

## Relatório 28 - Trabalho Final do Bootcamp: monte você mesmo (III)

Lucas Augusto Nunes de Barros

### Descrição das atividades

#### 1. Introdução

O projeto consiste em um classificador binário que realiza a distinção de embalagens de fio dental cheia e vazias. A motivação inicial deste trabalho surgiu durante a faculdade, quando um professor apresentou um problema que ocorreu na linha de produção de certa indústria em que era necessário separar garrafas pelo volume de líquido.

Inicialmente a ideia da classificação binária de garrafas foi considerada, porém ao perceber que fotografar objetos grandes, muitas vezes contendo líquidos transparentes e sendo altamente reflexivos, ia causar bastante problemas na montagem de um dataset robusto o suficiente para treinar uma CNN.

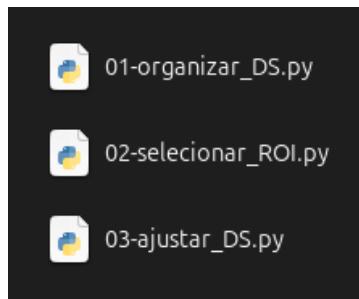
Por isso houve uma adaptação na embalagem que seria o foco do estudo, escolhendo embalagens de fio dental pelo seu tamanho reduzido, facilidade de transporte e manuseio, bem como um certa variedade de formatos e tamanhos a preços acessíveis (apesar de nem todas as embalagens serem transparentes).



Algumas embalagens usadas para montagem do dataset

## 2. Conjunto de Dados e Pré-Processamento das Imagens

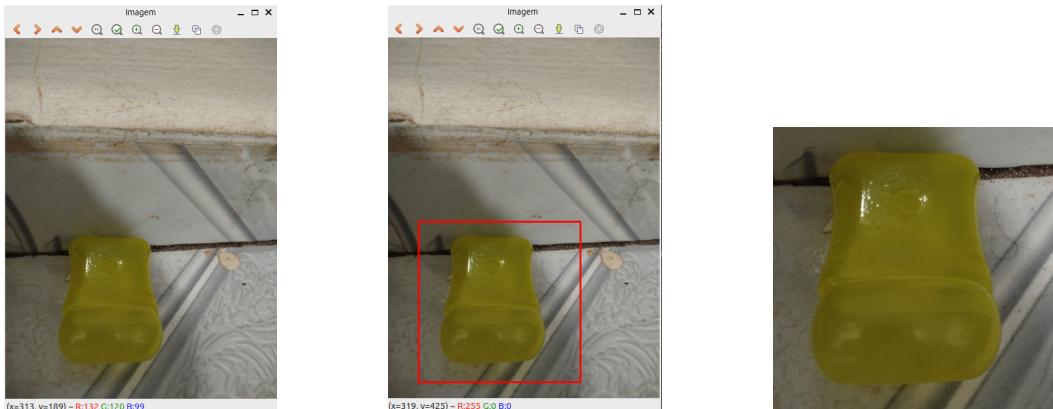
Para iniciar a montagem do dataset foram tiradas 4066 fotos, sendo 2081 da classe *empty* e 1985 da classe *full*. Para essa atividade foram utilizados três códigos em python que compõem o *pipeline* de dados da aplicação.



Para garantir que o modelo recebesse dados adequados, as imagens originais passaram por um *pipeline* de pré-processamento dividido em três etapas, utilizando os *scripts* localizados no diretório *data\_pipeline*.

O primeiro código do pré-processamento (*data\_pipeline/01-organizar\_DS.py*) renomeia as imagens da pasta de entrada e as transfere para a pasta de saída, permitindo organizar grandes conjuntos de dados aos lotes, continuando a partir do último índice localizado na pasta de saída.

As fotos originais continham muito ruído de fundo. Para que o modelo pudesse focar exclusivamente na embalagem, o segundo *script* (*data\_pipeline/02-selecionar\_ROI.py*) foi implementado para recortar a Região de Interesse (ROI - *Region of Interest*) de cada imagem.



Seleção manual da ROI através do script

Finalizando o *pipeline* de dados, o terceiro código (*data\_pipeline/03-ajustar\_DS.py*) faz a preparação das imagens recortadas das regiões de interesse, redimensionando as imagens para 128x128 e convertendo a imagem para escala de cinza, facilitando assim a importação e manipulação do *dataset*.



Redimensionando as Regiões de Interesse

Ao término do *pipeline* a base de dados estava pronta para uso, então foi dado prosseguimento com o treinamento da rede neural.

O *dataset* foi dividido em 75% treinamento, 15% teste e para 10% validação, no subconjunto utilizado para treinamento foi aplicada a técnica de *data augmentation* com o objetivo de aumentar a variabilidade das imagens, modificando o contraste e empregando rotações horizontais. Esse tipo de técnica tende a melhorar a capacidade de generalização do modelo, desde que utilize transformações plausíveis de ocorrer na vida real. Inicialmente foi aplicada, de maneira arbitrária, a configuração de *data augmentation* apresentada a seguir.

```
data_augmentation = keras.Sequential([
    layers.RandomFlip("horizontal"),      # espelhamento horizontal
    layers.RandomContrast(0.2)           # variação de contraste
])

train_ds = train_ds.map(lambda x, y: (data_augmentation(x, training=True), y))
```

### 3. Aplicando a Otimização Bayesiana

Para determinação dos valores aplicados à cada hiperparâmetro foi utilizada a técnica de otimização Bayesiana, que tem como base modelos probabilísticos que são usados para calcular o ponto ótimo da função de avaliação. Diferente de outras técnicas como o Grid Search, essa otimização não testa todas as combinações de hiperparâmetros, mas constrói um modelo probabilístico que é usado na busca pelos melhores valores, ou seja, para cada tentativa o modelo de probabilidade recalcula o novo valor para cada um dos hiperparâmetros analisados. Por padrão usou-se *batch size* igual a 32.

No caso em questão a otimização bayesiana foi aplicada com os seguintes intervalos de valores, eles foram escolhidos arbitrariamente com base em problemas similares presentes no próprio *bootcamp*:

Hiperparâmetro	Valor Mínimo	Valor Máximo	Passo
<b>Camadas convolucionais</b>	0	6	1
<b>Filtros por camada convolucional</b>	16	256	16
<b>Neurônios na camada densa</b>	16	256	16
<b>Taxa de dropout</b>	0	0.5	0.05
<b>Taxa de aprendizagem</b>	$10^{-5}$	$10^{-1}$	--

Das trinta combinações de hiperparâmetros feitas durante a aplicação da otimização bayesiana, os resultados das duas melhores foram utilizados como valores de referência.

Combinação	#7	#25
<b>Camadas convolucionais</b>	5	5
<b>Filtro 0</b>	160	224
<b>Filtro 1</b>	128	80
<b>Filtro 2</b>	144	240
<b>Filtro 3</b>	240	208
<b>Filtro 4</b>	112	48
<b>Neurônios na camada densa</b>	128	112
<b>Taxa de dropout</b>	0.45	0.25
<b>Taxa de aprendizagem</b>	3.79 e-05	8.02 e-05
<b>Acurácia</b>	0.8655	0.8623

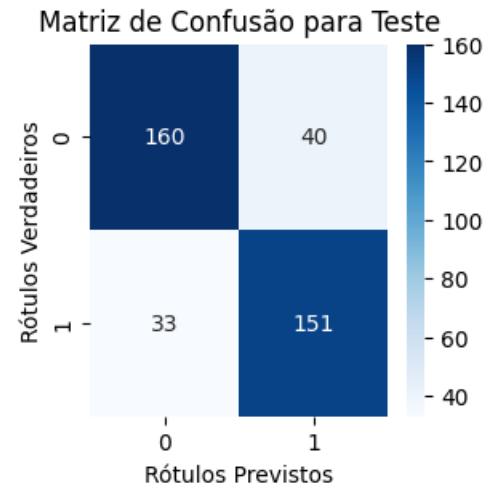
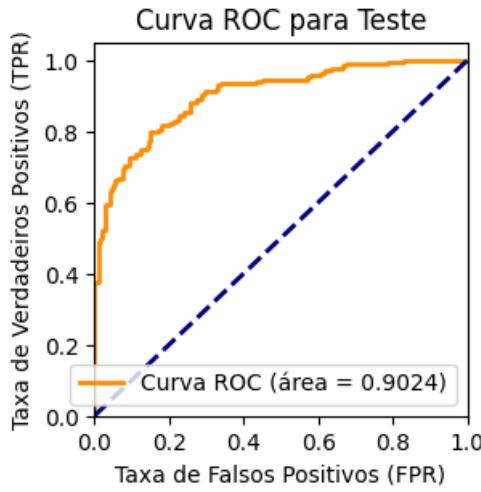
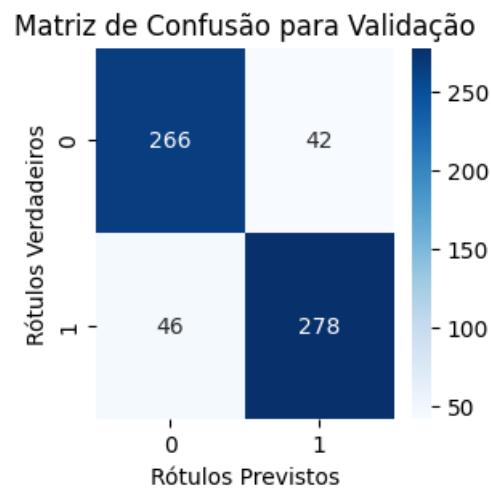
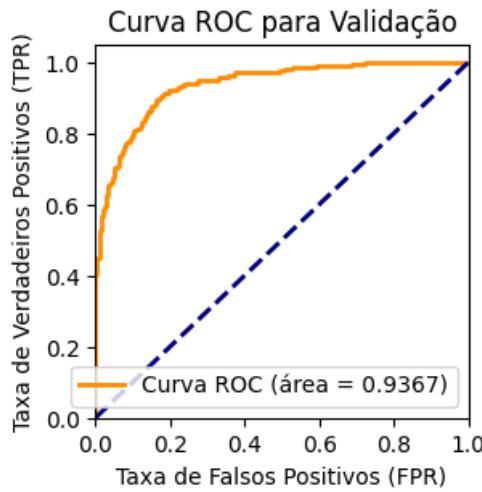
Com base nas informações foi aplicado o primeiro teste, utilizando a combinação #7 que apresentou a maior acurácia, com a finalidade de capturar os melhores pesos para o modelo durante um treinamento de 50 épocas. Para isso foi utilizada a função de *callback ModelCheckpoint* que permite salvar os pesos da melhor época durante o treinamento. Os resultados obtidos são apresentados abaixo.

