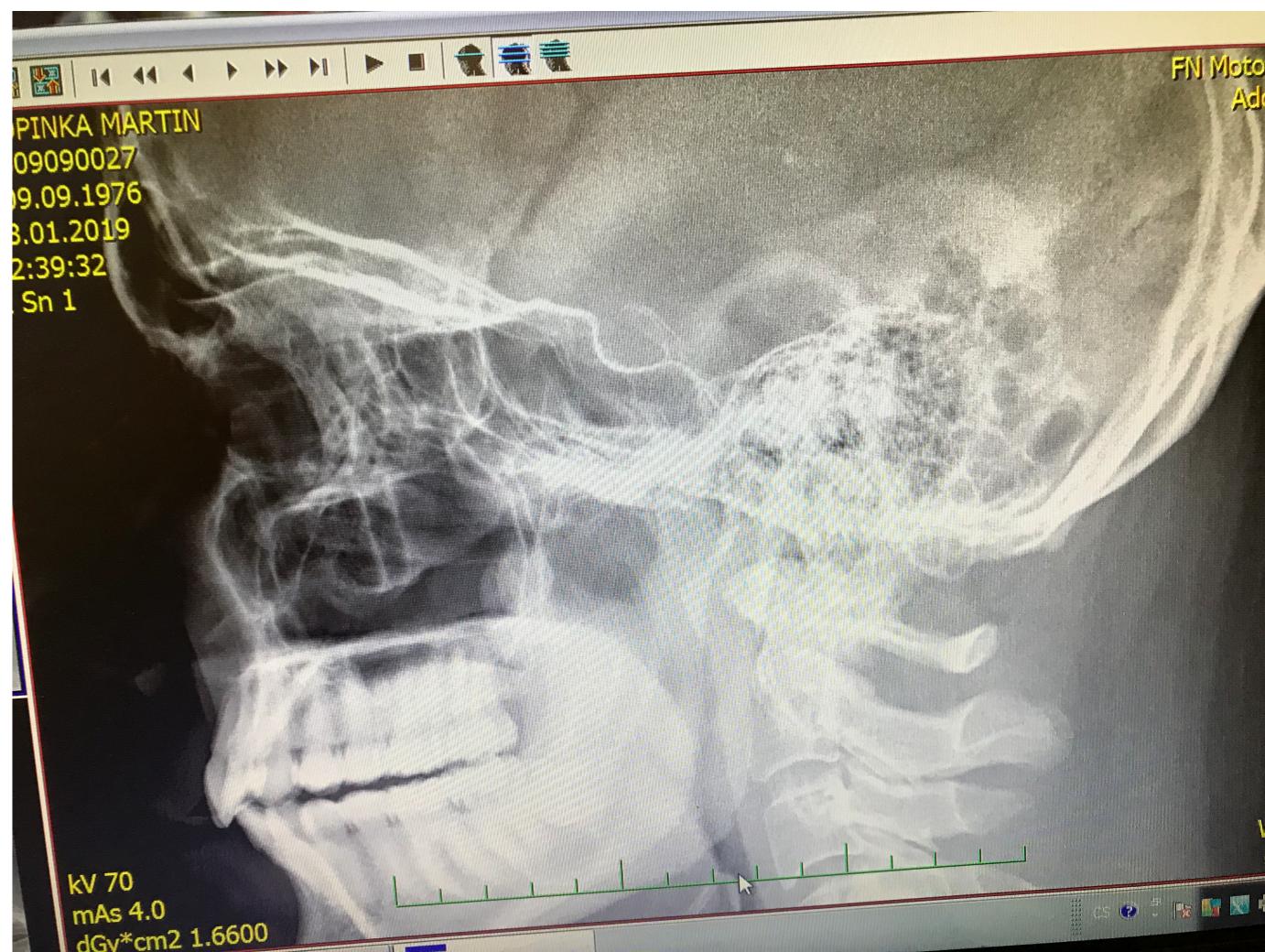


Neural Networks

DIAS ML course by Martin Topinka

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





Outline

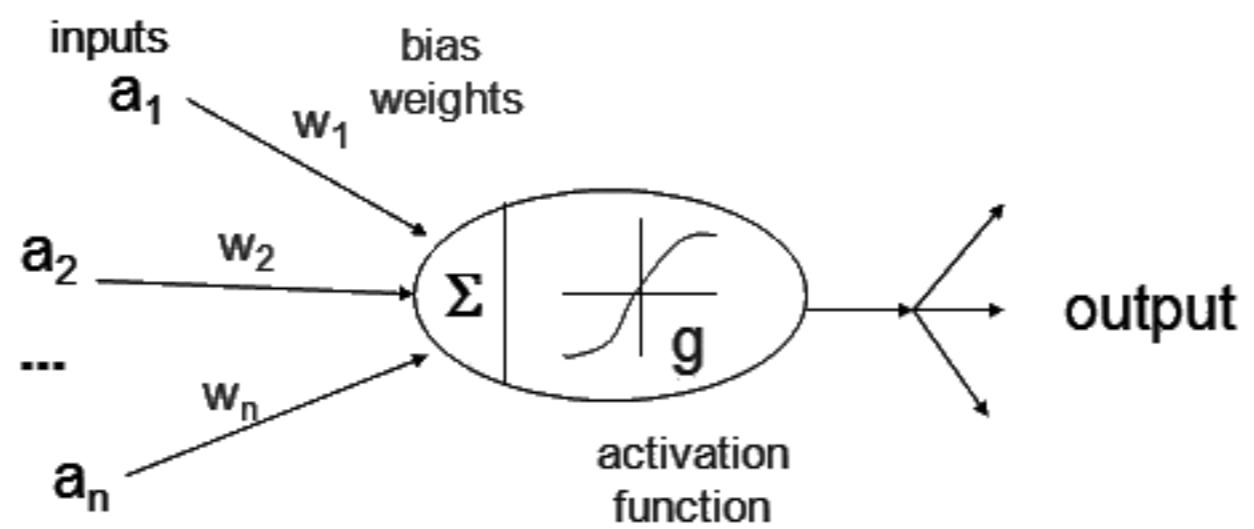
- Introduction to NN: the idea behind, SGD, vanishing gradient, fighting over-fitting)
- Convolutional Neural Network
- Encoder-decoders
- GANs
- TensorFlow/Keras
- Hands-on session: NN star/quasar classification, ConvNN source finding/denoising
- Not included: Reinforcement Learning

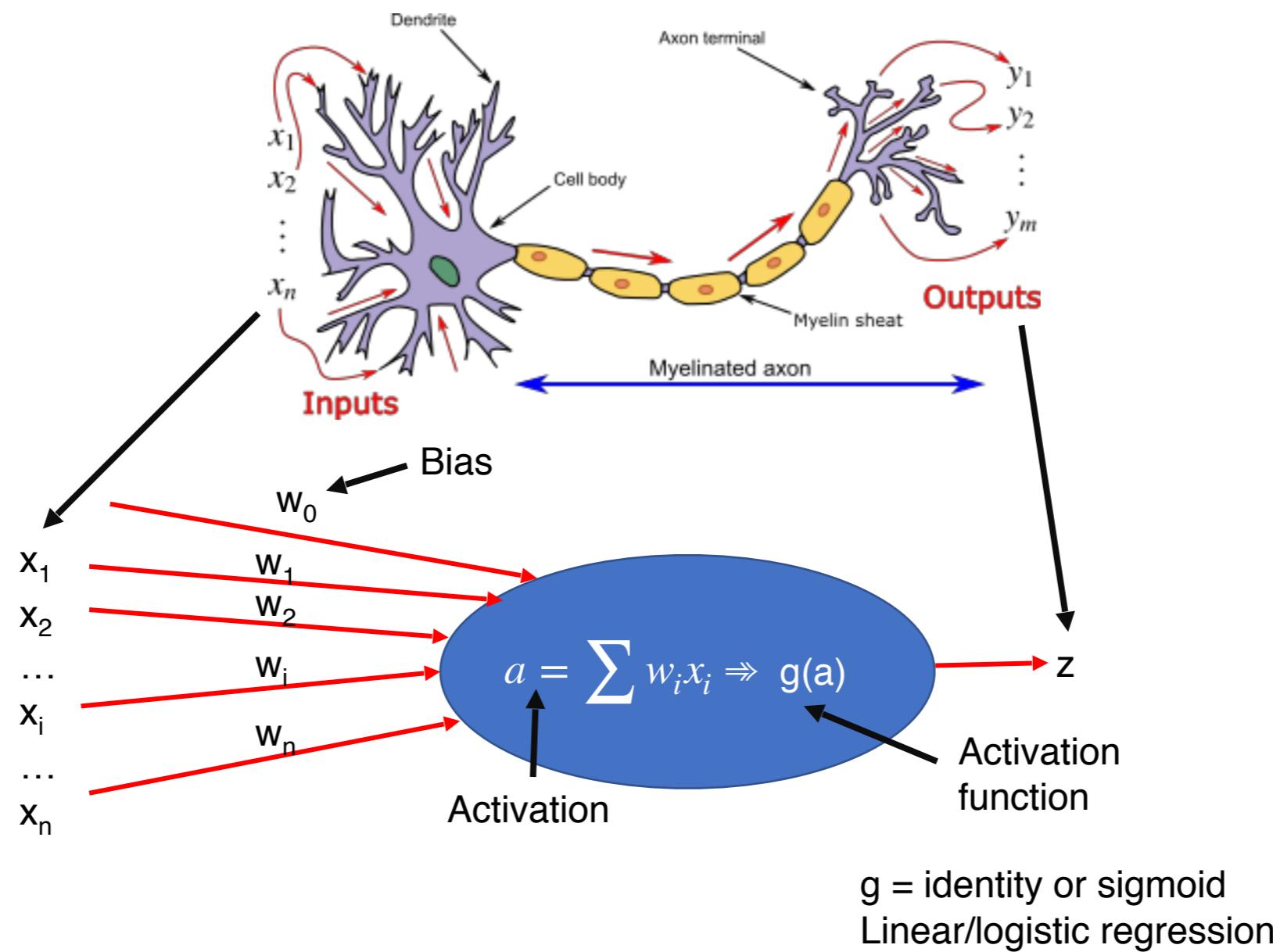
“Deep learning (neural networks) is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it....”

