In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import os
import cv2
import tensorflow as tf
import time
import glob
import random
from PIL import Image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, GlobalAveragePooling2D,Dropout
from tensorflow.keras.callbacks import TensorBoard
from tensorflow.keras.optimizers import Adam
from keras.preprocessing import image
import time
import pandas as pd
from tensorflow.keras.optimizers import Adam


NAME = "Tcc_cnn_64_{}".format(int(time.time()))
```

In [ ]:
```
print(tf.config.list_physical_devices('GPU'))
print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))

if tf.test.gpu_device_name():
    print('Default GPU Device: {}'.format(tf.test.gpu_device_name()))
else:
    print("Please install GPU version of TF")


print(tf.test.is_built_with_cuda())
```

```
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
Num GPUs Available:  1
Default GPU Device: /device:GPU:0
True
```

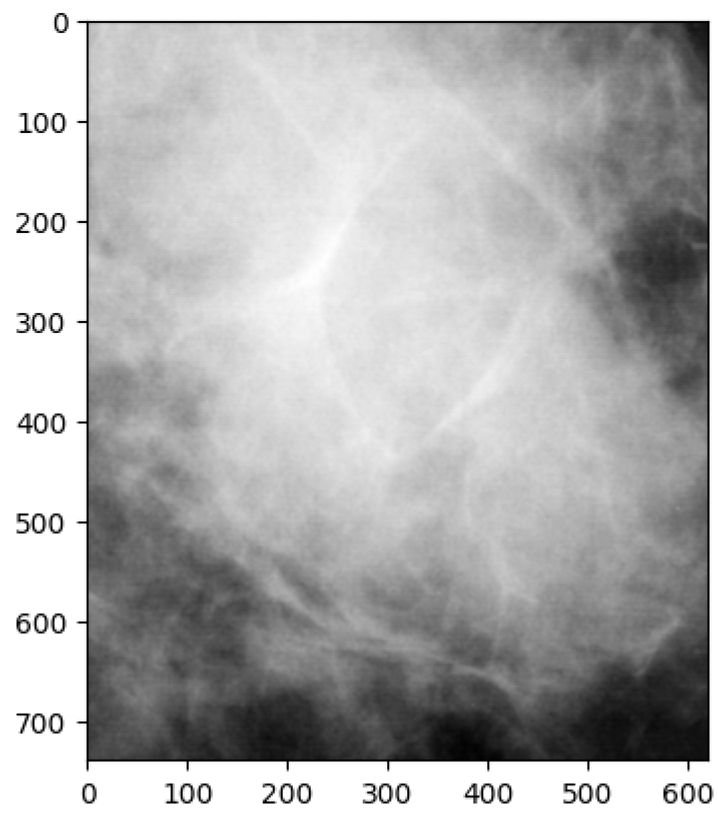
Carregando arquivos

In [ ]:
```
all_traning_paths = glob.glob('DATASETS/Training-Mass/*/*.jpg')
all_val_paths = glob.glob('DATASETS/Validation-Mass/*/*.jpg')
```

In [ ]:
```
img_path=random.choice(all_traning_paths)
cropped_img = Image.open(img_path)

gray = cropped_img.convert('L')
median =cv2.blur(np.array(gray), (3, 3))
print(img_path)
plt.imshow(median, cmap='gray')
```

```
DATASETS/Training-Mass\WITH CANCER\1-236.jpg
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x23c24ed4b50>
```



Criando datagens

In [ ]:
```
datagen_resnet = ImageDataGenerator(preprocessing_function=preprocess_input)

train_gen = datagen_resnet.flow_from_directory('DATASETS/Training-Mass/',
                            target_size=(224,224),
                            class_mode="categorical",
                            batch_size=16
                            )


validation_gen = datagen_resnet.flow_from_directory('DATASETS/Validation-Mass/',
                            target_size=(224,224),
                            class_mode="categorical",
                            batch_size=16
                            )

test_gen = datagen_resnet.flow_from_directory('DATASETS/Testing-Mass/',
                            target_size=(224,224),
                            class_mode="categorical",
                            batch_size=16,

                            )
```

```
Found 945 images belonging to 2 classes.
Found 146 images belonging to 2 classes.
Found 464 images belonging to 2 classes.
```

In [ ]:
```
base_model=ResNet50(include_top=False,
                    input_shape=(224,224,3)
)

for layer in base_model.layers:  # ´Passo para eu não retreinar as camadas do RESNET
    layer.trainable=False

    tensorBoard = TensorBoard(log_dir='logs/{}'.format(NAME))
```

Modelo inicial

In [ ]:
```
modelo = Sequential([ base_model,
                     GlobalAveragePooling2D(),
                     Dense(128, activation='relu'),
                     Dropout(0.2),
                     Dense(2, activation='Softmax')
])

modelo.summary() ## me
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 resnet50 (Functional)       (None, 7, 7, 2048)        23587712  
                                                                 
 global_average_pooling2d (G  (None, 2048)             0         
 lobalAveragePooling2D)                                          
                                                                 
 dense (Dense)               (None, 128)               262272    
                                                                 
 dropout (Dropout)           (None, 128)               0         
                                                                 
 dense_1 (Dense)             (None, 2)                 258       
                                                                 
=================================================================
Total params: 23,850,242
Trainable params: 262,530
Non-trainable params: 23,587,712
_________________________________________________________________
```

In [ ]:
```
modelo.compile(optimizer=Adam(learning_rate=1e-4),
               loss='categorical_crossentropy',
               metrics=['accuracy']
               )

history = modelo.fit(train_gen,
           validation_data=validation_gen,
           epochs=10,
           batch_size=16,
           callbacks=[tensorBoard]
        )
```