Fica nítido durante a aplicação da otimização bayesiana, que a camada de dropout faz uma diferença significativa nas métricas do modelo após o treinamento, permitindo inclusive o treinamento por um número maior de épocas sem que ocorra *overfitting*. Essa camada elimina aleatoriamente uma porcentagem de neurônios durante o treinamento, evitando que a rede fique muito dependente deles, e consequentemente favorecendo um equilíbrio entre os todos neurônios.

Métricas da Estrutura #7 Apenas com Otimização Bayesiana

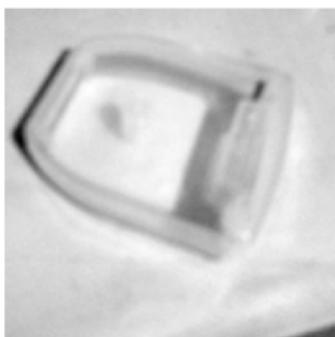
Métricas	Validação	Teste
Acurácia	0,8449	0,7995
Precisão	0,8522	0,8080
Recall	0,8468	0,7958
AUC	0,9367	0,9024

Informações Gráficas da Estrutura #7



### Imagens Classificadas Erroneamente pela Estrutura #7

V: [0.] | P: 1



V: [0.] | P: 1



V: [0.] | P: 1

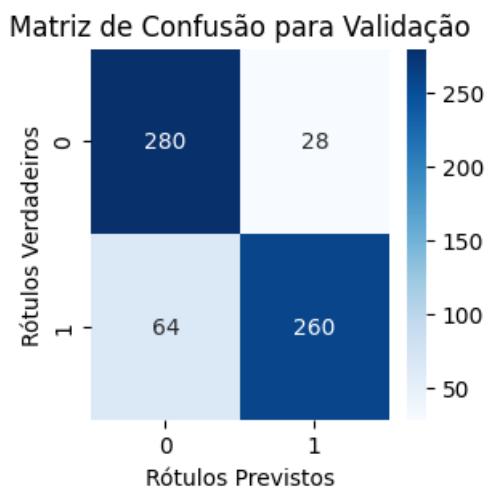
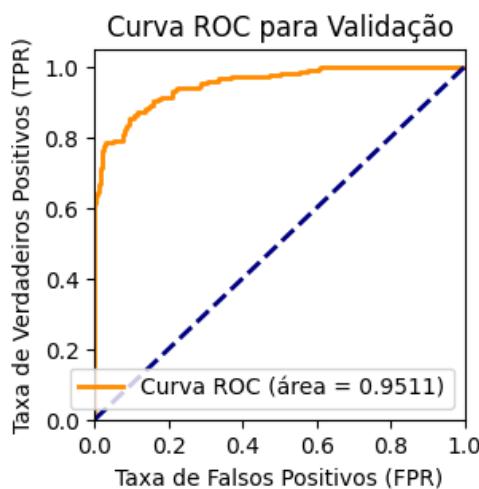


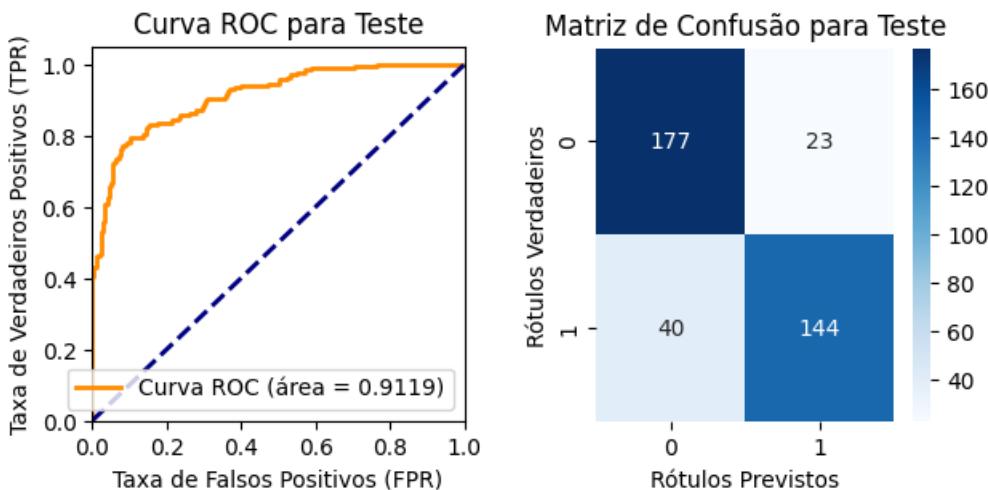
Para esse modelo a maior parte das classificações incorretas eram de fotos embaçadas e com a presença de rótulos.

### Métricas da Estrutura #25 Apenas com Otimização Bayesiana

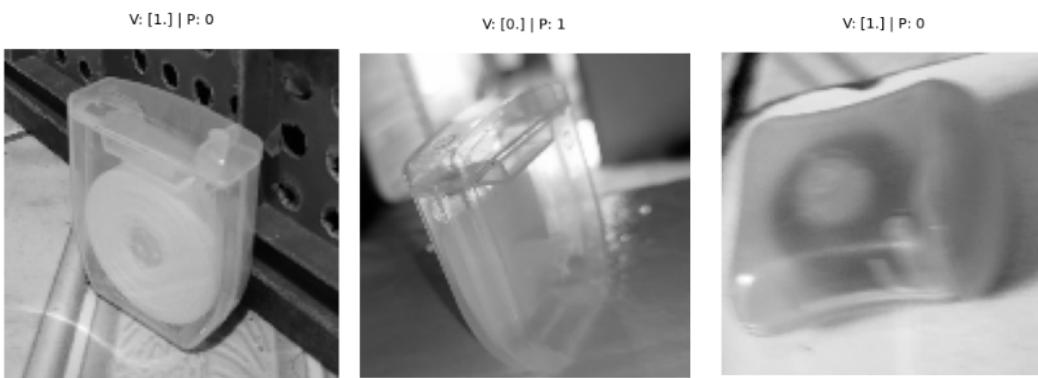
Métricas	Validação	Teste
Acurácia	0.8718	0.8359
Precisão	0.8814	0.8390
Recall	0.8739	0.8338
AUC	0.9511	0.9119

### Informações Gráficas da Estrutura #25





Imagens Classificadas Erroneamente pela Estrutura #25



Uma característica notável das imagens classificadas incorretamente por esse modelo é a angulação das embalagens, esse pareceu ser o fator mais significativo para a confusão do modelo.

#### 4. Refinando os Hiperparâmetros com Grid Search

Utilizando como referência os valores de hiperparâmetros obtidos nos melhores resultados através da técnica de otimização bayesiana, foram analisadas outras combinações de hiperparâmetros com intervalos menores, com o objetivo de afunilar os valores e tornar o modelo o mais eficiente possível.

Como o número de camadas não variou entre os melhores resultados de acurácia da otimização bayesiana, o modelo adotado foi o de 5 camadas convolucionais, utilizando as estruturas convolucionais das combinações #7 e #25 como referência, enquanto a taxa de aprendizado, taxa de dropout e quantidade de neurônios na camada densa foram reavaliadas com a técnica de Grid Search aplicando novos valores com base no intervalo observado.

Intervalos de Combinações para Aplicação da Técnica de Grid Search.

Hiperparâmetro	Valor Mínimo	Valor Máximo	Passo
Taxa de aprendizagem	4e-05	8e-05	1e-05
Neurônios na camada densa	96	144	16
Taxa de dropout	0.25	0.45	0.05

Os limites de cada hiperparâmetro foram escolhidos de forma que se mantivessem próximos dos valores obtidos através da otimização bayesiana (estrutura #7 e #25) e ainda houvesse uma margem de variação para testar valores mais precisos.

As melhores combinações de resultados utilizando a estrutura #7 são apresentadas abaixo.