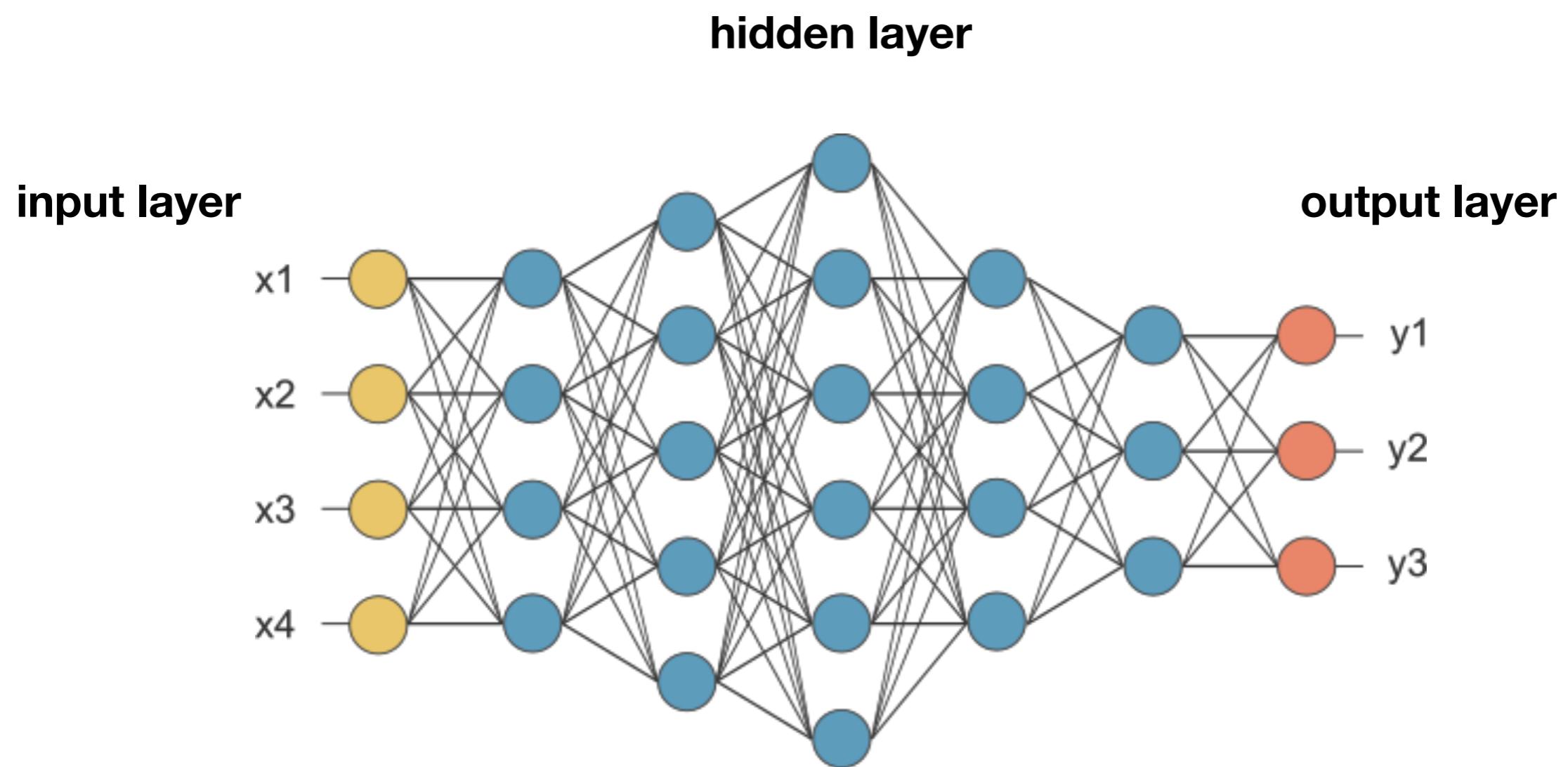
– a ML expert

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

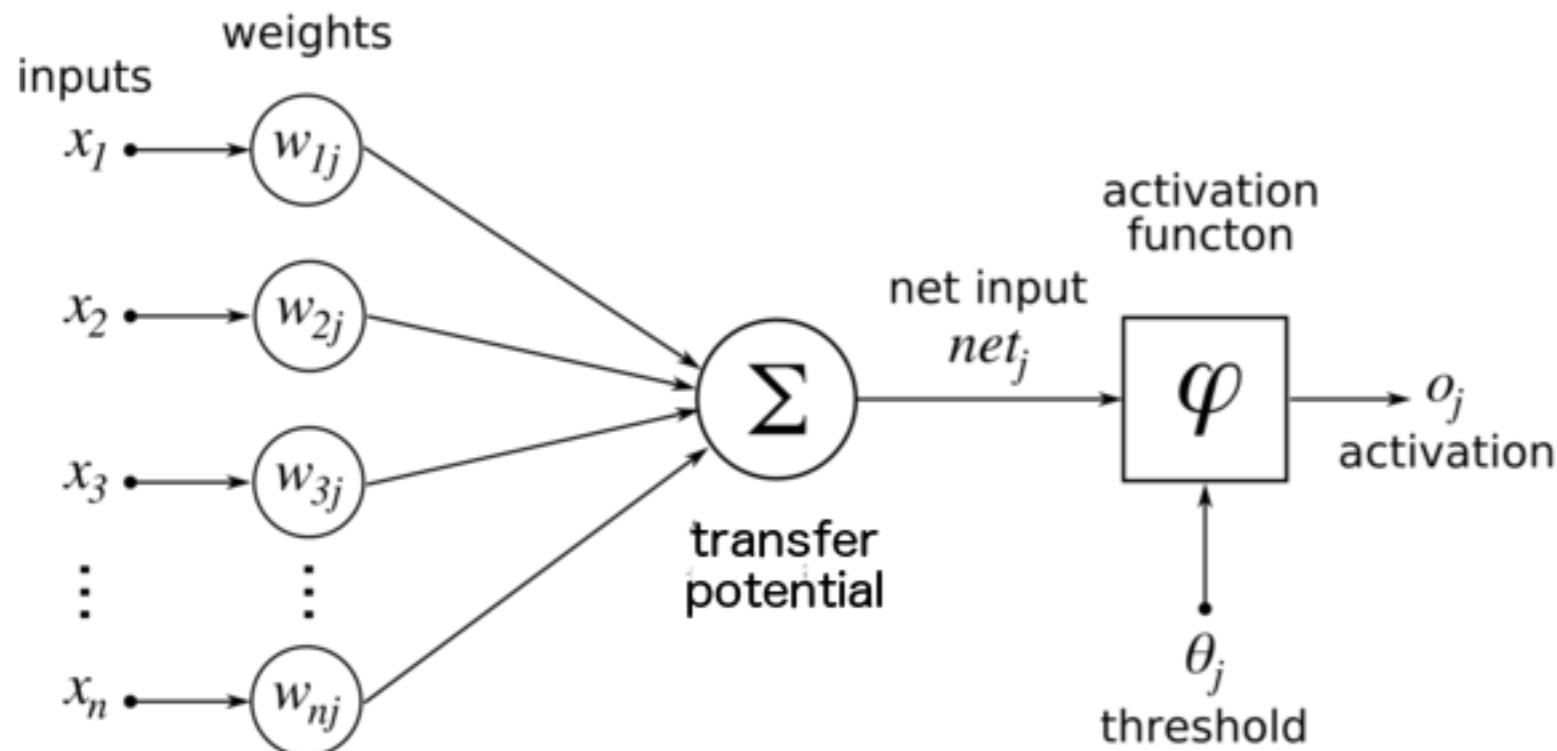






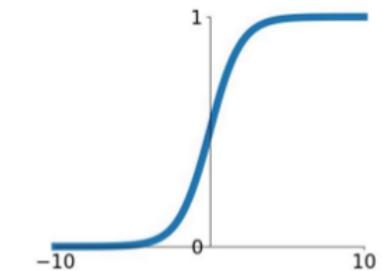


Activation Functions



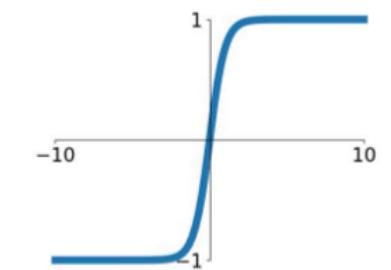
Sigmoid

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



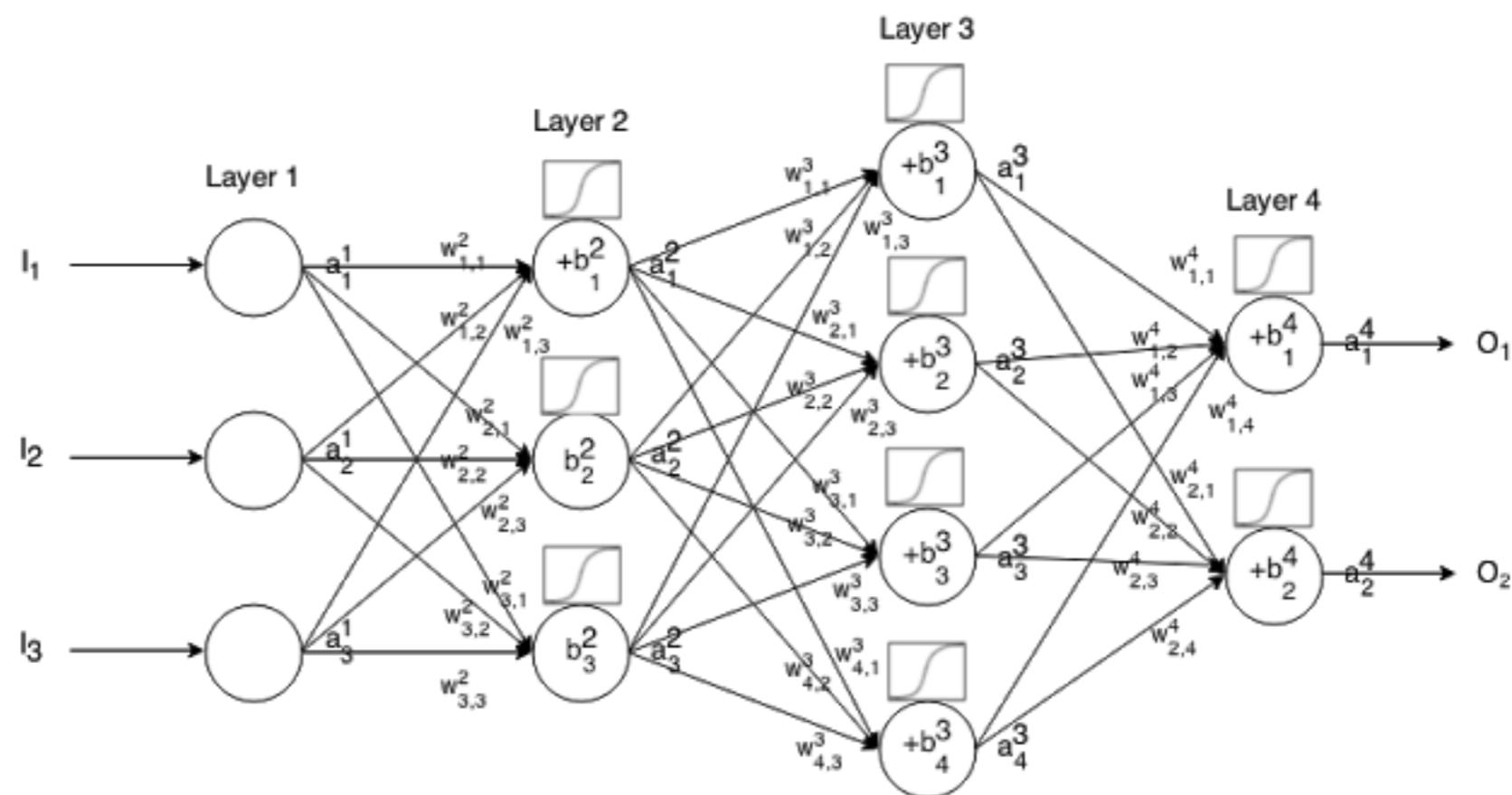
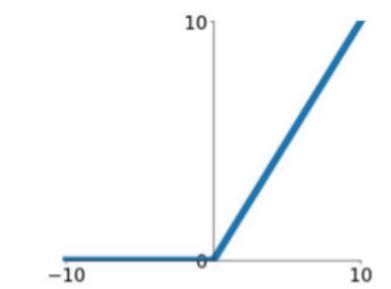
tanh

