Bootcamp Machine Learning

Atlântico Avanti

SEGMENTAÇÃO PULMONAR

Equipe 05:

Edson Mokiu Yabiku
Erick Vieira Maestrello
Henri Silva de Oliveira Reis
Jonathas Magalhães
José Lisiomar de Souza
Robson Amorim de Melo
Vinícius Arantes Emrich Leão

Atlântico Avanti - Bootcamp Machine Learning

Objetivo:

Este estudo visa aplicar técnicas de aprendizado de máquina para segmentar imagens de pulmão de forma precisa e eficiente, visando auxiliar profissionais da saúde no diagnóstico e tratamento de doenças respiratórias. Neste apanhado de dados existem 704 máscaras, mas 800 imagens. Portanto, o objetivo é criar uma correspondência 1-1 entre máscaras e imagens, o que implica que haverá algumas imagens sem máscaras.

Metodologia:

Utilizaremos a técnica de segmentação de imagens com algoritmos de Machine Learning para identificar e demarcar as áreas das radiografias que apresentam anomalias.
Os algoritmos serão treinados com um conjunto de dados rotulados com as respectivas anomalias e aprenderão a identificar padrões nas imagens que indiquem a presença de irregularidades.

Atlântico Avanti - Bootcamp Machine Learning

Principais Metodologia:

1. Pré-processamento das Imagens: As imagens foram pré-processadas de várias maneiras, incluindo:
    - Normalização: As intensidades de pixel foram normalizadas para um intervalo específico, por exemplo, dividindo por 127 e subtraindo 1, para trazer os valores para o intervalo [-1, 1].
    - Redimensionamento: As imagens foram redimensionadas para um tamanho específico, utilizando a função cv2.resize.
2. Divisão dos Dados: Os dados foram divididos em conjuntos de treinamento, teste e validação, utilizando a função train_test_split da biblioteca sklearn.model_selection. Uma proporção comum usada foi de 80% para treinamento, 10% para validação e 10% para teste.
3. Geração de Máscaras: Para tarefas de segmentação, as máscaras correspondentes às regiões de interesse foram geradas a partir das imagens originais. Isso muitas vezes envolveu a binarização das máscaras e o ajuste de seus valores de intensidade.
4. Visualização dos Dados: Foi realizada uma visualização dos dados para garantir que o pré-processamento tenha sido realizado corretamente e para entender melhor a natureza dos dados. Isso incluiu a plotagem de imagens de exemplo, máscaras e suas correspondentes após o pré-processamento.
5. Verificação de Dados: Foram realizadas verificações de integridade dos dados, como a correspondência entre imagens e máscaras, garantindo que não houvesse erros no carregamento ou pré-processamento.

Primeiro passo:

```
import numpy as np
import tensorflow as tf
import pandas as pd
from tqdm import tqdm
import os
from cv2 import imread, createCLAHE
import cv2
from glob import glob
%matplotlib inline
import matplotlib.pyplot as plt

image_path = os.path.join("../input/data/Lung Segmentation/CXR_png")
mask_path = os.path.join("../input/data/Lung Segmentation/","masks/")
```

Inicialmente é necessário importar algumas bibliotecas para processamento de imagens (como OpenCV), manipulação de dados (como Pandas e NumPy), aprendizado de máquina (como TensorFlow) e visualização (como Matplotlib).
Além disso, é preciso definir os caminhos para os diretórios onde as imagens e as máscaras estão armazenadas.

Primeira metodologia - **Carregamento dos Dados**

Atlântico Avanti

Importância:

- Facilita a Manipulação de Dados: Permite acessar e manipular imagens e máscaras no ambiente de programação.
- Pré-requisito para Pré-processamento: Sem carregar os dados, não seria possível aplicar qualquer técnica de pré-processamento, visualização ou treinamento de modelo.
- Escalabilidade: Estruturar adequadamente o carregamento de dados torna o código mais escalável e fácil de adaptar a novos conjuntos de dados.