Combinações com Maiores Valor de Acurácia Usando a Estrutura #7

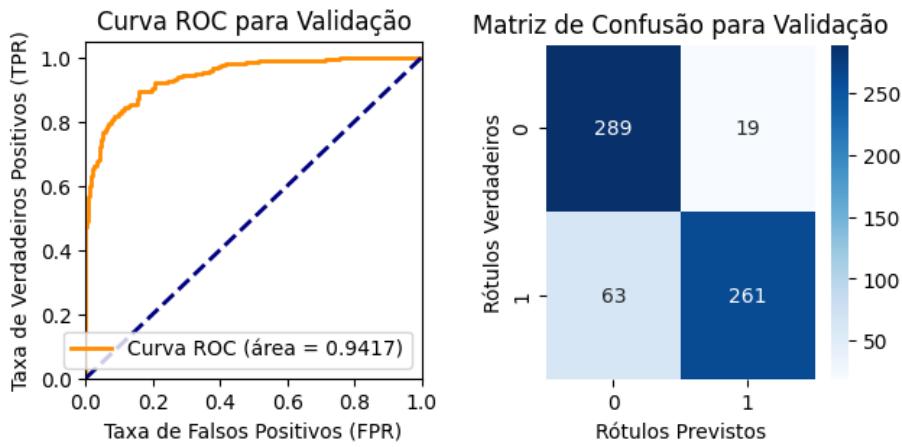
Combinação	Neurônios na camada densa	Taxa de dropout	Taxa de aprendizagem
1	112	0.25	6e-05
2	128	0.25	7e-05
3	112	0.30	7e-05
4	144	0.35	6e-05

De conhecimentos desses valores, foram feitos treinamentos de 50 épocas para cada uma das combinações listadas, com o objetivo de descobrir os melhores pesos possíveis para cada uma delas através da função *ModelCheckpoint*. Após o treinamento foram analisadas as métricas de acurácia, precisão, recall, AUC (*Area Under the Curve*).

Métricas De Validação das Melhores Combinações Usando a Estrutura #7

Combinação	Acurácia	Precisão	Recall	AUC
1	0.8601	0.8647	0.8570	0.9506
2	0.8466	0.8700	0.8453	0.9505
3	0.8703	0.8766	0.8719	0.9417
4	0.8641	0.8734	0.8709	0.9654

De posse dessas informações foi feita a seleção da combinação que obteve melhores resultados. Para essa combinação em específico foram plotadas a curva ROC (*Receiver Operating Characteristic*) e a matriz de confusão.



A mesma análise foi realizada com estrutura #25. As melhores combinações de acordo com o Grid Search, são apresentadas abaixo.

Combinações com Maiores Valor de Acurácia Usando a Estrutura #25

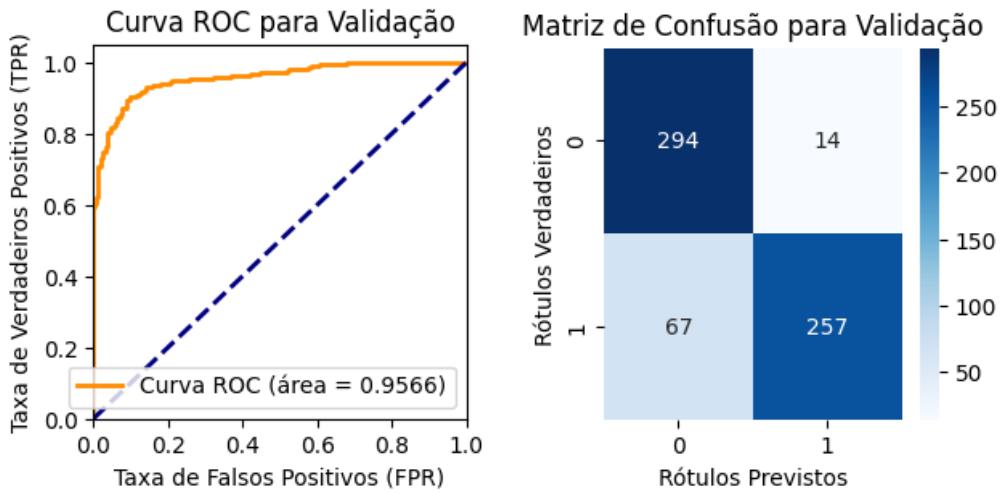
Combinação	Neurônios na camada densa	Taxa de dropout	Taxa de aprendizagem
1	128	0.25	7e-05
2	96	0.25	8e-05
3	144	0.25	8e-05
4	112	0.30	8e-05

O mesmo treinamento de 50 épocas foi aplicado a cada uma das combinações de hiperparâmetros utilizando a estrutura #25 como modelo e a função *ModelCheckpoint*. Abaixo os resultados das métricas obtidas.

Métricas de Validação das Melhores Combinações Usando a Estrutura #25

Combinação	Acurácia	Precisão	Recall	AUC
1	0.8666	0.8774	0.8787	0.9465
2	0.8645	0.8735	0.8665	0.9425
3	0.8640	0.8640	0.8742	0.9343
4	0.8544	0.8584	0.8558	0.9566

E para a combinação com melhores resultados, foi plotada a matriz de confusão e a curva ROC.



## 5. Comparando Resultados: Otimização Bayesiana x Grid Search

A seguir são apresentadas as informações referentes aos conjuntos de validação dos modelos de estrutura #7 e #25 em ambas situações propostas, para a análise dos valores de cada hiperparâmetro.

Estrutura	#7 [Otimização Bayesiana]	#25 [Otimização Bayesiana]	#7 [Grid Search]	#25 [Grid Search]
<b>Acurácia</b>	0,8449	0.8718	0.8703	0.8544
<b>Precisão</b>	0,8522	0.8814	0.8766	0.8584
<b>Recall</b>	0,8468	0.8739	0.8719	0.8558
<b>AUC</b>	0,9367	0.9511	0.9417	0.9566
<b>Taxa de aprendizagem</b>	3.79 e-05	8.02 e-05	7e-05	8e-05
<b>Número de neurônios</b>	128	112	112	112
<b>Taxa de dropout</b>	0.45	0.25	0.3	0.30

Analizando as informações é possível concluir que os valores apresentam um empate técnico entre os modelos, os hiperparâmetros encontrados possuem valores semelhantes e as métricas também, apesar dessa observação um dos valores possui métricas ligeiramente superiores, por esse motivo a estrutura #25 utilizando o número de filtros convolucionais e hiperparâmetros encontrados através da técnica de

otimização bayesiana foi a selecionada como modelo base para prosseguir com as análises.

A partir dessa escolha foi dado prosseguimento na análise da influência do *batch size*, para essa análise foram escolhidos valores arbitrários de 16, 32, 64 e 128 imagens por lote. Os resultados de cada avaliação são apresentados a seguir.

## 6. Analisando o Último Hiperparâmetro: Batch Size

Esse hiperparâmetro foi analisado separadamente devido a sua importância durante a separação dos dados de teste e validação, o que acaba dificultando a alteração dele juntamente com os demais hiperparâmetros, por isso a abordagem foi avaliar os hiperparâmetros diretamente ligados ao modelo e deixar o *batch size* para uma análise posterior.

Abaixo são apresentados os resultados obtidos a partir do conjunto de validação para as avaliações referentes aos diferentes tamanhos do *batch size*.

Métricas	Batch Size = 16	Batch Size = 32	Batch Size = 64	Batch Size = 128	Batch Size = 256
<b>Acurácia</b>	0.8821	0.9018	0.8908	0.9001	0.8429
<b>Precisão</b>	0.8906	0.8976	0.8732	0.9007	0.8463
<b>Recall</b>	0.9079	0.8987	0.8860	0.8911	0.8431
<b>AUC</b>	0.9543	0.9551	0.9417	0.9496	0.9288

É notável que *batch size* entre 16 e 128, não alteram significativamente os valores das métricas, com a exceção do *batch size* igual a 256 que apresenta um valor das métricas claramente inferior quando comparado com os demais. Justificando a escolha de manter o *batch size* igual a 32 imagens por lote.

## 7. Analisando o Modelo

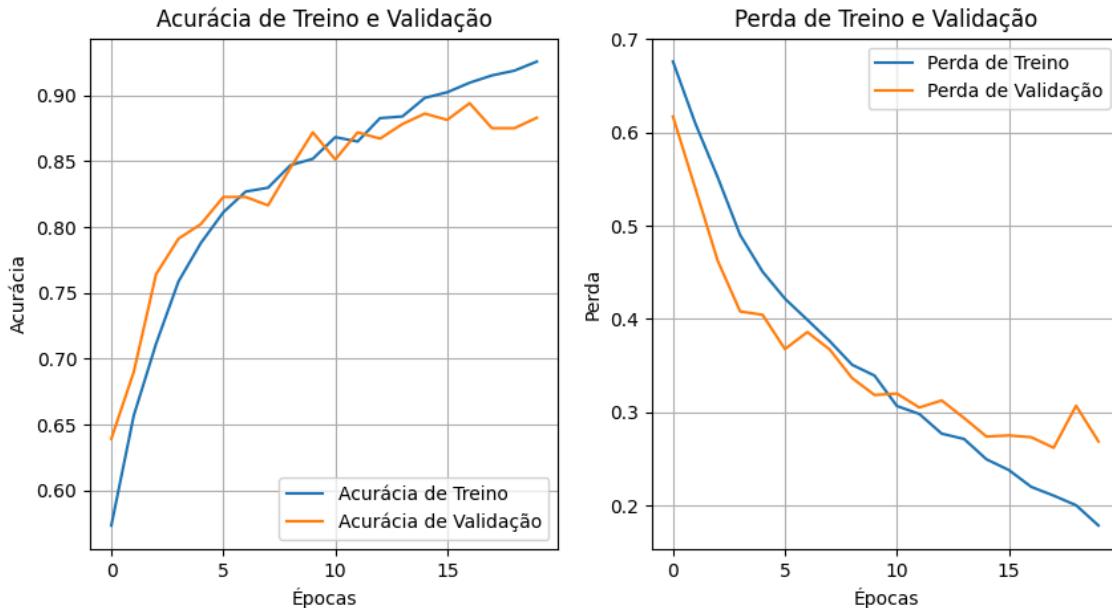
Alcançados os valores adequados de todos os hiperparâmetros, foi dado prosseguimento com o treinamento e a avaliação do modelo, bem como a preparação do ambiente para previsões com imagens de fora da base de dados.

