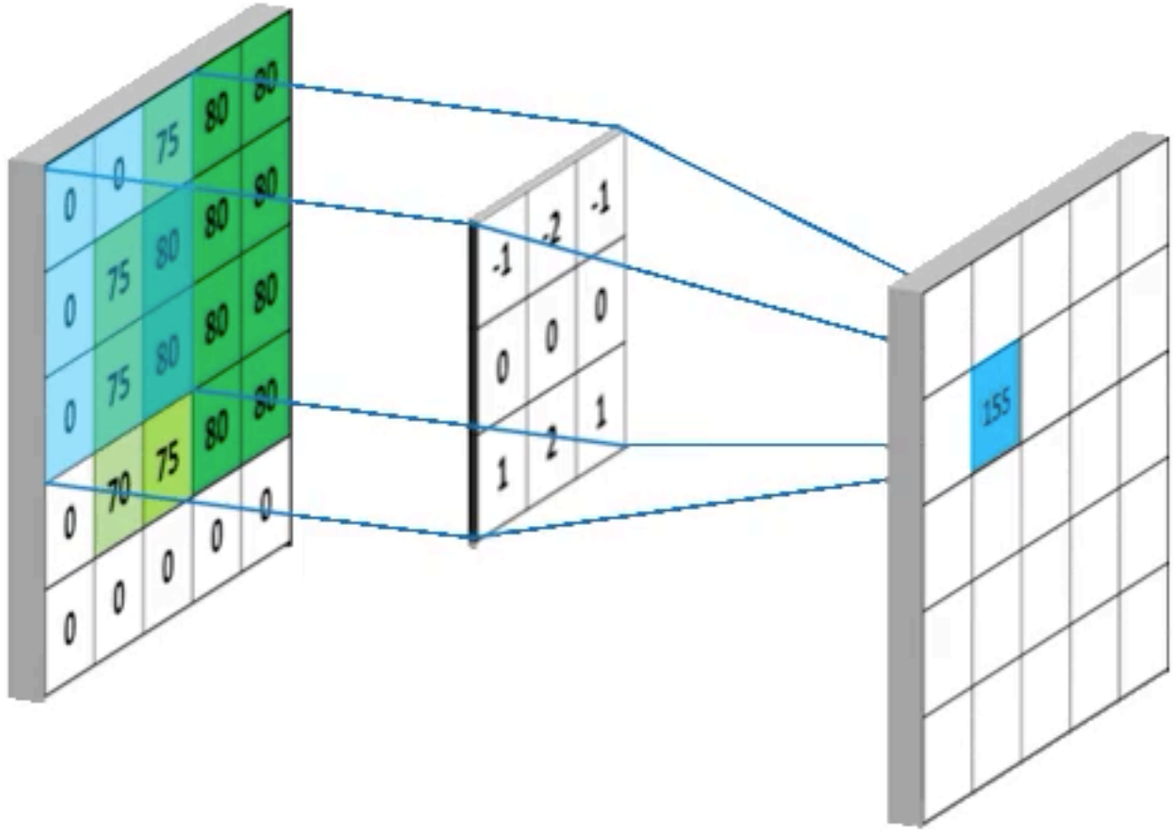


Convolutional Neural Networks



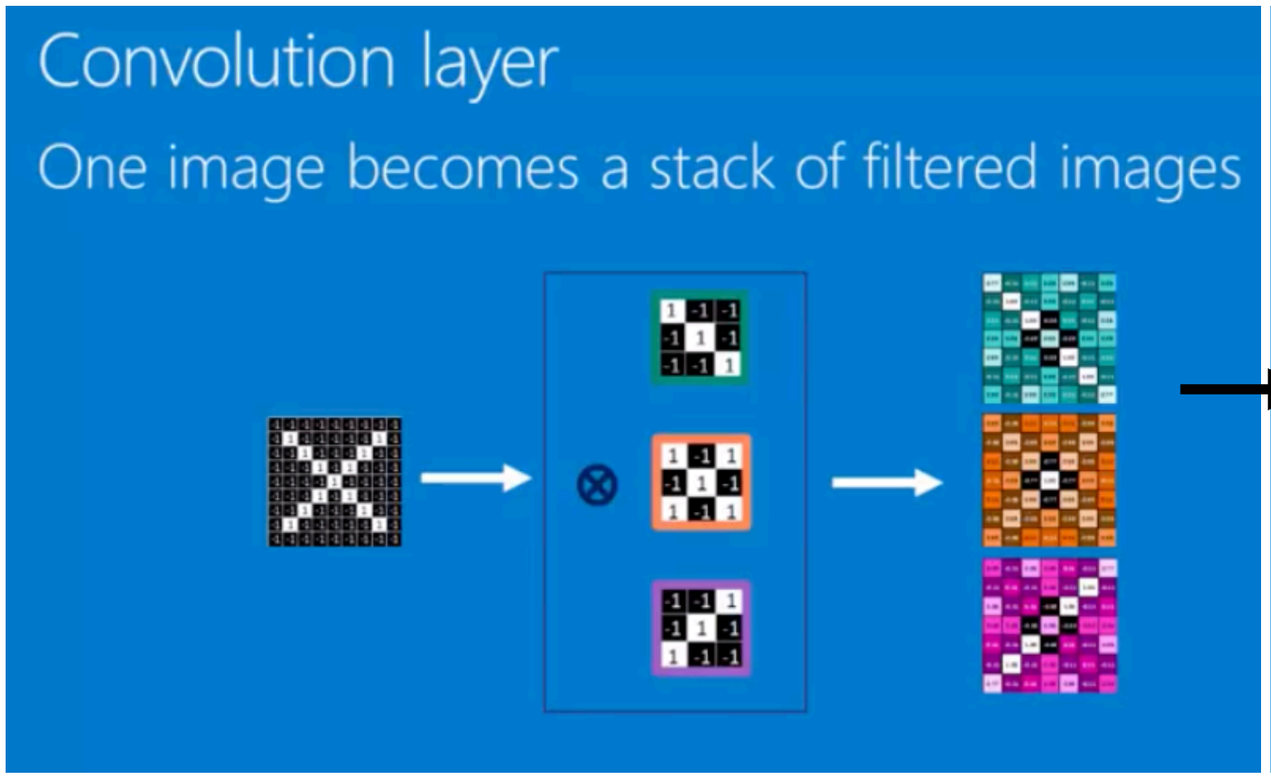


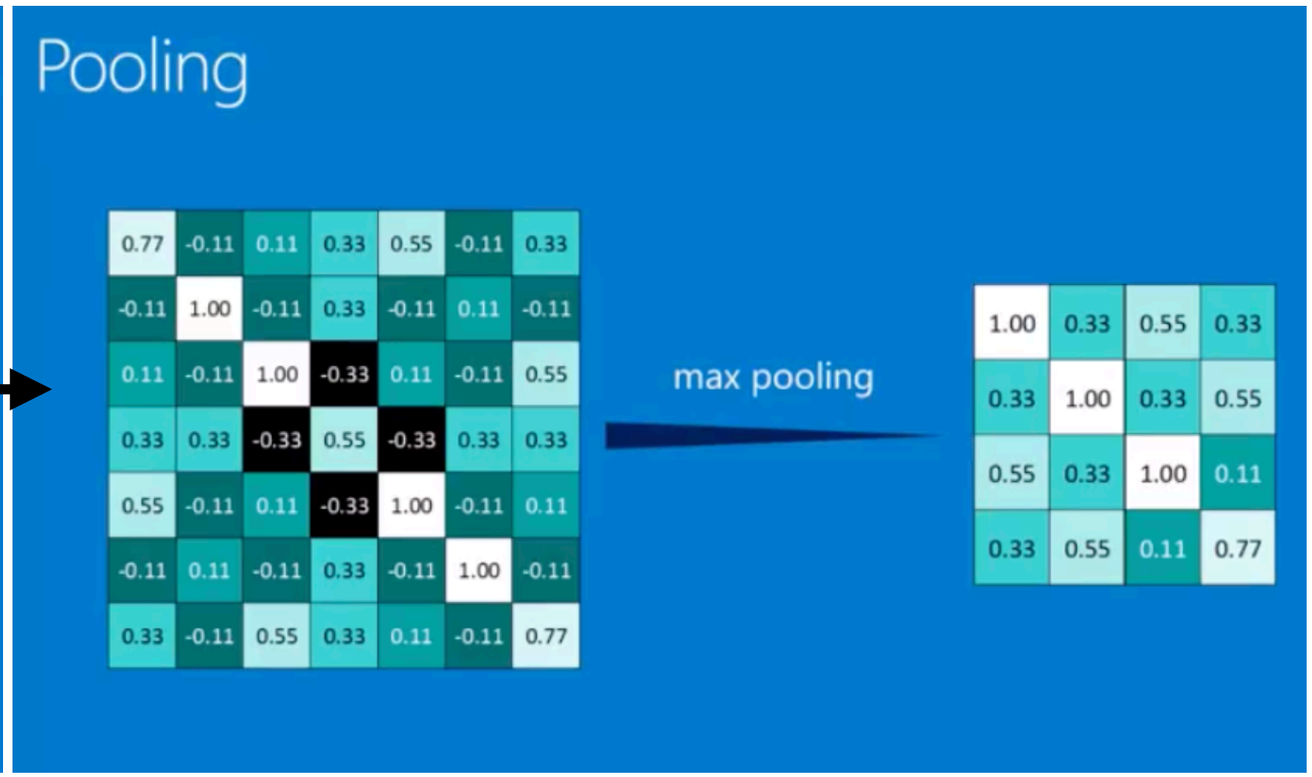
Convolution (with a kernel that is a parameter to be fit)



