Detection of Lung Diseases Using ResNet50

- Using ResNet50 for pulmonary disease detection generally involves a process of transfer learning and fine-tuning to leverage the capabilities of this pre-trained model and achieve accurate results in classifying chest X-ray images.

```
import cv2
import numpy as np
import pandas as pd
import seaborn as sns
import tensorflow as tf
import opendatasets as od
from tensorflow import keras
import matplotlib.pyplot as plt
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications import ResNet50V2
from sklearn.model_selection import train_test_split
from tensorflow.keras import layers, models, utils, callbacks
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.preprocessing.image import (ImageDataGenerator,
                                                  load_img, img_to_array)
from sklearn.metrics import (classification_report, accuracy_score,precision_score,
                             recall_score,confusion_matrix,roc_curve,roc_auc_score,
                             confusion_matrix)
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
import numpy as np
from glob import glob
import matplotlib.pyplot as plt
```

Importing the dataset directly from Kaggle

1. Élément de liste
2. Élément de liste

This dataset mainly consists of the chest X-ray images of Normal and Pneumonia affected patients. There is a total of 5840 chest X-ray images. It has two folders named train and test. Each of them has two sub-folders labeled as NORMAL and PNEUMONIA. This dataset can be used to detect pneumonia by training ResNet50 Model.

```
!pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (1.5.16)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle) (1.16.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from kaggle) (2023.11.17)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.31.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from kaggle) (4.66.1)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle) (8.0.1)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.0.7)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from kaggle) (6.1.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.6)
```

```
from google.colab import files
files.upload()
```

Sélect. fichiers Aucun fichier choisi Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving kaggle.json to kaggle.json
{'kaggle.json':
```

```
!pip install opendatasets
```

```
Collecting opendatasets
  Downloading opendatasets-0.1.22-py3-none-any.whl (15 kB)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from opendatasets) (4.66.1)
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (from opendatasets) (1.5.16)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from opendatasets) (8.1.7)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (1.16.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2023.11.17)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2.31.0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (8.0.1)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2.0.7)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (6.1.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->kaggle->opendatasets) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle->opendata
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle->opendatas
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle->opendatasets) (3.6)
Installing collected packages: opendatasets
Successfully installed opendatasets-0.1.22
```

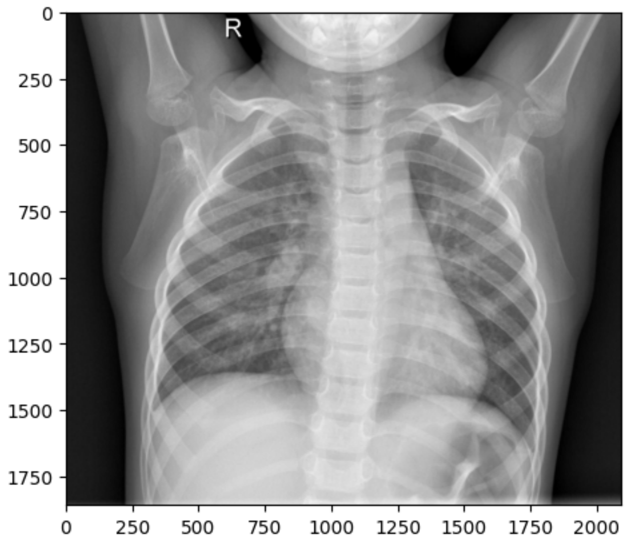
```
import opendatasets as od

# Download data from Kaggle using my API key
od.download("https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia")
```

```
Downloading chest-xray-pneumonia.zip to ./chest-xray-pneumonia
100%|██████████| 2.29G/2.29G [00:20<00:00, 123MB/s]
```

```
# Test plot images
img=cv2.imread('./chest-xray-pneumonia/chest_xray/train/NORMAL/IM-0115-0001.jpeg')
img2 = img[:,:,::-1]
plt.imshow(img2)
```

```
<matplotlib.image.AxesImage at 0x7aaf92e309a0>
```




```
train_path = './chest-xray-pneumonia/chest_xray/train'
valid_path = './chest-xray-pneumonia/chest_xray/test'
```

```
resnet = ResNet50(input_shape = (224,224,3) ,weights = 'imagenet',include_top = False)
```

- ResNet50 is a convolutional neural network architecture introduced by Microsoft Research in 2015. It is part of the ResNet (Residual Network) family, known for its depth and efficient training methodologies.
- ResNet50 specifically refers to a Residual Network with 50 layers, including convolutional layers, pooling layers, fully connected layers, and skip connections. The key innovation of ResNet is the use of residual blocks, which enable the network to be significantly deeper while overcoming the vanishing gradient problem encountered in very deep networks.
- ResNet50 has been pretrained on a large dataset (e.g., ImageNet) and has shown excellent performance in various computer vision tasks, including image classification, object detection, and feature extraction. Due to its depth and learned features, it's often used as a feature extractor or as a base model for transfer learning in various image-related tasks.

```
for layer in resnet.layers[:15]:
    layer.trainable = False

for i, layer in enumerate(resnet.layers):
    print(i, layer.name, layer.trainable)
```

```
117 conv4_block4_2_bn True
118 conv4_block4_2_relu True
119 conv4_block4_3_conv True
120 conv4_block4_3_bn True
121 conv4_block4_add True
122 conv4_block4_out True
123 conv4_block5_1_conv True
124 conv4_block5_1_bn True
125 conv4_block5_1_relu True
126 conv4_block5_2_conv True
127 conv4_block5_2_bn True
128 conv4_block5_2_relu True
129 conv4_block5_3_conv True
130 conv4_block5_3_bn True
131 conv4_block5_add True
132 conv4_block5_out True
133 conv4_block6_1_conv True
134 conv4_block6_1_bn True
135 conv4_block6_1_relu True
136 conv4_block6_2_conv True
137 conv4_block6_2_bn True
138 conv4_block6_2_relu True
139 conv4_block6_3_conv True
140 conv4_block6_3_bn True
141 conv4_block6_add True
142 conv4_block6_out True
143 conv5_block1_1_conv True
144 conv5_block1_1_bn True
145 conv5_block1_1_relu True
146 conv5_block1_2_conv True
147 conv5_block1_2_bn True
148 conv5_block1_2_relu True
149 conv5_block1_0_conv True
150 conv5_block1_3_conv True
151 conv5_block1_0_bn True
152 conv5_block1_3_bn True
153 conv5_block1_add True
154 conv5_block1_out True
155 conv5_block2_1_conv True
156 conv5_block2_1_bn True
157 conv5_block2_1_relu True
158 conv5_block2_2_conv True
159 conv5_block2_2_bn True
160 conv5_block2_2_relu True
161 conv5_block2_3_conv True
162 conv5_block2_3_bn True
163 conv5_block2_add True
164 conv5_block2_out True
165 conv5_block3_1_conv True
166 conv5_block3_1_bn True
167 conv5_block3_1_relu True
168 conv5_block3_2_conv True
169 conv5_block3_2_bn True
170 conv5_block3_2_relu True
171 conv5_block3_3_conv True
172 conv5_block3_3_bn True
173 conv5_block3_add True
174 conv5_block3_out True
```

```
x = resnet.output
x = Flatten()(x) # Flatten dimensions to for use in FC layers
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x) # Dropout layer to reduce overfitting
x = Dense(256, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(128, activation='relu')(x)
x = Dense(2, activation='softmax')(x) # Softmax for multiclass
model = Model(inputs=resnet.input, outputs=x)
```

```
model.summary()
```

```
 conv5_block3_1_bn (BatchNo  (None, 7, 7, 512)            2048      ['conv5_block3_1_conv[0][0]']
 rmalization)

 conv5_block3_1_relu (Activ  (None, 7, 7, 512)            0         ['conv5_block3_1_bn[0][0]']
 ation)

 conv5_block3_2_conv (Conv2  (None, 7, 7, 512)            2359808   ['conv5_block3_1_relu[0][0]']
 D)

 conv5_block3_2_bn (BatchNo  (None, 7, 7, 512)            2048      ['conv5_block3_2_conv[0][0]']
 rmalization)

 conv5_block3_2_relu (Activ  (None, 7, 7, 512)            0         ['conv5_block3_2_bn[0][0]']
 ation)

 conv5_block3_3_conv (Conv2  (None, 7, 7, 2048)           1050624   ['conv5_block3_2_relu[0][0]']
 D)

 conv5_block3_3_bn (BatchNo  (None, 7, 7, 2048)           8192      ['conv5_block3_3_conv[0][0]']
 rmalization)

 conv5_block3_add (Add)      (None, 7, 7, 2048)           0         ['conv5_block2_out[0][0]',
                                                                     'conv5_block3_3_bn[0][0]']

 conv5_block3_out (Activati  (None, 7, 7, 2048)           0         ['conv5_block3_add[0][0]']
 on)

 flatten_2 (Flatten)         (None, 100352)               0         ['conv5_block3_out[0][0]']

 dense_8 (Dense)             (None, 512)                  5138073   ['flatten_2[0][0]']
                                                          6

 dropout_4 (Dropout)         (None, 512)                  0         ['dense_8[0][0]']

 dense_9 (Dense)             (None, 256)                  131328    ['dropout_4[0][0]']

 dropout_5 (Dropout)         (None, 256)                  0         ['dense_9[0][0]']

 dense_10 (Dense)            (None, 128)                  32896     ['dropout_5[0][0]']

 dense_11 (Dense)            (None, 2)                    258       ['dense_10[0][0]']

==================================================================================================
Total params: 75132930 (286.61 MB)
Trainable params: 74995586 (286.09 MB)
Non-trainable params: 137344 (536.50 KB)
__________________________________________________________________________________________________
```

```
model.compile(
  loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)
```

```
# Use the Image Data Generator to import the images from the dataset
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)
```

```
# Make sure you provide the same target size as initialied for the image size
training_set = train_datagen.flow_from_directory('./chest-xray-pneumonia/chest_xray/train',
                                                 target_size = (224, 224),
                                                 batch_size = 32,
                                                 class_mode = 'categorical')
```

```
Found 5216 images belonging to 2 classes.
```

```
test_set = test_datagen.flow_from_directory('./chest-xray-pneumonia/chest_xray/test',
                                            target_size = (224, 224),
                                            batch_size = 32,
                                            class_mode = 'categorical')
```

```
Found 624 images belonging to 2 classes.
```

```
r = model.fit_generator(
  training_set,
  validation_data=test_set,
  epochs=20,
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set)
)
```

