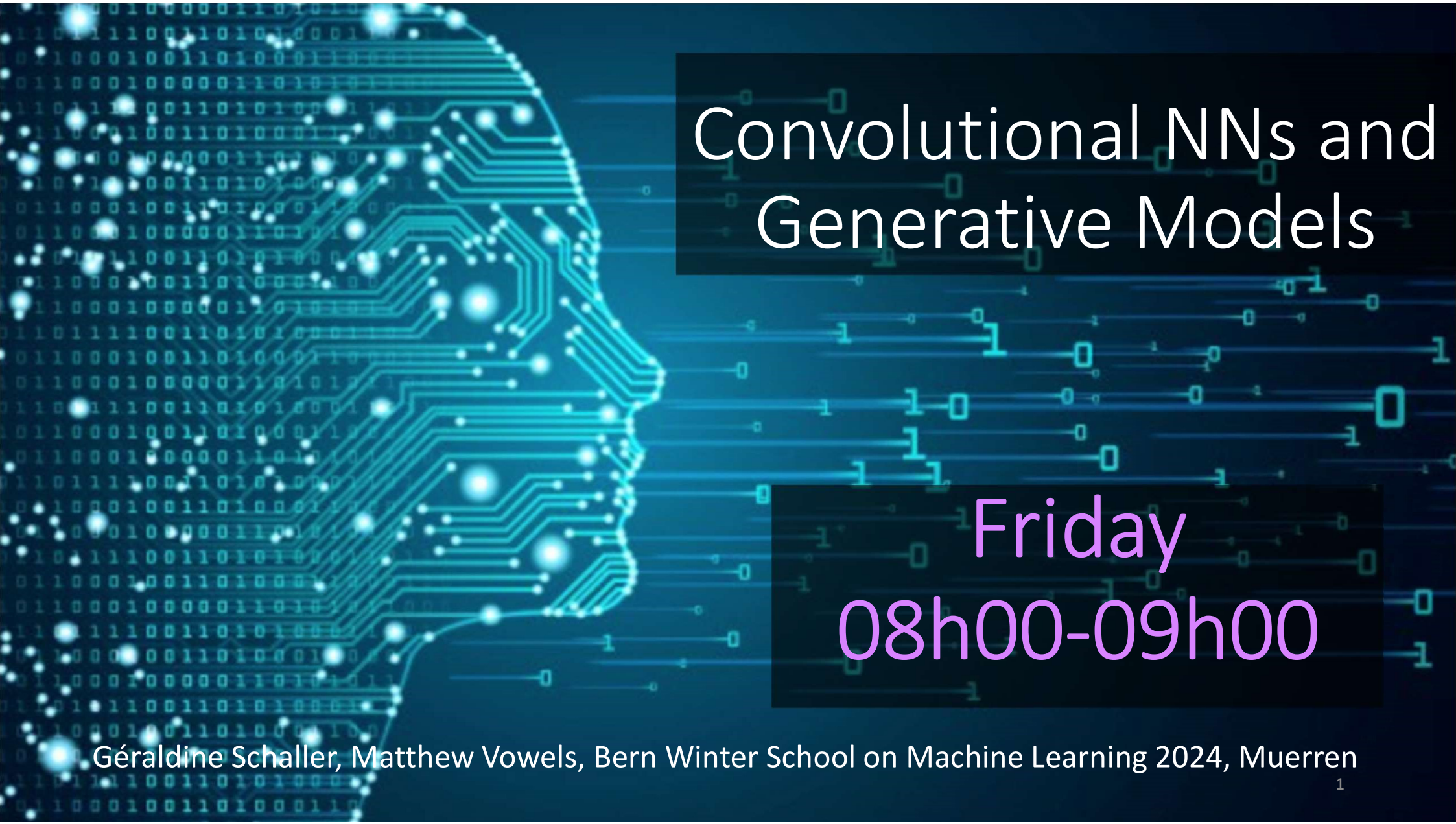


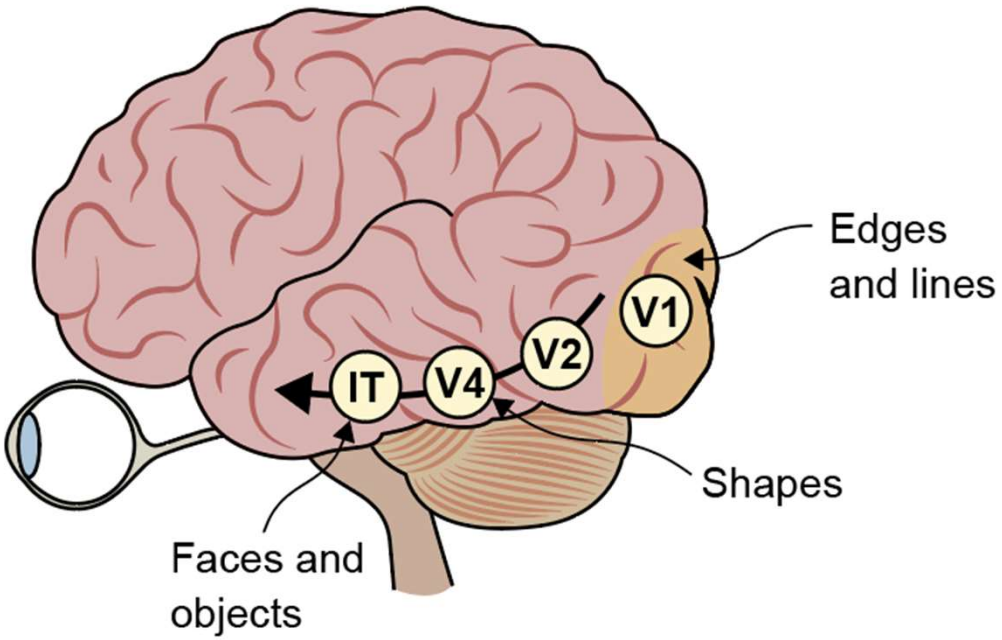
Convolutional NNs and Generative Models

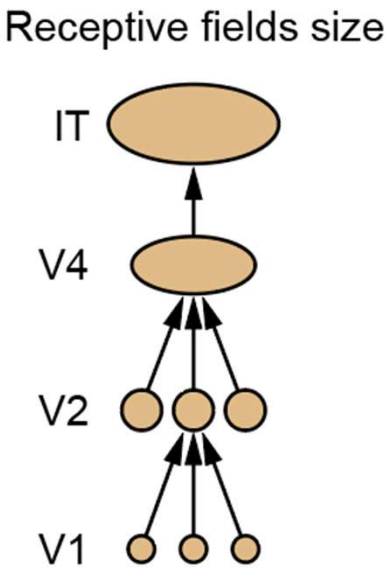
Friday
08h00-09h00

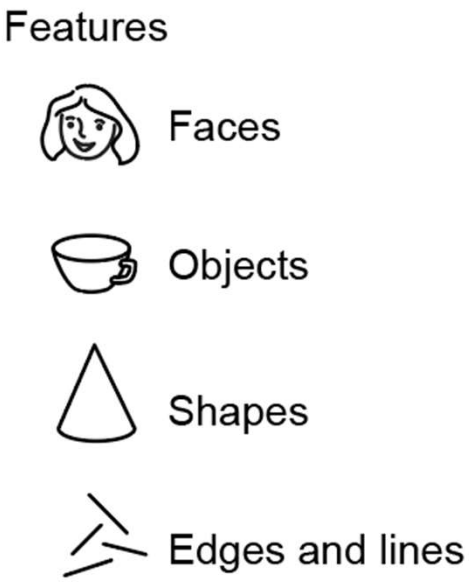
Géraldine Schaller, Matthew Vowels, Bern Winter School on Machine Learning 2024, Muerren


TO COMPLETE YOUR REGISTRATION, PLEASE TELL US WHETHER OR NOT THIS IMAGE CONTAINS A STOP SIGN:
STOP
NO
YES
ANSWER QUICKLY—OUR SELF-DRIVING CAR IS ALMOST AT THE INTERSECTION.
SO MUCH OF "AI" IS JUST FIGURING OUT WAYS TO OFFLOAD WORK ONTO RANDOM STRANGERS.

