Cours initiation à l'IA

Insta promo 2021

Sommaire

1. Introduction
2. Agents Intelligents
3. Algorithmes de recherche
4. Réseaux de neurones
5. TensorFlow 2.0

Introduction

Systèmes intelligents :

-systèmes qui se *comportent* comme un *humain*

-systèmes qui *pensent* comme un *humain*

-systèmes qui *pensent rationnellement*

-systèmes qui possèdent des *comportement rationnelles*

Brève histoire

-1943 : McCulloch et Pitts création du premier modèle de neurone artificielle.

-1950: Test de Turing

-1973: Crise de l'IA, arrêt des financement sur le sujet (rapport Lighthill)

-1986: retour des réseaux de neurones.

Domaines de l’IA

-Traitement du langage naturel

-Vision artificiel

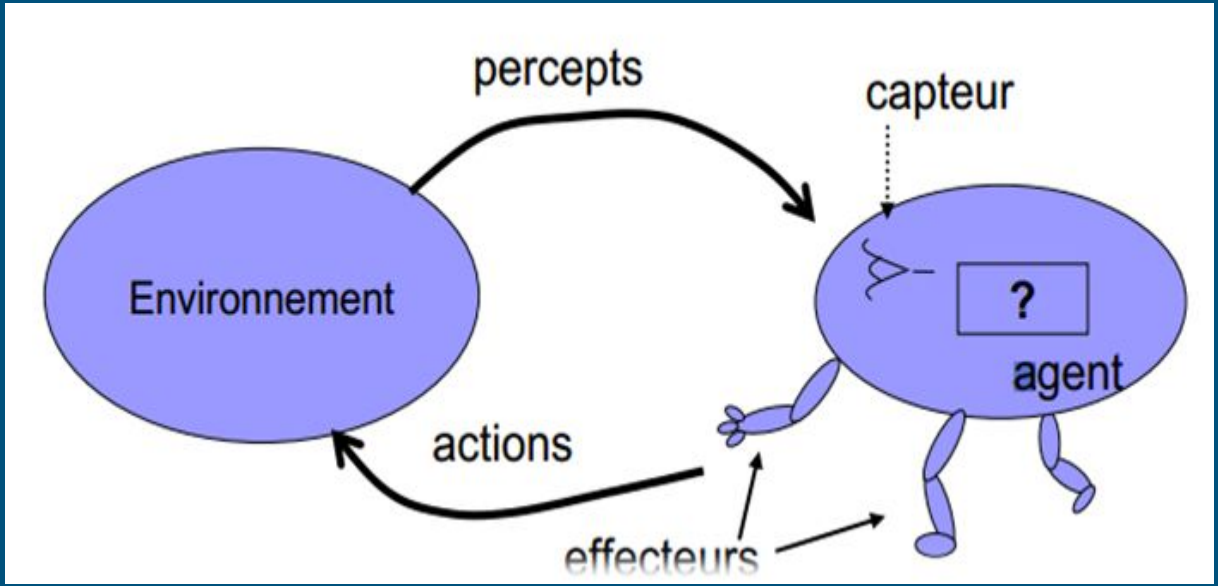
-Robotique

-Bourse, e-commerce, etc...

Agent Artificiel

processus agent f:

$f = p^* \longrightarrow A$



Agent Artificiel

fonction SKELETON-AGENT(precept) returns action

static memory,the agent's memory of the world

memory <= Update-Memory(memory,percept)

action <= Choose-Best-Action(memory)

memory<= Update-Memory(memory,action)

return action

Agent Artificiel

Un agent est dit *rationnel* si il effectue des actions correctes.

→ permis par une mesure de performance.

Un agent est dit *autonome* si il est capable de d'adapter son comportement en fonction de son expérience (il maximise la mesure de performance).

Modèle PEAS

Performance

Environnement

Actuators

Sensors

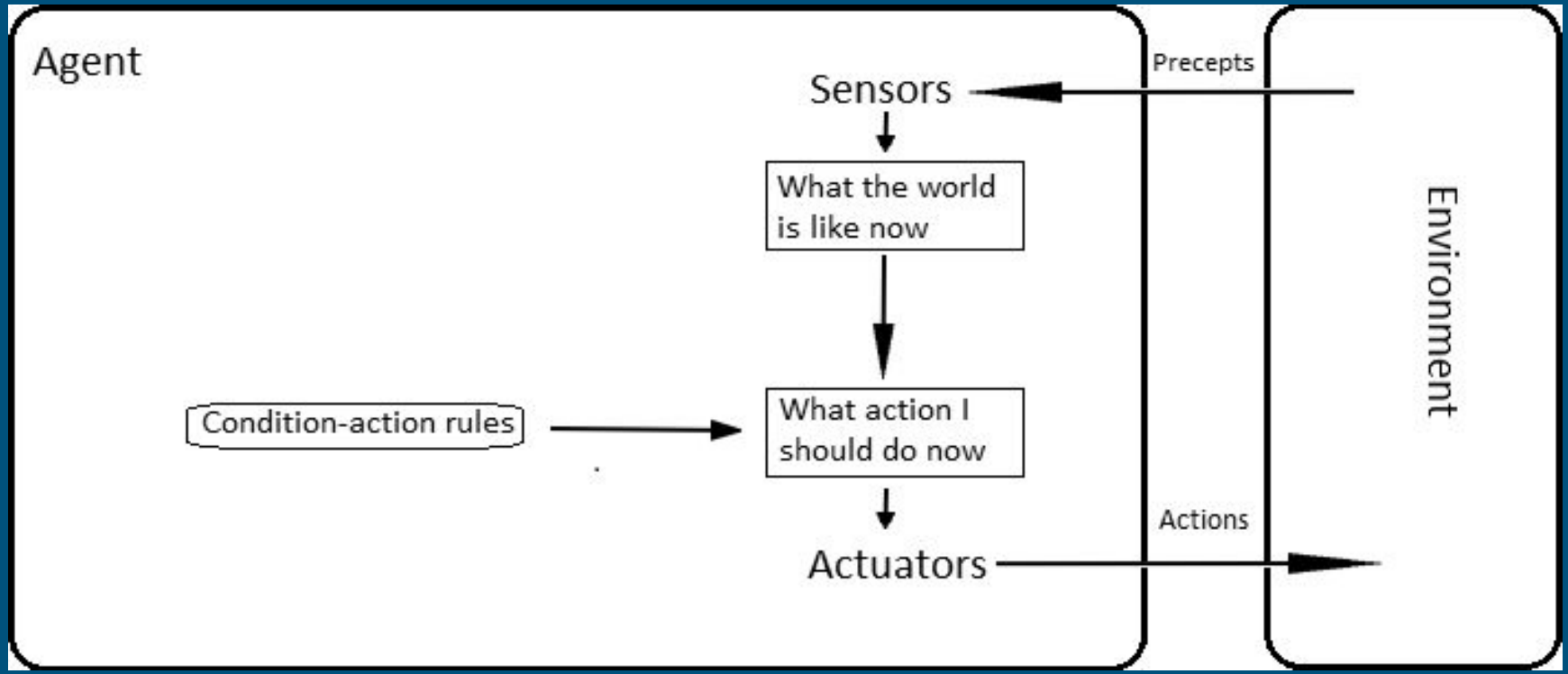
Caractéristiques d'environnement

- Complètement observable (vs partiellement observable)
- Déterministe (vs stochastique)
- Épisodique (vs séquentiel)
- Statique (vs dynamique)
- Agent unique (vs multi-agents)

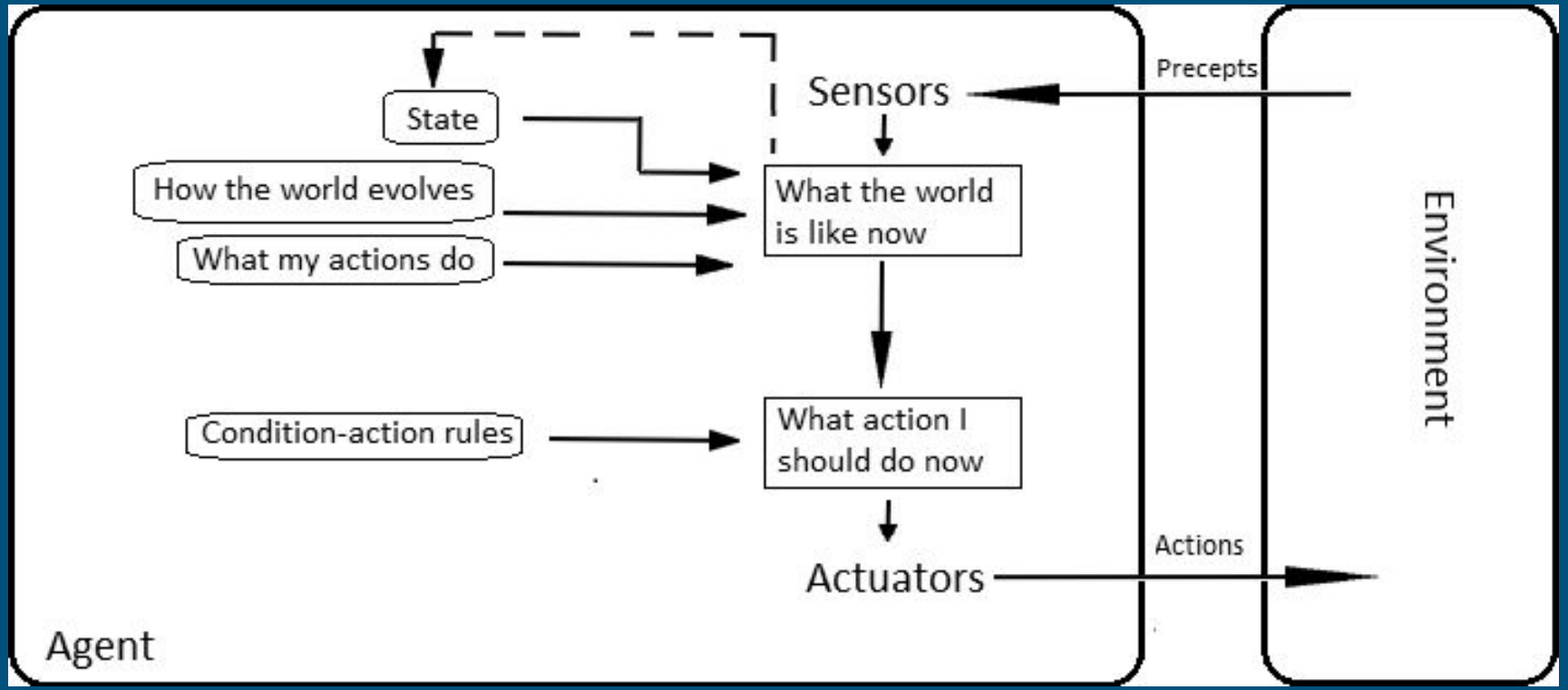
Agent Artificiel

- Simple reflex agent
- Model-based reflex agent
- Goal-based agent
- Utility-based agent