Input layer (source data)

— Dim reduction
— Spatial tolerance





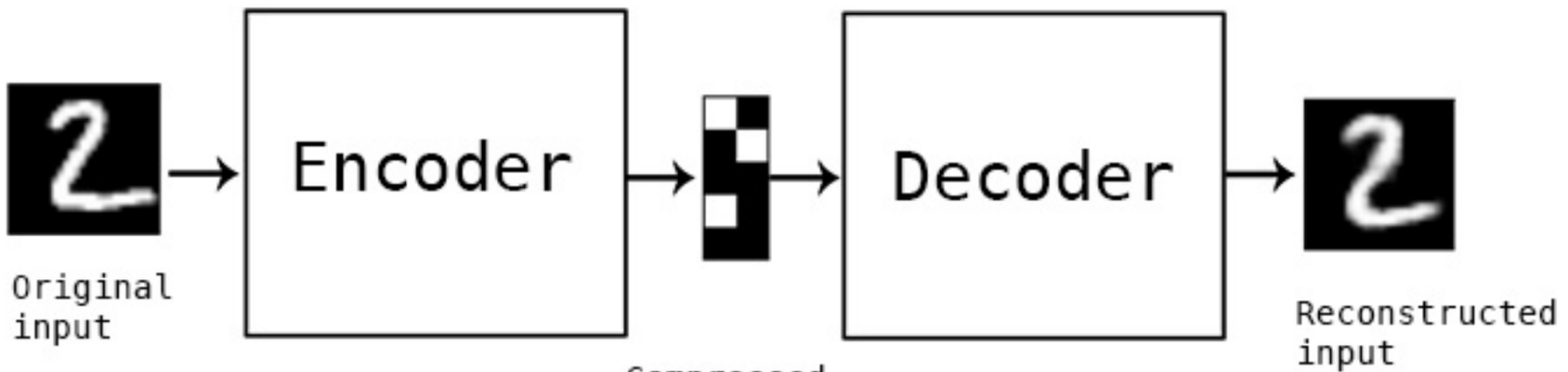
Repeat many times to build-up hierarchically more complex patterns