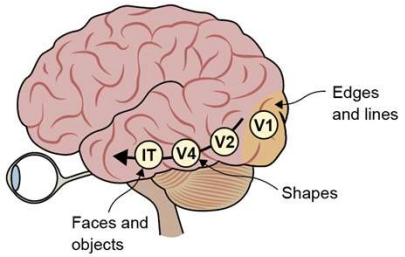
Human vision

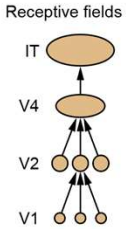


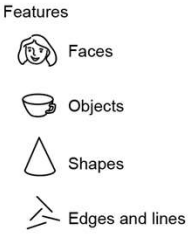


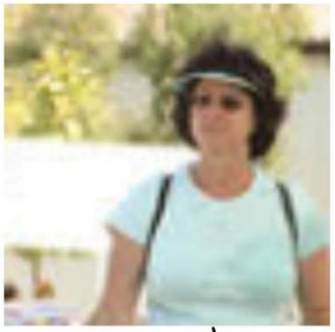


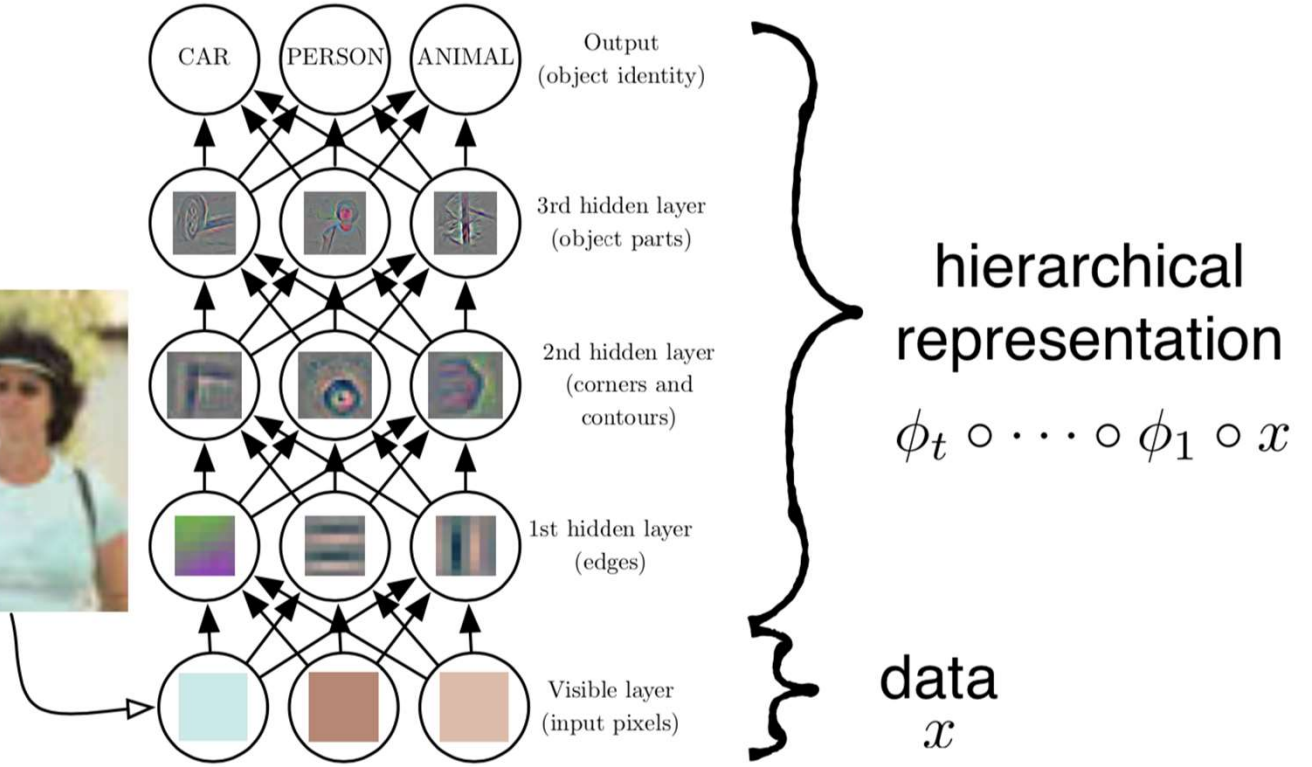
Computer vision inspired from Nature









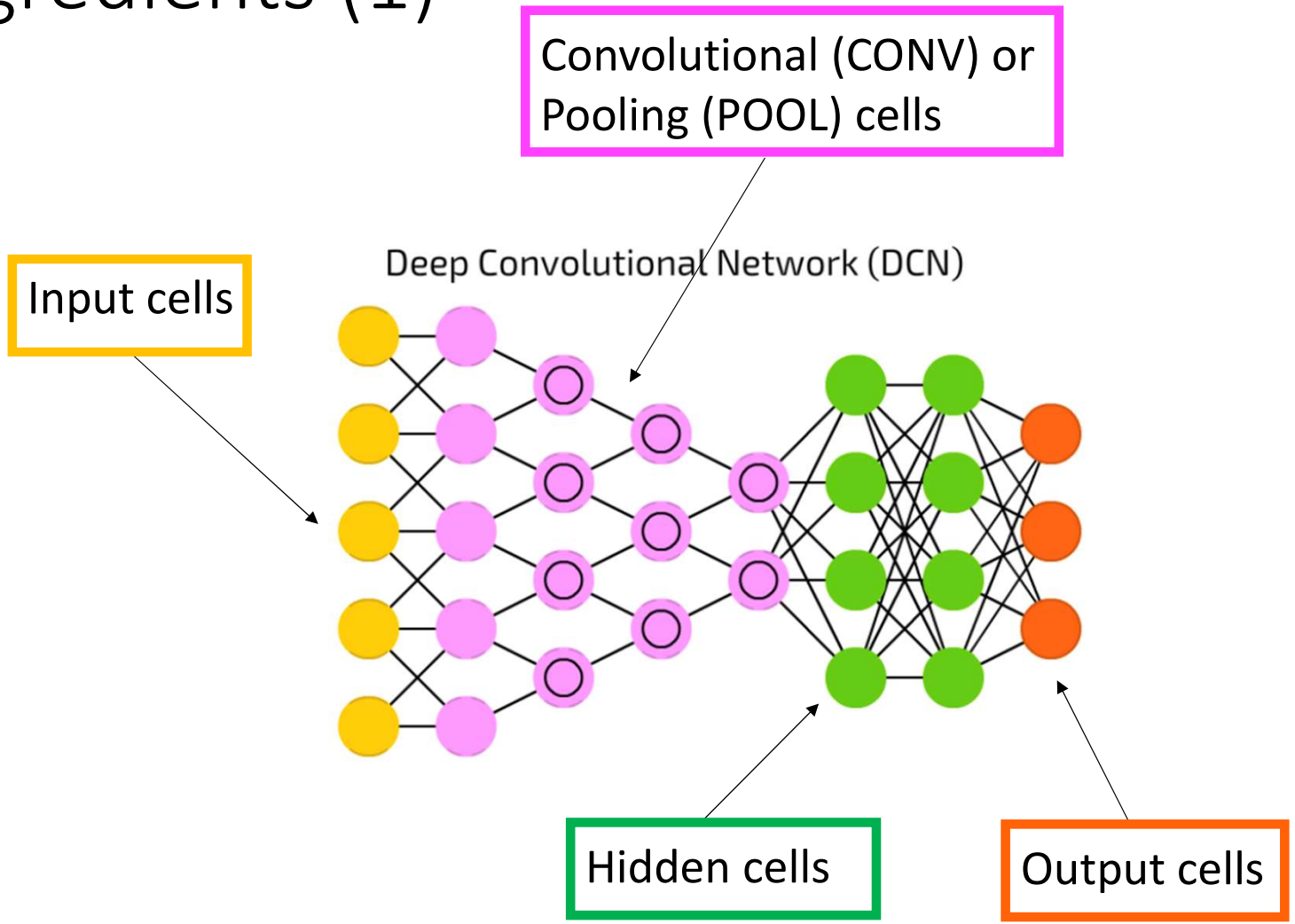


Input data

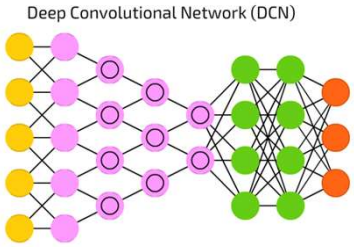
- Specialized Neural Network for data arranged on a grid
  - Images
  - DNA sequences
  - …

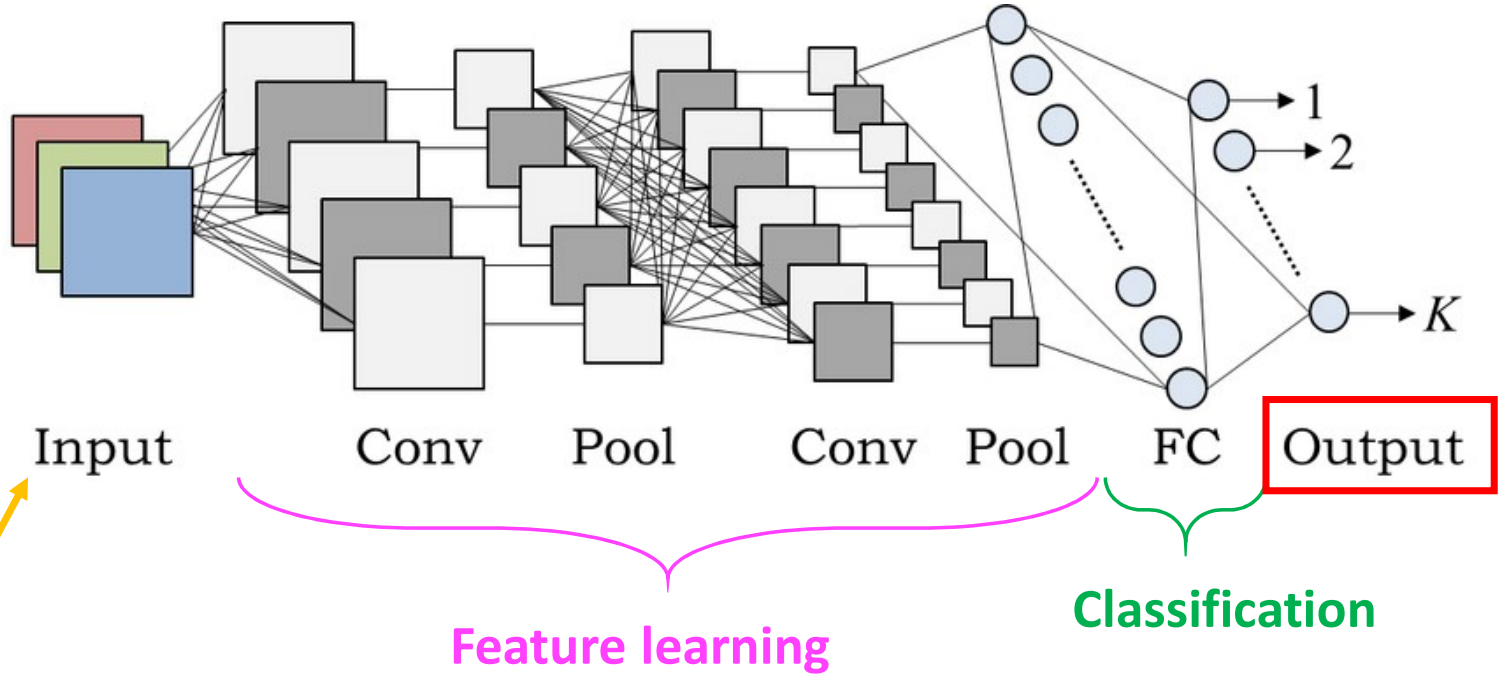
	A	C	G	T	W	S	M	K	R	Y	B	D	H	V	N	Z
A	1	0	0	0	1/2	0	1/2	0	1/2	0	0	1/3	1/3	1/3	1/4	0
C	0	1	0	0	0	1/2	1/2	0	0	1/2	1/3	0	1/3	1/3	1/4	0
G	0	0	1	0	0	1/2	0	1/2	1/2	0	1/3	1/3	0	1/3	1/4	0
T	0	0	0	1	1/2	0	0	1/2	0	1/2	1/3	1/3	1/3	0	1/4	0