O modelo final apresenta a seguinte estrutura:

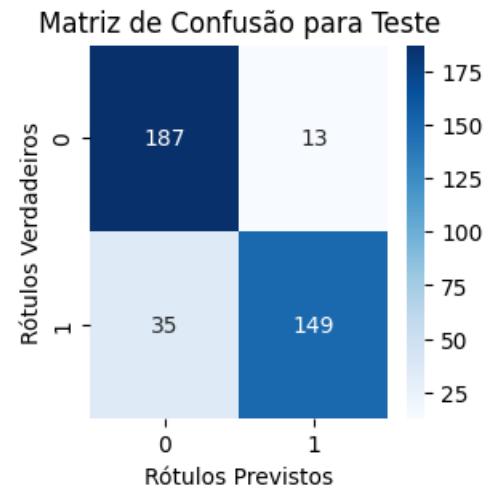
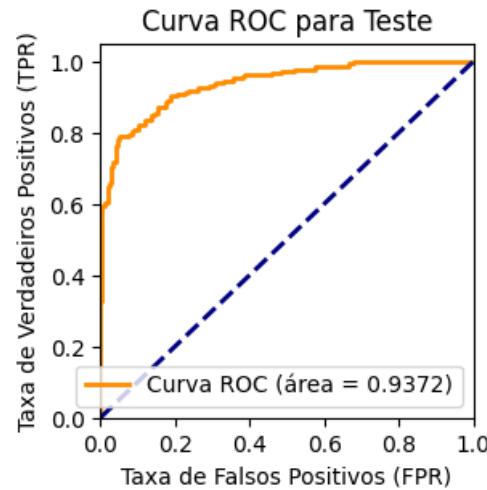
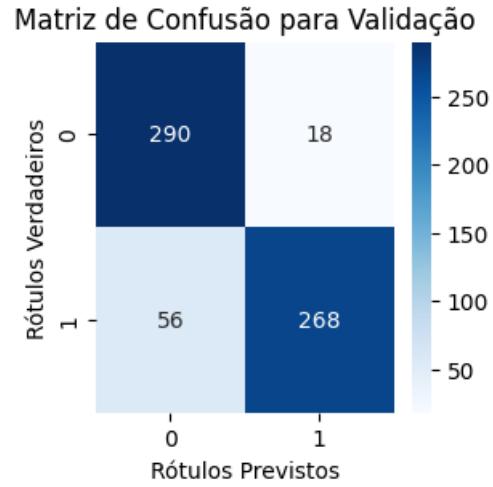
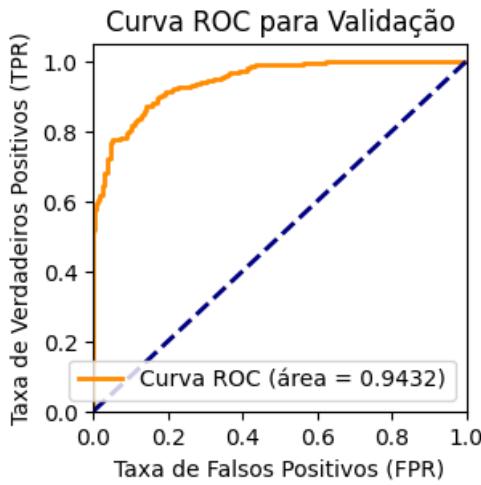
Model: "sequential_1"		
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	320
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_1 (Conv2D)	(None, 61, 61, 176)	50,864
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 176)	0
conv2d_2 (Conv2D)	(None, 28, 28, 160)	253,600
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 160)	0
conv2d_3 (Conv2D)	(None, 12, 12, 64)	92,224
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_4 (Conv2D)	(None, 4, 4, 128)	73,856
max_pooling2d_4 (MaxPooling2D)	(None, 2, 2, 128)	0
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 64)	32,832
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 2)	130

Total params: 503,826 (1.92 MB)  
 Trainable params: 503,826 (1.92 MB)  
 Non-trainable params: 0 (0.00 B)

Esse modelo apresenta os seguintes dados de treino e validação.

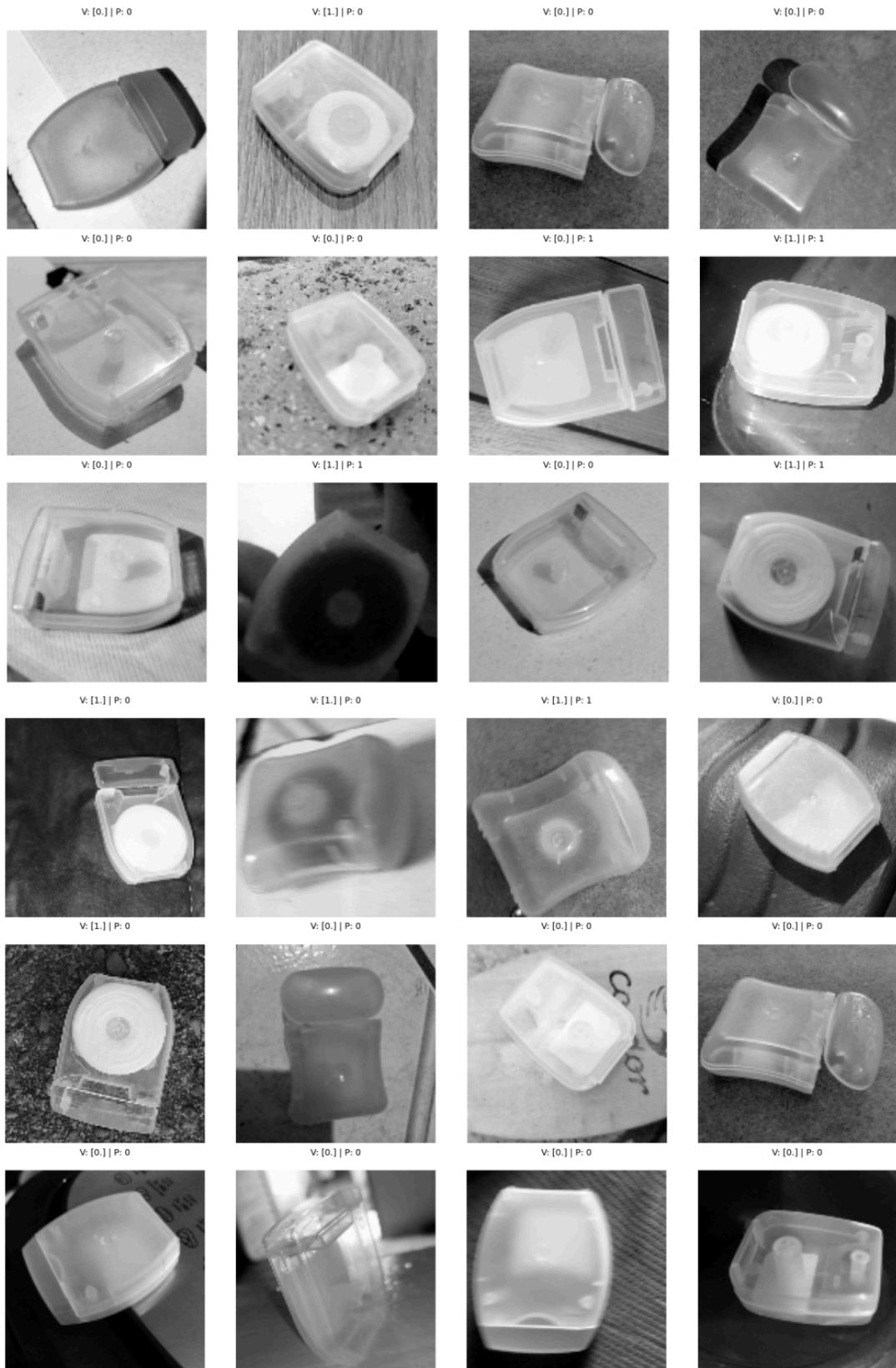


A matriz de confusão do modelo final e a curva ROC são apresentadas abaixo.



## 7.1. Buscando Falhas

Para a análise de falhas do modelo, foi usado o conjunto de testes, obtendo o seguinte resultado.



Dentre as 24 previsões realizadas utilizando o conjunto de testes, 5 imagens foram classificadas de forma errada, elas são apresentadas abaixo.

V: [1.] | P: 0



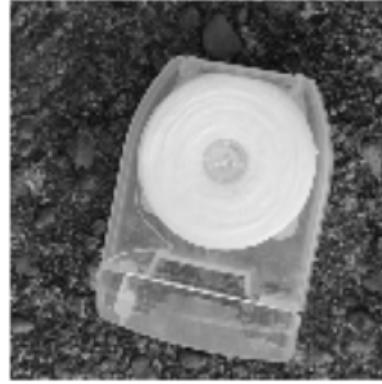
V: [1.] | P: 0



V: [0.] | P: 1



V: [1.] | P: 0



V: [1.] | P: 0



Apesar das imagens nítidas, o modelo não conseguiu discernir corretamente, este erro pode ser causado por uma ou mais características, desde o conjunto de sombras e reflexos, o formato das embalagens, a presença de rótulos/manchas ou ainda pela baixa nitidez da imagem.

Uma das opções possíveis para contornar problemas de classificação é o aumento do conjunto de dados, de forma natural ou artificial. Outra abordagem válida

é modificar as entradas do modelo, ou seja, usar outras combinações de processamento dos dados brutos, podendo ainda mesclar com configurações diferentes na técnica de *data augmentation*.

