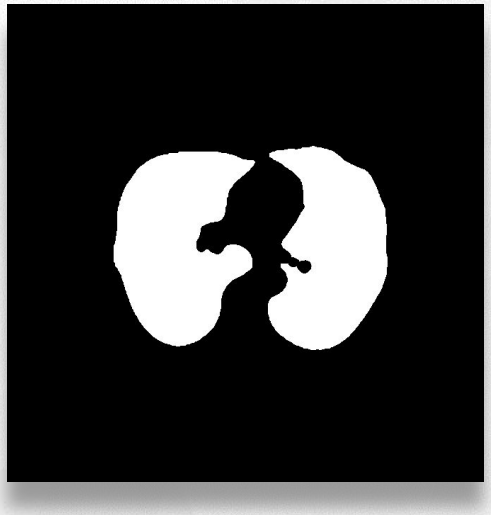
YANA S. PEREIRA

SEGMENTAÇÃO DE IMAGENS DE TOMOGRAFIA COMPUTADORIZADA PULMONAR ATRAVÉS DA CNN U-NET

Orientação: Prof. Dr. Anderson Meneses e Prof. Me. Davi Guimarães

Sumário

1.1. Banco de Dados

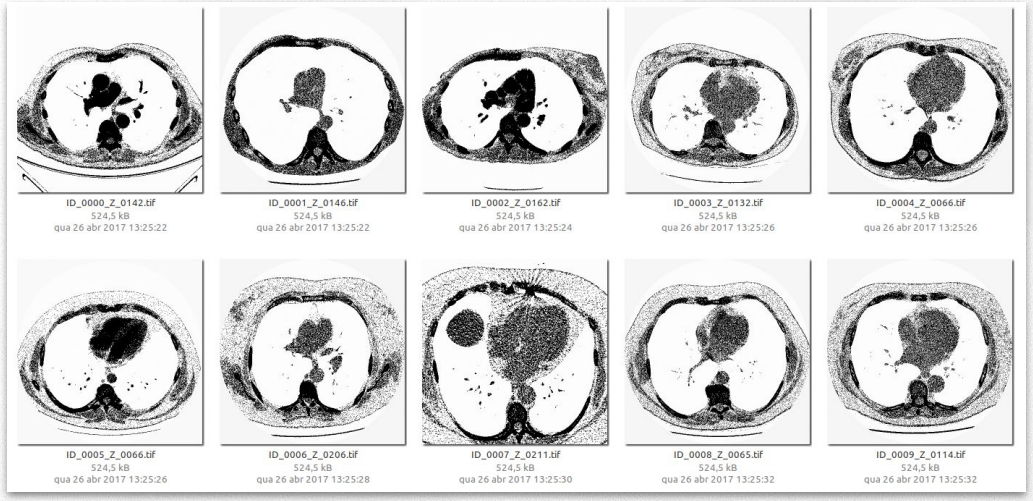


- Banco de Dados: **Finding and Measuring Lungs in CT Data**

Disponível em: <https://www.kaggle.com/datasets/kmader/finding-lungs-in-ct-data?select=2d_masks.zip>