```
Epoch 1/10
60/60 [==============================] - 13s 57ms/step - loss: 0.7535 - accuracy: 0.5566 - val_loss: 0.6851 - val_accuracy: 0.5
890
Epoch 2/10
60/60 [==============================] - 3s 43ms/step - loss: 0.6440 - accuracy: 0.6392 - val_loss: 0.6757 - val_accuracy: 0.62
33
Epoch 3/10
60/60 [==============================] - 3s 42ms/step - loss: 0.6044 - accuracy: 0.6677 - val_loss: 0.6401 - val_accuracy: 0.65
75
Epoch 4/10
60/60 [==============================] - 3s 42ms/step - loss: 0.5624 - accuracy: 0.7005 - val_loss: 0.6404 - val_accuracy: 0.64
38
Epoch 5/10
60/60 [==============================] - 3s 44ms/step - loss: 0.5589 - accuracy: 0.7005 - val_loss: 0.6151 - val_accuracy: 0.67
12
Epoch 6/10
60/60 [==============================] - 3s 43ms/step - loss: 0.5149 - accuracy: 0.7598 - val_loss: 0.6115 - val_accuracy: 0.67
81
Epoch 7/10
60/60 [==============================] - 3s 44ms/step - loss: 0.5056 - accuracy: 0.7693 - val_loss: 0.6108 - val_accuracy: 0.67
81
Epoch 8/10
60/60 [==============================] - 3s 44ms/step - loss: 0.4906 - accuracy: 0.7619 - val_loss: 0.5917 - val_accuracy: 0.67
81
Epoch 9/10
60/60 [==============================] - 3s 45ms/step - loss: 0.4684 - accuracy: 0.7725 - val_loss: 0.5932 - val_accuracy: 0.69
18
Epoch 10/10
60/60 [==============================] - 3s 46ms/step - loss: 0.4708 - accuracy: 0.7651 - val_loss: 0.5910 - val_accuracy: 0.67
12
```

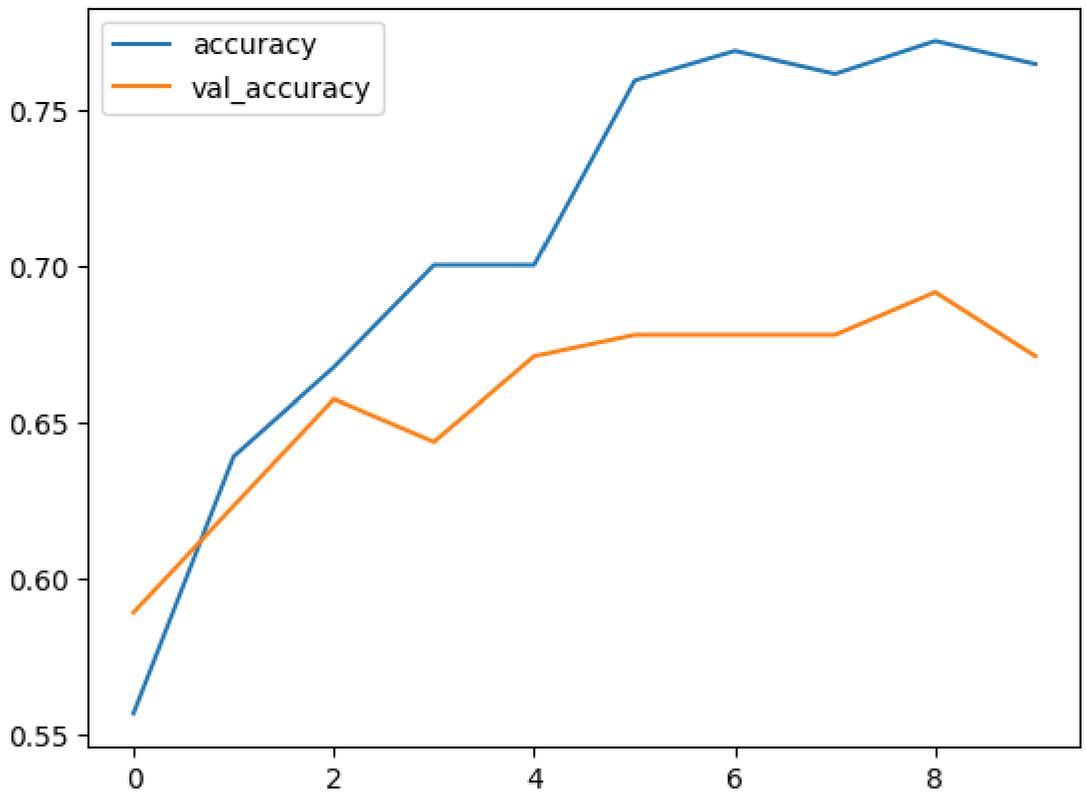
In [ ]:
```
pd.DataFrame(history.history)
```

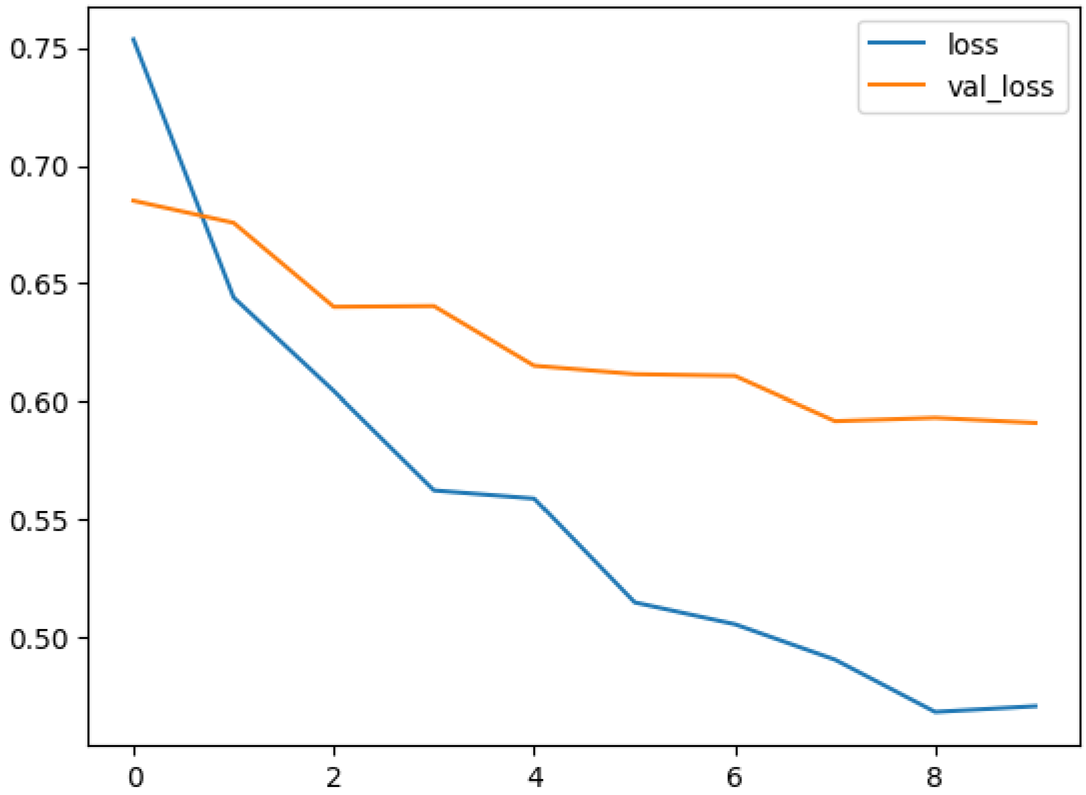
Out[ ]:

	loss	accuracy	val_loss	val_accuracy
0	0.753492	0.556614	0.685075	0.589041
1	0.644023	0.639153	0.675698	0.623288
2	0.604379	0.667725	0.640057	0.657534
3	0.562373	0.700529	0.640370	0.643836
4	0.558892	0.700529	0.615067	0.671233
5	0.514859	0.759788	0.611495	0.678082
6	0.505593	0.769312	0.610784	0.678082
7	0.490554	0.761905	0.591738	0.678082
8	0.468376	0.772487	0.593174	0.691781
9	0.470836	0.765079	0.591003	0.671233

In [ ]:
```
df = pd.DataFrame(history.history)

df[['accuracy', 'val_accuracy']].plot()
df[['loss', 'val_loss']].plot()
```

Out[ ]: <Axes: >





In [ ]:

Modelo 2 Aumentando as camadas

In [ ]:
```
modelo2 = Sequential([ base_model,
                      GlobalAveragePooling2D(),
                      Dense(128, activation='relu'),
                      Dropout(0.2),
                      Dense(2, activation='Softmax')
])

modelo2.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 7, 7, 2048)        23587712

 global_average_pooling2d_1   (None, 2048)             0
 (GlobalAveragePooling2D)

 dense_2 (Dense)             (None, 128)               262272

 dropout_1 (Dropout)         (None, 128)               0

 dense_3 (Dense)             (None, 2)                 258

=================================================================
Total params: 23,850,242
Trainable params: 262,530
Non-trainable params: 23,587,712
_________________________________________________________________
```

In [ ]:
```
modelo2.compile(optimizer=Adam(learning_rate=1e-4),
               loss='categorical_crossentropy',
               metrics=['accuracy']
               )
history = modelo2.fit(train_gen,
           validation_data=validation_gen,
           epochs=15,
           batch_size=16,
           callbacks=[tensorBoard]
        )
```

```
Epoch 1/15
60/60 [==============================] - 5s 56ms/step - loss: 0.7599 - accuracy: 0.5683 - val_loss: 0.7099 - val_accuracy: 0.62
33
Epoch 2/15
60/60 [==============================] - 3s 44ms/step - loss: 0.6677 - accuracy: 0.6317 - val_loss: 0.6942 - val_accuracy: 0.59
59
Epoch 3/15
60/60 [==============================] - 3s 43ms/step - loss: 0.6014 - accuracy: 0.6836 - val_loss: 0.6616 - val_accuracy: 0.58
90
Epoch 4/15
60/60 [==============================] - 3s 42ms/step - loss: 0.5691 - accuracy: 0.6952 - val_loss: 0.6365 - val_accuracy: 0.59
59
Epoch 5/15
60/60 [==============================] - 3s 43ms/step - loss: 0.5244 - accuracy: 0.7418 - val_loss: 0.6501 - val_accuracy: 0.58
90
Epoch 6/15
60/60 [==============================] - 3s 43ms/step - loss: 0.5228 - accuracy: 0.7280 - val_loss: 0.6416 - val_accuracy: 0.65
07
Epoch 7/15
60/60 [==============================] - 3s 44ms/step - loss: 0.5159 - accuracy: 0.7302 - val_loss: 0.6465 - val_accuracy: 0.60
96
Epoch 8/15
60/60 [==============================] - 3s 44ms/step - loss: 0.4857 - accuracy: 0.7481 - val_loss: 0.6324 - val_accuracy: 0.60
96
Epoch 9/15
60/60 [==============================] - 3s 44ms/step - loss: 0.4557 - accuracy: 0.7947 - val_loss: 0.6348 - val_accuracy: 0.61
64
Epoch 10/15
60/60 [==============================] - 3s 45ms/step - loss: 0.4497 - accuracy: 0.7852 - val_loss: 0.6352 - val_accuracy: 0.63
70
Epoch 11/15
60/60 [==============================] - 3s 44ms/step - loss: 0.4338 - accuracy: 0.7989 - val_loss: 0.6222 - val_accuracy: 0.62
33
Epoch 12/15
60/60 [==============================] - 3s 46ms/step - loss: 0.4304 - accuracy: 0.7915 - val_loss: 0.6378 - val_accuracy: 0.63
01
Epoch 13/15
60/60 [==============================] - 3s 45ms/step - loss: 0.4055 - accuracy: 0.8148 - val_loss: 0.6403 - val_accuracy: 0.60
96
Epoch 14/15
60/60 [==============================] - 3s 45ms/step - loss: 0.4126 - accuracy: 0.8212 - val_loss: 0.6150 - val_accuracy: 0.65
75
Epoch 15/15
60/60 [==============================] - 3s 44ms/step - loss: 0.3949 - accuracy: 0.8243 - val_loss: 0.6143 - val_accuracy: 0.65
75
```

In [ ]:

```
pd.DataFrame(history.history)
```

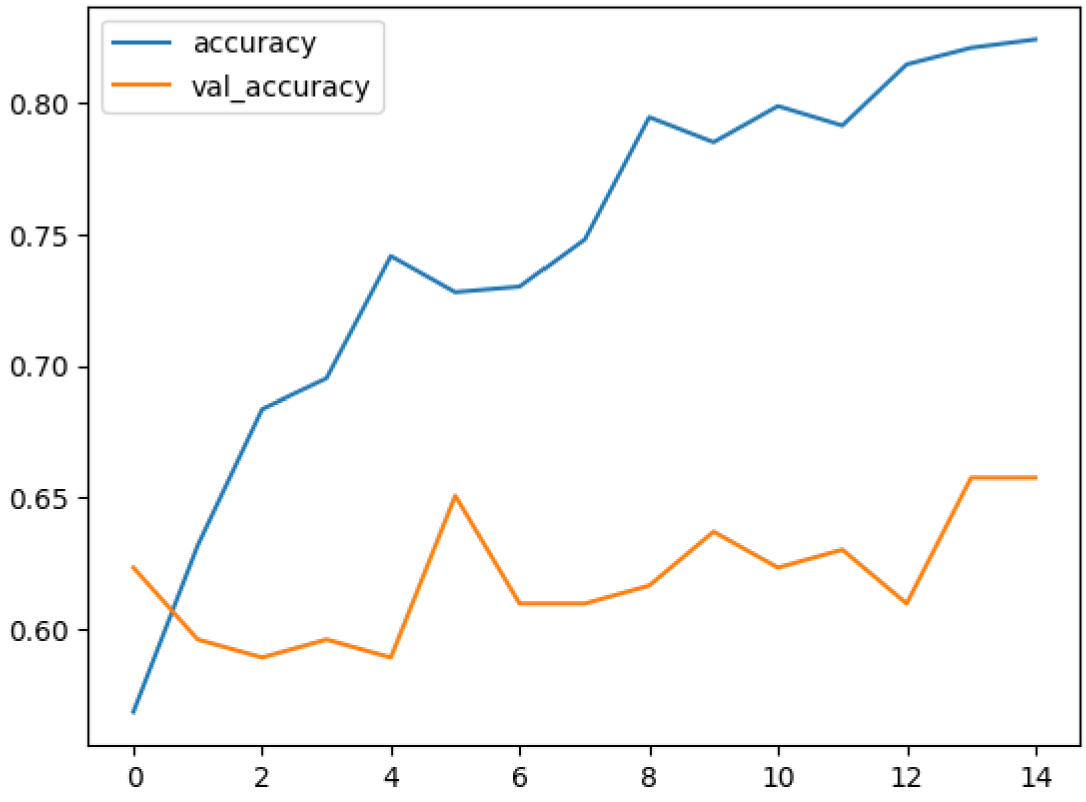
Out[ ]:

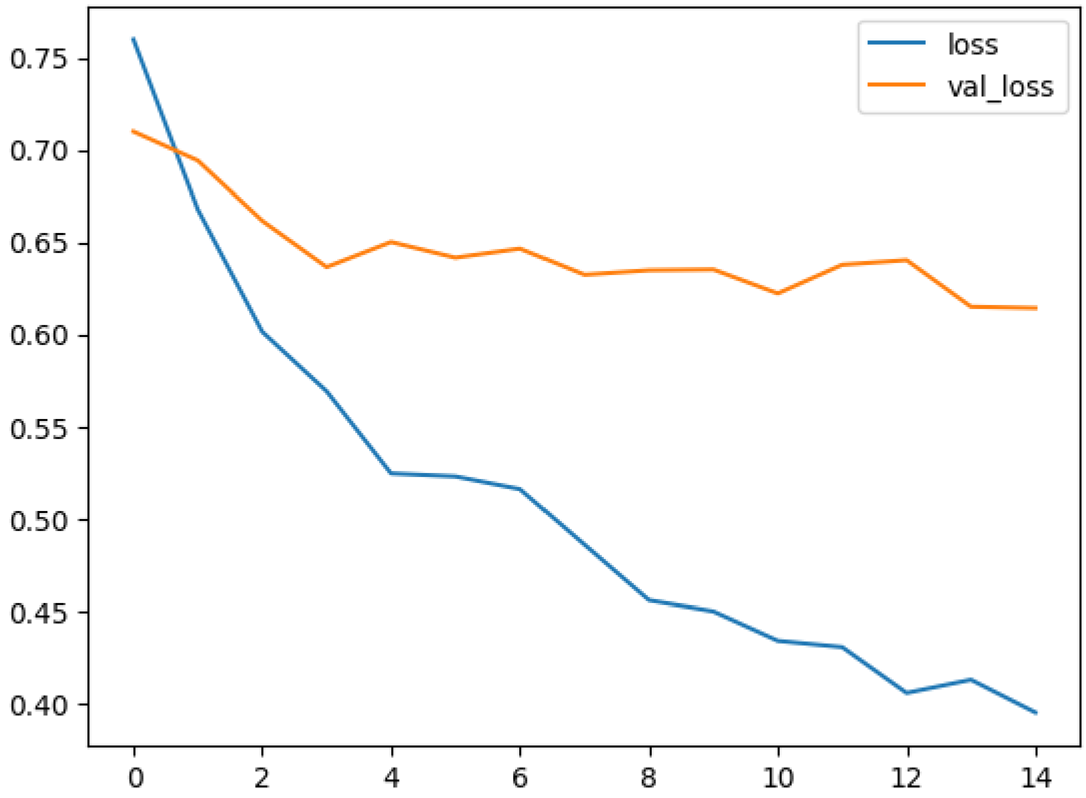
	loss	accuracy	val_loss	val_accuracy
0	0.759903	0.568254	0.709909	0.623288
1	0.667656	0.631746	0.694188	0.595890
2	0.601387	0.683598	0.661586	0.589041
3	0.569085	0.695238	0.636529	0.595890
4	0.524445	0.741799	0.650144	0.589041
5	0.522804	0.728042	0.641632	0.650685
6	0.515900	0.730159	0.646549	0.609589
7	0.485738	0.748148	0.632361	0.609589
8	0.455683	0.794709	0.634818	0.616438
9	0.449658	0.785185	0.635210	0.636986
10	0.433755	0.798942	0.622208	0.623288
11	0.430352	0.791534	0.637779	0.630137
12	0.405475	0.814815	0.640306	0.609589
13	0.412644	0.821164	0.615006	0.657534
14	0.394876	0.824339	0.614274	0.657534

In [ ]:

```
df = pd.DataFrame(history.history)

df[['accuracy', 'val_accuracy']].plot()
df[['loss', 'val_loss']].plot()
```

Out[ ]: <Axes: >





Modelo 3 ADICIONANDO MAIS UMA CADMADA DENSE E UMA DE DROPOUT

In [ ]:

```
modelo3 = Sequential([ base_model,
                     GlobalAveragePooling2D(),
                     Dense(128, activation='relu'),
                     Dropout(0.2),
                     Dense(64, activation='relu'),
                     Dropout(0.2),
                     Dense(2, activation='Softmax')
])

modelo3.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 resnet50 (Functional)       (None, 7, 7, 2048)        23587712  
                                                                 
 global_average_pooling2d_2   (None, 2048)             0         
 (GlobalAveragePooling2D)                                        
                                                                 
 dense_4 (Dense)             (None, 128)               262272    
                                                                 
 dropout_2 (Dropout)         (None, 128)               0         
                                                                 
 dense_5 (Dense)             (None, 64)                8256      
                                                                 
 dropout_3 (Dropout)         (None, 64)                0         
                                                                 
 dense_6 (Dense)             (None, 2)                 130       
                                                                 
=================================================================
Total params: 23,858,370
Trainable params: 270,658
Non-trainable params: 23,587,712
_________________________________________________________________
```

In [ ]:

```
modelo3.compile(optimizer=Adam(learning_rate=1e-4),
               loss='categorical_crossentropy',
               metrics=['accuracy']
               )

history3 = modelo3.fit(train_gen,
           validation_data=validation_gen,
           epochs=15,
           batch_size=16,
           callbacks=[tensorBoard]
        )
```

```
Epoch 1/15
60/60 [==============================] - 5s 54ms/step - loss: 0.7300 - accuracy: 0.5608 - val_loss: 0.6764 - val_accuracy: 0.56
85
Epoch 2/15
60/60 [==============================] - 3s 43ms/step - loss: 0.6484 - accuracy: 0.6254 - val_loss: 0.6762 - val_accuracy: 0.58
90
Epoch 3/15
60/60 [==============================] - 3s 43ms/step - loss: 0.6380 - accuracy: 0.6508 - val_loss: 0.6578 - val_accuracy: 0.62
33
Epoch 4/15
60/60 [==============================] - 3s 43ms/step - loss: 0.6046 - accuracy: 0.6783 - val_loss: 0.6514 - val_accuracy: 0.65
75
Epoch 5/15
60/60 [==============================] - 3s 43ms/step - loss: 0.5811 - accuracy: 0.6804 - val_loss: 0.6437 - val_accuracy: 0.66
44
Epoch 6/15
60/60 [==============================] - 3s 43ms/step - loss: 0.5639 - accuracy: 0.7090 - val_loss: 0.6384 - val_accuracy: 0.62
33
Epoch 7/15
60/60 [==============================] - 3s 44ms/step - loss: 0.5478 - accuracy: 0.7344 - val_loss: 0.6268 - val_accuracy: 0.67
12
Epoch 8/15
60/60 [==============================] - 3s 45ms/step - loss: 0.5150 - accuracy: 0.7439 - val_loss: 0.6172 - val_accuracy: 0.68
49
Epoch 9/15
60/60 [==============================] - 3s 45ms/step - loss: 0.5116 - accuracy: 0.7280 - val_loss: 0.6204 - val_accuracy: 0.67
12
Epoch 10/15
60/60 [==============================] - 3s 45ms/step - loss: 0.5142 - accuracy: 0.7397 - val_loss: 0.6311 - val_accuracy: 0.65
07
Epoch 11/15
60/60 [==============================] - 3s 45ms/step - loss: 0.4959 - accuracy: 0.7492 - val_loss: 0.6150 - val_accuracy: 0.66
44
Epoch 12/15
60/60 [==============================] - 3s 45ms/step - loss: 0.4790 - accuracy: 0.7661 - val_loss: 0.6538 - val_accuracy: 0.60
96
Epoch 13/15
60/60 [==============================] - 3s 45ms/step - loss: 0.4509 - accuracy: 0.7778 - val_loss: 0.6170 - val_accuracy: 0.66
44
Epoch 14/15
60/60 [==============================] - 3s 46ms/step - loss: 0.4525 - accuracy: 0.7915 - val_loss: 0.6138 - val_accuracy: 0.66
44
Epoch 15/15
60/60 [==============================] - 3s 46ms/step - loss: 0.4497 - accuracy: 0.7788 - val_loss: 0.6489 - val_accuracy: 0.61
64
```

In [ ]:

```
pd.DataFrame(history3.history)
```

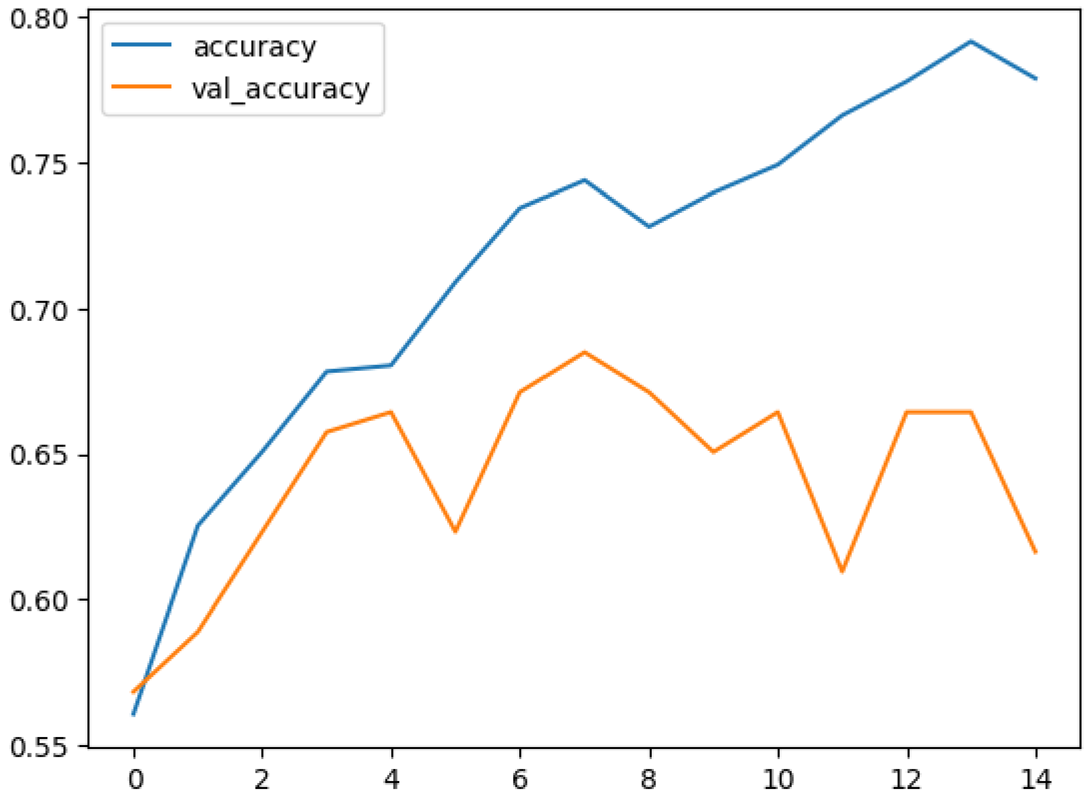
Out[ ]:

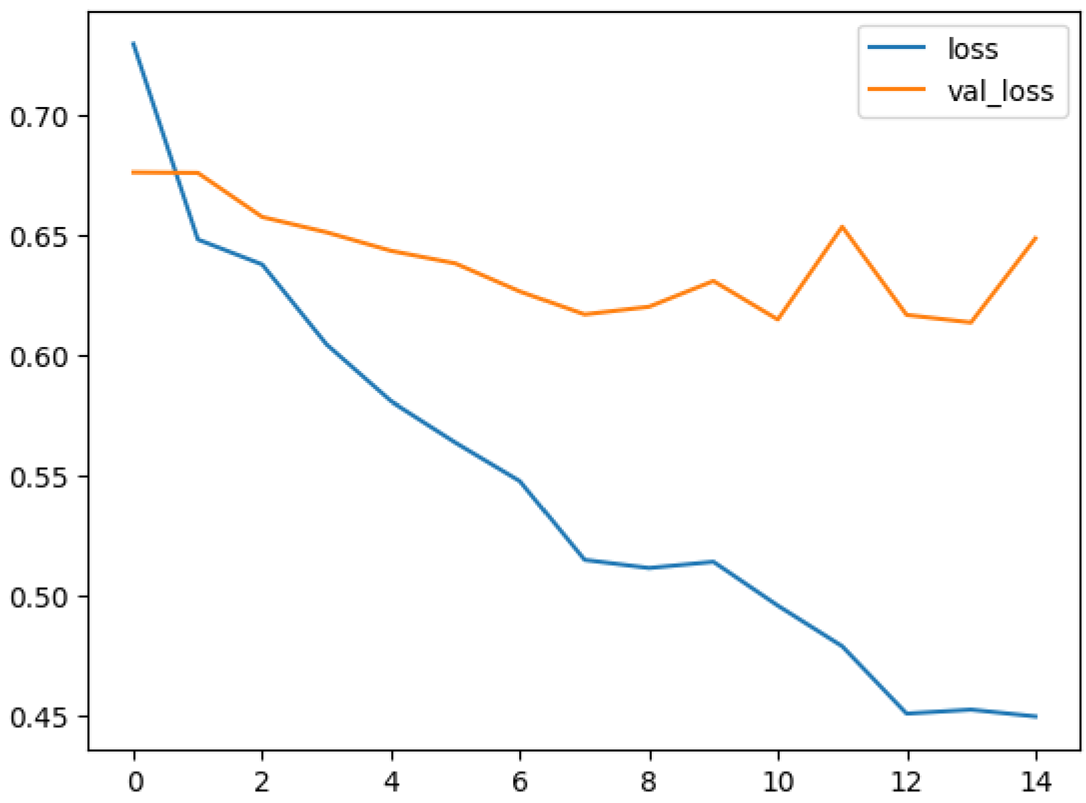
	loss	accuracy	val_loss	val_accuracy
0	0.730000	0.560847	0.676440	0.568493
1	0.648435	0.625397	0.676223	0.589041
2	0.638014	0.650794	0.657828	0.623288
3	0.604642	0.678307	0.651429	0.657534
4	0.581074	0.680423	0.643710	0.664384
5	0.563901	0.708995	0.638448	0.623288
6	0.547844	0.734392	0.626763	0.671233
7	0.515010	0.743915	0.617181	0.684932
8	0.511602	0.728042	0.620384	0.671233
9	0.514208	0.739683	0.631126	0.650685
10	0.495924	0.749206	0.615044	0.664384
11	0.478982	0.766138	0.653758	0.609589
12	0.450874	0.777778	0.616974	0.664384
13	0.452535	0.791534	0.613810	0.664384
14	0.449691	0.778836	0.648917	0.616438

In [ ]:

```
df = pd.DataFrame(history3.history)

df[['accuracy', 'val_accuracy']].plot()
df[['loss', 'val_loss']].plot()
```

Out[ ]: <Axes: >





Adicionando ao modelo 3 mais uma dense e de dropout

In [ ]:
```
modelo4 = Sequential([ base_model,
                     GlobalAveragePooling2D(),
                     Dense(128, activation='relu'),
                     Dropout(0.4),
                     Dense(64, activation='relu'),
                     Dropout(0.2),
                     Dense(32, activation='relu'),
                     Dropout(0.2),
                     Dense(2, activation='Softmax')
])

modelo4.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 7, 7, 2048)        23587712

 global_average_pooling2d_3   (None, 2048)             0
 (GlobalAveragePooling2D)

 dense_7 (Dense)             (None, 128)               262272

 dropout_4 (Dropout)         (None, 128)               0

 dense_8 (Dense)             (None, 64)                8256

 dropout_5 (Dropout)         (None, 64)                0

 dense_9 (Dense)             (None, 32)                2080

 dropout_6 (Dropout)         (None, 32)                0

 dense_10 (Dense)            (None, 2)                 66

=================================================================
Total params: 23,860,386
Trainable params: 272,674
Non-trainable params: 23,587,712
_________________________________________________________________
```

In [ ]:
```
modelo4.compile(optimizer=Adam(learning_rate=1e-4),
               loss='categorical_crossentropy',
               metrics=['accuracy']
               )

history3 = modelo4.fit(train_gen,
           validation_data=validation_gen,
           epochs=15,
           batch_size=16,
           callbacks=[tensorBoard]
        )
```

