Atlântico Avanti

Equipe 2 - Segmentação de pulmão

Apresentar métodos da literatura e propor ideias para métodos próprios.

Alunos:

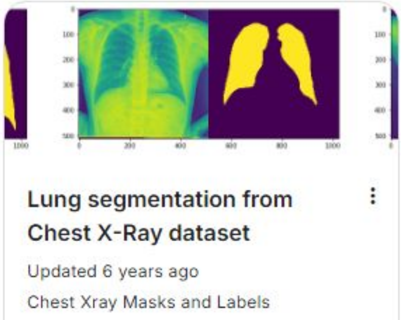
Raphael
Luan
Ygor
Thiago

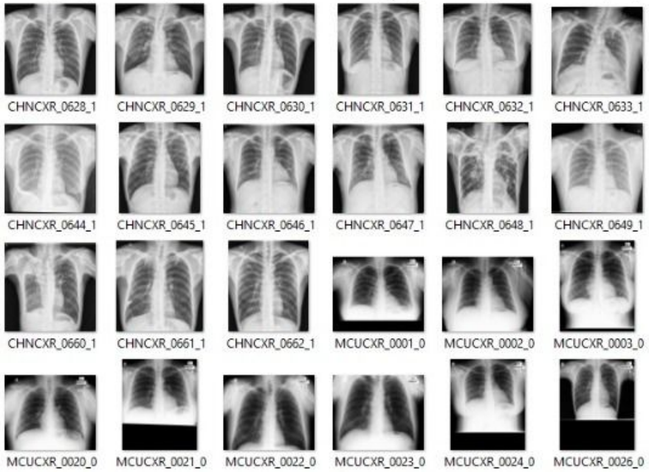
METODOLOGI
A

Análise dos notebooks

O notebook escolhido então, foi o Lung segmentation form Chest X-Ray dataset do usuário Nikhil panday. Por ter sido executado com sucesso e entendido por grande parte dos membros do grupo. Nele vamos tratar de pontos chave do dataset como:

- O conjunto de dados é composto por imagens e máscara segmentada de duas fontes diferentes(Shenzhen, Montgomery).
- Há uma ligeira anormalidade na convenção de nomenclatura das máscaras.
- Algumas imagens não possuem suas máscaras correspondentes.





Métodos da literatura - Etapas seguidas no codigo

- Arquitetura do Modelo: U-Net
- Funções de Perda e Métricas
- Treinamento do Modelo
- Callbacks para Treinamento
- Avaliação e Visualização dos Resultados

Métodos da literatura

```python
# we have 704 masks but 800 images. Hence we are going to
# make a 1-1 correspondance from mask to images, not the usual other way.
images = os.listdir(image_path)
mask = os.listdir(mask_path)
mask = [fName.split(".png")[0] for fName in mask]
image_file_name = [fName.split("_mask")[0] for fName in mask]
```

```python
check = [i for i in mask if "mask" in i]
print("Total mask that has modified name:",len(check))
```

```
Total mask that has modified name: 566
```

Checando quantas máscaras tem o nome alterado.

CHNCXR_0652_1_mask.png
CHNCXR_0653_1_mask.png
CHNCXR_0654_1_mask.png
CHNCXR_0655_1_mask.png
CHNCXR_0656_1_mask.png
CHNCXR_0657_1_mask.png
CHNCXR_0658_1_mask.png
CHNCXR_0659_1_mask.png
CHNCXR_0660_1_mask.png
CHNCXR_0661_1_mask.png
CHNCXR_0662_1_mask.png
MCUCXR_0001_0.png
MCUCXR_0002_0.png
MCUCXR_0003_0.png
MCUCXR_0004_0.png
MCUCXR_0005_0.png
MCUCXR_0006_0.png
MCUCXR_0008_0.png
MCUCXR_0011_0.png
MCUCXR_0013_0.png

Métodos da literatura

- função para automatizar o resize das imagens, que funciona tanto para treino quanto para teste

```
testing_files = set(os.listdir(image_path)) & set(os.listdir(mask_path))
training_files = check

def getData(X_shape, flag = "test"):
    im_array = []
    mask_array = []

    if flag == "test":
        for i in tqdm(testing_files):
            im = cv2.resize(cv2.imread(os.path.join(image_path,i)),(X_shape,X_shape))[:,:,0]
            mask = cv2.resize(cv2.imread(os.path.join(mask_path,i)),(X_shape,X_shape))[:,:,0]

            im_array.append(im)
            mask_array.append(mask)

        return im_array,mask_array

    if flag == "train":
        for i in tqdm(training_files):
            im = cv2.resize(cv2.imread(os.path.join(image_path,i.split("_mask")[0]+".png")),(X_shape,X_shape))[:,:,0]
            mask = cv2.resize(cv2.imread(os.path.join(mask_path,i+".png")),(X_shape,X_shape))[:,:,0]

            im_array.append(im)
            mask_array.append(mask)

        return im_array,mask_array
```

Métodos da literatura

Processo da CNN usada (U-net)

```
def unet(input_size=(256,256,1)):
    inputs = Input(input_size)

    conv1 = Conv2D(32, (3, 3), activation='relu', padding='same')(inputs)
    conv1 = Conv2D(32, (3, 3), activation='relu', padding='same')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)

    conv2 = Conv2D(64, (3, 3), activation='relu', padding='same')(pool1)
    conv2 = Conv2D(64, (3, 3), activation='relu', padding='same')(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)

    conv3 = Conv2D(128, (3, 3), activation='relu', padding='same')(pool2)
    conv3 = Conv2D(128, (3, 3), activation='relu', padding='same')(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)

    conv4 = Conv2D(256, (3, 3), activation='relu', padding='same')(pool3)
    conv4 = Conv2D(256, (3, 3), activation='relu', padding='same')(conv4)
    pool4 = MaxPooling2D(pool_size=(2, 2))(conv4)

    conv5 = Conv2D(512, (3, 3), activation='relu', padding='same')(pool4)
    conv5 = Conv2D(512, (3, 3), activation='relu', padding='same')(conv5)

    up6 = concatenate([Conv2DTranspose(256, (2, 2), strides=(2, 2), padding='same')(conv5), conv4], axis=3)
    conv6 = Conv2D(256, (3, 3), activation='relu', padding='same')(up6)
    conv6 = Conv2D(256, (3, 3), activation='relu', padding='same')(conv6)

    up7 = concatenate([Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same')(conv6), conv3], axis=3)
    conv7 = Conv2D(128, (3, 3), activation='relu', padding='same')(up7)
    conv7 = Conv2D(128, (3, 3), activation='relu', padding='same')(conv7)
```

```
up8 = concatenate([Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same')(conv7), conv2], axis=3)
conv8 = Conv2D(64, (3, 3), activation='relu', padding='same')(up8)
conv8 = Conv2D(64, (3, 3), activation='relu', padding='same')(conv8)

up9 = concatenate([Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same')(conv8), conv1], axis=3)
conv9 = Conv2D(32, (3, 3), activation='relu', padding='same')(up9)
conv9 = Conv2D(32, (3, 3), activation='relu', padding='same')(conv9)

conv10 = Conv2D(1, (1, 1), activation='sigmoid')(conv9)

return Model(inputs=[inputs], outputs=[conv10])
```

Métodos da literatura

```
from IPython.display import clear_output
from keras.optimizers import Adam
from sklearn.model_selection import train_test_split

model.compile(optimizer=Adam(lr=2e-4),
              loss=[dice_coef_loss],
           metrics = [dice_coef, 'binary_accuracy'])

train_vol, validation_vol, train_seg, validation_seg = train_test_split((images-127.0)/127.0,
                                                            (mask>127).astype(np.float32),
                                                            test_size = 0.1,random_state = 2018)

train_vol, test_vol, train_seg, test_seg = train_test_split(train_vol,train_seg,
                                                            test_size = 0.1,
                                                            random_state = 2018)

loss_history = model.fit(x = train_vol,
                       y = train_seg,
                         batch_size = 16,
                  epochs = 50,
                  validation_data =(test_vol,test_seg) ,
                  callbacks=callbacks_list)
```

- Processo de treinamento do modelo.
- divisão treino-teste.
- tamanho do lote.

Métodos da literatura

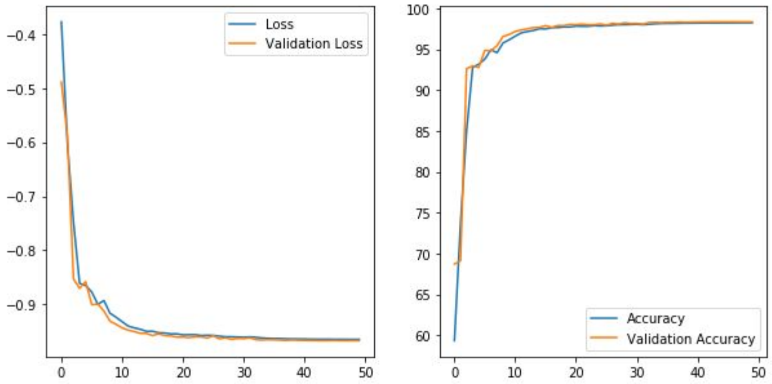
- Validação dos resultados.