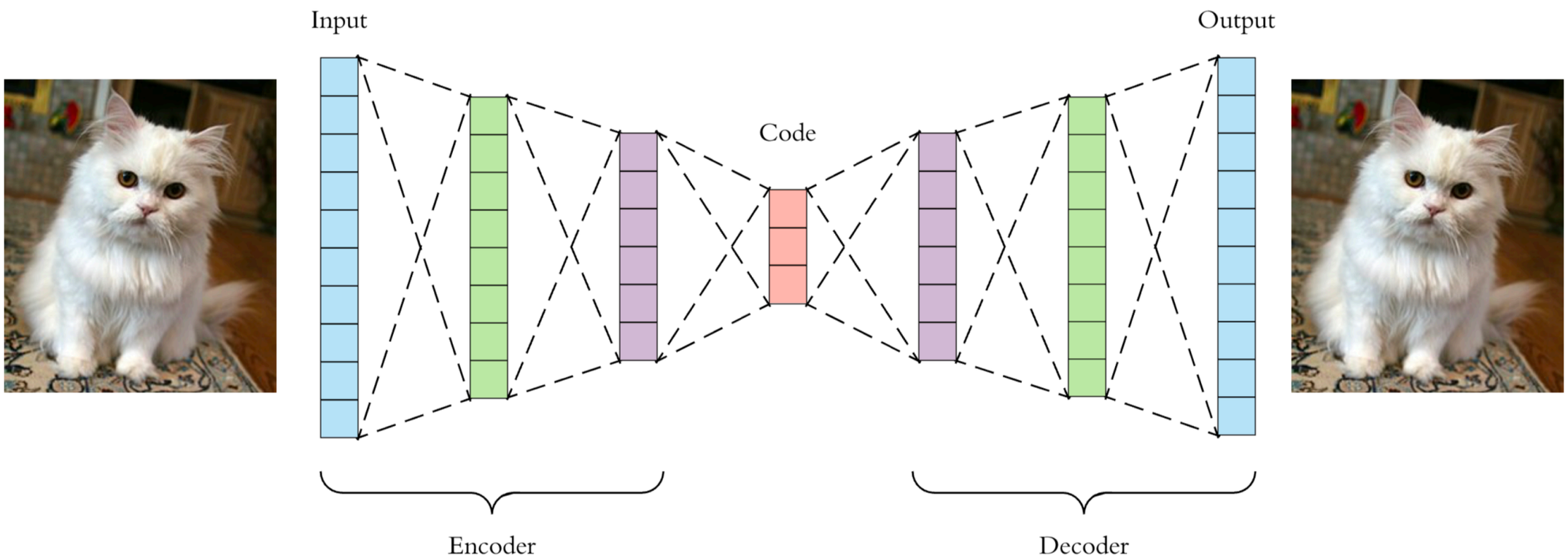
Dense NN layer — "learning"

Automatic feature extraction!

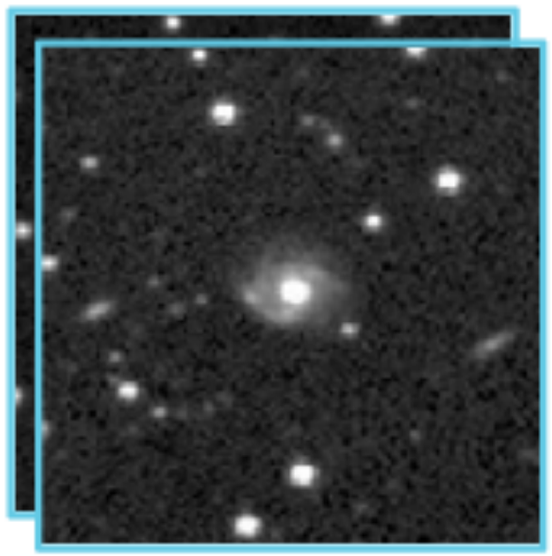
Output layer
Calculate error to adjust weights

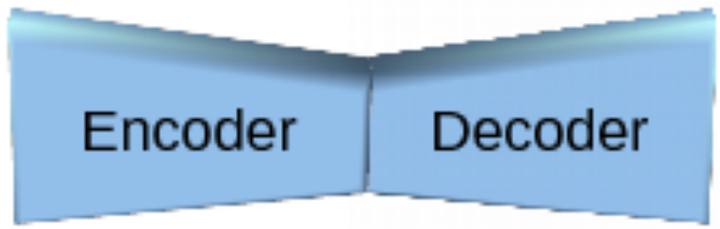
Encoder-decoders (Autoencoders)





