**Nome: Marcus Vinicius Oliveira Nunes**
**Assunto: Relatório do projeto final**
**RA:2554100**

LAMIA
Data Science & Computer Vision for Industry

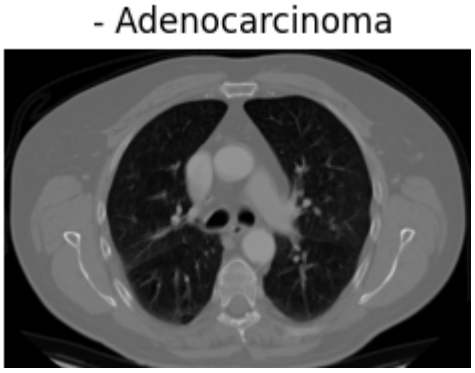
## 1 - Objetivo do Trabalho

Treinar e comparar modelos de aprendizado de máquina capazes de distinguir as tomografias torácicas de cânceres pulmonares e uma tomografia torácica normal.

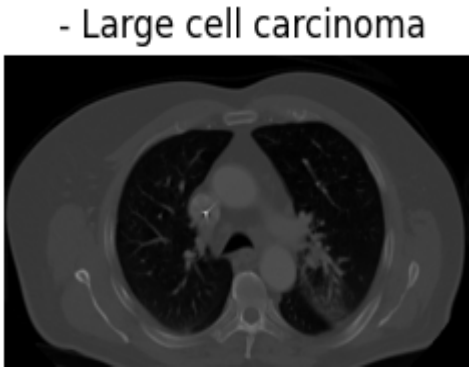
## 2 - Base de dados

A base de dados utilizada para o desenvolvimento do trabalho foi a “Chest CT-Scan images Dataset” que é basicamente uma base de dados com imagens de tomografia computadorizada do tórax. As imagens são divididas em 3 classificações de câncer no pulmão e outra classificação de sem câncer. Os tipos de classes são:

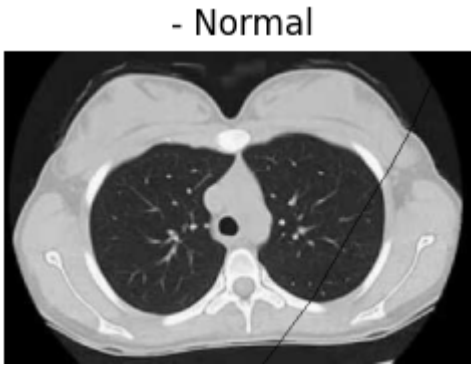
- **Adenocarcinoma:** 338 imagens
- **Carcinoma de células grandes:** 187 imagens
- **Normal:** 215 imagens
- **Carcinoma de células escamosas:** 260 imagens



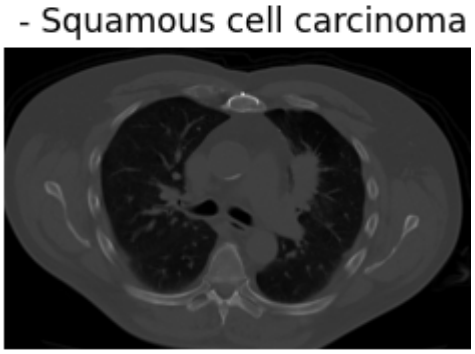

Adenocarcinoma




Carcinoma de células grandes




Normal




Carcinoma de células escamosas

Do total de 1000 imagens foram pré divididas em 70% para treinamento, 20% para teste e 10% para validação.