1.2. Organização inicial dos dados

- Imagens 2D (*.tif*) - 267 itens



- Máscaras 2D (*.tif*) - 267 itens

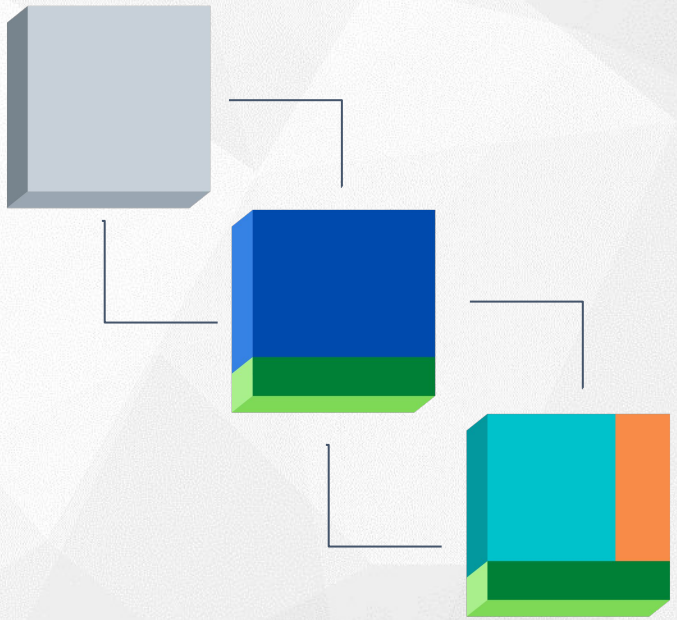
2.1. Importando bibliotecas

- Os;
- Numpy;
- Pandas;
- Cv2;
- Matplotlib;
- Sklearn;
- TensorFlow;
- Keras.

```
import os
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import cv2
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

from keras import *
from keras.models import Model
from keras.layers import *
from tensorflow.keras.optimizers import Adam
from keras.regularizers import l2
from keras.preprocessing.image import ImageDataGenerator
import keras.backend as K
from keras.callbacks import LearningRateScheduler, ModelCheckpoint
```

2.2. Divisão geral dos dados

- Divisão dos dados



Dados de Treino e Validação (80% = 214)

Dados de Teste (20% = 53)

Dados de Treino (80%*80% = 171)

Dados de Validação (20%*80% = 43)

2.3. Importando dados (treino e validação)

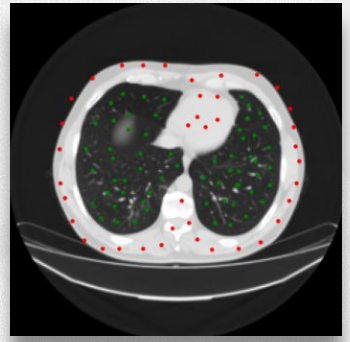
- Criação de variáveis para o conteúdo dos diretórios

```
#Dados de treino e validação
IMAGE_LIB = '/path/2d_images/'
MASK_LIB = '/path/2d_masks/'
```

```
#Dados de teste
IMAGE_TEST_LIB = '/path/2d_images_test/'
MASK_TEST_LIB = '/path/2d_masks_test/'
```

- Determinação das dimensões das imagens e do *seed*

```
IMG_HEIGHT, IMG_WIDTH = 32, 32
SEED=42
```



2.3. Importando dados (treino e validação)

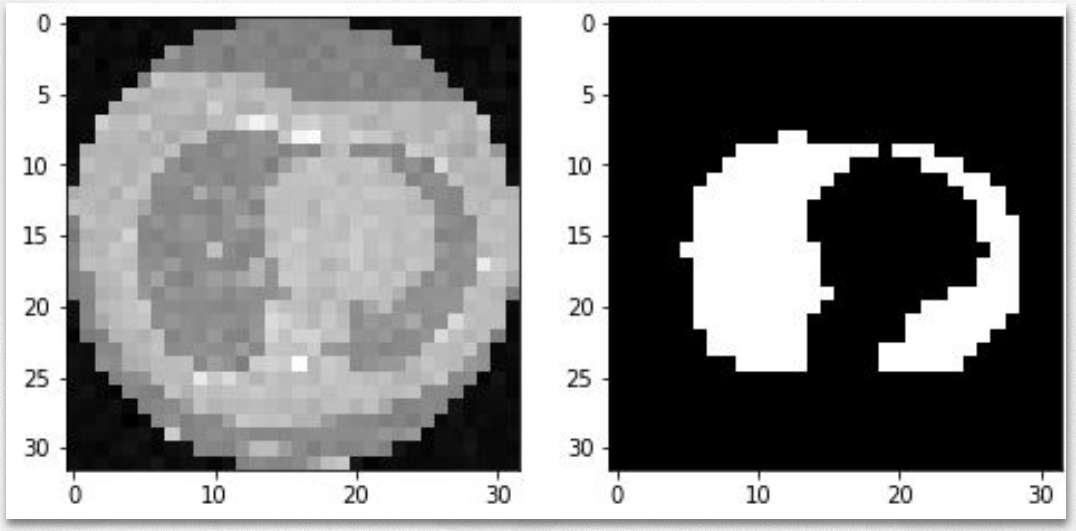
- Mudança dos nomes dos arquivos e redimensionamento dos elementos, salvando-os em matrizes

```
all_images = [x for x in sorted(os.listdir(IMAGE_LIB)) if x[-4:] == '.tif']


x_data = np.empty((len(all_images), IMG_HEIGHT, IMG_WIDTH), dtype='float32')
for i, name in enumerate(all_images):
    im = cv2.imread(IMAGE_LIB + name, cv2.IMREAD_UNCHANGED).astype("int16").astype('float32')
    im = cv2.resize(im, dsize=(IMG_WIDTH, IMG_HEIGHT), interpolation=cv2.INTER_LANCZOS4)
    im = (im - np.min(im)) / (np.max(im) - np.min(im))
    x_data[i] = im


y_data = np.empty((len(all_images), IMG_HEIGHT, IMG_WIDTH), dtype='float32')
for i, name in enumerate(all_images):
    im = cv2.imread(MASK_LIB + name, cv2.IMREAD_UNCHANGED).astype('float32')/255.
    im = cv2.resize(im, dsize=(IMG_WIDTH, IMG_HEIGHT), interpolation=cv2.INTER_NEAREST)
    y_data[i] = im
```

