# Computer Vision

01 **Computer Vision**
Introduction
Evaluation of models

02 **Transfer Learning & Data Augmentation**
State of the art models

03 **Robustness**
Adversarial Attacks

04 **Generative Models**
GAN & VEE

# Team


Olivier Randavel

**Data Scientist**

@ olivier.randavel@gmail.com



Hippolyte Mayard

**Data Scientist**

@ hippolyte.mayard@dauphine.eu

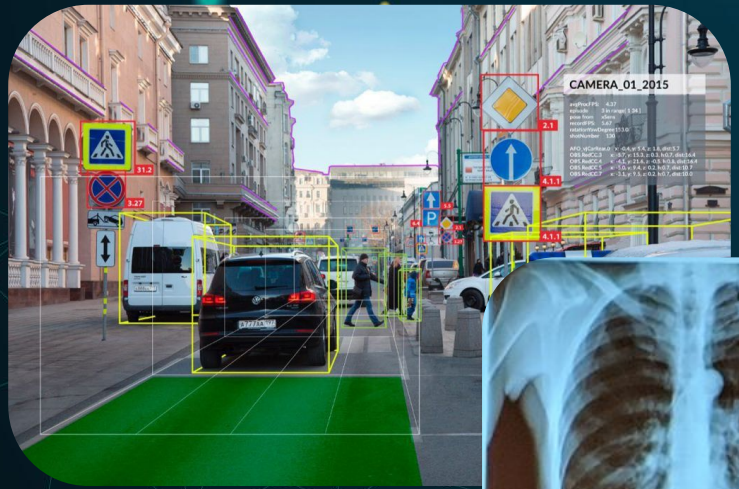
# Computer Vision

# 01

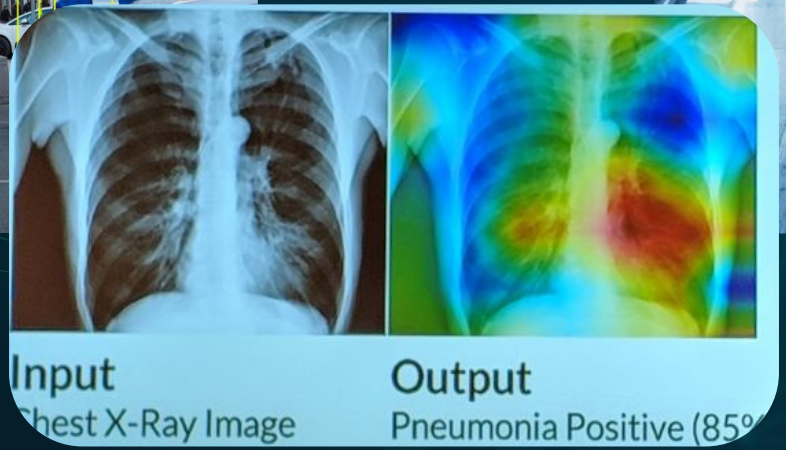
Introduction
Evaluation of models

# Applications

Autonomous Vehicles




Face Recognition


Radiology

# Aims

Race identification

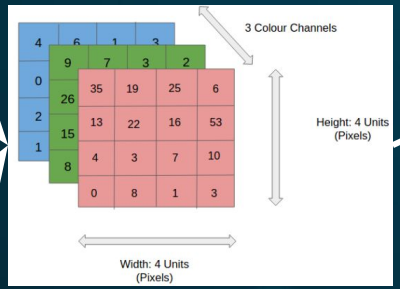
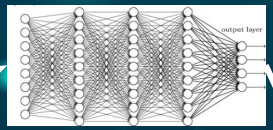



