Projeto - IA 2 - Deep Learning -Análise de Radiografia de Pulmões

PROFESSOR CESAR ALMIÑANA

ALUNOS: MARCIO FERNANDES CRUZ, EDUARDO BOLYHOS, ALISSON SALES e PAULO VIEGAS

LABDATA - AGOSTO/2022



Objetivo do Trabalho

Criar uma ferramenta de apoio a decisão a área médica, baseado em um modelo de IA, para tentar automatizar a detecção de anomalias em radiografias pulmonares.

Importação de Bibliotecas

Clique duas vezes (ou pressione "Enter") para editar

```
random_state = 20220731

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

from google.colab import auth
auth.authenticate_user()

from IPython.display import clear_output

import requests
gcloud_token = !gcloud auth print-access-token
gcloud_tokeninfo = requests.get('https://www.googleapis.com/oauth2/v3/tokeninfo?access_token=' + gcloud_token[0]).json()

email_logado = gcloud_tokeninfo['email']

import numpy as np
from sklearn.model_selection import train_test_split
from random import sample
import os
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt

from keras.applications import vgg16 as vgg
from keras.preprocessing import image as im
from keras.applications import vgg16 as vgg
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input, decode_predictions
from tensorflow.keras.applications import Xception

from keras.models import Model, Sequential
from keras.layers import Dense, Flatten
from tensorflow.keras.utils import to_categorical
import tensorflow as tf

clear_output()
```

Funções Utilizadas no Notebook

```
# Varre determinado diretório e gera uma lista respeitando o numero máximo de Observações
def get_lista_observacoes(pasta, limite_observacoes):
  if (limite_observacoes<len(os.listdir(pasta))):
    return sample(os.listdir(pasta), limite_observacoes)
  else:
```

```
    return os.listdir(pasta)

# A partir de uma lista de imagem, retorna um array de imagens e suas classes com o objetivo de "Fitar" o modelo
def get_lista_imagens(pasta, lista_observacoes, classe, target_width, target_height, aplicar_normalizacao):
  imagens, classes = [], []

  for arquivo in lista_observacoes:
    caminho_arquivo = os.path.join(pasta, arquivo)
    imagem_original = im.img_to_array(im.load_img(caminho_arquivo, target_size=(target_width, target_height)))

    if (aplicar_normalizacao==True):
      imagens.append(vgg.preprocess_input(imagem_original))
    else:
      imagens.append(imagem_original)

    classes.append(np.array(int(classe)))

  imagens = np.array(imagens)
  classes = np.array(classes)

  return imagens, classes

def get_acuracia_base_teste(history):
  return np.mean(history.history['val_accuracy'])

def get_variabilidade_base_teste(history):
  return np.std(history.history['val_accuracy'])

def get_grafico_analise(history):
  fig, (ax1, ax2) = plt.subplots(1, 2)

  fig.set_figwidth(30)
  fig.set_figheight(5)

  ax1.plot(history.history['loss'], label='Erro no Treinamento')
  ax1.plot(history.history['val_loss'], label='Erro no Teste')
  ax1.set_title('Análise de Erro')
  ax1.set_ylabel('Loss')
  ax1.set_xlabel('Época')
  ax1.legend();

  ax2.plot(history.history['accuracy'], label='Acurácia no Treino')
  ax2.plot(history.history['val_accuracy'], label='Acurácia no Teste')
  ax2.set_title('Análise da Acurácia')
  ax2.set_ylabel('Accuracy')
  ax2.set_xlabel('Época')
  ax2.legend();

  fig.suptitle('Análise Gráfica')
  fig.show()


# Função para exibir a avaliação na base de Teste
def exibir_avaliacao(history):
  print('Acurácia na Base de Teste: ', get_acuracia_base_teste(history))
  print('Variabilidade na Base de Teste: ', get_variabilidade_base_teste(history))
  get_grafico_analise(history)

# Usado nos métodos de Teste para Exibir o Pivot
def exibir_pivot_teste(modelo, imagens_test, classes_test):
  classificacoes = modelo.predict(imagens_test)
  classificacoes_modelo = (i.argmax() for i in classificacoes)

  matrix_dataframe = pd.DataFrame({'classe_original': classes_test, 'classe_modelo': classificacoes_modelo}, index=range(imagens_test.
  matrix_dataframe['count'] = 1

  pivot = matrix_dataframe.pivot_table(values = 'count', index='classe_original', columns='classe_modelo', aggfunc='sum').fillna(0)

  pivot.rename(index={0: 'Normal', 1: 'Pneumo-Bact', 2: 'Pneumo-Viral', 3: 'Covid' },
               columns={0: 'Normal', 1: 'Pneumo-Bact', 2: 'Pneumo-Viral', 3: 'Covid' }, inplace=True)

  print(pivot)
  plt.figure(figsize=(20,8))
  sb.heatmap(pivot / pivot.sum(axis=1), cmap='YlGnBu', annot=True, annot_kws={"fontsize":12});

def exibir_pivot_validacao(df):
  pivot = df.pivot_table(values = 'count', index='classe_original', columns='classe_modelo', aggfunc='sum').fillna(0)

  pivot.rename(index={0: 'Normal', 1: 'Pneumo-Bact', 2: 'Pneumo-Viral', 3: 'Covid' },
               columns={0: 'Normal', 1: 'Pneumo-Bact', 2: 'Pneumo-Viral', 3: 'Covid' }, inplace=True)

  print(pivot)
```

```
  plt.figure(figsize=(20,8))
  sb.heatmap(pivot / pivot.sum(axis=1), cmap='YlGnBu', annot=True, annot_kws={"fontsize":12});


# Função criada para executar previsões sobre arquivos unitários e alimentar em 1 array
def get_df_previsao(modelo, pasta, lista_observacoes, classe, target_width, target_height, aplicar_normalizacao):

  df_resultado = pd.DataFrame(columns=['classe_original', 'classe_modelo', 'count'])

  for arquivo in lista_observacoes:
    caminho = os.path.join(pasta, arquivo)
    imagem = im.img_to_array(im.load_img(caminho, target_size=(target_width, target_height)))

    if (aplicar_normalizacao==True):
      imagem = vgg.preprocess_input(imagem)

    imagem = np.expand_dims(imagem, axis=0)

    classificacao = modelo.predict(imagem)

    dicionario = dict(classe_original = classe,
                classe_modelo = classificacao.argmax(),
                count= 1)
    df_auxiliar = pd.DataFrame([dicionario])

    df_resultado = pd.concat([df_resultado, df_auxiliar])

  return df_resultado
```