2.4. Verificação dos dados

- Exibindo a primeira imagem e a primeira máscara

```
fig, ax = plt.subplots(1,2, figsize = (8,4))
ax[0].imshow(x_data[0], cmap='gray')
ax[1].imshow(y_data[0], cmap='gray')
plt.show()
```



2.5. Organização dos dados

- Redimensionamento das matrizes

```
print('\nMatriz original: ', x_data.shape)

x_data = x_data[:,:,:,np.newaxis] #redimensiona a matriz
y_data = y_data[:,:,:,np.newaxis]

print('\nMatriz redimensionada: ', x_data.shape)
```

```
Matriz original:  (214, 32, 32)

Matriz redimensionada:  (214, 32, 32, 1)
```

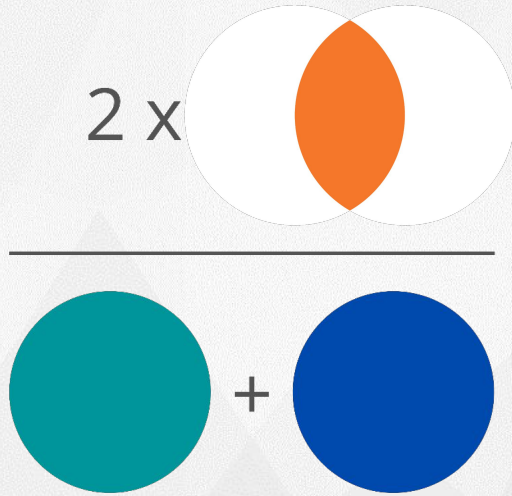
- Dados de treino (80%) e dados de validação (20%)

```
x_train, x_val, y_train, y_val = train_test_split(x_data, y_data, test_size = 0.2)

print('\nDados de Treino: ', x_train.shape)
print('\nDados de Validação: ', x_val.shape)
```

```
Dados de Treino:  (171, 32, 32, 1)

Dados de Validação:  (43, 32, 32, 1)
```

2.6. Coeficiente Dice (Sørensen–Dice coefficient)

```python
def dice_coef(y_true, y_pred):
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = K.sum(y_true_f * y_pred_f)
    return (2. * intersection + \
        K.epsilon()) / (K.sum(y_true_f) + K.sum(y_pred_f) + K.epsilon())
```