Image to numbers using pixel representations

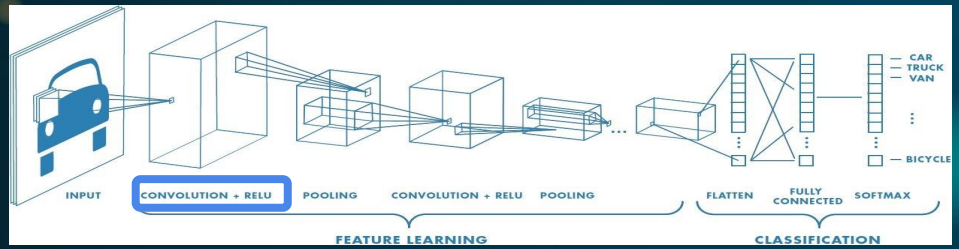
Deep Neural network



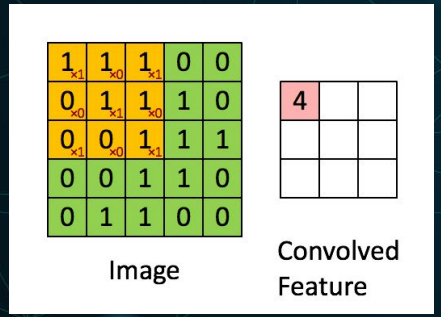


Classification

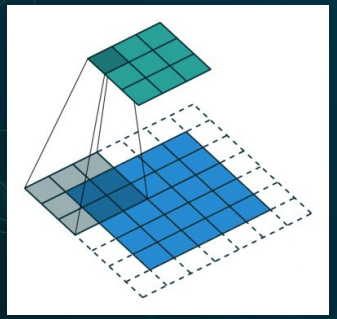
# Neural Network



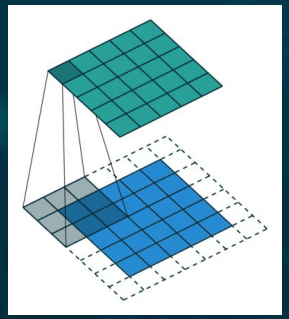

The aim is to train specialised weights in order to detect shape, colors through convolutional layers.



