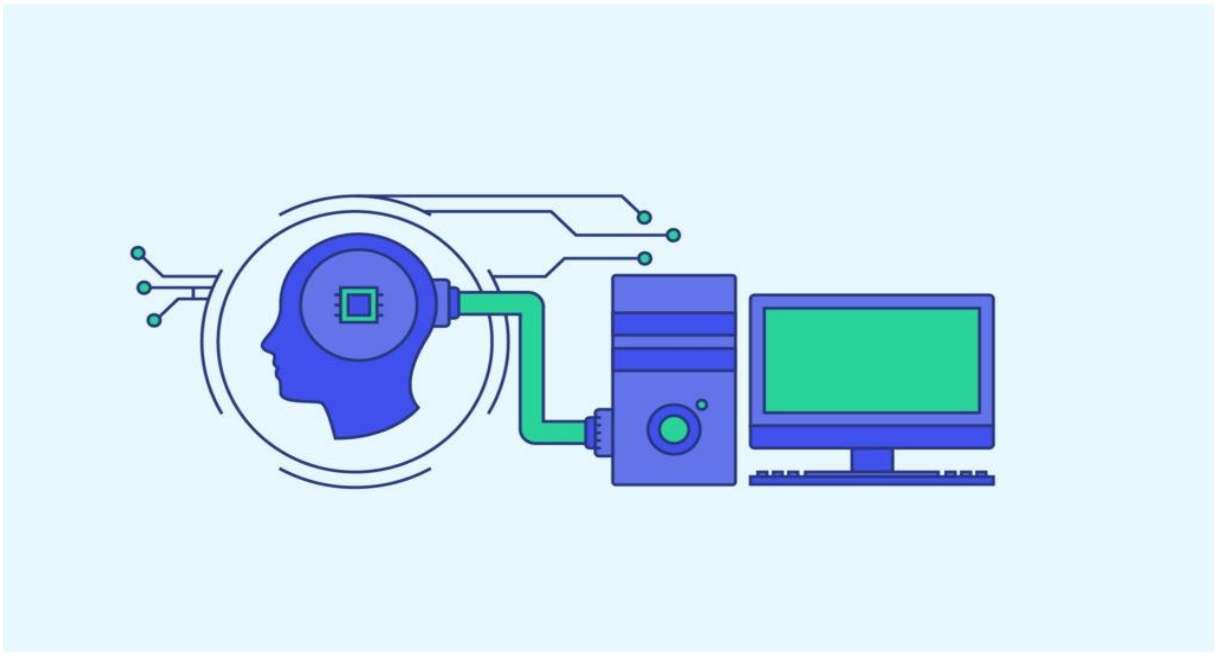
كلية العلوم و التقنيات فاس
ⵜⴰⵖⵚⵍⵓⵏⵜ ⵏ ⵜⵇⴰⵙⵙⵉⵙ ⴰ ⵜⵏⵙⵙⵜⵙⵏ
Faculté des Sciences et Techniques de Fès

جامعة سيدي محمد بن عبد الله
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵙⵉⴷⵉ ⵎⵓⵃⴰⵎⴻⴷ ⴱⵏ ⵄⴱⴷⵍⵍⴰⵀ
Université Sidi Mohamed Ben Abdellah

Master Systèmes Intelligents et Réseaux

# TP du Deep Learning avec Google Colab



**Encadré Par :**

Pr. Jamal Kharroubi

**Réalisé Par :**

- Soukaina Ezzaghdidi
- Soukayna Guarmouh
- Najwa Nadif

# Objectif :

L'objectif de ce TP est de créer un modèle de réseau neuronal convolutif **CNN** de **Deep Learning** en utilisant **google colab**, pour la classification des images radiographiques dans le cadre de la COVID-19.

# Qu'est-ce que Google Colaboratory :

Google Colaboratory ou Colab, un outil Google simple et gratuit pour vous initier au Deep Learning ou collaborer avec vos collègues sur des projets en science des données.
Colab permet :

- De développer des applications en Deep Learning en utilisant des bibliothèques Python populaires telles que ***Keras***, ***TensorFlow***, PyTorch et ***OpenCV***.
- D'utiliser un environnement de développement (***Jupyter Notebook***) qui ne nécessite aucune configuration.