Máscara Predita

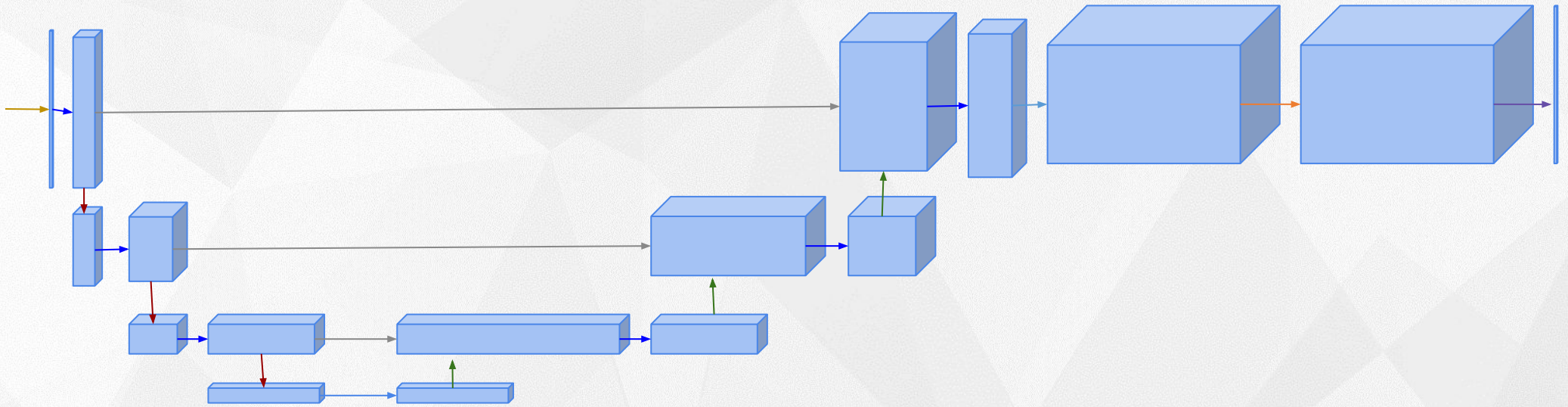
Interseção

Máscara Verdadeira

2.7. Criando o modelo

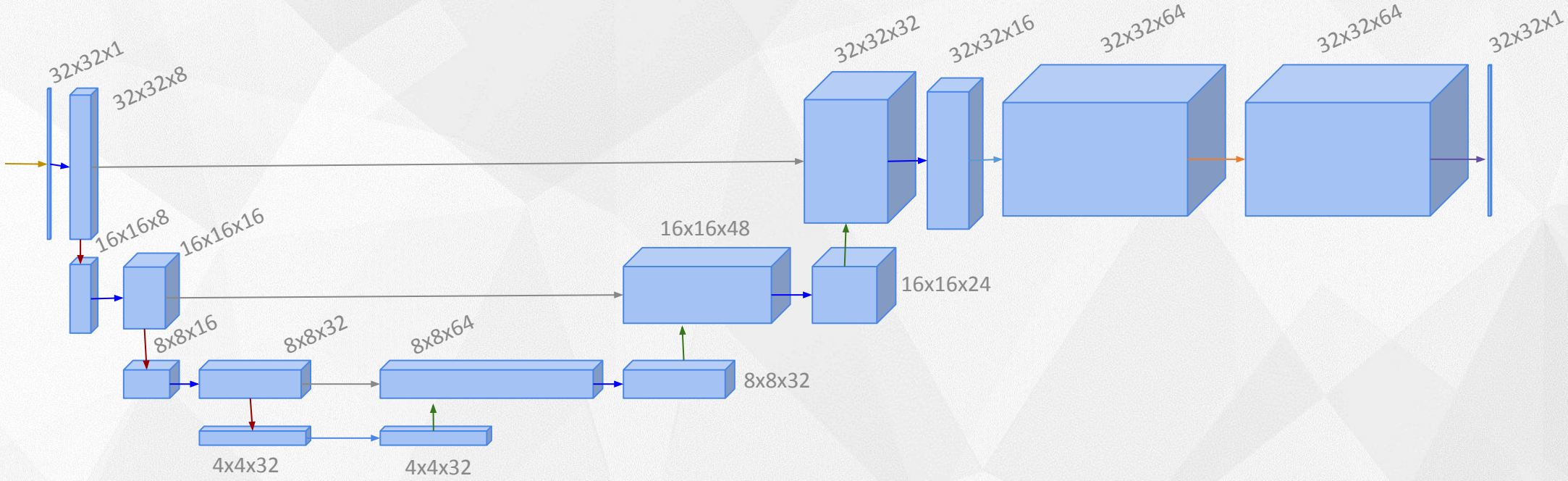
```
input_layer = Input(shape=x_train.shape[1:]) #somente dimensões

c1 = Conv2D(filters=8, kernel_size=(3,3), activation='relu', padding='same')(input_layer)
l = MaxPool2D(strides=(2,2))(c1)
c2 = Conv2D(filters=16, kernel_size=(3,3), activation='relu', padding='same')(l)
l = MaxPool2D(strides=(2,2))(c2)
c3 = Conv2D(filters=32, kernel_size=(3,3), activation='relu', padding='same')(l)
l = MaxPool2D(strides=(2,2))(c3)
c4 = Conv2D(filters=32, kernel_size=(1,1), activation='relu', padding='same')(l)

l = concatenate([UpSampling2D(size=(2,2))(c4), c3], axis=-1)
l = Conv2D(filters=32, kernel_size=(2,2), activation='relu', padding='same')(l)
l = concatenate([UpSampling2D(size=(2,2))(l), c2], axis=-1)
l = Conv2D(filters=24, kernel_size=(2,2), activation='relu', padding='same')(l)
l = concatenate([UpSampling2D(size=(2,2))(l), c1], axis=-1)
l = Conv2D(filters=16, kernel_size=(2,2), activation='relu', padding='same')(l)
l = Conv2D(filters=64, kernel_size=(1,1), activation='relu')(l)
l = Dropout(0.5)(l)

output_layer = Conv2D(filters=1, kernel_size=(1,1), activation='sigmoid')(l)

model = Model(input_layer, output_layer)
```