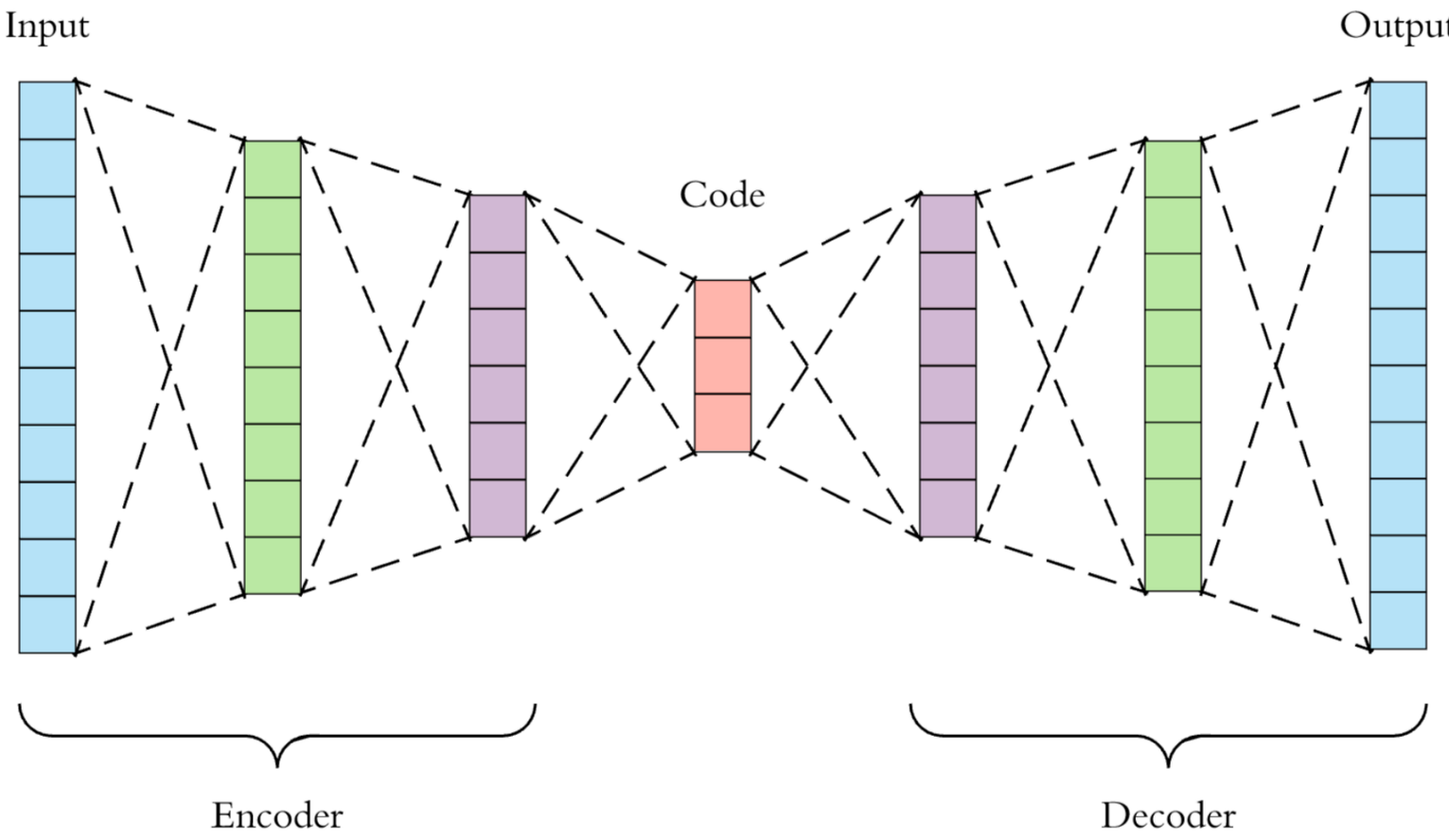
TransiNet




Encoder
Decoder

Input
Conv1
7x7 : 128
Conv2
5x5 : 256
Conv3
5x5 : 512
Conv3_1
3x3 : 512
Conv4
3x3 : 1024
Conv4_1
3x3 : 1024
Conv5
3x3 : 1024
Conv5_1
3x3 : 1024
Conv6
3x3 : 2048
Conv6_1
3x3 : 2048
L1 Loss
Output
Upconv0
4x4 : 32
Upconv1
4x4 : 64
Upconv2
4x4 : 128
Upconv3
4x4 : 256
Upconv4
4x4 : 512
Upconv5
4x4 : 1024