Mais la fonctionnalité qui distingue Colab des autres services est l'accès à un **processeur graphique GPU.**

# Etapes pour activer Google Colab :

## Etape 1 :

Allez au site **:** https://colab.research.google.com

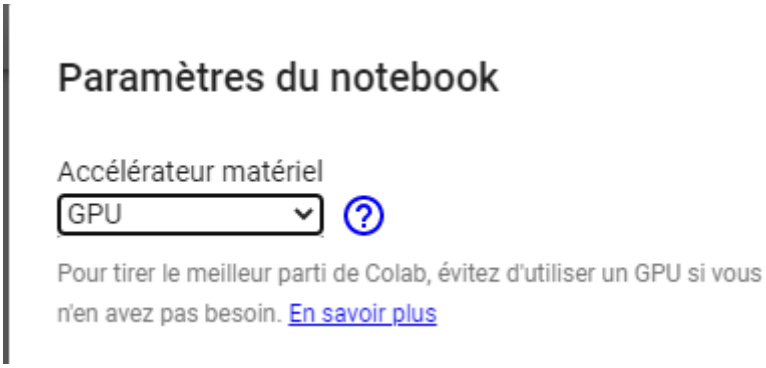
On choisit un nouveau notebook :


## Etape 2 :

Une fois dans le nouveau fichier, vous pouvez le renommer en cliquant sur le nom en haut du document :

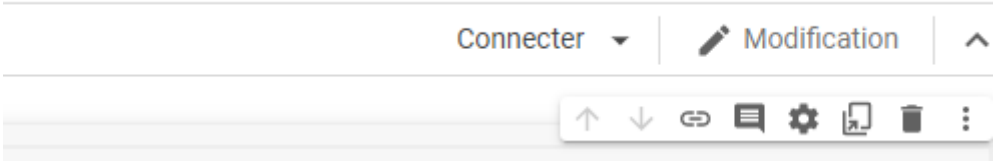
## Etape 3 :

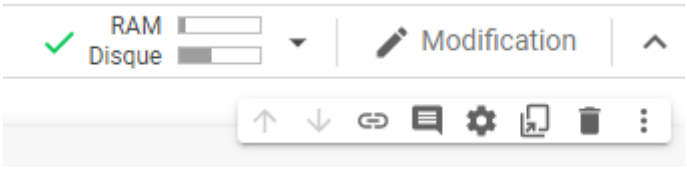
Pour configurer le GPU il suffit de cliquer sur **Exécution > Modifier le type d'exécution** et sélectionner **GPU** comme accélérateur matériel (pour que l'exécution soit rapide) :

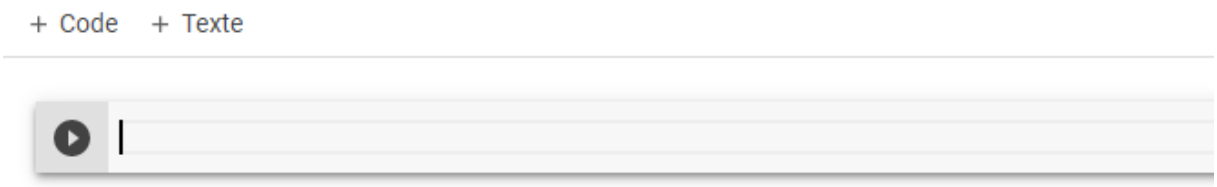

## Etape 4:

Cliquer sur **Connecter :**





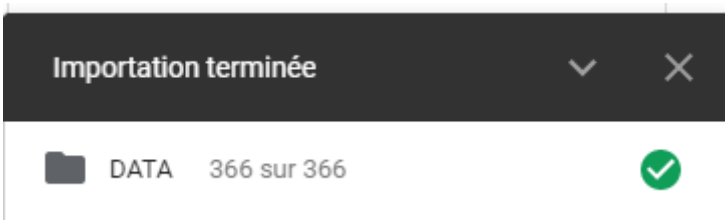

Nous pouvons dès maintenant commencer à utiliser Colab :




## Jeu de données :

Dans ce TP nous avons utilisé un ensemble d'images radiographiques durant d'un dépistage du COVID-19, notre jeu de données est devisé en deux partie une partie d'apprentissage (250 images) et une partie du test (110 images).

D'abord, il faut Importer le jeu de données dans google drive :

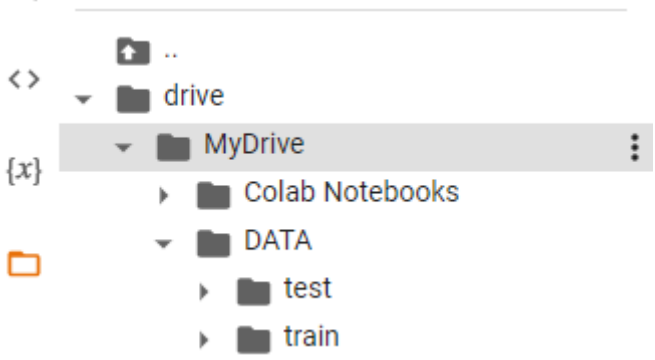



Ensuite, importer google drive dans notre colab :

```
#Importer google drive dans notre colab
from google.colab import drive
drive.mount('/content/drive')
```

```
drive.mount('/content/drive')
```

Maintenant on peut accéder au jeu de données (DATA) à partir du colab :



# Librairies utilisées :

```
import tensorflow as tf
import tensorflow.keras
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense, Activation, Flatten, Conv2D, MaxPool2D , Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.losses import binary_crossentropy
from tensorflow.keras import metrics
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, confusion_matrix, plot_confusion_matrix, ConfusionMatrixDisplay
import numpy as np
```

- ✓ **TensorFlow** : est une bibliothèque open source, permettant d'exécuter des applications de machine learning et de deep learning.
- ✓ **Keras** : est une bibliothèque open source qui permet de développer de nouveaux systèmes d'intelligence artificielle (IA). Dans un process de deep learning, il est ainsi possible de créer des réseaux de neurones artificiels.

## Construction du modèle CNN :

Construire le réseau neuronal nécessite de configurer les couches du modèle, puis de compiler le modèle :

```
# construire le modèle CNN
#conv2d(32, (3,3)=>32 filtre de taille 3X3
x = Conv2D(32, (3,3) , padding = 'same', activation='relu', name ='layer_1') (img_input)
x = Conv2D(64, (3,3) , padding = 'same', activation='relu', name ='layer_2') (x)
#MaxPool2D((2,2)=>réduire la taille de l'image ,strides=(2,2) glissement de 2X2
x = MaxPool2D((2,2), strides=(2,2), name ='layer_3') (x)
# couche sortie: dropout=0,25 pour améliorer la performance
x =Dropout(0.25)(x)
```

```
x = Conv2D(128, (3,3) , padding = 'same', activation='relu', name ='layer_6') (x)
x = MaxPool2D((2,2), strides=(2,2), name ='layer_7') (x)
x =Dropout(0.25)(x)

x =Flatten(name='layer_8')(x)
# 64 neuronnes
x = Dense(64, name = 'layer_9')(x)
x =Dropout(0.5)(x)
#on a deux classe dans la sortie (COVID+ ou COVID-)
x =Dense(2, activation='sigmoid', name ='predections')(x)
```

```
[ ] # génération du modèle
    model = Model(inputs = img_input, outputs  =x, name = 'CNN_COVID_19' )
```

```
# afficher la structure du réseau
model.summary()
```

```
Model: "CNN_COVID_19"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 img_input (InputLayer)      [(None, 224, 224, 3)]     0

 layer_1 (Conv2D)            (None, 224, 224, 32)      896

 layer_2 (Conv2D)            (None, 224, 224, 64)      18496

 layer_3 (MaxPooling2D)      (None, 112, 112, 64)      0

 dropout (Dropout)           (None, 112, 112, 64)      0

 layer_4 (Conv2D)            (None, 112, 112, 64)      36928

 layer_5 (MaxPooling2D)      (None, 56, 56, 64)        0

 dropout_1 (Dropout)         (None, 56, 56, 64)        0

 layer_6 (Conv2D)            (None, 56, 56, 128)       73856

 layer_7 (MaxPooling2D)      (None, 28, 28, 128)       0

 dropout_2 (Dropout)         (None, 28, 28, 128)       0

 layer_8 (Flatten)           (None, 100352)            0

 layer_9 (Dense)             (None, 64)                6422592

 dropout_3 (Dropout)         (None, 64)                0

 predections (Dense)         (None, 2)                 130

=================================================================
Total params: 6,552,898
Trainable params: 6,552,898
Non-trainable params: 0
```

## Compilation du modèle :

```
# Compiler le modèle
model.compile(optimizer = 'adam', loss = binary_crossentropy , metrics=['accuracy'])
```