2.8. Representação do modelo



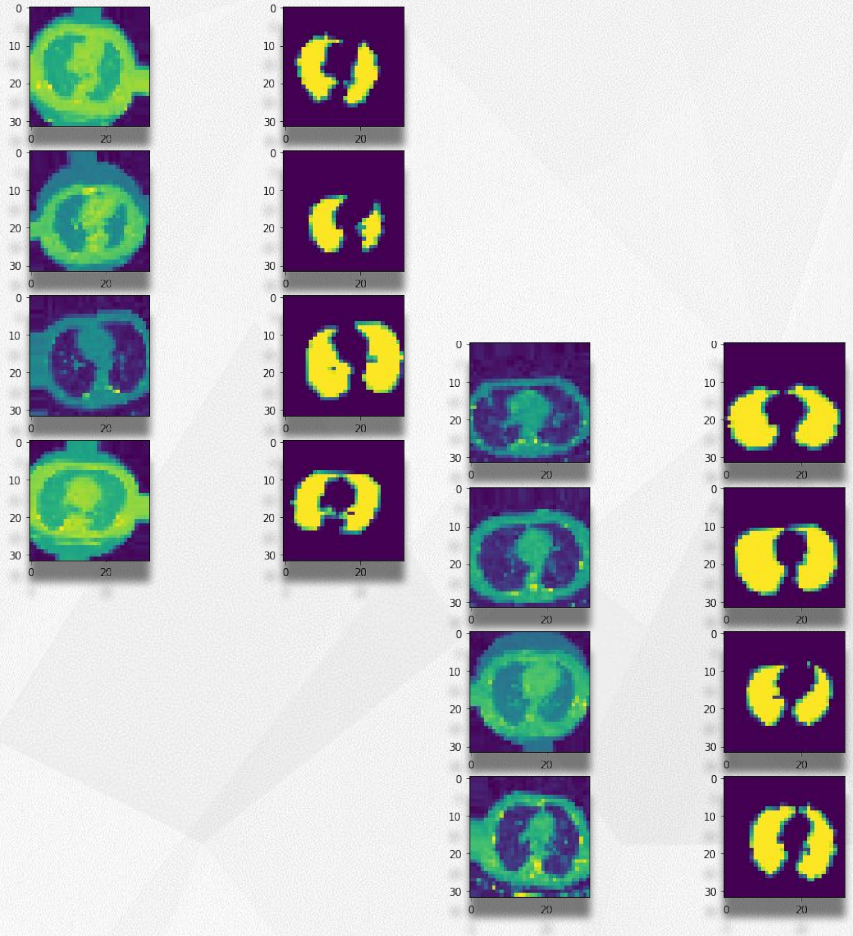
Input
Max Pool 2x2
Copy and Crop
Dropout
Conv 2x2, ReLU
Conv 1x1, ReLU
UpConv 2x2
Output

