# Convolutional NNs and Generative Models

Friday

08h00-09h00

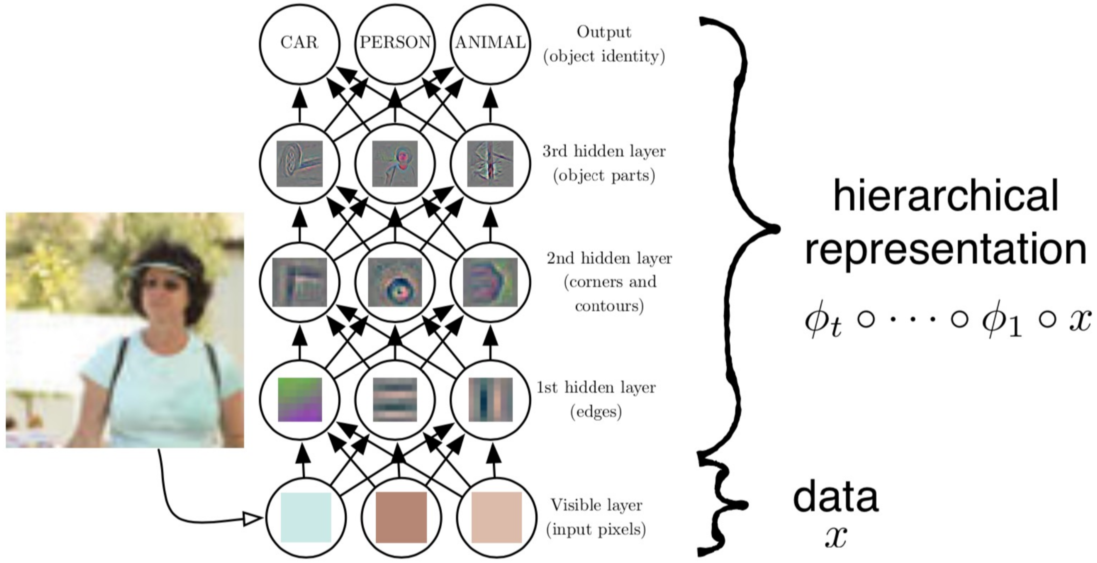
Géraldine Conti, Matthew Vowels, Bern Winter School on Machine Learning 2023, Muerren

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.

# Convolutional Neural Networks

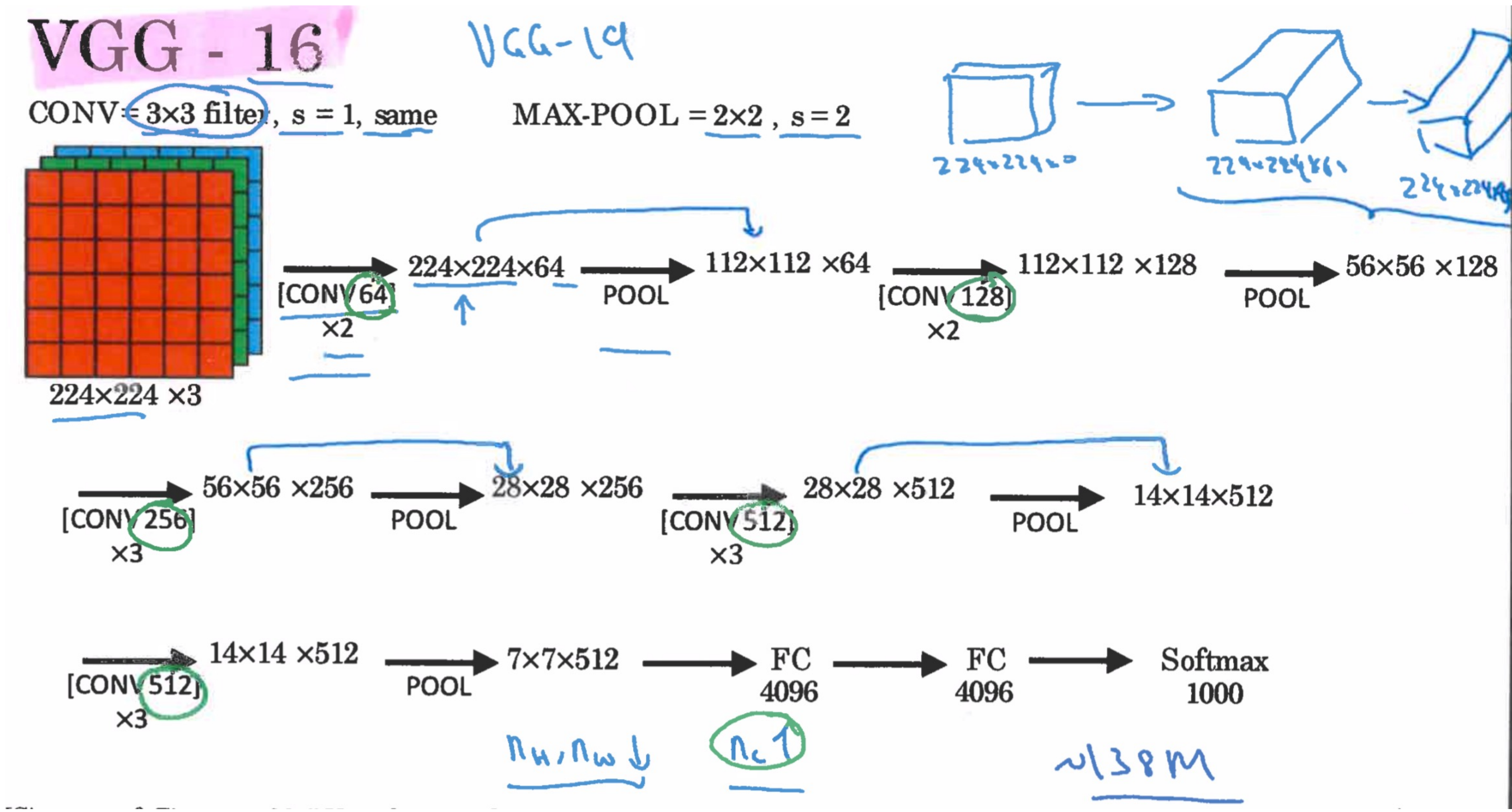
- Specialized Neural Network for data arranged on a grid
  - Images
  - DNA sequences
  - …

| | 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 |



# Types of layer in CNN

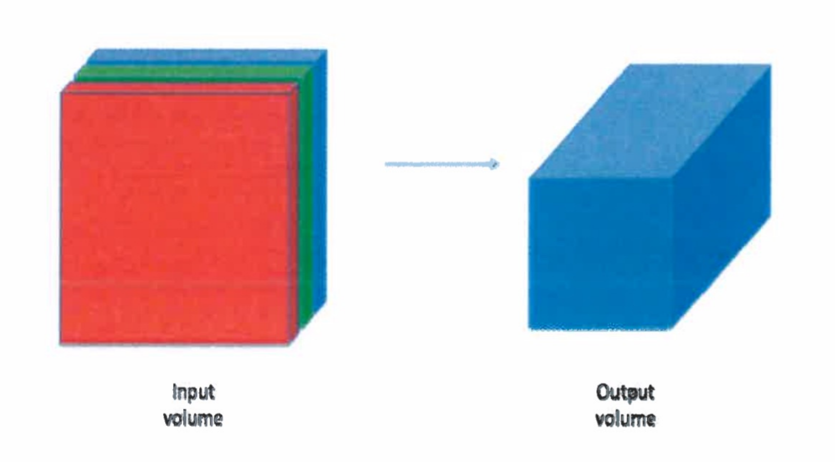
- Convolution (CONV)
- Pooling (POOL)
- Fully connected (FC)
- *Usually multiple CONV layers followed by a POOL layer, and FC layers in the last few layers*



# Convolution Layer (CONV)

- **Convolution** transforms an input volume into an **output volume** of different size, also called feature map

- **Filter kernels** are used to detect features (for example, edge detection in 1st hidden layer)

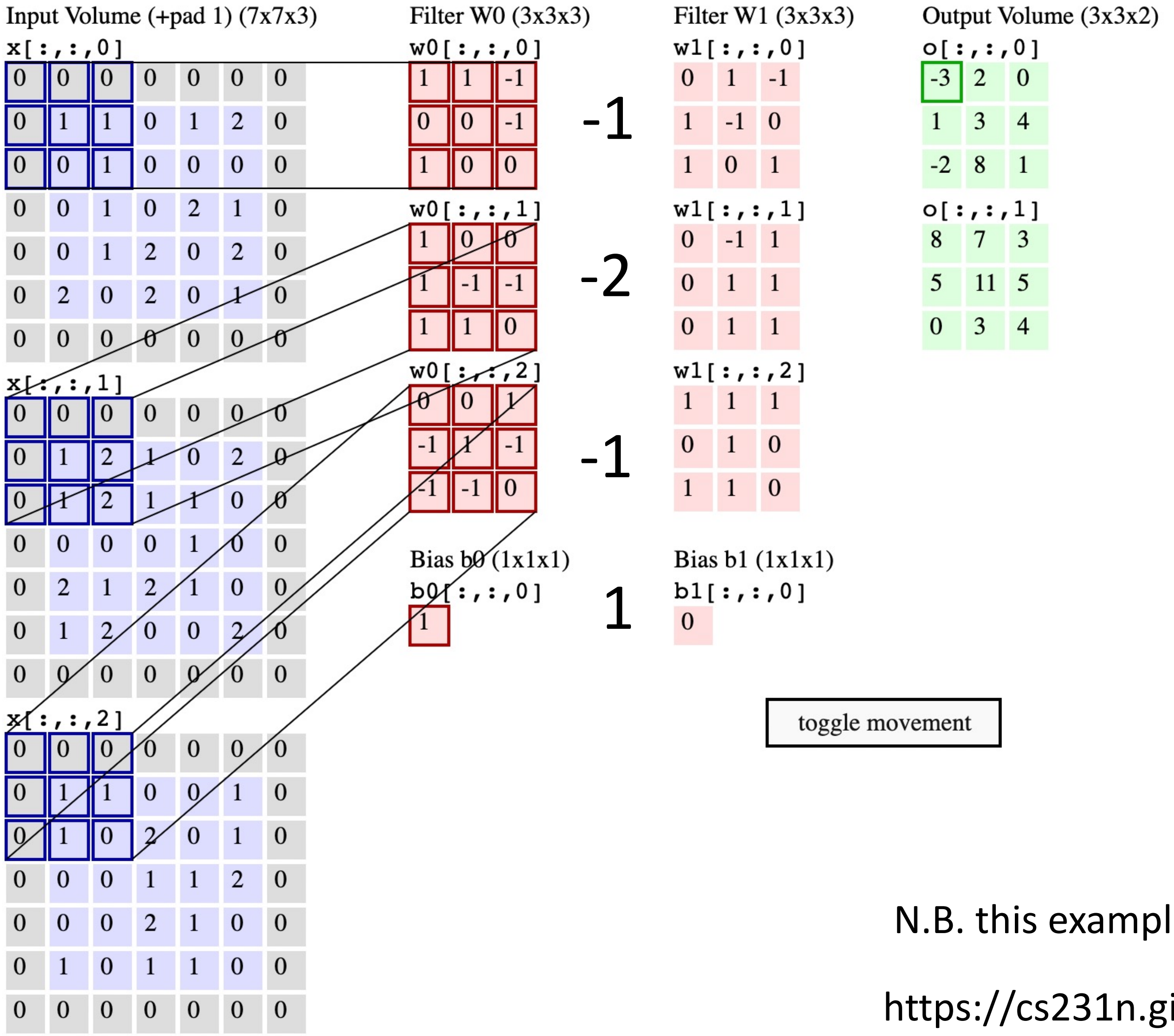



$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} * \boxed{\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}} = \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \end{bmatrix} * \boxed{\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}} = \begin{bmatrix} 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \end{bmatrix}$$

$n \times n$ image $\quad$ $f \times f$ filter $\quad$ $\text{out} = n - f + 1$

# Convolution Layer (CONV)




Hyperparameters:

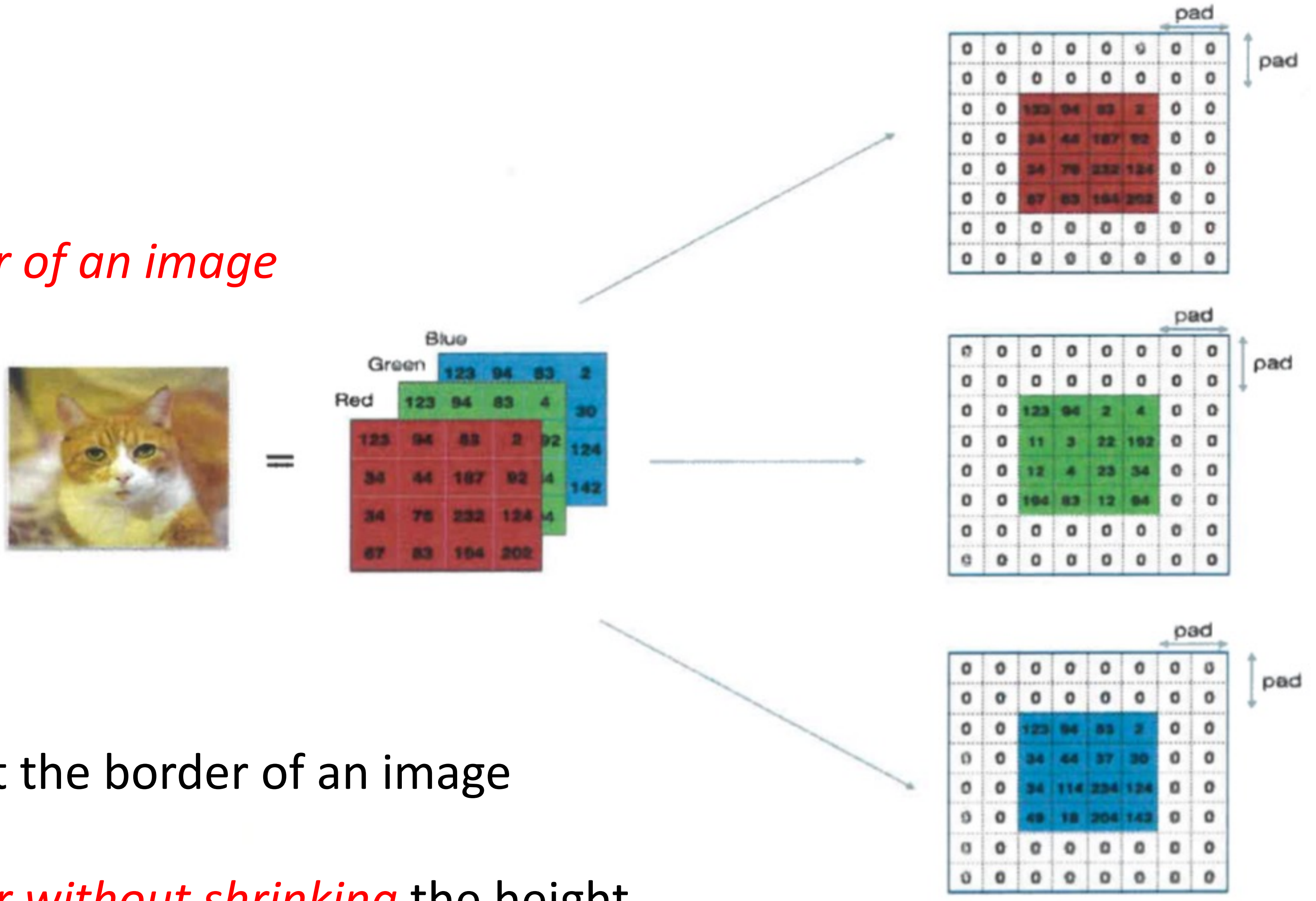
- Filter size
- Number of filter sets (channels)
- Padding
- Stride

N.B. this example the **stride** = 2, **padding** = 1, **channels = 2**

https://cs231n.github.io/convolutional-networks/

# Padding

*Adds zeros around the border of an image*

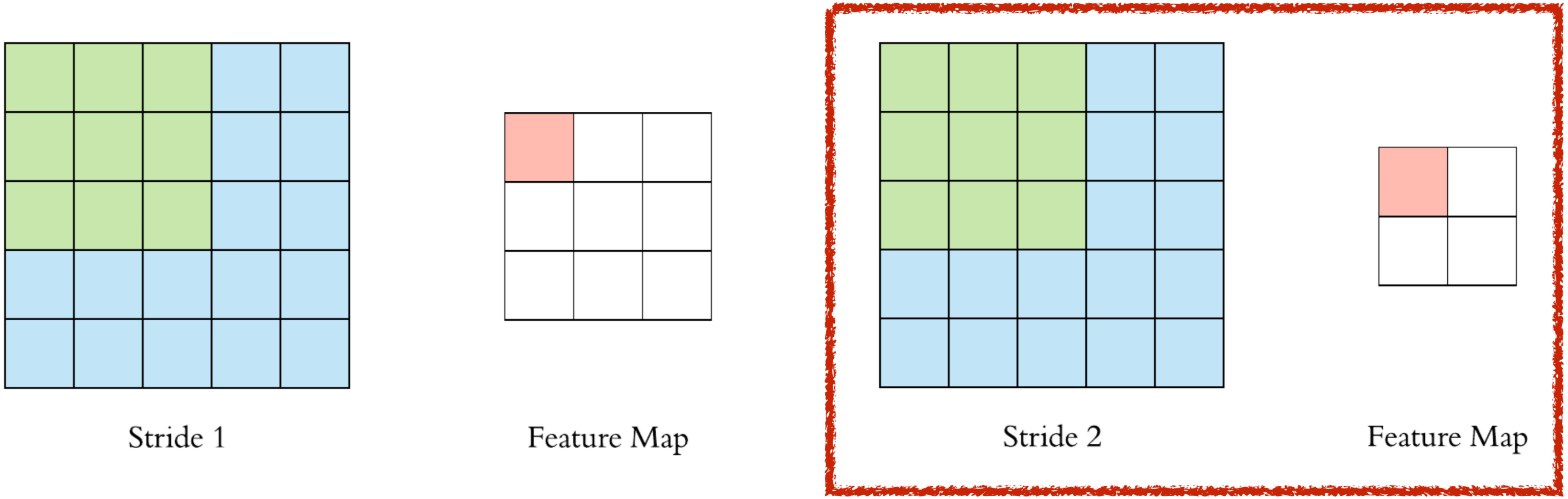



**Use cases :**

- 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)

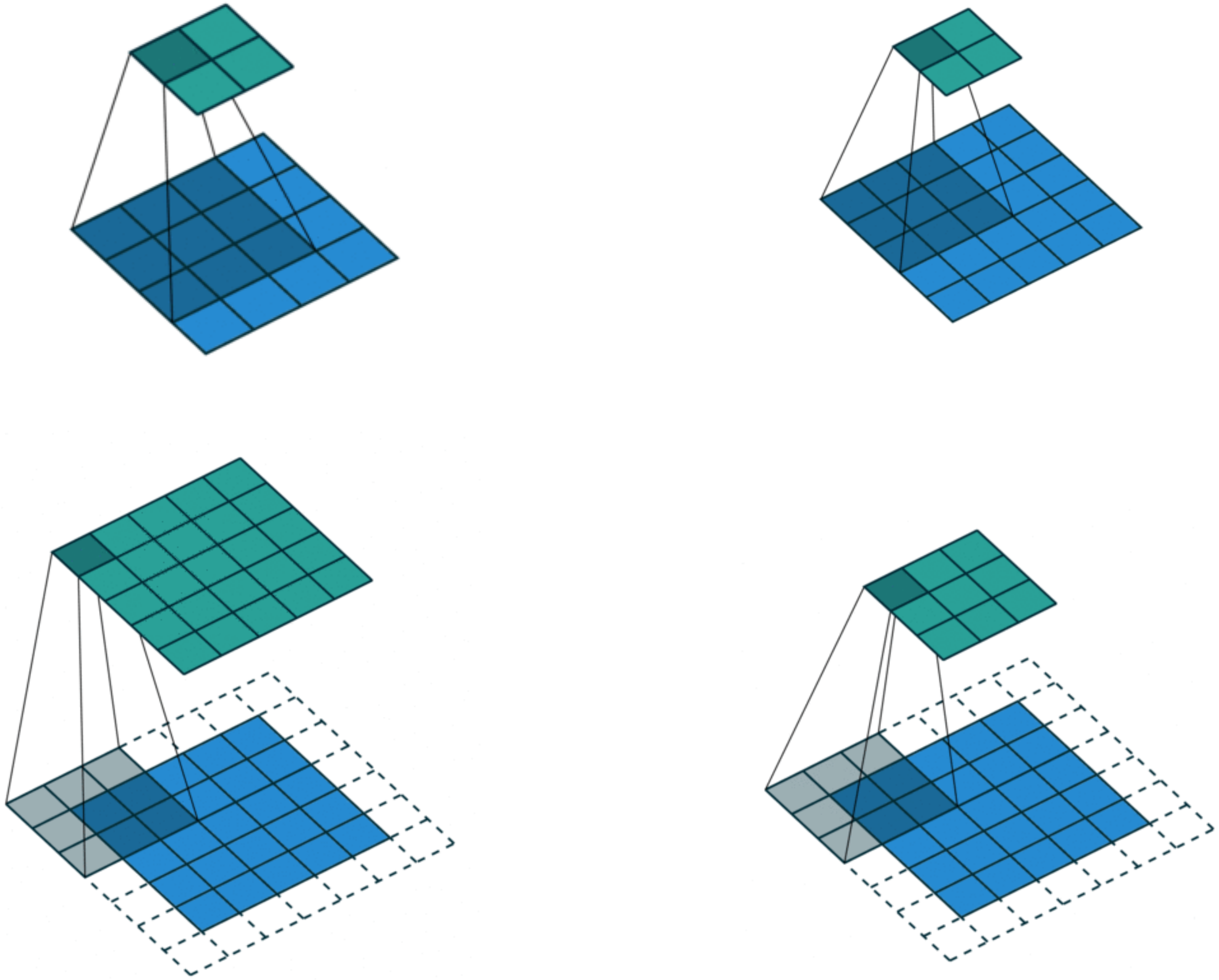
# Strided convolutions

*By how much you move the filter*




**increasing stride from 1 to 2**

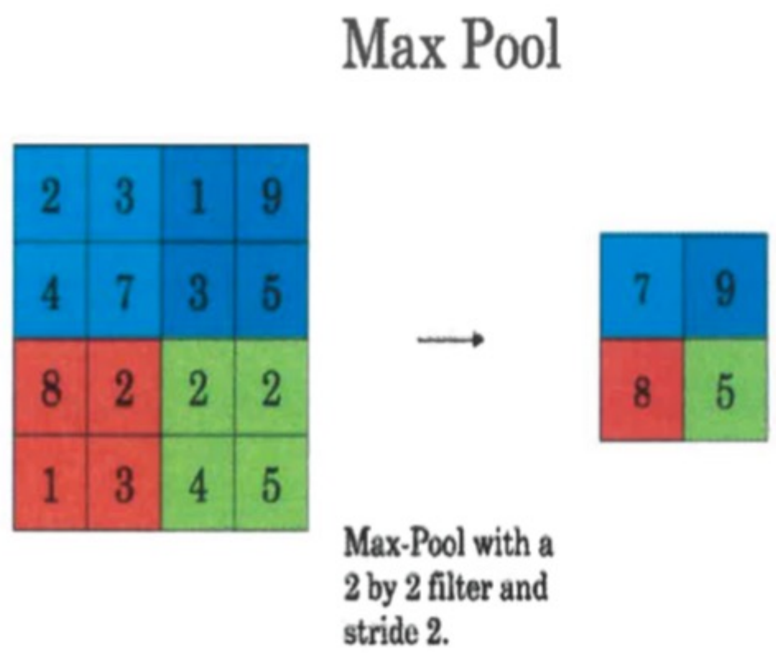
# Illustration



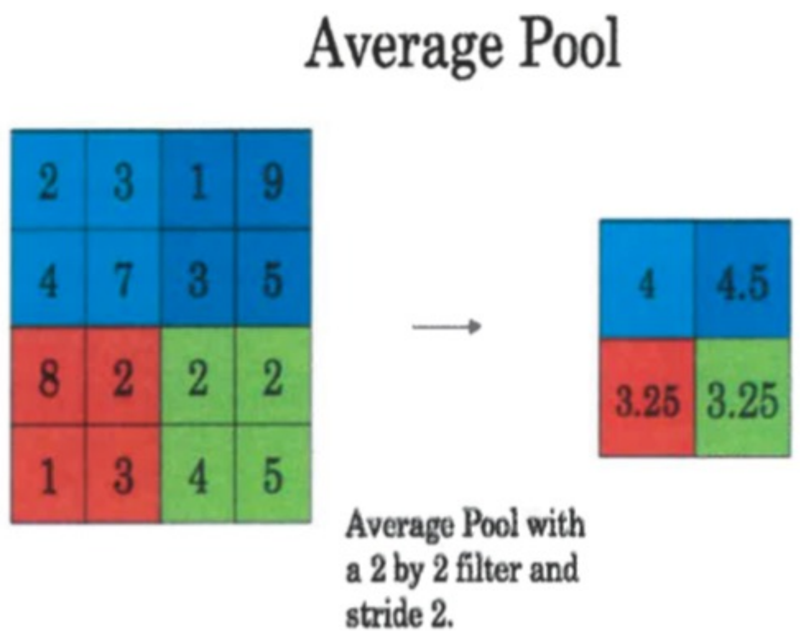
https://hannibunny.github.io/mlbook/neuralnetworks/convolutionDemos.html

# Pooling Layer (POOL)

- reduces the spatial dimension ($n_H$ and $n_W$) to decrease computational power