CNN ingredients (1)



CNN ingredients (2)



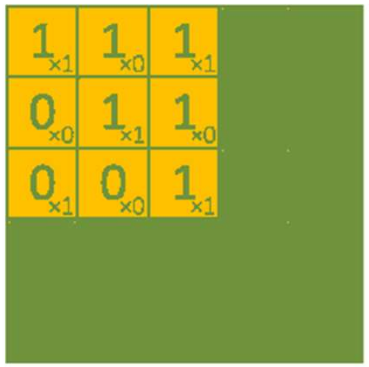


RGB image separated
by its 3 color planes

CONV layer : the Kernel

- Used to detect features (vertical/horizontal filter,…)

1	0	1
0	1	0
1	0	1



Image

how well the pattern in the kernel matches the content in that part of the image

Convolved Feature = Feature map

- Different kernels to create different feature maps → learn to see various patterns and details in images

1	0	-1
1	0	-1
1	0	-1

Vertical

1	1	1
0	0	0
-1	-1	-1

Horizontal

Vertical edges

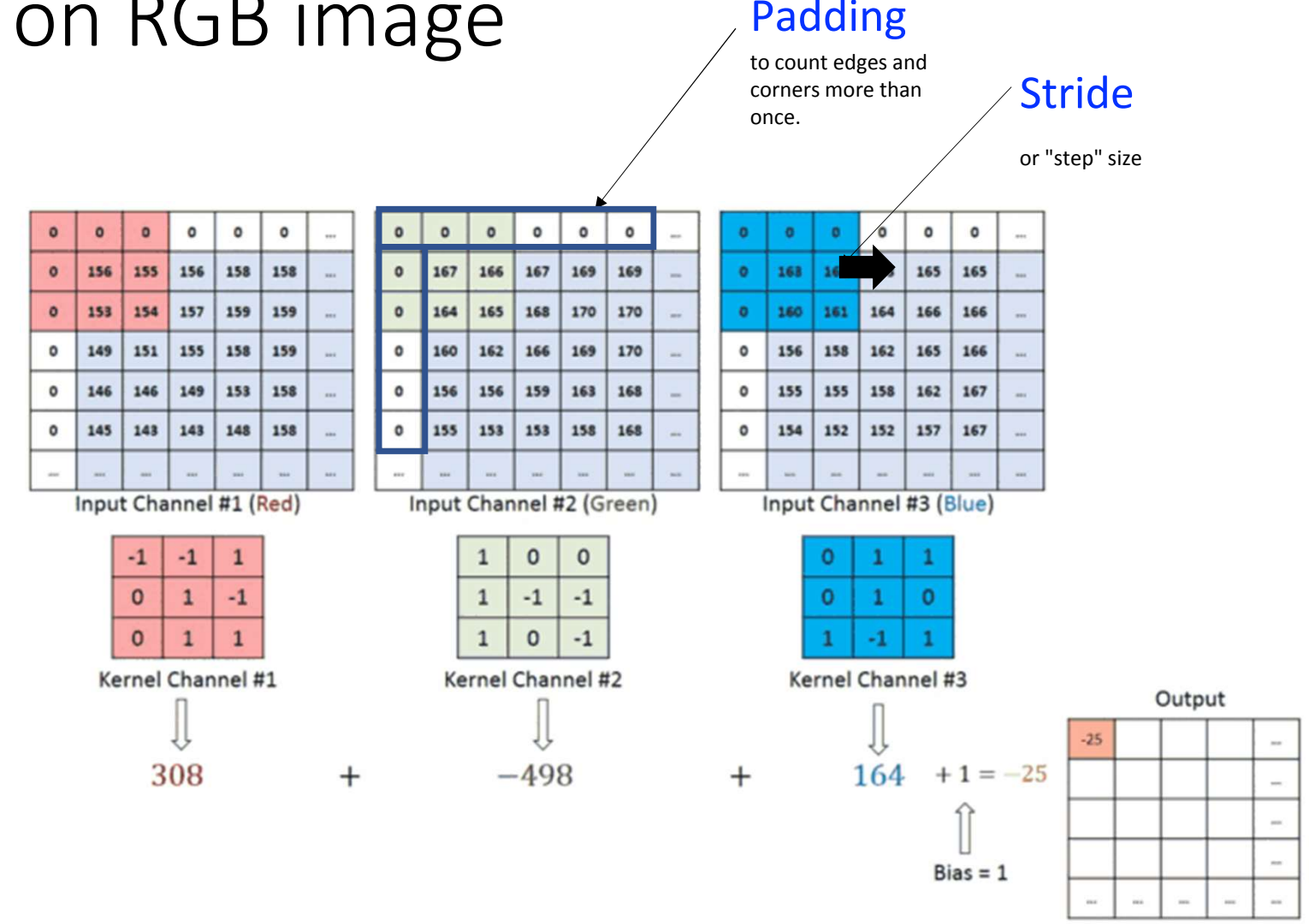
Horizontal edges

Convolution on RGB image

- In this example, use of a 3x3x3 filter

Width, height, #input channels

In a standard color image, the number of input channels is 3 (for red, green, and blue)



Kernels (filters)

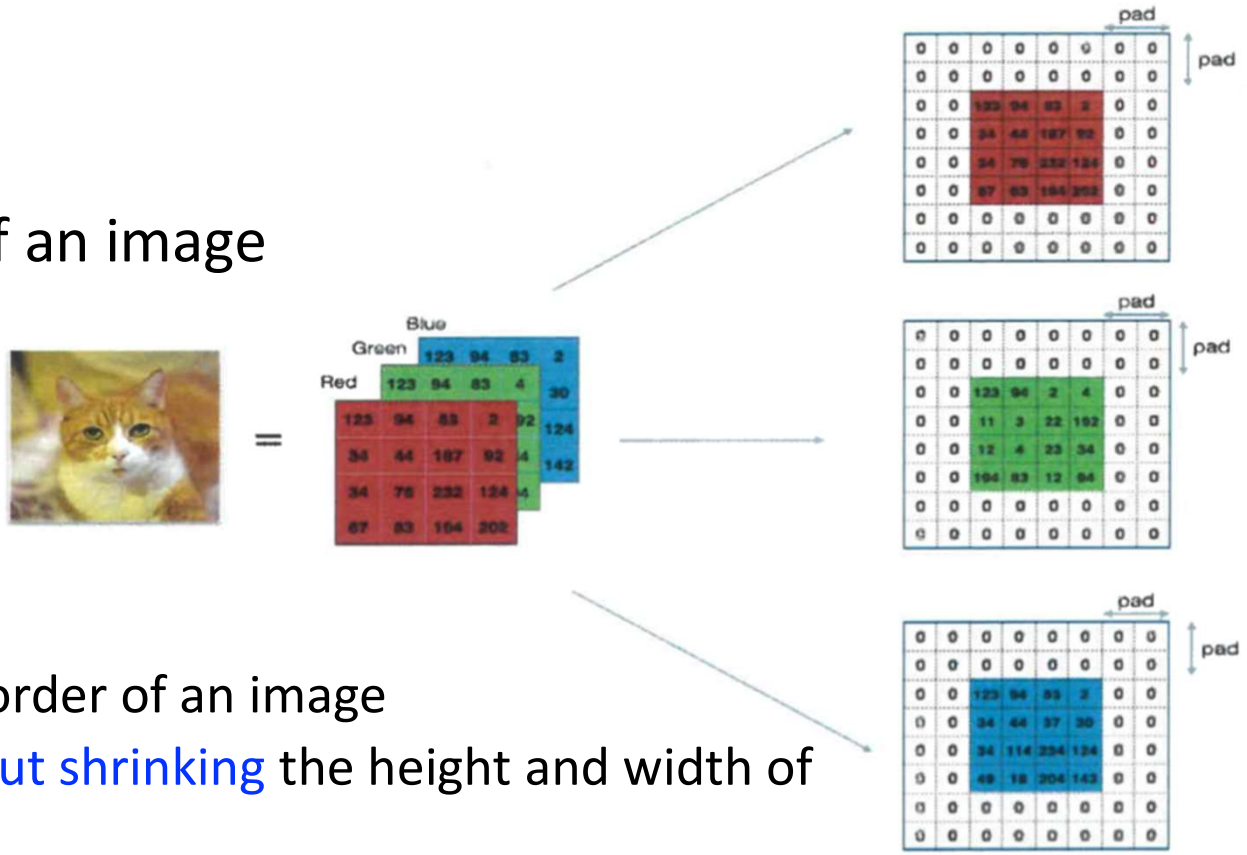
- Values inside a kernel are learned during the training process of the CNN
- Two aspects to consider for kernels :
  - Architecture :
    - how large they should be
    - How many input channels the kernel operates on
    - how many of them are applied to the input data

parameters = (Filter Width * Filter Height * Input Channels * Number of Filters / Stride) + Number of Filters

  - Initialization : how they should look at the start of training
    - Random initialization (Xavier/He initialization)
    - Transfer learning