O carregamento de dados é, portanto, a base sobre a qual todo o pipeline de processamento e análise de dados é construído.

```python
import os
import cv2

# Definindo os caminhos dos diretórios das imagens e máscaras
image_path = os.path.join("../input/data/Lung Segmentation/CXR_png")
mask_path = os.path.join("../input/data/Lung Segmentation/masks/")

# Listando todos os arquivos nos diretórios especificados
images = os.listdir(image_path)
masks = os.listdir(mask_path)
```

Os dados foram carregados a partir de arquivos de imagem no formato PNG utilizando bibliotecas como OpenCV (`cv2`) e `os`

Segunda metodologia – **Pré-processamento das Imagens**

Atlântico Avanti

Importância:

Pré-processamento das Imagens: As imagens foram pré-processadas de várias maneiras, incluindo:

1. Normalização:
   - Melhoria do Desempenho do Modelo: Centralizar os dados ajuda a acelerar a convergência durante o treinamento do modelo.
   - Estabilidade Numérica: Normalizar os valores dos pixels reduz o risco de problemas de estabilidade numérica durante o treinamento.
2. Redimensionamento:
   - Consistência de Entrada: Modelos de deep learning geralmente requerem entradas de tamanho fixo. Redimensionar as imagens garante que todas as entradas sejam consistentes.
   - Redução de Complexidade Computacional: Redimensionar para um tamanho menor pode reduzir a carga computacional e o tempo de treinamento.

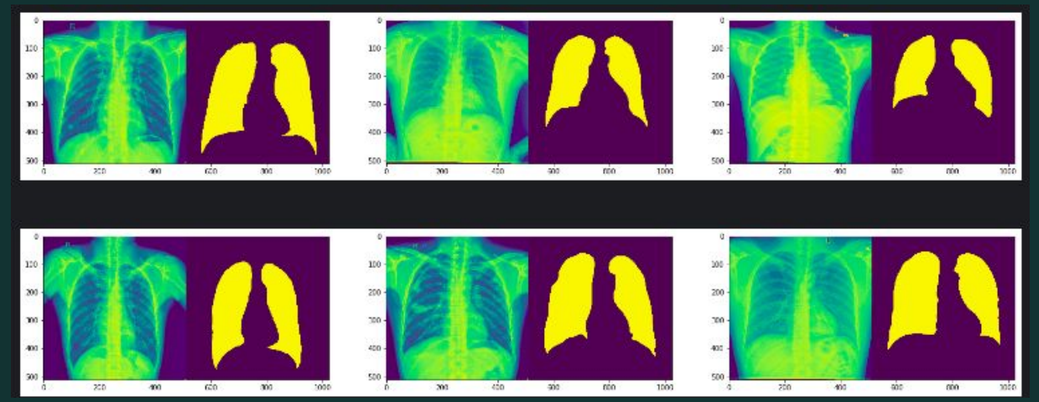
```
def getData(X_shape, flag = "test"):
    im_array = []
    mask_array = []

    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_sha
pe,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
```

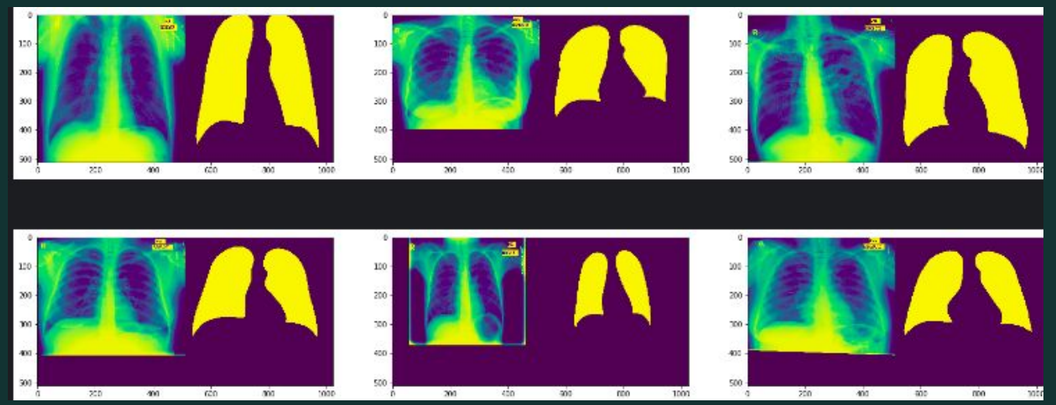
Terceira metodologia – **Divisão dos Dados em Conjuntos**

Atlântico Avanti

Importância:

- Treinamento:
  i. Ajuste dos Pesos do Modelo: O conjunto de treinamento é usado para ajustar os pesos do modelo através de iterações (épocas) durante o processo de treinamento.
- Validação:
  i. Ajuste de Hiperparâmetros: O conjunto de validação é usado para ajustar os hiperparâmetros do modelo, como a taxa de aprendizado, e para monitorar a performance do modelo durante o treinamento.
  ii. Prevenção de Overfitting: Ajuda a identificar e prevenir overfitting ao fornecer uma medida de como o modelo generaliza para novos dados não vistos durante o treinamento.
- Teste:
  i. Avaliação Final do Modelo: O conjunto de teste é usado para avaliar a performance final do modelo após o treinamento e ajuste de hiperparâmetros.
  ii. Medição da Generalização: Proporciona uma estimativa imparcial de como o modelo performa em dados novos e não vistos.