2.8. Representação do modelo



2.9. Data Augmentation

```python
def my_generator(x_train, y_train, batch_size):
    data_generator = ImageDataGenerator(
            width_shift_range=0.1,
            height_shift_range=0.1,
            rotation_range=10,
            zoom_range=0.1).flow(x_train, x_train, batch_size, seed=SEED)
    mask_generator = ImageDataGenerator(
            width_shift_range=0.1,
            height_shift_range=0.1,
            rotation_range=10,
            zoom_range=0.1).flow(y_train, y_train, batch_size, seed=SEED)
    while True:
        x_batch, _ = data_generator.next()
        y_batch, _ = mask_generator.next()
        yield x_batch, y_batch
```



2.10. Compilação e Treinamento

- 2.9.1. Compilação

```
model.compile(optimizer=Adam(2e-4), \
    loss='binary_crossentropy', metrics=[dice_coef])
```

- 2.9.2. Salvando o melhor modelo (pesos)

```
weight_saver = ModelCheckpoint('lung.h5',\
    monitor='val_dice_coef', save_best_only=True, \
        save_weights_only=True)
```

- 2.9.3. Valor atualizado da taxa de aprendizado (optmizer)

```
newLearningRate = LearningRateScheduler(lambda x: 1e-3 * 0.8 ** x)
```

- 2.9.4. Treinamento

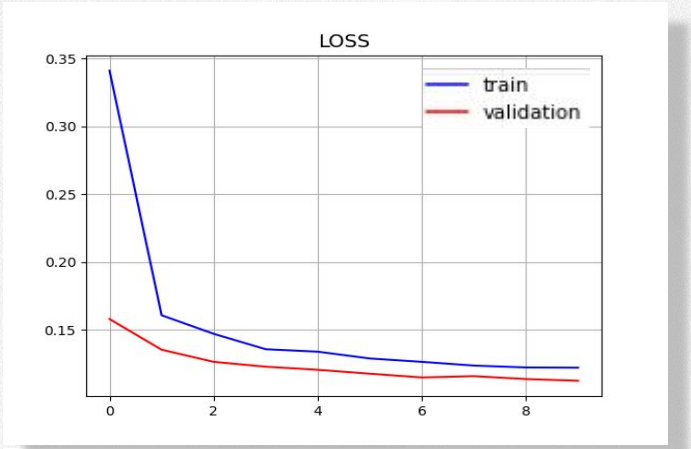
```
hist = model.fit_generator(my_generator(x_train, y_train, 8),
    steps_per_epoch = 200,
    validation_data = (x_val, y_val),
    epochs=10, verbose=2,
    callbacks = [weight_saver, newLearningRate])
```

- 2.9.5. Carregando os melhores pesos

```
model.load_weights('lung.h5')
```

3. Análise de desempenho: treino e validação

```
Epoch 1/10
200/200 - 29s - loss: 0.3410 - dice_coef: 0.5340 - val_loss: 0.1581 - val_dice_coef: 0.7886
Epoch 2/10
200/200 - 21s - loss: 0.1609 - dice_coef: 0.7981 - val_loss: 0.1355 - val_dice_coef: 0.8353
Epoch 3/10
200/200 - 26s - loss: 0.1473 - dice_coef: 0.8200 - val_loss: 0.1266 - val_dice_coef: 0.8357
Epoch 4/10
200/200 - 27s - loss: 0.1359 - dice_coef: 0.8334 - val_loss: 0.1230 - val_dice_coef: 0.8519
Epoch 5/10
200/200 - 27s - loss: 0.1341 - dice_coef: 0.8389 - val_loss: 0.1207 - val_dice_coef: 0.8510
Epoch 6/10
200/200 - 25s - loss: 0.1291 - dice_coef: 0.8415 - val_loss: 0.1179 - val_dice_coef: 0.8588
Epoch 7/10
200/200 - 27s - loss: 0.1266 - dice_coef: 0.8458 - val_loss: 0.1151 - val_dice_coef: 0.8575
Epoch 8/10
200/200 - 21s - loss: 0.1239 - dice_coef: 0.8474 - val_loss: 0.1161 - val_dice_coef: 0.8574
Epoch 9/10
200/200 - 28s - loss: 0.1225 - dice_coef: 0.8506 - val_loss: 0.1139 - val_dice_coef: 0.8596
Epoch 10/10
200/200 - 25s - loss: 0.1223 - dice_coef: 0.8512 - val_loss: 0.1127 - val_dice_coef: 0.8581
```