## 2.1 - Data augmentation ou aumento de dados

Com o intuito de aumentar a capacitação do modelo no reconhecimento das imagens foi aplicado o "Data Augmentation" que é o aumento dos dados durante o treinamento aplicando mudanças nos dados que já existem na base de dados. Um exemplo de mudança é virar a imagem de cabeça para baixo, mudar a imagem de lado, aplicar zoom, mudar o eixo etc. As mudanças aplicadas neste trabalho estão abaixo:

```
train = ImageDataGenerator(
                           preprocessing_function = preprocess_input,
                           rotation_range =10,
                           width_shift_range =0.2,
                           height_shift_range =0.2,
                           shear_range =0.2,
                           zoom_range =0.2,
                           horizontal_flip = True,
                           vertical_flip = False,
                           dtype = 'float32', #optimization reason
                           )
```

exceto pela "preprocessing_function" e pela "dytpe" as outras funções aplicam mudanças com o intuito de aumentar o número de imagens.

## 2.2 - Pré-processamento de dados

Com o intuito de otimizar o processamento e a qualidade da generalização do modelo foram aplicadas duas funções de pré-processamento.

### 2.2.1 - “preprocess_input” e “dtype”

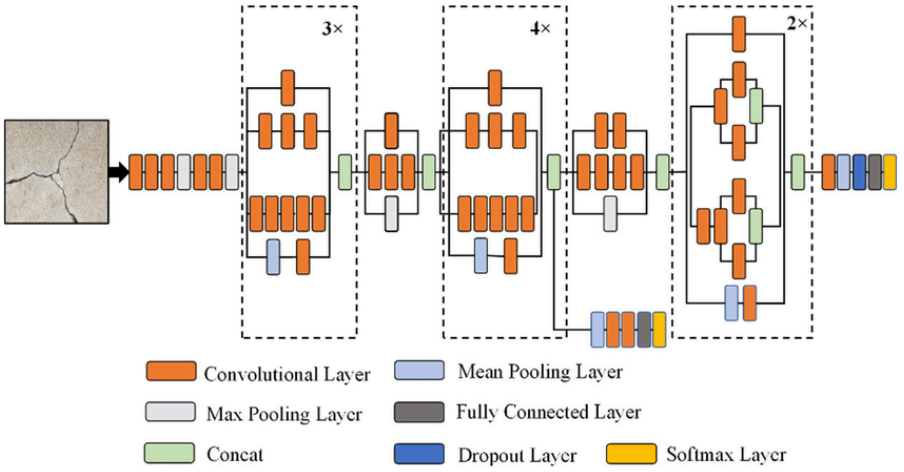
Essa função normaliza os valores dos pixels, que normalmente estão na escala de [0,255], de cada imagem para [-1, 1]. A seguinte normalização diminui a complexidade dos pixels das imagens para uma escala menor, melhorando a otimização do processamento. A função foi importada da seguinte forma:

```
from tensorflow.keras.applications.resnet import preprocess_input
```

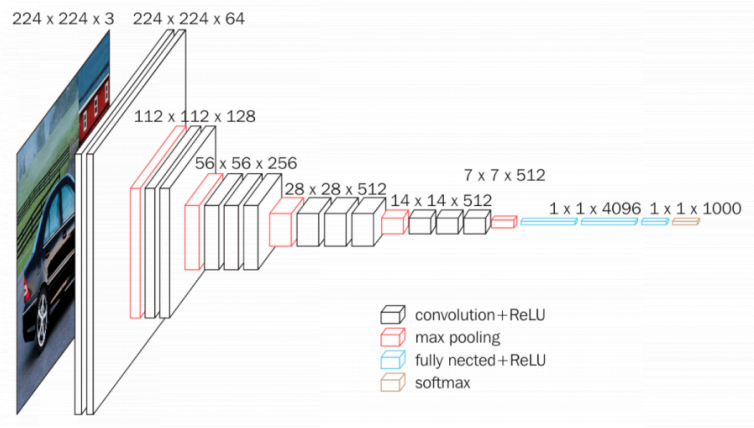
Além do “preprocess_input” outra função de pré -processamento foi utilizada, a “dtype” que garante que cada tensor de cada imagem gerada seja convertida para float32, também ajudando na otimização além de outros benefícios.