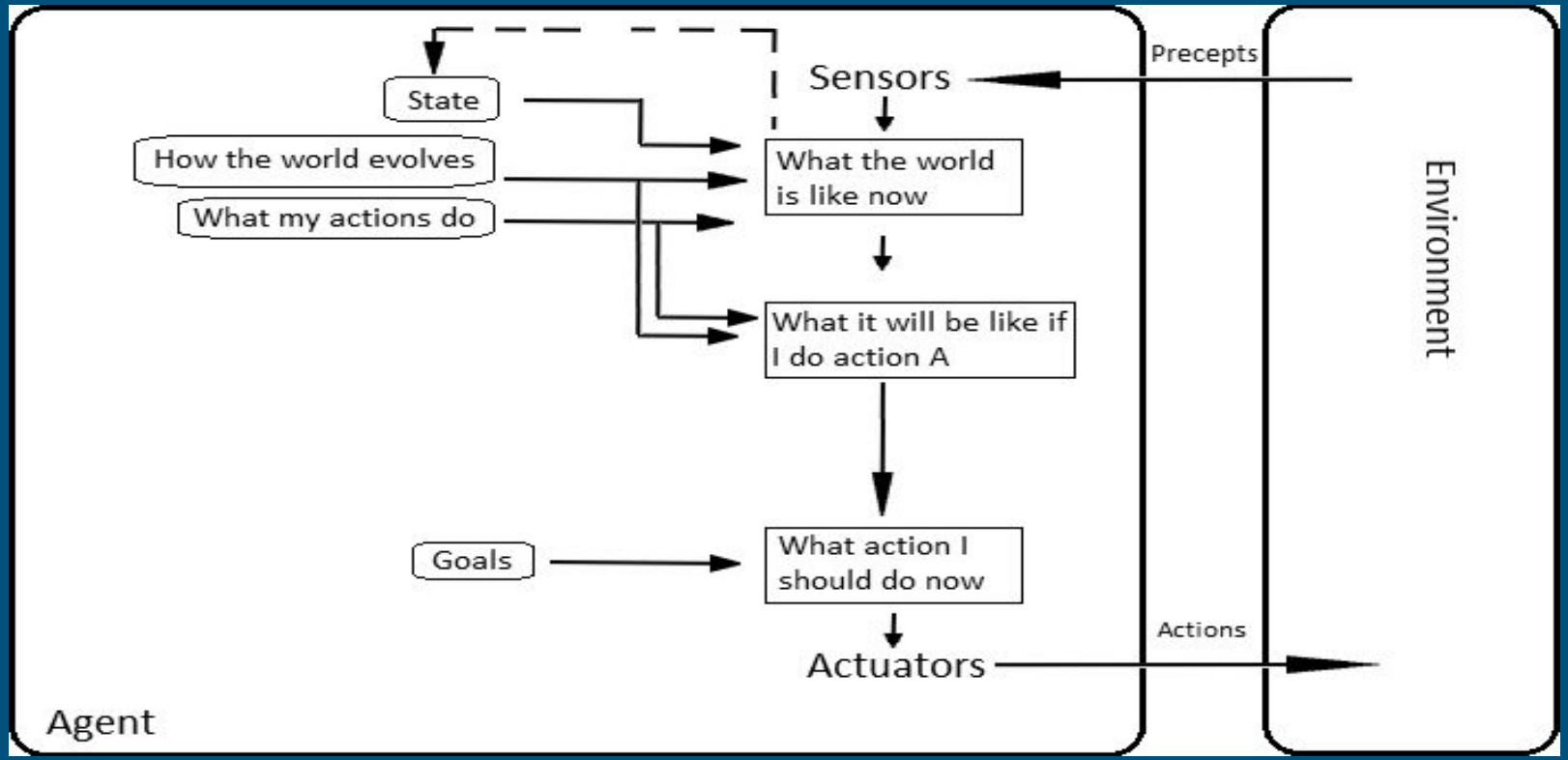
Simple reflex agent



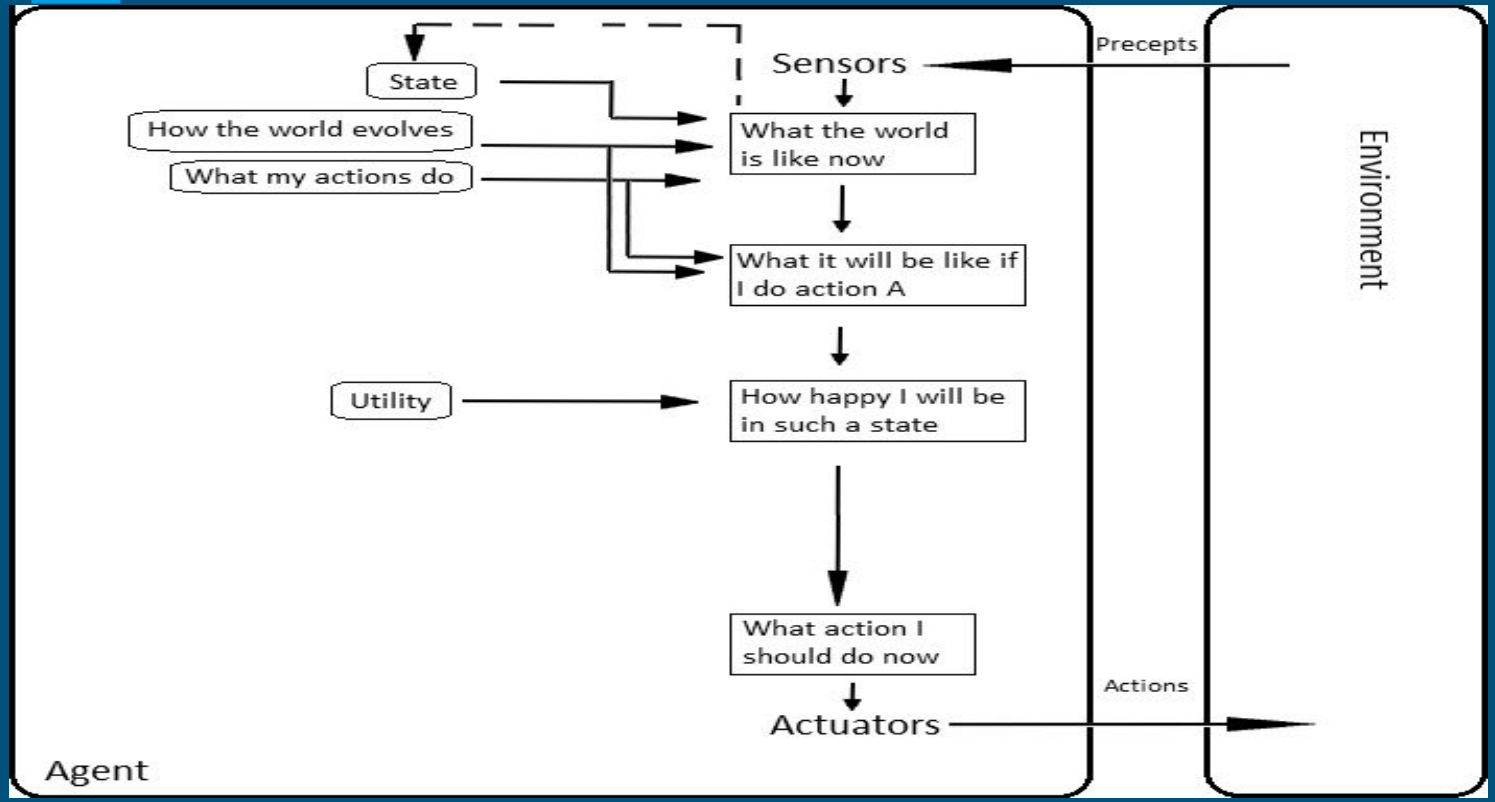
Model based reflex agent



Goal based agent



Utility based agent



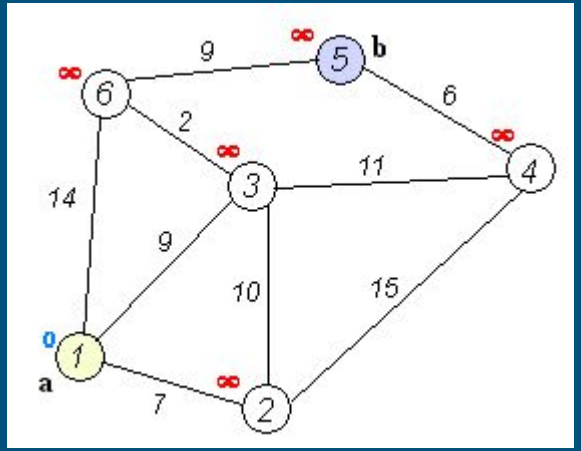
Algorithme de recherche

Problème ???

- Modéliser la situation actuel
- Énumérer les solutions possibles
- Évaluer la valeur des solutions
- Retenir la meuilleur options possible satisfaisant le but

Algorithme de recherche

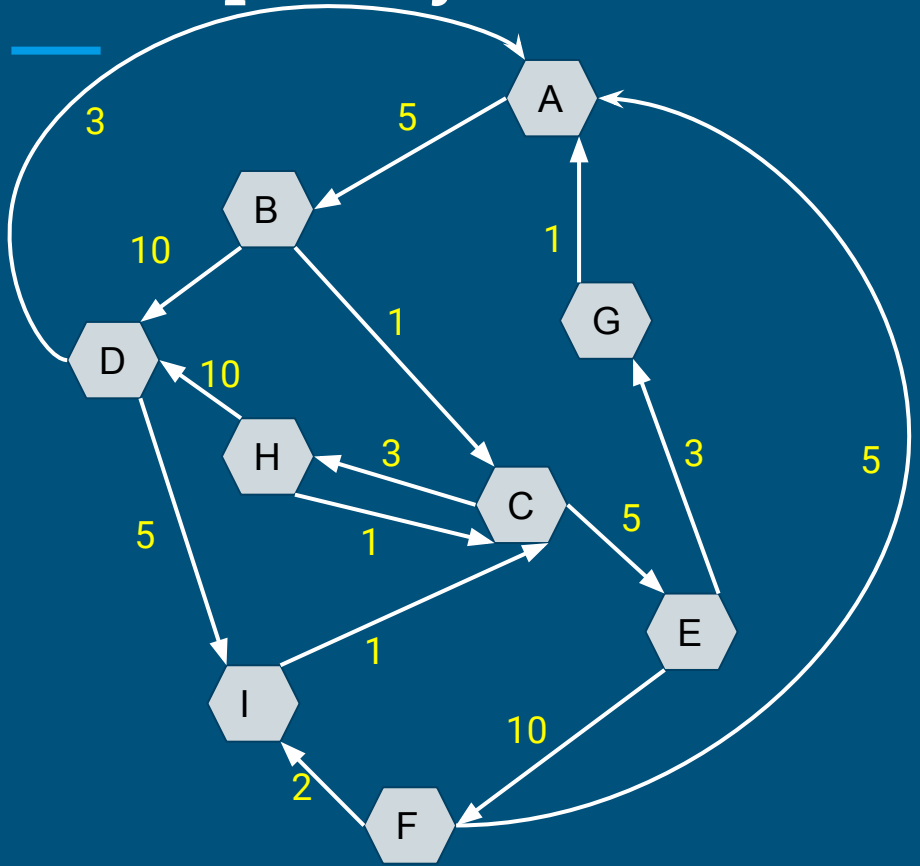
- Chaque noeud correspond à un état de l'environnement
- Chaque chemin représente alors une suite d'actions prise par l'agent
- Pour résoudre le problème on choisit le chemin le plus performant



Algorithme de recherche

- Dijkstra
- Recherche en profondeur, largeur
- best first search
- gready best first search
- A*

Exemple Dijkstra



A: Attente événements
B: Réception commandes
C: Compréhension
D: Incompréhension
E: Exécute action
F: Erreur
G: Confirmation action
H: Question à l'utilisateur + attente reponse.
I: Apprentissage par l'utilisateur

Exemple Dijkstra (Dérouler de A à G)

	A	B	C	D	E	F	G	H	I
Départ	0	-	-	-	-	-	-	-	-
A(0)		5	-	-	-	-	-	-	-
B(5)			1	10	-	-	-	-	-
C(1)					5	-	-	3	-
H(3)			1	10		-	-		-
D(10)	3		1			-	-		5
I(5)			1			-	-		
E(5)						10	3		

Exemple Dijkstra

Entrées : $G = (S, A)$ un graphe avec une pondération positive $poids$ des arcs, s_{deb} un sommet de S

$P := \emptyset$
$d[a] := +\infty$ pour chaque sommet a
$d[s_{deb}] = 0$
Tant qu'il existe un sommet hors de P
    Choisir un sommet a hors de P de plus petite distance $d[a]$
    Mettre a dans P
    Pour chaque sommet b hors de P voisin de a
        Si d[b] > d[a] + poids(a, b)
            $d[b] = d[a] + poids(a, b)$
            prédécesseur[b] := a
    Fin Pour
Fin Tant Que

Algorithme A*

A* est équivalent à Dijkstra avec une heuristique.