- relação entre as métricas e o número de épocas.

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (10, 5))
ax1.plot(loss_history.history['loss'], '-', label = 'Loss')
ax1.plot(loss_history.history['val_loss'], '-', label = 'Validation Loss')
ax1.legend()

ax2.plot(100*np.array(loss_history.history['binary_accuracy']), '-',
         label = 'Accuracy')
ax2.plot(100*np.array(loss_history.history['val_binary_accuracy']), '-',
         label = 'Validation Accuracy')
ax2.legend()
```

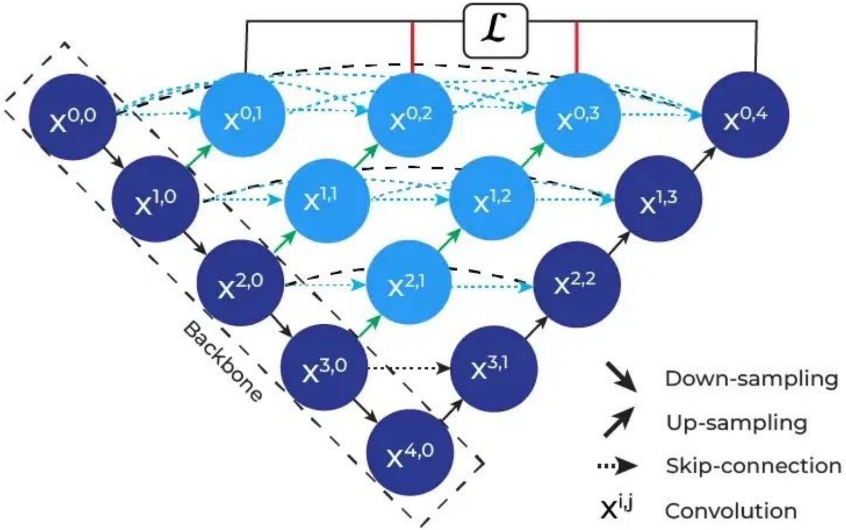
```
<matplotlib.legend.Legend at 0x7daa1c1d4be0>
```



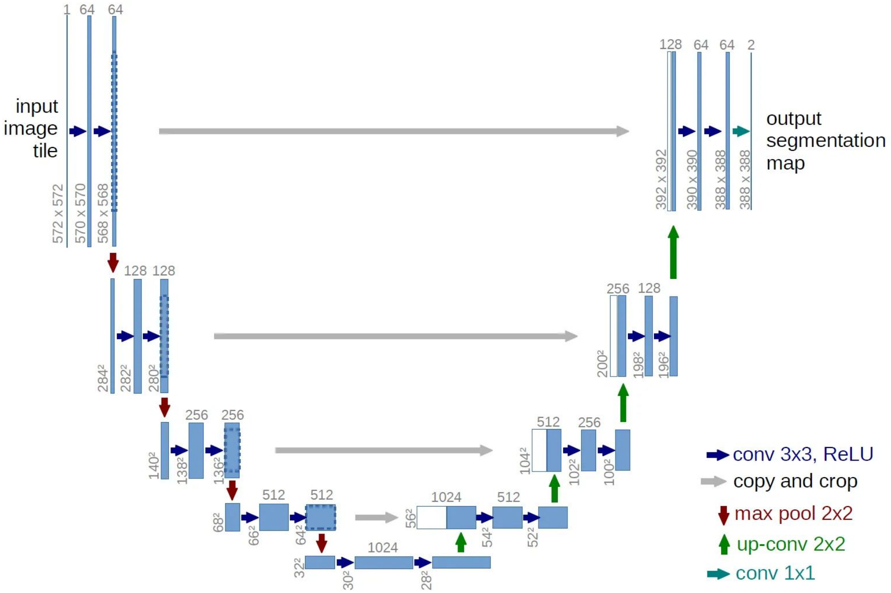
Métodos Próprios – Unet++

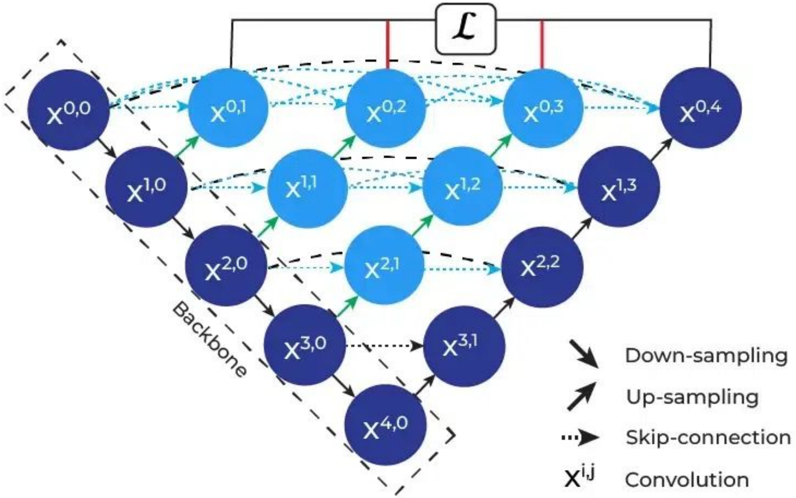
A UNet++ adiciona blocos de convolução densamente conectados entre o encoder e o decoder. Esses blocos ajudam a reduzir a diferença semântica entre os mapas de características do encoder e decoder, tornando a tarefa de aprendizado mais fácil e eficiente.

Incorpora uma supervisão profunda, onde saídas intermediárias são geradas em diferentes níveis da rede. Isso ajuda a treinar a rede de forma mais eficaz, fornecendo feedback em várias etapas do processo de segmentação



Métodos Próprios – Unet vs Unet++(Nested unet)





Codigo da Unet++

```python
# Definindo o bloco convolucional
def conv_block(inputs, num_filters):
    # Applying the sequence of Convolutional, Batch Normalization
    # and Activation Layers to the input tensor
    x = tf.keras.Sequential([
        # Convolutional Layer
        tf.keras.layers.Conv2D(num_filters, 3, padding='same'),
        # Batch Normalization Layer
        tf.keras.layers.BatchNormalization(),
        # Activation Layer
        tf.keras.layers.Activation('relu'),
        # Convolutional Layer
        tf.keras.layers.Conv2D(num_filters, 3, padding='same'),
        # Batch Normalization Layer
        tf.keras.layers.BatchNormalization(),
        # Activation Layer
        tf.keras.layers.Activation('relu')
    ])(inputs)

    # Returning the output of the Convolutional Block
    return x
```

```python
# Defining the Unet++ Model
def unet_plus_plus_model(input_shape=(256, 256, 3), num_classes=1, deep_supervision=True):
    inputs = tf.keras.layers.Input(shape=input_shape)

    # Caminho de Codificação
    x_00 = conv_block(inputs, 32)  # Reduzido de 64 para 32
    x_10 = conv_block(tf.keras.layers.MaxPooling2D()(x_00), 64)  # Reduzido de 128 para 64
    x_20 = conv_block(tf.keras.layers.MaxPooling2D()(x_10), 128)  # Reduzido de 256 para 128
    x_30 = conv_block(tf.keras.layers.MaxPooling2D()(x_20), 256)  # Reduzido de 512 para 256
    x_40 = conv_block(tf.keras.layers.MaxPooling2D()(x_30), 512)  # Reduzido de 1024 para 512

    # Caminho de Decodificação Aninhado
    x_01 = conv_block(tf.keras.layers.concatenate([x_00, tf.keras.layers.UpSampling2D()(x_10)]), 32)
    x_11 = conv_block(tf.keras.layers.concatenate([x_10, tf.keras.layers.UpSampling2D()(x_20)]), 64)
    x_21 = conv_block(tf.keras.layers.concatenate([x_20, tf.keras.layers.UpSampling2D()(x_30)]), 128)
    x_31 = conv_block(tf.keras.layers.concatenate([x_30, tf.keras.layers.UpSampling2D()(x_40)]), 256)

    x_02 = conv_block(tf.keras.layers.concatenate([x_00, x_01, tf.keras.layers.UpSampling2D()(x_11)]), 32)
    x_12 = conv_block(tf.keras.layers.concatenate([x_10, x_11, tf.keras.layers.UpSampling2D()(x_21)]), 64)
    x_22 = conv_block(tf.keras.layers.concatenate([x_20, x_21, tf.keras.layers.UpSampling2D()(x_31)]), 128)

    x_03 = conv_block(tf.keras.layers.concatenate([x_00, x_01, x_02, tf.keras.layers.UpSampling2D()(x_12)]), 32)
    x_13 = conv_block(tf.keras.layers.concatenate([x_10, x_11, x_12, tf.keras.layers.UpSampling2D()(x_22)]), 64)

    x_04 = conv_block(tf.keras.layers.concatenate([x_00, x_01, x_02, x_03, tf.keras.layers.UpSampling2D()(x_13)]), 32)