4.1. Importando dados de teste

- Mudança dos nomes dos arquivos e redimensionamento dos elementos, salvando-os em matrizes

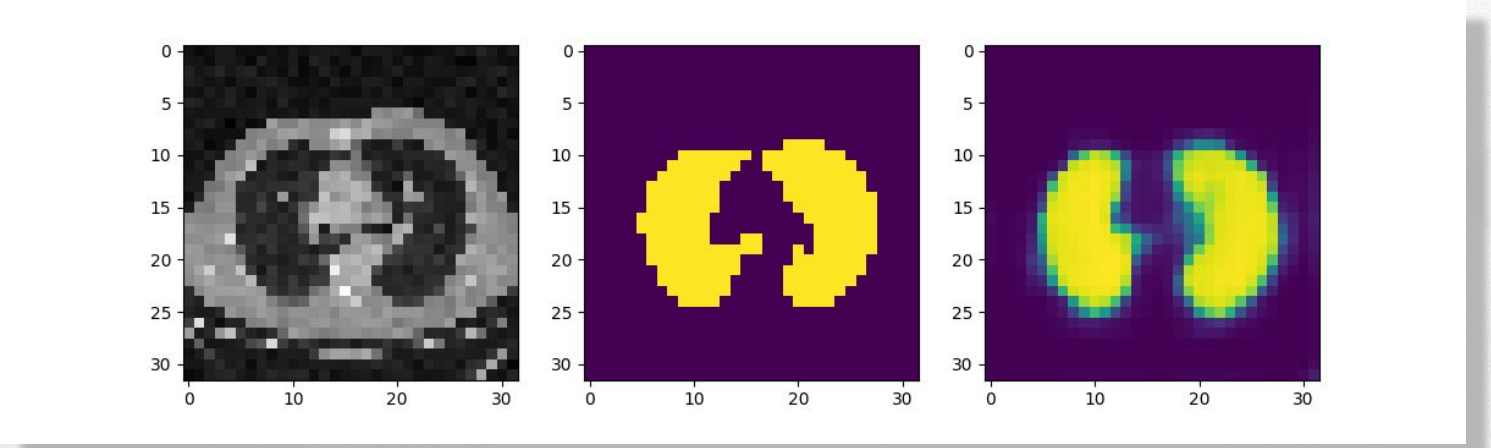
```
all_test_images = [x for x in sorted(os.listdir(IMAGE_TEST_LIB)) if x[-4:] == '.tif']
x_test = np.empty((len(all_test_images), IMG_HEIGHT, IMG_WIDTH), dtype='float32')

for i, name in enumerate(all_test_images):
    im = cv2.imread(IMAGE_TEST_LIB + name, cv2.IMREAD_UNCHANGED).astype("int16").astype('float32')
    im = cv2.resize(im, dsize=(IMG_WIDTH, IMG_HEIGHT), interpolation=cv2.INTER_LANCZOS4)
    im = (im - np.min(im)) / (np.max(im) - np.min(im))
    x_test[i] = im


y_test = np.empty((len(all_test_images), IMG_HEIGHT, IMG_WIDTH), dtype='float32')
for i, name in enumerate(all_test_images):
    im = cv2.imread(MASK_TEST_LIB + name, cv2.IMREAD_UNCHANGED).astype('float32')/255.
    im = cv2.resize(im, dsize=(IMG_WIDTH, IMG_HEIGHT), interpolation=cv2.INTER_NEAREST)
    y_test[i] = im

x_test = x_test[:,:,:,np.newaxis] #redimensiona a matriz
y_test = y_test[:,:,:,np.newaxis]
```

4.2. Predição

- Realizando a predição a partir dos dados de teste

```
y_hat = model.predict(x_test)
```

- Primeira imagem de teste, primeira máscara de teste, predição do modelo para primeira imagem