*Get the max value*




*Get the average value*

# 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?

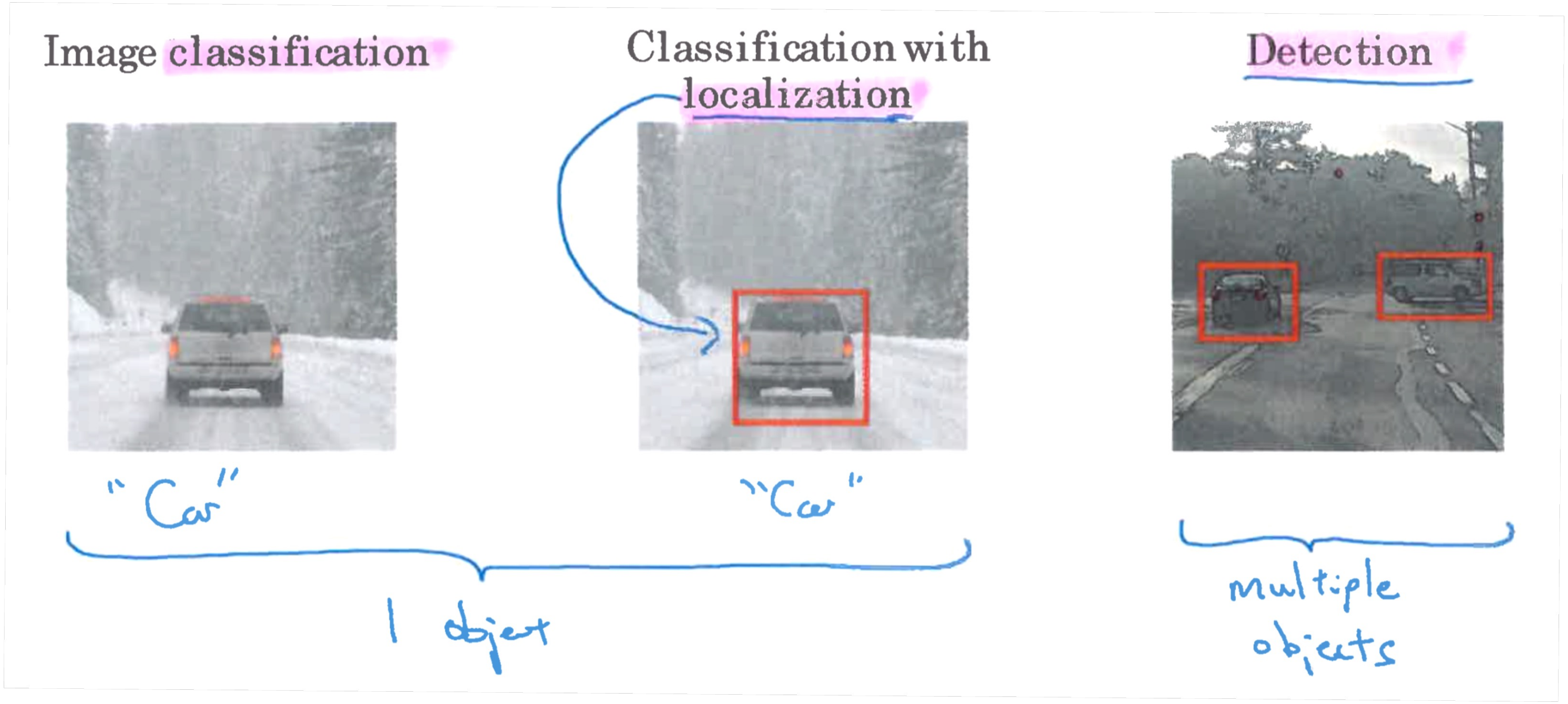
What's the use! Call and see what you are buying. One look at our display of automobile and motorcycle accessories will convince you of the fact.

THAT WE HAVE EVERYTHING FOR THE AUTO

## Piqua Auto Supply House

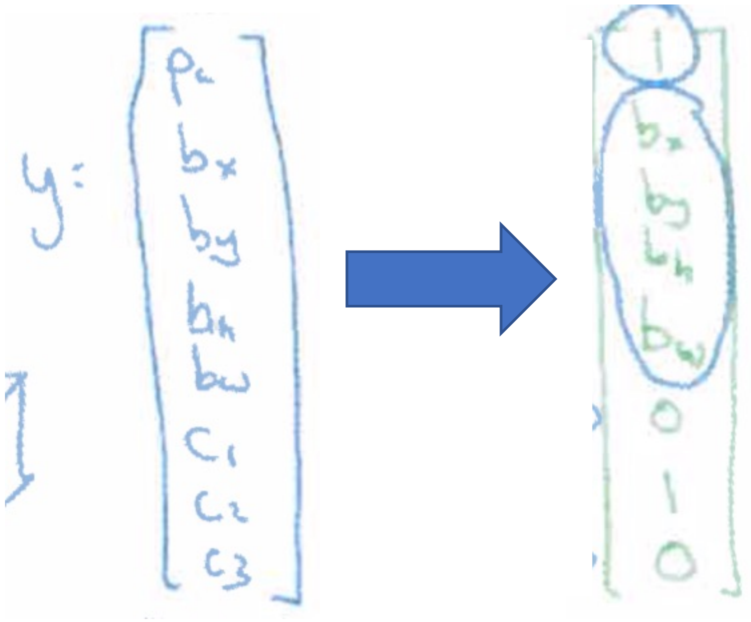
133 N. Main St.—Piqua, O.

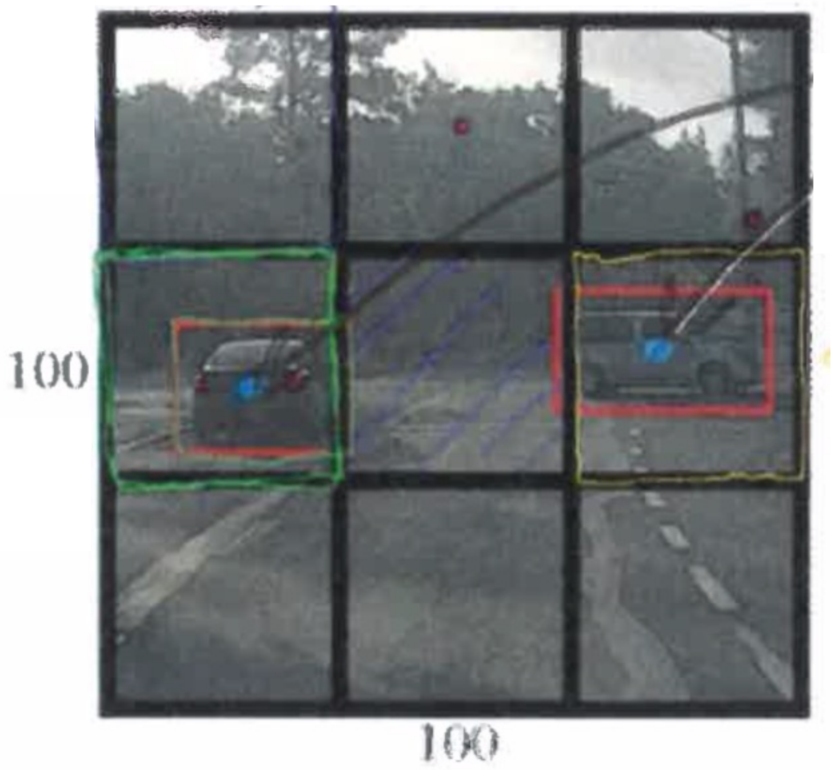
# Object Detection



# Yolo (*YOu Look only Once*)

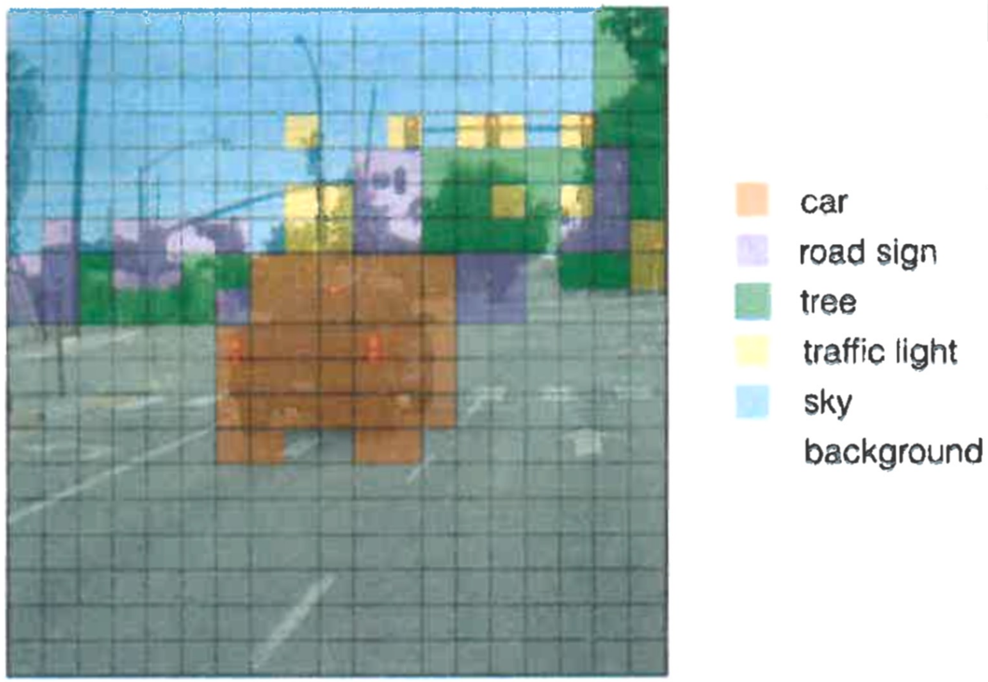
- define a grid in the image
- apply the training to each cell (need ground-truth bounding boxes)
- For each ‘anchor box’ and 3 classes we have:

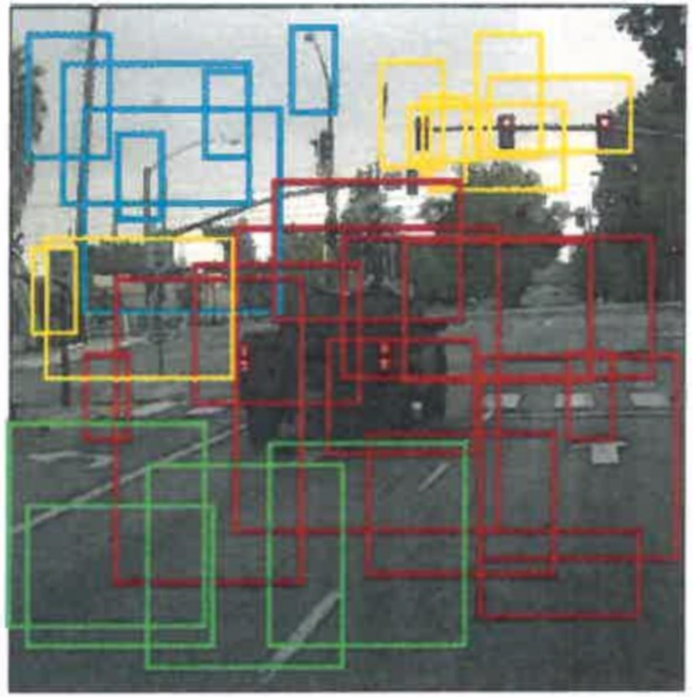



- Allows for overlapping objects

# YOLO prediction visualisation

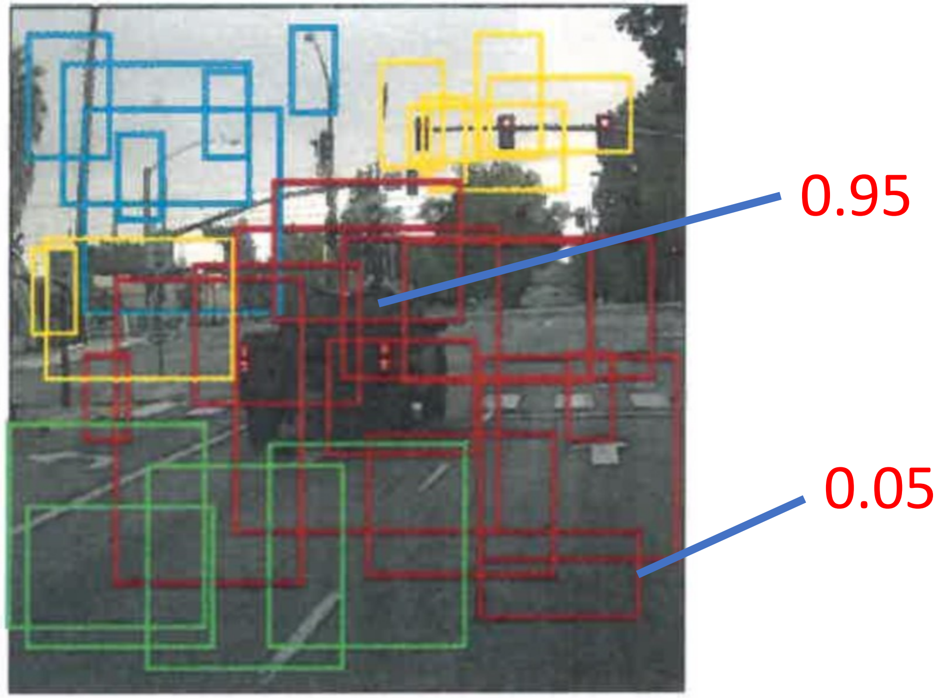


- Filter the boxes using :
  1) score thresholding
  2) non-max suppression

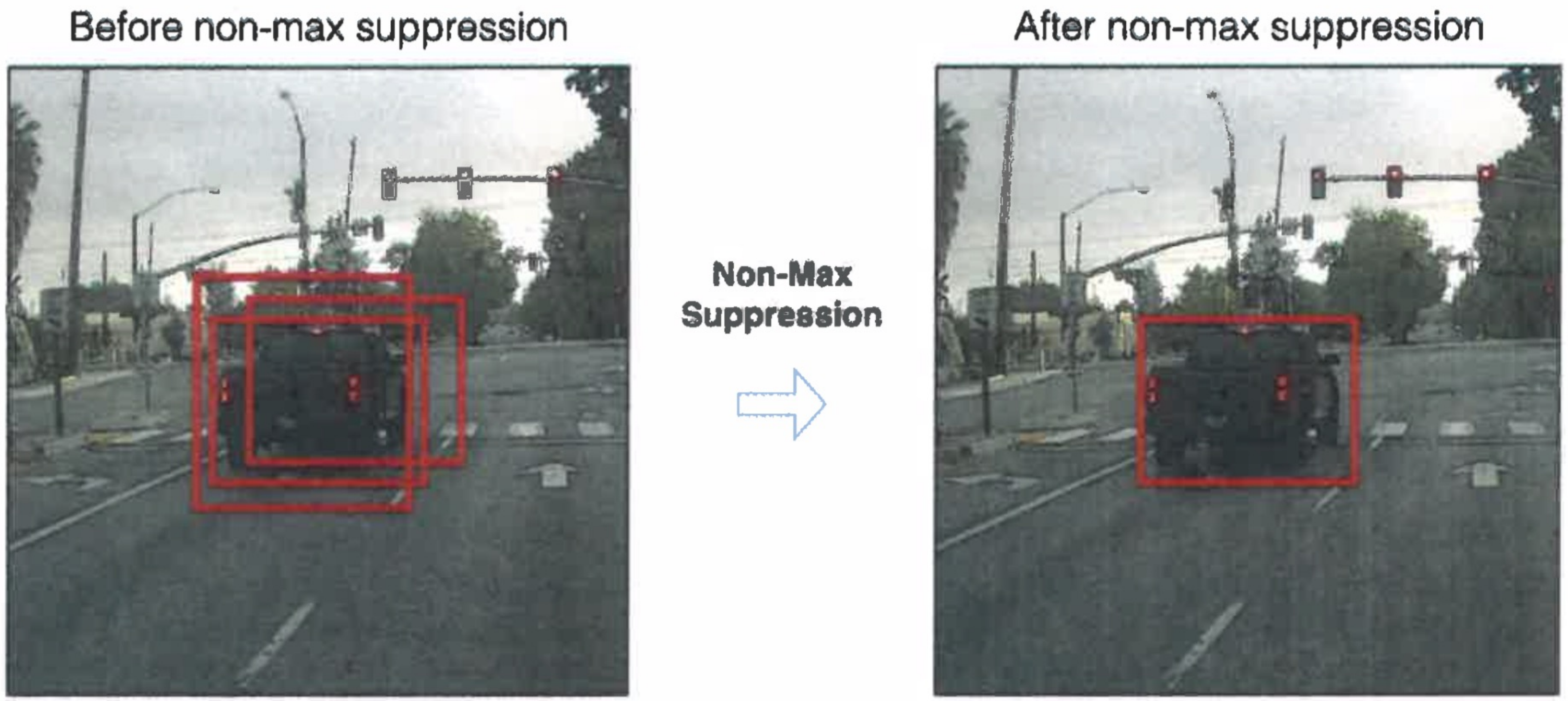
# Score Thresholding

- Throw away boxes that have detected a class with a score less than the threshold (*0.6 for example*)



# Non-max suppression

- ensures that an object is detected only once
  - Choice based on the $p_C$ value : *keep the largest $p_C$ output* and *discard any remaining box with IoU>0.5*

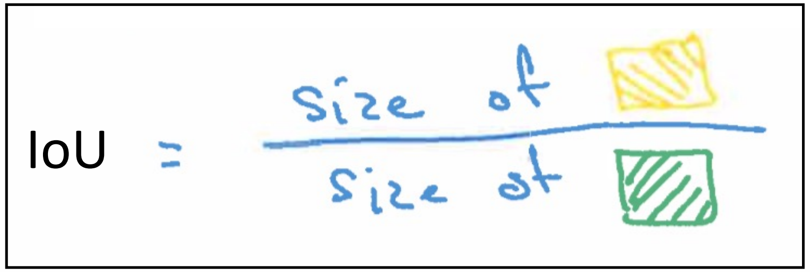


# Intersection Over Union (IOU)

- performance metric on how similar two boxes are with each other
- (higher the better!)

Outcome of the algorithm

True bounding box



# Intersection Over Union (IOU)

Yellow = intersection I
Green = union U

IoU  = I/U

Can express this as:

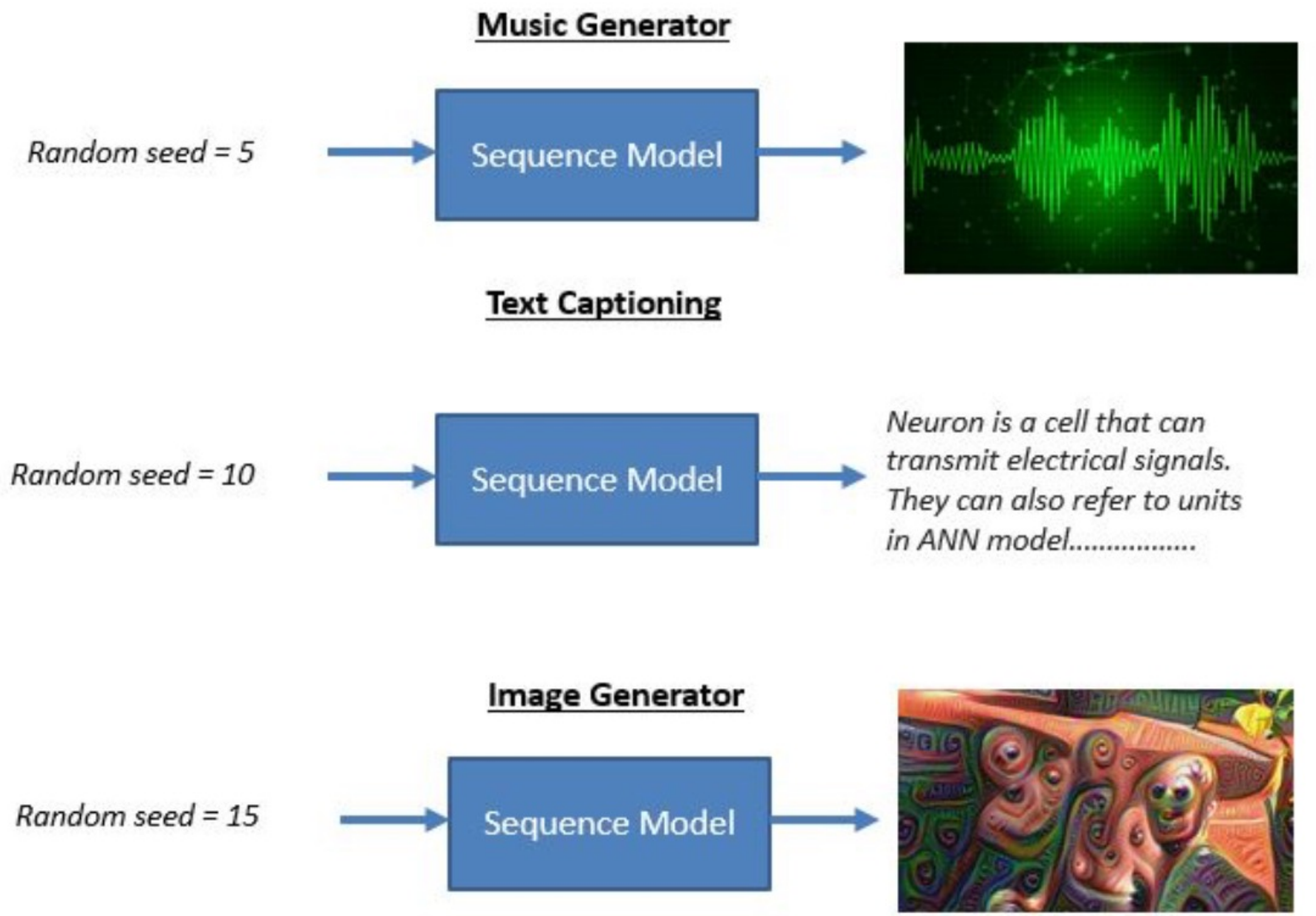
TP / (TP + FN + FP)

[see also, Jaccard Index]

# Generative Models

Holy grail of Deep Learning these days

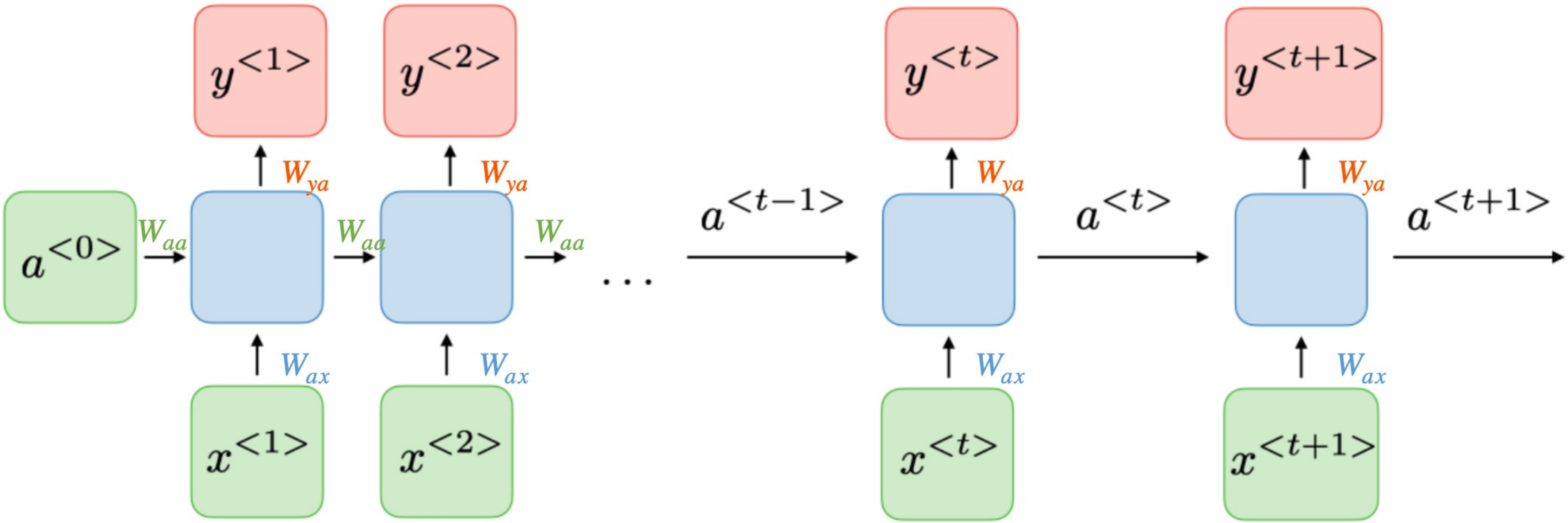
# Generative Models Examples




[DeepBach]

[GauGAN]