    # Caminho de Supervisão Profunda
    if deep_supervision:
        outputs = [
            tf.keras.layers.Conv2D(num_classes, 1)(x_01),
            tf.keras.layers.Conv2D(num_classes, 1)(x_02),
            tf.keras.layers.Conv2D(num_classes, 1)(x_03),
            tf.keras.layers.Conv2D(num_classes, 1)(x_04)
        ]
        outputs = tf.keras.layers.concatenate(outputs, axis=0)
    else:
        outputs = tf.keras.layers.Conv2D(num_classes, 1)(x_04)

    model = tf.keras.Model(inputs=inputs, outputs=outputs, name='Unet_plus_plus')
    return model
```

RESULTADOS

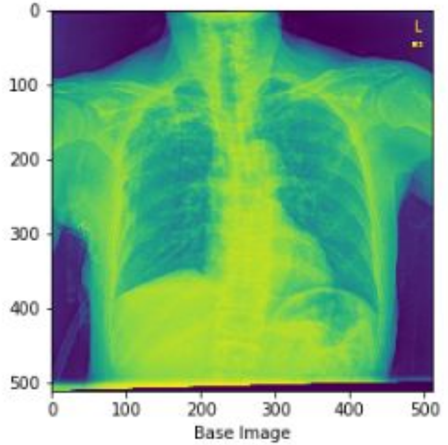
Atlântico Avanti

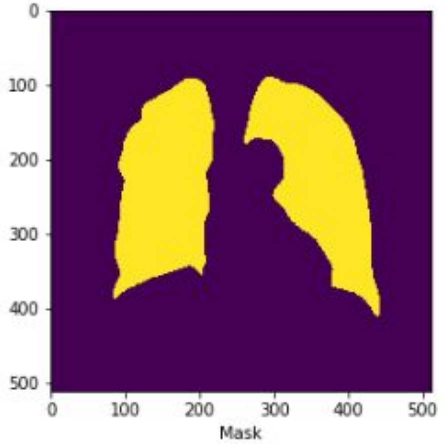
Métricas de avaliação da segmentação

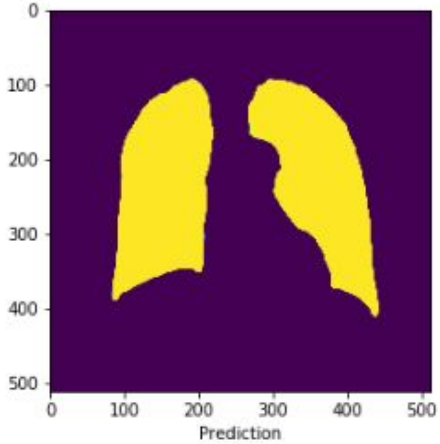
- **Dice Similarity Coefficient (DSC)**
- **Fitness Adjust**
- **Size Adjust**
- **Position Adjust**
- **Intersection over Union**

Métrica de avaliação – Literatura Unet

```
Imagem 1:
Dice Similarity: 0.9357612220812024
Fitness Adjust:  0.9342025793157352
Size Adjust:     0.4011253932970372
Position Adjust: 0.9928071527780167
IoU:             0.25087983236171185
```

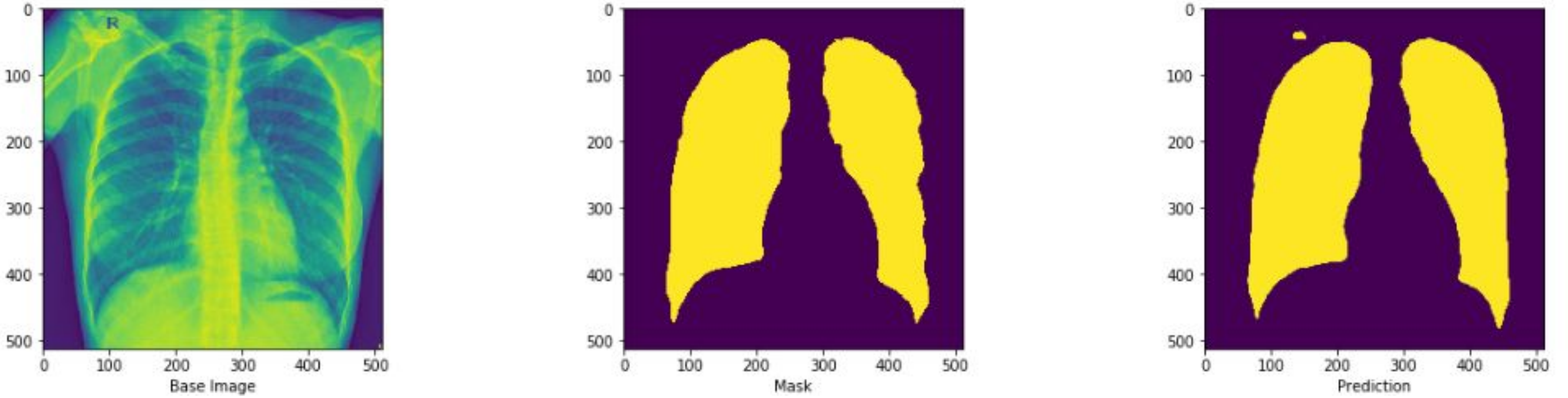






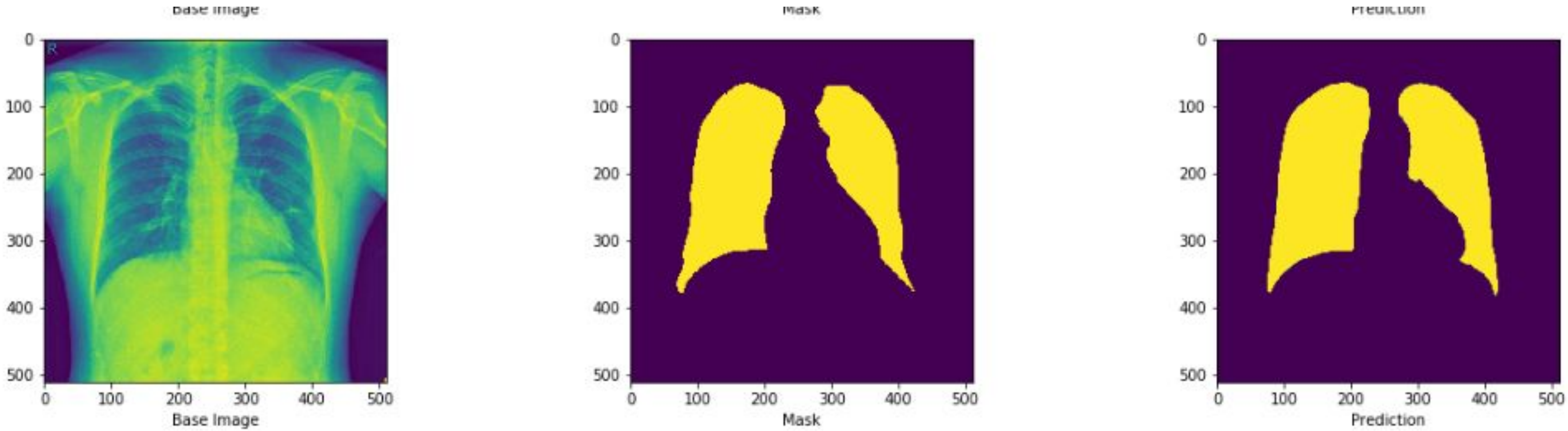
Métrica de avaliação - Literatura Unet

```
Imagem 2:
Dice Similarity: 0.9484382813590854
Fitness Adjust:  0.9425177831453454
Size Adjust:     0.5380276152019721
Position Adjust: 0.9855194835895658
IoU:             0.36801489603806764
```



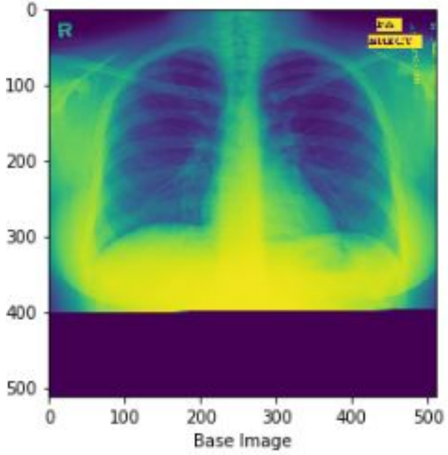
Métrica de avaliação – Literatura Unet

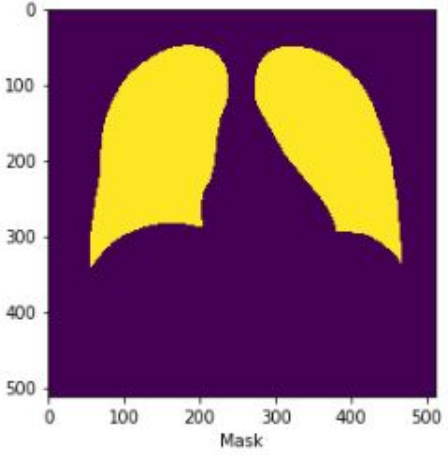
```
Imagem 3:
Dice Similarity: 0.9310765776696781
Fitness Adjust:  0.9251929080520704
Size Adjust:     0.3324653939175629
Position Adjust: 0.9487397242692571
IoU:             0.19937540888499378
```

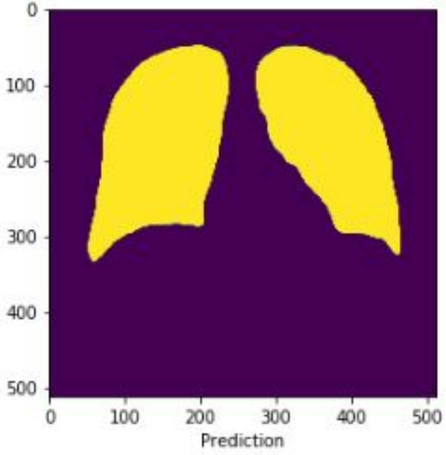


Métrica de avaliação - Literatura Unet