```
Epoch 1/15
60/60 [==============================] - 6s 54ms/step - loss: 0.8431 - accuracy: 0.4815 - val_loss: 0.6737 - val_accuracy: 0.56
85
Epoch 2/15
60/60 [==============================] - 3s 43ms/step - loss: 0.7237 - accuracy: 0.5545 - val_loss: 0.6704 - val_accuracy: 0.58
90
Epoch 3/15
60/60 [==============================] - 3s 44ms/step - loss: 0.6861 - accuracy: 0.5947 - val_loss: 0.6666 - val_accuracy: 0.61
64
Epoch 4/15
60/60 [==============================] - 3s 44ms/step - loss: 0.6935 - accuracy: 0.5810 - val_loss: 0.6657 - val_accuracy: 0.59
59
Epoch 5/15
60/60 [==============================] - 3s 43ms/step - loss: 0.6719 - accuracy: 0.6000 - val_loss: 0.6692 - val_accuracy: 0.55
48
Epoch 6/15
60/60 [==============================] - 3s 43ms/step - loss: 0.6740 - accuracy: 0.6074 - val_loss: 0.6648 - val_accuracy: 0.60
96
Epoch 7/15
60/60 [==============================] - 3s 44ms/step - loss: 0.6414 - accuracy: 0.6296 - val_loss: 0.6570 - val_accuracy: 0.59
59
Epoch 8/15
60/60 [==============================] - 3s 44ms/step - loss: 0.6483 - accuracy: 0.6265 - val_loss: 0.6450 - val_accuracy: 0.62
33
Epoch 9/15
60/60 [==============================] - 3s 44ms/step - loss: 0.6363 - accuracy: 0.6360 - val_loss: 0.6502 - val_accuracy: 0.63
01
Epoch 10/15
60/60 [==============================] - 3s 44ms/step - loss: 0.6316 - accuracy: 0.6476 - val_loss: 0.6418 - val_accuracy: 0.62
33
Epoch 11/15
60/60 [==============================] - 3s 45ms/step - loss: 0.6149 - accuracy: 0.6508 - val_loss: 0.6392 - val_accuracy: 0.66
44
Epoch 12/15
60/60 [==============================] - 3s 45ms/step - loss: 0.6045 - accuracy: 0.6794 - val_loss: 0.6337 - val_accuracy: 0.64
38
Epoch 13/15
60/60 [==============================] - 3s 46ms/step - loss: 0.5871 - accuracy: 0.6741 - val_loss: 0.6276 - val_accuracy: 0.65
75
Epoch 14/15
60/60 [==============================] - 3s 45ms/step - loss: 0.5882 - accuracy: 0.6815 - val_loss: 0.6213 - val_accuracy: 0.65
07
Epoch 15/15
60/60 [==============================] - 3s 45ms/step - loss: 0.5706 - accuracy: 0.6952 - val_loss: 0.6262 - val_accuracy: 0.65
75
```

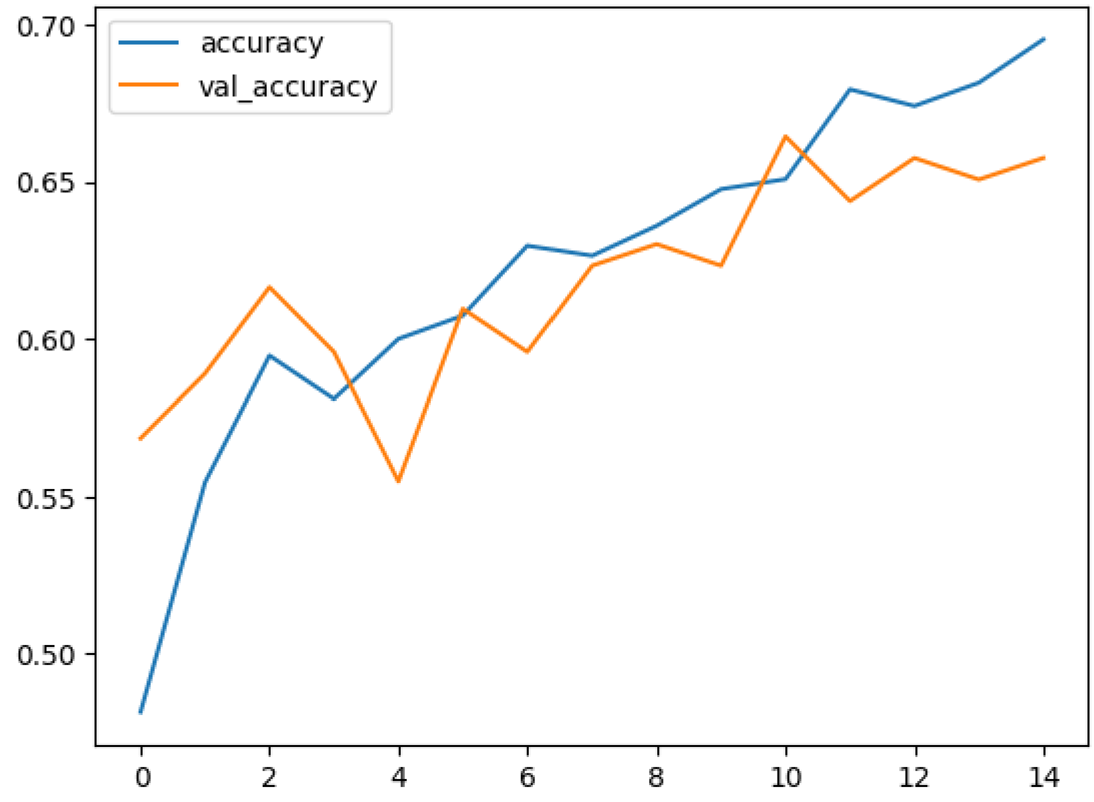
In [ ]:
```
pd.DataFrame(history3.history)
```

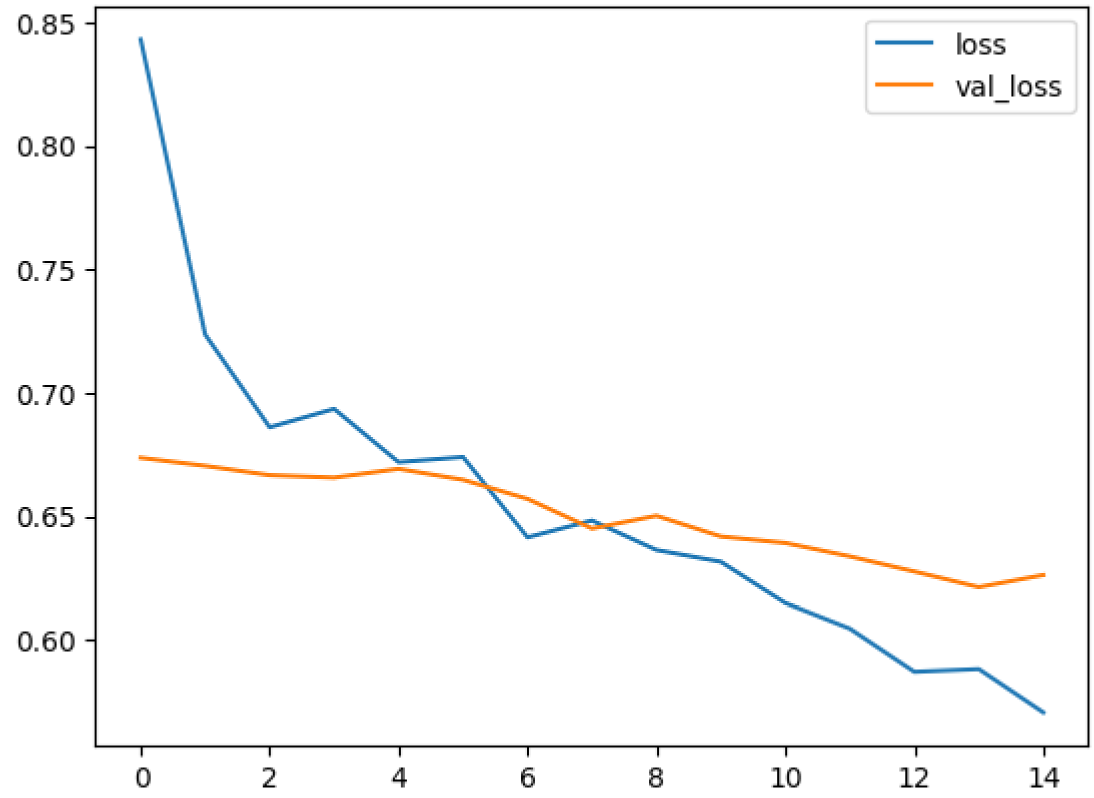
Out[ ]:

	loss	accuracy	val_loss	val_accuracy
0	0.843121	0.481481	0.673684	0.568493
1	0.723682	0.554497	0.670413	0.589041
2	0.686067	0.594709	0.666642	0.616438
3	0.693518	0.580952	0.665650	0.595890
4	0.671918	0.600000	0.669199	0.554795
5	0.673999	0.607407	0.664767	0.609589
6	0.641436	0.629630	0.657030	0.595890
7	0.648265	0.626455	0.645025	0.623288
8	0.636314	0.635979	0.650180	0.630137
9	0.631558	0.647619	0.641797	0.623288
10	0.614887	0.650794	0.639178	0.664384
11	0.604542	0.679365	0.633729	0.643836
12	0.587087	0.674074	0.627609	0.657534
13	0.588198	0.681481	0.621289	0.650685
14	0.570622	0.695238	0.626174	0.657534

In [ ]:
```
df = pd.DataFrame(history3.history)