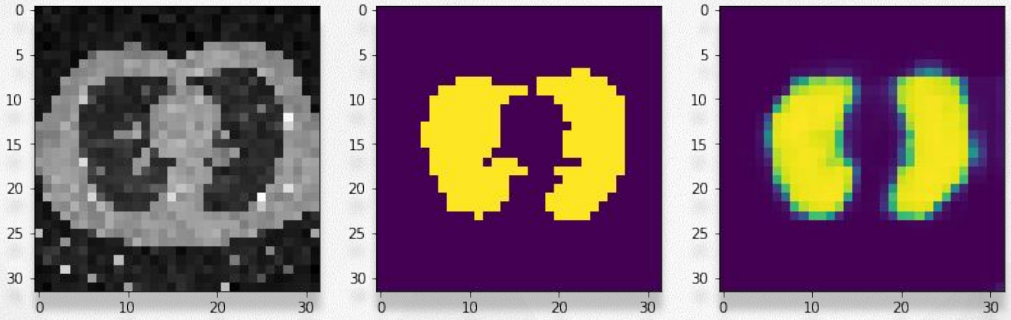


5. Análise de desempenho: teste

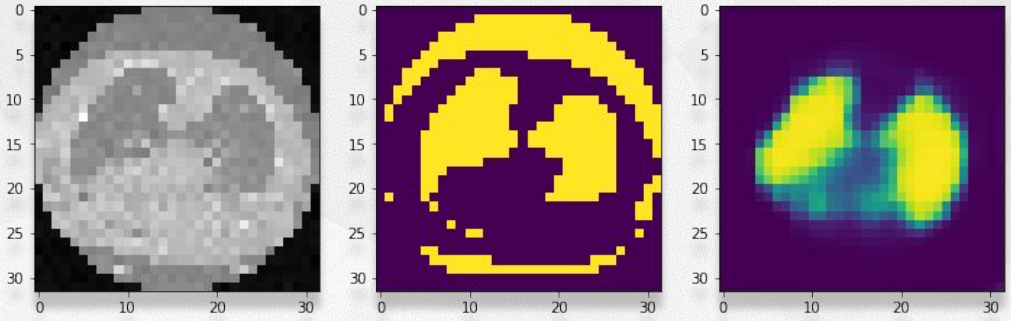
	Values
Tomo 1	0.859420
Tomo 2	0.841888
Tomo 3	0.872676
...	
Tomo 51	0.863713
Tomo 52	0.825393
Tomo 53	0.522851

- Média: 0.83
- Mediana: 0.84
- Desvio-Padrão: 0.06
- Valor Máximo: 0.89
- Valor Mínimo: 0.52

- Maior Dice



- Menor Dice



3. Conclusão

- Importância da segmentação de imagens biomédicas
- Contribuição do Deep Learning às futuras aplicações médicas
- U-net e a capacidade de treino com pequenos volumes de dados

4. Referências

A Lung CT In Keras. **Kaggle**. Disponível em: <https://www.kaggle.com/code/toregil/a-lung-u-net-in-keras/notebook>. Acesso em: 11 de Maio de 2022.

Image Augmentation using Keras ImageGenerator. **Analytics Vidhya**, 2020. Disponível em: <https://www.analyticsvidhya.com/blog/2020/08/image-augmentation-on-the-fly-using-keras-imagedatagenerator/>. Acesso em: 13 de Maio de 2022.

ModelCheckPoint. **Keras**. Disponível em: <https://keras.io/api/callbacks/model_checkpoint/>. Acesso em: 15 de Maio de 2022.

Dice Coefficient, IOU. **Medium**, 2019. Disponível em: <https://medium.com/@karan_jakhar/100-days-of-code-day-7-84e4918cb72c>. Acesso em: 19 de Maio de 2022.

TOMAR, N. What is Unet?. **Medium**, 2021. Disponível em: <https://medium.com/analytics-vidhya/what-is-unet-157314c87634>. Acesso em 02 de Junho de 2022.

ARAÚJO, A. R. et al. Segmentação das áreas pulmonares em radiografias torácicas digitais. Anais do Simpósio Brasileiro de Computação Aplicada à Saúde, 2021. Niterói. Disponível em: <https://sol.sbc.org.br/index.php/sbcas/article/view/16079>. Acesso em 02 de Junho de 2022.

BRUZADIN, A. J. Segmentação Interativa de Imagens de Tomografia Computadorizada Pulmonar com Covid-19 via Aprendizado Profundo. Repositório Unesp, 2021. São José do Rio Preto. Disponível em: <https://repositorio.unesp.br/bitstream/handle/11449/211039/bruzadin_aj_me_sjrp.pdf?sequence=5&isAllowed=y>. Acesso em 02 de Junho de 2022.