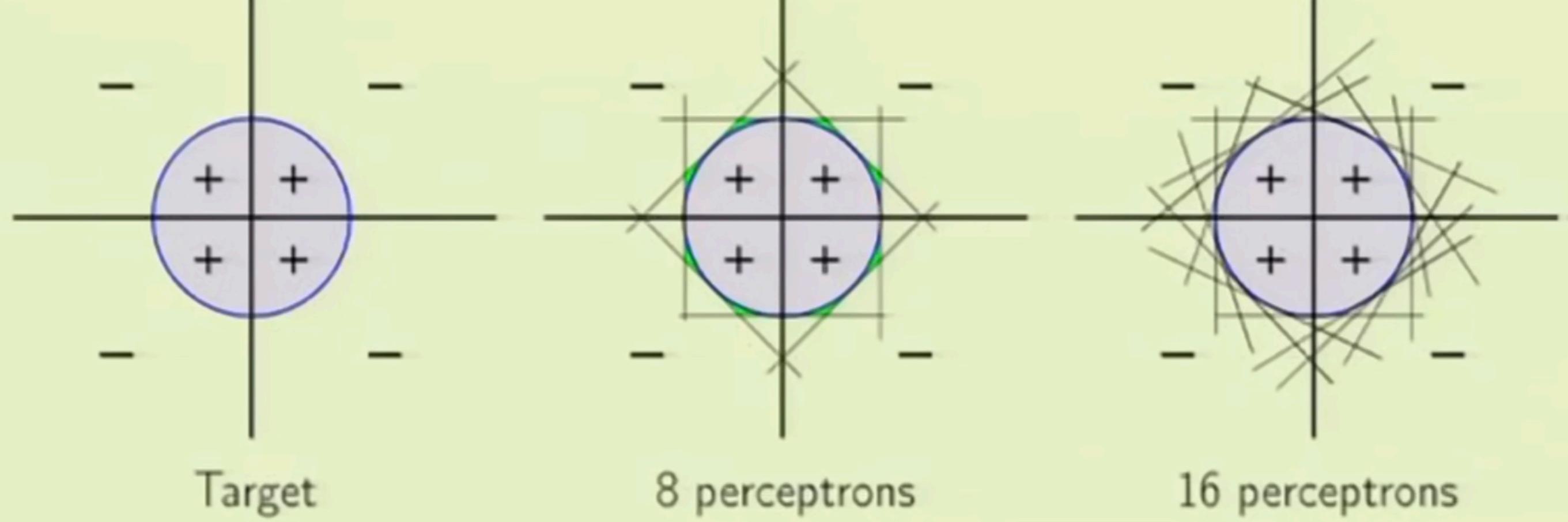
$$\tanh(x)$$



ReLU

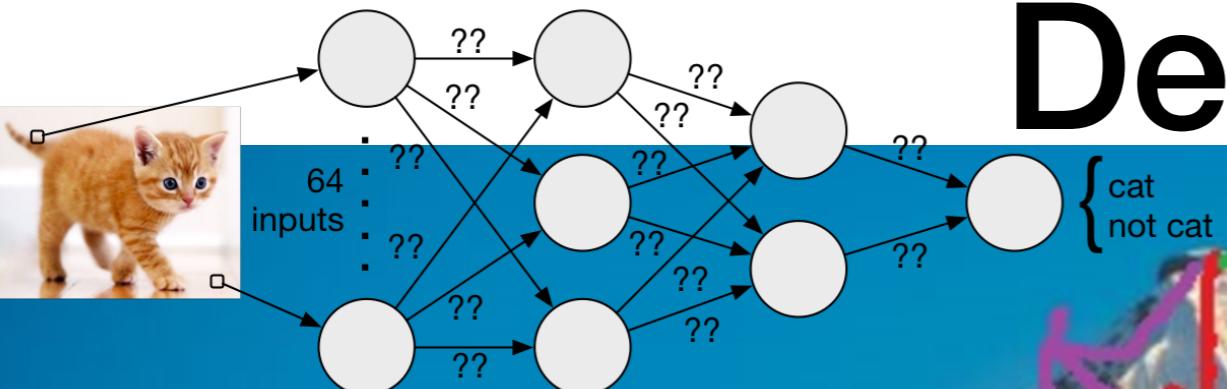
$$\max(0, x)$$



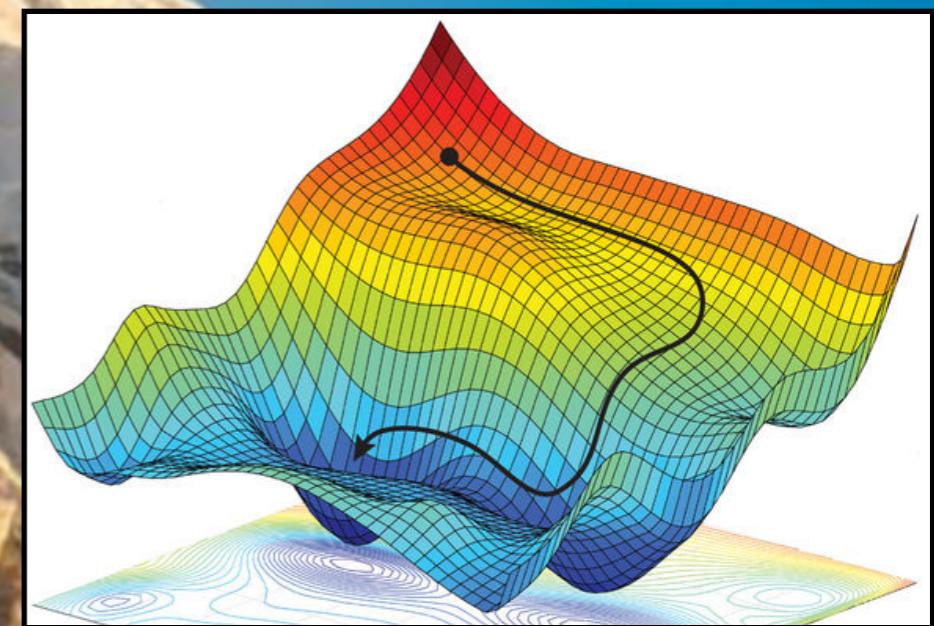
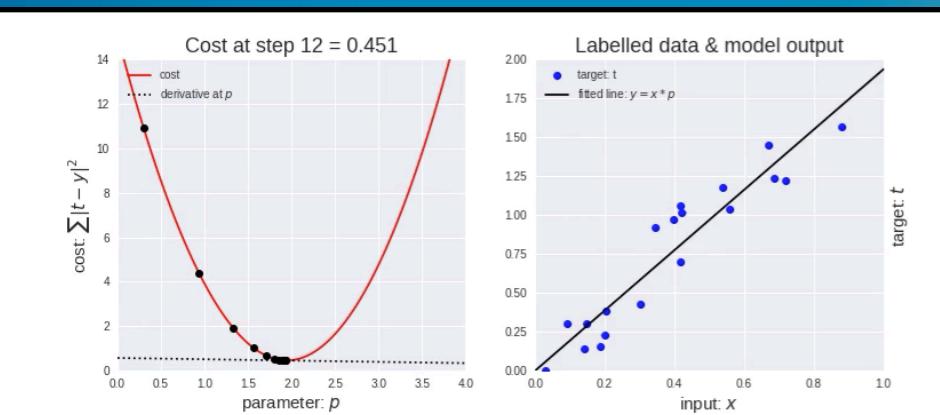


Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$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