Padding

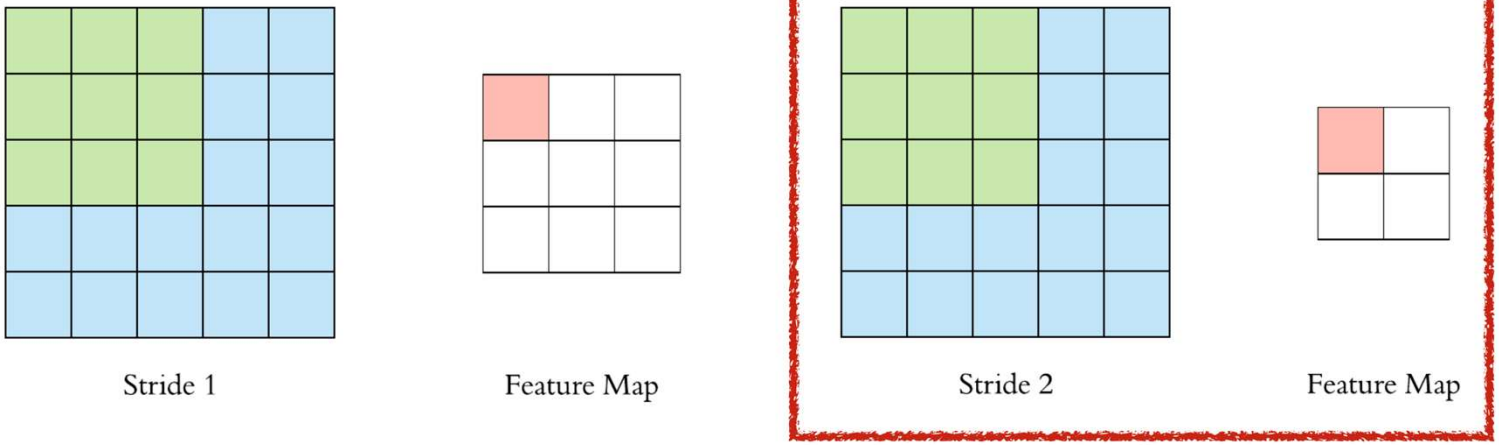
- Adds zeros around the border of an image



- Use cases :
  - Keeps more information at the border of an image
  - Allows to use a CONV layer without shrinking the height and width of the volumes
    - *important for deeper networks*

Stride

- Can be thought of as a "step size": i.e., how much you move the filter
- Use cases :
  - Connect a large input layer to a much smaller layer by spacing out the receptive fields (reduce dimensionality)



increasing stride from 1 to 2

- 300x300 RGB image, how many parameters does a hidden layer have (incl. bias) in the following cases:

a) Hidden layer with 100 neurons, each fully connected to the input (*no convolution*)

params = (number of input connections + 1) * number of neurons

ANSWER:
>>> params = (300*300*3+1)*100
>>> print(params)
27000100

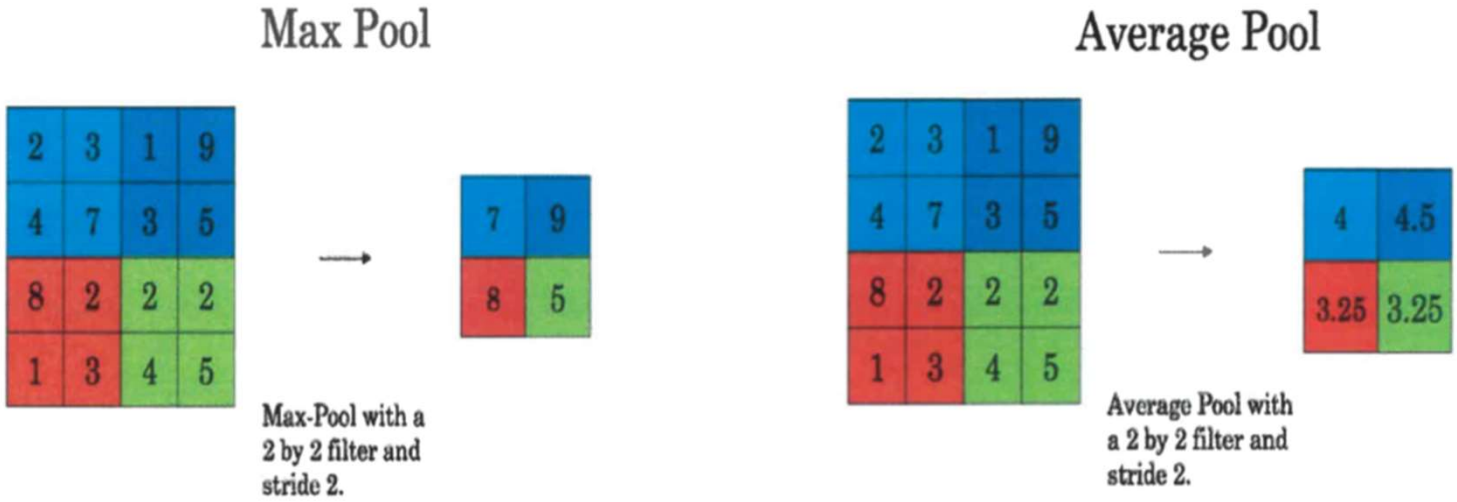
a) Hidden *convolution* layer with 100 filters that are each 5x5

params = (Filter Width * Filter Height * Number of Input Channels / Stride^2)
* Number of Filters + Number of Filters

ANSWER:
>>> paramsC = (5*5*3)*100+100
>>> print(paramsC)
7600

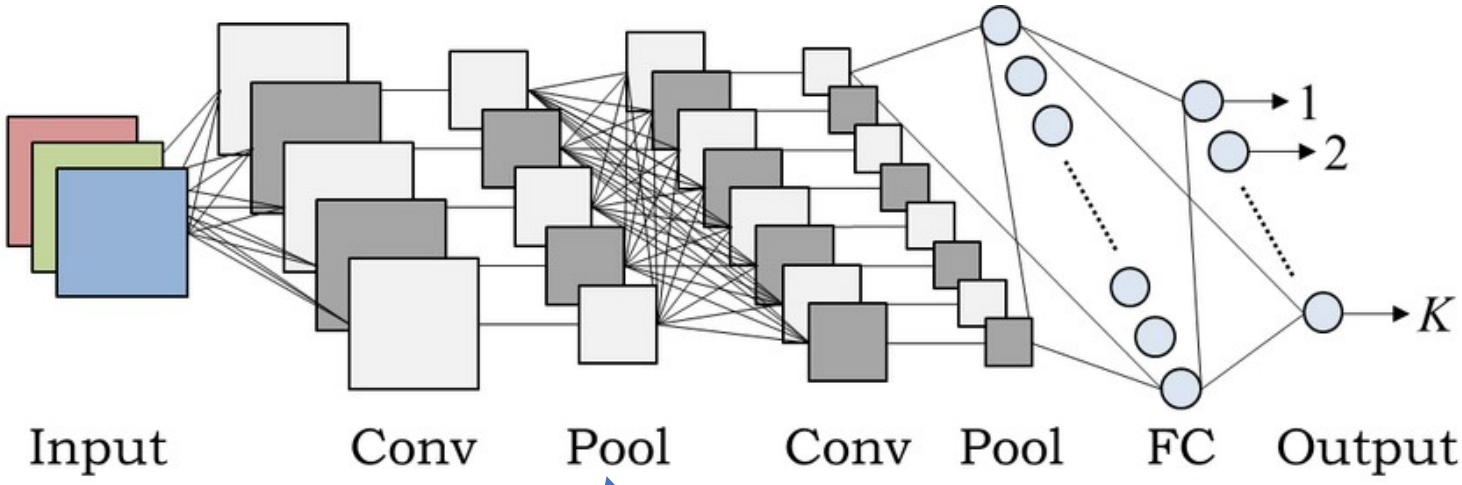
POOL layer

- Goal is to subsample the input image : the exact location of a feature is less important than its rough location relative to other features.
  - Reduce the computational load, memory usage, number of parameters (hence overfitting)



- Pooling neuron has *no weights*, it aggregates the inputs with an aggregation function (max, mean)

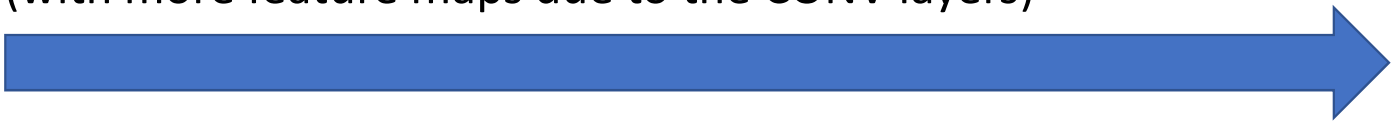
CNN Architecture : Summary



detects patterns in multiple sub-regions in the input field using receptive fields

progressively reduces the spatial size of the representation (fewer parameters, less overfitting)

Image gets smaller and smaller, but also deeper and deeper (with more feature maps due to the CONV layers)



One Look Is Worth A Thousand Words--

One look at our line of Republic, Firestone, Miller and United States tires can tell you more than a hundred personal letters or advertisements.

WE WILL PROVE THEIR VALUE BEFORE YOU INVEST ONE DOLLAR IN THEM.

Ever consider buying Supplies from a catalog?

What's the use! Call and see what you are buying. One look at our display of automobile and motorcycle accessories will convince you of the fact.