Configurações que podem ser trocadas pelo Usuário

```
# Definir limite de arquivos a serem observados e tamanho de imagem
# o total de arquivos não pode estourar o numero de registros que tem em determinada pasta
# Estas observações, são por tipo de classe

limite_observacoes = 500
limite_predicoes = 600

ratio = (400/300)

target_width = 80 # Para XCepion, no minimo 71
target_height = round(target_width*ratio)
epocas = 10


# Verifiquei que o shape da imagem normal é 300 x 400 por 3 canais, mas as outras com doença associada possuem tamanhos maiores

# Definir caminhos das pastas onde estão as observaçòes
if email_logado=='marcio@marciofcruz.com':
  pasta_raiz = '/content/drive/MyDrive/Colab Notebooks/DataSet Data Science/imagenspulmao/Curated X-Ray Dataset'
elif email_logado=='alissondeandrade@gmail.com':
  pasta_raiz = '/content/drive/MyDrive/MBA_FIA_DS_IA/DISCIPLINAS/4.IA_2/Curated X-Ray Dataset'
elif email_logado=='sonekabolyhos@gmail.com':
  pasta_raiz = '/content/drive/MyDrive/Colab Notebooks/DataSet Data Science/imagenspulmao/Curated X-Ray Dataset'
elif email_logado=='prv.viegas@gmail.com':
  pasta_raiz = '/content/drive/MyDrive/Colab Notebooks/DataSet Data Science/imagenspulmao/Curated X-Ray Dataset'
else:
  pasta_raiz = "Não encontrado o email"

pasta_normal = os.path.join(pasta_raiz, 'Normal')
pasta_pneumonia_bacterial = os.path.join(pasta_raiz, 'Pneumonia-Bacterial')
pasta_pneumonia_viral = os.path.join(pasta_raiz, 'Pneumonia-Viral')
pasta_covid_19  = os.path.join(pasta_raiz, 'COVID-19')

print(f'A pasta raiz do {email_logado} é {pasta_raiz}')

if (not os.path.exists(pasta_normal)):
  print('Caminho não existe: ', pasta_normal, ':', len(os.listdir(pasta_normal)))

if (not os.path.exists(pasta_normal)):
  print('Caminho não existe: ', pasta_pneumonia_bacterial, ':', len(os.listdir(pasta_pneumonia_bacterial)))

if (not os.path.exists(pasta_normal)):
  print('Caminho não existe: ', pasta_pneumonia_viral, ':', len(os.listdir(pasta_pneumonia_viral)))

if (not os.path.exists(pasta_normal)):
  print('Caminho não existe: ', pasta_covid_19, ':', len(os.listdir(pasta_covid_19)))

observacoes_normal = get_lista_observacoes(pasta_normal, limite_observacoes)
observacoes_pneunomia_bacterial = get_lista_observacoes(pasta_pneumonia_bacterial, limite_observacoes)
```

```
observacoes_pneunomia_viral = get_lista_observacoes(pasta_pneumonia_viral, limite_observacoes)
observacoes_covid_19 = get_lista_observacoes(pasta_covid_19, limite_observacoes)

# Lista de imagens para Validação onde não estou considerando imagens já processadas
observacoes_normal_validacao = sample(list(set(os.listdir(pasta_normal)) - set(observacoes_normal)), limite_predicoes)
observacoes_pneumonia_bacterial_validacao  = sample(list(set(os.listdir(pasta_pneumonia_bacterial)) - set(observacoes_pneunomia_bacter