(Stochastic) Gradient Descent



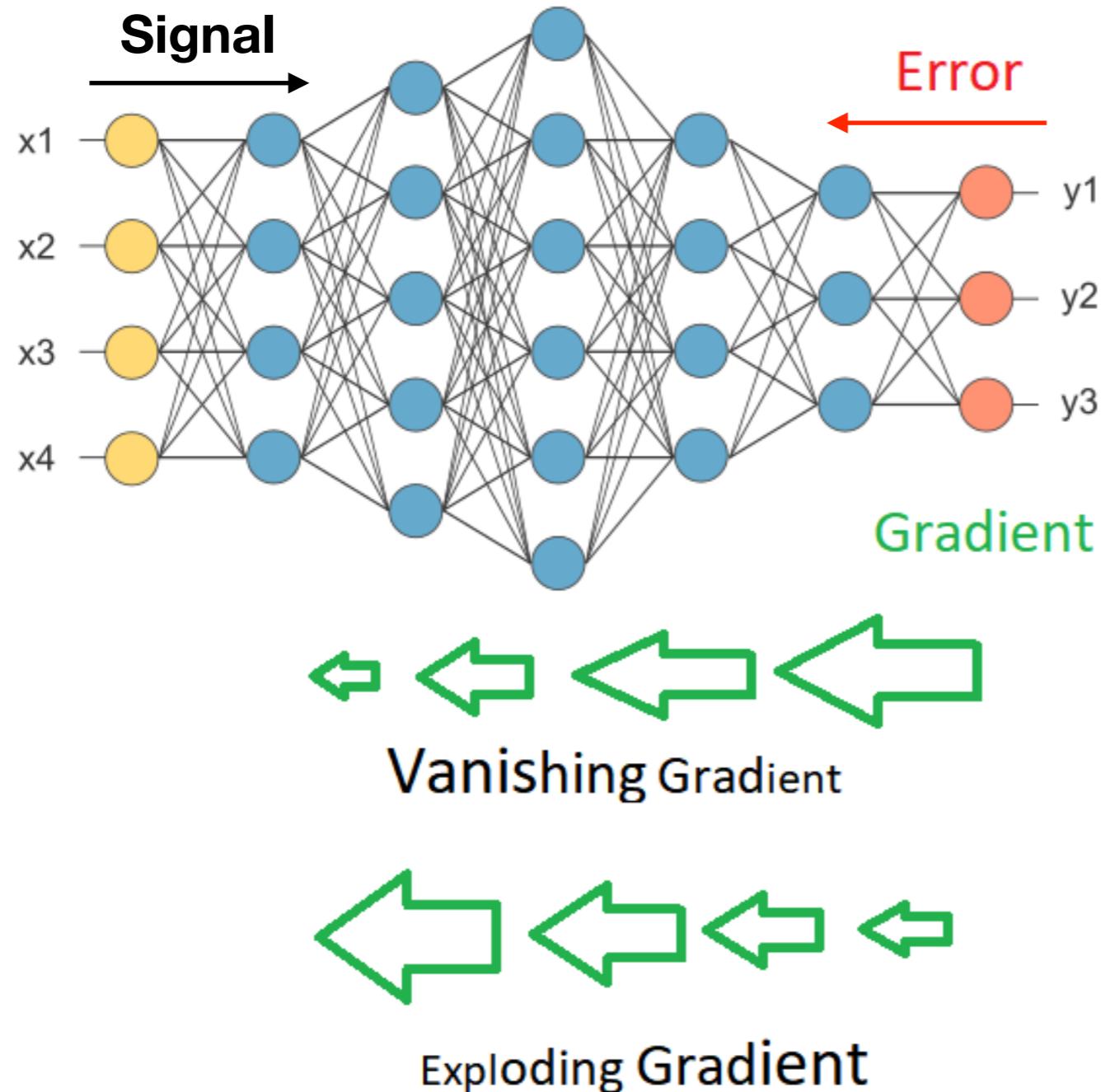
$$w = w - \alpha \nabla_w J(w)$$



Back-propagation

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

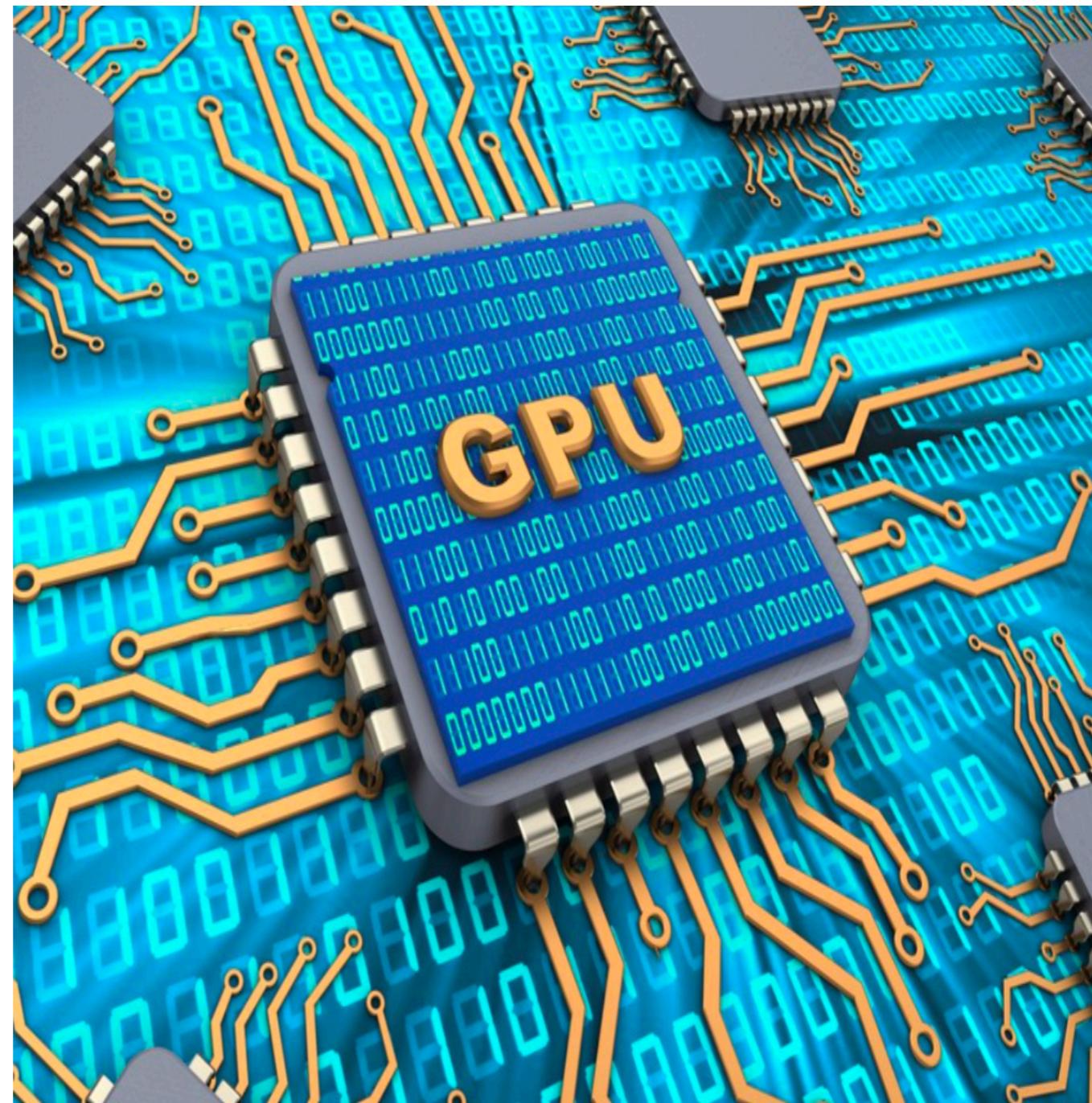
Back-propagation: apply chain rule to gradient of loss function to **adjust weights/biases**



Remedies against vanishing gradient:

- ReLU
- Re-normalisation

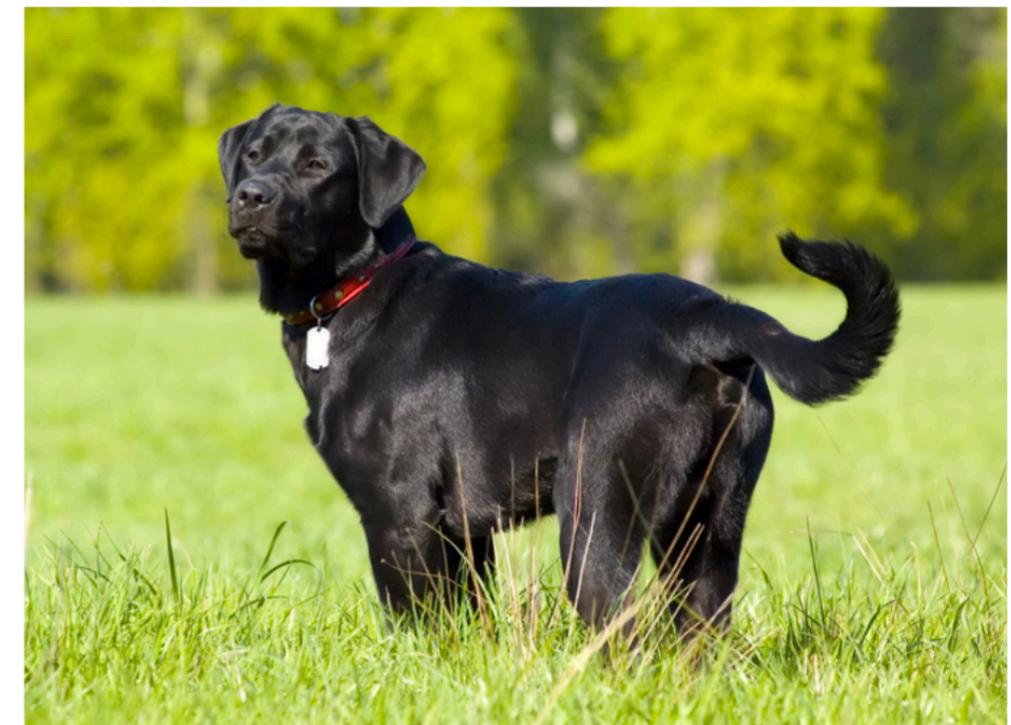
**Algorithms + Computational Resources + Available data = Boom of ML
(specially deep learning)**



Small Dataset



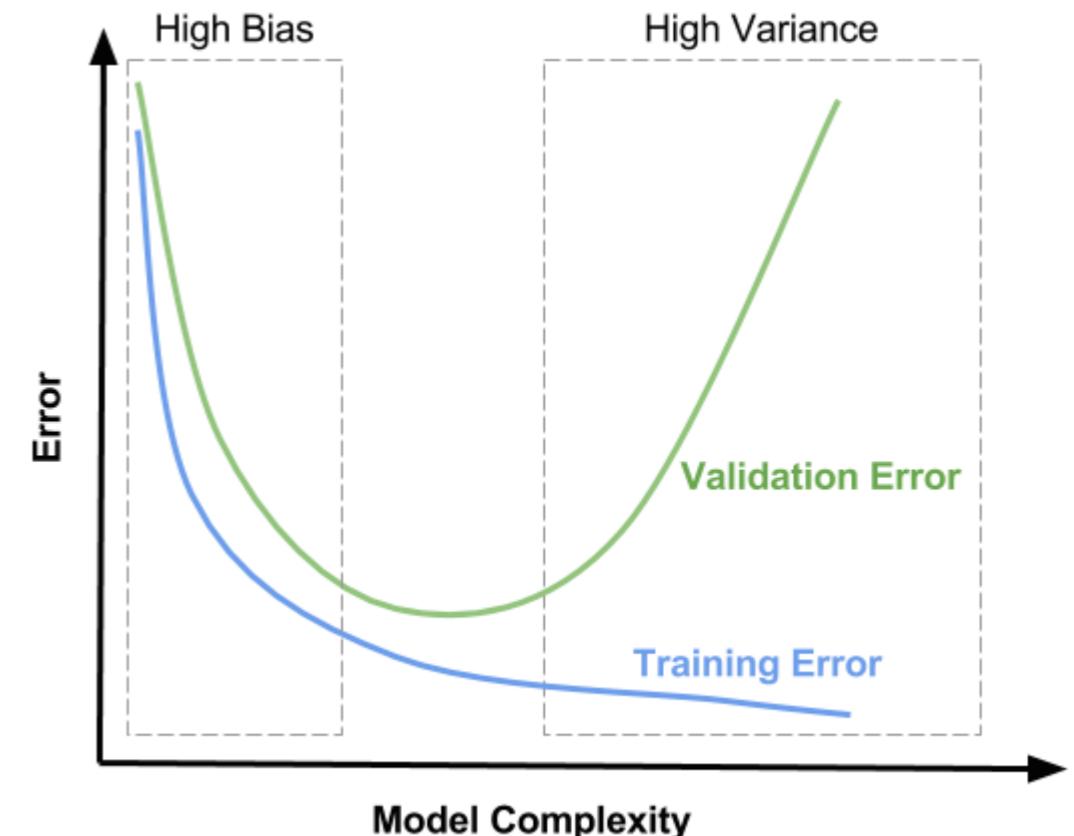
over-fitting



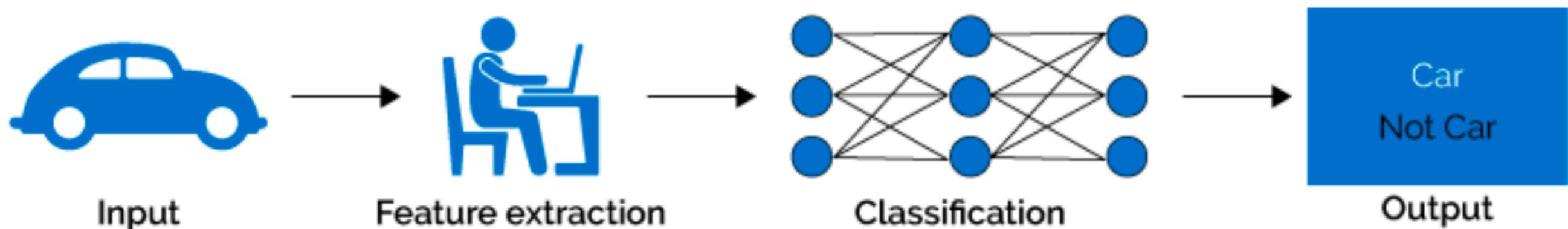
It's a cat

Fighting over-fitting

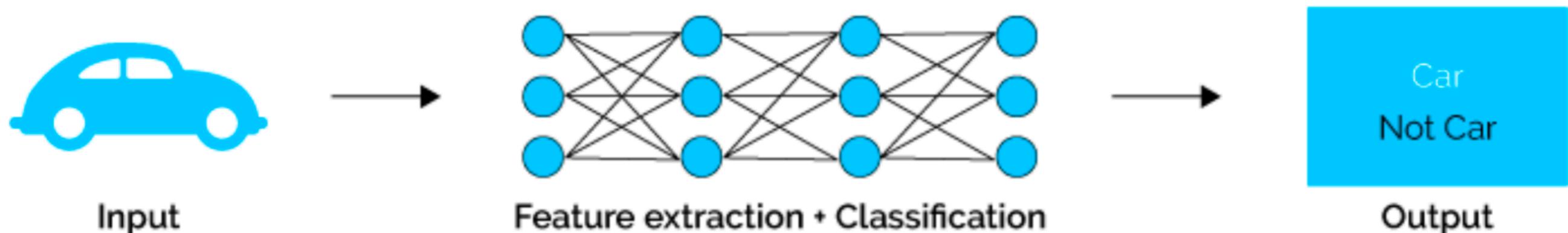
- Dropout
- Regularisation
- Early stopping
- Simplify the architecture
- More data (even if you have to make it up: translate, rotate, flip, crop, lighten/darken, add noise)
of weights ~ VC dimension ~ # degrees of freedom