df[['accuracy', 'val_accuracy']].plot()
df[['loss', 'val_loss']].plot()
```

Out[ ]: <Axes: >





Aumentando o bach size com base no modelo 3

In [ ]:
```
datagen_resnet = ImageDataGenerator(preprocessing_function=preprocess_input)

train_gen_32 = datagen_resnet.flow_from_directory('DATASETS/Training-Mass/',
                            target_size=(224,224),
                            class_mode="categorical",
                            batch_size=32
                            )


validation_gen_32 = datagen_resnet.flow_from_directory('DATASETS/Validation-Mass/',
                            target_size=(224,224),
                            class_mode="categorical",
                            batch_size=32
                            )
```

```
Found 945 images belonging to 2 classes.
Found 146 images belonging to 2 classes.
```

In [ ]:
```
modelo_op = Sequential([ base_model,
                     GlobalAveragePooling2D(),
                     Dense(128, activation='relu'),
                     Dropout(0.2),
                     Dense(64, activation='relu'),
                     Dropout(0.2),
                     Dense(2, activation='Softmax')
])

modelo_op.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 7, 7, 2048)        23587712

 global_average_pooling2d_4   (None, 2048)             0
 (GlobalAveragePooling2D)

 dense_11 (Dense)            (None, 128)               262272

 dropout_7 (Dropout)         (None, 128)               0

 dense_12 (Dense)            (None, 64)                8256

 dropout_8 (Dropout)         (None, 64)                0

 dense_13 (Dense)            (None, 2)                 130

=================================================================
Total params: 23,858,370
Trainable params: 270,658
Non-trainable params: 23,587,712
_________________________________________________________________
```

In [ ]:

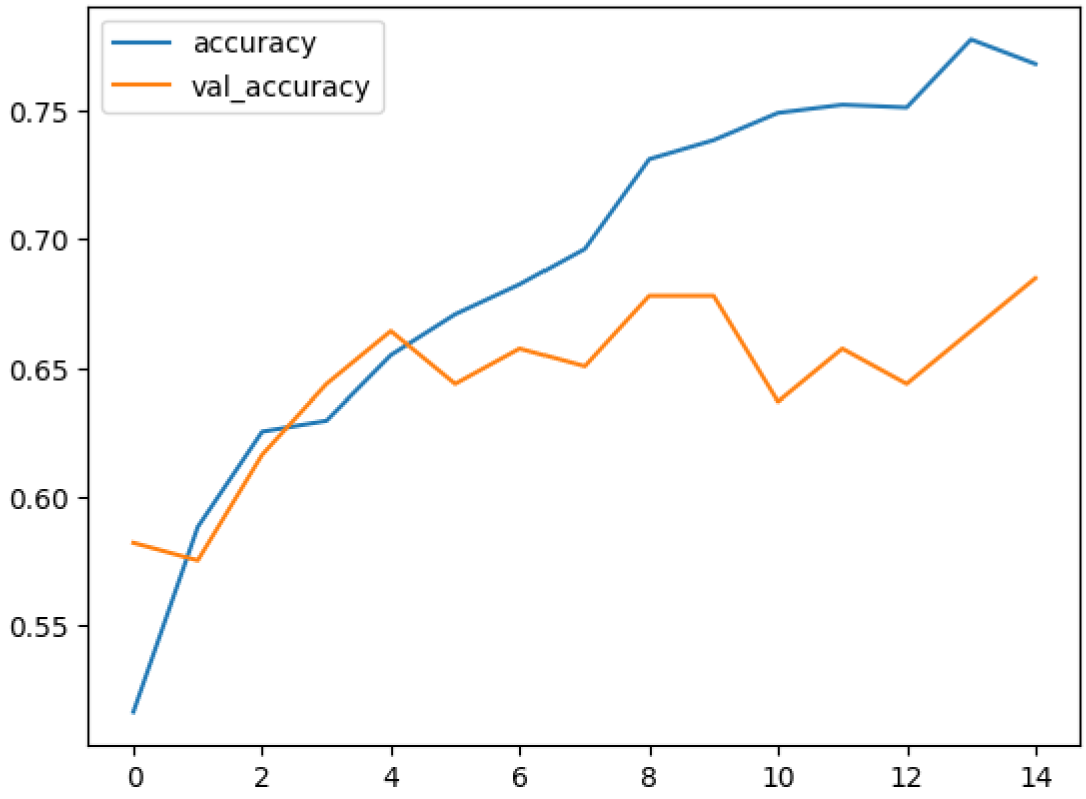
```
modelo_op.compile(optimizer=Adam(learning_rate=1e-4),
               loss='categorical_crossentropy',
               metrics=['accuracy']
               )

history_op = modelo_op.fit(train_gen_32,
           validation_data=validation_gen_32,
           epochs=15,
           batch_size=32,
           callbacks=[tensorBoard]
        )
```

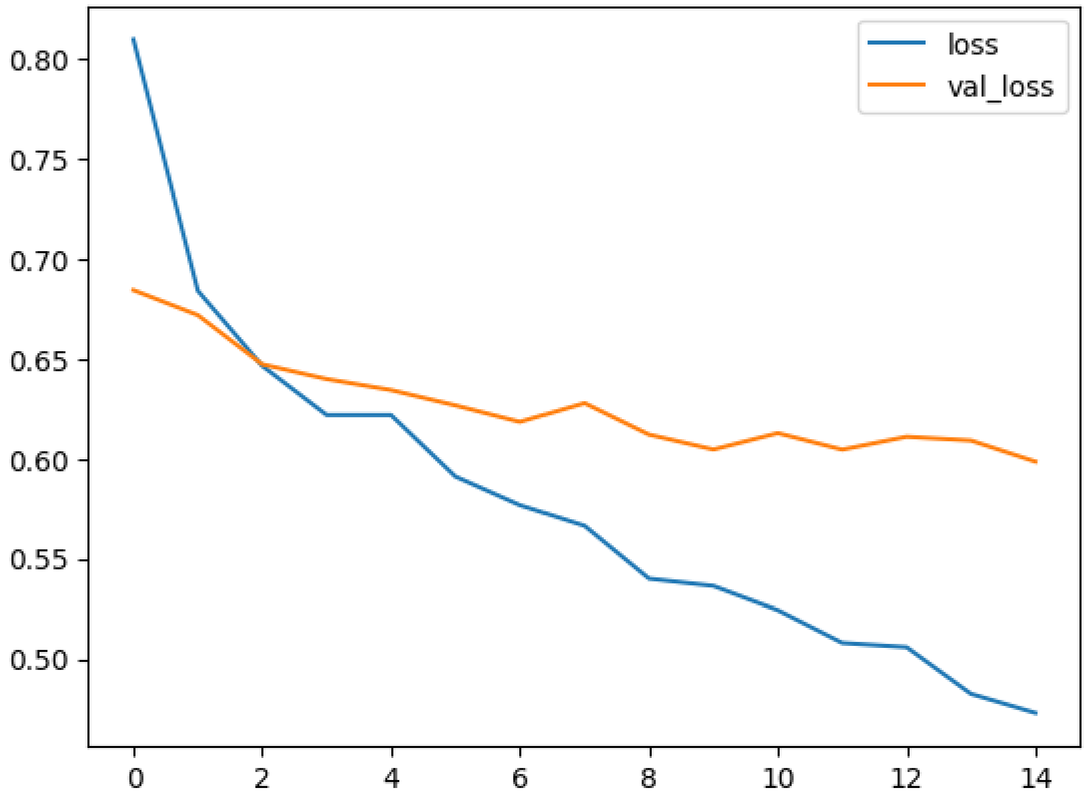
```
Epoch 1/15
30/30 [==============================] - 5s 99ms/step - loss: 0.8098 - accuracy: 0.5164 - val_loss: 0.6845 - val_accuracy: 0.58
22
Epoch 2/15
30/30 [==============================] - 2s 71ms/step - loss: 0.6842 - accuracy: 0.5884 - val_loss: 0.6720 - val_accuracy: 0.57
53
Epoch 3/15
30/30 [==============================] - 2s 70ms/step - loss: 0.6466 - accuracy: 0.6254 - val_loss: 0.6474 - val_accuracy: 0.61
64
Epoch 4/15
30/30 [==============================] - 2s 71ms/step - loss: 0.6220 - accuracy: 0.6296 - val_loss: 0.6400 - val_accuracy: 0.64
38
Epoch 5/15
30/30 [==============================] - 2s 70ms/step - loss: 0.6220 - accuracy: 0.6550 - val_loss: 0.6345 - val_accuracy: 0.66
44
Epoch 6/15
30/30 [==============================] - 2s 77ms/step - loss: 0.5912 - accuracy: 0.6709 - val_loss: 0.6268 - val_accuracy: 0.64
38
Epoch 7/15
30/30 [==============================] - 2s 72ms/step - loss: 0.5768 - accuracy: 0.6825 - val_loss: 0.6185 - val_accuracy: 0.65
75
Epoch 8/15
30/30 [==============================] - 2s 82ms/step - loss: 0.5665 - accuracy: 0.6963 - val_loss: 0.6279 - val_accuracy: 0.65
07
Epoch 9/15
30/30 [==============================] - 2s 71ms/step - loss: 0.5401 - accuracy: 0.7312 - val_loss: 0.6122 - val_accuracy: 0.67
81
Epoch 10/15
30/30 [==============================] - 2s 72ms/step - loss: 0.5364 - accuracy: 0.7386 - val_loss: 0.6047 - val_accuracy: 0.67
81
Epoch 11/15
30/30 [==============================] - 2s 72ms/step - loss: 0.5244 - accuracy: 0.7492 - val_loss: 0.6129 - val_accuracy: 0.63
70
Epoch 12/15
30/30 [==============================] - 2s 71ms/step - loss: 0.5080 - accuracy: 0.7524 - val_loss: 0.6046 - val_accuracy: 0.65
75
Epoch 13/15
30/30 [==============================] - 2s 71ms/step - loss: 0.5059 - accuracy: 0.7513 - val_loss: 0.6111 - val_accuracy: 0.64
38
Epoch 14/15
30/30 [==============================] - 2s 72ms/step - loss: 0.4825 - accuracy: 0.7778 - val_loss: 0.6092 - val_accuracy: 0.66
44
Epoch 15/15
30/30 [==============================] - 2s 78ms/step - loss: 0.4730 - accuracy: 0.7683 - val_loss: 0.5986 - val_accuracy: 0.68
49
```

In [ ]:

```
df = pd.DataFrame(history_op.history)

df[['accuracy', 'val_accuracy']].plot()
df[['loss', 'val_loss']].plot()
```

Out[ ]: <Axes: >





reduzindo epocas do modelo op de 15 para 9

In [ ]:
```
modelo_nove = Sequential([ base_model,
                     GlobalAveragePooling2D(),
                     Dense(128, activation='relu'),
                     Dropout(0.2),
                     Dense(64, activation='relu'),
                     Dropout(0.2),
                     Dense(2, activation='Softmax')
])

modelo_nove.summary()
```

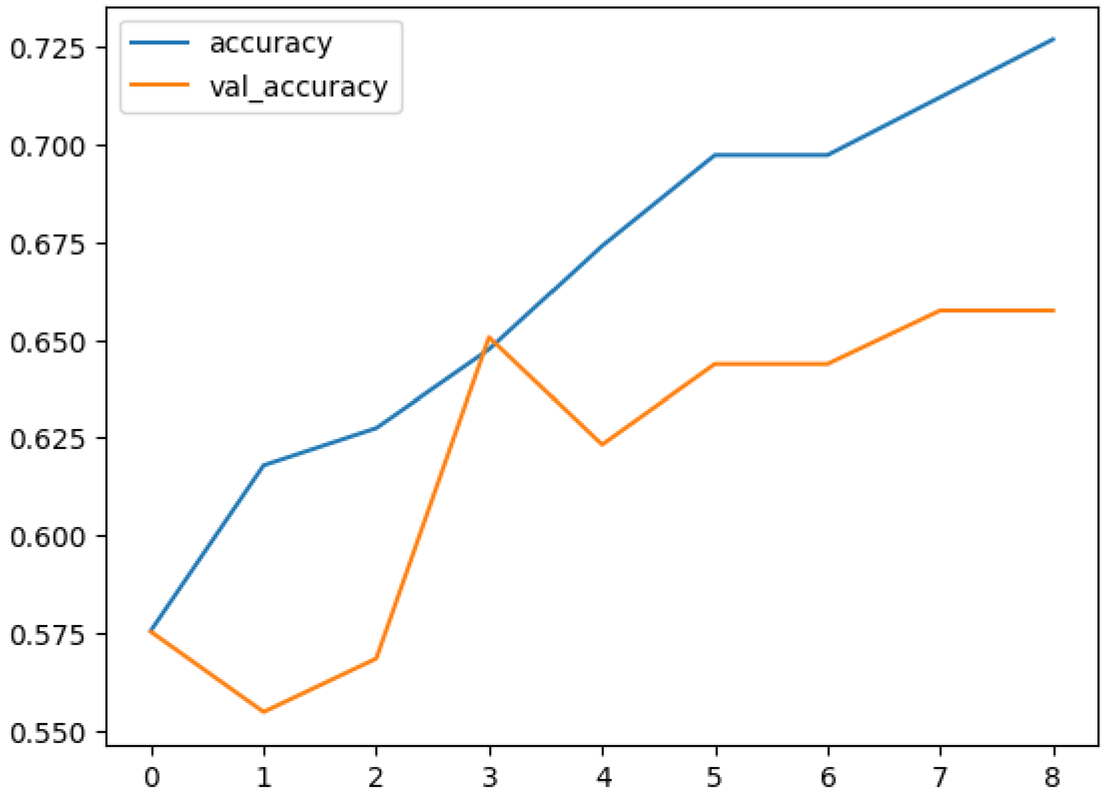
```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 resnet50 (Functional)       (None, 7, 7, 2048)        23587712  
                                                                 
 global_average_pooling2d_5   (None, 2048)             0         
 (GlobalAveragePooling2D)                                        
                                                                 
 dense_14 (Dense)            (None, 128)               262272    
                                                                 
 dropout_9 (Dropout)         (None, 128)               0         
                                                                 
 dense_15 (Dense)            (None, 64)                8256      
                                                                 
 dropout_10 (Dropout)        (None, 64)                0         
                                                                 
 dense_16 (Dense)            (None, 2)                 130       
                                                                 
=================================================================
Total params: 23,858,370
Trainable params: 270,658
Non-trainable params: 23,587,712
_________________________________________________________________
```

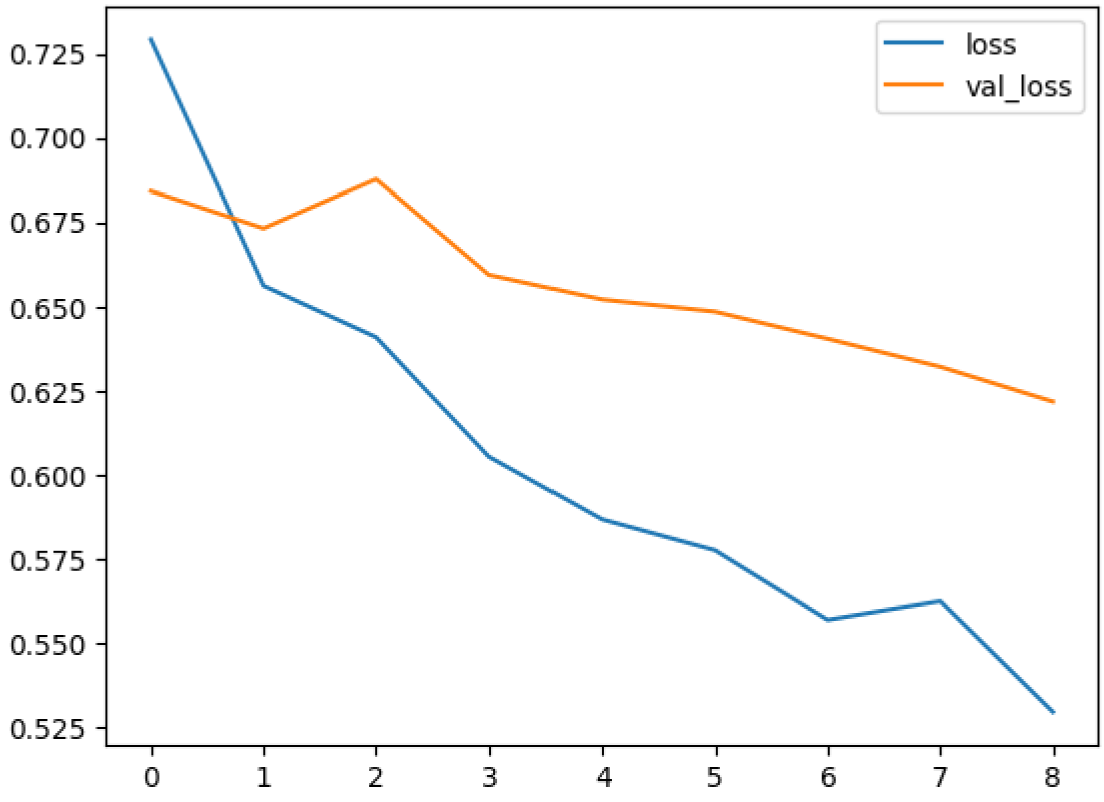
In [ ]:
```
modelo_nove.compile(optimizer=Adam(learning_rate=1e-4),
               loss='categorical_crossentropy',
               metrics=['accuracy']
               )

history_nove = modelo_nove.fit(train_gen_32,
           validation_data=validation_gen_32,
           epochs=9,
           batch_size=32,
           callbacks=[tensorBoard]
        )
```