THAT WE HAVE EVERYTHING FOR THE AUTO

Piqua Auto Supply House

133 N. Main St.—Piqua, O.

https://www.youtube.com/watch?v=1zvohULpe_0

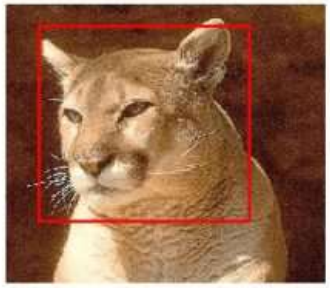
Object Detection

Introduction

Classification

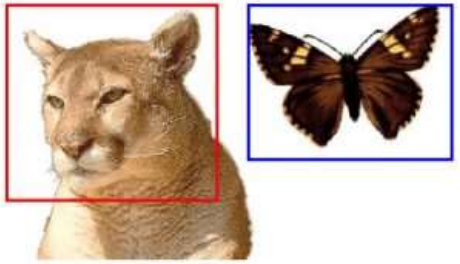
Cougar

Classification + Localization



Cougar

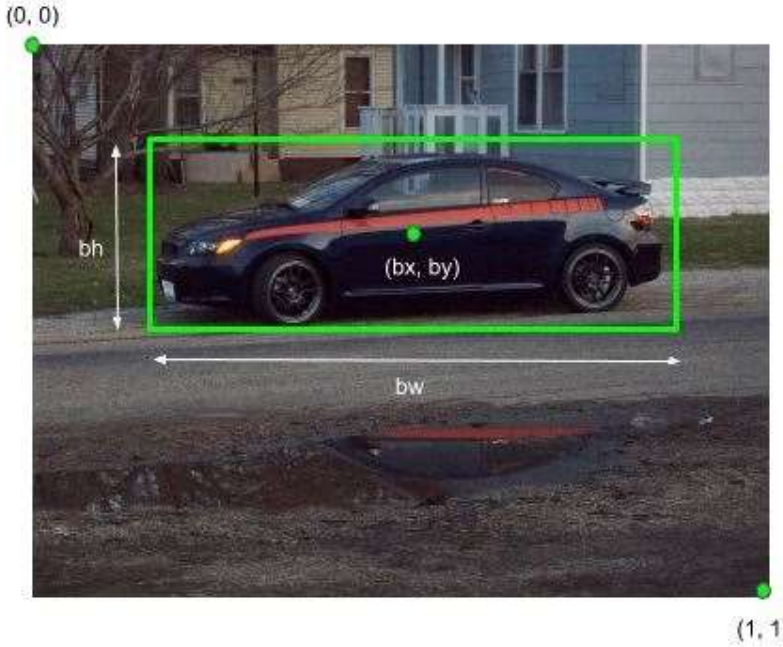
Object Detection



Cougar, Butterfly

Multiple Objects

Object localisation



Target variable :

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$

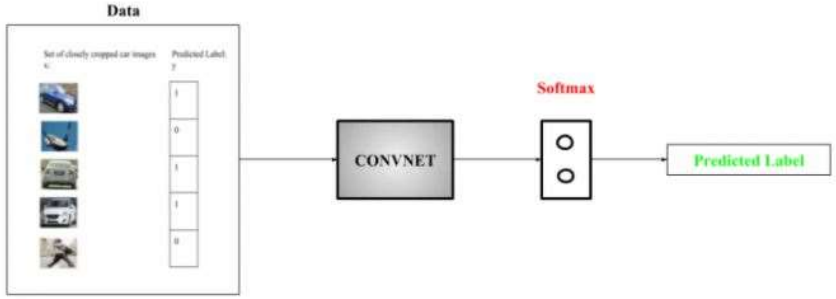
pc = Probability/confidence of an object (i.e the four classes) being present in the bounding box

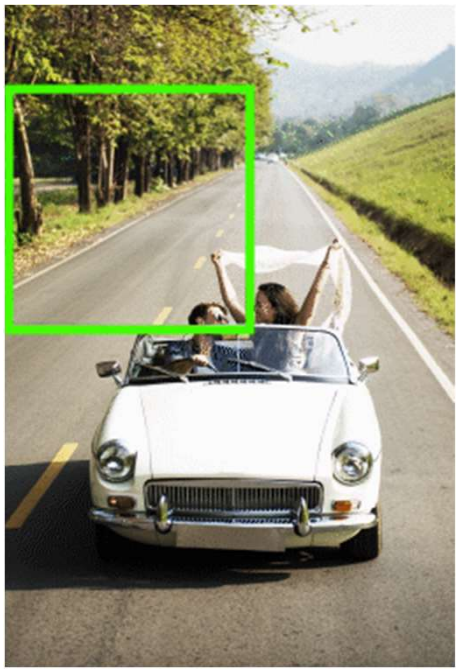
ci= Probability of the ith class the object belongs to

- Bounding box parameters :
  - bx, by : coordinates of the center of the bounding box
  - bw : width of the bounding box w.r.t the image width
  - bh : height of the bounding box w.r.t the image height

Object detection principle

1) A classifier that can classify closely cropped images of an object





2) Use a sliding window mechanism and crop a part of the image in each slide

3) Each cropped image is then passed to a ConvNet model, which in turn predicts the probability that the cropped image is a car

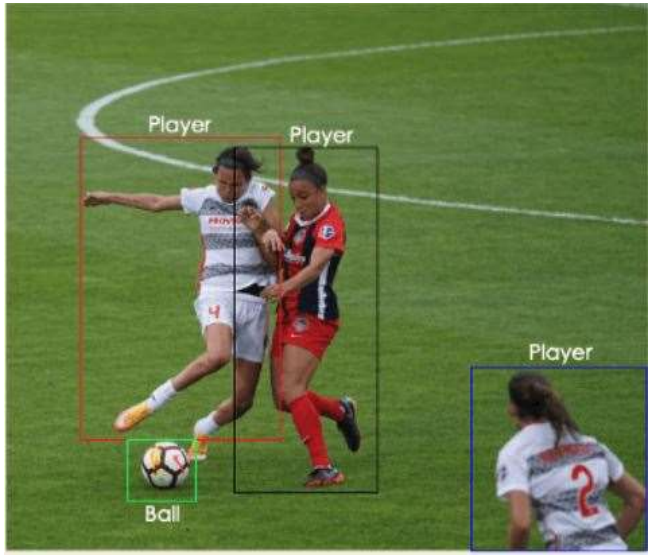
Self-driving cars

Object Detection Example

YOLO algorithm

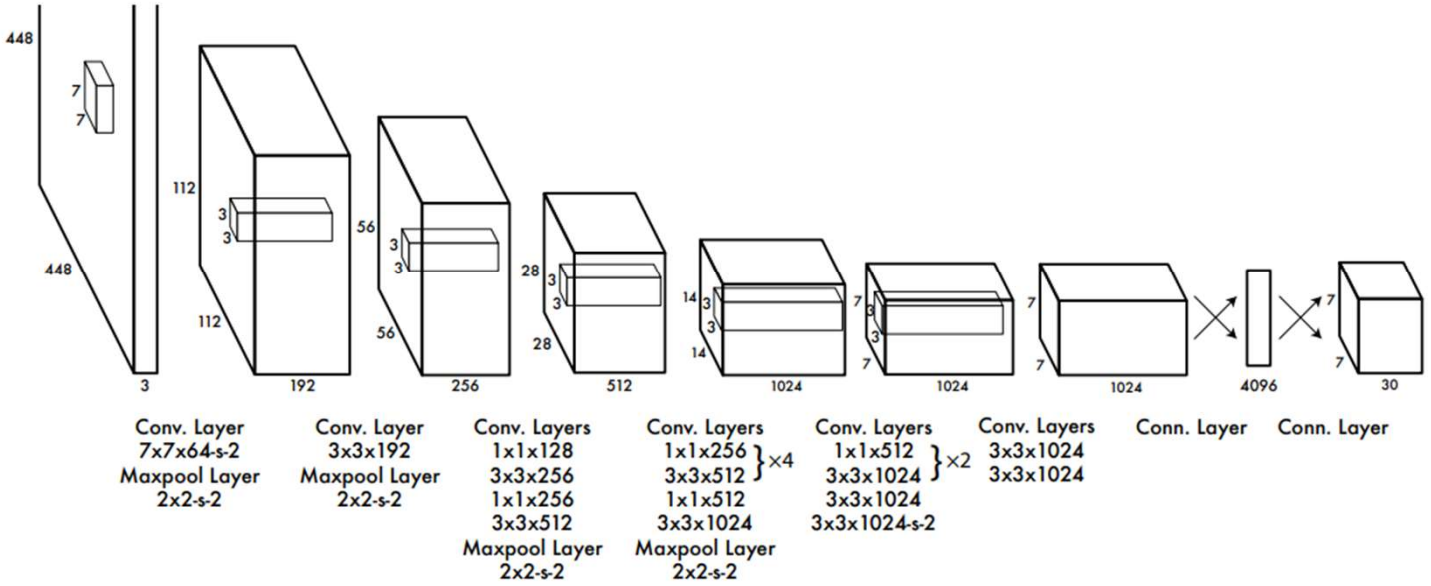
- You Only Look Once (YOLO) : real-time object detection algorithm

- object detection problem framed as a regression problem instead of a classification task
  - spatially separate bounding boxes and associate probabilities to each of the detected images using a CNN