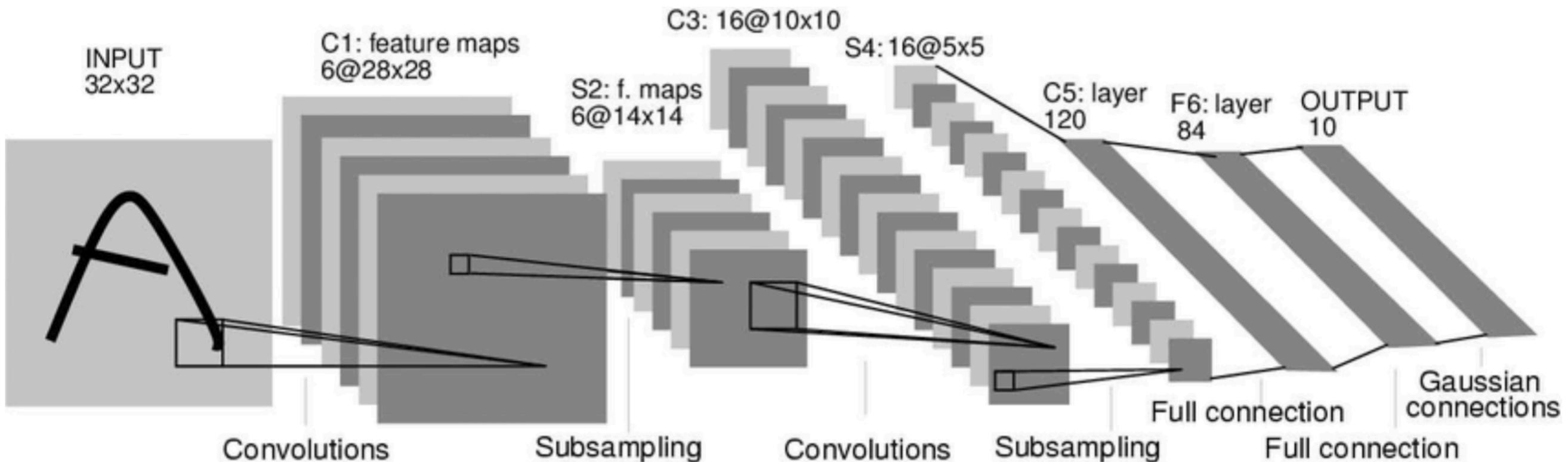
Machine Learning

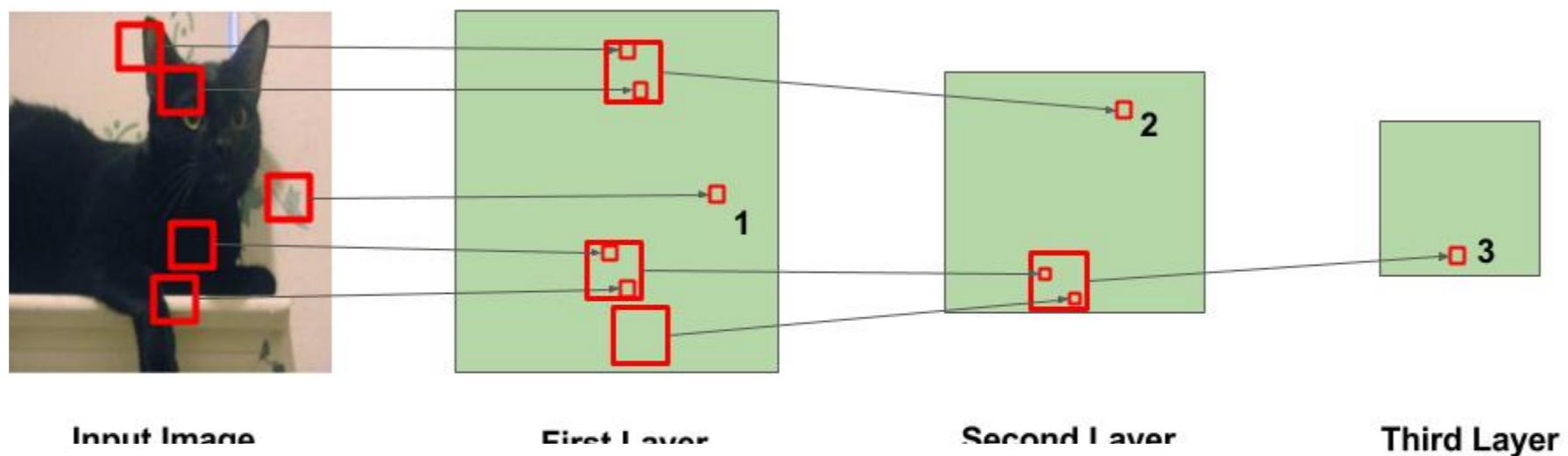


Deep Learning

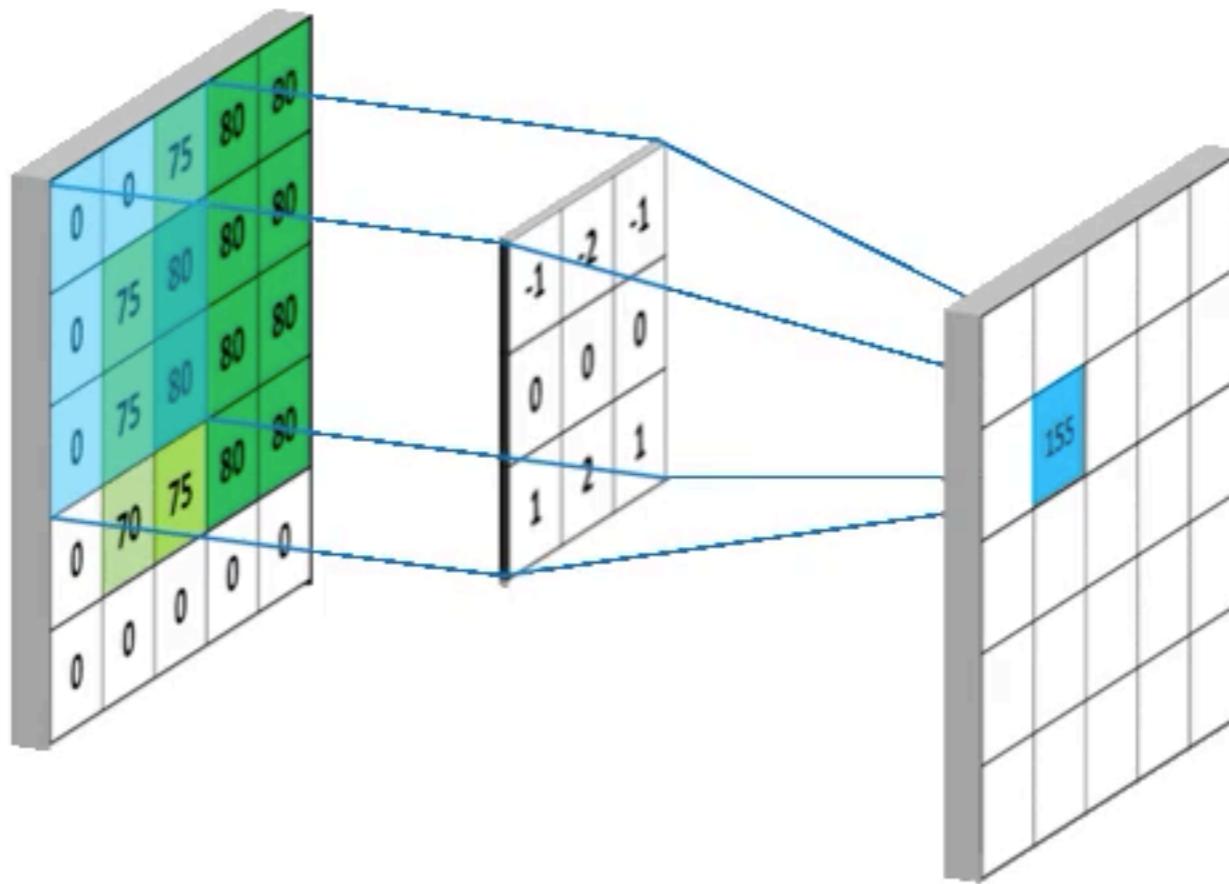


Convolutional Neural Networks





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



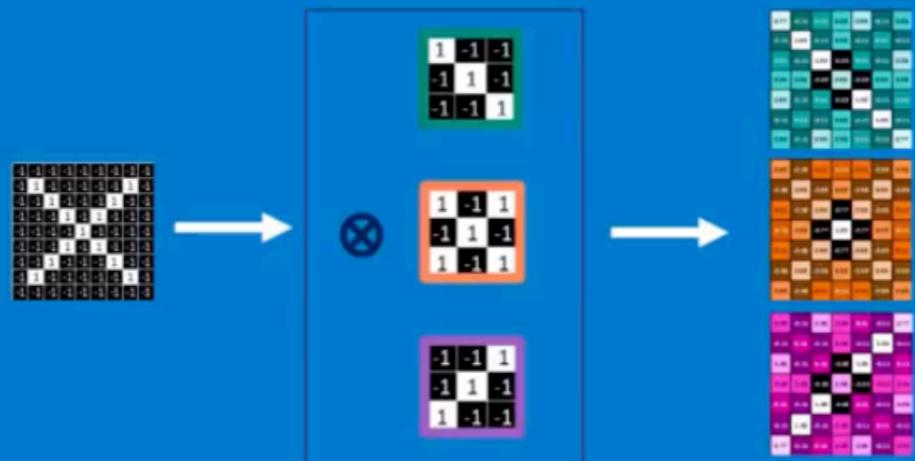
Input layer (source data)