Quarta metodologia – **Geração de Máscaras**

Atlântico Avanti

Importância:

- Supervisão Direta:
  - Treinamento Supervisionado: Máscaras fornecem a supervisão necessária para treinar modelos de segmentação, indicando exatamente onde a característica de interesse está localizada na imagem.
- Precisão na Segmentação:
  - Melhora da Performance do Modelo: Máscaras binárias ajudam o modelo a aprender a separar regiões de interesse com alta precisão.
- Preparação dos Dados:
  - Consistência de Dimensões: Redimensionar as máscaras para coincidir com as dimensões das imagens garante que cada pixel da imagem de entrada tenha uma correspondência direta na máscara, facilitando o processo de treinamento.

```
#perform sanity check

def plotMask(X,y):
    sample = []

    for i in range(6):
        left = X[i]
        right = y[i]
        combined = np.hstack((left,right))
        sample.append(combined)


    for i in range(0,6,3):

        plt.figure(figsize=(25,10))

        plt.subplot(2,3,1+i)
        plt.imshow(sample[i])

        plt.subplot(2,3,2+i)
        plt.imshow(sample[i+1])


        plt.subplot(2,3,3+i)
        plt.imshow(sample[i+2])

        plt.show()
```

Quinta metodologia – **Visualização dos Dados:**

Atlântico Avanti

Importância:

1. Verificação de Alinhamento:
   - Correspondência Correta: Assegura que cada imagem está corretamente alinhada com sua máscara correspondente, crucial para o sucesso do treinamento do modelo.
2. Identificação de Erros:
   - Erro de Carregamento: Ajuda a identificar problemas como imagens mal carregadas ou máscaras incorretamente associadas.
   - Inconsistências nos Dados: Permite detectar inconsistências nos dados que poderiam prejudicar o desempenho do modelo.
3. Avaliação Qualitativa:
   - Performance do Modelo: Visualizar previsões do modelo junto com as máscaras reais ajuda a avaliar qualitativamente a performance do modelo, complementando as métricas quantitativas.

```python
import matplotlib.pyplot as plt

def plotMask(X, y):
    sample = []

    for i in range(6):
        left = X[i]
        right = y[i]
        combined = np.hstack((left, right))
        sample.append(combined)

    for i in range(0, 6, 3):
        plt.figure(figsize=(25, 10))

        plt.subplot(2, 3, 1 + i)
        plt.imshow(sample[i], cmap='gray')

        plt.subplot(2, 3, 2 + i)
        plt.imshow(sample[i + 1], cmap='gray')

        plt.subplot(2, 3, 3 + i)
        plt.imshow(sample[i + 2], cmap='gray')

        plt.show()

# Realizar a verificação de sanidade dos dados de treinamento e teste
print("training set")
plotMask(X_train, y_train)
```

Sexta metodologia – **Verificação dos Dados:**

Atlântico Avanti

Importância:

1. Garantia de Qualidade:
   - Assegura que os dados estejam limpos, consistentes e prontos para o treinamento do modelo, ajudando a evitar problemas durante o processo de treinamento.
2. Identificação de Anomalias:
   - Permite detectar e corrigir anomalias nos dados, como imagens corrompidas, máscaras ausentes ou mal alinhadas, que podem prejudicar a performance do modelo.
3. Prevenção de Erros:
   - Ajuda a prevenir erros durante o treinamento do modelo, garantindo que os dados utilizados sejam de alta qualidade e confiáveis.

```
# Realizar a verificação de sanidade dos dados de treinamento e teste
print("training set")
plotMask(X_train, y_train)
print("testing set")
plotMask(X_test, y_test)
```


```
# Realizar a verificação de sanidade dos dados de treinamento e teste
print("training set")
plotMask(X_train, y_train)
print("testing set")
plotMask(X_test, y_test)
```

Conclusões e Resultados Finais:

1. Desempenho do Modelo:
    - O modelo U-Net, após o treinamento, apresentou boas métricas de desempenho, com uma perda de validação de -0.97 e uma alta acurácia binária. Esses resultados indicam que o modelo é eficaz na segmentação de áreas pulmonares em imagens de raios-X

2. Qualidade das Segmentações:

    - A visualização das predições do modelo mostrou que as segmentações geradas são visualmente consistentes com as máscaras reais. Isso sugere que o modelo aprendeu bem a tarefa de segmentação, distinguindo corretamente as áreas de interesse na maioria dos casos.

