X-ray image classification based on Pneumonia Positive or Negative (Clean/Unbalanced data)

```
import tensorflow as tf
import numpy as np
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

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

```
    Mounted at /content/drive
```

```
# Loading training dataset
training_dir="/content/drive/My Drive/problem2/Training"
training_generator=ImageDataGenerator(rescale=1/255,featurewise_center=False,
        rotation_range = 30,
        zoom_range = 0.2,
        width_shift_range=0.1,
        height_shift_range=0.1,
        horizontal_flip = False,
        vertical_flip=False)
train_generator=training_generator.flow_from_directory(training_dir,target_size=(200,200),batch_size=4,class_mode='binary')
```

```
    Found 400 images belonging to 2 classes.
```

```
# Loading validation dataset
validation_dir="/content/drive/My Drive/problem2/Validation"
validation_generator=ImageDataGenerator(rescale=1/255)
val_generator=validation_generator.flow_from_directory(validation_dir,target_size=(200,200),batch_size=4,class_mode='binary'
```

```
    Found 16 images belonging to 2 classes.
```

```
# Loading test dataset
test_dir="/content/drive/My Drive/problem2/Testing"
test_generator=ImageDataGenerator(rescale=1/255)
```

```
test_generator=test_generator.flow_from_directory(test_dir,target_size=(200,200),batch_size=16,class_mode='binary')
```

```
Found 60 images belonging to 2 classes.
```

```
# Training the model
model=tf.keras.Sequential([
    tf.keras.layers.Conv2D(32,(3,3),input_shape=(200,200,3),activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(64,(3,3),activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Dropout(0.2),

    tf.keras.layers.Conv2D(128,(3,3),activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Dropout(0.2),

    tf.keras.layers.Conv2D(256,(3,3),activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(256,activation='relu'),
    tf.keras.layers.Dense(1,activation='sigmoid')

])


# Minimize the loss usign optimizers
model.compile(optimizer=tf.keras.optimizers.Adam(lr=0.001),loss='binary_crossentropy',metrics=['acc'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 198, 198, 32)      896       
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 99, 99, 32)        0         
_________________________________________________________________
```

```
conv2d_1 (Conv2D)            (None, 97, 97, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 48, 48, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 48, 48, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 46, 46, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 23, 23, 128)       0
_________________________________________________________________
dropout_1 (Dropout)          (None, 23, 23, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 21, 21, 256)       295168
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 10, 10, 256)       0
_________________________________________________________________
flatten (Flatten)            (None, 25600)             0
_________________________________________________________________
dropout_2 (Dropout)          (None, 25600)             0
_________________________________________________________________
dense (Dense)                (None, 256)               6553856
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 257
=================================================================
Total params: 6,942,529
Trainable params: 6,942,529
Non-trainable params: 0
_________________________________________________________________
```

```
# Validating train and validation dataset
history = model.fit_generator(train_generator,
          validation_data = val_generator,
          epochs = 30,
          verbose = 1)
```

```
Epoch 1/30
100/100 [==============================] - 49s 491ms/step - loss: 0.4331 - acc: 0.7900 - val_loss: 0.5520 - val_acc:
Epoch 2/30
100/100 [==============================] - 49s 486ms/step - loss: 0.3746 - acc: 0.8225 - val_loss: 0.5217 - val_acc:
Epoch 3/30
100/100 [==============================] - 48s 478ms/step - loss: 0.3649 - acc: 0.8050 - val_loss: 0.7151 - val_acc:
Epoch 4/30
100/100 [==============================] - 48s 484ms/step - loss: 0.3569 - acc: 0.8075 - val_loss: 0.5180 - val_acc:
```

```
100/100 [==============================] - 48s 484ms/step - loss: 0.5569 - acc: 0.8075 - val_loss: 0.5186 - val_acc:
Epoch 5/30
100/100 [==============================] - 48s 484ms/step - loss: 0.4788 - acc: 0.7950 - val_loss: 0.6176 - val_acc:
Epoch 6/30
100/100 [==============================] - 48s 479ms/step - loss: 0.3606 - acc: 0.8150 - val_loss: 0.5512 - val_acc:
Epoch 7/30
100/100 [==============================] - 49s 490ms/step - loss: 0.3583 - acc: 0.8350 - val_loss: 0.4569 - val_acc:
Epoch 8/30
100/100 [==============================] - 49s 491ms/step - loss: 0.3457 - acc: 0.8400 - val_loss: 0.4835 - val_acc:
Epoch 9/30
100/100 [==============================] - 49s 488ms/step - loss: 0.5854 - acc: 0.8600 - val_loss: 1.3036 - val_acc:
Epoch 10/30
100/100 [==============================] - 48s 480ms/step - loss: 0.5035 - acc: 0.7975 - val_loss: 0.5920 - val_acc:
Epoch 11/30
100/100 [==============================] - 48s 477ms/step - loss: 0.4118 - acc: 0.8075 - val_loss: 0.6336 - val_acc:
Epoch 12/30
100/100 [==============================] - 48s 481ms/step - loss: 0.3580 - acc: 0.8350 - val_loss: 0.6477 - val_acc:
Epoch 13/30
100/100 [==============================] - 48s 480ms/step - loss: 0.3795 - acc: 0.8400 - val_loss: 0.6371 - val_acc:
Epoch 14/30
100/100 [==============================] - 49s 488ms/step - loss: 0.3253 - acc: 0.8500 - val_loss: 0.5674 - val_acc:
Epoch 15/30
100/100 [==============================] - 49s 494ms/step - loss: 0.3139 - acc: 0.8725 - val_loss: 0.6613 - val_acc:
Epoch 16/30
100/100 [==============================] - 47s 474ms/step - loss: 0.3283 - acc: 0.8600 - val_loss: 0.7091 - val_acc:
Epoch 17/30
100/100 [==============================] - 48s 475ms/step - loss: 0.3220 - acc: 0.8400 - val_loss: 0.6331 - val_acc:
Epoch 18/30
100/100 [==============================] - 48s 476ms/step - loss: 0.3215 - acc: 0.8550 - val_loss: 0.5974 - val_acc:
Epoch 19/30
100/100 [==============================] - 48s 477ms/step - loss: 0.3292 - acc: 0.8625 - val_loss: 0.6389 - val_acc:
Epoch 20/30
100/100 [==============================] - 49s 487ms/step - loss: 0.3019 - acc: 0.8575 - val_loss: 0.6758 - val_acc:
Epoch 21/30
100/100 [==============================] - 48s 484ms/step - loss: 0.3103 - acc: 0.8700 - val_loss: 0.6284 - val_acc:
Epoch 22/30
100/100 [==============================] - 50s 497ms/step - loss: 0.2936 - acc: 0.8700 - val_loss: 0.6036 - val_acc:
Epoch 23/30
100/100 [==============================] - 47s 474ms/step - loss: 0.2674 - acc: 0.8725 - val_loss: 0.5718 - val_acc:
Epoch 24/30
100/100 [==============================] - 48s 476ms/step - loss: 0.3026 - acc: 0.8800 - val_loss: 0.5466 - val_acc:
Epoch 25/30
100/100 [==============================] - 47s 475ms/step - loss: 0.2841 - acc: 0.8850 - val_loss: 0.5577 - val_acc:
Epoch 26/30
```

```
Epoch 26/30
100/100 [==============================] - 48s 476ms/step - loss: 0.2814 - acc: 0.8750 - val_loss: 0.6191 - val_acc:
Epoch 27/30
100/100 [==============================] - 49s 488ms/step - loss: 0.2705 - acc: 0.8775 - val_loss: 0.5135 - val_acc:
Epoch 28/30
100/100 [==============================] - 49s 491ms/step - loss: 0.2637 - acc: 0.8900 - val_loss: 0.4861 - val_acc:
Epoch 29/30
100/100 [==============================] - 49s 486ms/step - loss: 0.2325 - acc: 0.8925 - val_loss: 0.5136 - val_acc:
```

```
# Plotting histographic graph for Training and Validation Accuracy
%matplotlib inline
import matplotlib.pyplot as plt
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'r', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend(loc=0)
plt.figure()
```

```
<Figure size 432x288 with 0 Axes>
```

Training and validation accuracy

```
# Finding Loss and Accuracy of the model
print("Loss of the model is - " , model.evaluate(test_generator)[0]*100 , "%")
print("Accuracy of the model is - " , model.evaluate(test_generator)[1]*100 , "%")
```

```
4/4 [==============================] - 2s 411ms/step - loss: 1.1622 - acc: 0.5500
Loss of the model is -  116.21673107147217 %
4/4 [==============================] - 2s 399ms/step - loss: 1.1622 - acc: 0.5500
Accuracy of the model is -  55.000001192092896 %
```

```
predicted_classes=model.predict_classes(test_generator)
```

0 5 10 15 20 25 30

```
# Confusion matrix in order to number of correct and wrond prediction
from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(test_generator.classes,predicted_classes))
```

```
[[ 6 24]
 [ 1 29]]
```

```
# Finding precision,recall,f1-score
print(classification_report(test_generator.classes,predicted_classes))
```

```
              precision    recall  f1-score   support

           0       0.29      0.07      0.11        30
           1       0.47      0.83      0.60        30

    accuracy                           0.45        60
   macro avg       0.38      0.45      0.36        60
weighted avg       0.38      0.45      0.36        60
```