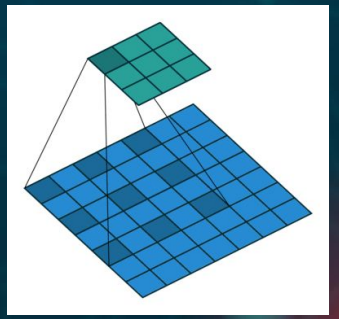

Kernel : reduce dimension



Stride : detect edges

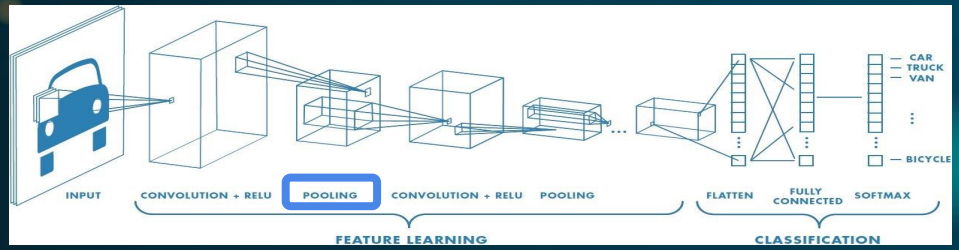


Padding: keep information from borders. Increase or keep same length

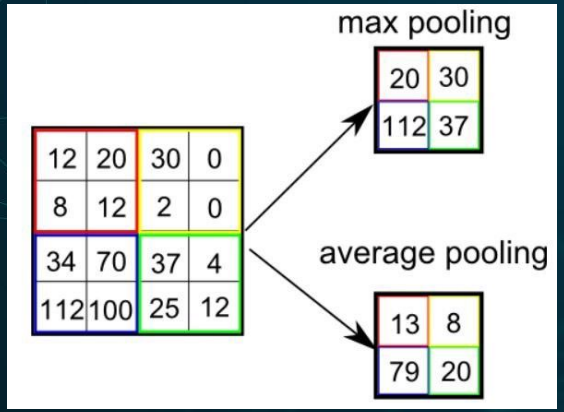


Dilation: segmentation

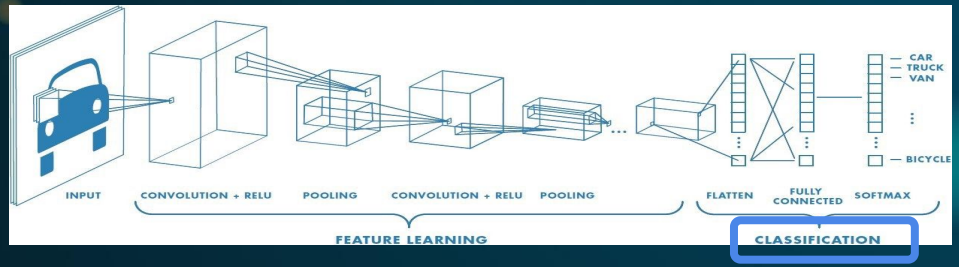
# Neural Network




The pooling layer aims to reduce the size of the picture. This helps to reduce the learning time.

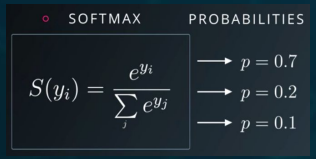


# Neural Network



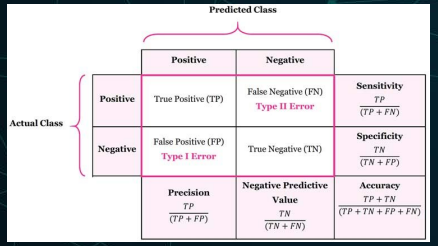

The learning process is done then the classification process starts. This is a basic neural network :

- The flatten layer transforms a 3D space into 1D. The picture is a flatten vector
- The fully connected layer uses to extract learnings from all previous layers
- The softmax function  to classify pictures