- Dim reduction
- Spatial tolerance

Convolution layer

One image becomes a stack of filtered images



Pooling

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

max pooling

1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

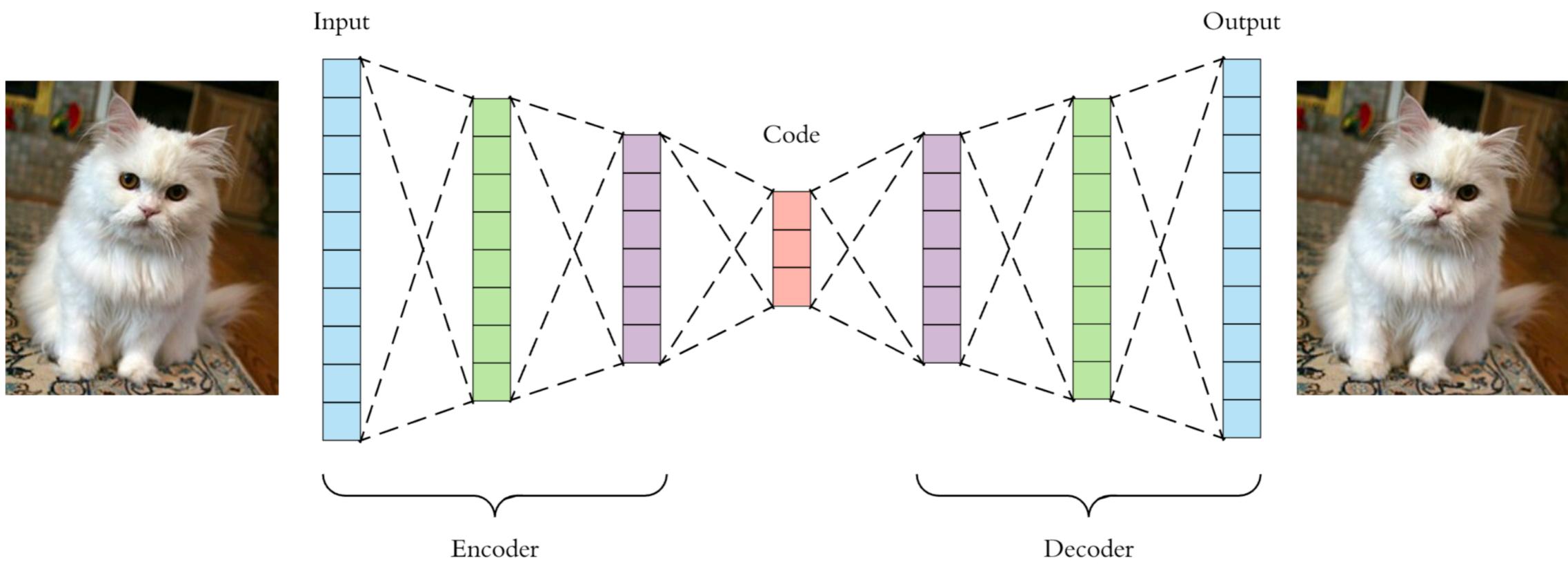
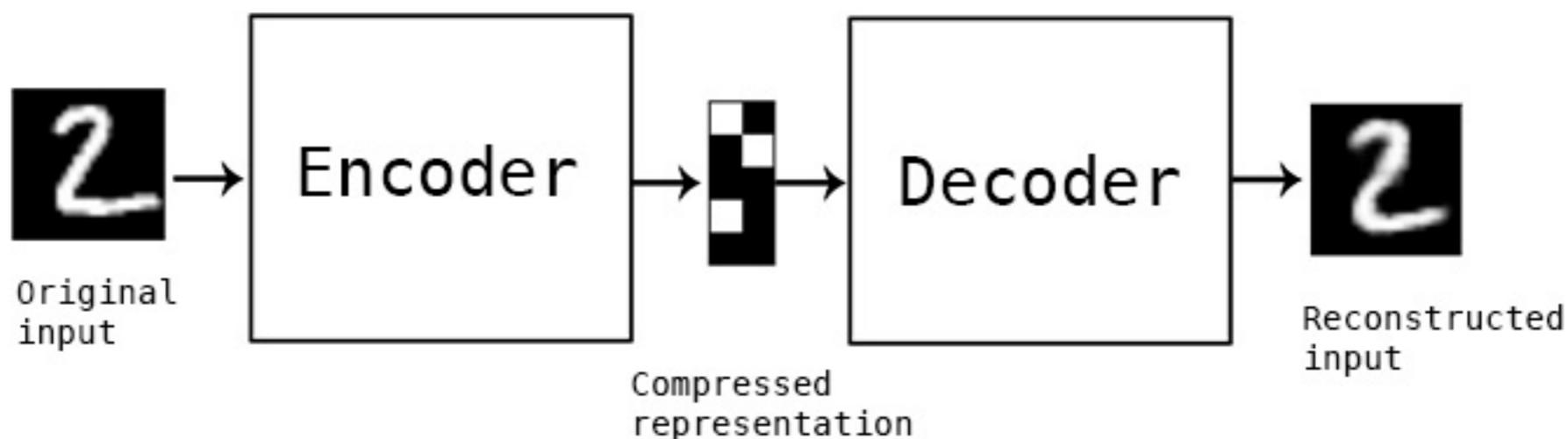
Repeat many times to build-up
hierarchically more complex patterns

Dense NN layer – "learning"

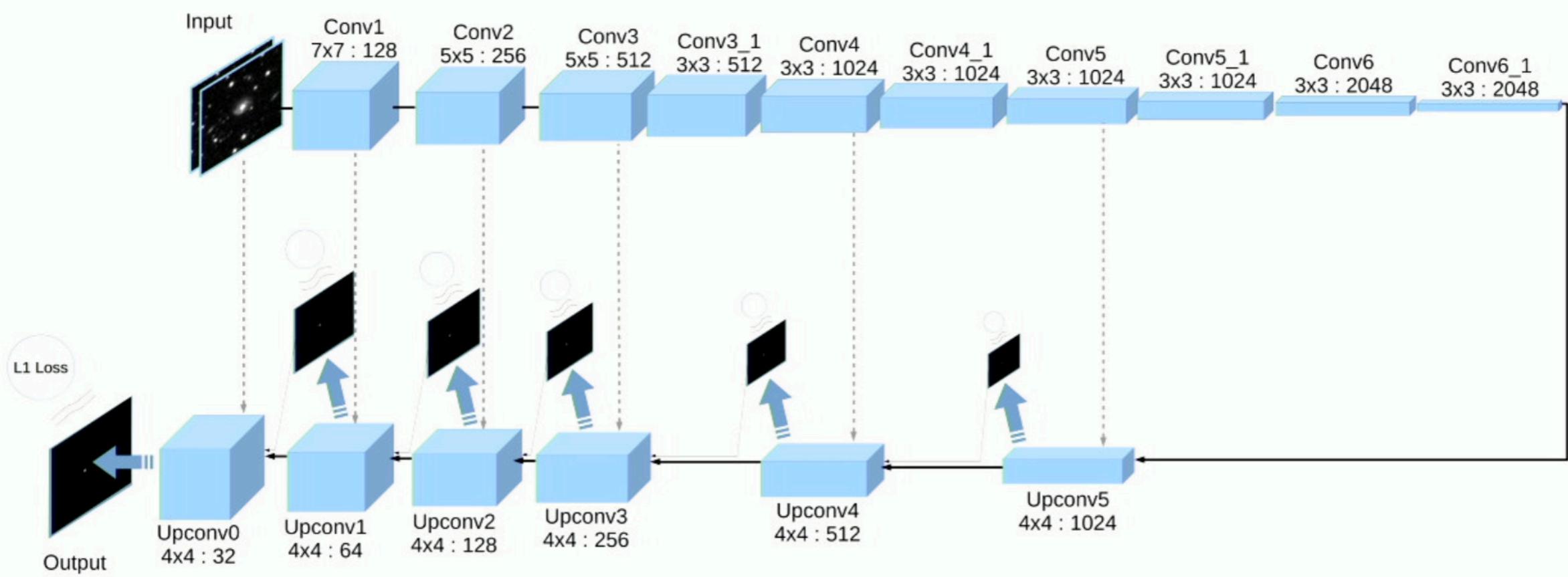
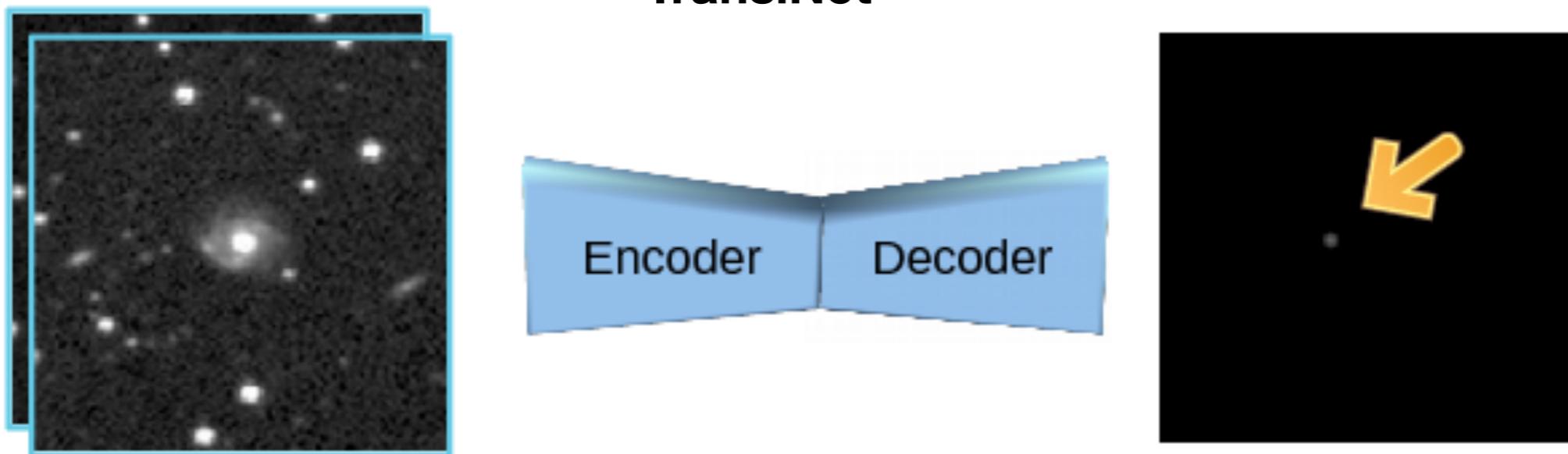
Automatic feature extraction!

Output layer
Calculate error to adjust weights

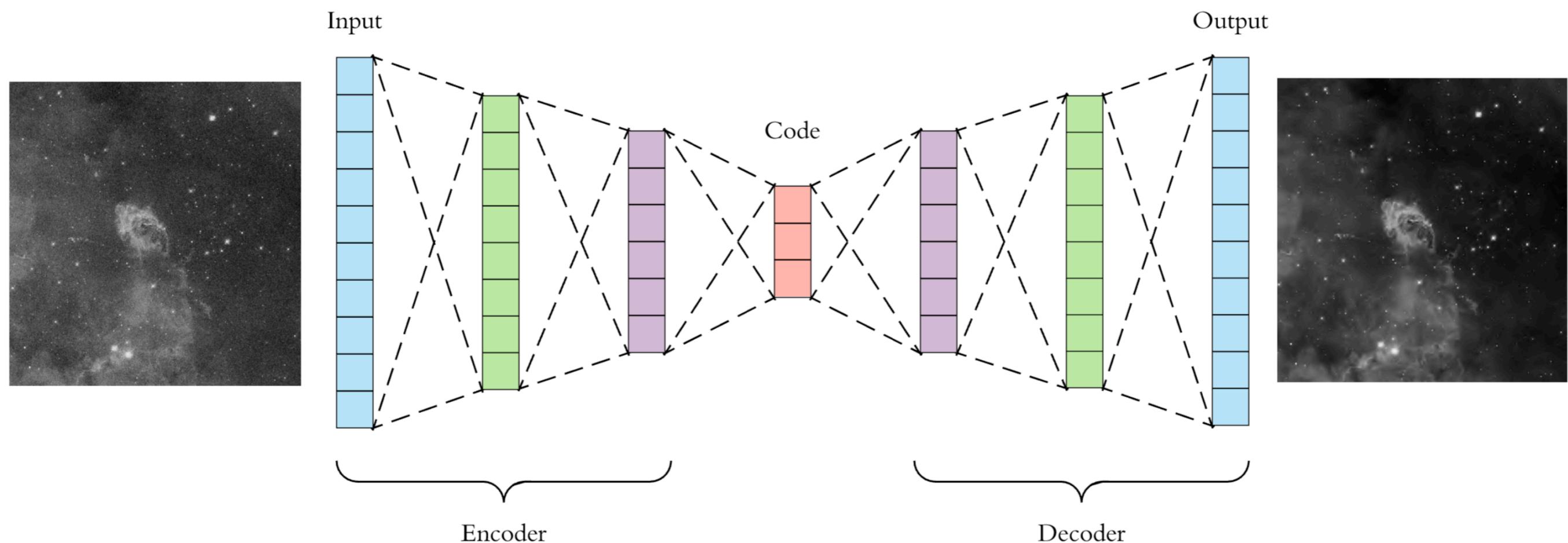
Encoder-decoders (Autoencoders)