YOLO Architecture

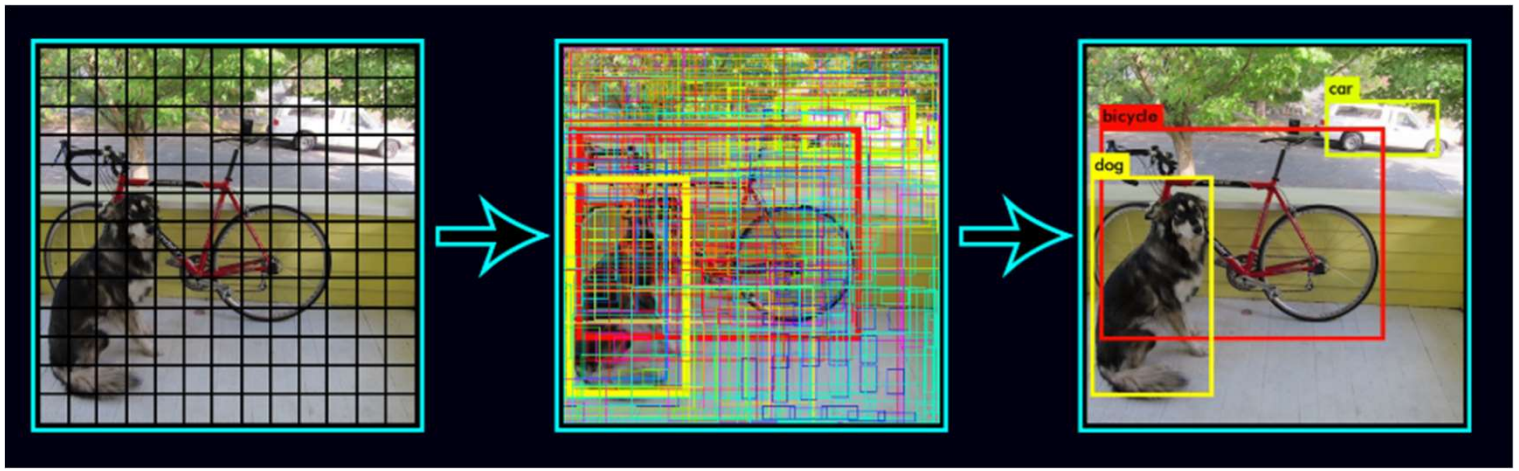
- YOLO architecture is similar to GoogleNet
  - 24 convolutional layers
  - 4 max-pooling layers
  - 2 fully connected layers



- Pretrained YOLOv2 model, trained on the COCO image dataset with classes similar to the Berkeley Driving Dataset
  - load pretrained weights
  - freeze all the weights except for the last layer during training

How does it work

- The algorithm works based on the four ingredients :
  - Residual blocks : divide the image into NxN grid cells of equal shape
  - Bounding box regression
  - Intersection Over Unions (IoU)
  - Non-Max Suppression : keep only the boxes with the highest probability score of detection.

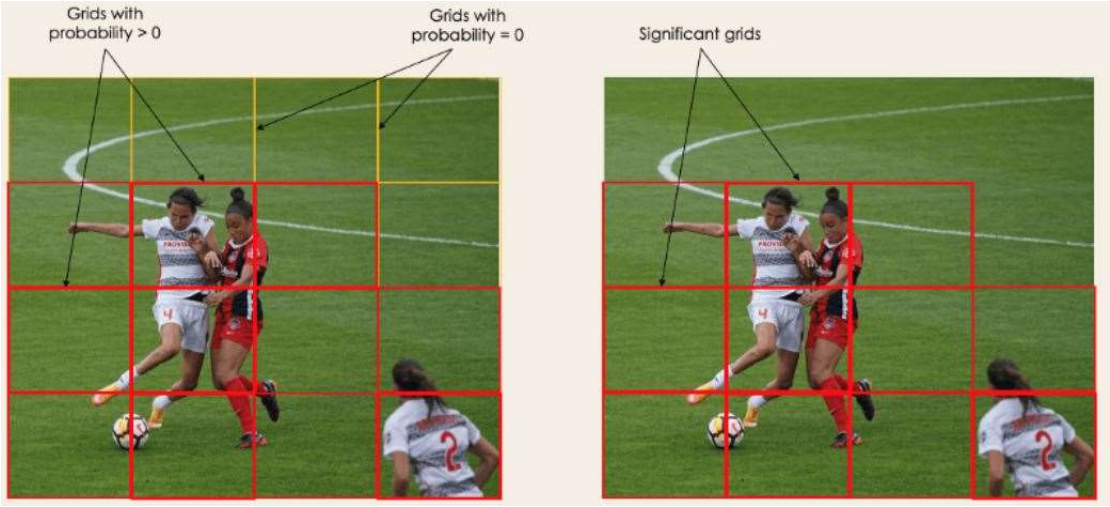


Bounding box regression

- YOLO determines the attributes of these bounding boxes using a single regression module
  - Y is the final vector representation for each bounding box :

    Y = [pc, bx, by, bh, bw, c1, c2]

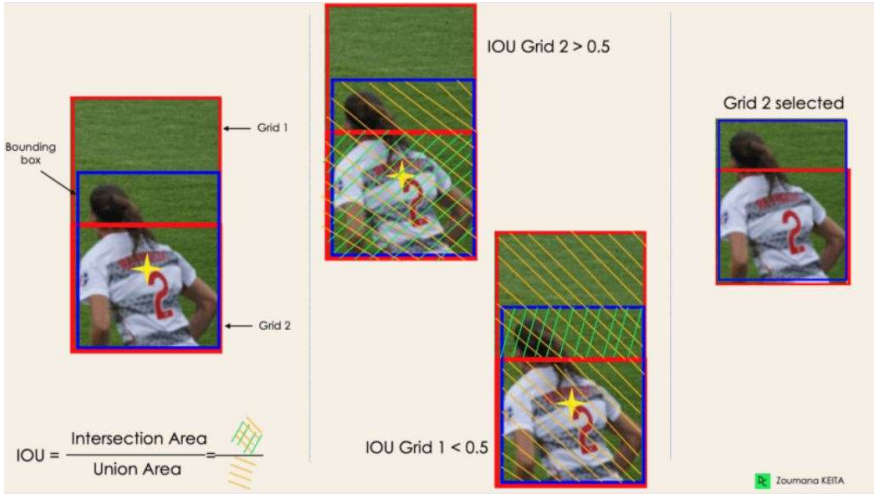
- pc corresponds to the probability score of the grid containing an object



Intersection Over Union (IoU)

- a single object can have multiple grid box candidates for prediction

- IoU (between 0 and 1) helps to discard grid boxes to keep only the relevant ones
  - IoU computed for each grid cell = Intersection area divided by the Union Area.
  - IoU selection threshold is defined parameter (0.5)
  - ignore the prediction of the grid cells having an IOU ≤ threshold