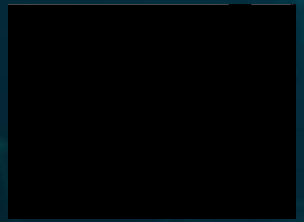
# Model Evaluation

Confusion Matrix




Minimise the loss

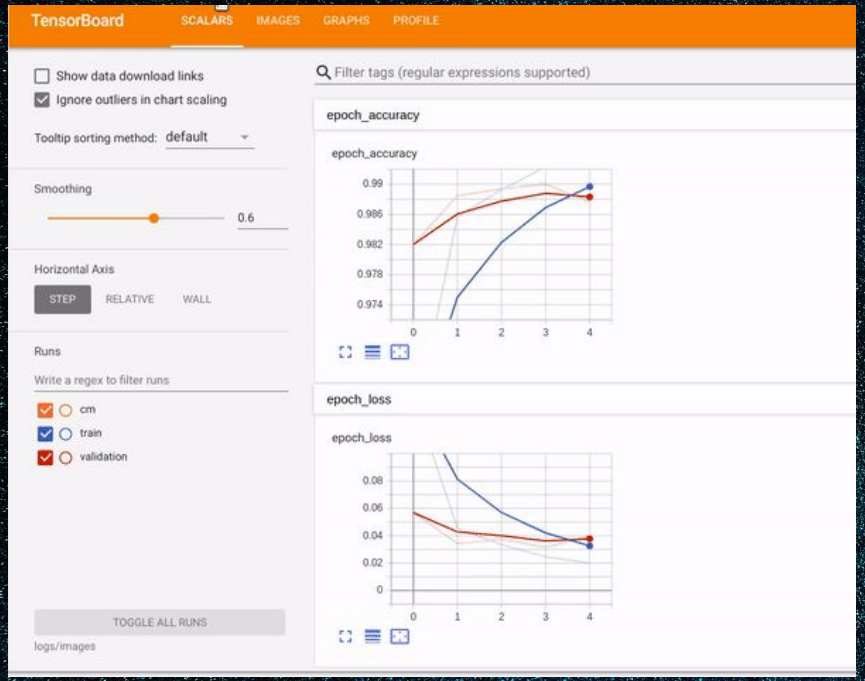
Learn from weights

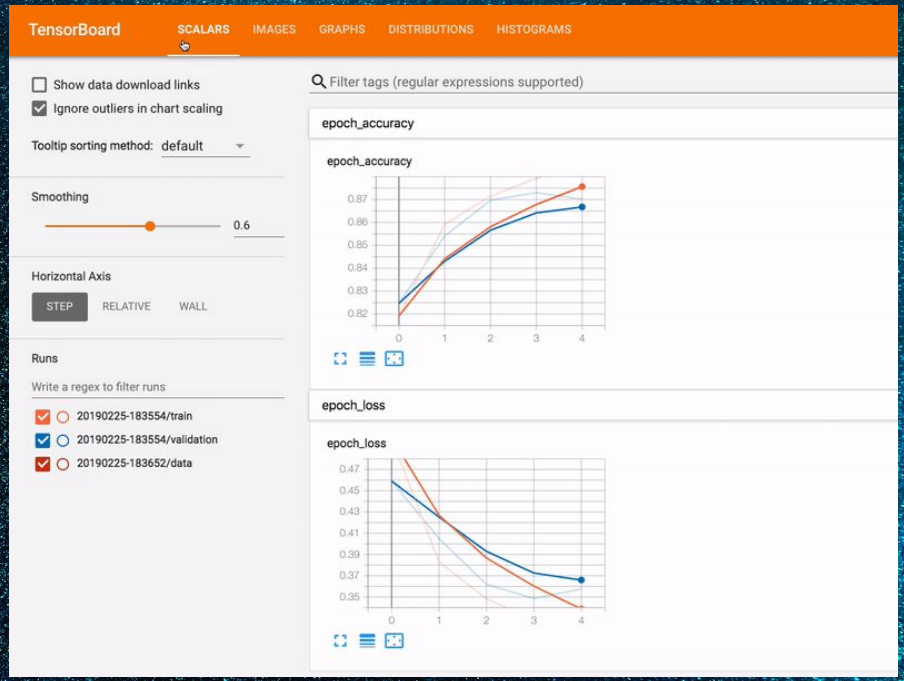


Improve the accuracy

Try it yourself

# Use Tensorboard



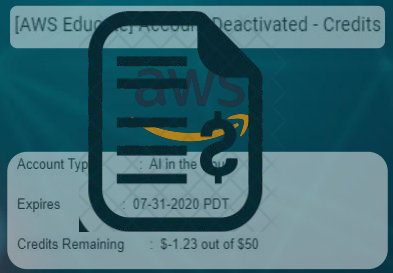
# Transfer Learning & Data Augmentation

# 02

State of the art models

# Why do we use transfer learning ?

Learning on large amount of data can cost a lot



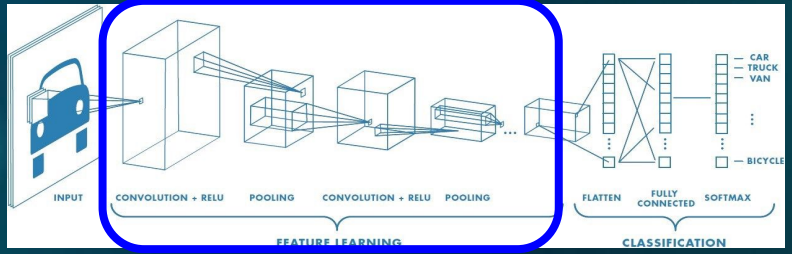

Learning on large amount of data takes times

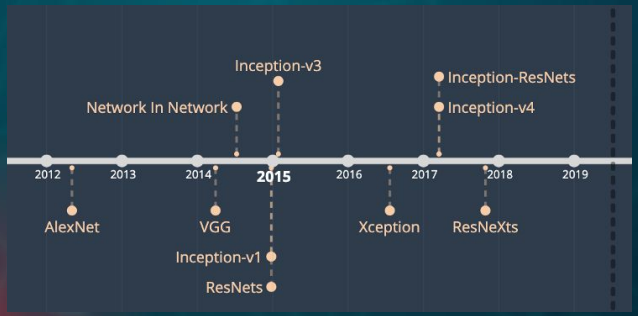
It is preferable when the model need to be trained regularly

“Situation where what has been learned in one setting is exploited to improve generalization in another setting.”