Os hiperparâmetros com melhores métricas foram encontrados com a aplicação da otimização bayesiana - que se mostrou bastante eficiente - e validados com a técnica de Grid Search em um primeiro momento. Devido ao custo computacional elevado proveniente da aplicação combinada das duas técnicas simultaneamente, para os testes seguintes, a otimização bayesiana foi o método utilizado para obtenção dos hiperparâmetros. Foram utilizadas as mesmas configurações apresentadas anteriormente.

Foram utilizadas diversas combinações de configuração do *data augmentation*, em conjunto com alterações no processamento dos dados brutos. A tabela com as combinações testadas é apresentada no anexo I.

Para prosseguir com os novos testes, o *dataset* foi processado novamente de forma a manter as novas imagens com a escala de cores original (RGB) e em tamanho 512x512. O novo *dataset* precisou ser hospedado no Google Drive devido ao seu maior tamanho, não aceito pelo Github.

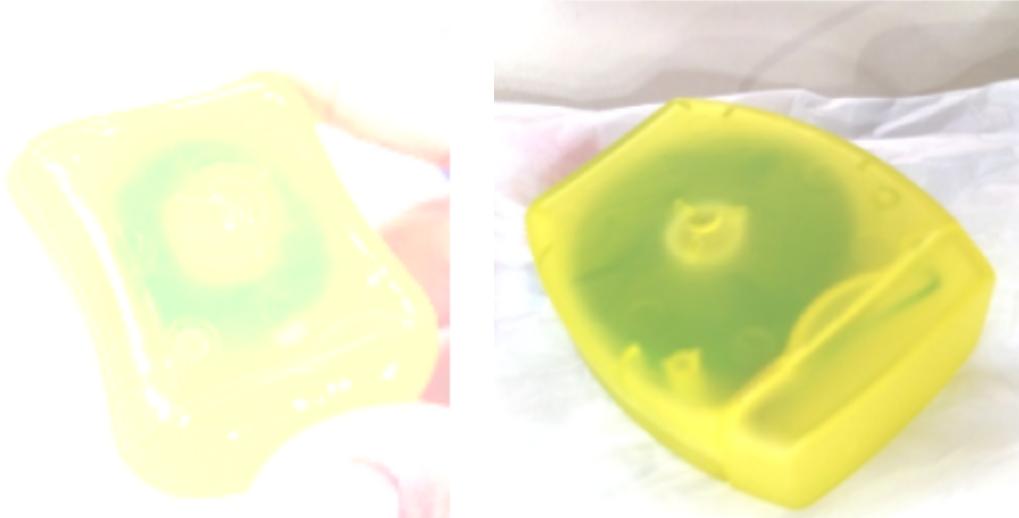
## 7.2 Fazendo Testes

Inicialmente foi feito um treino do modelo com as imagens no novo formato do *dataset*, 512x512 pixels, na escala de cores RGB, porém mesmo utilizando a GPU A100 disponibilizada no ambiente do Google Colab, uma única aplicação da técnica de otimização bayesiana se mostrou inviavelmente demorada, com duração beirando duas horas e trinta minutos, por esse motivo os teste seguintes usaram imagens nos tamanhos 64x64, 128x128 e 256x256.

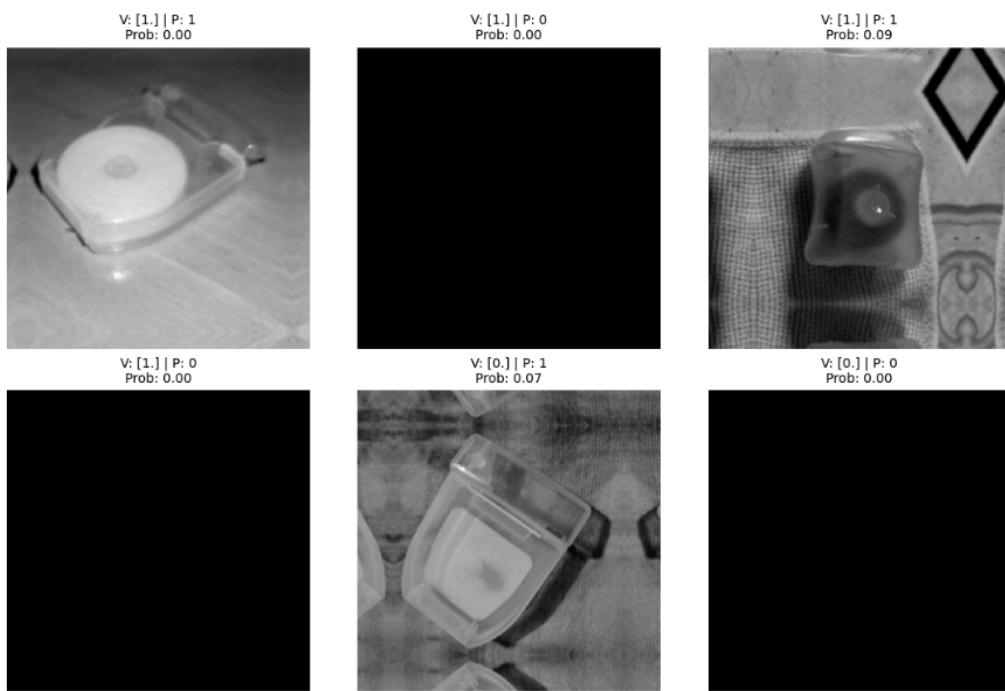
Devido ao aumento das imagens, para a aplicação da técnica de otimização bayesiana, o número de camadas foi ampliado, sendo possível para esses testes, redes com combinações de até 8 camadas convolucionais. As combinações foram feitas com base nos parâmetros a seguir.

Tamanho (px)	Data Augmentation	Escala de Cor
256 / 128 / 64	Flip Horiz. + Contraste 20%	Cinza
		RGB
	Flip Horiz. + Contraste 50% + Brilho 50% + Zoom 50%	Cinza
		RGB
	Flip Horiz. + Contraste 30% + Brilho 30%	Cinza
		RGB
	Flip Horiz.	Cinza
		RGB
	Flip Horiz. + Rotação 20% + Zoom 20%	Cinza
		RGB
	Flip Horiz. + Contraste 20% + Brilho 20%	Cinza
		RGB

As combinações com maiores quantidades de parâmetros apresentaram resultados insatisfatórios nos valores das métricas - Anexo I. Em muitos casos, mesmo com a aplicação de pequenas taxas de zoom e brilho as imagens tornavam-se praticamente ilegíveis ao modelo. Alguns exemplos assim são apresentados abaixo.



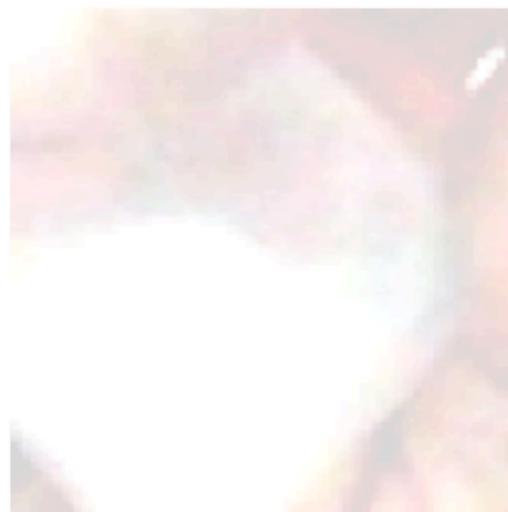
Com as imagens em escala de cinza, essas pequenas variações (10%-30%) causavam grande impacto em algumas imagens, tornando-as totalmente escuras.



Esse comportamento foi observado de forma oposta nas imagens RGB, tornando-as totalmente ou parcialmente esbranquiçadas.

V: [1.] | P: 0

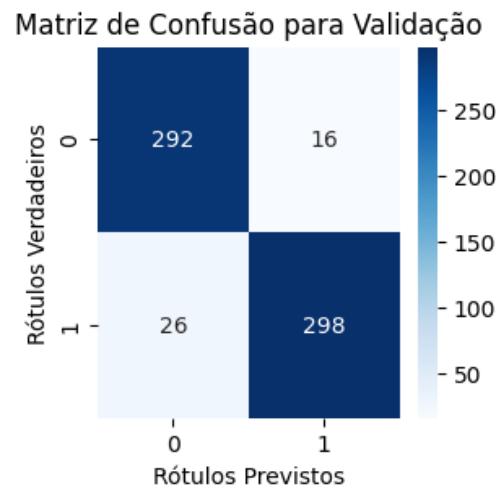
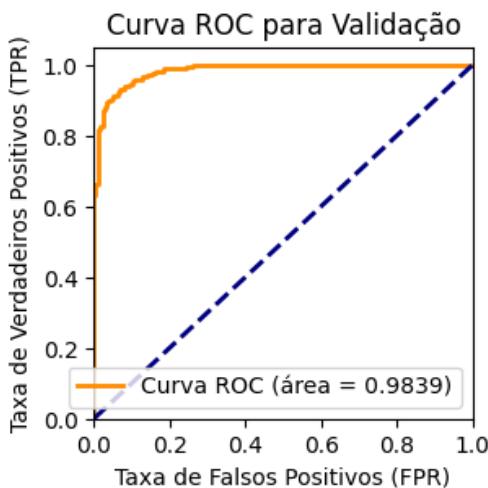
V: [1.] | P: 0