E.g. De-noising



Encoder/decoder usage

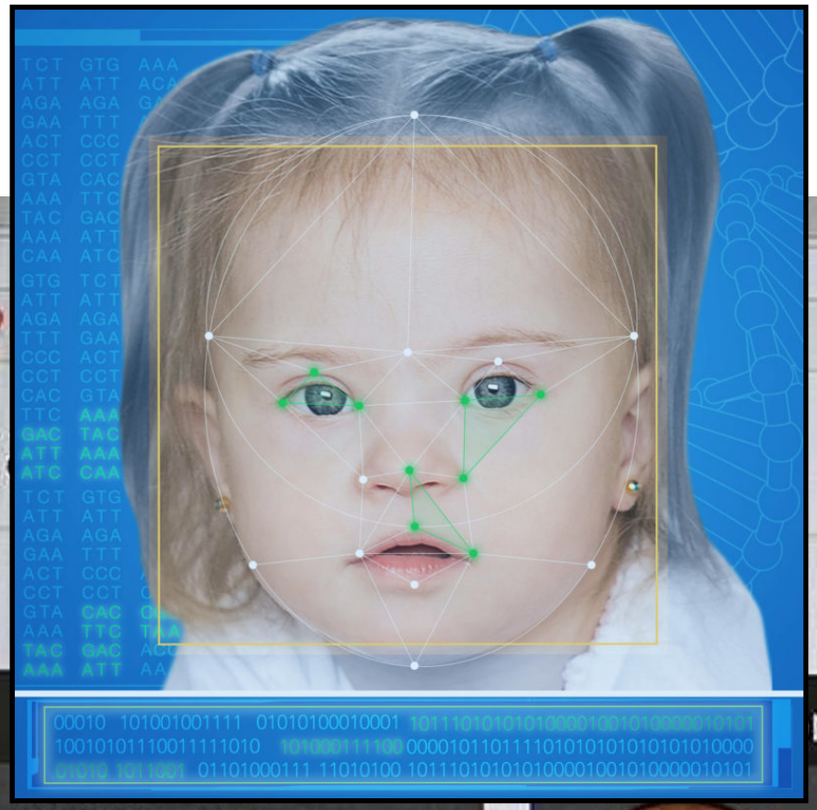
- Dimensionality reduction
- Denoising
- Outlier detection (measure of reconstruction error)
- Predicting next move
- Many others…
- (Can be considered as an unsupervised learning)

Secure | https://app.face2gene.com/case/286359

Insurance Portals | Genetics links | Misc | OBGYN links

Tufts Medic

SUGGESTED SYNDROMES (30)