Une heuristique est une function h(n) d’estimation du coût restant entre un noeud n d’un graphe et le but.

Algorithme A*

Principe général : évaluation du coût total d'un sommet

Coût total (**F**) = Coût depuis la source (**G**) +
Coût vers la destination (**H**)

G : Coût depuis la source

- Algorithmes classiques (Ford, Bellman, Dijkstra)
- $G_i = \min G_j + C_{ij}$ / i prédécesseur de j
  C_{ij} coût de l'arc (i,j)

H : Coût vers la destination

- Difficile puisque le reste du chemin (vers la destination) est encore inconnu.

Algorithme A*

Initialisation

Sommet source (S)

Sommet destination (D)

Liste des sommets à explorer (E) : sommet source S

Liste des sommets visités (V) : vide

Tant que (la liste E est non vide) et (D n'est pas dans E) Faire

+ Récupérer le sommet X de coût total F minimum.

+ Ajouter X à la liste V

+ Ajouter les successeurs de X (non déjà visités) à la liste E en évaluant leur coût total F et en identifiant leur prédécesseur.

+ Si (un successeur est déjà présent dans E) et

(nouveau coût est inférieur à l'ancien) Alors

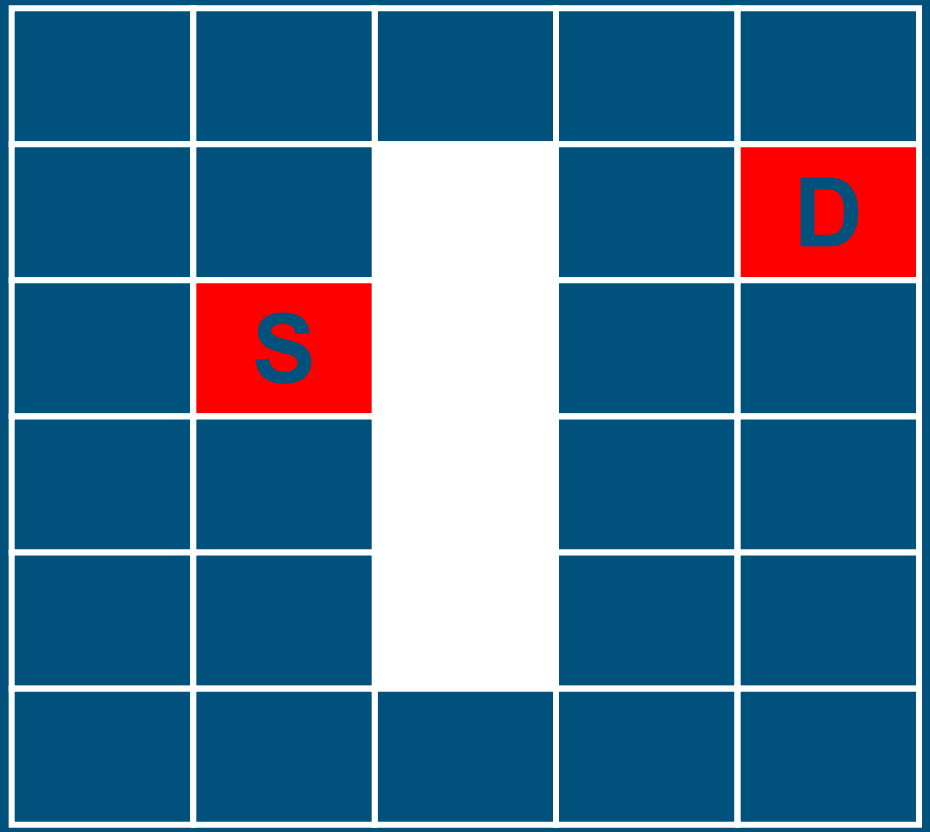
Changer son coût total

Changer son prédécesseur

FinSi

FinFaire

Exemple A*

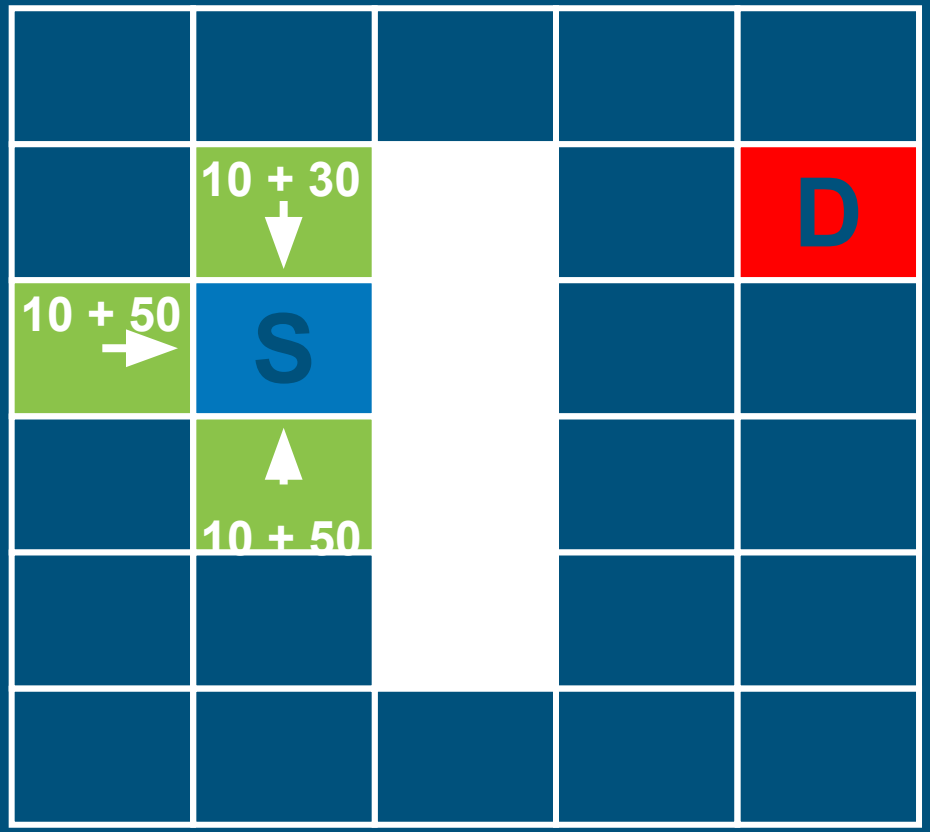


S Sommet source

D Sommet destination

Obstacle

Exemple A*



Sommet déjà visité

Sommet à explorer

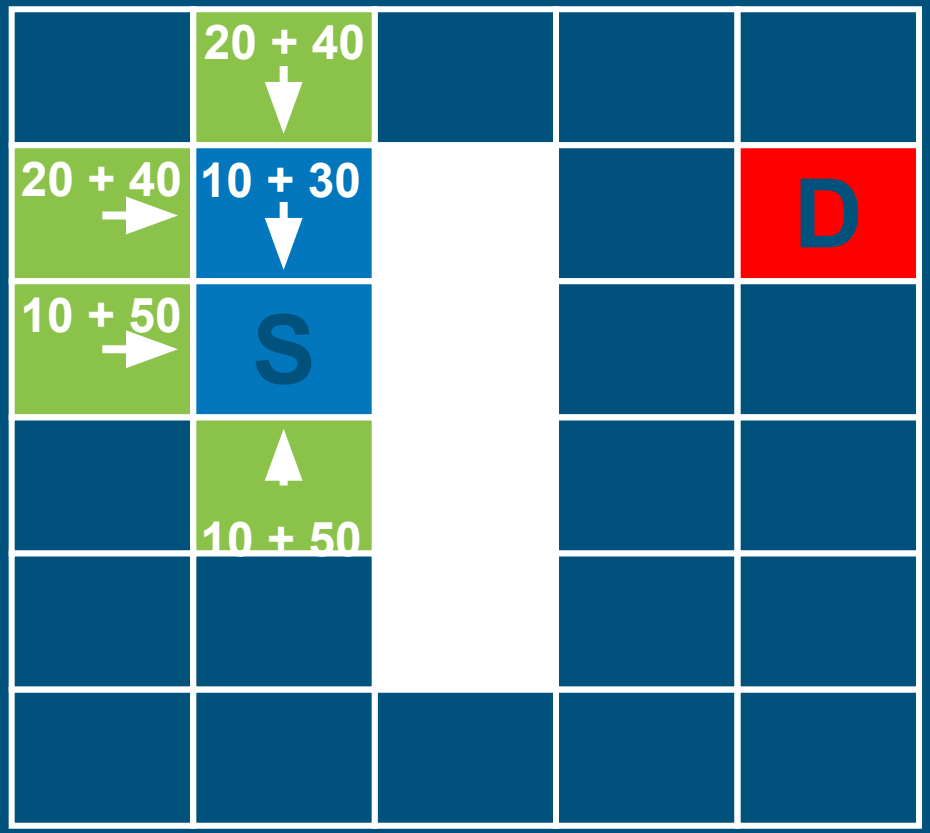
Coût depuis la source

Coût vers la destination

G + H

Référence au prédécesseur

Exemple A*



Sommet déjà visité

Sommet à explorer

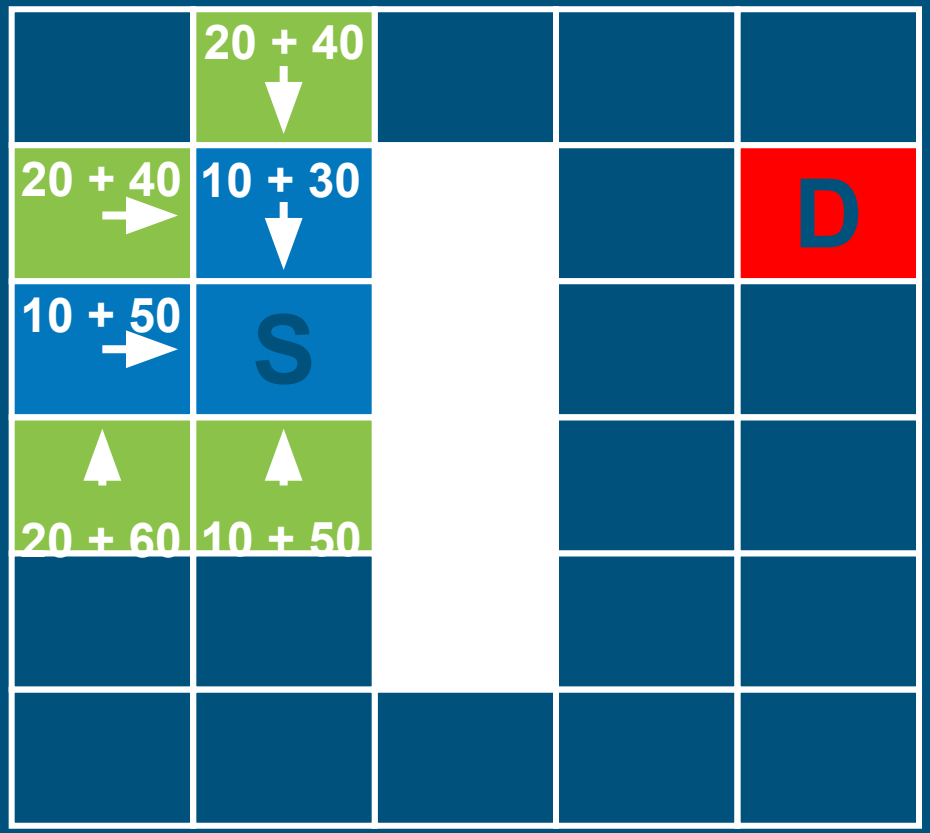
Coût depuis la source

Coût vers la destination

G + H

Référence au prédécesseur

Exemple A*



Sommet déjà visité

Sommet à explorer

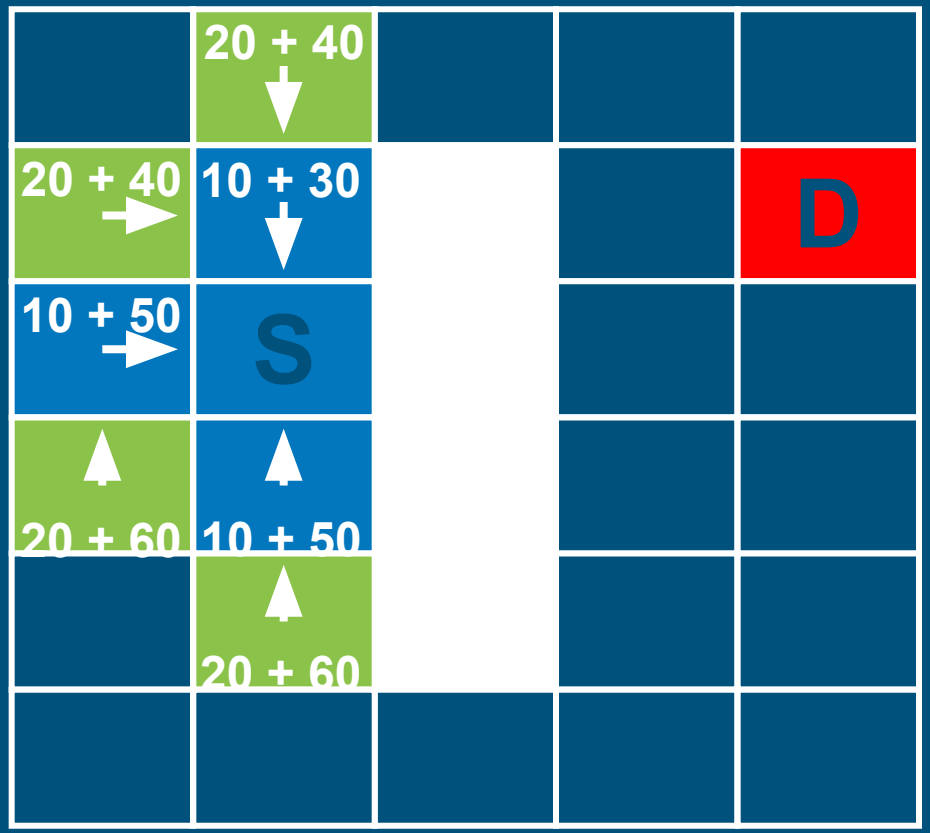
Coût depuis la source

Coût vers la destination

G + H

Référence au prédécesseur

Exemple A*



Sommet déjà visité

Sommet à explorer

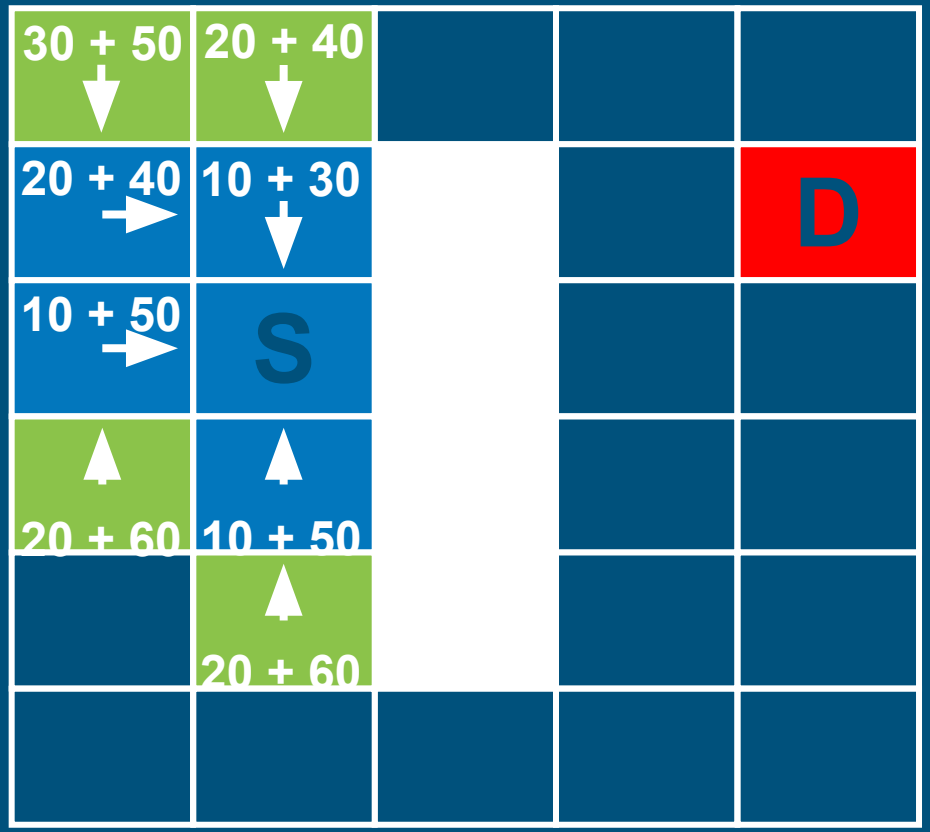
Coût depuis la source

Coût vers la destination

G + H

Référence au prédécesseur

Exemple A*



Sommet déjà visité

Sommet à explorer

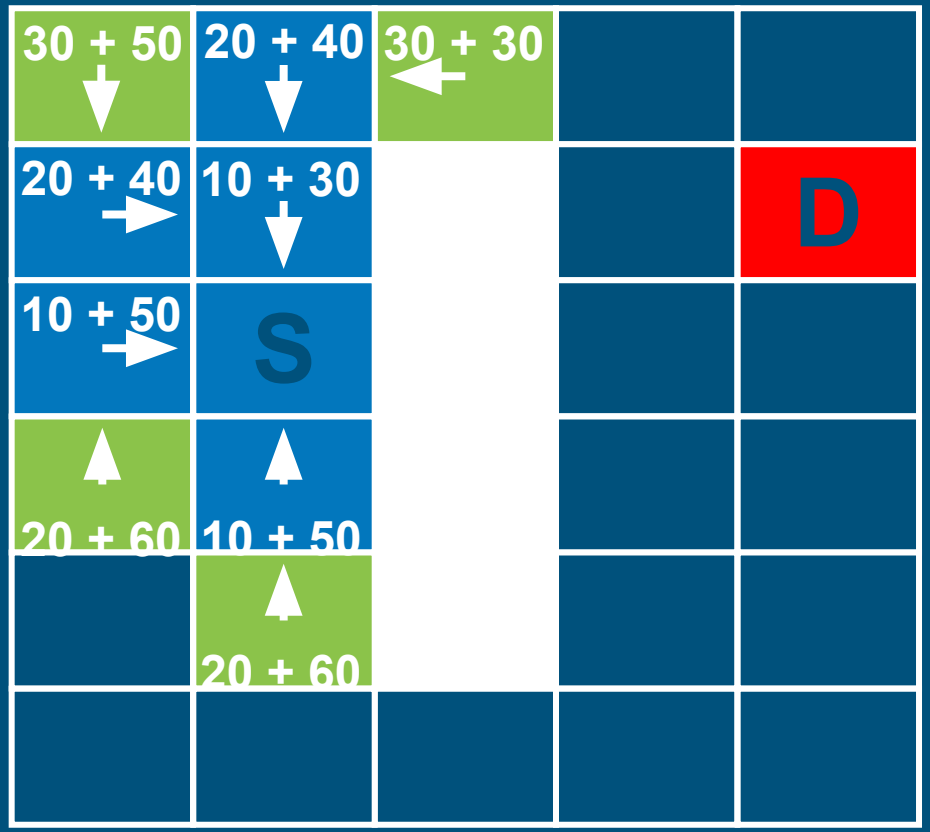
Coût depuis la source

Coût vers la destination

G + H

Référence au prédécesseur