```
Epoch 1/9
30/30 [==============================] - 5s 94ms/step - loss: 0.7292 - accuracy: 0.5757 - val_loss: 0.6842 - val_accuracy: 0.57
53
Epoch 2/9
30/30 [==============================] - 2s 70ms/step - loss: 0.6561 - accuracy: 0.6180 - val_loss: 0.6731 - val_accuracy: 0.55
48
Epoch 3/9
30/30 [==============================] - 2s 72ms/step - loss: 0.6408 - accuracy: 0.6275 - val_loss: 0.6877 - val_accuracy: 0.56
85
Epoch 4/9
30/30 [==============================] - 2s 70ms/step - loss: 0.6053 - accuracy: 0.6476 - val_loss: 0.6593 - val_accuracy: 0.65
07
Epoch 5/9
30/30 [==============================] - 2s 71ms/step - loss: 0.5867 - accuracy: 0.6741 - val_loss: 0.6521 - val_accuracy: 0.62
33
Epoch 6/9
30/30 [==============================] - 2s 70ms/step - loss: 0.5775 - accuracy: 0.6974 - val_loss: 0.6484 - val_accuracy: 0.64
38
Epoch 7/9
30/30 [==============================] - 2s 70ms/step - loss: 0.5568 - accuracy: 0.6974 - val_loss: 0.6404 - val_accuracy: 0.64
38
Epoch 8/9
30/30 [==============================] - 2s 70ms/step - loss: 0.5624 - accuracy: 0.7122 - val_loss: 0.6321 - val_accuracy: 0.65
75
Epoch 9/9
30/30 [==============================] - 2s 75ms/step - loss: 0.5295 - accuracy: 0.7270 - val_loss: 0.6217 - val_accuracy: 0.65
75
```

In [ ]:
```
pd2 = pd.DataFrame(history_nove.history)

pd2[['accuracy', 'val_accuracy']].plot()
pd2[['loss', 'val_loss']].plot()
```

Out[ ]: <Axes: >





In [ ]:
```
predictions = modelo4.predict(test_gen, verbose=1)
```

```
29/29 [==============================] - 1s 44ms/step
```

import numpy as np

labels = (test_gen.class_indices) res = tf.math.confusion_matrix(labels=test_gen.classes.astype(int), predictions=np.argmax(np.array(predictions).reshape(-1, 2), axis=1))

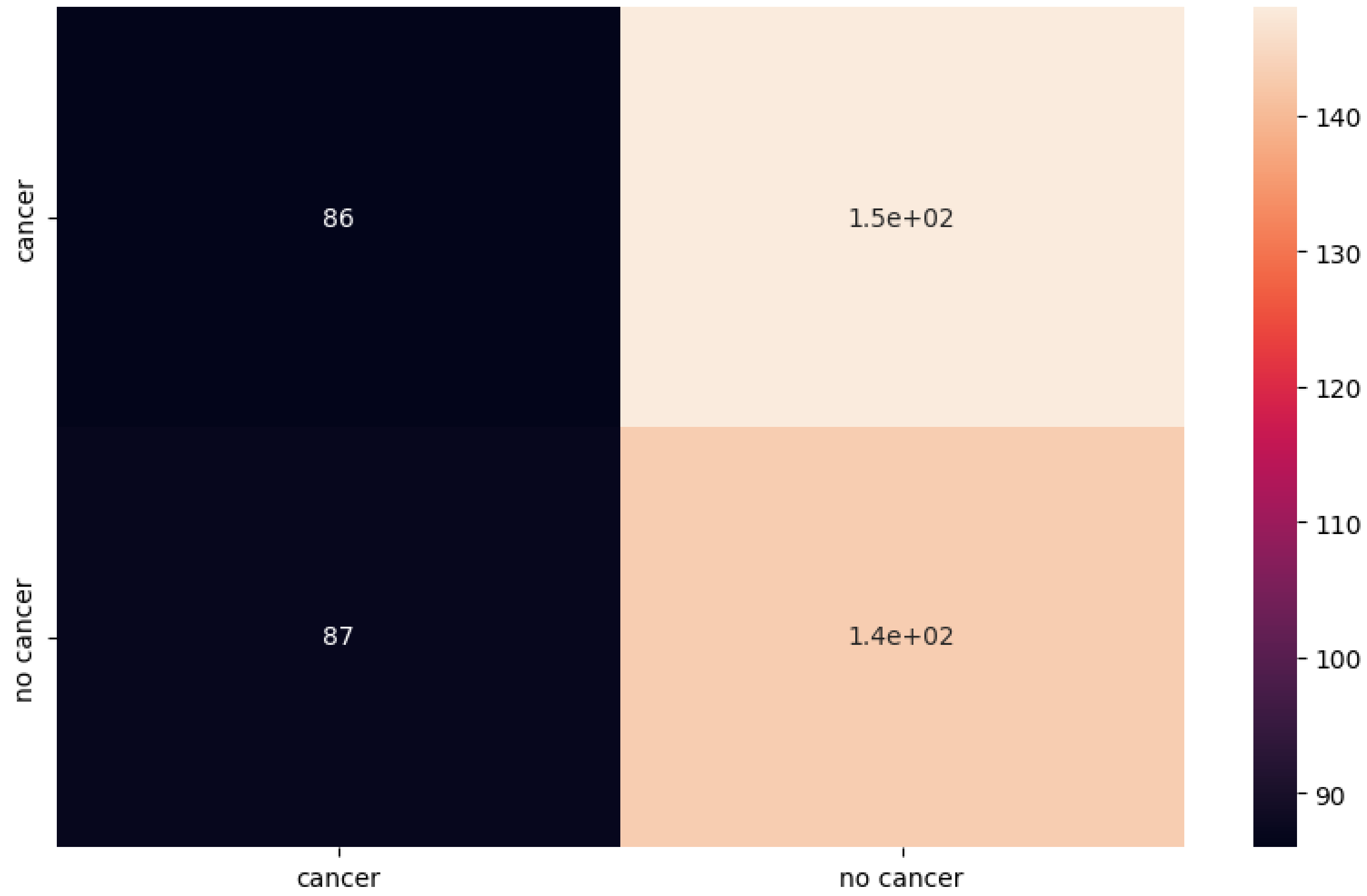
In [ ]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
classes_name = ('cancer', 'sem cancer')
height, width = (224, 224)


test_set = test_gen
test_set.reset()
y_pred = np.argmax(predictions, axis=-1)

y_test = test_gen.labels
cm = confusion_matrix(y_test,y_pred)

print(cm)

cm2 = classification_report(test_gen.classes, y_pred)

print(classification_report(test_gen.classes, y_pred))


file = open('time_bin_metrics.txt', 'w')
file.write('tempo %s' % cm2)
file.close()
```

```
[[ 86 148]
 [ 87 143]]
              precision    recall  f1-score   support

           0       0.50      0.37      0.42       234
           1       0.49      0.62      0.55       230

    accuracy                           0.49       464
   macro avg       0.49      0.49      0.49       464
weighted avg       0.49      0.49      0.49       464
```

In [ ]:
```python
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
print(cm)
index = ['cancer','no cancer']
columns = ['cancer','no cancer']
cm_df = pd.DataFrame(cm,columns,index)
plt.figure(figsize=(10,6))
sns.heatmap(cm_df, annot=True)
```

```
[[ 86 148]
 [ 87 143]]
```

Out[ ]: <Axes: >



In [ ]:
```python
import numpy as np
```

In [ ]:
```python
def predicao(modelo, path):
    image = Image.open(path)

# Redimensione a imagem
    resized_image = image.resize((224, 224))

    # Certifique-se de que a imagem seja colorida (3 canais)
    if resized_image.mode == 'L':
        # Converta a imagem em escala de cinza em uma imagem RGB (colorida)
        resized_image = resized_image.convert('RGB')

    # Converta a imagem redimensionada em uma matriz NumPy
    np_array = np.array(resized_image)


    img_np = preprocess_input(np_array)
    imp_np2=img_np.reshape(1,224,224,3)
    result = modelo2.predict(imp_np2)
    id_max= result[0].argmax()
    index_to_class = {v: k for k, v in train_gen.class_indices.items()}
    plt.title(f'Resultado: {index_to_class[id_max]}')
    plt.imshow(resized_image)
```

In [ ]:
```python
predicao(modelo,'DATASETS/Validation-Mass/WITH CANCER/2-233.jpg')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
c:\Users\yagok\OneDrive\Área de Trabalho\Resnet50\Breast-Cancer-Detection-AI-System\TCC2_MODELS.ipynb Célula 45 line 1
----> <a href='vscode-notebook-cell:/c%3A/Users/yagok/OneDrive/%C3%81rea%20de%20Trabalho/Resnet50/Breast-Cancer-Detection-AI-System/TCC2_MODELS.ipynb#X62sZmlsZQ%3D%3D?line=0'>1</a> predicao(modelo,'DATASETS/Validation-Mass/WITH CANCER/2-233.jpg')

NameError: name 'modelo' is not defined
```

end