# RNN model

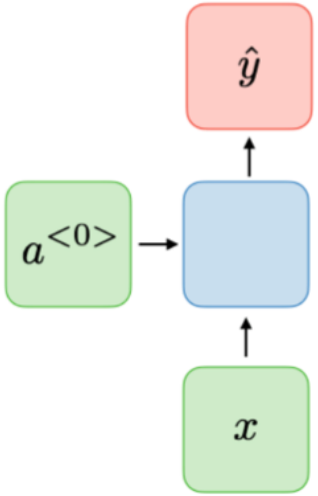



$$\boxed{a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)}$$

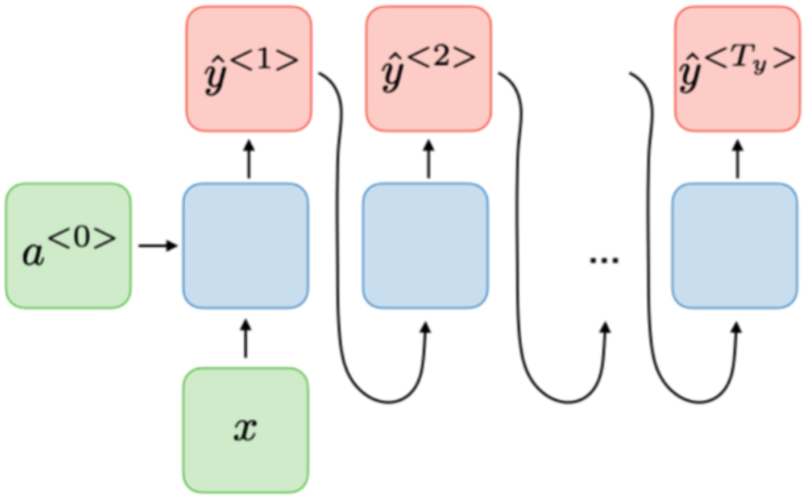
$$\boxed{y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)}$$

$W_{ax}$ ,$W_{aa}$, $W_{ya}$, $b_a$ and $b_y$ are weights that are shared temporally and $g_1$, $g_2$ activation functions

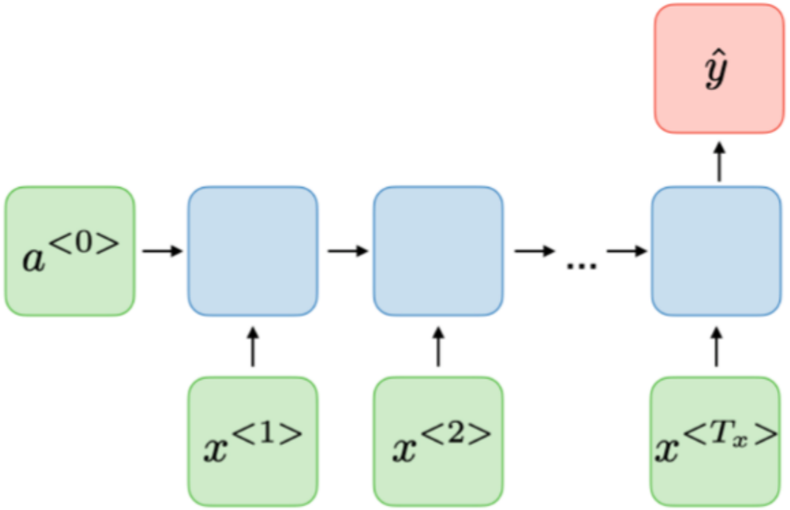
# Applications of RNN

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

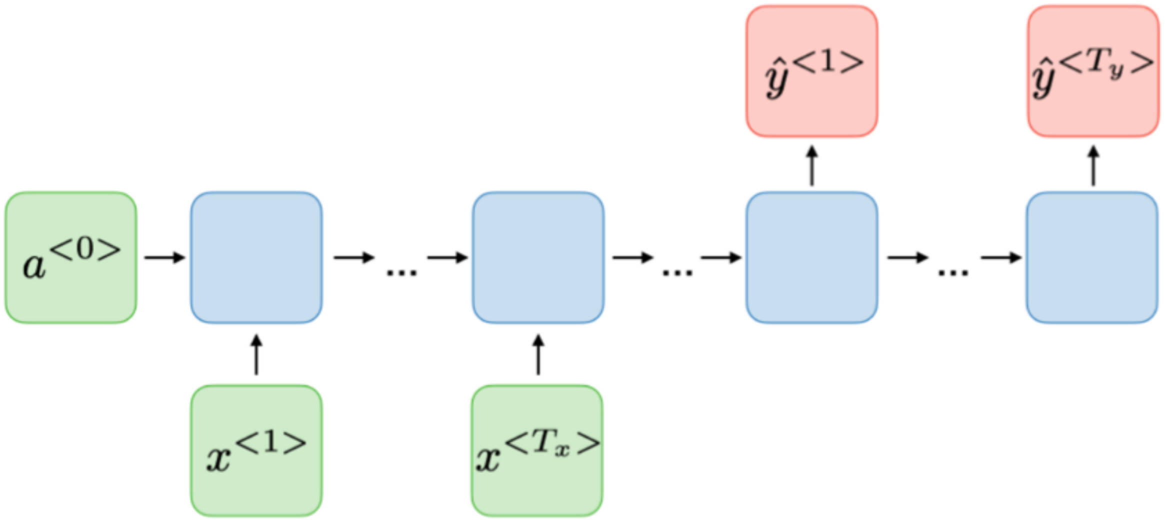
# Applications of RNN

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

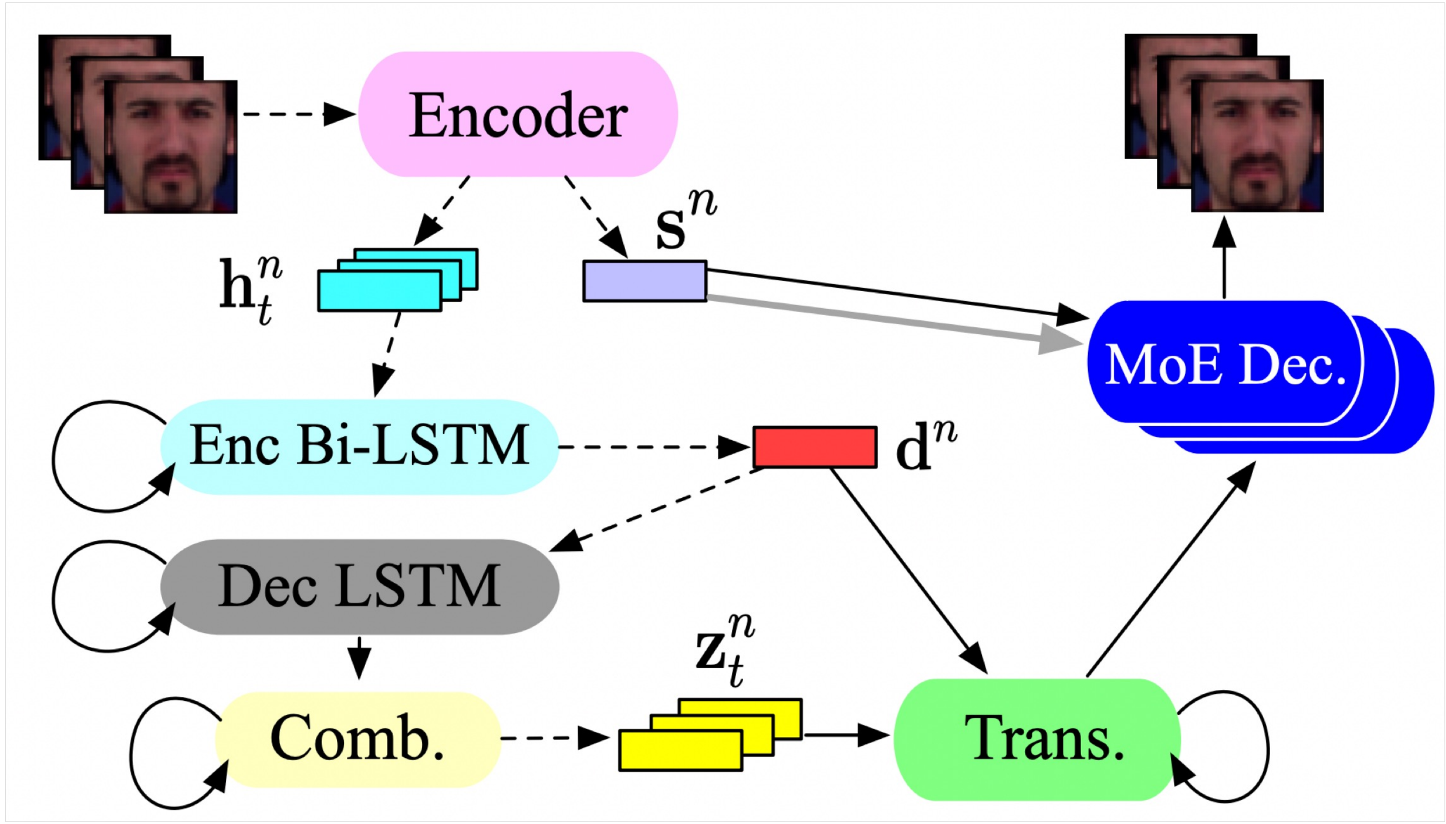
# Applications of RNN

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

# Applications of RNN

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

# VDSM




| | |
|---|---|
| $n$ | **Individual** |
| $t$ | **Video Frame** |
| $\mathbf{d}^n$ | **Action Performed** |
| $\mathbf{s}^n$ | **Identity** |
| $\mathbf{z}_t^n$ | **Per-Frame Pose** |
| $\mathbf{h}_t^n$ | **Per-image embedding** |