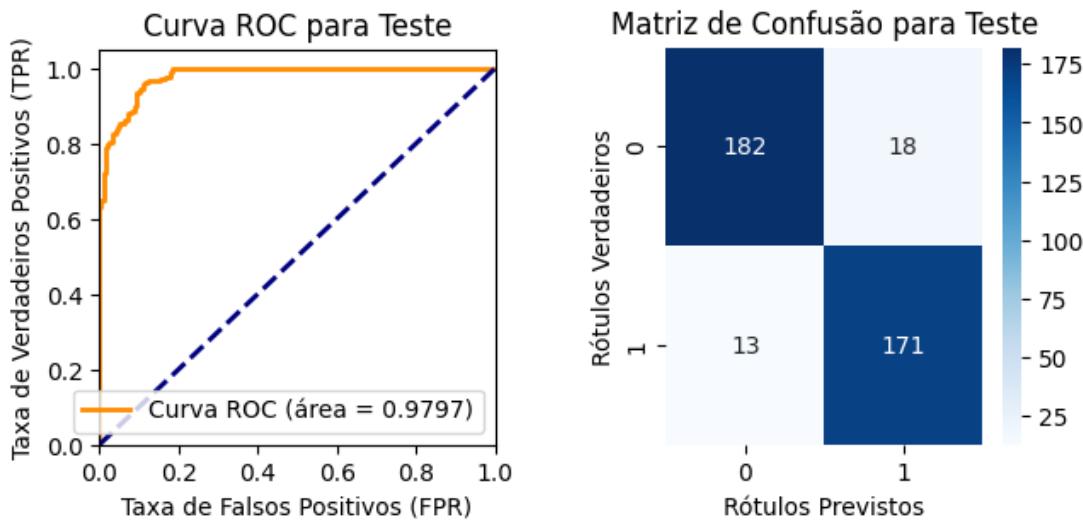


Os testes foram realizados na ordem presente no Anexo I, sendo numerados de 1 a 28, por isso, assim que eram identificados parâmetros insatisfatórios nos testes com as imagens maiores, apenas os melhores parâmetros eram selecionados para prosseguir com os testes em imagens menores, foi observado também que a não aplicação da técnica de *data augmentation* manteve as métricas similares, comprovando que um excesso de parâmetros poderia ser prejudicial ao modelo.

Baseado nas métricas do conjunto de teste, o modelo que obteve melhor resultado foi o de ID 14, utilizando imagens 128x128, com escala de cores RGB e possuindo apenas 4 camadas convolucionais. As demais informações sobre o modelo, incluindo seus hiperparâmetros (obtidos via otimização bayesiana) e configurações de *data augmentation* estão no Anexo II.

A seguir são apresentadas algumas métricas do modelo 14.





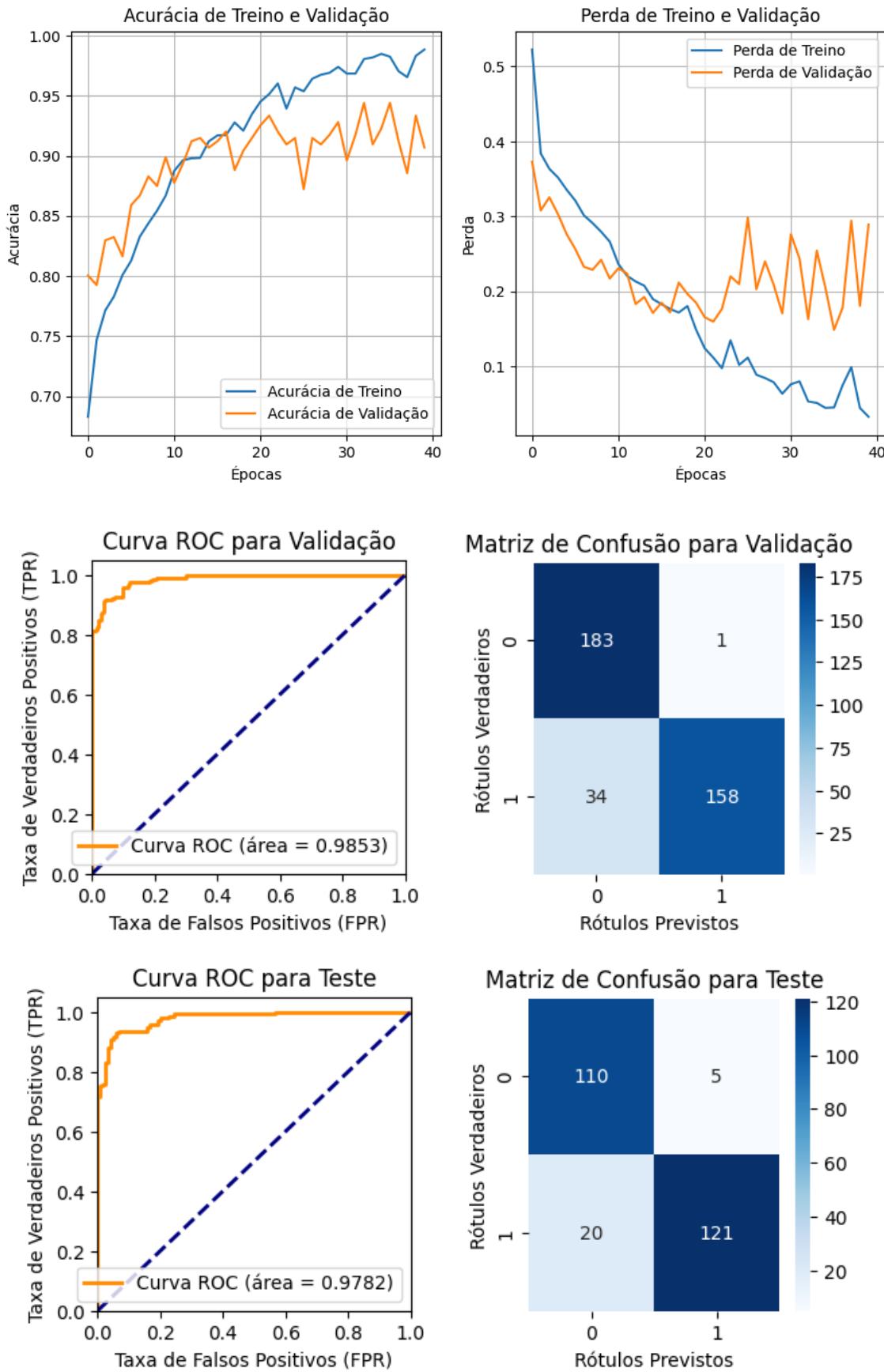
## 8. Analisando os Resultados do Modelo Final

Por fim, o modelo 14 foi treinado em 40 épocas, utilizando a função de *callback ModelCheckpoint* para armazenar os melhores pesos observados durante o treinamento, a estrutura do respectivo modelo é apresentada a seguir.

Model: "sequential_1"		
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 128, 128, 224)	6,272
max_pooling2d (MaxPooling2D)	(None, 63, 63, 224)	0
conv2d_1 (Conv2D)	(None, 61, 61, 16)	32,272
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 16)	0
conv2d_2 (Conv2D)	(None, 28, 28, 64)	9,280
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 64)	0
conv2d_3 (Conv2D)	(None, 12, 12, 112)	64,624
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 112)	0
flatten (Flatten)	(None, 4032)	0
dense (Dense)	(None, 192)	774,336
dropout (Dropout)	(None, 192)	0
dense_1 (Dense)	(None, 2)	386

Total params: 887,170 (3.38 MB)  
Trainable params: 887,170 (3.38 MB)  
Non-trainable params: 0 (0.00 B)

### Gráficos das Métricas do Modelo 14



Métricas do Modelo 14

Métricas	Validação	Teste
Acurácia	0.9069	0.9023
Precisão	0.9185	0.9032
Recall	0.9087	0.9073
AUC	0.9853	0.9782

### Imagens Classificadas pelo Modelo 14



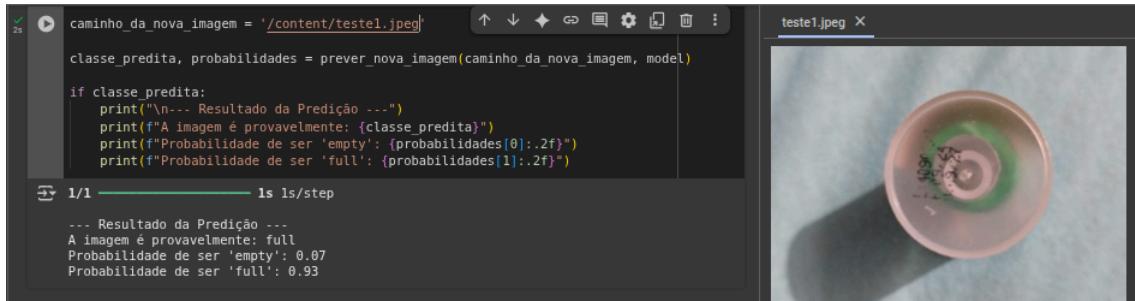
Das 256 imagens do conjunto de teste, o modelo classificou apenas 25 incorretamente, o que representa uma taxa de acerto ligeiramente acima dos 90%.

### 9. Fazendo Previsões

Foram implementadas as metodologias de previsão, duas no próprio *notebook* utilizando o Google Colab, permitindo tanto a predição direta pelo código como através da biblioteca Gradio, e mais uma que recebe o modelo salvo e faz inferências offline.

Para realizar a previsão diretamente via código é necessário fazer o *upload* das imagens à serem avaliadas para o ambiente do Google Colab e então o valor da variável *caminho\_da\_nova\_imagem* deve ser alterado diretamente no código.