3. Importância da Normalização e Pré-Processamento:

    - A normalização das imagens e a transformação das máscaras em valores binários foram etapas cruciais que contribuíram para o sucesso do treinamento do modelo. Esses passos garantiram que os dados estivessem em um formato adequado para o aprendizado eficiente da rede neural.

4. Treinamento e Validação:

    - Utilizar uma divisão de dados em 80-10-10 para treinamento, teste e validação se mostrou uma estratégia eficaz para avaliar a generalização do modelo. Essa abordagem permitiu um monitoramento constante da performance em dados não vistos, evitando overfitting.

5. Uso de Callbacks:

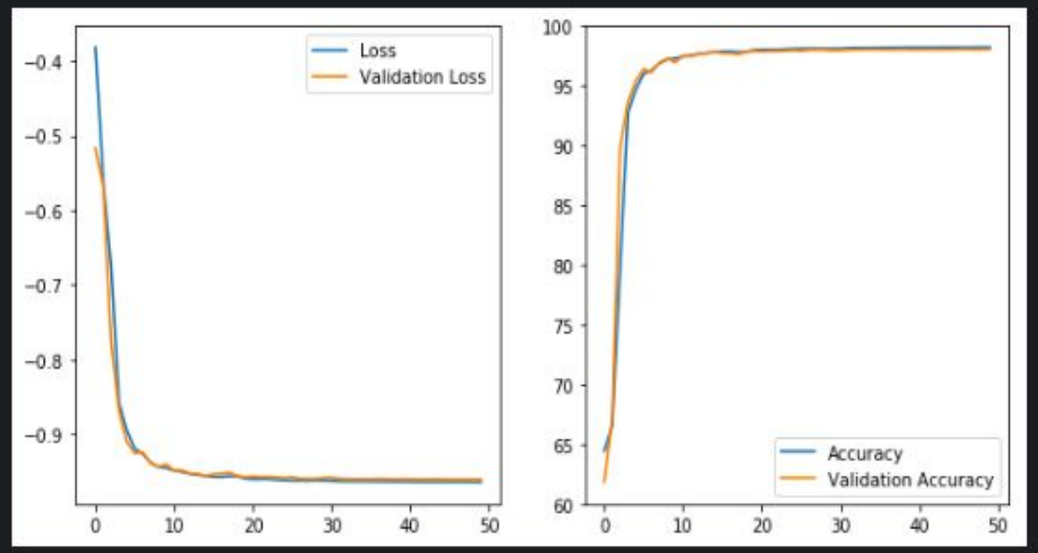
    - A utilização de callbacks como ModelCheckpoint, EarlyStopping e ReduceLROnPlateau foi fundamental para melhorar a eficiência do treinamento e evitar o overfitting. Esses mecanismos permitiram salvar o melhor modelo, parar o treinamento quando não havia mais melhorias e ajustar a taxa de aprendizado dinamicamente.

Resultado Final:

1. Métricas de Desempenho:
   - A perda de validação (val_loss) e a acurácia binária de validação (val_binary_accuracy) apresentaram valores que indicam um bom ajuste do modelo aos dados de treinamento e validação.
   - A função de perda baseada no coeficiente Dice foi eficaz para a tarefa de segmentação, penalizando adequadamente predições incorretas e incentivando a maximização da sobreposição correta entre predições e máscaras reais.
2. Visualização de Resultados:
   - As visualizações das predições comparadas com as máscaras reais mostram que o modelo consegue segmentar corretamente as áreas pulmonares na maioria dos casos, demonstrando a eficácia do U-Net para essa tarefa específica.
   - Pequenas discrepâncias observadas nas segmentações sugerem que há espaço para melhorias, possivelmente através de ajustes de hiperparâmetros, maior quantidade de dados de treinamento ou técnicas de data augmentation mais avançadas.

```
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 0x7f5ce83ba978>
```



Considerações Finais:

O projeto demonstrou a capacidade do modelo U-Net de realizar segmentação de pulmões em imagens de raios-X com alta precisão. As técnicas de pré-processamento, normalização dos dados e a utilização de callbacks foram cruciais para o sucesso do treinamento. A abordagem de segmentação pode ser aplicada a outros conjuntos de dados médicos com ajustes específicos, o que destaca a flexibilidade e a utilidade das redes neurais convolucionais em tarefas de visão computacional no campo da saúde.

Esse trabalho contribui para o desenvolvimento de ferramentas automatizadas que podem auxiliar médicos na análise de imagens médicas, potencialmente melhorando a eficiência e a precisão dos diagnósticos.

Imagem: Mascará: Resultado Gerado: