Evènement	**TECHNO IT V1.0** ©
Formation	Deep Learning
Année universitaire	2020-2021
Intervenant	Dr. Taoufik BEN ABDALLAH, *Enseignant permanent à IIT*
Contenu	• Apprentissage de réseau de neurones profond • Application d'un réseau de neurones convolutif pour la classification • Estimation de performances

L'objectif de cet atelier est de vous faire acquérir les notions de base sur les réseaux de neurones convolutifs (CNNs). Ils constituent une architecture spéciale permettant de faire de l'apprentissage de tâches ayant trait à la vision par ordinateur. L'application que vous allez effectuer dans cet atelier consiste à détecter si une personne porte une bavette (mask) ou non en utilisant la base de données **Face-Mask**. Cette dernière représente un ensemble d'images de visage en couleur, de taille variée, qui sont annotées par deux classes : **"WithMask"** ou **"WithoutMask"**. Elle est composée d'un ensemble d'apprentissage de 10 000 exemples (5000 **"WithMask"** et 5000 **"WithoutMask"**), d'un ensemble de validation de 800 exemples (400 **"WithMask"** et 400 **"WithoutMask"**), et d'un ensemble de test de 992 exemples (483 **"WithMask"** et 509 **"WithoutMask"**).

On va utiliser **Google colab** ou **Colaboratory**. Il représente un service cloud, offert par Google (gratuit), basé sur Jupyter Notebook, et destiné principalement pour l'apprentissage automatique, particulièrement l'apprentissage profond. La fonctionnalité qui distingue Colab des autres services est l'accès à un processeur graphique **GPU** gratuitement. Par ailleurs, Il est livré avec des packages importants préinstallés et prêts à l'emploi. Ainsi, il enregistre explicitement tous les fichiers et notebooks en cours de traitement sur Google Drive.

- Télécharger la base de données **Face-Mask** sur votre compte Google drive.
- Télécharger le fichier ***"F_DL_seq.ipynb"*** via Moodle. Lancer Google colab dans un navigateur Web puis importer ***"F_DL_seq.ipynb"***

Importation des bibliothèques

```
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, accuracy_score
```

Pour assembler votre compte Drive dans Colaboratory, il suffit de créer ces deux instructions :

```
from google.colab import drive
drive.mount('/content/drive')
```

Travail à faire :

1. Décompresser le fichier du dataset ***FaceMask.zip***

2. Charger les données d'apprentissage, de validation et de test de la base de données **Face-Mask** respectivement dans des conteneurs de type **BatchDataset** $\boldsymbol{train\_ds}$, $\boldsymbol{val\_ds}$, et $\boldsymbol{test\_ds}$**.** Spécifier la taille des images à $\mathbf{28 \times 28}$ (**image_size=(28, 28)**)(utiliser la méthode **image_dataset_from_directory** de **tf.keras.preprocessing**)

3. Afficher la taille du premier batch de $\boldsymbol{train\_ds}$**.** Puis afficher les labels (classes) du premier batch de $\boldsymbol{train\_ds}$

NB. BatchDataset est un objet contenant les batches d'images. Cet objet représente :

- Un tenseur (matrice à $\boldsymbol{n}$ dimensions) de la forme $(batch\_size, H\_image, L\_image, nb\_channels)$ des images de chaque batch.
- Un tenseur unidimensionnelle de la forme $(batch\_size\ ,)$ des labels des images de chaque batch (label_batch)

4. Afficher les 9 premières images de l'ensemble d'apprentissage du premier batch (3 images dans chaque ligne) (*Spécifier le nom de la classe sous chaque image*)

Afin d'engendrer un gain de temps d'exécution sur le traitement de données, il suffit de créer les instructions suivantes :

```
AUTOTUNE = tf.data.AUTOTUNE
train_ds = train_ds.cache().prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
test_ds = test_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

BatchDataset. cache sert à forcer le maintien des données en cache dans la mémoire pour éviter la recharge des images à chaque fois, et **BatchDataset. prefetch** active le traitement de batch ultérieur en même temps avec l'apprentissage et l'évaluation du batch courant

5.

a. Générer un modèle CNN, nommé ***Fmask_cnn*** en respectant l'architecture suivante :

- Les valeurs des pixels de chaque image en entrée sont transformés de sorte que l'échelle devient entre 0 et 1 (utiliser **`Rescaling`** de **`tf.keras.layers.experimental.preprocessing`**)
- Le premier bloc convolutif comporte **deux couches** :
  - Couche de convolution dont le nombre de filtres est **3**, chacun de taille $\mathbf{3 \times 3}$, **stride=1**, et **padding="valid"**. La sortie de cette couche doit être passée à une fonction d'activation ReLU (utiliser la classe **`Conv2D`** de **`tf.keras.layers`**)
  - Couche de Maxpooling dont la taille du noyau est $\mathbf{2 \times 2}$ avec **stride=2** (utiliser la classe **`MaxPooling2D`** de **`tf.keras.layers`**)
- Le deuxième bloc convolutif comporte **trois couches** :
  - Deux couches de convolution dont le nombre de filtres est **2** chacun de taille $\mathbf{3 \times 3}$, **stride=1**, et **padding= "same"** . La sortie de ces couches doit être passée à une fonction d'activation ReLU
  - Couche de Maxpooling dont la taille du noyau est $\mathbf{2 \times 2}$ avec **stride=2**
- Le troisième bloc comporte **le Flatten** (utiliser la classe **`Flatten`** de **`tf.keras.layers`**)
- Le quatrième bloc constitue des couches entièrement connectées :
  - Deux couches chacun de 128 neurones. ***L'activation des neurones de ces couches se fait par la fonction* ReLU**
  - Une couche de sortie de 1 neurone retournant le résultat de la classification. ***L'activation de ce neurone se fait par la fonction* sigmoïde**
    (Utiliser la classe **`Dense`** de **`tf.keras.layers`** pour configurer les couches cachées et de sortie)

b. Compiler ***model_cnn*** en spécifiant l'optimiseur à ***"adam"***, la fonction de perte à **`"BinaryCrossentropy"`** (**`loss= tf.keras.losses.BinaryCrossentropy(from_logits=True)`**), et la métrique d'évaluation des étapes d'apprentissage à **`"accuracy"`**

c. Construire ***Fmask_cnn*** en spécifiant le nombre d'époques à 5 (passer l'ensemble de validation lors d'apprentissage)

6. Afficher une synthèse du modèle ***Fmask_cnn*** dans un TensorBoard. NB. TensorBoard est une boîte à outils de visualisation fournie avec TensorFlow)

7. Représenter la courbe d'évaluation de ***Fmask_cnn*** en fonction du nombre époques sur les données d'apprentissage et les données de validation

8. Représenter graphiquement la matrice de confusion associée aux données de test selon ***Fmask_cnn***

9. Calculer les mesures de perte (loss) et de précision (accuracy) de ***Fmask_cnn*** sur l'ensemble de données de test

Bon Travail ♣