Deep Learning, Goodfellow

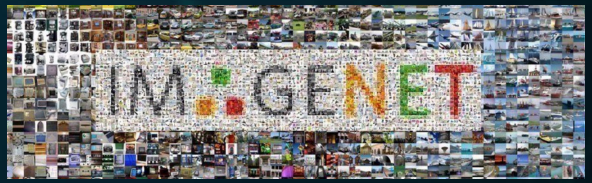
# Transfer learning




| Model | Size | Top-1 Accuracy | Top-5 Accuracy | Parameters | Depth |
|---|---|---|---|---|---|
| Xception | 88 MB | 0.790 | 0.945 | 22,910,480 | 126 |
| VGG16 | 528 MB | 0.713 | 0.901 | 138,357,544 | 23 |
| VGG19 | 549 MB | 0.713 | 0.900 | 143,667,240 | 26 |
| ResNet50 | 98 MB | 0.749 | 0.921 | 25,636,712 | - |
| ResNet101 | 171 MB | 0.764 | 0.928 | 44,707,176 | - |
| ResNet152 | 232 MB | 0.766 | 0.931 | 60,419,944 | - |
| ResNet50V2 | 98 MB | 0.760 | 0.930 | 25,613,800 | - |
| ResNet101V2 | 171 MB | 0.772 | 0.938 | 44,675,560 | - |
| ResNet152V2 | 232 MB | 0.780 | 0.942 | 60,380,648 | - |
| InceptionV3 | 92 MB | 0.779 | 0.937 | 23,851,784 | 159 |
| InceptionResNetV2 | 215 MB | 0.803 | 0.953 | 55,873,736 | 572 |
| MobileNet | 16 MB | 0.704 | 0.895 | 4,253,864 | 88 |
| MobileNetV2 | 14 MB | 0.713 | 0.901 | 3,538,984 | 88 |
| DenseNet121 | 33 MB | 0.750 | 0.923 | 8,062,504 | 121 |
| DenseNet169 | 57 MB | 0.762 | 0.932 | 14,307,880 | 169 |
| DenseNet201 | 80 MB | 0.773 | 0.936 | 20,242,984 | 201 |
| NASNetMobile | 23 MB | 0.744 | 0.919 | 5,326,716 | - |
| NASNetLarge | 343 MB | 0.825 | 0.960 | 88,949,818 | - |
| EfficientNetB0 | 29 MB | - | - | 5,330,571 | - |
| EfficientNetB1 | 31 MB | - | - | 7,856,239 | - |
| EfficientNetB2 | 36 MB | - | - | 9,177,569 | - |
| EfficientNetB3 | 48 MB | - | - | 12,320,535 | - |
| EfficientNetB4 | 75 MB | - | - | 19,466,823 | - |
| EfficientNetB5 | 118 MB | - | - | 30,562,527 | - |
| EfficientNetB6 | 166 MB | - | - | 43,265,143 | - |
| EfficientNetB7 | 256 MB | - | - | 66,658,687 | - |

K Keras



# Data Augmentation

**Make the most of few training examples !**
**& Reduce Overfitting**

Keras

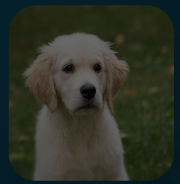
```
from keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(
        rotation_range=40,
        width_shift_range=0.2,
        height_shift_range=0.2,
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest')
```

