PSI3472 Concepção e Implementação de Sistemas Eletrônicos Inteligentes
Classificador de radiografias de pulmão em Covid-19

Nome: Igor Costa Doliveira NUSP: 11391446

Jupyter Notebook:
https://colab.research.google.com/drive/1CbxlVcqgfE5Fv_8JmNsVQA3Kxly7g2-L?usp=sharing

1. Introdução

O objetivo deste projeto é classificar imagens em categorias que incluem Covid-19, infecção viral não-Covid, infecção bacteriana não-Covid e normal (representando pacientes saudáveis). O modelo de classificação será treinado usando imagens de treinamento do banco de dados da Kaggle (**COVID-QU-Ex dataset**), que contém imagens de raio-x de pulmão em formato PNG de 256x256 pixels, representando diferentes condições médicas. As imagens são difíceis de serem classificadas visualmente, o que torna o desafio interessante.

2. Banco de dados Kaggle

Primeiramente, para obter todos os dados necessários para o treinamento do modelo tive que criar uma conta no Kaggle e criar um token de acesso remoto (API). A figura 1 ilustra a coleta do banco de dados.

Figura 1: Baixar banco de dados.

Baixar o banco de dados do kaggle

```
#https://www.geeksforgeeks.org/how-to-import-kaggle-datasets-directly-into-google-colab/
!pip3 install -q kaggle
!mkdir ~/.kaggle
!echo '{"username":"igorcostadoliveira","key":"d671763b97bcfec44bfc31f345077460"}' > ~/.kaggle/kaggle.json
!chmod 600 ~/.kaggle/kaggle.json
!kaggle datasets download anasmohammedtahir/covidqu
!unzip -u covidqu.zip
```

Já a figura 2 ilustra a quantidade de imagens neste banco de dados, será utilizada as imagens de validação/teste conjuntamente para calcular a acuracidade.

Figura 2: Quantidade de imagens de BD COVID-QU-Ex.

	Train	Val	Test	Total
Covid-19	7658	1903	2395	11956
Non-Covid	7208	1802	2253	11263
Normal	6849	1712	2140	10701
Total	21715	5417	6788	33920

A figura 3 ilustra o código para o tratamento das imagens do banco de dados para o treinamento do modelo, foi realizado. O conjunto de 33.920 imagens em escala de cinza com dimensões de 224x224 pixels ocupa 1,7 gigabytes. Dessa maneira, para que as imagens sejam acomodadas na memória do Google Colab foi necessário deixá-las com único canal de cor e usar elementos do tipo uint8.

Figura 3: Lendo banco de dados.

Função para ler imagens da lista wildcards

```
[10] def leImagens(wildcards, classes, nl, nc):
       classe = 0; i = 0
       if len(classes) == 6: n = 12205
       else: n = 21715
       AX=np.empty((n,nl,nc,1),dtype='uint8'); ay = []; # Tipo: uint8, 1 banda
       for diretorios in wildcards:
           arq = glob.glob(diretorios)
           for img in arq:
             t = image.load_img(img, color_mode='grayscale', target_size=(nl,nc))
             x = image.img_to_array(t)
             AX[i] = np.expand_dims(x, axis=0)
             ay.append(classes[classe])
             i += 1
           classe += 1
       AY = keras.utils.to_categorical(np.array(ay), 3)
       return AX, AY
```