```
Imagem 4:
Dice Similarity: 0.9306262602829745
Fitness Adjust:  0.9365160214854603
Size Adjust:     0.4799601840743726
Position Adjust: 0.9660816248486026
IoU:             0.3157550078924091
```

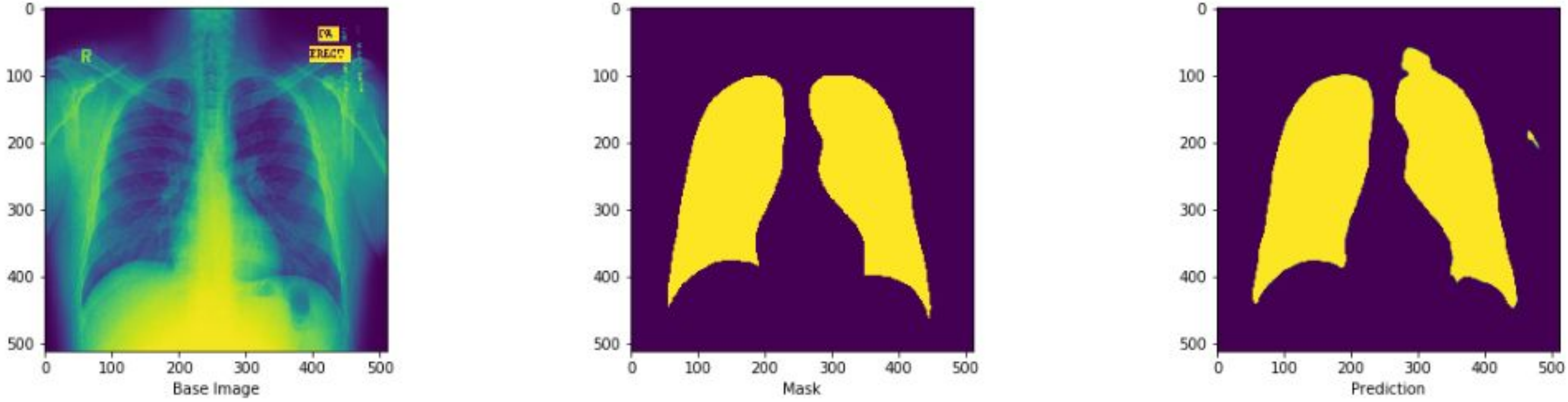






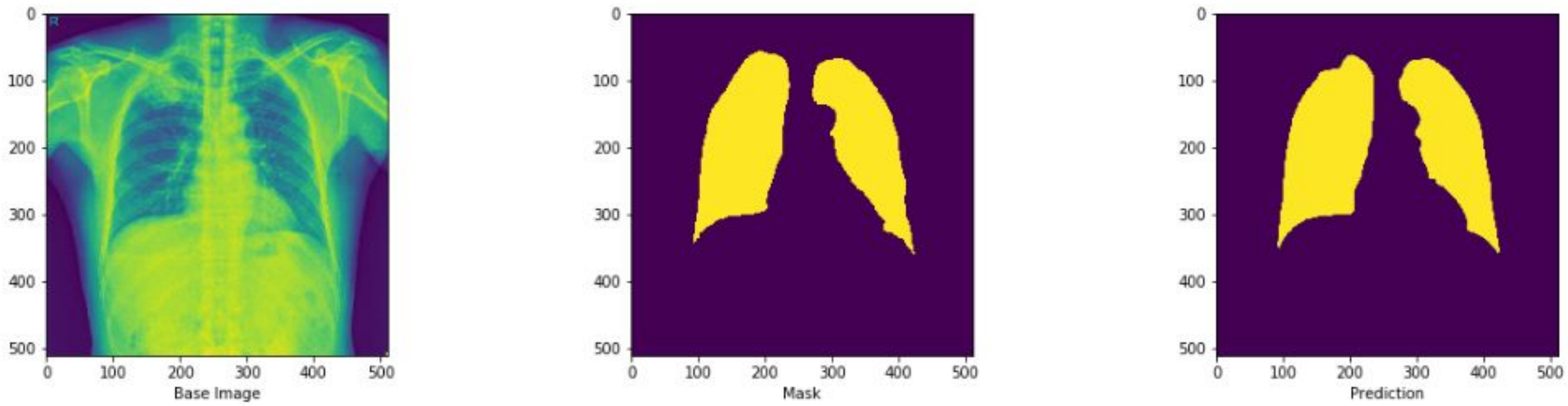
Métrica de avaliação - Literatura Unet

```
Imagem 5:
Dice Similarity: 0.9230360623612662
Fitness Adjust:  0.9302496752778179
Size Adjust:     0.43573730541760647
Position Adjust: 0.991927416515231
IoU:             0.27855762777359716
```



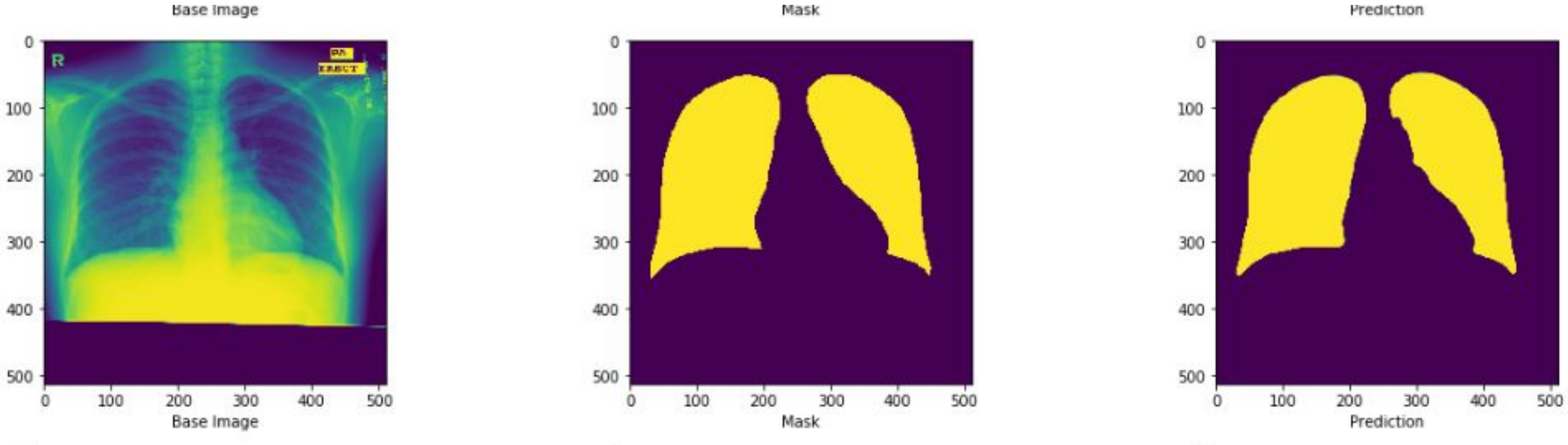
Métrica de avaliação - Literatura Unet

```
Imagem 6:
Dice Similarity: 0.9269391379529145
Fitness Adjust:  0.9184827347168667
Size Adjust:     0.32121482551756864
Position Adjust: 0.953214822082111
IoU:             0.19133765915975462
```



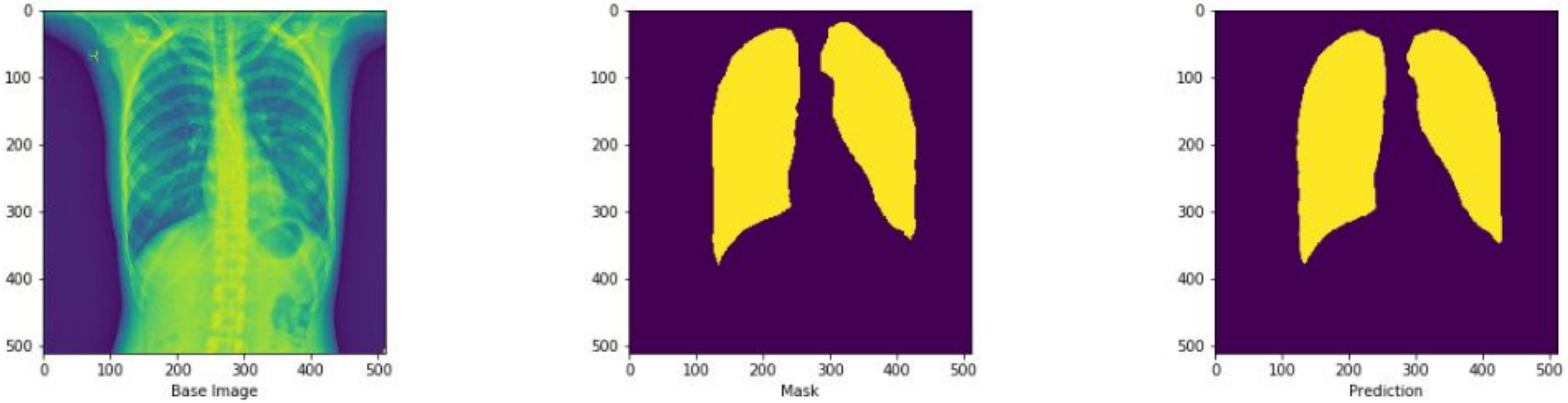
Métrica de avaliação - Literatura Unet