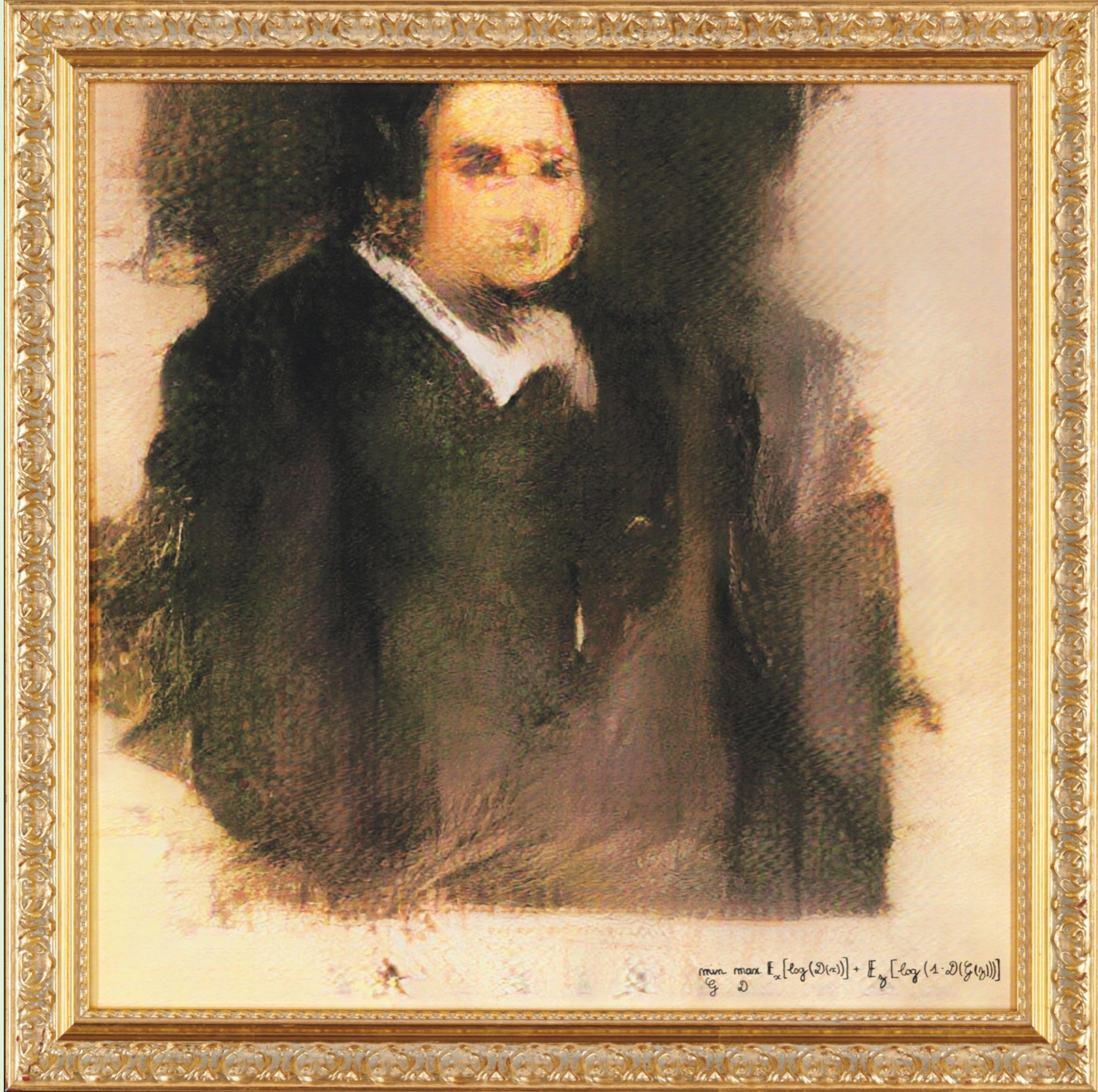
x

# Generative Adversarial Network

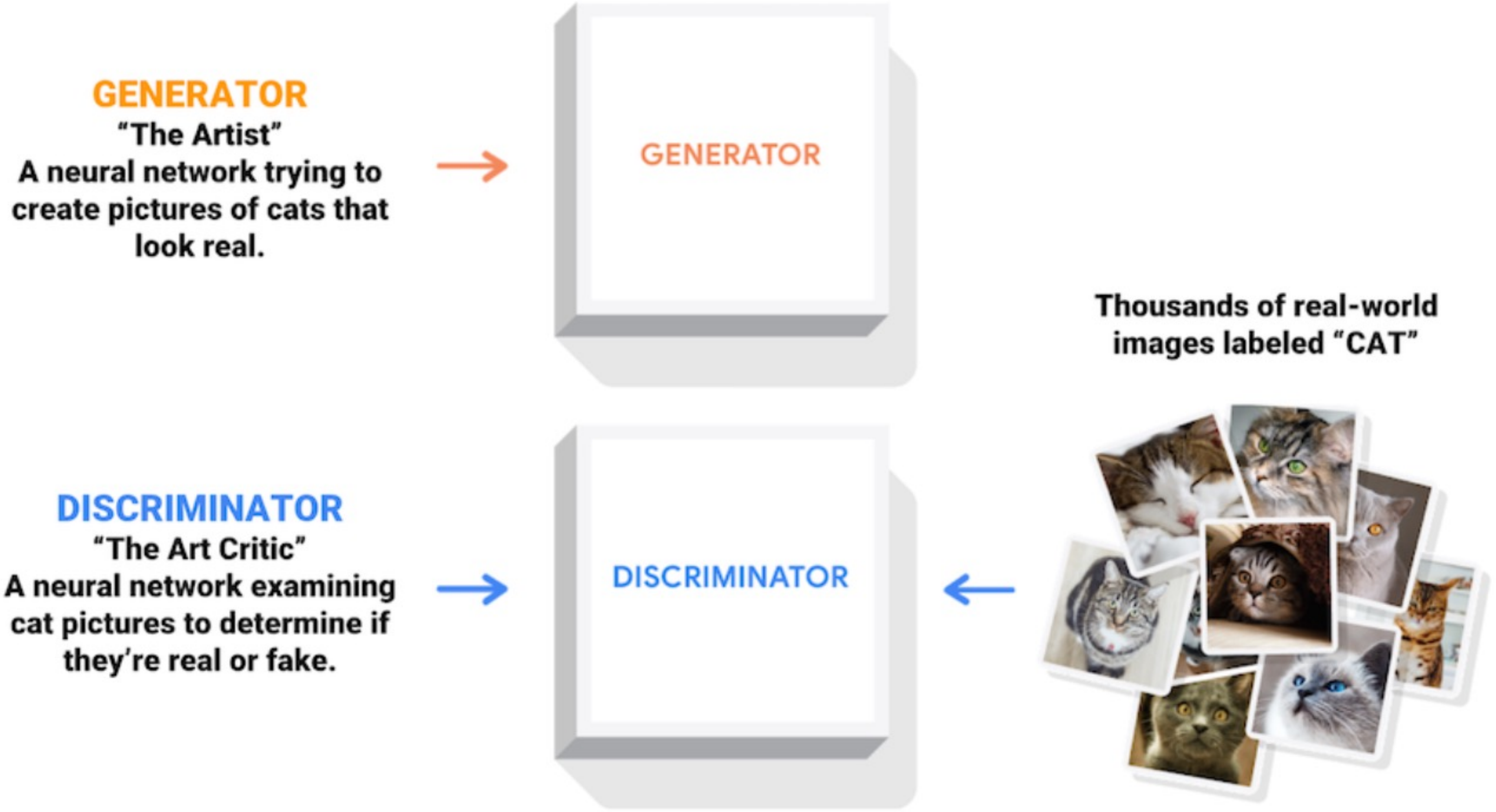
Invented by Ian Goodfellow

*How much are you ready to pay for it ?*

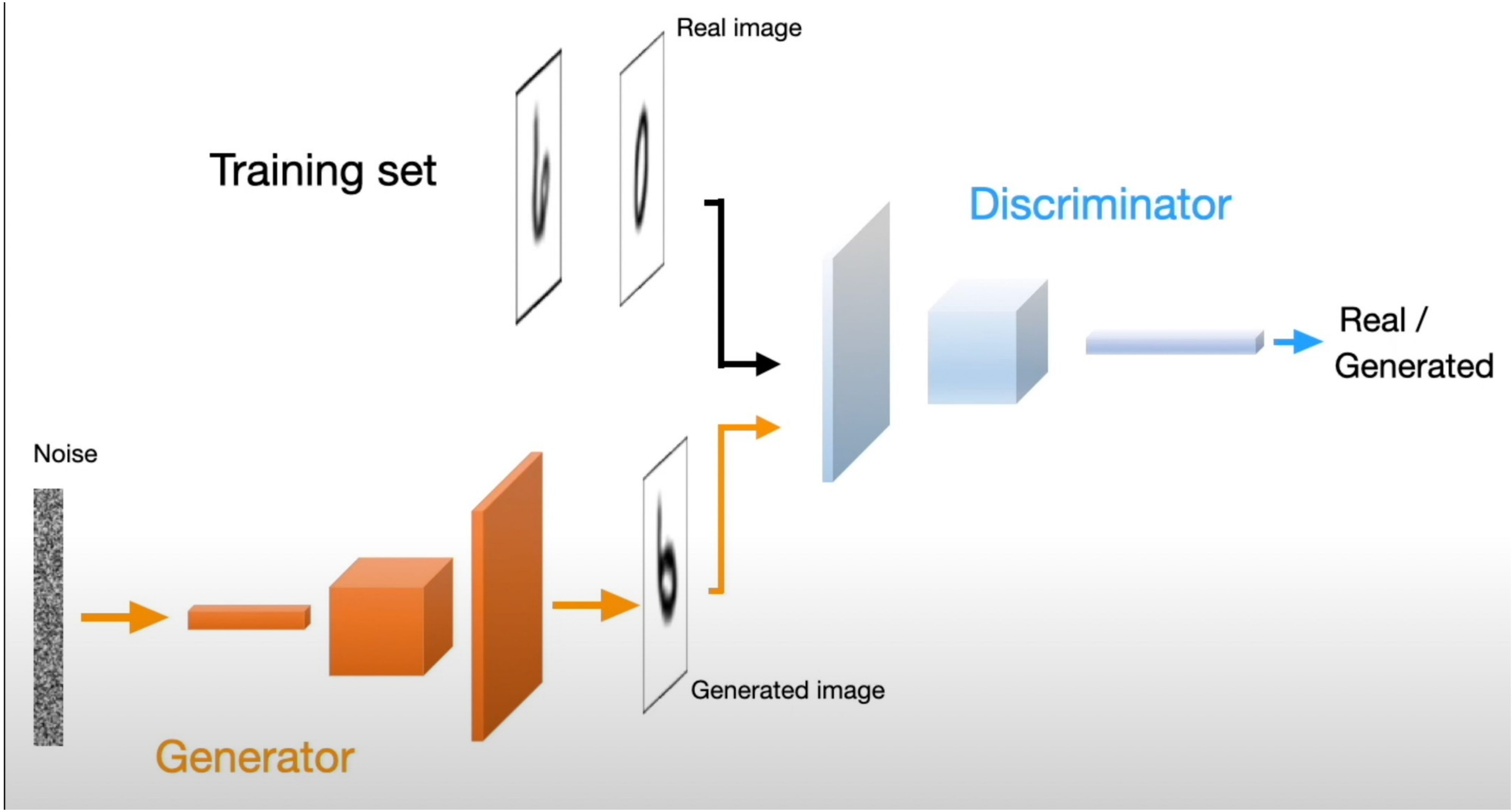
Christie's New York 2018



# Principle


GENERATOR
"The Artist"
A neural network trying to create pictures of cats that look real.
GENERATOR
Thousands of real-world images labeled "CAT"
DISCRIMINATOR
"The Art Critic"
A neural network examining cat pictures to determine if they're real or fake.
DISCRIMINATOR

# Principle




https://youtu.be/-Zi5SReze6U

# Principle

Step 1: Train Discriminator and 'Freeze' Generator parameters
Step 2: Train Generator and 'Freeze' Discriminator parameters

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

https://youtu.be/-Zi5SReze6U

# Principle

Step 1: Train Discriminator and ‘Freeze’ Generator parameters
Step 2: Train Generator and ‘Freeze’ Discriminator parameters

$$\min_{G} \max_{D} V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Discriminator gradient for update (gradient ascent):

$$\nabla_{\mathbf{w}_D} \frac{1}{n} \sum_{i=1}^{n} \Big[ \underbrace{\log D\left(\boldsymbol{x}^{(i)}\right)}_{\substack{\text{predict well on real images} \\ \text{=> want probability close to 1}}} + \underbrace{\log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)}_{\substack{\text{predict well on fake images} \\ \text{=> want probability close to 0}}} \Big]$$

https://youtu.be/-Zi5SReze6U

# Principle

Step 1: Train Discriminator and 'Freeze' Generator parameters
Step 2: Train Generator and 'Freeze' Discriminator parameters

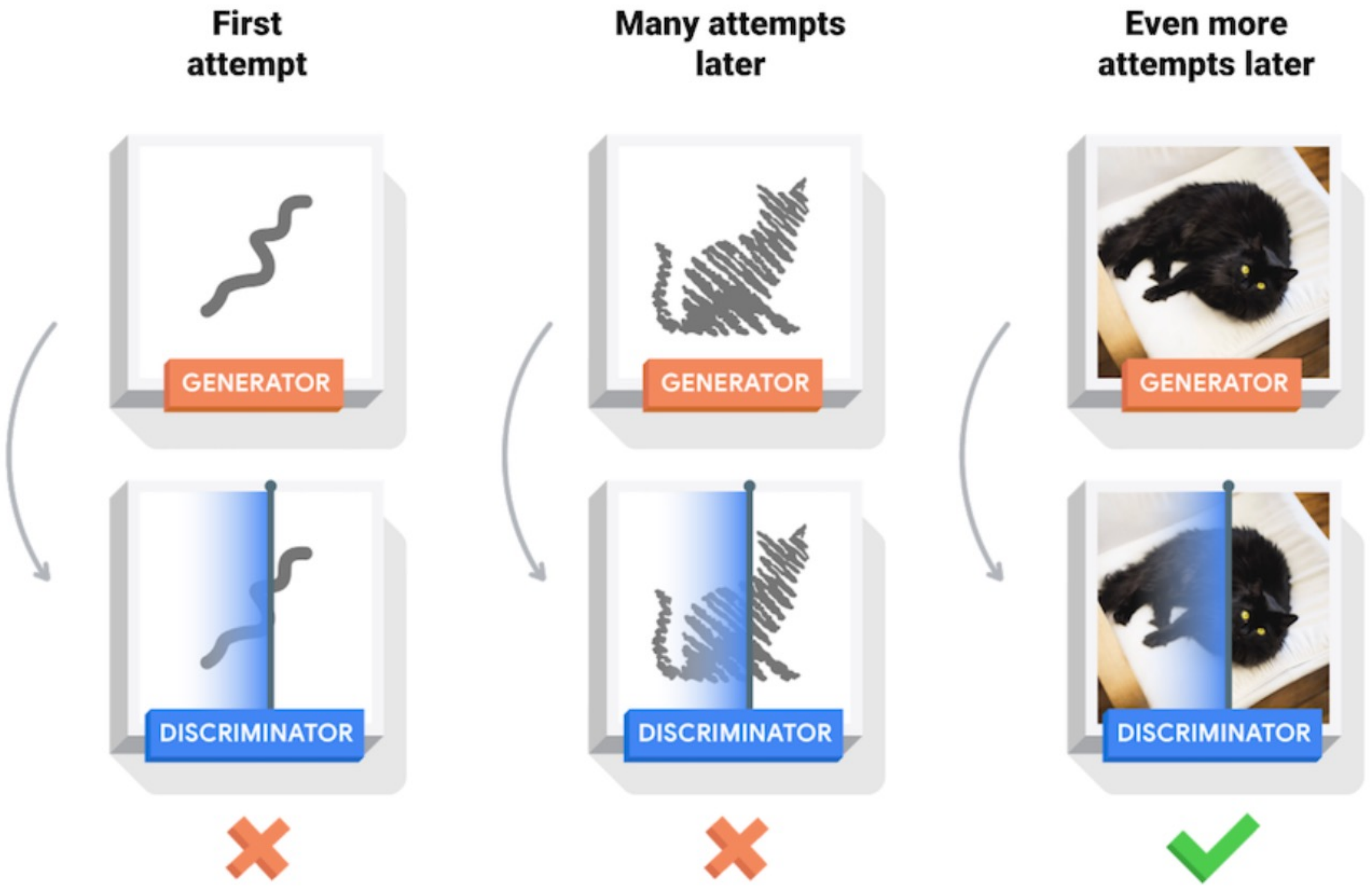
$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

**Generator gradient for update (gradient descent):**

$$\nabla_{\mathbf{W}_G} \frac{1}{n} \sum_{i=1}^{n} \log \left(1 - \overbrace{D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)}^{\text{predict badly on fake images => want probability close to 1}}\right)$$