observacoes_pneumonia_viral_validacao  = sample(list(set(os.listdir(pasta_pneumonia_viral)) - set(observacoes_pneunomia_viral)), limit
observacoes_covid_19_validacao  = sample(list(set(os.listdir(pasta_covid_19)) - set(observacoes_covid_19)), limite_predicoes)
```

```
A pasta raiz do marcio@marciofcruz.com é /content/drive/MyDrive/Colab Notebooks/DataSet Data Science/imagenspulmao/Curated X-Ray Da
```

```
pasta_raiz = "/content/drive/MyDrive/Colab Notebooks/DataSet Data Science/imagenspulmao/Curated X-Ray Dataset"

# Definir limite de arquivos a serem observados e tamanho de imagem
# o total de arquivos não pode estourar o numero de registros que tem em determinada pasta
# Estas observações, são por tipo de classe



limite_observacoes = 500
limite_predicoes = 600

ratio = (400/300)

target_width = 80 # Para XCepion, no minimo 71
target_height = round(target_width*ratio)
epocas = 10

pasta_normal = os.path.join(pasta_raiz, 'Normal')
pasta_pneumonia_bacterial = os.path.join(pasta_raiz, 'Pneumonia-Bacterial')
pasta_pneumonia_viral = os.path.join(pasta_raiz, 'Pneumonia-Viral')
pasta_covid_19  = os.path.join(pasta_raiz, 'COVID-19')

print(f'A pasta raiz do {email_logado} é {pasta_raiz}')

if (not os.path.exists(pasta_normal)):
  print('Caminho não existe: ', pasta_normal, ':', len(os.listdir(pasta_normal)))

if (not os.path.exists(pasta_normal)):
  print('Caminho não existe: ', pasta_pneumonia_bacterial, ':', len(os.listdir(pasta_pneumonia_bacterial)))

if (not os.path.exists(pasta_normal)):
  print('Caminho não existe: ', pasta_pneumonia_viral, ':', len(os.listdir(pasta_pneumonia_viral)))

if (not os.path.exists(pasta_normal)):
  print('Caminho não existe: ', pasta_covid_19, ':', len(os.listdir(pasta_covid_19)))

observacoes_normal = get_lista_observacoes(pasta_normal, limite_observacoes)
observacoes_pneunomia_bacterial = get_lista_observacoes(pasta_pneumonia_bacterial, limite_observacoes)
observacoes_pneunomia_viral = get_lista_observacoes(pasta_pneumonia_viral, limite_observacoes)
observacoes_covid_19 = get_lista_observacoes(pasta_covid_19, limite_observacoes)

# Lista de imagens para Validação onde não estou considerando imagens já processadas
observacoes_normal_validacao = sample(list(set(os.listdir(pasta_normal)) - set(observacoes_normal)), limite_predicoes)
observacoes_pneumonia_bacterial_validacao  = sample(list(set(os.listdir(pasta_pneumonia_bacterial)) - set(observacoes_pneunomia_bacter
observacoes_pneumonia_viral_validacao  = sample(list(set(os.listdir(pasta_pneumonia_viral)) - set(observacoes_pneunomia_viral)), limit
observacoes_covid_19_validacao  = sample(list(set(os.listdir(pasta_covid_19)) - set(observacoes_covid_19)), limite_predicoes)
```

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

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

Carregar Array de Imagens e Classes

```
imagens_normal, classes_normal = get_lista_imagens(pasta_normal, observacoes_normal, 0, target_width, target_height, True)
imagens_pneumonia_bacterial, classes_pneumonia_bacterial = get_lista_imagens(pasta_pneumonia_bacterial, observacoes_pneunomia_bacteria
imagens_pneumonia_viral, classes_pneumonia_viral = get_lista_imagens(pasta_pneumonia_viral ,observacoes_pneunomia_viral, 2, target_wid
imagens_covid_19, classes_covid_19 = get_lista_imagens(pasta_covid_19, observacoes_covid_19, 3, target_width, target_height, True)

# Separar Train e Test
imagens_normal_train, imagens_normal_test, classes_normal_train, classes_normal_test = train_test_split(imagens_normal, classes_normal
imagens_pneumonia_bacterial_train, imagens_pneumonia_bacterial_test, classes_pneumonia_bacterial_train, classes_pneumonia_bacterial_te
imagens_pneumonia_viral_train, imagens_pneumonia_viral_test, classes_pneumonia_viral_train, classes_pneumonia_viral_test = train_test_
imagens_covid_19_train, imagens_covid_19_test, classes_covid_19_train, classes_covid_19_test = train_test_split(imagens_covid_19, clas

imagens_train = np.concatenate((imagens_normal_train, imagens_pneumonia_bacterial_train, imagens_pneumonia_viral_train, imagens_covid_
classes_train = np.concatenate((classes_normal_train, classes_pneumonia_bacterial_train, classes_pneumonia_viral_train, classes_covid_
```

```
classes_train_cat = to_categorical(classes_train, 4) # Categorizar as 4 classes - Normal, Covid, Pneumonia Viral e Bacteriana

imagens_test = np.concatenate((imagens_normal_test, imagens_pneumonia_bacterial_test, imagens_pneumonia_viral_test, imagens_covid_19_t
classes_test = np.concatenate((classes_normal_test, classes_pneumonia_bacterial_test, classes_pneumonia_viral_test, classes_covid_19_t
classes_test_cat = to_categorical(classes_test, 4) # Categorizar as 4 classes - Normal, Covid, Pneumonia Viral e Bacteriana

# Apagar objetos para economizar RAM
del imagens_normal_train, imagens_pneumonia_bacterial_train, imagens_pneumonia_viral_train, imagens_covid_19_train
del classes_normal_train, classes_pneumonia_bacterial_train, classes_pneumonia_viral_train, classes_covid_19_train

del imagens_normal_test, imagens_pneumonia_bacterial_test, imagens_pneumonia_viral_test, imagens_covid_19_test
del classes_normal_test, classes_pneumonia_bacterial_test, classes_pneumonia_viral_test, classes_covid_19_test

```

Arquitetura VGG16

Inicialização e Compilação

```
base_vgg = vgg.VGG16(weights='imagenet', include_top=False, input_shape=(target_width, target_height, 3))
base_vgg.trainable = False

modelo = Sequential(base_vgg)
modelo.add(Flatten())
modelo.add(Dense(64, activation='relu'))
modelo.add(Dense(32, activation='relu'))
modelo.add(Dense(4, activation='softmax'))

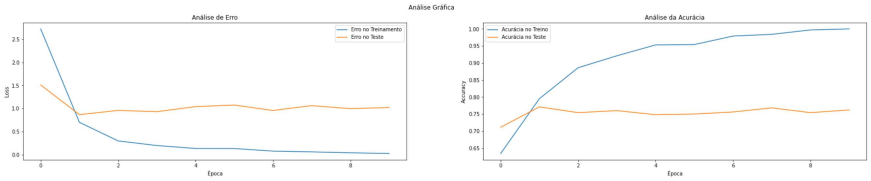
# modelo.summary()

# Compilar o Modelo
modelo.compile(loss='categorical_crossentropy', #loss=tf.losses.SparseCategoricalCrossentropy(),
              optimizer='adam',
              metrics=['accuracy'])

# Fitar o Modelo
history = modelo.fit(imagens_train, classes_train_cat, epochs=epocas, validation_data=(imagens_test, classes_test_cat))
clear_output()

# Exibir a Avaliação do Modelo
exibir_avaliacao(history)
```

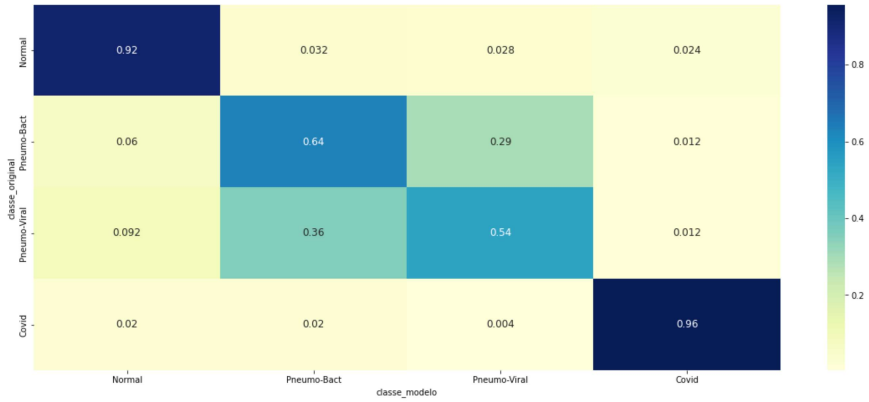
```
Acurácia na Base de Teste:  0.7534000098705291
Variabilidade na Base de Teste:  0.0157683194011401
```



Resultados sobre a base de Teste

```
exibir_pivot_teste(modelo, imagens_test, classes_test)
```

classe_original	Normal	Pneumo-Bact	Pneumo-Viral	Covid
Normal	229	8	7	6
Pneumo-Bact	15	159	73	3
Pneumo-Viral	23	89	135	3
Covid	5	5	1	239

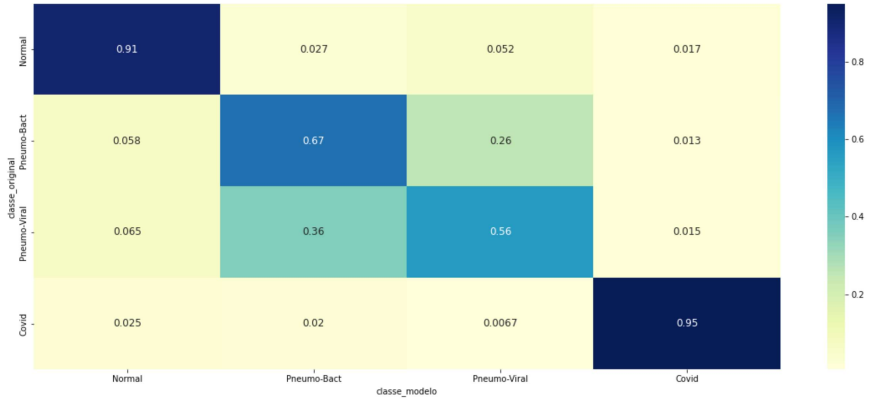


Validação

```
df_normal_validacao = get_df_previsao(modelo, pasta_normal, observacoes_normal_validacao, 0, target_width, target_height, True)
df_observacoes_pneumonia_bacterial_validacao = get_df_previsao(modelo, pasta_pneumonia_bacterial, observacoes_pneumonia_bacterial_vali
df_observacoes_pneumonia_viral_validacao = get_df_previsao(modelo, pasta_pneumonia_viral, observacoes_pneumonia_viral_validacao, 2, ta
df_observacoes_covid_19_validacao = get_df_previsao(modelo, pasta_covid_19, observacoes_covid_19_validacao, 3, target_width, target_he

exibir_pivot_validacao(pd.concat([df_normal_validacao,
                                  df_observacoes_pneumonia_bacterial_validacao,
                                  df_observacoes_pneumonia_viral_validacao,
                                  df_observacoes_covid_19_validacao]))
```

classe_original	Normal	Pneumo-Bact	Pneumo-Viral	Covid
Normal	543	16	31	10
Pneumo-Bact	35	403	154	8
Pneumo-Viral	39	214	338	9
Covid	15	12	4	569



Conclusões

O modelo tem acima de 90% de acuracidade para acertar radiografias com aspecto normal e também com Covid. Porém, para pneumonias virais e bacterianas não tem bom desempenho. Foram executadas 10 épocas mas, a partir da 2.a já percebe-se um equilibrio, ou seja, não há necessidade de se ocupar tantas épocas.

Arquitetura ResNet 50

Inicialização e Compilação

```
base_resnet = ResNet50(weights='imagenet', include_top=False, input_shape=(target_width, target_height, 3))
base_resnet.trainable = False

modelo = Sequential(base_resnet)
modelo.add(Flatten())
modelo.add(Dense(64, activation='relu'))
modelo.add(Dense(32, activation='relu'))
modelo.add(Dense(4, activation='softmax'))

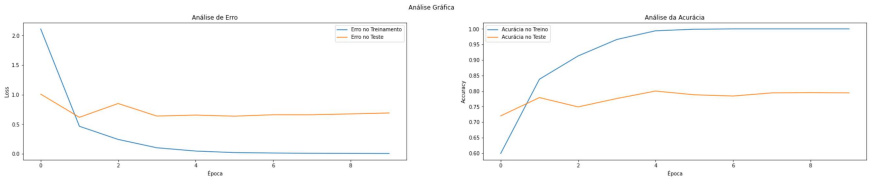
modelo.compile(loss='categorical_crossentropy', #loss=tf.losses.SparseCategoricalCrossentropy(),
               optimizer='adam',
               metrics=['accuracy'])

clear_output()

history = modelo.fit(imagens_train, classes_train_cat, epochs=epocas, validation_data=(imagens_test, classes_test_cat))
clear_output()

exibir_avaliacao(history)
```

```
Acurácia na Base de Teste:  0.7779000103473663
Variabilidade na Base de Teste:  0.023729513948254826
```



Resultados sobre a base de Teste

```
exibir_pivot_teste(modelo, imagens_test, classes_test)
```

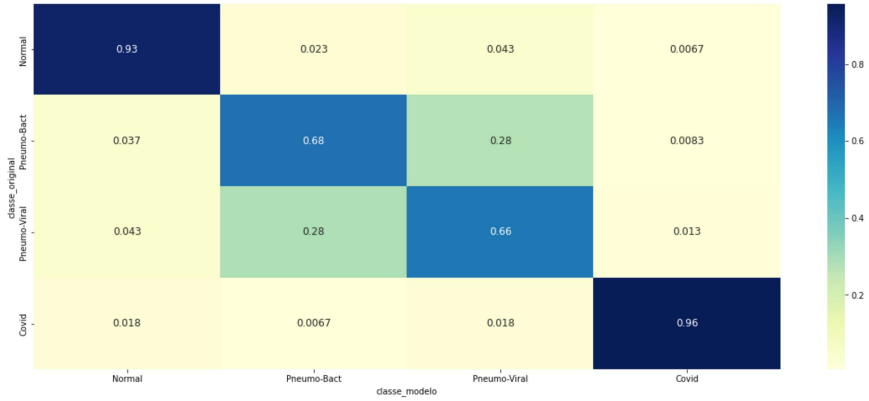
classe_original	classe_modelo	Normal	Pneumo-Bact	Pneumo-Viral	Covid
Normal		234	7	7	2

Validação

```
df_normal_validacao = get_df_previsao(modelo, pasta_normal, observacoes_normal_validacao, 0, target_width, target_height, True)
df_observacoes_pneumonia_bacterial_validacao = get_df_previsao(modelo, pasta_pneumonia_bacterial, observacoes_pneumonia_bacterial_vali
df_observacoes_pneumonia_viral_validacao = get_df_previsao(modelo, pasta_pneumonia_viral, observacoes_pneumonia_viral_validacao, 2, ta
df_observacoes_covid_19_validacao = get_df_previsao(modelo, pasta_covid_19, observacoes_covid_19_validacao, 3, target_width, target_he

exibir_pivot_validacao(pd.concat([df_normal_validacao,
                                  df_observacoes_pneumonia_bacterial_validacao,
                                  df_observacoes_pneumonia_viral_validacao,
                                  df_observacoes_covid_19_validacao]))
```

classe_original	classe_modelo	Normal	Pneumo-Bact	Pneumo-Viral	Covid
Normal		556	14	26	4
Pneumo-Bact		22	407	166	5
Pneumo-Viral		26	171	395	8
Covid		11	4	11	574



Conclusões

O modelo tem acima de 93% de acuracidade para acertar radiografias com aspecto normal e também com 96% de Covid. Da mesma forma que o VGG16, para pneumonias virais e bacterianas não tem bom desempenho. Foram executadas 10 épocas mas, a partir da 2.a já percebe-se um equilibrio, ou seja, não há necessidade de se ocupar tantas épocas.

Arquitetura Xception

Inicialização e Compilação

```
if (target_width>=71):
  base_model = Xception(weights='imagenet', include_top=False, input_shape=(target_width, target_height, 3))
  base_model.trainable = False

  modelo = Sequential(base_model)
  modelo.add(Flatten())
  modelo.add(Dense(64, activation='relu'))
  modelo.add(Dense(32, activation='relu'))
  modelo.add(Dense(4, activation='softmax'))

  modelo.compile(loss='categorical_crossentropy', #loss=tf.losses.SparseCategoricalCrossentropy(),
                 optimizer='adam',
                 metrics=['accuracy'])

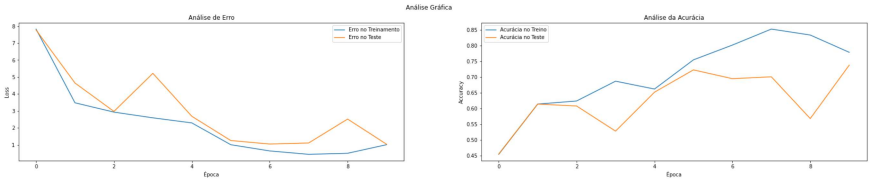
```

```
clear_output()

history = modelo.fit(imagens_train, classes_train_cat, epochs=epocas, validation_data=(imagens_test, classes_test_cat))
clear_output()

exibir_avaliacao(history)
```

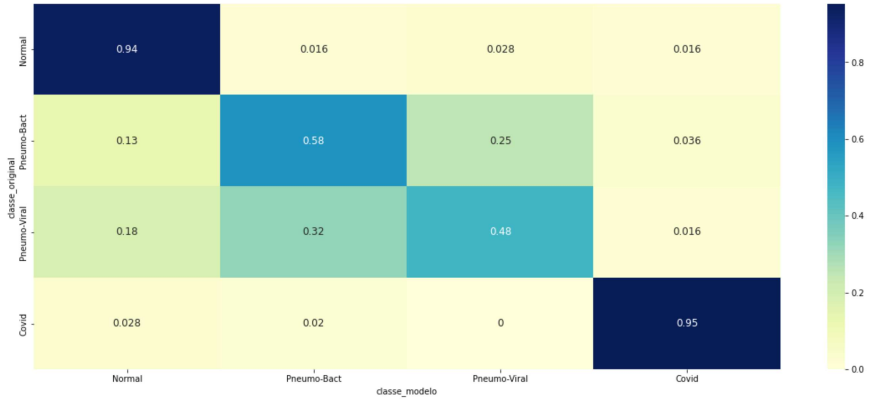
```
Acurácia na Base de Teste:  0.6281999975442887
Variabilidade na Base de Teste:  0.08697331925205011
```



Resultados sobre a base de Teste

```
if (target_width>=71):
  exibir_pivot_teste(modelo, imagens_test, classes_test)
```

classe_modelo	Normal	Pneumo-Bact	Pneumo-Viral	Covid
classe_original				
Normal	235.0	4.0	7.0	4.0
Pneumo-Bact	32.0	146.0	63.0	9.0
Pneumo-Viral	46.0	81.0	119.0	4.0
Covid	7.0	5.0	0.0	238.0

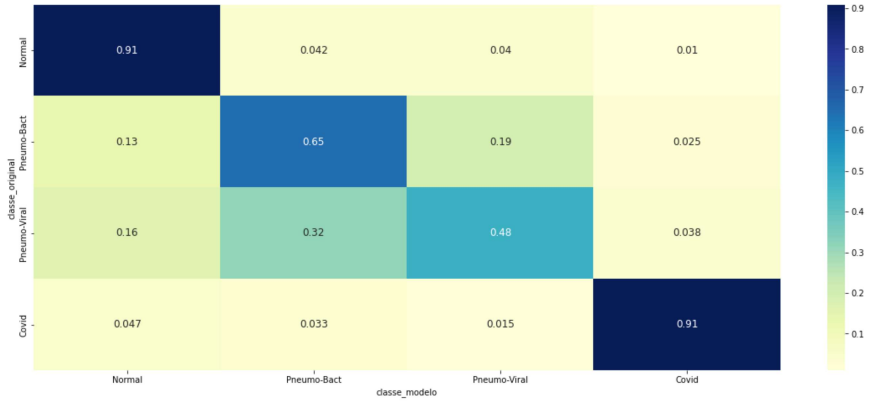


Validação

```
if (target_width>=71):
  df_normal_validacao = get_df_previsao(modelo, pasta_normal, observacoes_normal_validacao, 0, target_width, target_height, True)
  df_observacoes_pneumonia_bacterial_validacao = get_df_previsao(modelo, pasta_pneumonia_bacterial, observacoes_pneumonia_bacterial_va
  df_observacoes_pneumonia_viral_validacao = get_df_previsao(modelo, pasta_pneumonia_viral, observacoes_pneumonia_viral_validacao, 2,
  df_observacoes_covid_19_validacao = get_df_previsao(modelo, pasta_covid_19, observacoes_covid_19_validacao, 3, target_width, target_

  exibir_pivot_validacao(pd.concat([df_normal_validacao,
                                    df_observacoes_pneumonia_bacterial_validacao,
                                    df_observacoes_pneumonia_viral_validacao,
                                    df_observacoes_covid_19_validacao]))
```

classe_original \ classe_modelo	Normal	Pneumo-Bact	Pneumo-Viral	Covid
Normal	545	25	24	6
Pneumo-Bact	78	391	116	15
Pneumo-Viral	98	191	288	23
Covid	28	20	9	543



Conclusões

O modelo tem acima de 91% de acuracidade para acertar radiografias com aspecto normal e COVID. Da mesma forma que o VGG16, para pneumonias virais e bacterianas não tem bom desempenho. Da mesma fomra que as outras redes, a partir da 2.a já percebe-se um equilibrio, ou seja, não há necessidade de se ocupar tantas épocas.

Conclusão Final sobre o Experimento

Quantidade de imagens Utilizadas:

- 1000 imagens de Treinamento (250 por classe);
- 1000 imagens de Teste (250 por classe);
- 2400 imagens de Validação (600 por classe).

Redes Bases Utilizadas:

- VGG 16
- ResNet 50
- Xception

Camadas Adicionadas:

Foram mantidas as mesmas camadas, com o mesmo número de neurônios e função de ativação para justamente tentar descobrir se haveria muita diferença entre as acuracidades verificadas.

Foram executados 3 experimentos onde se observou resultados parecidos de acuracidade para cada uma das 4 classes analisadas, sendo:

- Radiografias Normal
- Radiografias com Pneumonia Viral
- Radiografias com Pneumonia Bacteriana
- Radiografias com COVID-19

Quais conclusões que chegamos:

- Os 3 modelos criados tem ótima acurácia para detectar individuos com pulmão normal e também com COVID-19;
- Manter a arquitetura inicial pode ajudar a melhorar o desempenho e o tamanho geral do modelo mas, se não for alterado a arquitetura posterior, os números praticamente se mantém.
- É importante estudar e testar várias arquiteturas mas, testar alterações em próximas camadas seja mais importante ainda.
- Apesar de acuracidade de 90% ser ótimo, este modelo está longe de ser um modelo final de decisão mas, pode ser útil para modelo de apoio inicial de decisão médica.

Não vimos necessidade de usar técnicas como *data augmentation* pois o número de amostras já é suficiente. E, como as radiografias já são extraídas num certo padrão e em ambiente controlado (sala de Raio-X), não foi verificada a necessidade de usar tal técnica.

Sugestão de trabalho futuro:

Este trabalho é apenas um esboço do que pode ser feito com redes neurais para análise de radiografias. Existem inúmeras arquiteturas que podem ser utilizadas bem como configurações que possam ser aplicadas. Pode parecer óbvio mas todo experimento deve ser acompanhado de uma telemetria bem feita. E, por último, pelo domínio do negócio ser a saúde, deve ser usado sempre o maior número de observações possível.