**Image rotation**

**Image Zooming**

**Image Shifting**


**Image Brightness**



# Data Augmentation

**Make the most of few training examples !**
**& Reduce Overfitting**

Keras

```
from keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(
        rotation_range=40,
        width_shift_range=0.2,
        height_shift_range=0.2,
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest')
```

**Image Shear Intensity**

**Horizontal flip**

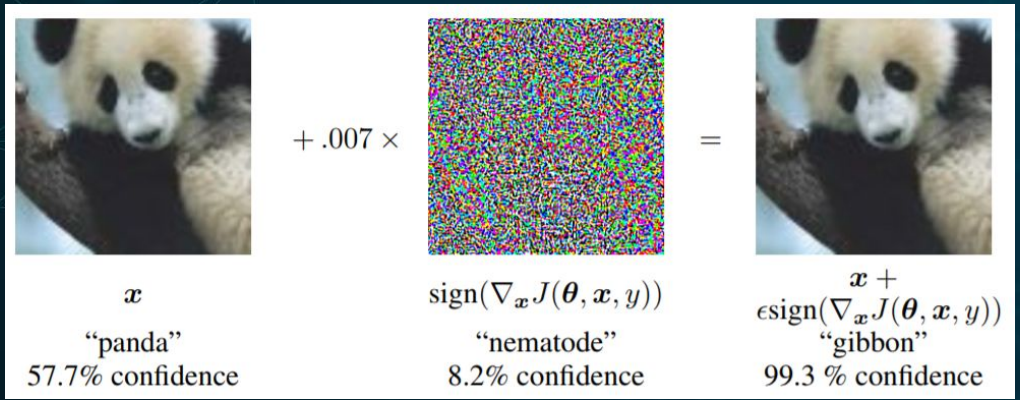
# Robustness 03

# Accuracy vs robustness


- The accuracy is not always the right metric
- Adding some well-chosen noise to a picture can completely

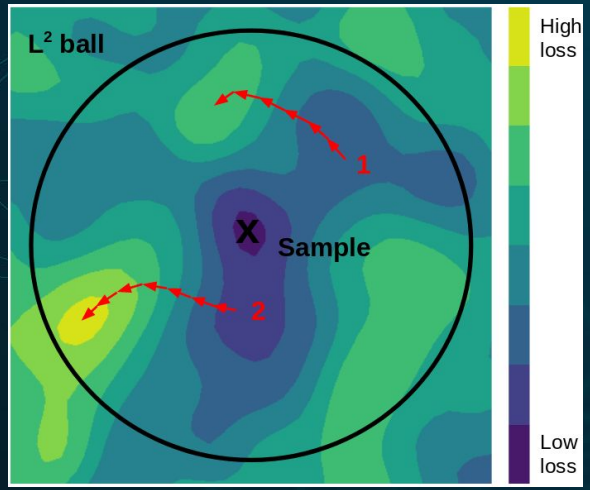
# Fast Gradient Sign Method (FGSM)

- Introduced by Goodfellow and al. Explaining and Harnessing Adversarial Examples (ICLR 2015)
- Idea: compute the sign of the gradient of the loss wrt to each pixel of the input image



# Projected Gradient Descent (PGD)

- Introduced in Kurakin et al. 2016
- Idea: Apply FGSM several times while ensuring we stay in an ball around the original image



# More complex attacks: Carlini & Wagner

- Introduced by Carlini and Wagner, Towards Evaluating the Robustness of Neural Networks, IEEE 2017
- Idea: For a given example $x \in X$ of the class $y \in Y$, the l2 Carlini & Wagner attack (C&W) aims at solving the following optimization problem :

$$\min_{x+\delta} c\|\delta\|_2 + g(x+\delta)$$

- where $g(x+\delta) \leq 0$ if $f(x+\delta) \neq y$

# The defense: How to make a model Robust

- A simple but yet effective way to defend against attacks is to add attacked images to the training set
- Introduced by Madry, Aleksander et al. , Towards deep learning models resistant to adversarial attacks (2017)