Down Syndrome

HIGH
MED
LOW
GESTALT
FEATURE

- Differential
- Clinically Diagnosed
- Molecularly Diagnosed

me 2q37 Deletion Syndrome

HIGH
MED
LOW
GESTALT
FEATURE

- Differential
- Clinically Diagnosed
- Molecularly Diagnosed

Kleefstra Syndrome

HIGH
MED
LOW
GESTALT
FEATURE

- Differential
- Clinically Diagnosed
- Molecularly Diagnosed

Cornelia De Lange Syndrome

HIGH
MED
LOW
GESTALT
FEATURE

- Differential
- Clinically Diagnosed
- Molecularly Diagnosed

Angelman Syndrome; AS

HIGH

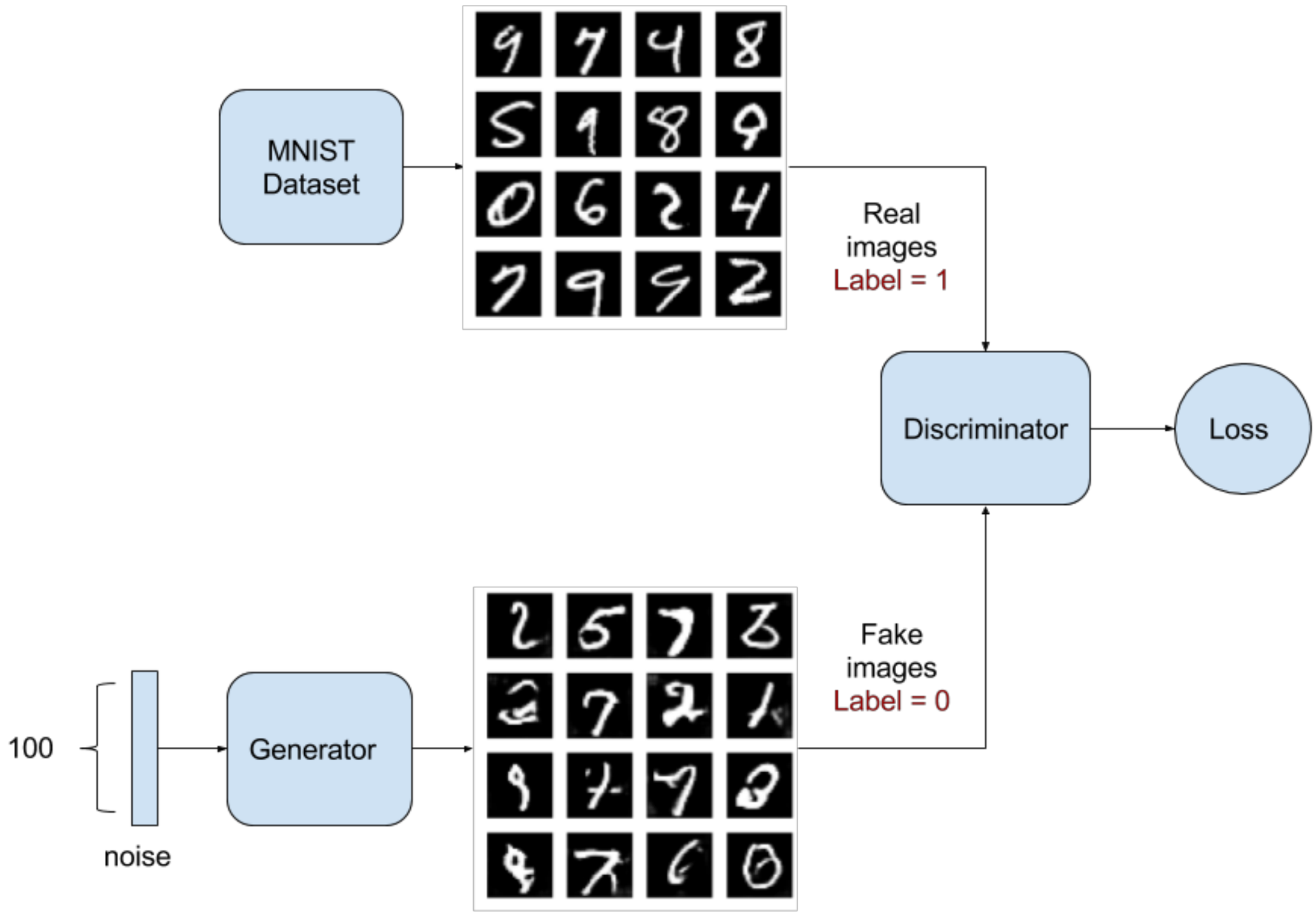
- Differential

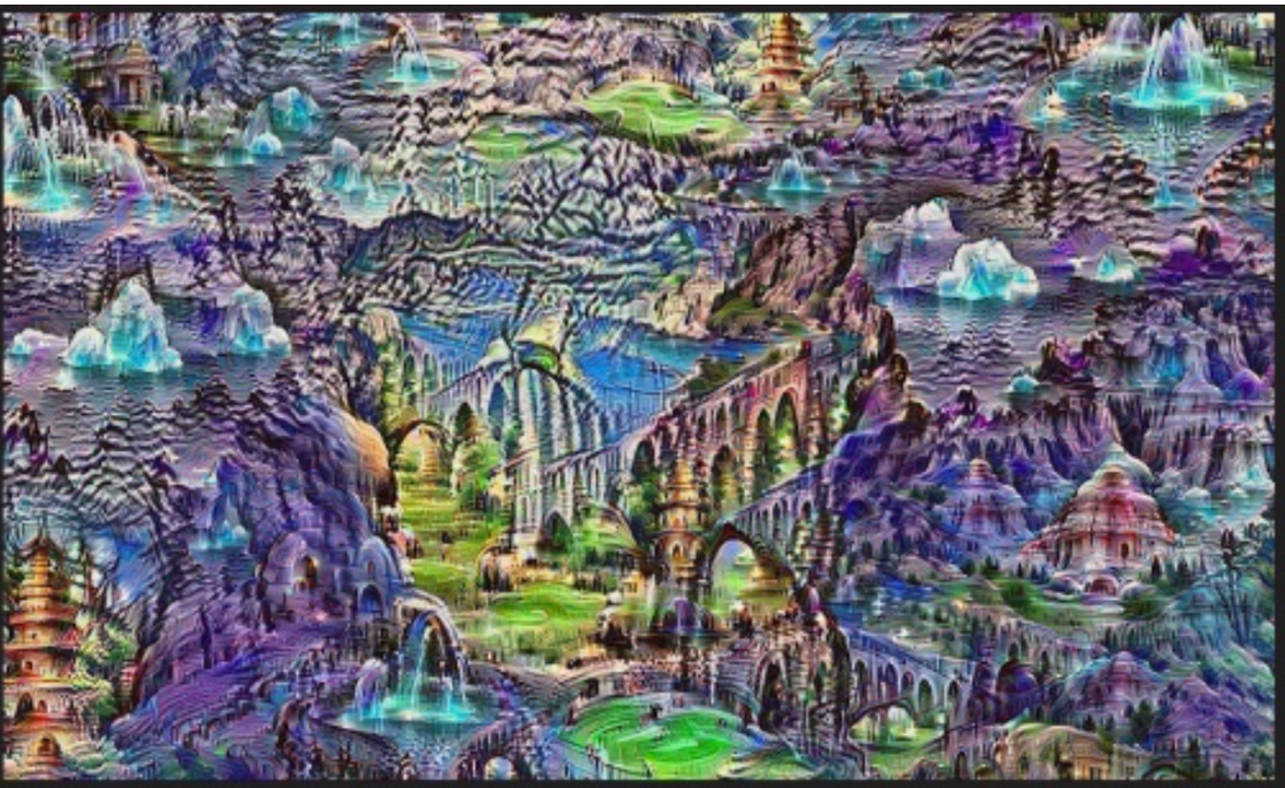
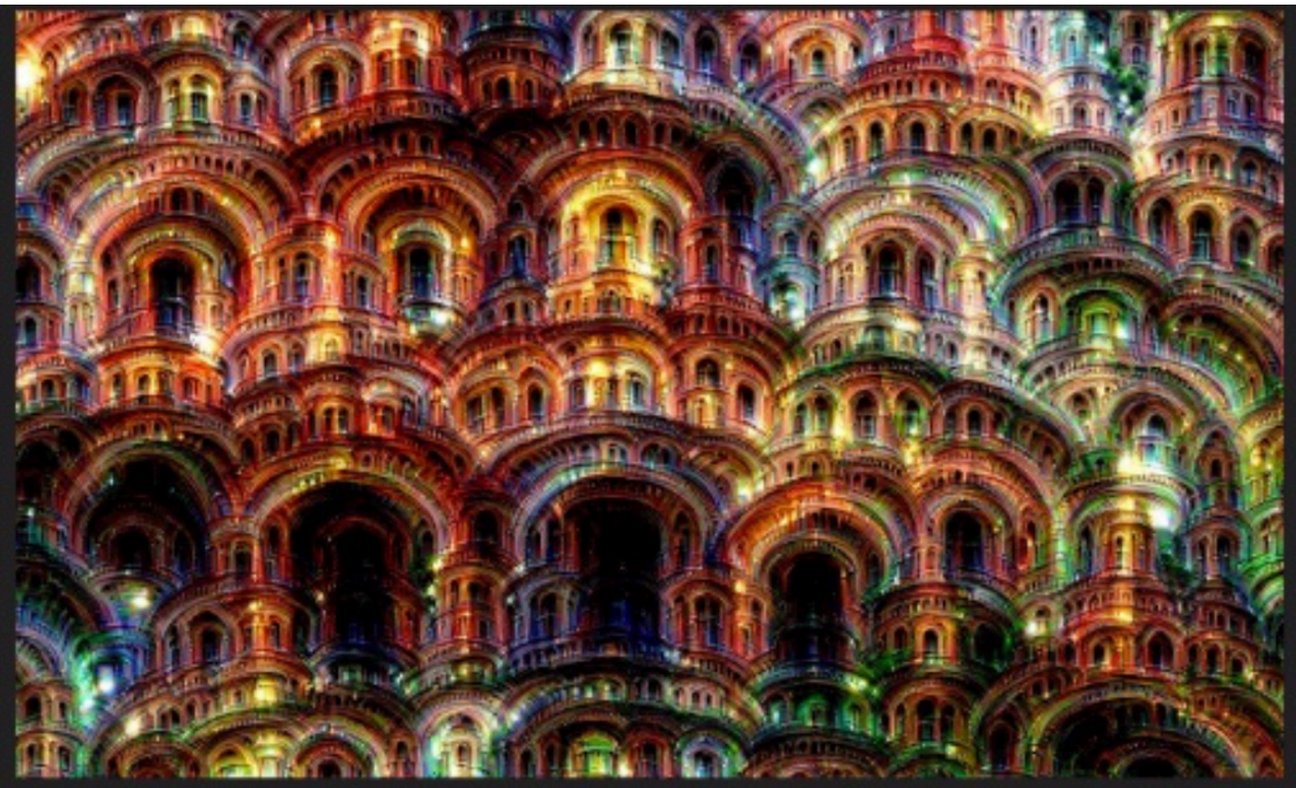
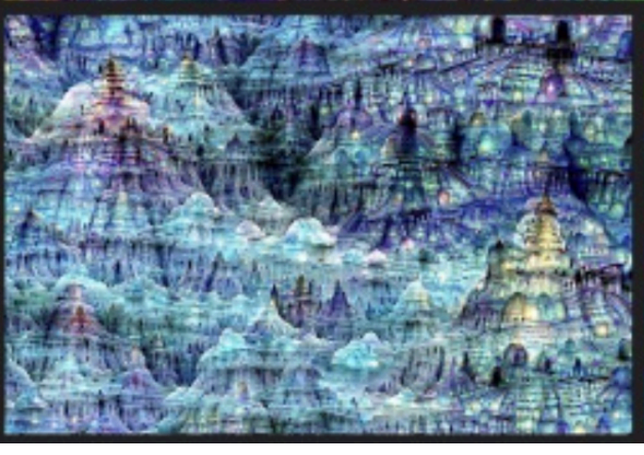
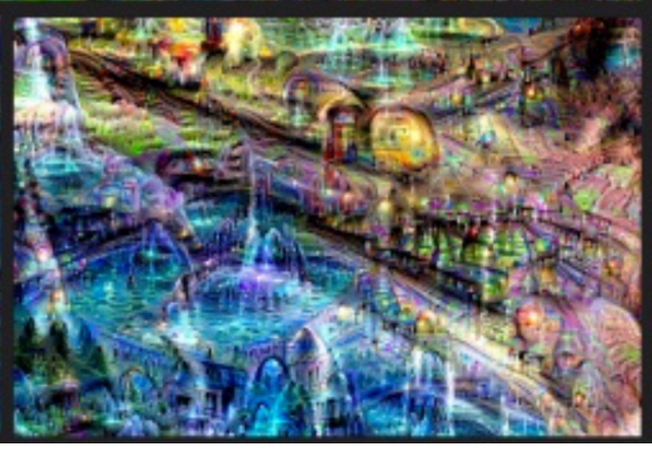
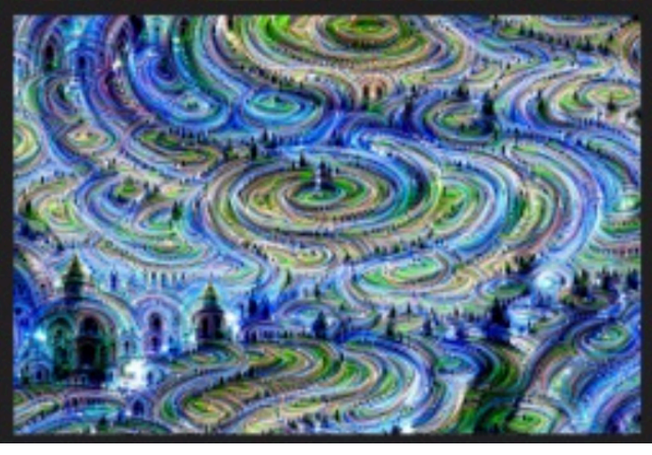
Chromosome 1p36 Deletion Syndrome

HIGH

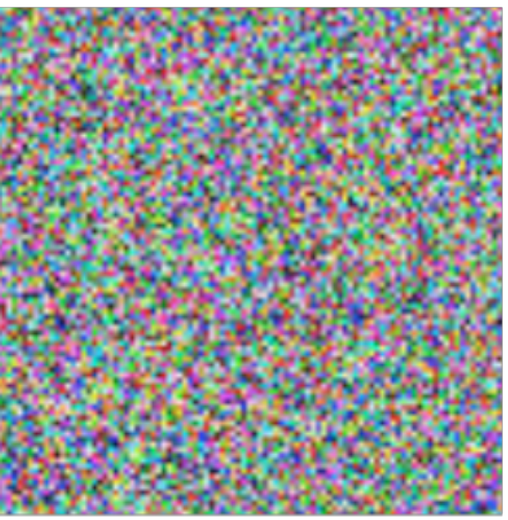
- Differential

GAN (Generative Adversarial Network)




Noise ~ N(0,1)

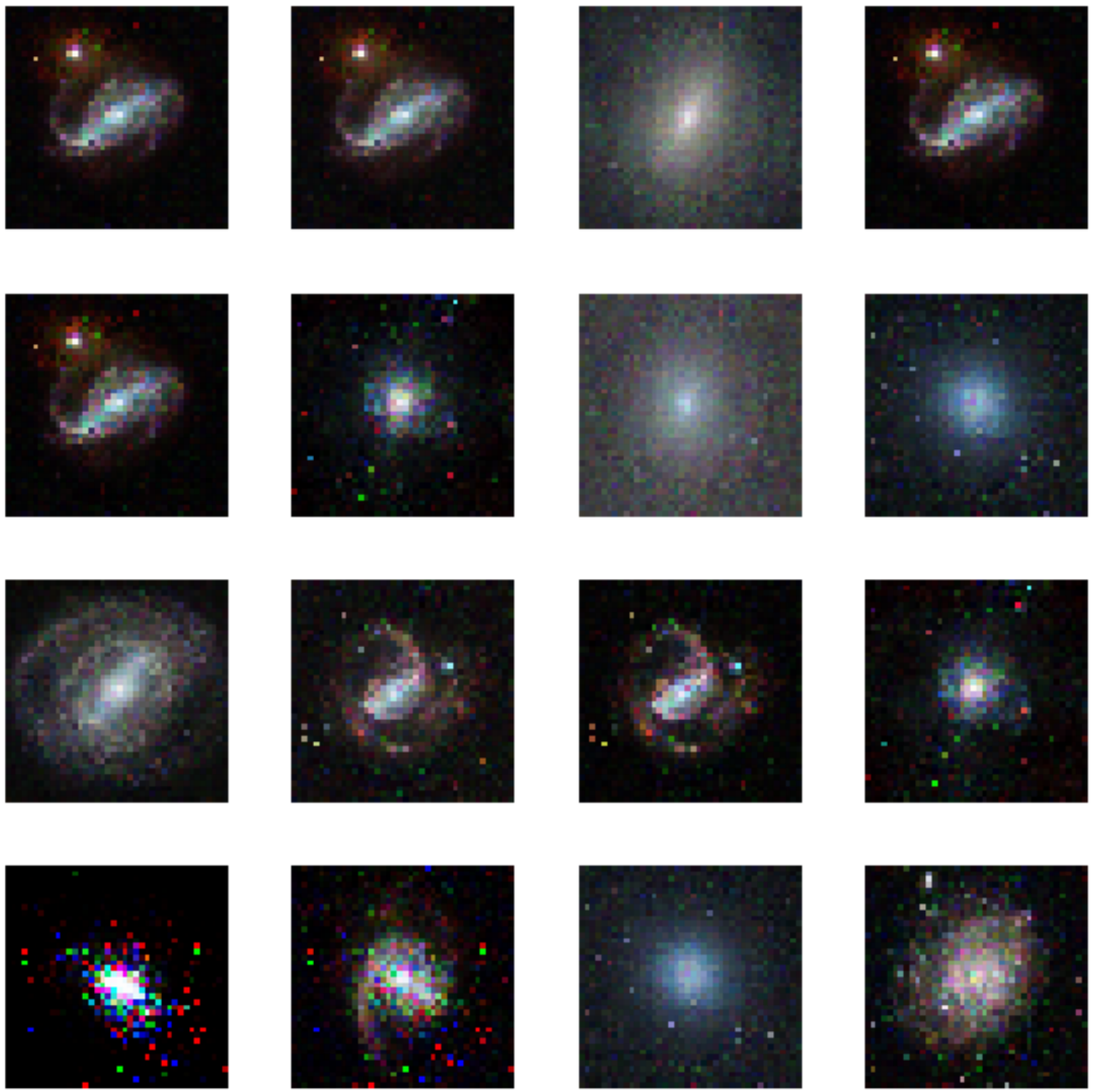


Generative Model

Elliptical

Barred Spiral

Spiral



GAN usage

- Mock catalogs
- Denoising
- Compete gaming

Logic behind  Keras

1- **Import** model class

from keras.models import **Sequential**

2 - **Instantiate** model class

model = Sequential()

3 - **Add layers** with the add() method specifying input_dim or input_shape

model.**add**(Dense(32, input_dim=784))

4 - Add activation functions

model.add(**Activation**('relu'))

5 - Configure **training** with compile(loss=,optimizer=, metrics[])

model.**compile**(optimizer='rmsprop', loss='binary_crossentropy',
metrics=['accuracy'])

6 - **Train** with the fit() method

model.**fit**(data, labels, epochs=10, batch_size=32)

7- **Evaluate** the model performance with the evaluate() method:

score = model.**evaluate**(x_test, y_test, verbose=0)

8 – Make **predictions** with predict():

predictions = model.**predict**(x_test)

in TensorFlow
in anaconda for python 3.6

Thanks, folks!

What's next?