```
Imagem 7:
Dice Similarity: 0.9298272744155568
Fitness Adjust:  0.9282373640916132
Size Adjust:     0.48444791180587965
Position Adjust: 0.9564734451678248
IoU:             0.3196511130034013
```



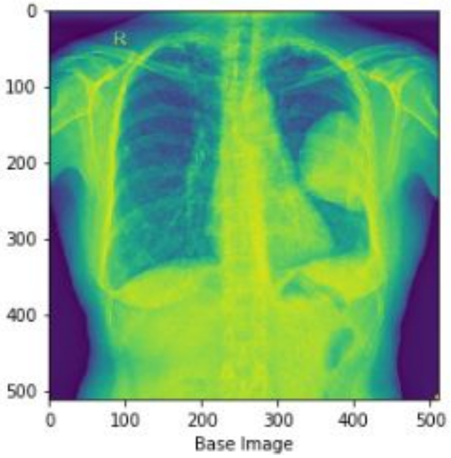
Métrica de avaliação – Literatura Unet

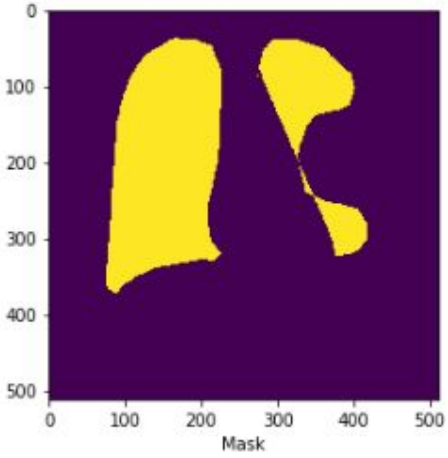
```
Imagem 8:
Dice Similarity: 0.9427260465933863
Fitness Adjust:  0.939779644815616
Size Adjust:     0.4304192475193044
Position Adjust: 0.949304728287733
IoU:             0.2742256152409069
```

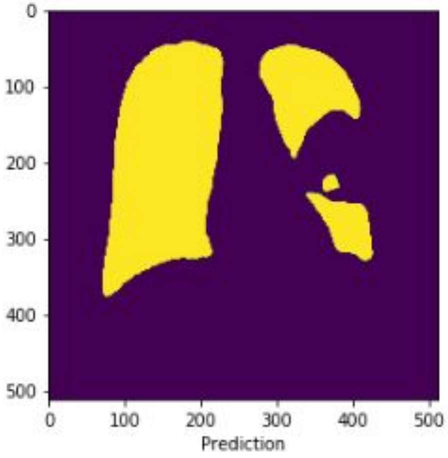


Métrica de avaliação - Literatura Unet

```
Imagem 9:
Dice Similarity: 0.9043988719527856
Fitness Adjust:  0.8862543401971889
Size Adjust:     0.3544034766075915
Position Adjust: 0.9221062296553819
IoU:             0.21536474559206428
```

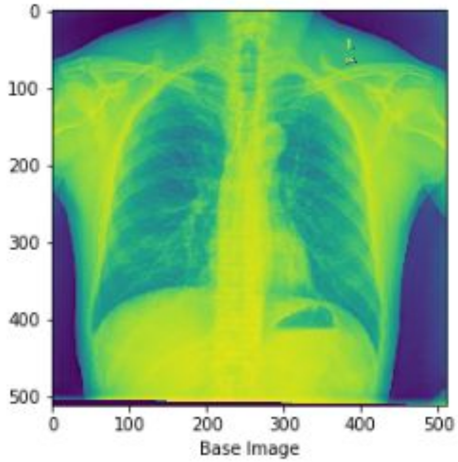


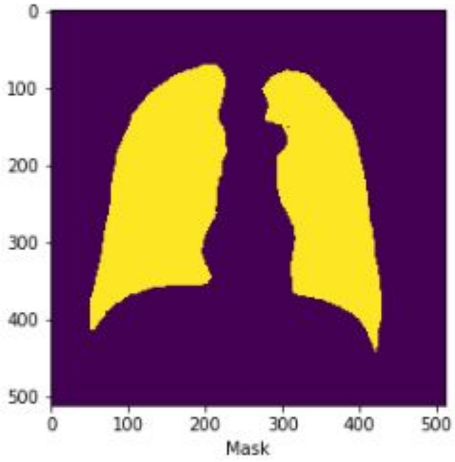


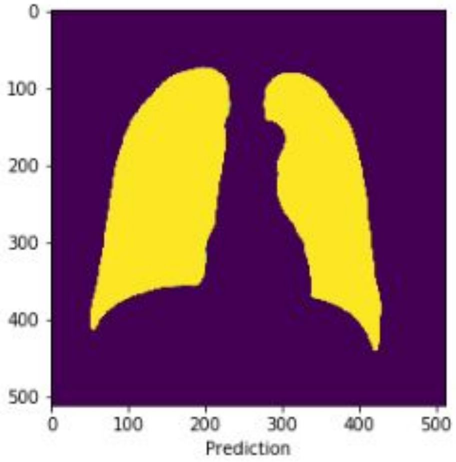


Métrica de avaliação - Literatura Unet

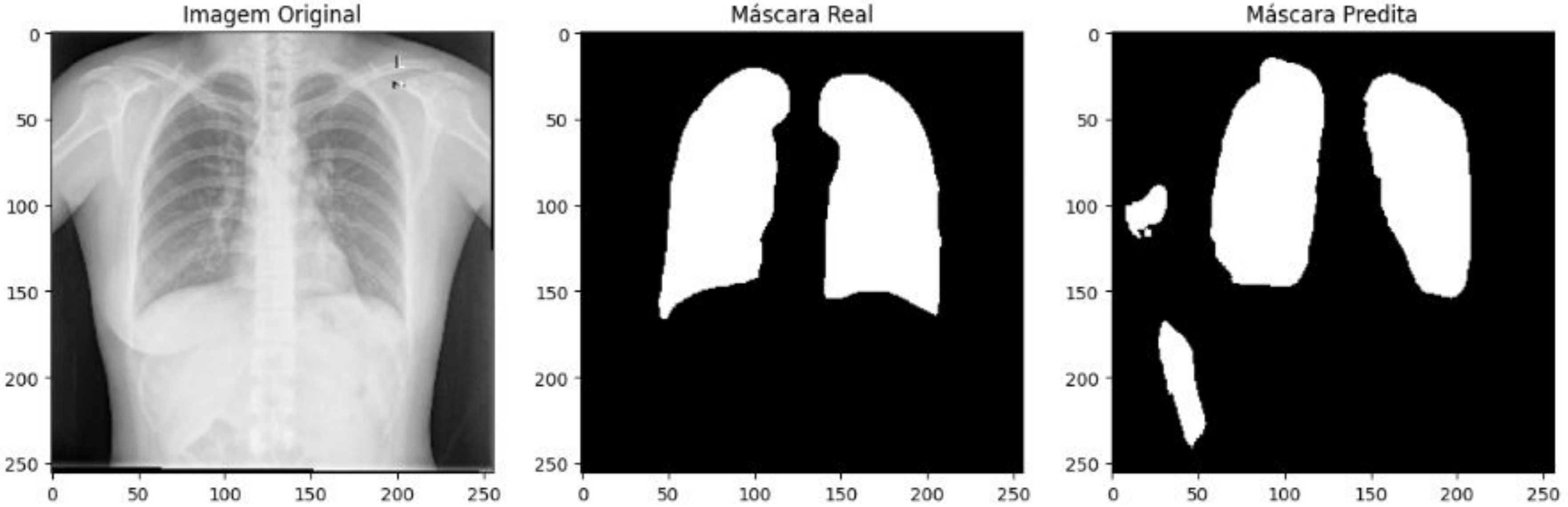
```
Imagem 10:
Dice Similarity: 0.9273947067378991
Fitness Adjust:  0.9085282080205938
Size Adjust:     0.453685700302898
Position Adjust: 0.9876239915983196
IoU:             0.29339811472465055
```





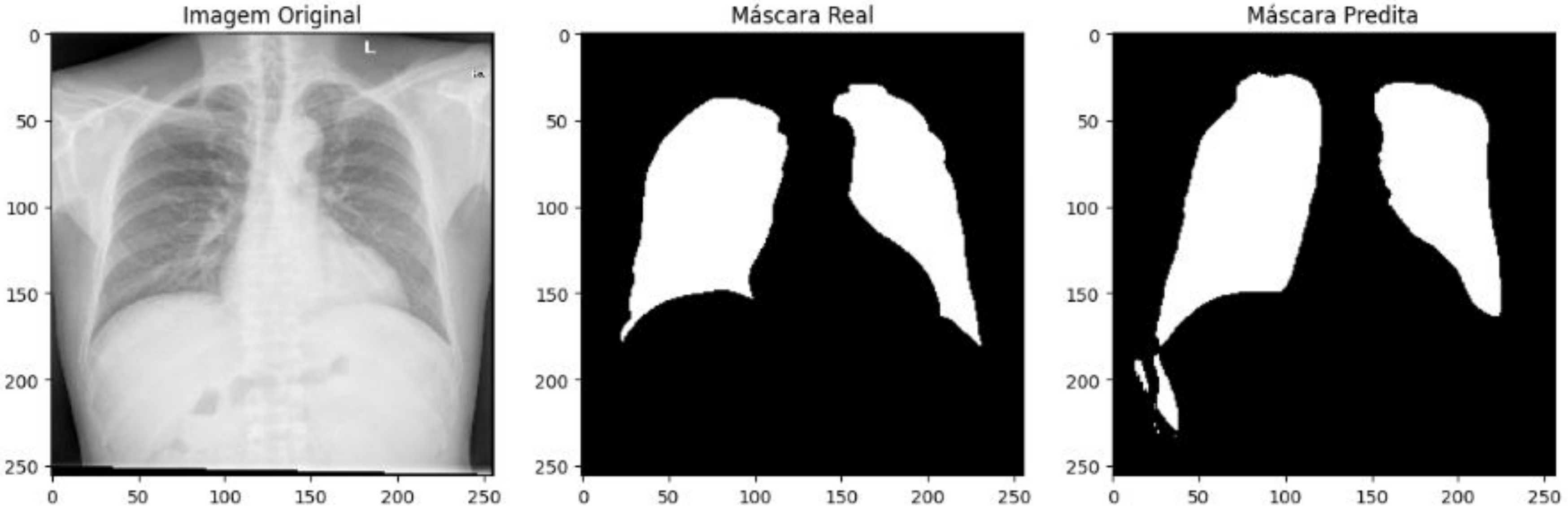


Métrica de avaliação



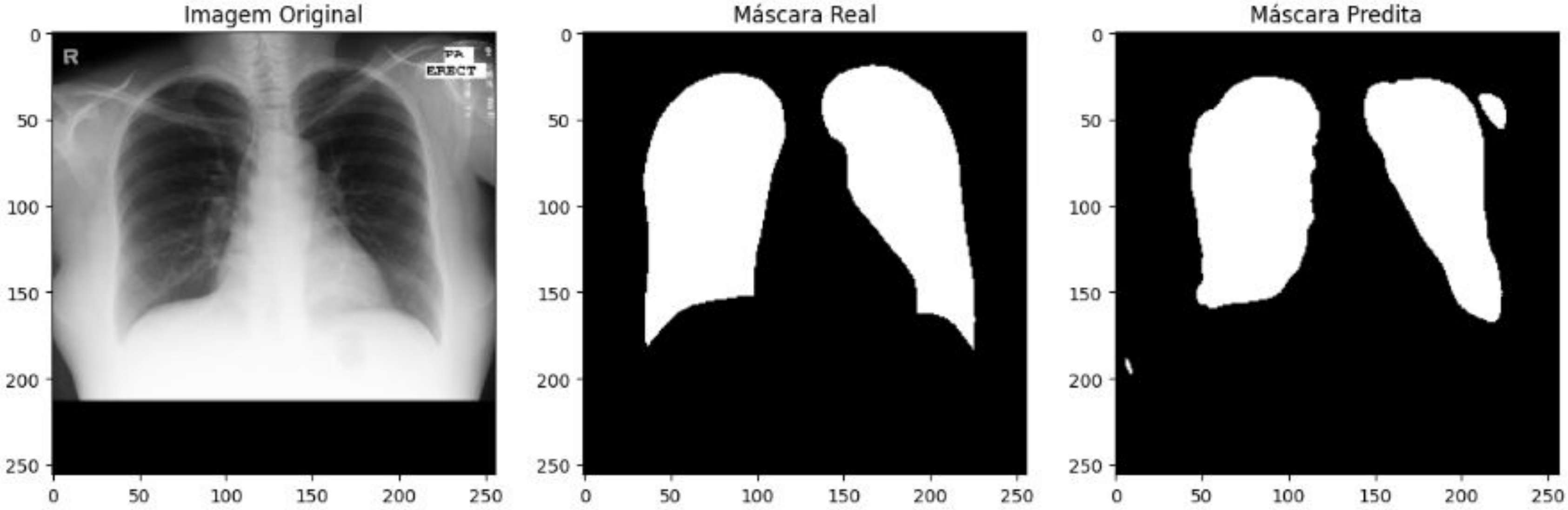