## 3 - Modelos

Durante a realização desse projeto foram usados dois modelos prontos de redes convolucionais: a rede Inception-V3 e a VGG16, as quais são usadas justamente para classificação de imagens. O objetivo de testar dois modelos diferentes foi testar um modelo que já utilizado e testado por outra pessoa na base de dados utilizada nesse projeto, no caso o VGG16 e comparar com outro modelo da minha escolha, que no caso acabou sendo o Inception-V3.




Arquitetura da rede Inception-V3




Arquitetura da rede VGG16

## 4 - Treinamentos, resultados e validação cruzada

Com a finalidade de capacitar os modelos a fazerem classificações acuradas, foram efetuados treinamentos dos mesmos. Após os treinamentos, para garantir que os modelos estavam capacitados foram submetidos aos dados de teste. Ademais, os modelos foram submetidos à validação cruzada com o objetivo de medir a capacidade de generalização dos modelos.

## 4.1 - VGG16

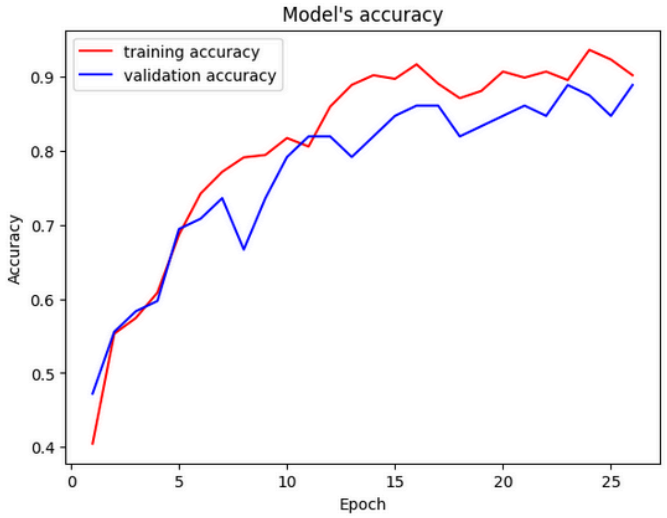
Para o treinamento do VGG16 foi escolhido um total de 100 épocas com um early stopper com 10 épocas de paciência, o qual tem a função de parar o treinamento se não houver melhora na perda,  monitorando cada época do treinamento.

```
epochs = 100
history = model.fit(
    training_data,
    validation_data=validation_data,
    epochs=epochs,
    callbacks = earlystopper,
    verbose=1
)
```

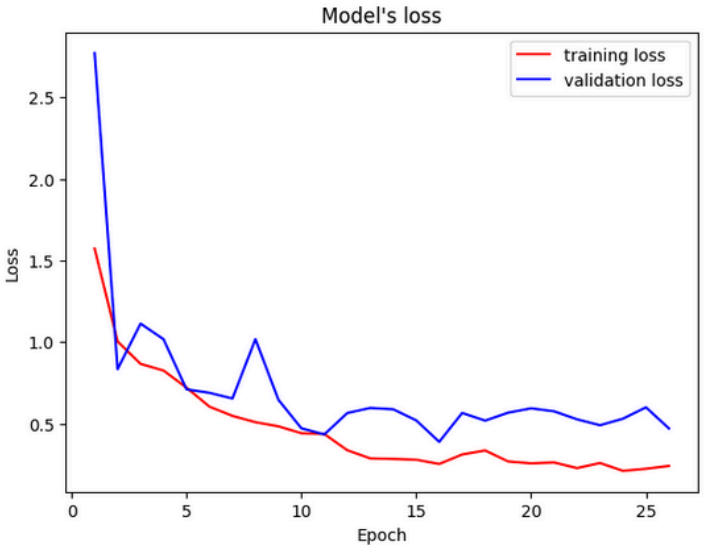
Parâmetros do treinamento

O treino acabou durando 26 épocas devido a falta de melhora durante as 10 épocas anteriores. O treino finalizou com os seguintes resultados:

- Perda: 0.2398
- Acurácia: 0.9021
- Perda de validação: 0.4698
- Acurácia de validação: 0.8889