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

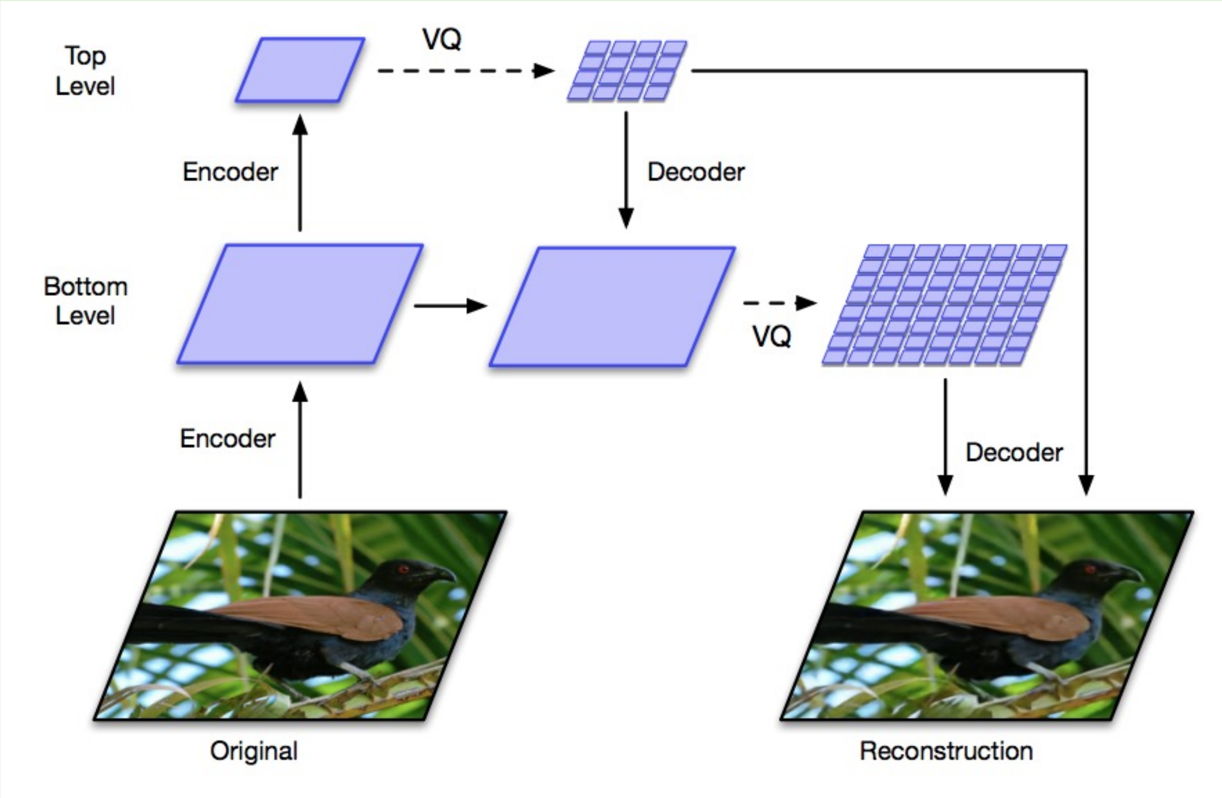
(Karras et al, 2018)

(Brock et al, 2018)

# Adversarial Principles are Widely Applicable

- Fairness / Privacy Preservation
- Disentanglement

# Variational AutoEncoder



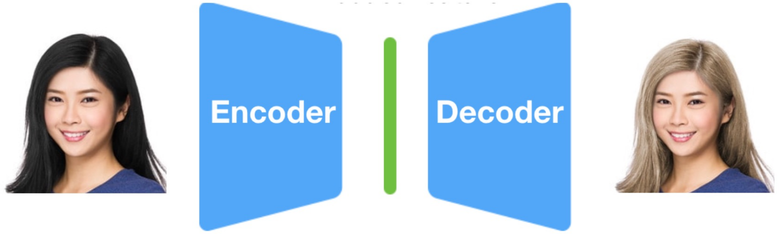

VQ-VAE (2) (van den Oord et al. 2017, Razavi et al. 2019)

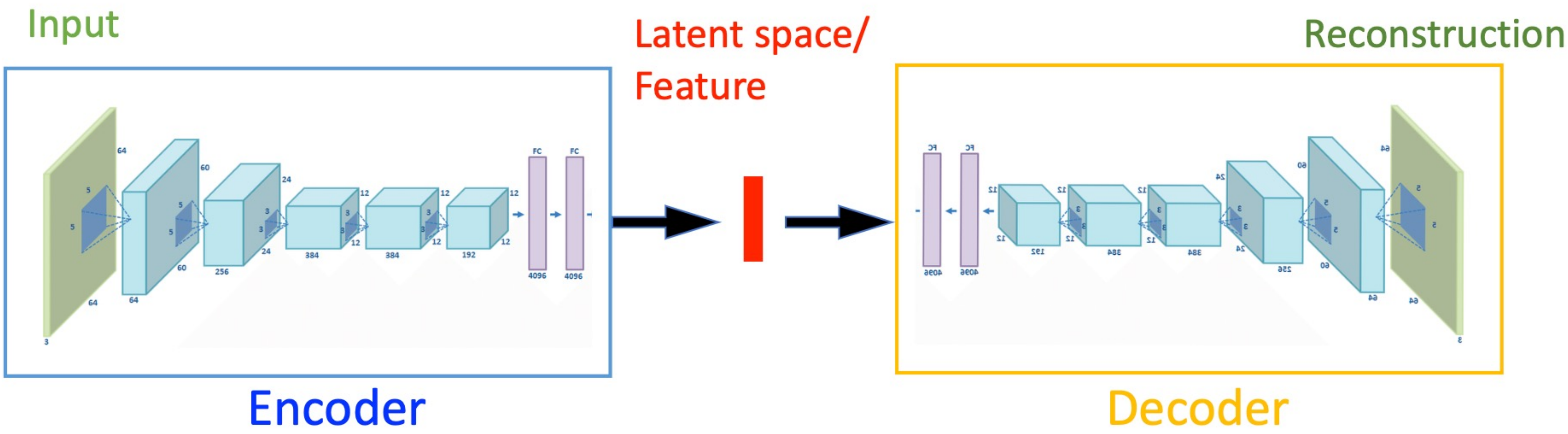
# Principle

## AutoEncoder

### Data Generation

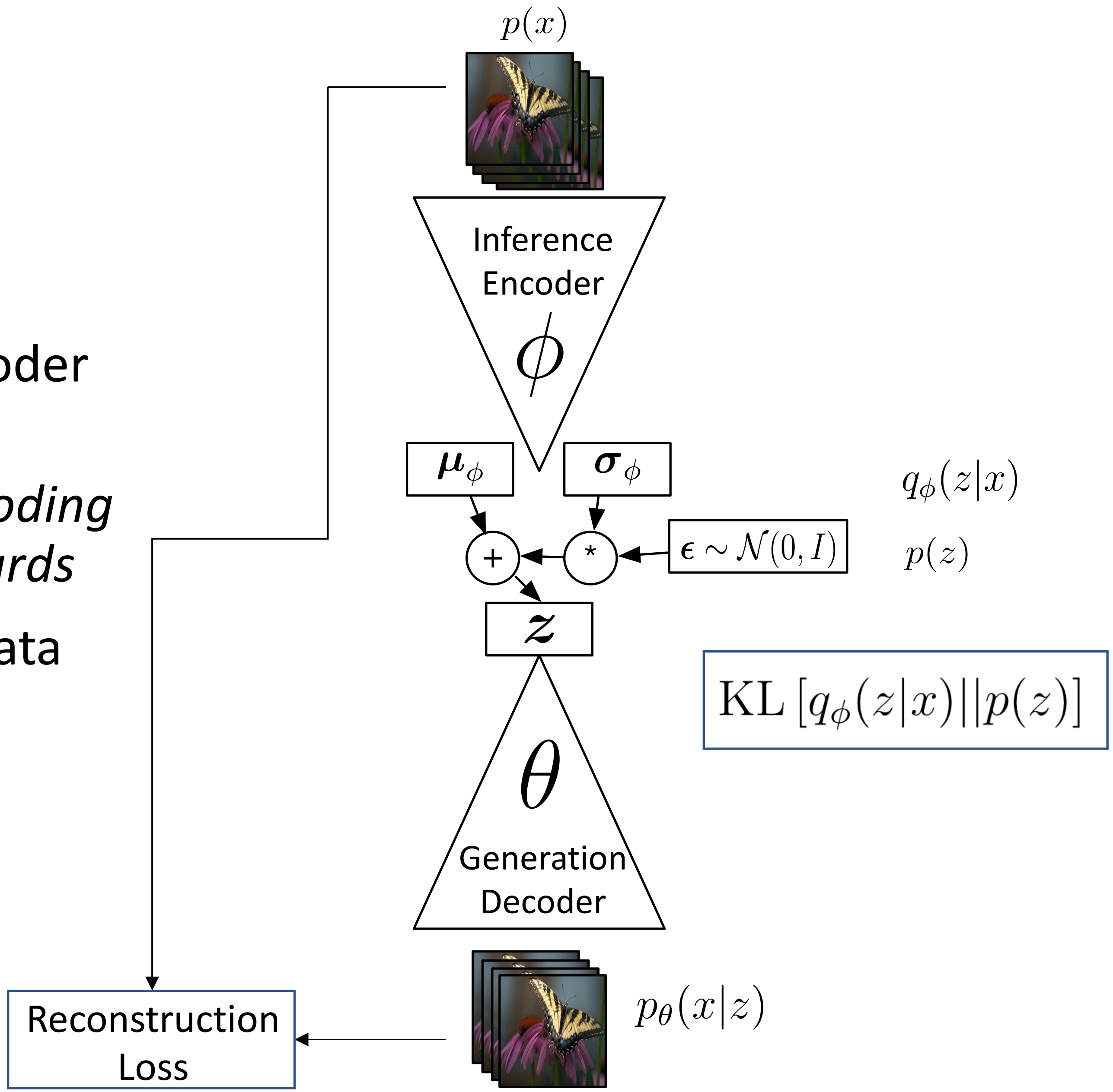
- generate appropriately novel data



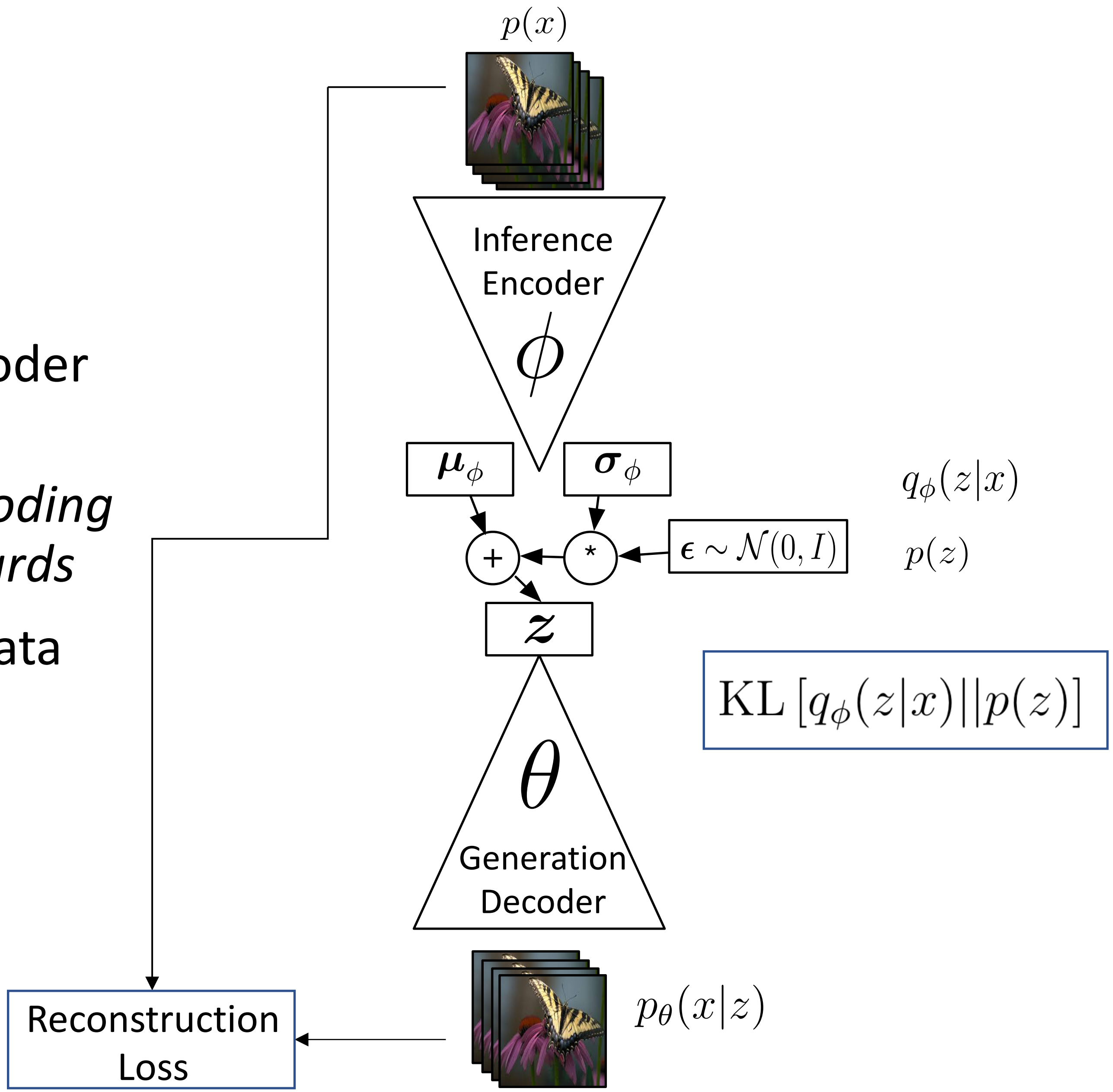
# Principle

- Pass images/data in through encoder *'bottleneck'*
- *Parameterize this bottleneck encoding so we can sample from it afterwards*
- Reconstruct the original image/data from the encoding