# Generative Models

# 04

Gans

# Generative models

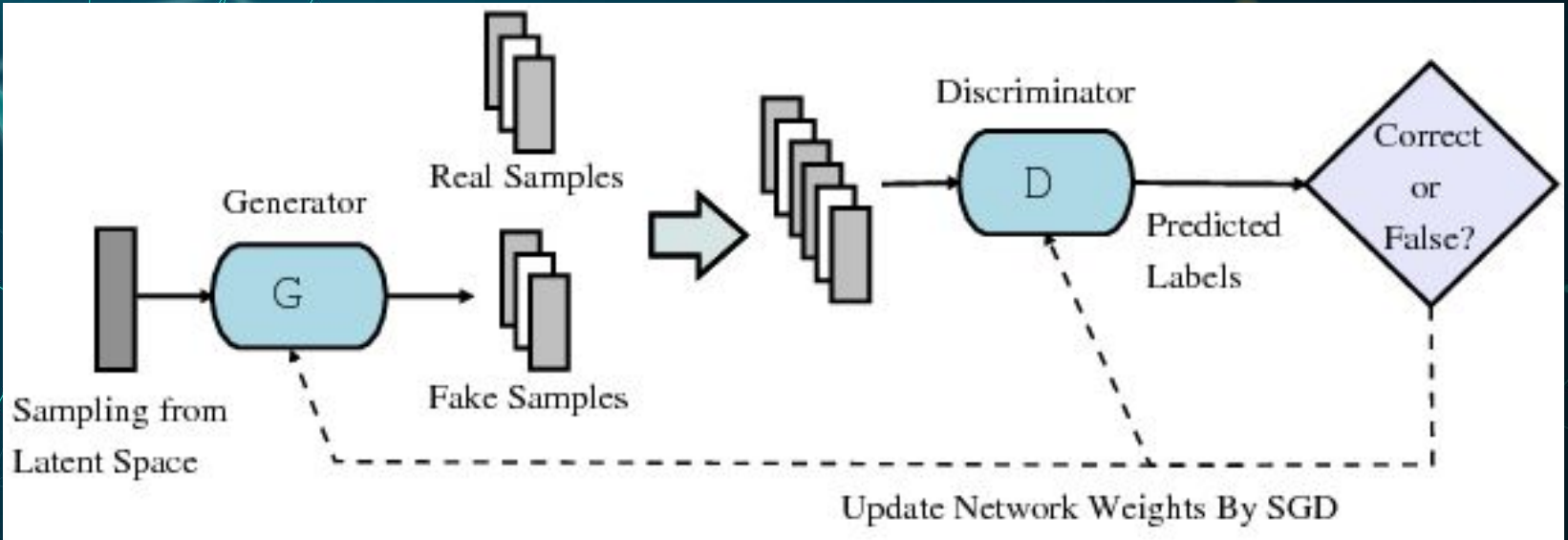
- Models can detect, classify, but also generate.

Image generated by StyleGAN, based on portrait analysis.

# What is a GAN?

- **Generative Adversarial Networks** Introduced by Goodfellow and al. (2014)
- **New approach**: using Game Theory for training an image synthesis model
- Solving a minimax problem between a **Generator** and a **Discriminator**

# Architecture of a GAN




source: https://www.semanticscholar.org/paper/Generative-Adversarial-Networks-for-Unsupervised-Plakias-Boutalis/627d4f69b76bb3fc88283de9e8e9f7ee6c598ea7

# The optimisation problem - classical GANs

The **Discriminator** is trained to maximize:

$$L = E[\log P(S = real \mid X_{real})] + E[\log P(S = fake \mid X_{fake})]$$

⇨ Log-likelihood that the Discriminator assigns the generated image to the correct source

The **Generator** is trained to minimize:

$$E[\log P(S = fake \mid X_{fake})]$$

⇨ Second term in the first loss. Minimizing the log-likelihood that the Discriminator detects the fake images.

# THANKS!

Do you have any questions?

olivier.randavel@gmail.com

hippolyte.mayard@dauphine.eu