Exemple A*



Sommet déjà visité

Sommet à explorer

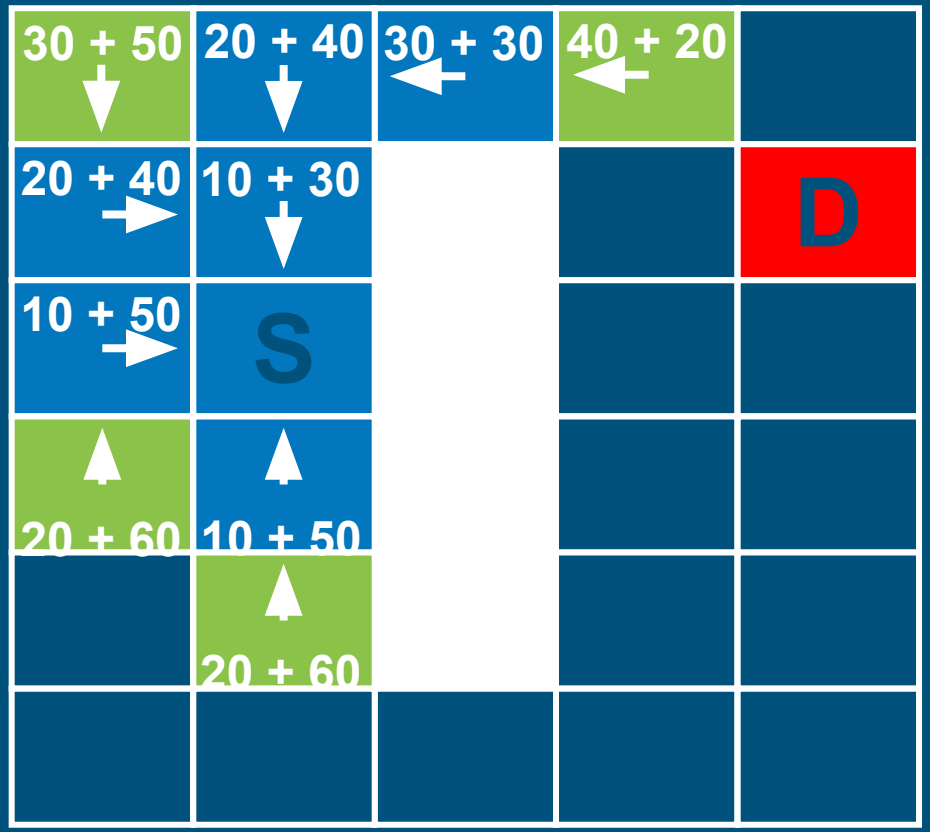
Coût depuis la source

Coût vers la destination

G + H

Référence au prédécesseur

Exemple A*



Sommet déjà visité

Sommet à explorer

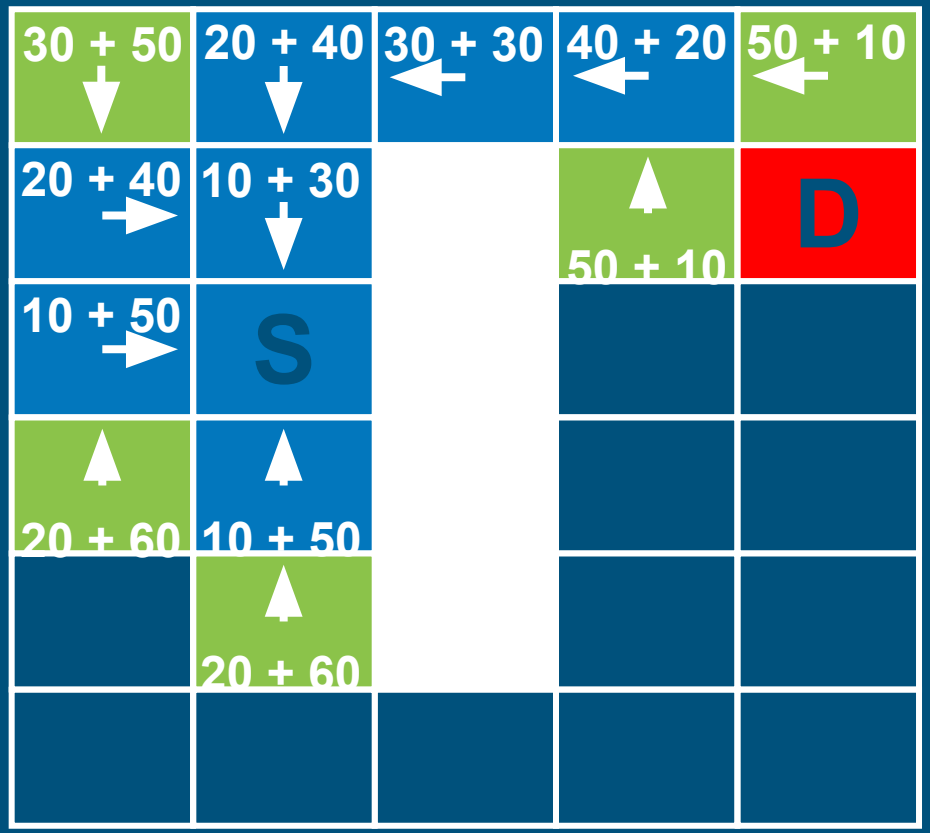
Coût depuis la source

Coût vers la destination

G + H

Référence au prédécesseur

Exemple A*



Sommet déjà visité

Sommet à explorer

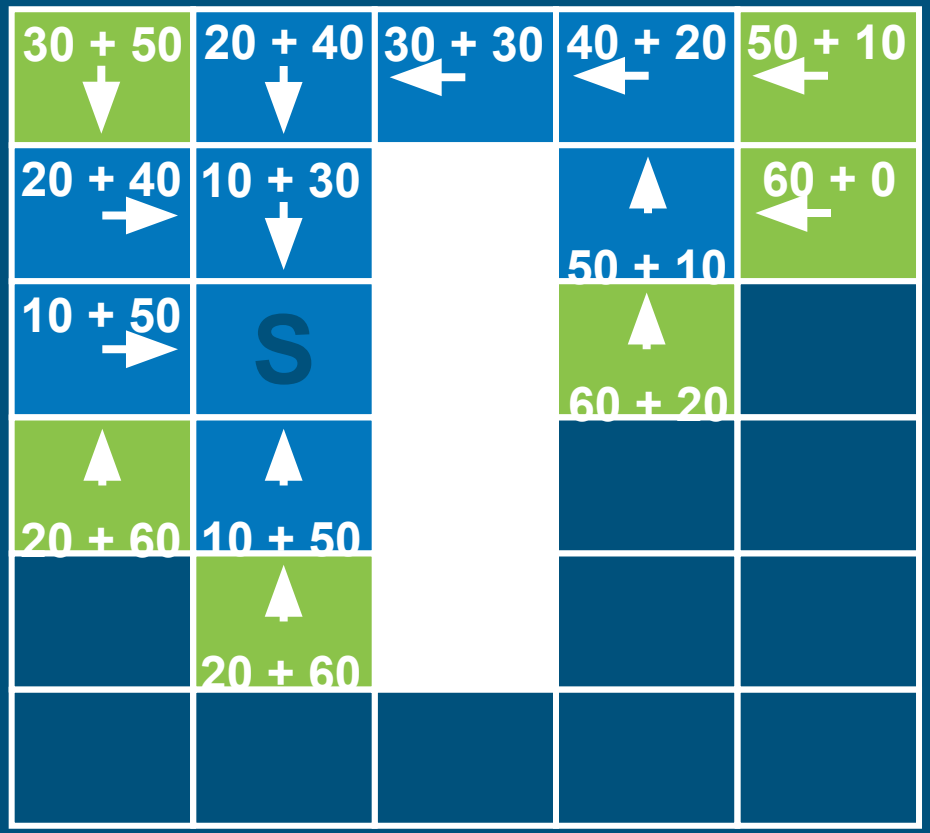
Coût depuis la source

Coût vers la destination

G + H

Référence au prédécesseur

Exemple A*



Sommet déjà visité

Sommet à explorer

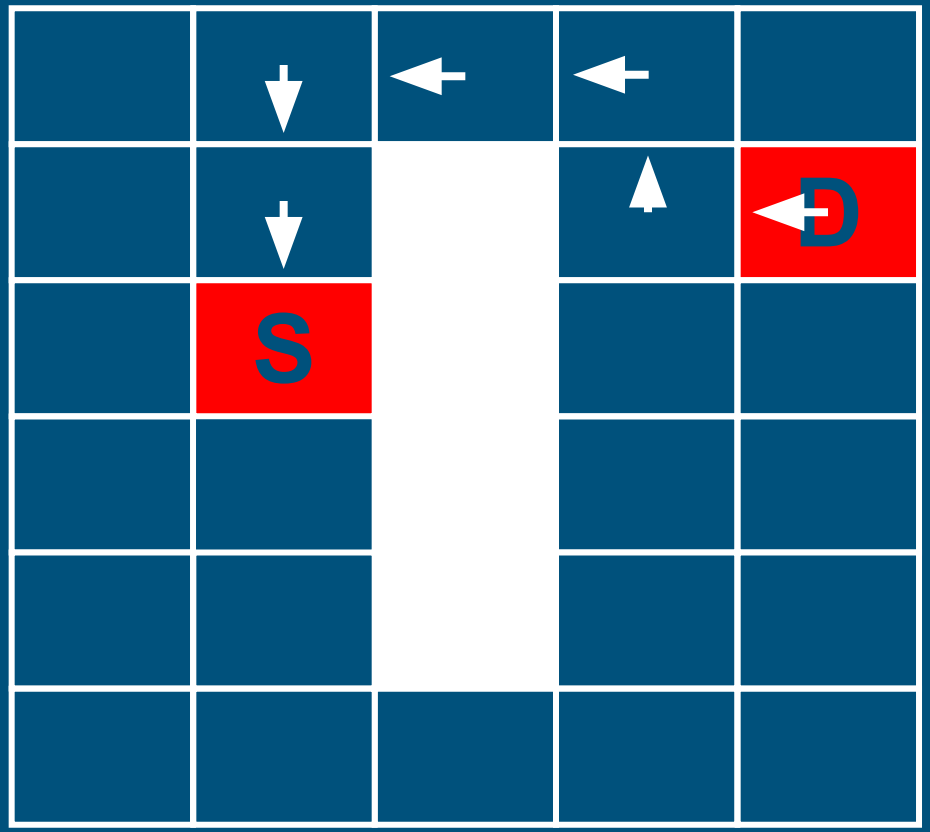
Coût depuis la source

Coût vers la destination

G + H

Référence au prédécesseur

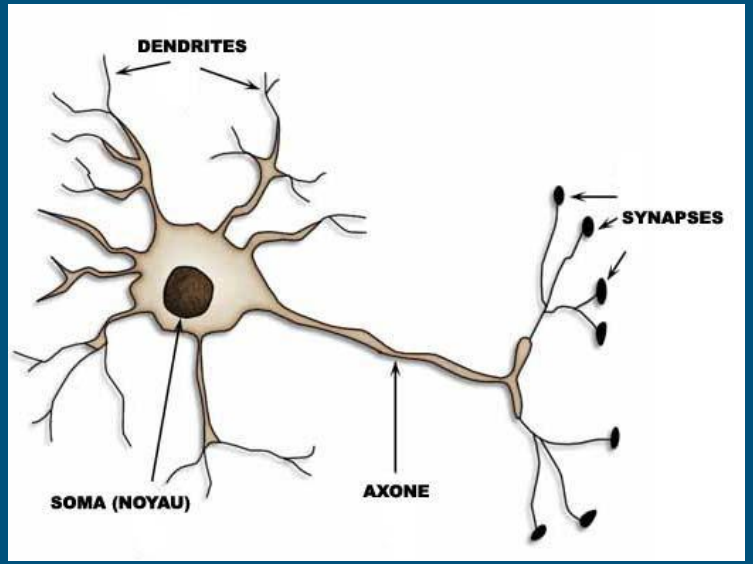
Exemple A*



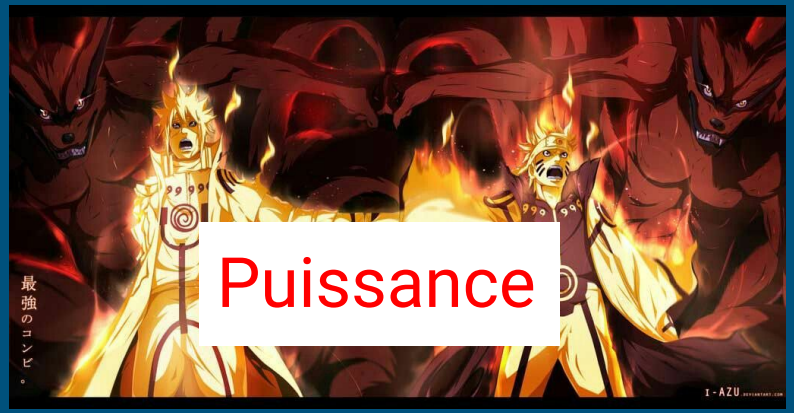
Réseaux de neurones

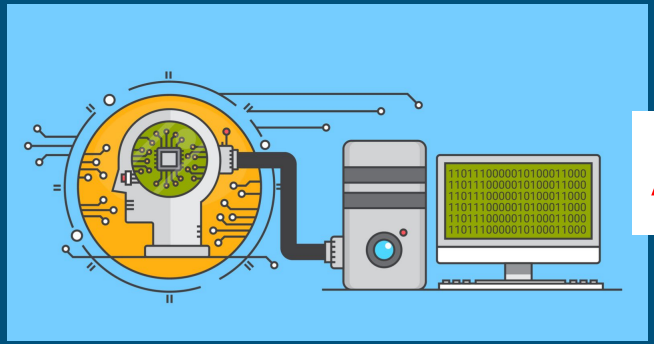
L’ intelligence artificiel basé sur le modèle du cerveau humain.

- analyse logique des tâches basée sur la cognition humaine.
- le cerveau produit la pensée, connexion de réseaux de neurones pour rendre créatif une ia