```
Imagem 1:
Dice Similarity: 0.7878851467142307
Fitness Adjust:  0.6500086560101563
Size Adjust:     0.9781415031651104
Position Adjust: 0.987189769777513
IoU:             0.6500086560101563
```

Métrica de avaliação – Unet++



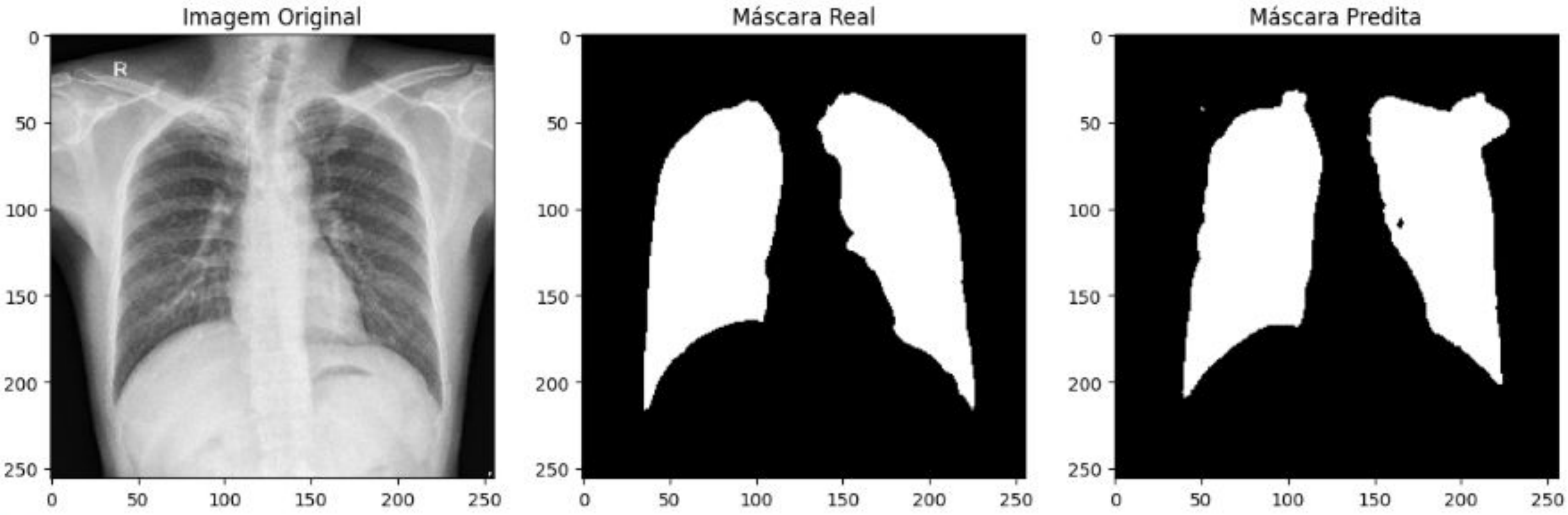