```
<ipython-input-72-dcaa1ca38143>:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please
  r = model.fit_generator(
Epoch 1/20
163/163 [==============================] - 169s 793ms/step - loss: 0.8292 - accuracy: 0.8474 - val_loss: 1628173.7500 - val_accuracy
Epoch 2/20
163/163 [==============================] - 125s 766ms/step - loss: 0.2821 - accuracy: 0.8986 - val_loss: 11986.3428 - val_accuracy:
Epoch 3/20
163/163 [==============================] - 136s 833ms/step - loss: 0.1998 - accuracy: 0.9220 - val_loss: 0.7606 - val_accuracy: 0.62
Epoch 4/20
163/163 [==============================] - 130s 798ms/step - loss: 0.1968 - accuracy: 0.9287 - val_loss: 1.7064 - val_accuracy: 0.62
Epoch 5/20
163/163 [==============================] - 132s 807ms/step - loss: 0.1481 - accuracy: 0.9454 - val_loss: 0.6186 - val_accuracy: 0.73
Epoch 6/20
163/163 [==============================] - 132s 804ms/step - loss: 0.1332 - accuracy: 0.9534 - val_loss: 0.5153 - val_accuracy: 0.78
Epoch 7/20
163/163 [==============================] - 124s 759ms/step - loss: 0.1409 - accuracy: 0.9513 - val_loss: 0.4282 - val_accuracy: 0.88
Epoch 8/20
163/163 [==============================] - 133s 814ms/step - loss: 1.2311 - accuracy: 0.8984 - val_loss: 814.4869 - val_accuracy: 0
Epoch 9/20
163/163 [==============================] - 144s 882ms/step - loss: 0.2686 - accuracy: 0.9089 - val_loss: 0.6840 - val_accuracy: 0.75
Epoch 10/20
163/163 [==============================] - 128s 784ms/step - loss: 0.2251 - accuracy: 0.9293 - val_loss: 0.3580 - val_accuracy: 0.85
Epoch 11/20
163/163 [==============================] - 120s 736ms/step - loss: 0.1779 - accuracy: 0.9358 - val_loss: 1.1493 - val_accuracy: 0.86
Epoch 12/20
163/163 [==============================] - 119s 729ms/step - loss: 0.1833 - accuracy: 0.9312 - val_loss: 0.3554 - val_accuracy: 0.85
Epoch 13/20
163/163 [==============================] - 120s 734ms/step - loss: 0.1570 - accuracy: 0.9415 - val_loss: 0.3281 - val_accuracy: 0.87
Epoch 14/20
163/163 [==============================] - 119s 731ms/step - loss: 0.1575 - accuracy: 0.9404 - val_loss: 0.3280 - val_accuracy: 0.86
Epoch 15/20
163/163 [==============================] - 118s 724ms/step - loss: 0.1469 - accuracy: 0.9475 - val_loss: 0.3266 - val_accuracy: 0.87
Epoch 16/20
163/163 [==============================] - 119s 732ms/step - loss: 0.1387 - accuracy: 0.9475 - val_loss: 0.2828 - val_accuracy: 0.88
Epoch 17/20
163/163 [==============================] - 119s 730ms/step - loss: 0.1293 - accuracy: 0.9475 - val_loss: 0.3267 - val_accuracy: 0.88
Epoch 18/20
163/163 [==============================] - 119s 727ms/step - loss: 0.2937 - accuracy: 0.9204 - val_loss: 0.7358 - val_accuracy: 0.66
Epoch 19/20
163/163 [==============================] - 120s 736ms/step - loss: 0.2845 - accuracy: 0.9218 - val_loss: 0.4679 - val_accuracy: 0.78
Epoch 20/20
163/163 [==============================] - 119s 730ms/step - loss: 0.1853 - accuracy: 0.9369 - val_loss: 0.3881 - val_accuracy: 0.86
```

Double-cliquez (ou appuyez sur Entrée) pour modifier

Conclusion: Les résultats affichent l'évolution de l'entraînement d'un modèle sur 20 epochs.

Perte (Loss) : La perte diminue généralement au fil des epochs, ce qui indique que le modèle apprend progressivement à réduire les erreurs lors de l'entraînement.

Exactitude (Accuracy) : L'exactitude augmente globalement, montrant que le modèle devient meilleur pour prédire les bonnes étiquettes pour les données d'entraînement.

Overfitting : Il semble y avoir des signes de surapprentissage (overfitting) car l'exactitude des données d'entraînement (training accuracy) est bien supérieure à celle des données de validation (validation accuracy) à partir de l'epoch 8. Cela peut indiquer que le modèle commence à trop s'adapter aux données spécifiques d'entraînement et ne généralise pas bien sur de nouvelles données.

Stabilité des performances : À partir de l'epoch 13, la perte et l'exactitude sur les données de validation semblent se stabiliser, ce qui peut indiquer que le modèle n'apprend plus significativement après cet epoch.

Perspectives:

Pour améliorer davantage ces résultats:

- Réduire l'overfitting : Envisager des approches de régularisation comme l'application de couches Dropout ou l'utilisation de la régularisation L2, ainsi que la simplification du modèle ou l'adoption de techniques d'augmentation de données pour renforcer sa généralisation.
- Appliquer des méthodes de réglage hyperparamétrique : Établir les hyperparamètres optimaux pour le modèle, par exemple, ajuster le taux d'apprentissage, la profondeur des couches, etc., à travers des itérations itératives.