As previsões a seguir foram realizadas com imagens inéditas para o modelo, sendo essas fotos tiradas por uma pessoa de fora do projeto, com outro *smartphone* e usando um modelo de embalagem que o modelo nunca viu durante o treinamento.



Code execution output:

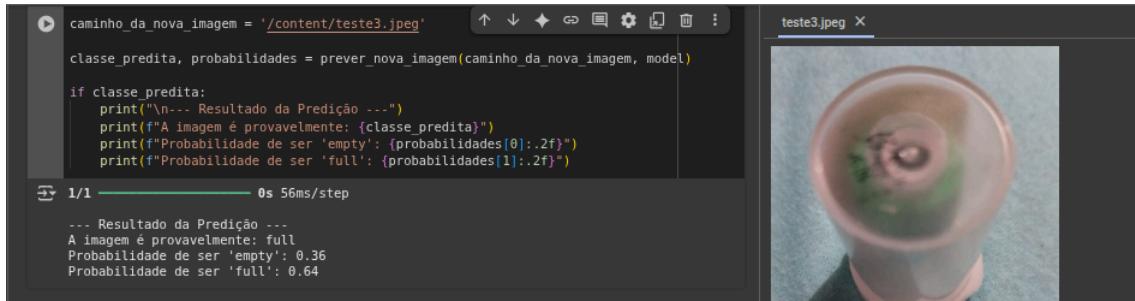
```

2s
caminho_da_nova_imagem = '/content/teste1.jpeg' ↑ ↓ ⏪ ⏫ ⏴ ⏵ ⏷ ⏸ ⏹ ⏺ : 
classe_predita, probabilidades = prever_nova_imagem(caminho_da_nova_imagem, model)

if classe_predita:
    print("\n--- Resultado da Predição ---")
    print(f"A imagem é provavelmente: {classe_predita}")
    print(f"Probabilidade de ser 'empty': {probabilidades[0]:.2f}")
    print(f"Probabilidade de ser 'full': {probabilidades[1]:.2f}")

1/1  ls 1s/step
--- Resultado da Predição ---
A imagem é provavelmente: full
Probabilidade de ser 'empty': 0.07
Probabilidade de ser 'full': 0.93

```



Code execution output:

```

caminho_da_nova_imagem = '/content/teste3.jpeg' ↑ ↓ ⏪ ⏫ ⏴ ⏵ ⏷ ⏸ ⏹ ⏺ : 
classe_predita, probabilidades = prever_nova_imagem(caminho_da_nova_imagem, model)

if classe_predita:
    print("\n--- Resultado da Predição ---")
    print(f"A imagem é provavelmente: {classe_predita}")
    print(f"Probabilidade de ser 'empty': {probabilidades[0]:.2f}")
    print(f"Probabilidade de ser 'full': {probabilidades[1]:.2f}")

1/1  os 56ms/step
--- Resultado da Predição ---
A imagem é provavelmente: full
Probabilidade de ser 'empty': 0.36
Probabilidade de ser 'full': 0.64

```

O outro método, mais elegante, é utilizando a biblioteca Gradio do Python que permite a criação simplificada de interfaces web interativas para modelos de *machine learning*. Sendo possível desta forma, inferir resultados para novas imagens sem a necessidade de alterar quaisquer valores diretamente no código. Dois exemplos com imagens da *internet* são apresentados abaixo.



Uma das maiores vantagens em utilizar a biblioteca Gradio é a maneira simples e rápida com que ela permite demonstrar o uso do modelo de aprendizado de máquina utilizando uma interface amigável, tornando mais simples e agradável a apresentação dos resultados.

Além das formas online utilizadas Google Colab também foi disponibilizado via repositório Github o código *predict\_offline.py* que recebe o modelo salvo em formato .h5 e o utiliza para fazer previsões de forma local e offline, sem a necessidade de fazer *uploads* de imagens ou alterar o código fonte.

```

import tensorflow as tf
import numpy as np
import os
import cv2
import argparse

parser = argparse.ArgumentParser()

parser.add_argument(
    '--pasta_modelo',
    type=str,
    required=True,
    help='Caminho para o arquivo do modelo (.h5).'
)

parser.add_argument(
    '--pasta_imagens',
    type=str,
    required=True,
    help='Caminho para a pasta contendo as imagens para inferência.'
)

args = parser.parse_args()

CLASS_NAMES = ['empty', 'full']

try:
    model = tf.keras.models.load_model(args.pasta_modelo)
    print("Modelo carregado com sucesso!")
except Exception as e:
    print(f"Erro ao carregar o modelo: {e}")
    exit()

```

Trecho do código *predict\_offline.py*

Com esse código salvo na máquina local juntamente com o arquivo do modelo, é necessário apenas fazer a passagem por parâmetros dos seus respectivos diretórios para que o modelo faça a inferência de todas as imagens contidas no caminho informado pelo parâmetro “*--pasta\_imagens*” . Por fim, o código gera uma saída como a mostrada a seguir.

```

Imagen: teste1.jpeg -> Classe Predita: full
Probabilidades: empty: 0.01, full: 0.99

-----
Imagen: dental2.png -> Classe Predita: full
Probabilidades: empty: 0.49, full: 0.51

-----
Imagen: teste2.jpeg -> Classe Predita: empty
Probabilidades: empty: 0.77, full: 0.23

-----
Imagen: dental11.png -> Classe Predita: empty
Probabilidades: empty: 0.98, full: 0.02

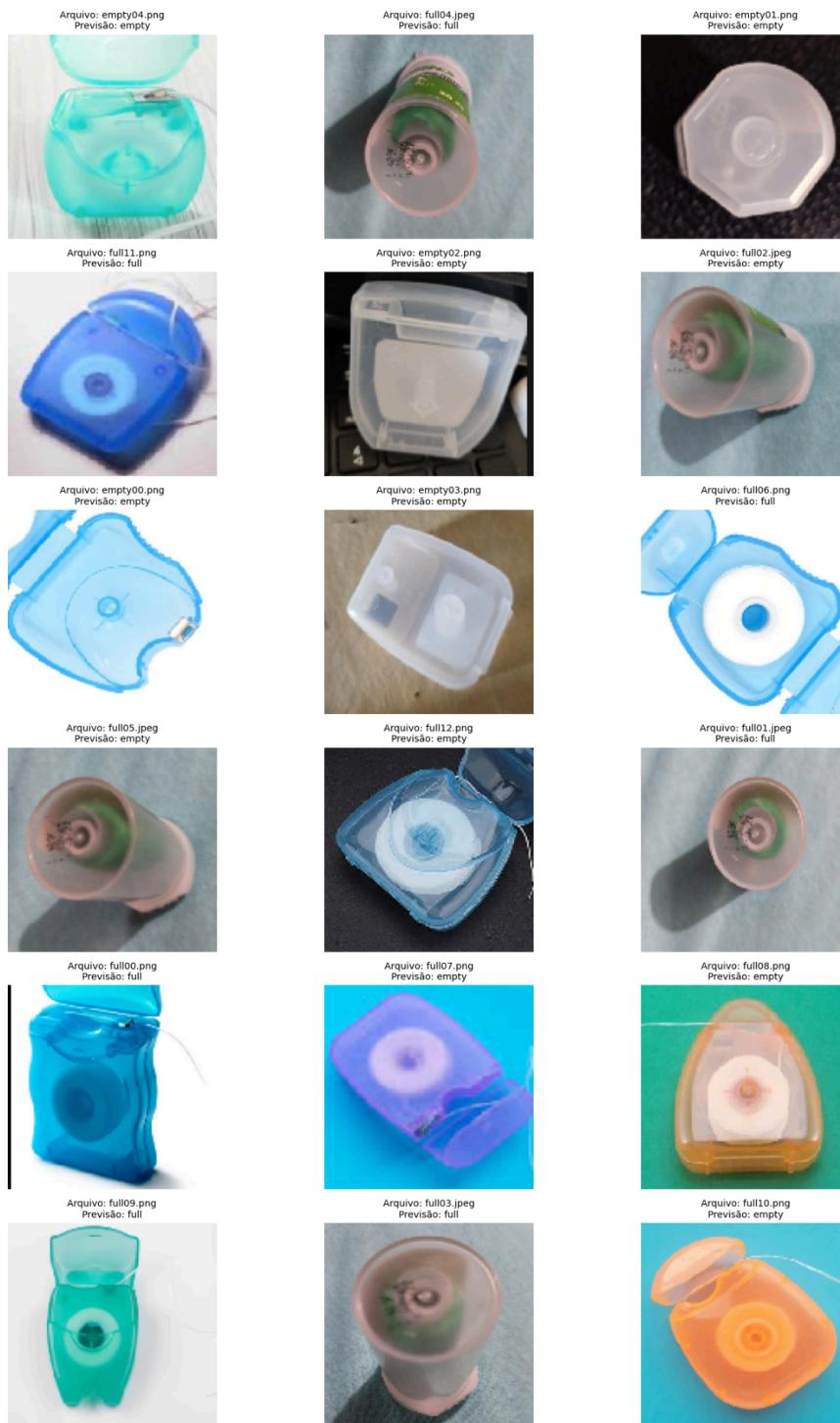
-----
Imagen: dental5.png -> Classe Predita: empty
Probabilidades: empty: 0.99, full: 0.01

-----
Imagen: dental4.png -> Classe Predita: full
Probabilidades: empty: 0.17, full: 0.83

```

Além das previsões utilizando fotos individuais, foi aplicado um último teste com o modelo 14 onde foram selecionadas 18 imagens inéditas, sendo a maioria da internet.

Previsões do Modelo para Novas Imagens



Com as imagens da internet a taxa de acerto do modelo ficou em cerca de 60%, classificando corretamente 12 das 18 fotos.