Réseaux de neurones





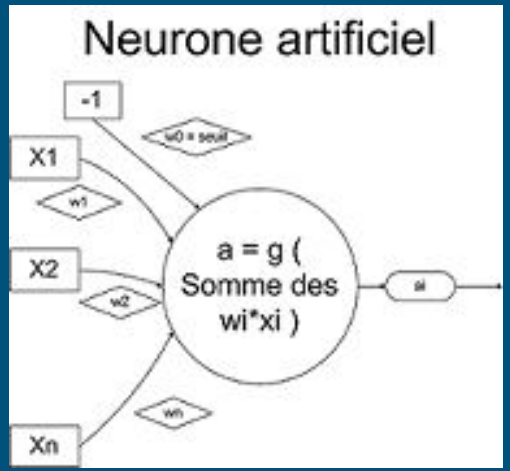
Algorithms

Un neurone

$X_{i\ (1<=i<k)}$ les k informations parvenant au neurone

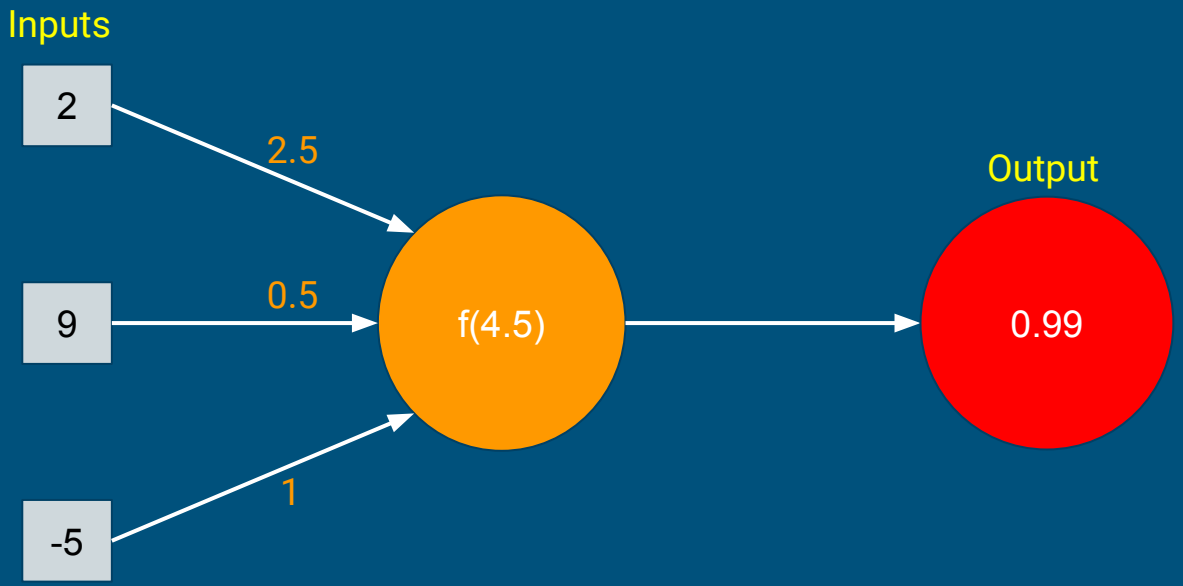
w_i le “poid”, coef lié à l’information X_i

w_0 le coefficient de biais lié à l’information X_0=-1



$$in = \sum_{i=0}^{k} w_i \times x_i = (\sum_{i=1}^{k} w_i \times x_i) - w_0$$

Neurone : exemple



Fonctions d’activations

Sigmoid

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

Leaky ReLU

$$\max(0.1x, x)$$

tanh

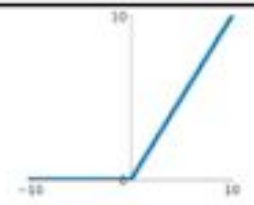
$$\tanh(x)$$

Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

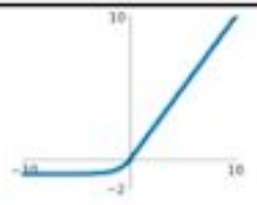
ReLU

$$\max(0, x)$$

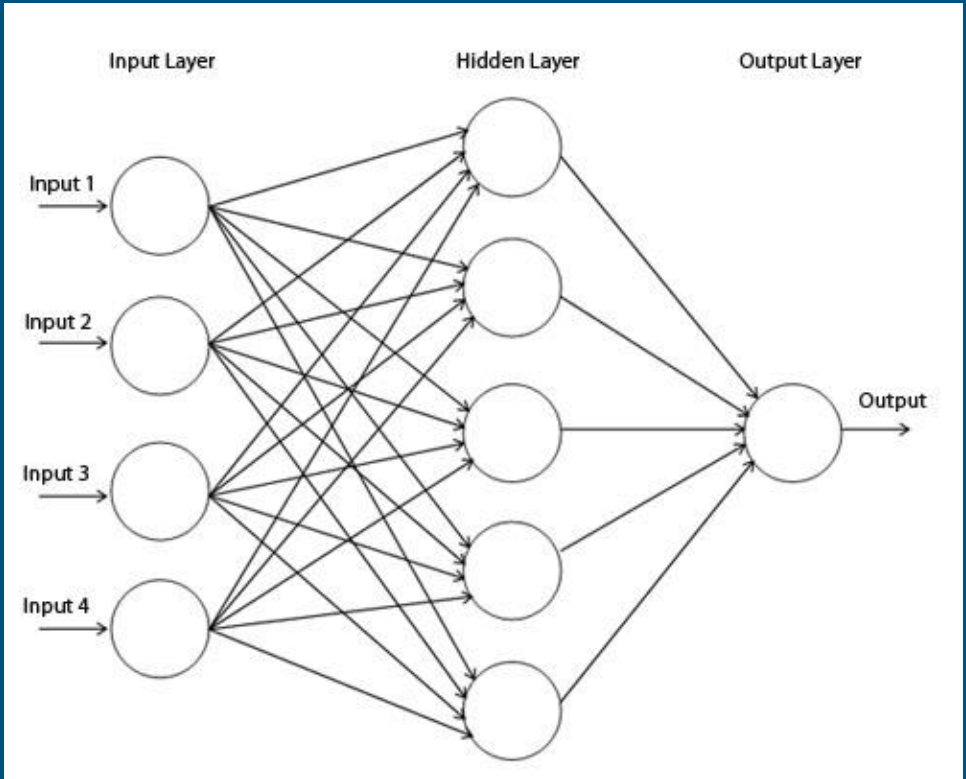


ELU

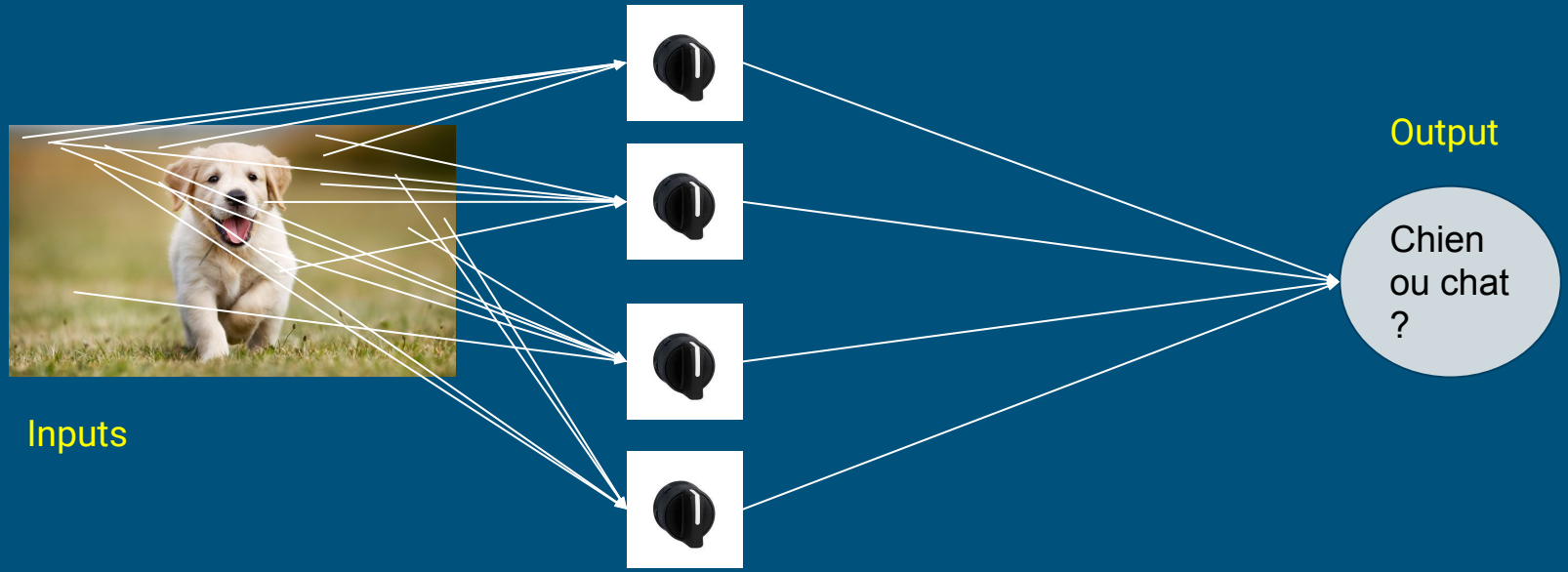
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Perceptron


Input Layer
Hidden Layer
Output Layer
Input 1
Input 2
Input 3
Input 4
Output

Exemple perceptron



Algorithme d'apprentissage par correction d'erreur

Entrée : un échantillon S de $R^n \times \{0,1\}$ ou $\{0,1\}^n \times \{0,1\}$

Initialisation aléatoire des poids w_i pour i entre 0 et n
Répéter
Prendre un exemple $(x^{®},c)$ dans S
Calculer la sortie o du perceptron pour l'entrée $x^{®}$
- - Mise à jour des poids - -
Pour i de 0 à n
$w_i \neg w_i + (c\text{-}o)x_i$
finpour
finRépéter

Sortie : Un perceptron P défini par $(w_0, w_1, ..., w_n)$

L'algorithme d'apprentissage peut être décrit succinctement de la manière suivante. On initialise les poids du perceptron à des valeurs quelconques. A chaque fois que l'on présente un nouvel exemple, on ajuste les poids selon que le perceptron l'a correctement classé ou non. L'algorithme s'arrête lorsque tous les exemples ont été présentés sans modification d'aucun poids.

Algorithme d'apprentissage par correction d'erreur

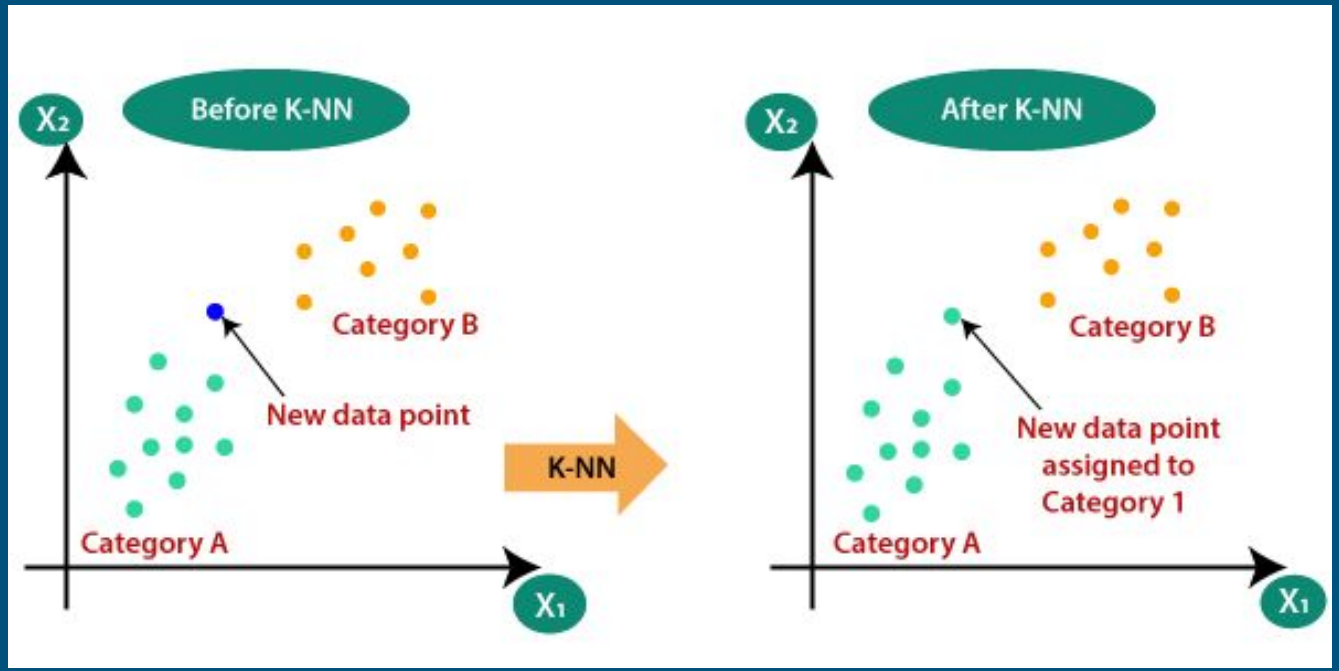
- Algorithme par descente du gradient

  Définition d’une fonction d’erreur, qu’on minimise avec la méthode de la descente du gradient.