Acurácia durante treinamento




Perda durante o treinamento

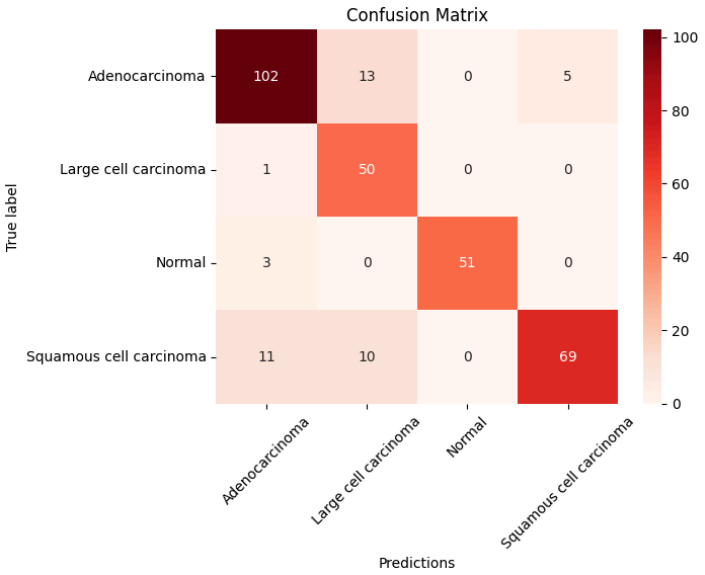
Na hora de avaliar com os dados de teste se obteve uma acurácia de 86% e perda de 41%. Além destes, outros resultados como precisão, recall, f1-score e matriz de confusão podem ser observados seguintes abaixo:

```
5/5 [==============================] - 6s 1s/step
              precision    recall  f1-score   support

           0       0.87      0.85      0.86       120
           1       0.68      0.98      0.81        51
           2       1.00      0.94      0.97        54
           3       0.93      0.77      0.84        90

    accuracy                           0.86       315
   macro avg       0.87      0.89      0.87       315
weighted avg       0.88      0.86      0.87       315
```

Resultados




Matriz de confusão dos resultados

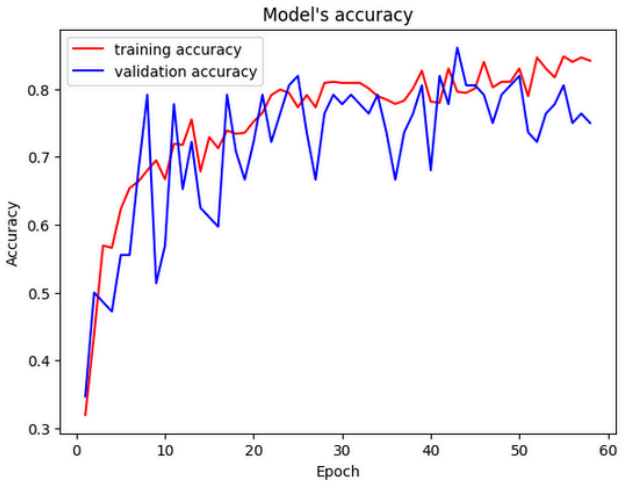
Após a avaliação com os dados de teste foi realizada uma validação cruzada para medir a capacidade de generalização do modelo. Foi escolhida uma quantidade de 100 épocas com um early-stopper de paciência de 10 épocas. Foram necessárias apenas 68 épocas. Os resultados foram os seguintes:

- Média da perda de validação: 0.0760248675942421
- Média da acurácia: 0.9918032884597778