TransiNet



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

abs Insurance Portals Genetics links Misc OBGYN links

Tufts Medic

SUGGESTED SYNDROMES (30) ^

Down Syndrome



GESTALT FEATURE

HIGH MED LOW

Differential
 Clinically Diagnosed
 Molecularly Diagnosed

Prader-Willi Syndrome 2q37 Deletion Syndrome



GESTALT FEATURE

HIGH MED LOW

Differential
 Clinically Diagnosed
 Molecularly Diagnosed

Kleefstra Syndrome



GESTALT FEATURE

HIGH MED LOW

Differential
 Clinically Diagnosed
 Molecularly Diagnosed

Cornelia De Lange Syndrome



GESTALT FEATURE

HIGH MED LOW

Differential
 Clinically Diagnosed
 Molecularly Diagnosed

Angelman Syndrome; AS



HIGH

Differential

Chromosome 1p36 Deletion Syndrome



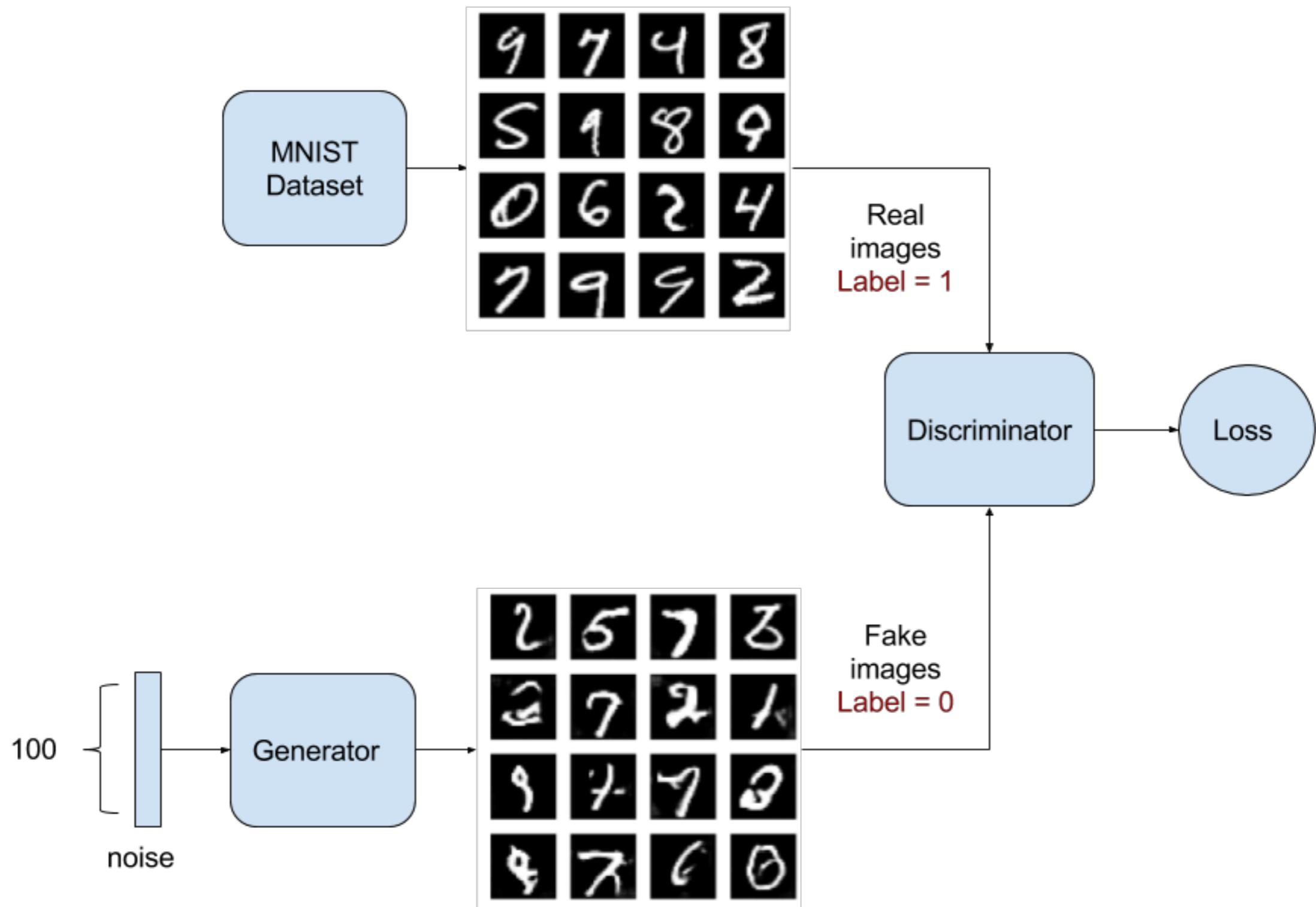
HIGH

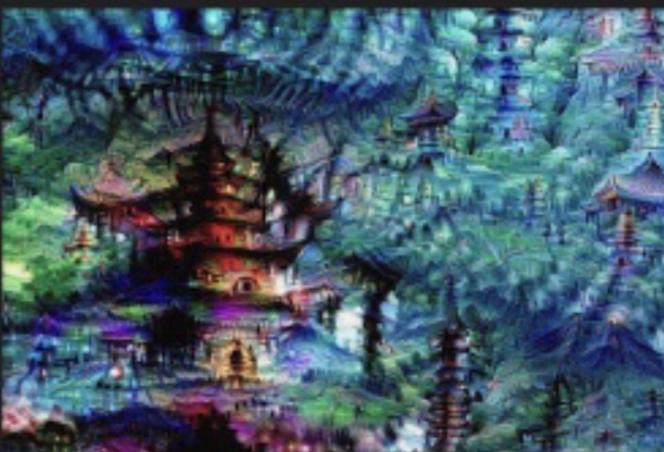
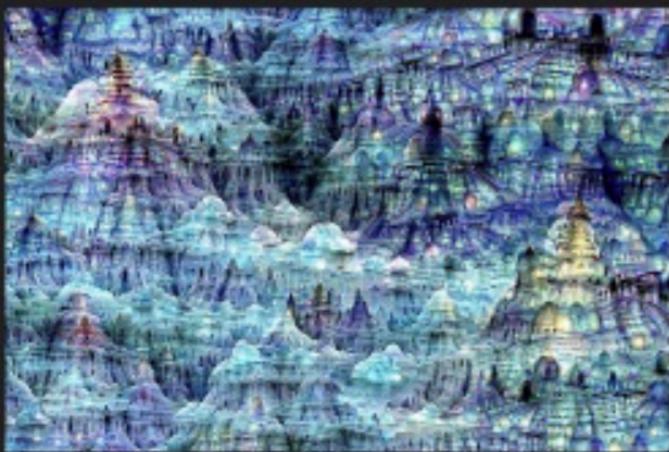
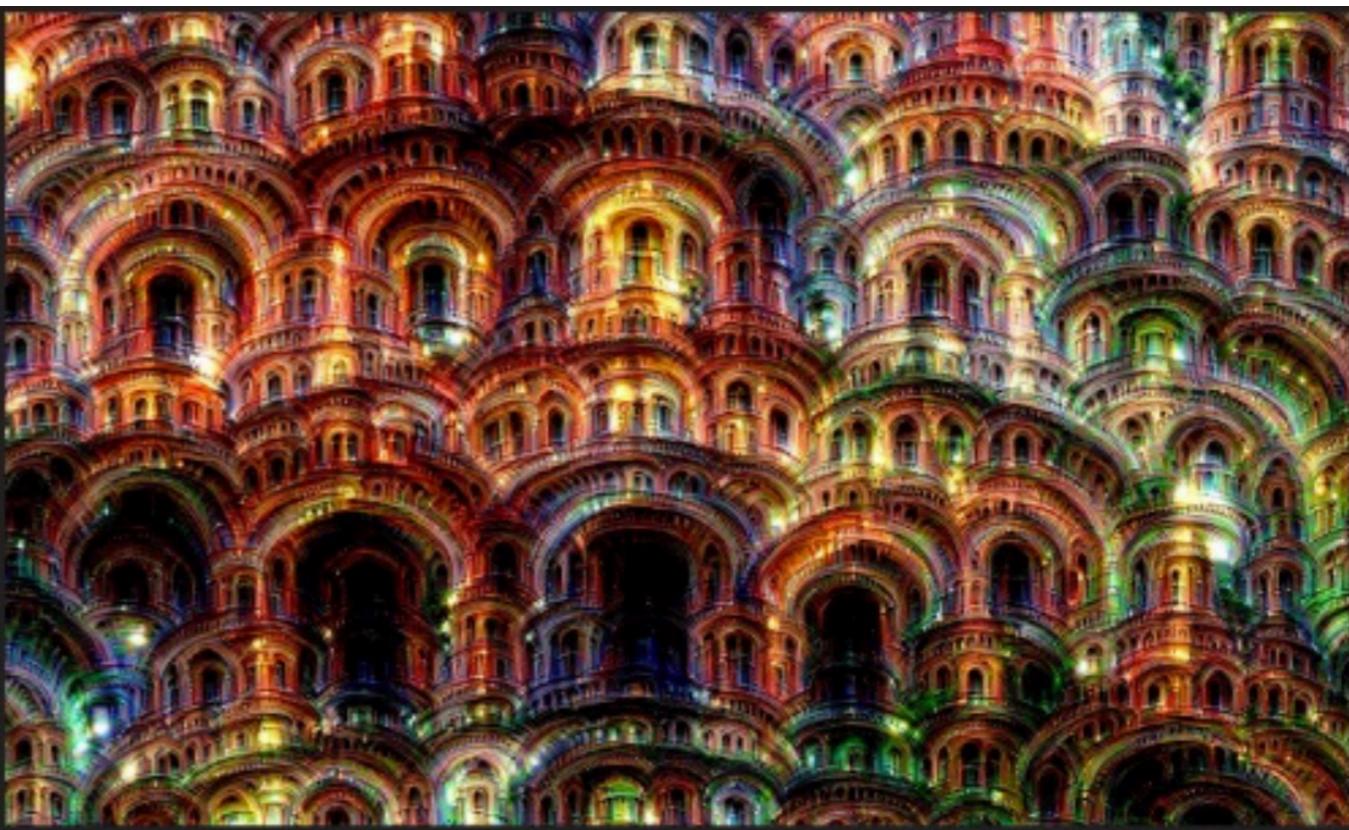
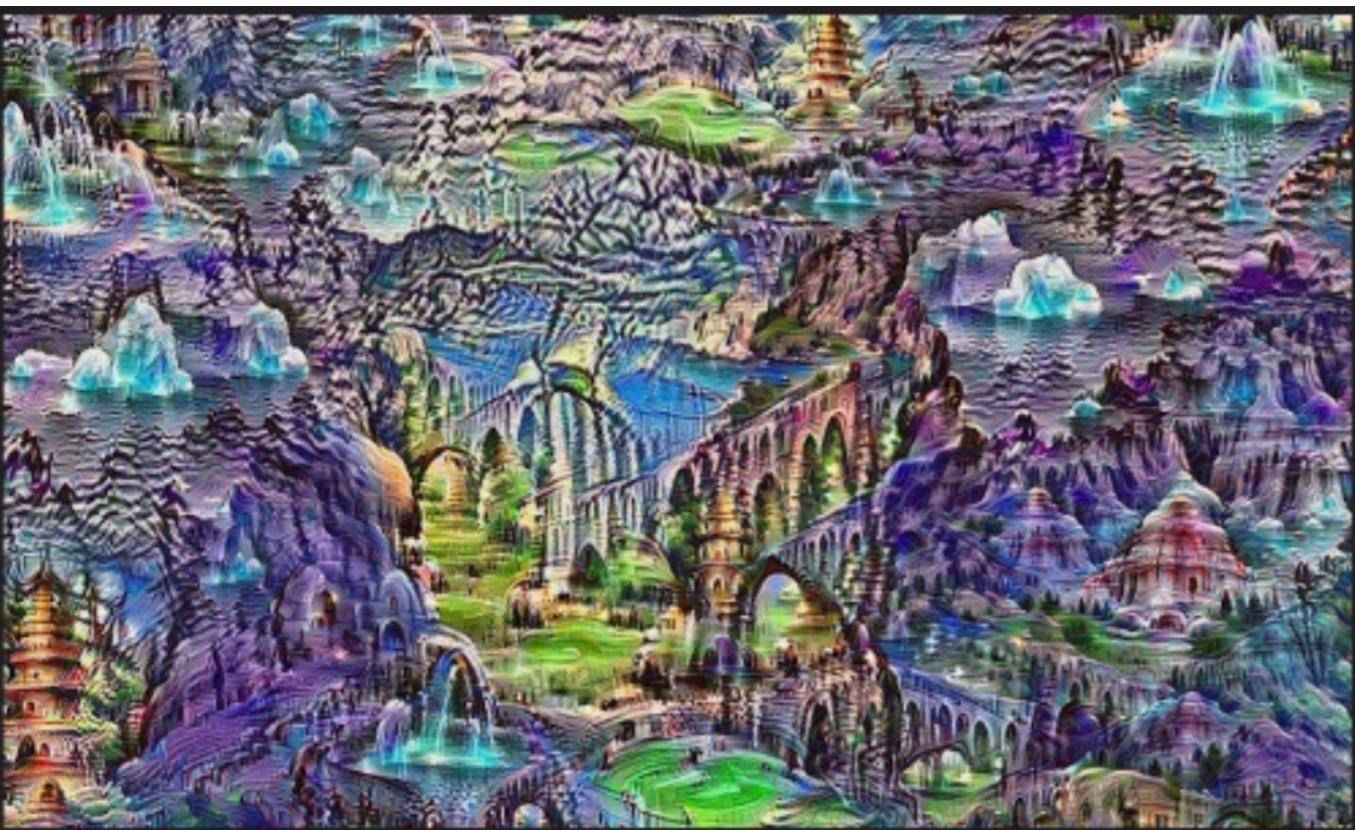
Differential

Insurance Portals Genetics links Misc OBGYN links

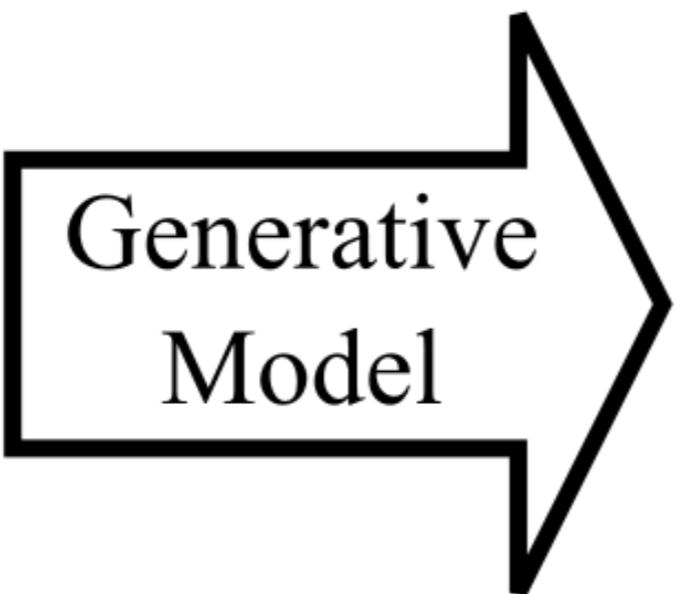
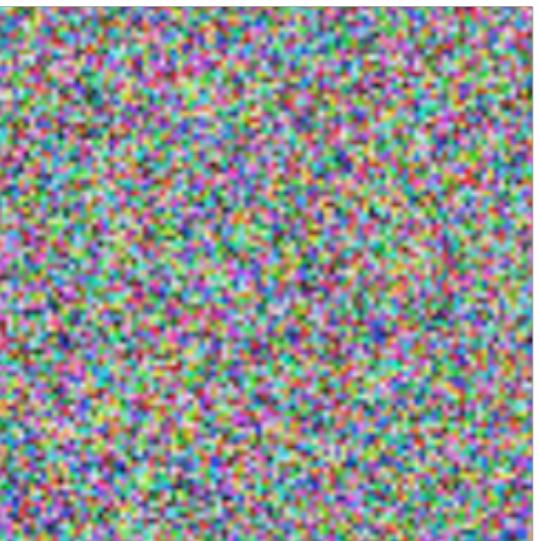
...

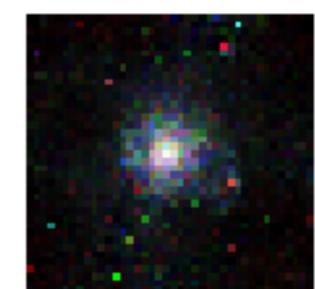
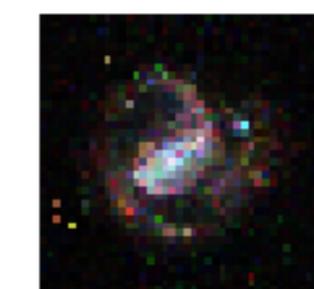
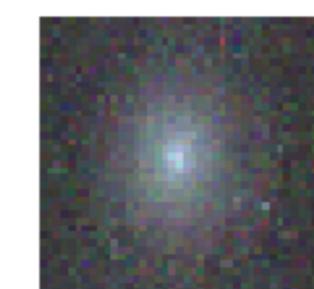
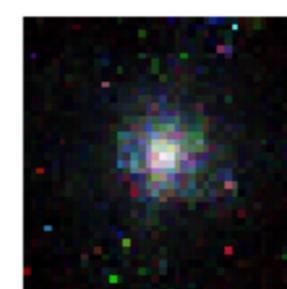
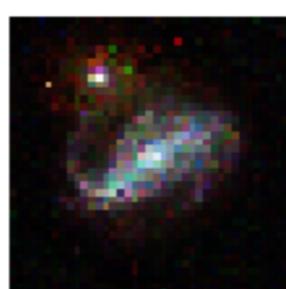
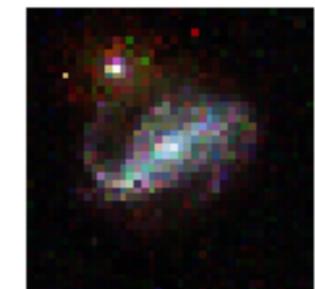
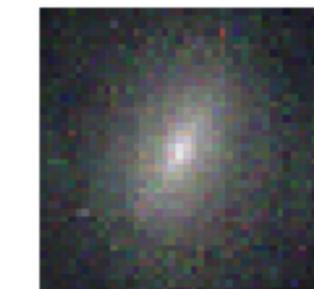
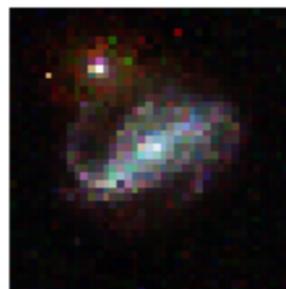
GAN (Generative Adversarial Network)





Noise $\sim N(0,1)$

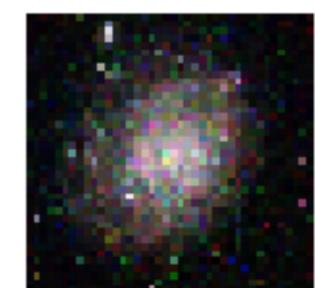
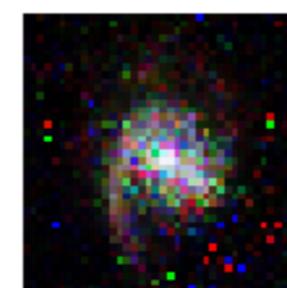
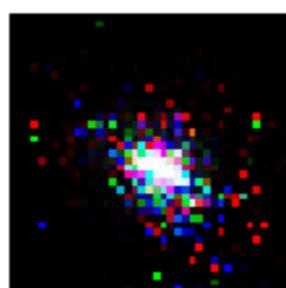




Elliptical

Barred Spiral

Spiral



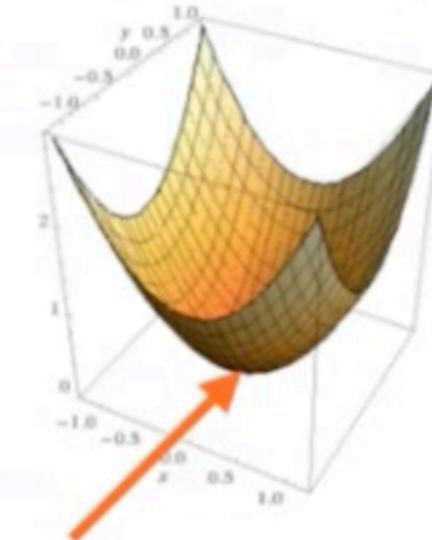
GAN usage

- Mock catalogs
- Denoising
- Compete gaming

Warning: Hard to train

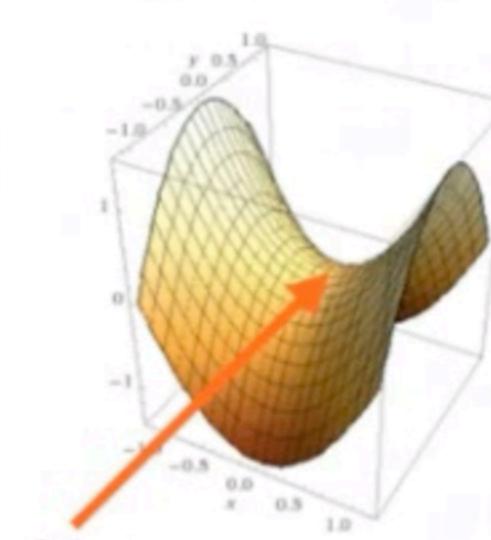
Adversarial Machine Learning

Traditional ML:
optimization



Minimum
One player,
one cost

Adversarial ML:
game theory



Equilibrium
More than one player,
more than one cost

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?