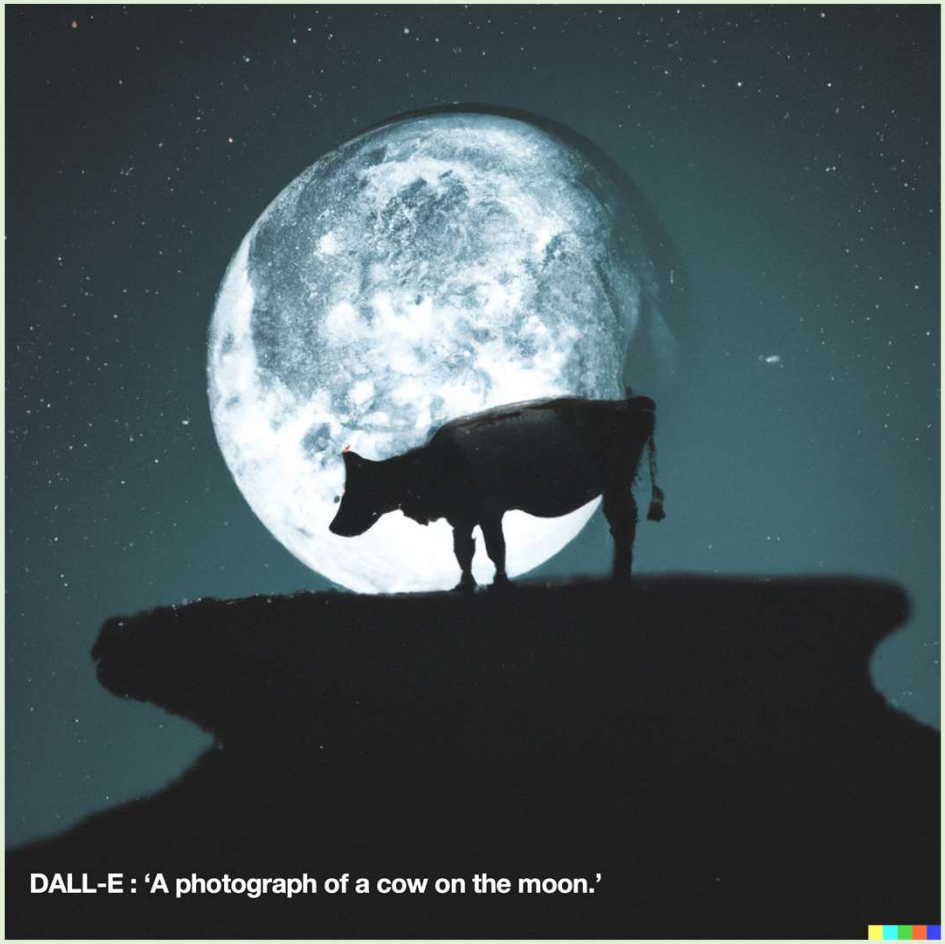


Result

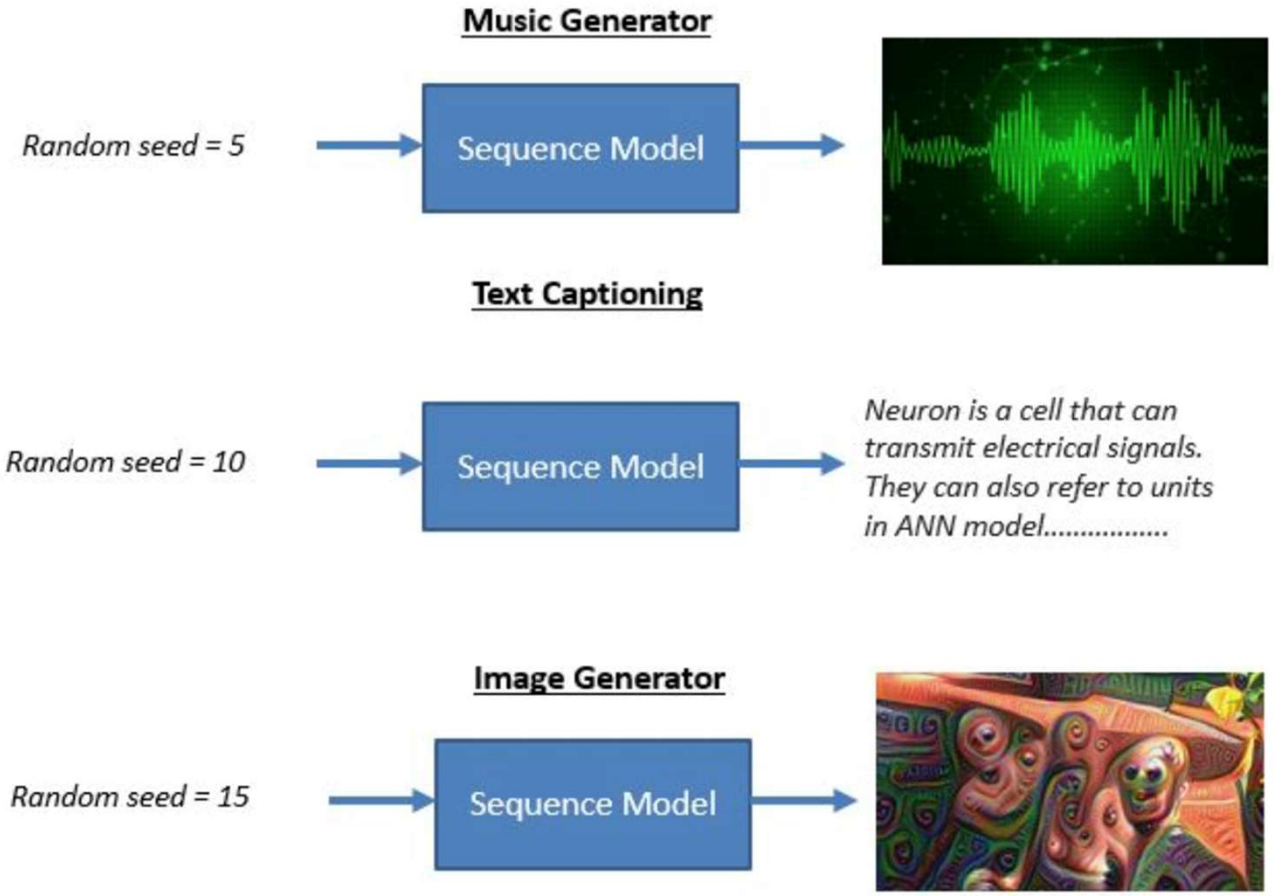
https://www.youtube.com/watch?v=1_SiUOYUoOI

Generative Models

Holy grail of Deep Learning these days



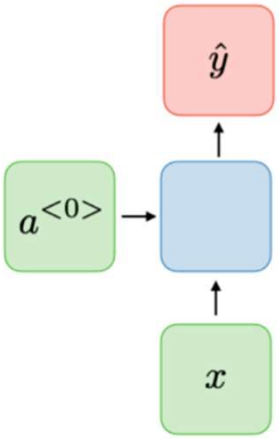
Generative Models Examples



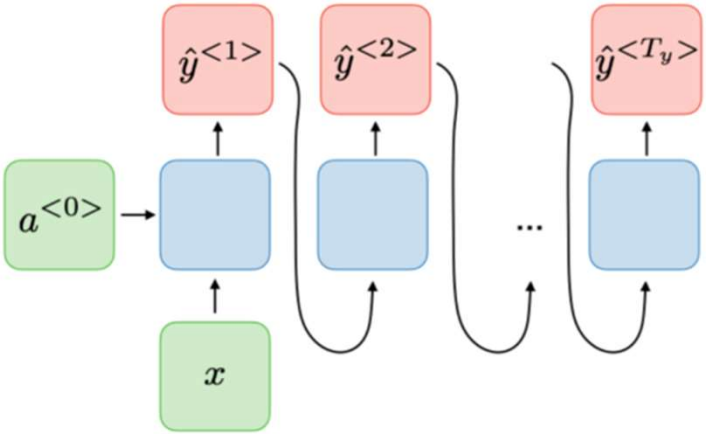
DeepBach

GauGAN

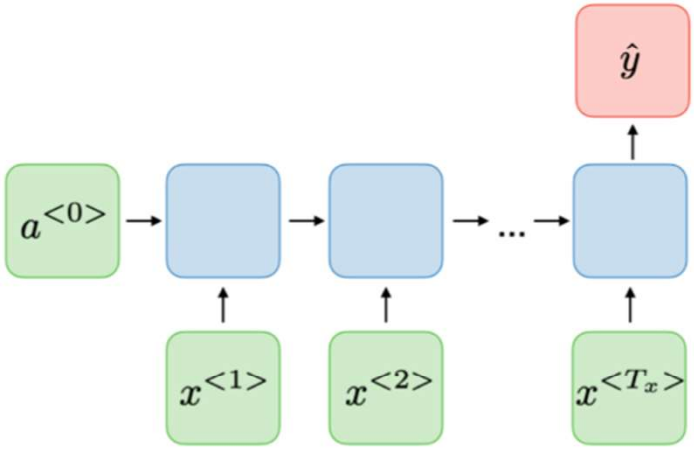
Applications of RNN (Recurrent Neural Network)

Type of RNN	Illustration	Example
One-to-one $T_x = T_y = 1$		Traditional neural network

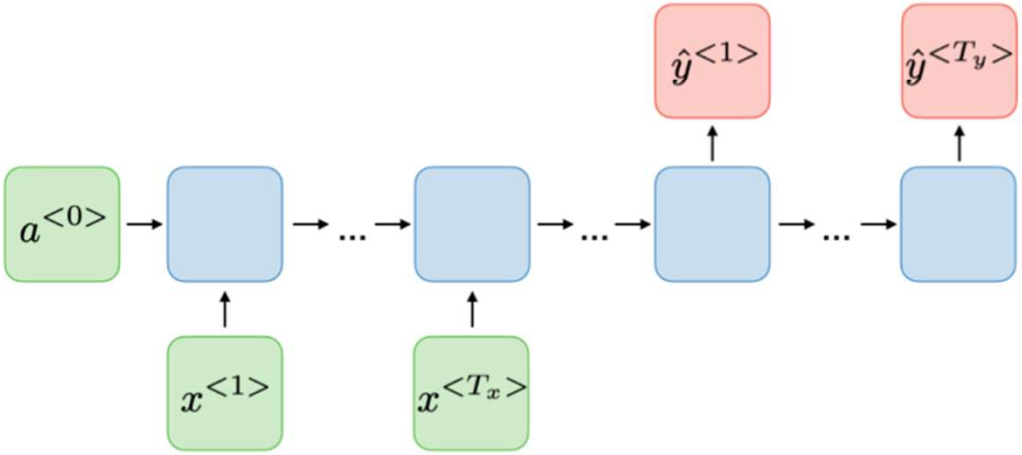
Applications of RNN

Type of RNN	Illustration	Example
One-to-many $T_x = 1, T_y > 1$		Music generation

Applications of RNN

Type of RNN	Illustration	Example
Many-to-one $T_x > 1, T_y = 1$		Sentiment classification

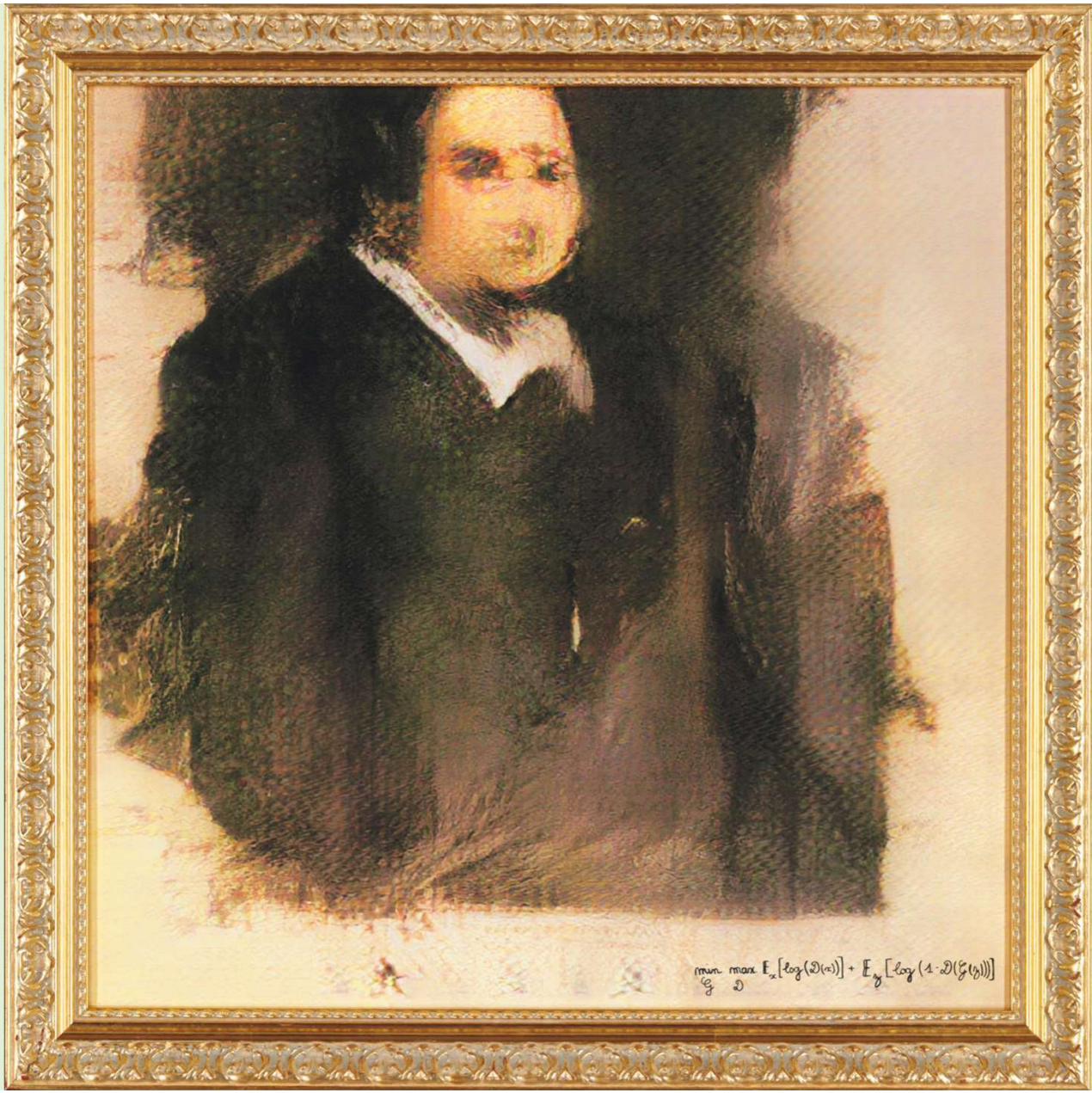
Applications of RNN

Type of RNN	Illustration	Example
Many-to-many $T_x \neq T_y$		Machine translation

Generative Adversarial Network

Invented by Ian Goodfellow

How much are you ready to pay for it ?