## Conclusão

Ao final, o projeto de classificação de embalagens de fio dental alcançou bons resultados. O modelo final foi capaz de alcançar um desempenho significativo, com métricas que demonstram alta capacidade de discernir entre as classes. As principais dificuldades, como a escolha de hiperparâmetros e a montagem de um dataset consistente, foram superadas com a implementação de técnicas sistemáticas como a otimização bayesiana e *Grid Search*, além do desenvolvimento de um *pipeline* de dados, que fez do pré-processamento um evento menos traumático. Apesar de todas as dificuldades técnicas, de tempo e de disponibilidade, o projeto foi finalizado com êxito.

## Referências

- [1] TENSORFLOW.ORG. Keras | TensorFlow Core  
<<https://www.tensorflow.org/guide/keras?hl=pt-br>>, Acesso em 28 de junho de 2025
- [2] KAGGLE.COM. Find Open Datasets and Machine Learning Projects.  
Disponível em: <<https://www.kaggle.com/datasets>> , Acesso em 24 de junho de 2025
- [3] VERSION1.COM. AI Performance Metrics: The Science & Art of Measuring AI - Version 1  
Disponível em:  
<<https://www.version1.com/blog/ai-performance-metrics-the-science-and-art-of-measuring-ai>> , Acesso em 28 de junho de 2025
- [4] GRADIO.APP. Gradio Documentation. Disponível em: <<https://www.gradio.app/docs>> ,  
Acesso em 6 de agosto de 2025
- [5] MEDIUM.COM Como Tunar Hiperparâmetros com Otimização Bayesiana.  
Disponível em:  
<<https://medium.com/@beniciowg/como-tunar-hiperpar%C3%A2metros-com-otimiza%C3%A7%C3%A3o-bayesiana-5687fd51370f>>, Acesso em 13 de agosto de 2025
- [6] KERAS.IO. ModelCheckpoint  
Disponível em: <[https://keras.io/api/callbacks/model\\_checkpoint/](https://keras.io/api/callbacks/model_checkpoint/)>, Acesso em 13 de agosto de 2025

Anexo I

Métricas de Validação

Tamanho (px)	Data Aug.	Escala de Cor	ID do Modelo	Acurácia [Val]	Precisão [Val]	Recall [Val]	AUC [Val]
256	H + Contraste 20%	Cinza	1	0,8766	0,8830	0,8783	0,9494
		RGB	2	0,8956	0,8955	0,8957	0,9511
	H + Contraste 50% + Brilho 50% + Zoom 50%	Cinza	3	0,5158	0,5130	0,5114	0,4764
		RGB	4	0,5553	0,5437	0,5029	0,5188
	H + Contraste 30% + Brilho 30%	Cinza	5	0,5127	0,2563	0,5000	0,5000
		RGB	6	0,8149	0,8149	0,8146	0,7646
	H	Cinza	7	0,7921	0,8015	0,8008	0,8007
		RGB	8	0,8544	0,8548	0,8549	0,9259
	H + Rotação 20% + Zoom 20%	Cinza	9	0,5127	0,2563	0,5000	0,5000
		RGB	10	0,5573	0,2632	0,5000	0,5000
	H + Contraste 20% + Brilho 20%	Cinza	11	0,5077	0,5564	0,5165	0,5142
		RGB	12	0,5158	0,5759	0,5261	0,5233
128	H + Contraste 20%	Cinza	13	0,8449	0,8522	0,8468	0,9064
		RGB	14	0,9335	0,9336	0,9339	0,9839
	H	Cinza	15	0,8703	0,8702	0,8704	0,9269
		RGB	16	0,9035	0,9036	0,9032	0,9688
	--	Cinza	17	0,8972	0,8973	0,8975	0,9701
		RGB	18	0,9193	0,9193	0,9196	0,9782
	H + Contraste 20% + Zoom 20%	Cinza	19	0,8703	0,8760	0,8719	0,9472
		RGB	20	0,9130	0,9129	0,9130	0,9707
64	H + Contraste 20%	Cinza	21	0,7769	0,7886	0,7794	0,828
		RGB	22	0,8797	0,8796	0,8798	0,9544
	H	Cinza	23	0,8497	0,8503	0,8503	0,9218
		RGB	24	0,9193	0,9203	0,9200	0,9787
	--	Cinza	25	0,8291	0,8295	0,8296	0,9093

		RGB	26	0,8703	0,8786	0,8722	0,9562
	H + Contraste 20% + Zoom 20%	Cinza	27	0,8291	0,8292	0,8294	0,9025
		RGB	28	0,9141	0,9254	0,9249	0,9769

### Métricas de Teste

Tamanho (px)	Data Aug.	Escala de Cor	ID do Modelo	Acurácia [Test]	Precisão [Test]	Recall [Test]	AUC [Test]
256	H + Contraste 20%	Cinza	1	0,8646	0,8704	0,8620	0,9267
		RGB	2	0,8802	0,8807	0,8793	0,9408
	H + Contraste 50% + Brilho 50% + Zoom 50%	Cinza	3	0,5104	0,5278	0,5209	0,5028
		RGB	4	0,5578	0,5443	0,5016	0,5361
	H + Contraste 30% + Brilho 30%	Cinza	5	0,4792	0,2396	0,5000	0,5000
		RGB	6	0,7839	0,7842	0,7847	0,7512
	H	Cinza	7	0,8304	0,7953	0,7972	0,8197
		RGB	8	0,8359	0,8357	0,8355	0,9159
	H + Rotação 20% + Zoom 20%	Cinza	9	0,4792	0,2396	0,5000	0,5000
		RGB	10	0,5039	0,2680	0,5000	0,5000
	H + Contraste 20% + Brilho 20%	Cinza	11	0,5297	0,6122	0,5240	0,5109
		RGB	12	0,5729	0,6401	0,5570	0,5501
128	H + Contraste 20%	Cinza	13	0,7995	0,8080	0,7958	0,8771
		RGB	14	0,9193	0,9190	0,9197	0,9797
	H	Cinza	15	0,8385	0,8388	0,8376	0,9040
		RGB	16	0,8984	0,8982	0,8984	0,9640
	--	Cinza	17	0,8906	0,8906	0,8902	0,9640
		RGB	18	0,8828	0,8825	0,8829	0,9658
	H + Contraste 20% + Zoom 20%	Cinza	19	0,8594	0,8632	0,8572	0,9289
		RGB	20	0,8646	0,8643	0,8643	0,9450

64	H + Contraste 20%	Cinza	21	0,7526	0,7553	0,7499	0,8018
		RGB	22	0,8542	0,8540	0,8546	0,9430
	H	Cinza	23	0,8229	0,8226	0,8228	0,9005
		RGB	24	0,8984	0,9017	0,8966	0,9661
	--	Cinza	25	0,8073	0,8077	0,8061	0,8802
		RGB	26	0,8385	0,8468	0,8352	0,9387
	H + Contraste 20% + Zoom 20%	Cinza	27	0,8099	0,8097	0,8092	0,8910
		RGB	28	0,8958	0,8987	0,8941	0,9621

Anexo II

Hiperparâmetros de cada modelo testado

ID do Modelo	Num. Camadas	Tx. Aprendizado	Tx. Dropout	Num. Neurônio	Camada 1	Camada 2	Camada 3	Camada 4	Camada 5	Camada 6	Camada 7	Camada 8
1	8	0.000035337	0.25	176	224	208	256	160	80	256	160	176
2	8	0.000212960	0.15	160	192	112	16	112	128	176	16	256
3	8	0.000032735	0.35	240	16	176	48	48	80	96	48	112
4	6	0.000876164	0.15	160	96	32	32	64	208	192	--	--
5	6	0.000696123	0.25	192	64	32	112	208	96	96	--	--
6	7	0.000543640	0	128	96	48	240	208	80	144	192	--
7	6	0.000018530	0.2	16	16	16	64	64	16	16	--	--
8	8	0.000615789	0.45	128	208	240	176	192	240	176	224	224
9	5	0.000154905	0.3	16	80	224	96	208	144	--	--	--
10	6	0.000741202	0.25	112	64	208	80	64	64	144	--	--
11	6	0.000136799	0.25	160	240	192	112	240	144	32	--	--
12	5	0.000209820	0.45	208	112	224	112	208	144	--	--	--
13	5	0.000340824	0.1	240	192	192	80	32	16	--	--	--
14	4	0.000466428	0.25	192	224	16	64	112	--	--	--	--
15	5	0.000456606	0.2	64	32	176	160	64	128	--	--	--
16	5	0.000377846	0.35	160	80	176	192	224	64	--	--	--
17	6	0.000214458	0.35	208	64	48	192	256	208	192	--	--
18	4	0.000227422	0.3	208	192	64	224	16	--	--	--	--
19	6	0.000075760	0.05	192	208	48	192	224	192	144	--	--
20	5	0.000475320	0.15	208	208	112	48	80	--	--	--	--
21	6	0.000311350	0.4	224	160	32	48	192	144	176	--	--
22	6	0.000871628	0.0	160	96	128	240	48	208	176	--	--
23	3	0.000429313	0.3	176	192	16	--	--	--	--	--	--
24	4	0.000104474	0.2	176	48	224	208	96	--	--	--	--
25	6	0.000391348	0.05	128	144	32	32	96	64	192	--	--
26	5	0.000034899	0.25	112	208	224	112	160	96	--	--	--
27	5	0.000966279	0.1	32	48	192	16	224	256	--	--	--
28	6	0.000634793	0.3	208	80	144	128	176	96	176	--	--