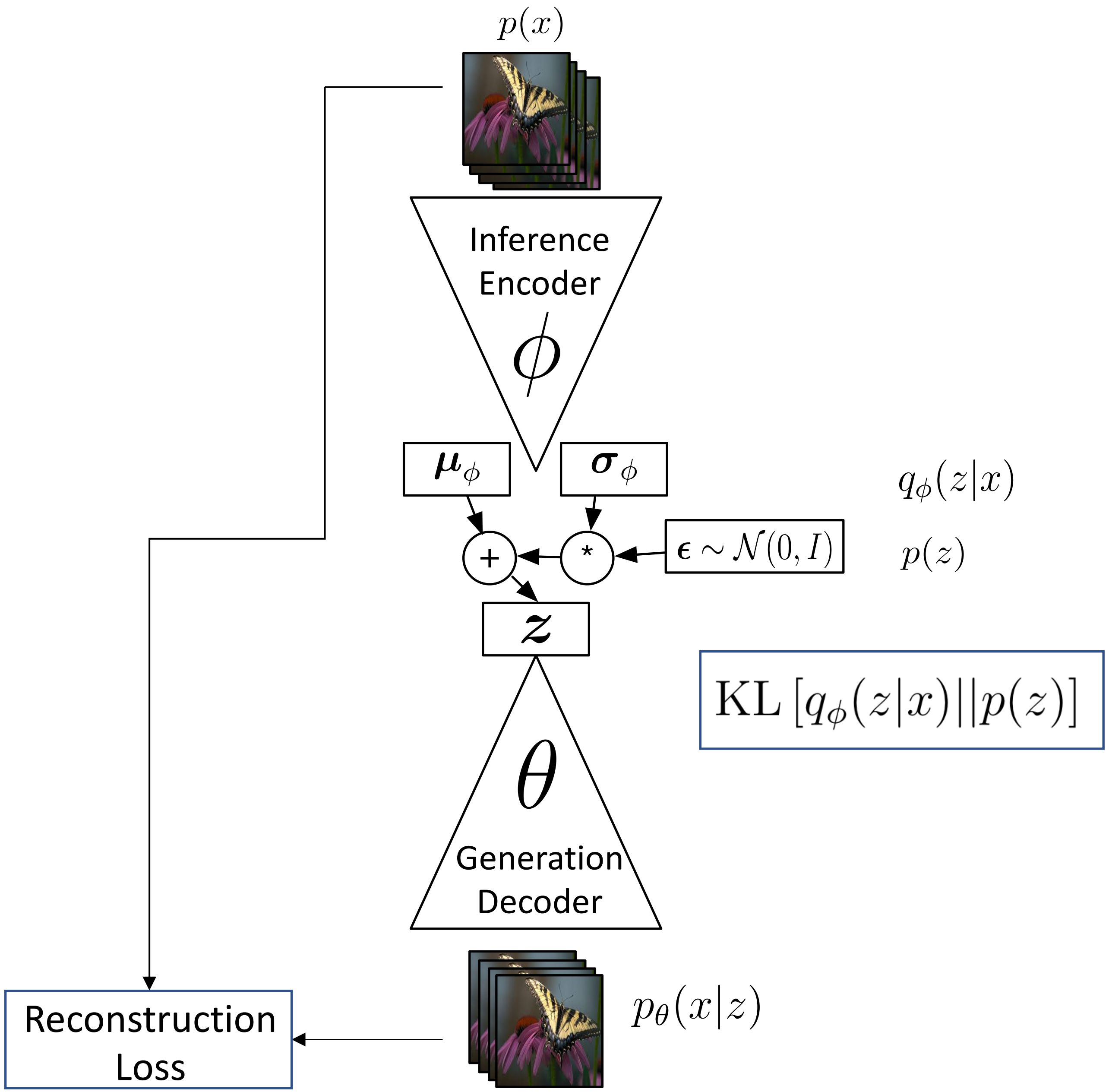
# Principle

- Pass images/data in through encoder *'bottleneck'*
- *Parameterize this bottleneck encoding so we can sample from it afterwards*
- Reconstruct the original image/data from the encoding

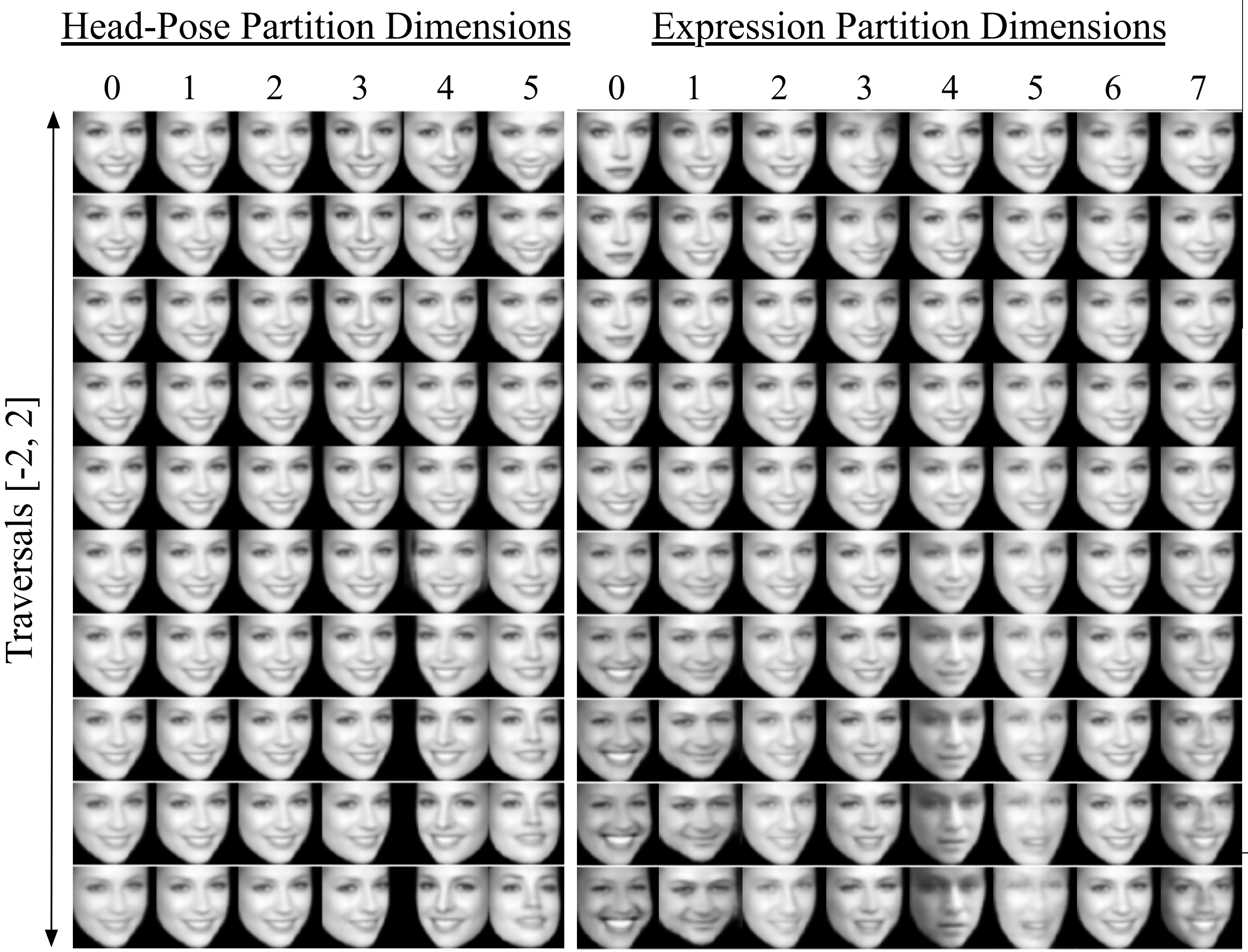


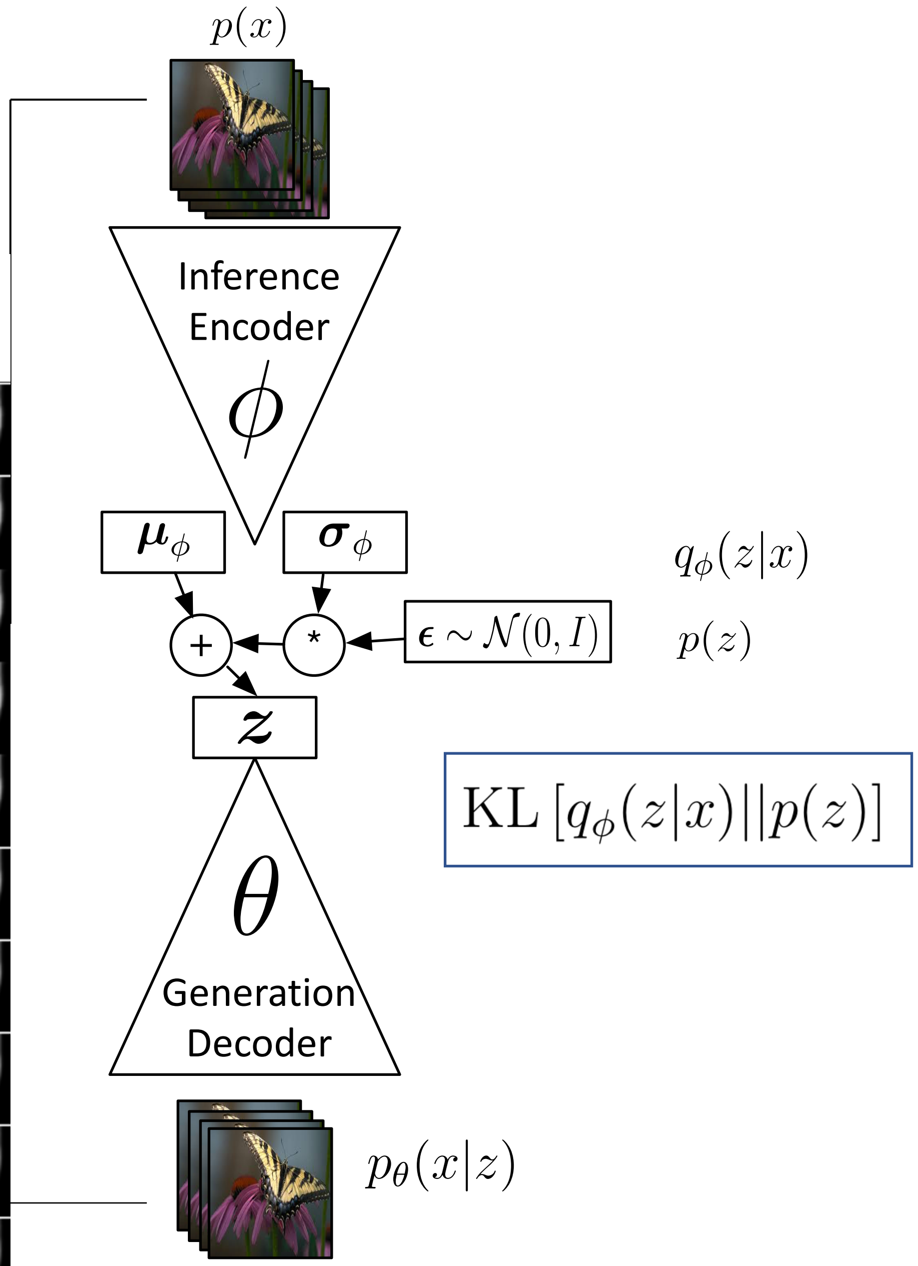
# VAE use cases

- Compression
- Data generation
- Latent variable modeling
- Density estimation



# VAE use cases



# Tutorial / Practical