Christie's New York 2018

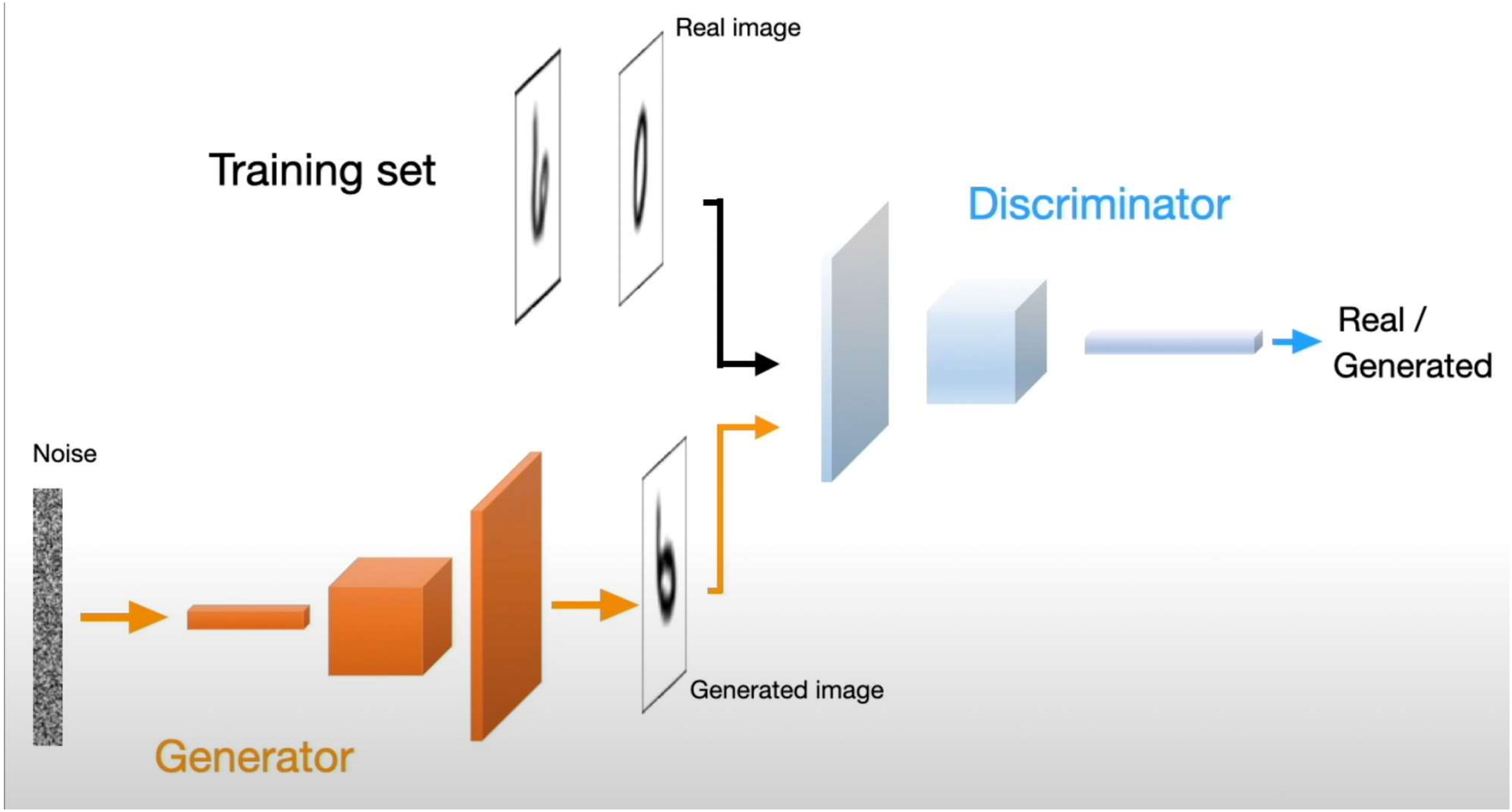
Principle

GENERATOR
"The Artist"
A neural network trying to create pictures of cats that look real.

DISCRIMINATOR
"The Art Critic"
A neural network examining cat pictures to determine if they're real or fake.

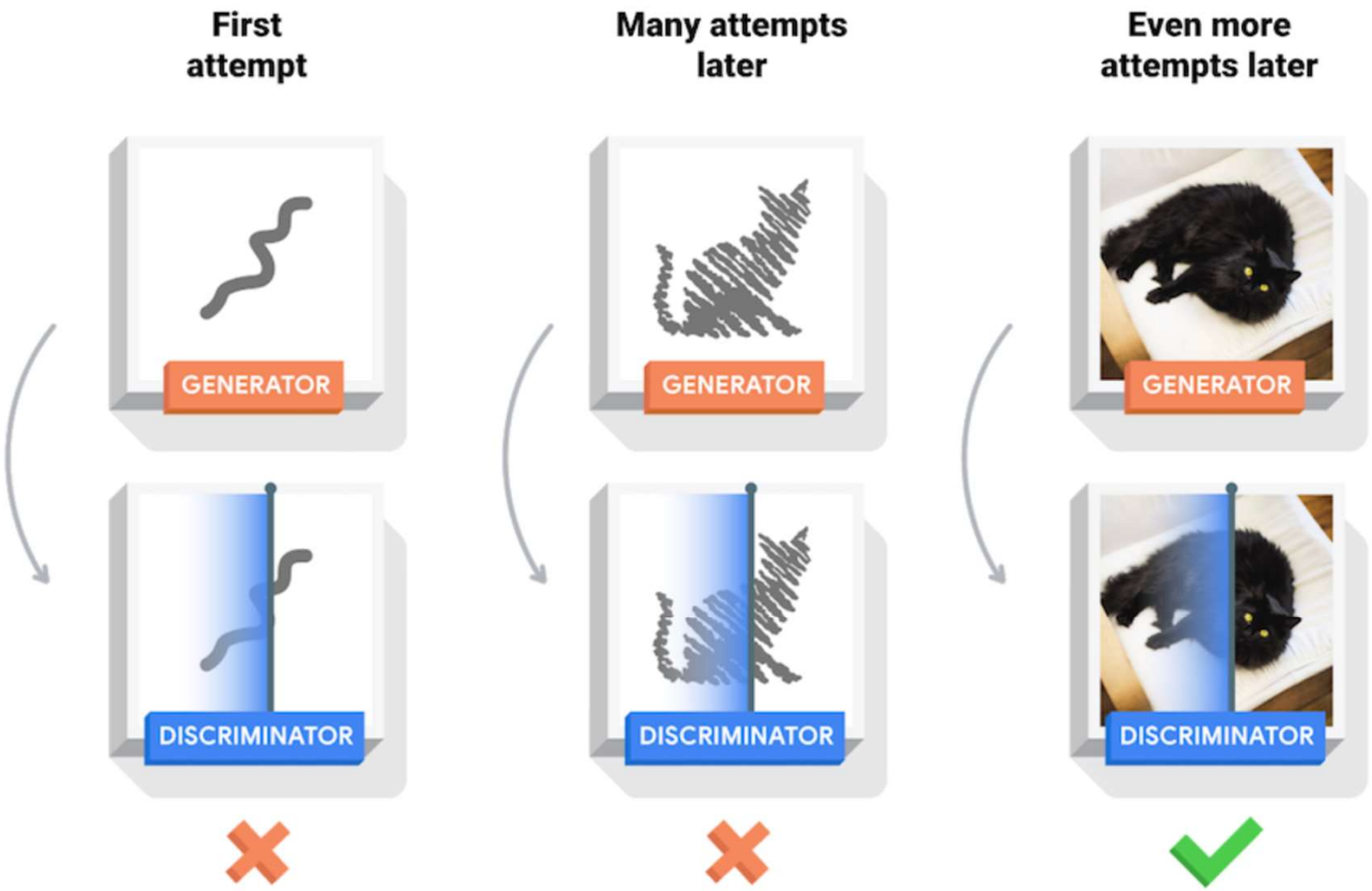
Thousands of real-world images labeled "CAT"

Principle



https://youtu.be/-Zi5SReze6U

Principle


First attempt
Many attempts later
Even more attempts later
GENERATOR
GENERATOR
GENERATOR
DISCRIMINATOR
DISCRIMINATOR
DISCRIMINATOR

GAN use case : generate images

Figure 3. Example results by our proposed StackGAN, GAWWN [20], and GAN-INT-CLS [22] conditioned on text descriptions from CUB test set. GAWWN and GAN-INT-CLS generate 16 images for each text description, respectively. We select the best one for each of them to compare with our StackGAN.

State Of The Art in GANs

(Karras et al, 2018)

(Brock et al, 2018)

Tutorial / Practical