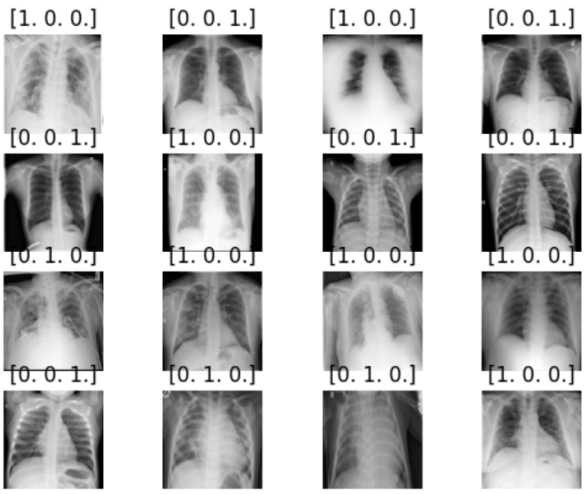
3. Classificação manual

Para avaliar a qualidade do modelo a ser gerado será classificado manualmente algumas imagens do banco de dados, para isso foi criada a função ***mostraLote()***. Ao estudar as imagens eu percebi que pulmões escuros são saudáveis, pulmões muito brancos são infecções bacterianas e pulmões intermediários eram Covid. Com esta lógica acertei 9 de 16 (**56,2%**) imagens geradas aleatoriamente.

Figura 4: Código para disponibilizar 16 imagens.

Classificação Manual

```
def mostraLote(ax, ay=None):
  for i in range(16):
      rand = random.randint(0, ax.shape[0]-1)
      plt.subplot(4, 4, i+1)
      if type(ay) != None: plt.title(f'{ay[rand]}')
      plt.imshow(ax[rand], cmap=plt.get_cmap('gray'))
      plt.axis('off')
  plt.show()
mostraLote(axb, ayb)
```



4. Modelo sem transfer learning

Para classificar as imagens de teste/validação foi escolhido modelo de rede neural inspirado na arquitetura LeNet, visto que ela foi utilizada em vários exercícios durante a disciplina. O modelo foi treinado a partir de pesos inicializados aleatoriamente e o treinamento foi realizado em apenas 16 épocas e com um tamanho de lote (batch_size) igual a 16 para economizar tempo.

O objetivo deste modelo é obter taxas de acerto no teste/validação melhores do que o obtido pela classificação manual (maior que 56,2%). Foram empregadas algumas técnicas para melhorar o desempenho do modelo de classificação, como **LearningRateScheduler**, função de ativação **Relu** e **Softmax** no treinamento do modelo e **ImageDataGenerator**, parâmetros já utilizados nos programas da disciplina.

Figura 5: Acurácia do modelo por época.

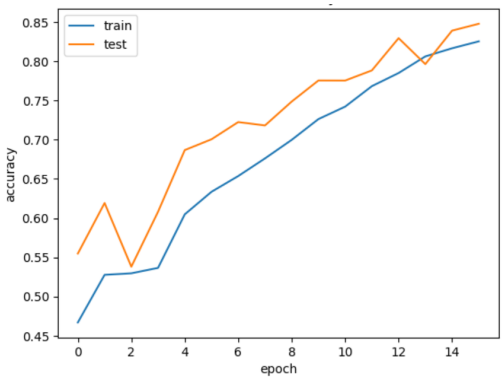


Figura 6: Resultados do modelo treinado.

```
Epoch 11/16
1358/1358 - 101s - loss: 0.6302 - accuracy: 0.7422 - val_loss: 0.5700 - val_accuracy: 0.7754 - lr: 2.0000e-04 - 101s/epoch - 74ms/step
Learning rate:  0.0002
Epoch 12/16
1358/1358 - 102s - loss: 0.5745 - accuracy: 0.7683 - val_loss: 0.5285 - val_accuracy: 0.7884 - lr: 2.0000e-04 - 102s/epoch - 75ms/step
Learning rate:  0.0002
Epoch 13/16
1358/1358 - 98s - loss: 0.5317 - accuracy: 0.7850 - val_loss: 0.4580 - val_accuracy: 0.8295 - lr: 2.0000e-04 - 98s/epoch - 72ms/step
Learning rate:  0.0002
Epoch 14/16
1358/1358 - 100s - loss: 0.4913 - accuracy: 0.8062 - val_loss: 0.5245 - val_accuracy: 0.7964 - lr: 2.0000e-04 - 100s/epoch - 74ms/step
Learning rate:  0.0002
Epoch 15/16
1358/1358 - 98s - loss: 0.4589 - accuracy: 0.8166 - val_loss: 0.4369 - val_accuracy: 0.8391 - lr: 2.0000e-04 - 98s/epoch - 72ms/step
Learning rate:  0.0002
Epoch 16/16
1358/1358 - 98s - loss: 0.4383 - accuracy: 0.8254 - val_loss: 0.4057 - val_accuracy: 0.8478 - lr: 2.0000e-04 - 98s/epoch - 72ms/step
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy', 'lr'])
```