**Loss** : fonction de perte, elle mesure la précision du modèle pendant l'entraînement.

## L'apprentissage :

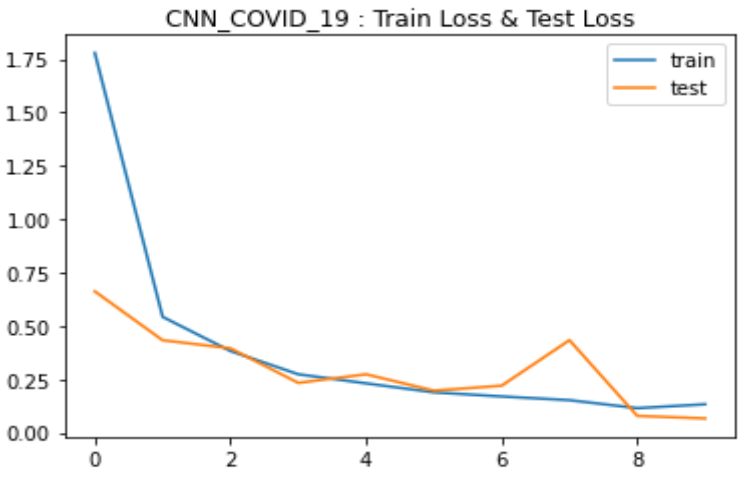
```
#32 image de données d'apprentissage
batch_size = 32
#l'historique de l'apprentissage
hist = model.fit(traindata,
                 steps_per_epoch = traindata.samples//batch_size,#250/32=7 itération pour le trainig dans chaque epoch
                 validation_data = testndata,
                 validation_steps = testndata.samples//batch_size,#110/32=3 itération pour le test
                 epochs = 10  #on a préciser 10 epoch
                 )
```

```
Epoch 1/10
7/7 [==============================] - 258s 36s/step - loss: 1.4348 - accuracy: 0.5826 - val_loss: 0.6762 - val_accuracy: 0.5000
Epoch 2/10
7/7 [==============================] - 14s 2s/step - loss: 0.5395 - accuracy: 0.8036 - val_loss: 0.5259 - val_accuracy: 0.7812
Epoch 3/10
7/7 [==============================] - 14s 2s/step - loss: 0.4673 - accuracy: 0.8440 - val_loss: 0.3989 - val_accuracy: 0.9167
Epoch 4/10
7/7 [==============================] - 14s 2s/step - loss: 0.2525 - accuracy: 0.9266 - val_loss: 0.3663 - val_accuracy: 0.8750
Epoch 5/10
7/7 [==============================] - 14s 2s/step - loss: 0.2204 - accuracy: 0.9062 - val_loss: 0.2377 - val_accuracy: 0.9271
Epoch 6/10
7/7 [==============================] - 14s 2s/step - loss: 0.2547 - accuracy: 0.9083 - val_loss: 0.2094 - val_accuracy: 0.9688
Epoch 7/10
7/7 [==============================] - 13s 2s/step - loss: 0.1892 - accuracy: 0.9404 - val_loss: 0.1707 - val_accuracy: 0.9583
Epoch 8/10
7/7 [==============================] - 14s 2s/step - loss: 0.1189 - accuracy: 0.9541 - val_loss: 0.1151 - val_accuracy: 0.9792
Epoch 9/10
7/7 [==============================] - 14s 2s/step - loss: 0.1104 - accuracy: 0.9633 - val_loss: 0.1930 - val_accuracy: 0.9479
Epoch 10/10
7/7 [==============================] - 14s 2s/step - loss: 0.0949 - accuracy: 0.9541 - val_loss: 0.2652 - val_accuracy: 0.9167
```

## Le graphe de la perte :

Maintenant, traçons les valeurs de la fonction de perte sur les ensembles de données d'apprentissage et de test :

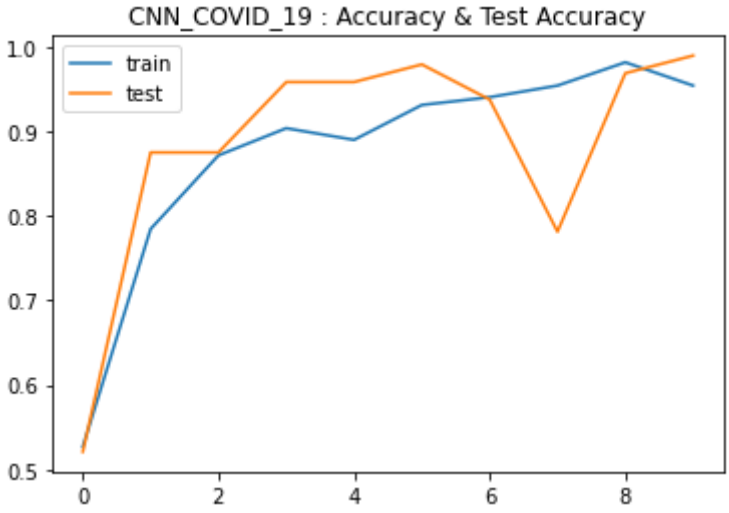
```
#plot la perte de train data et test data
plt.plot(hist.history['loss'], label = 'train')
plt.plot(hist.history['val_loss'], label= 'test')
plt.title('CNN_COVID_19 : Train Loss & Test Loss')
plt.legend()
plt.show()
```



## Le graphe de la précisssion :

Ensuite, comparez les performances du modèle sur l'ensemble de données de d'apprentissage et de test :

```
#plot la précision de train data et test data
plt.plot(hist.history['accuracy'], label = 'train')
plt.plot(hist.history['val_accuracy'], label= 'test')
plt.title('CNN_COVID_19 : Accuracy & Test Accuracy')
plt.legend()
plt.show()
```




## Schéma du TP :