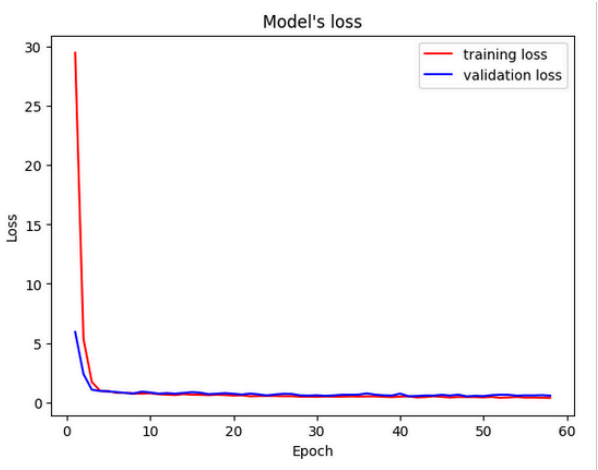
## 4.2 - Inception-V3

Para o Inception-V3 foram escolhidas 200 épocas de treinamento com um early stopper com paciência de 10 épocas. O Treinamento acabou com o total de 58 épocas, com 32 épocas a mais do que o VGG16. Os resultados depois do treinamento foram:

- Perda: 0.3615
- Acurácia categorial: 0.8418
- Perda de validação: 0.5531
- Acurácia categorial de validação: 0.7500




Acurácia durante o treinamento




Perda durante o treinamento

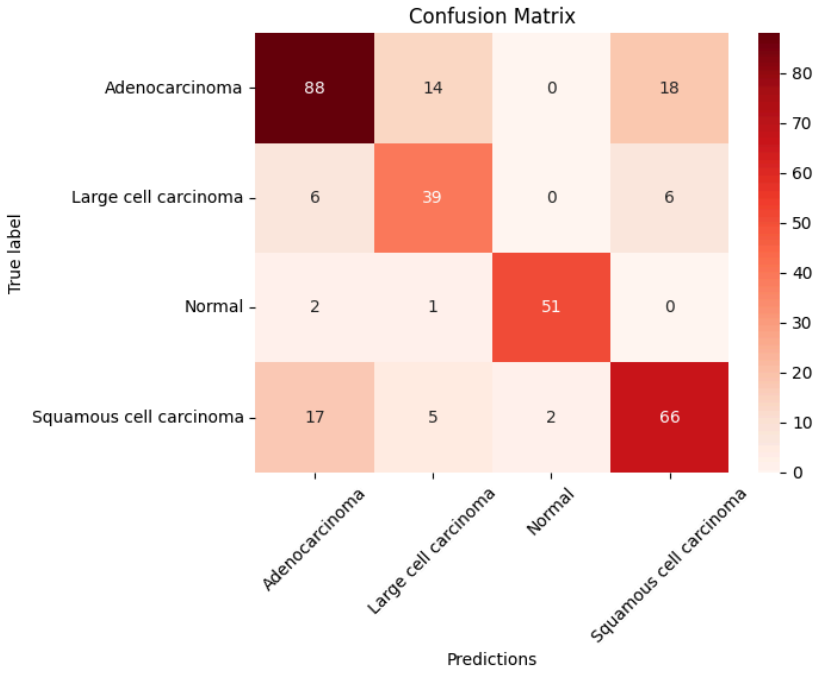
Com os dados de teste a acurácia foi de 77% e a perda de 56%. Outros resultados podem ser vistos abaixo:

```
5/5 [==============================] - 3s 701ms/step
              precision    recall  f1-score   support

           0       0.78      0.73      0.76       120
           1       0.66      0.76      0.71        51
           2       0.96      0.94      0.95        54
           3       0.73      0.73      0.73        90

    accuracy                           0.77       315
   macro avg       0.78      0.79      0.79       315
weighted avg       0.78      0.77      0.78       315
```

Resultados




Matriz de confusão dos resultados

Na validação cruzada foram decididas 100 épocas também com um early stopper com 5 épocas de paciência, 70 épocas foram efetuadas e os seguintes resultados se mostraram:

- Média da perda de validação: 0.4710097014904022
- Média da acurácia: 0.8360655903816223

**LINK PARA O DATASET:**

https://www.kaggle.com/datasets/mohamedhanyyy/chest-ctscan-images