A duração do treinamento do modelo foi de **26 minutos** aproximadamente e atingiu um acerto de treino de **82,45%** e de validação/teste de **84,78%**. Analisando a figura 5 é possível perceber que o modelo ainda não tinha estabilizado em uma acurácia limite, estava em crescimento na época 16, logo se utilizarmos mais épocas é possível que a acurácia do modelo ainda cresça mais. Entretanto, a acurácia atingida já impressiona em comparação à classificação manual (**56,2%**), visto que o modelo foi treinado durante poucas épocas.

5. Modelo com transfer learning

Para o modelo com transfer learning foi aplicada a técnica de transferência de aprendizado usando a rede neural pré-treinada **EfficientNet**, visto que ela foi classificada com as melhores performances para o desafio, em segundo lugar é a rede ResNet. O treinamento foi dividido em duas partes: 8 épocas para ajuste grosso (modelo-base congelado) e 8 épocas para ajuste fino (modelo-base descongelado), com um tamanho de lote de 16. Neste modelo as imagens foram remodeladas para 192x192 para que não estourasse a RAM da GPU.

Já o objetivo deste modelo é obter taxas de acerto no teste/validação melhores do que o obtido pela classificação manual (maior que 56,2%) e pelo modelo sem transfer learning (84,78%) em um menor tempo de treinamento, visto que utilizaremos uma rede já treinada.

Figura 7: Resultados do modelo com transfer learning treinado.

```
Epoch 2/8
1358/1358 [==============================] - 161s 118ms/step - loss: 0.2385 - accuracy: 0.9075 - val_loss: 0.1712 - val_accuracy: 0.9358
Epoch 3/8
1358/1358 [==============================] - 158s 116ms/step - loss: 0.1866 - accuracy: 0.9274 - val_loss: 0.1523 - val_accuracy: 0.9470
Epoch 4/8
1358/1358 [==============================] - 161s 118ms/step - loss: 0.1574 - accuracy: 0.9401 - val_loss: 0.1652 - val_accuracy: 0.9395
Epoch 5/8
1358/1358 [==============================] - 157s 116ms/step - loss: 0.1253 - accuracy: 0.9507 - val_loss: 0.1563 - val_accuracy: 0.9449
Epoch 6/8
1358/1358 [==============================] - 157s 116ms/step - loss: 0.1029 - accuracy: 0.9595 - val_loss: 0.2342 - val_accuracy: 0.9141
Epoch 7/8
1358/1358 [==============================] - 163s 120ms/step - loss: 0.0873 - accuracy: 0.9657 - val_loss: 0.2036 - val_accuracy: 0.9381
Epoch 8/8
1358/1358 [==============================] - 160s 118ms/step - loss: 0.0714 - accuracy: 0.9731 - val_loss: 0.1864 - val_accuracy: 0.9435
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

A duração do treinamento foi de aproximadamente **30 minutos** e atingiu um acerto de treino de **97,31%** e de validação/teste de **94,35%**, como é possível visualizar na figura 7. É notável como o desempenho desta rede supera substancialmente o desempenho obtido sem o uso de transfer learning, usando o mesmo tempo de treinamento.

6. Conclusão

Com os resultados dos modelos foi possível perceber como o modelo treinado com transfer learning é significativamente melhor do que um modelo treinado sem transfer learning por várias razões, **economizando tempo e recursos** computacionais, uma vez que a parte inicial da rede já aprendeu a extrair características úteis de dados semelhantes, uma **melhor inicialização** e **capacidade de generalização** resultando em modelos mais eficazes e eficientes.

Portanto, é possível concluir que o objetivo do projeto foi alcançado, foram criados dois modelos capazes de classificar imagens em categorias que incluem Covid-19, infecção viral não-Covid, infecção bacteriana não-Covid e normal. É possível melhorar ainda mais os modelos treinados ao aumentar a quantidade de épocas do treinamento.