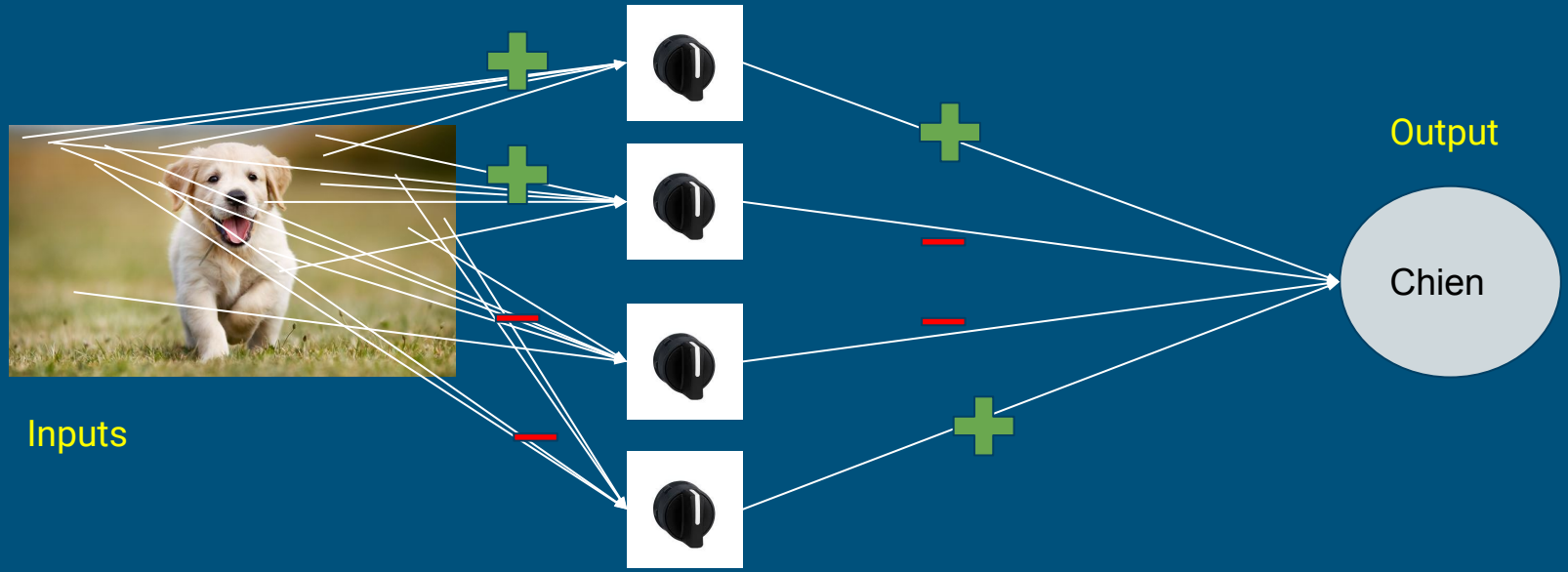
- Algorithme de Widrow et Hoff

  Modifie les poids à chaque présentation d'exemple à l’aide d’un coefficient d’apprentissage alpha.

Algorithme K Nearst Neighbors (K-NN)



Exemple perceptron



Exemple

x	-1	0	1	2	3	4
y	-3	-1	1	3	5	7

$$y = 2x - 1$$

```
from tensorflow import keras
import numpy as np

model = keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss="mean_squared_error")

xs= np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys= np.array([-3.0,-1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)

model.fit(xs, ys, epochs=500)

print(model.predict([10.0]))
```

Exemple Fashion MNIST

```
# TensorFlow and tf.keras
import tensorflow as tf
import datetime

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt

#Importer le jeu de données Fashion MNIST
fashion_mnist = tf.keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

Exemple Fashion MNIST

```
train_images = train_images / 255.0
test_images = test_images / 255.0

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```

Exemple Fashion MNIST

```
#Construisez le modèle
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10)
])
#Compilez le modèle
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)
```

Exemple Fashion MNIST

```
#Former le modèle
model.fit(train_images, train_labels, epochs=10,
          validation_data=(test_images, test_labels),
          callbacks=[tensorboard_callback])

#Évaluer la précision
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
print('\nTest accuracy:', test_acc)

probability_model = tf.keras.Sequential([model, tf.keras.layers.Softmax()])
predictions = probability_model.predict(test_images)
```

Exemple Fashion MNIST

```
def plot_image(i, predictions_array, true_label, img):
  true_label, img = true_label[i], img[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])
  plt.imshow(img, cmap=plt.cm.binary)
  predicted_label = np.argmax(predictions_array)
  if predicted_label == true_label:
    color = 'blue'
  else:
    color = 'red'

  plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                       100*np.max(predictions_array),
                                      class_names[true_label]),
                                       color=color)
```

Exemple Fashion MNIST

```
def plot_value_array(i, predictions_array, true_label):
  true_label = true_label[i]
  plt.grid(False)
  plt.xticks(range(10))
  plt.yticks([])
  thisplot = plt.bar(range(10), predictions_array, color="#777777")
  plt.ylim([0, 1])
  predicted_label = np.argmax(predictions_array)

  thisplot[predicted_label].set_color('red')
  thisplot[true_label].set_color('blue')
```

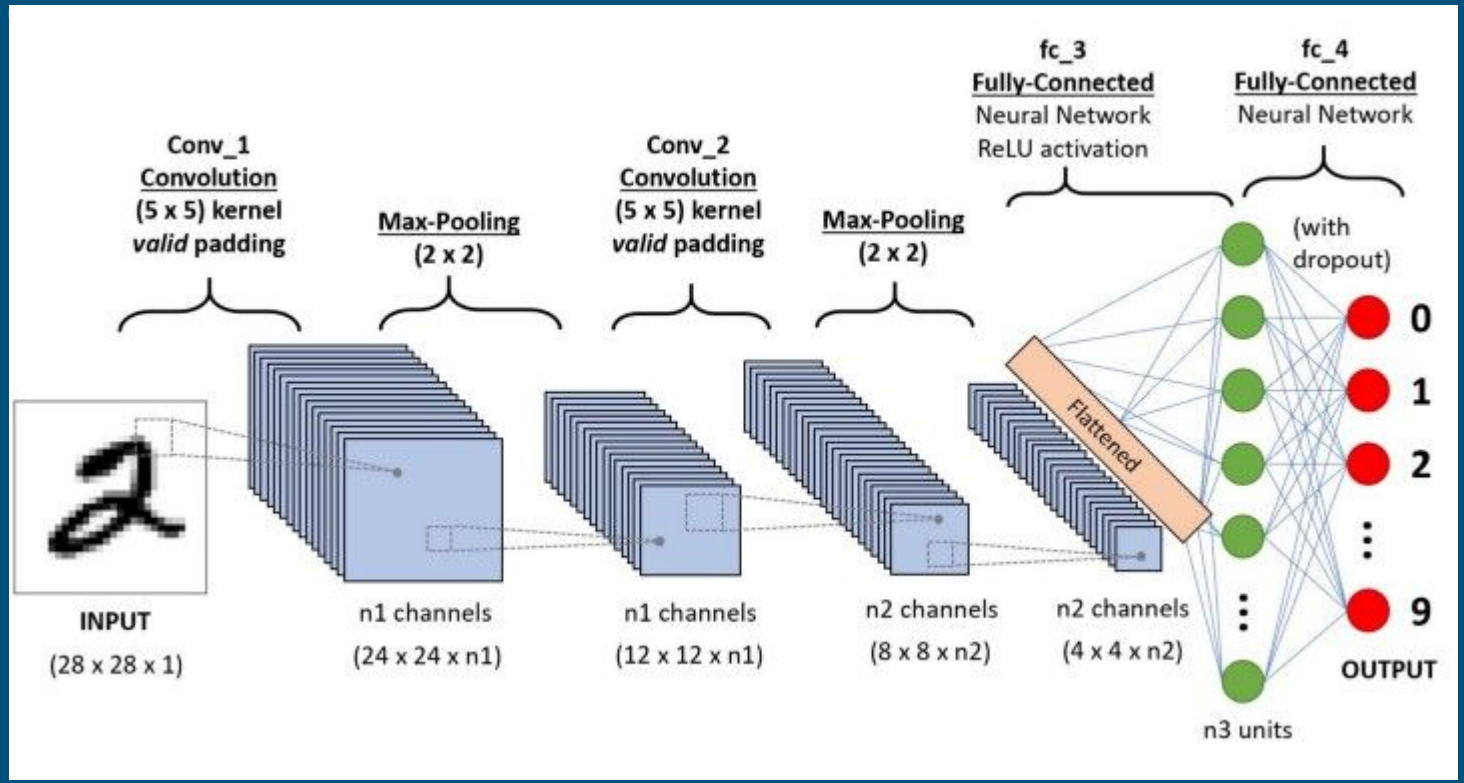
Exemple Fashion MNIST

```
plt.figure()
indexPlot=1
for i in range(10,22):
    plt.subplot(4,6,indexPlot)
    plot_image(i, predictions[i], test_labels, test_images)
    plt.subplot(4,6,indexPlot+1)
    plot_value_array(i, predictions[i],  test_labels)
    indexPlot +=2
plt.show()
```

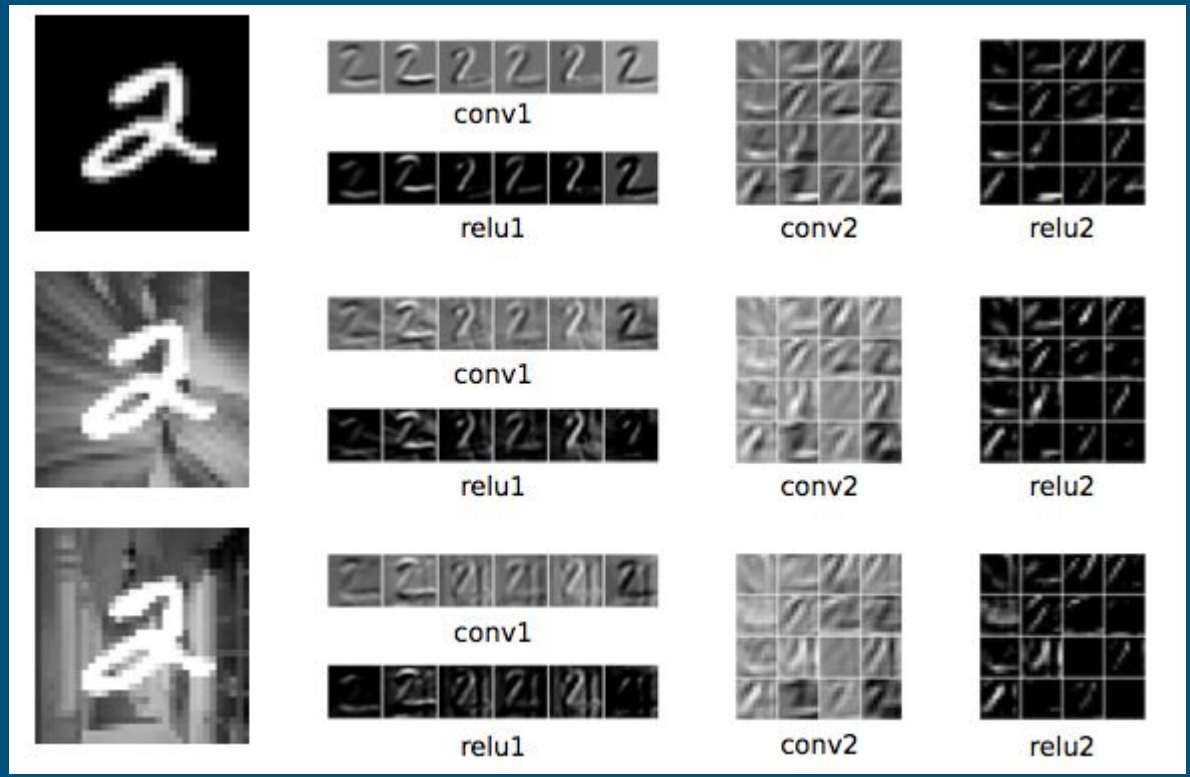
Exemple avec tensorFlow

- https://playground.tensorflow.org
- https://colab.research.google.com/github/keras-team/keras-io/blob/master/examples/vision/ipynb/image_classification_from_scratch.ipynb
- https://github.com/ageitgey/face_recognition