```
Imagem 2:
Dice Similarity: 0.8828506569443489
Fitness Adjust:  0.7902709359605912
Size Adjust:     0.947513242071954З
Position Adjust: 0.9965843090194046
IoU:             0.7902709359605912
```

Métrica de avaliação - Unet++



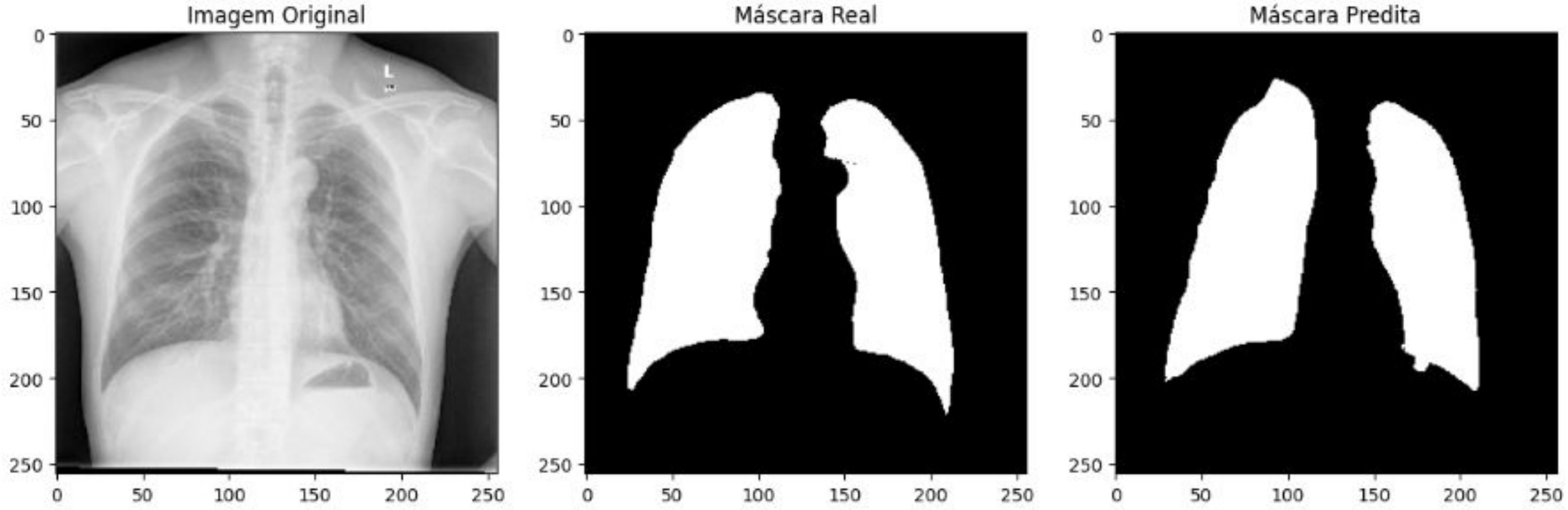

```
Imagem 3:
Dice Similarity: 0.8764281611029553
Fitness Adjust:  0.780037493609044
Size Adjust:     0.9264058211527414
Position Adjust: 0.9939920008789298
IoU:             0.780037493609044
```

Métrica de avaliação – Unet++



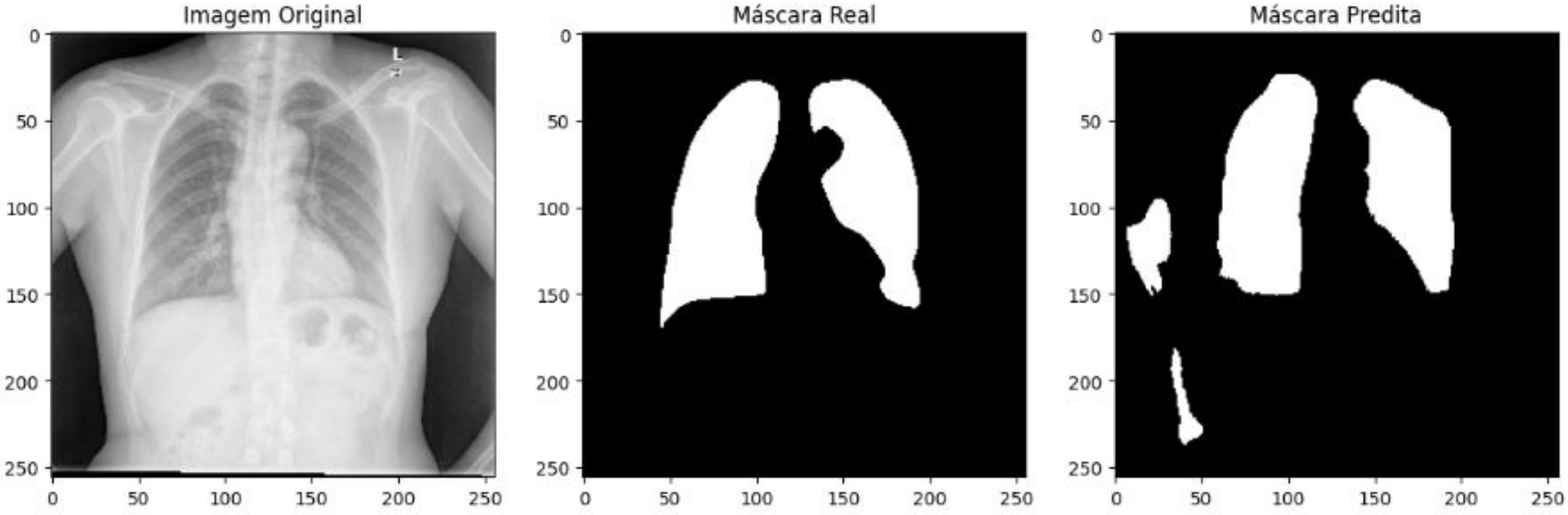

```
Imagem 4:
Dice Similarity: 0.8892198084559683
Fitness Adjust:  0.8005362494085484
Size Adjust:     0.993225881803317
Position Adjust: 0.9881442611934609
IoU:             0.8005362494085484
```

Métrica de avaliação - Unet++



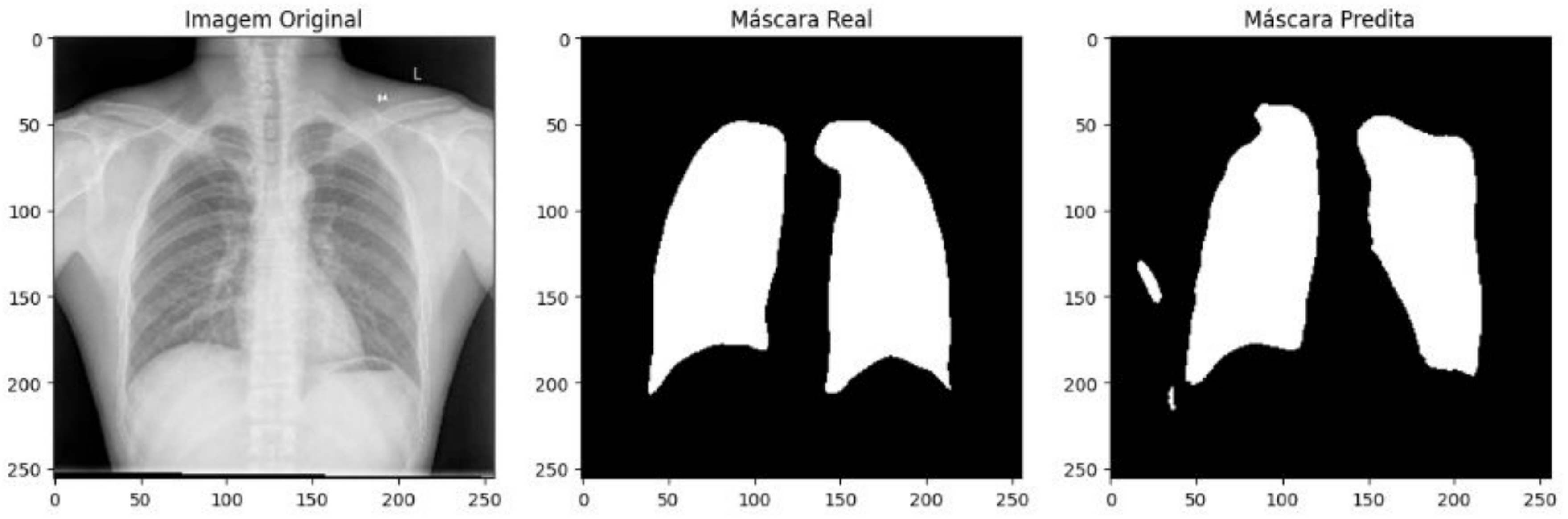

```
Imagem 5:
Dice Similarity: 0.8999252950844165
Fitness Adjust:  0.8180583473678492
Size Adjust:     0.9737337516808606
Position Adjust: 0.9954593322145358
IoU:             0.8180583473678492
```

Métrica de avaliação – Unet++



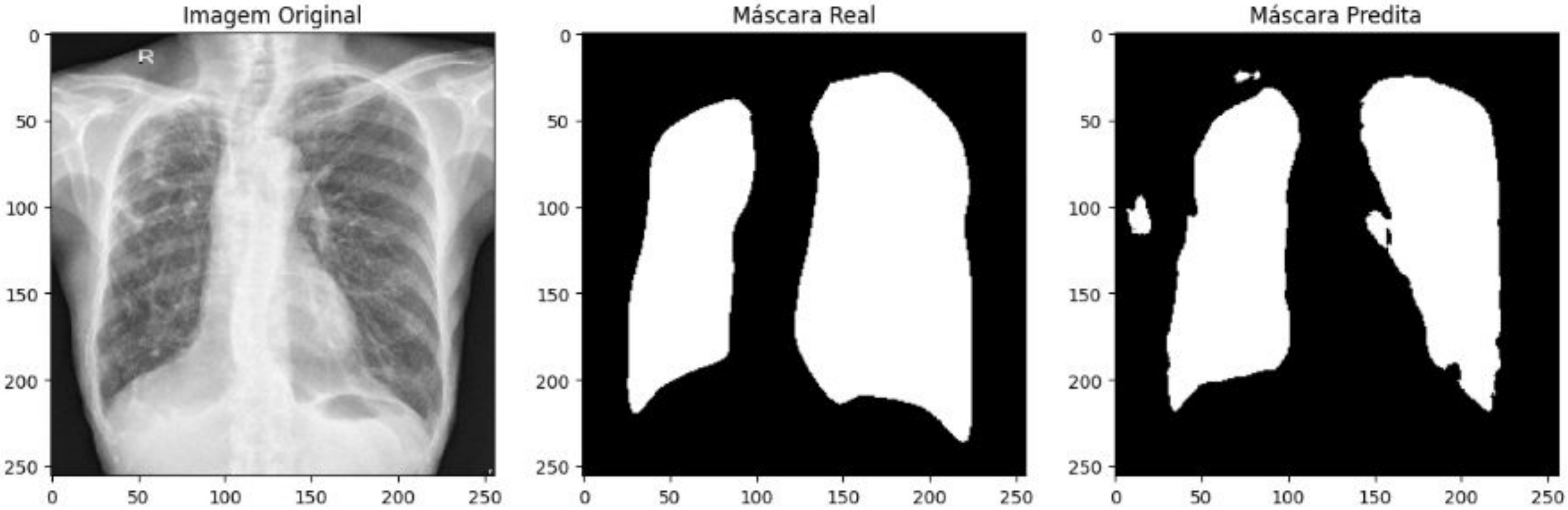