Réseaux à convolution

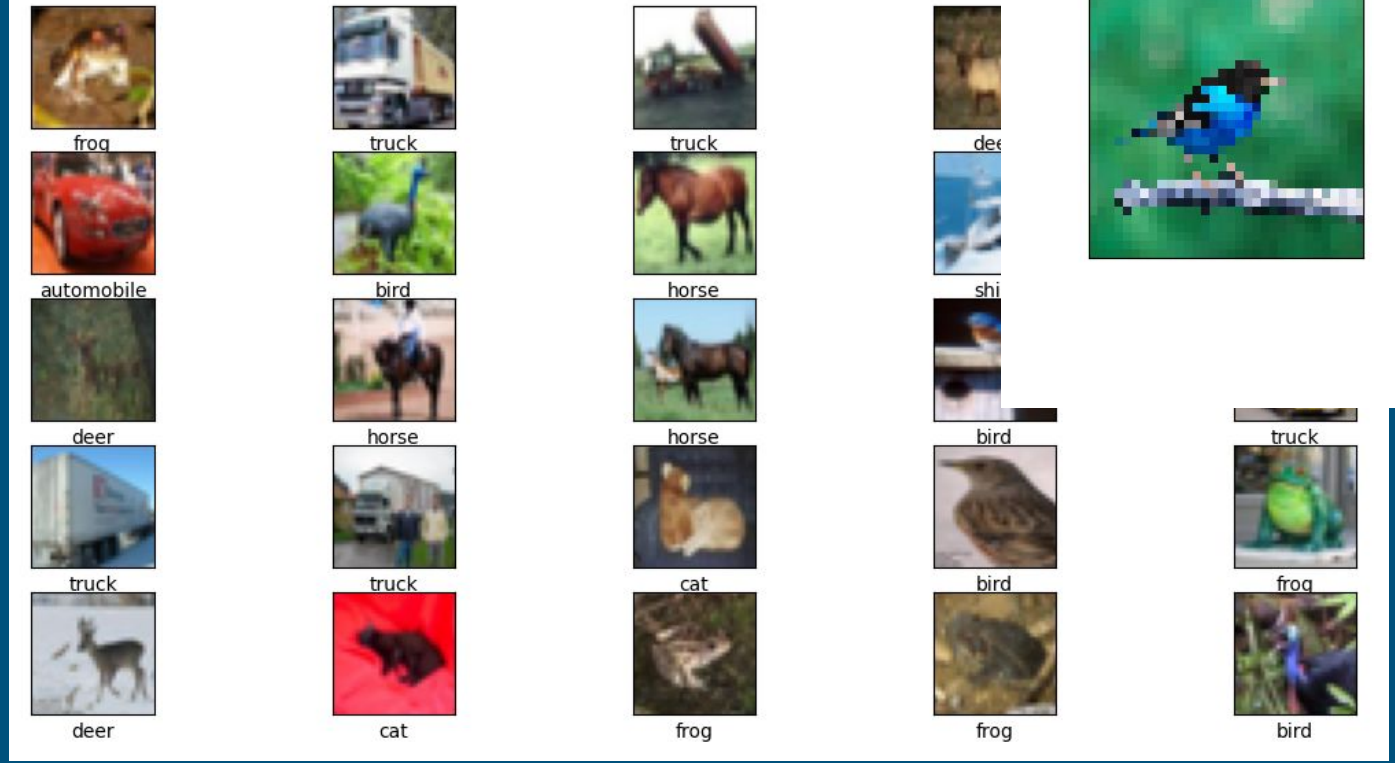


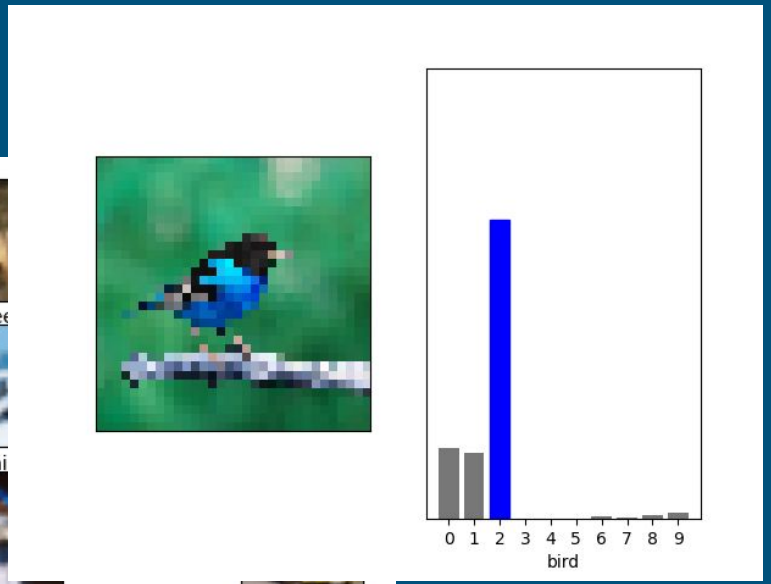
Réseaux à convolution



Réseaux à convolution

https://www.tensorflow.org/tutorials/images/cnn





Exercice du Snake

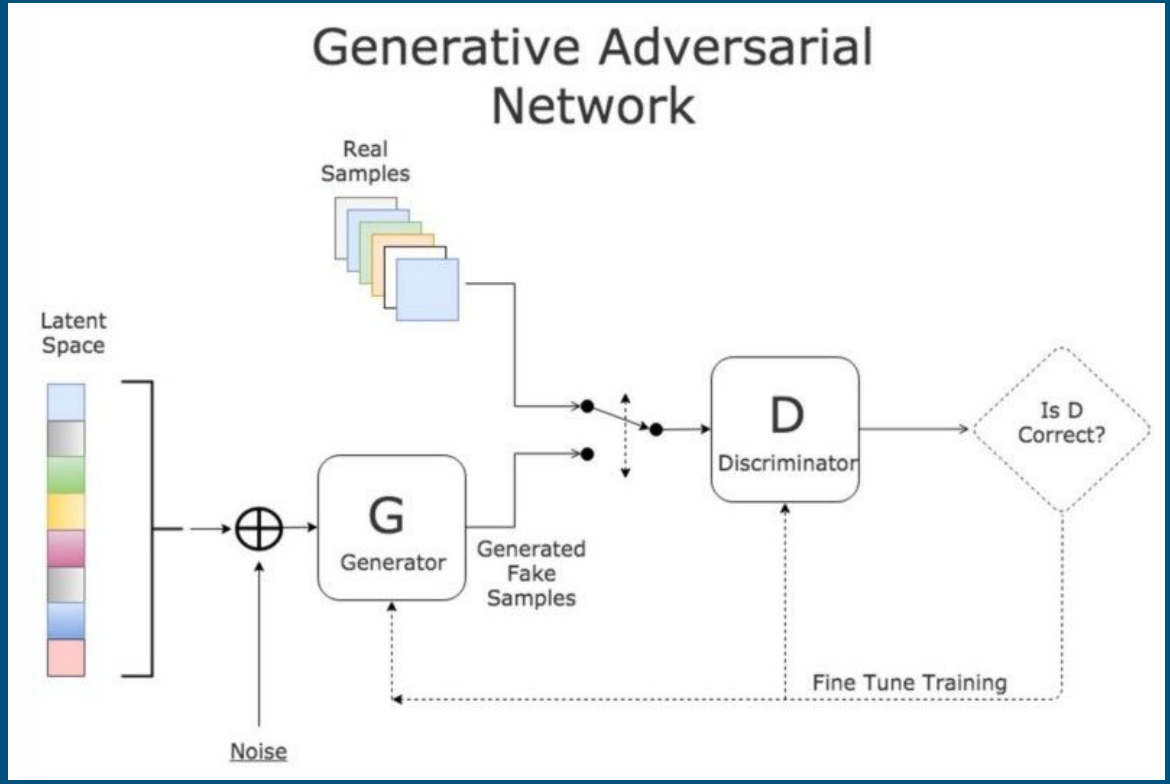
Implémenter un réseau de neurones avec les paramètres d'entré suivants:

- colisition haut: 1 si oui 0 sinon
- colisition droit: 1 si oui 0 sinon
- colisition bas: 1 si oui 0 sinon
- colisition gauche: 1 si oui 0 sinon
- deltaX tête - nouriture: 0 si egale 1 si positif -1 sinon
- deltaY tête - nouriture: 0 si egale 1 si positif -1 sinon

Et qui indique en sortie une des 4 directions que le serpent doit prendre.

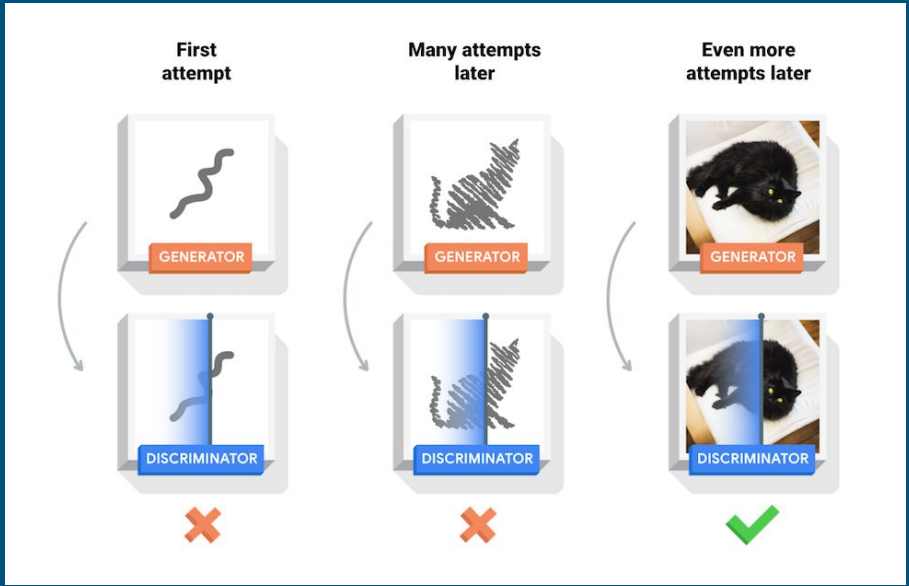
Vous pouvez entraîner le serpent manuellement ou de manière autonome.

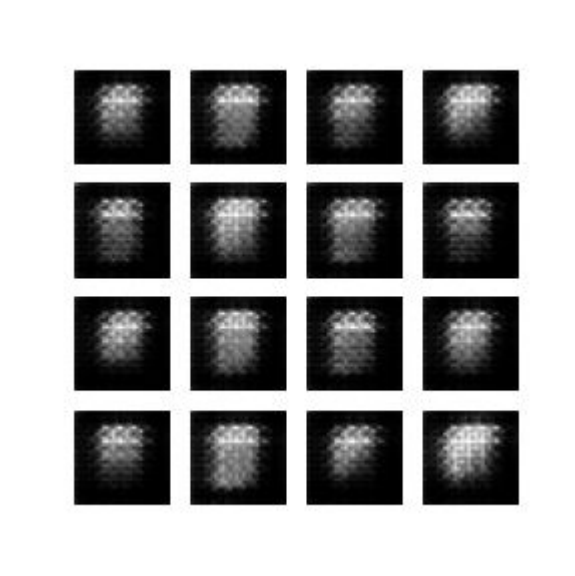
Réseaux GAN


Generative Adversarial Network
Real Samples
Latent Space
G
Generator
Generated Fake Samples
D
Discriminator
Is D Correct?
Noise
Fine Tune Training

Réseaux GAN

https://www.tensorflow.org/tutorials/generative/dcgan





Réseaux GAN

YOLO

https://github.com/zzh8829/yolov3-tf2