```
Imagem 6:
Dice Similarity: 0.8167983743508692
Fitness Adjust:  0.6903289825204183
Size Adjust:     0.9786407766990292
Position Adjust: 0.9933658897388697
IoU:             0.6903289825204183
```

Métrica de avaliação – Unet++




```
Imagem 7:
Dice Similarity: 0.8390009027986759
Fitness Adjust:  0.7226542249870399
Size Adjust:     0.9633463737586518
Position Adjust: 0.9875842742504379
IoU:             0.7226542249870399
```

Métrica de avaliação – Unet++

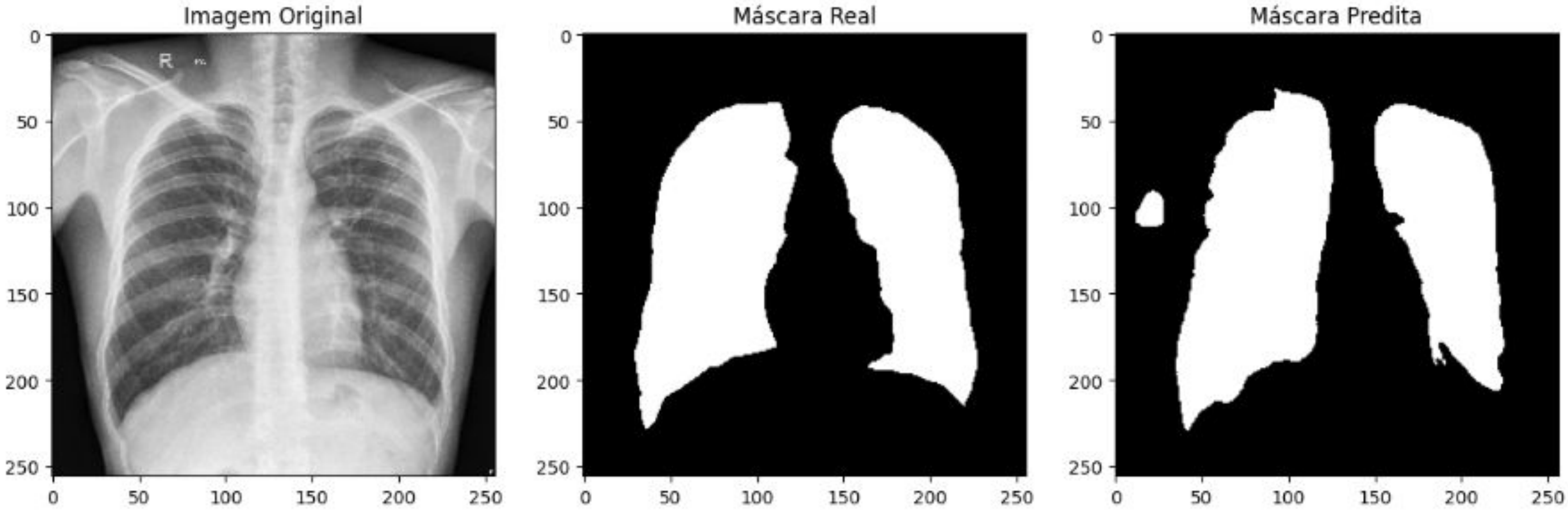



```
Imagem 8:
Dice Similarity: 0.7881995661605206
Fitness Adjust:  0.6504367750250608
Size Adjust:     0.8845119305856833
Position Adjust: 0.9760786515679101
IoU:             0.6504367750250608
```

Métrica de avaliação – Unet++


```
Imagem 9:
Dice Similarity: 0.8430839538273648
Fitness Adjust:  0.7287339099639039
Size Adjust:     0.9585943347910019
Position Adjust: 0.9887528345931965
IoU:             0.7287339099639039
```

Métrica de avaliação – Unet++




```
Imagem 10:
Dice Similarity: 0.8842325557654025
Fitness Adjust:  0.7924882199551672
Size Adjust:     0.9970394568934715
Position Adjust: 0.9967973279691346
IoU:             0.7924882199551672
```

CONCLUSÕES

Comparando a média das métricas de uma amostra de 10 imagens aleatorias

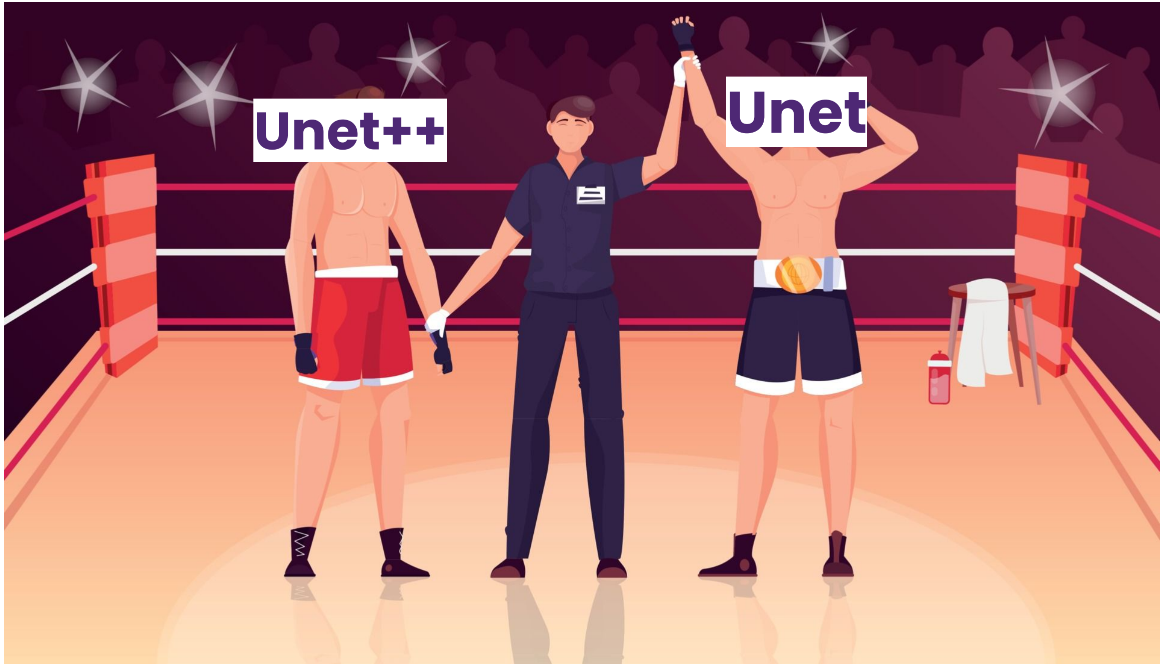
Unet

- **Média Dice Similarity** : 0.9308
- **Média Fitness Adjust** : 0.9257
- **Média Size Adjust** : 0.42354
- **Média Position Adjust** : 0.9651
- **Média IoU** : 0.2687

Unet++

- **Média Dice Similarity** : 0.8605
- **Média Fitness Adjust** : 0.7721
- **Média Size Adjust** : 0.9398
- **Média Position Adjust** : 0.9907
- **Média IoU** : 0.7721

Unet padrão saiu melhor nesse trabalho, porém



Possibilidade de estudo em mais aplicações da arquitetura

- Outras segmentação de Imagens Médicas:
- Tumores Cerebrais: uso na segmentação tumores em imagens de ressonância magnética (MRI).
- Nódulos Pulmonares: Segmentação de nódulos em tomografias computarizadas do tórax.
- Segmentação de células e tecidos em imagens de microscopicas

Referências Bibliográficas

- PANDEY, N. **Chest Xray Masks and Labels**. Disponível em: <https://www.kaggle.com/datasets/nikhilpandey360/chest-xray-masks-and-labels/data>. Acesso em: 10 ago. 2024.
- **U-Net: Convolutional Networks for Biomedical Image Segmentation** https://arxiv.org/abs/1505.04597
- **Origem das funções usadas como métricas de avaliação** https://github.com/reneripardocalixto/tests_accuracy/blob/master/compute_metrics.py
- **Artigo com explicação e código Unet++** https://www.geeksforgeeks.org/unet-architecture-explained/

Obrigado!

Atlântico Avanti