INSTITUTO ATLÂNTICO

ATLÂNTICO ACADEMY FUTURE – COMPUTAÇÃO COGNITIVA

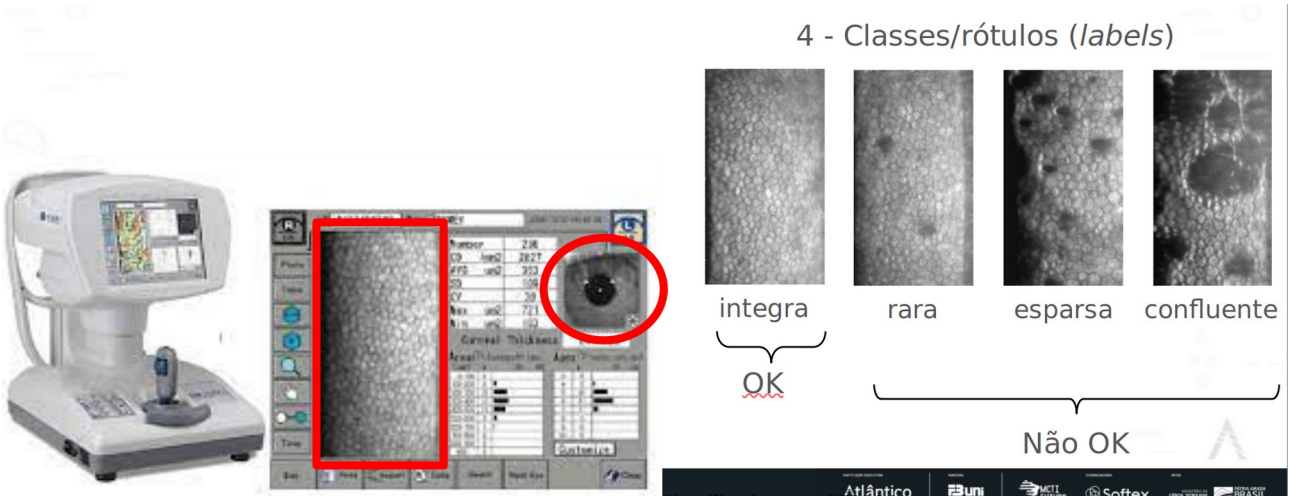
PROJETO VISÃO COMPUTACIONAL

2021

PROF.: ALYSON BEZERRA NOGUEIRA

FRANCISCO MARCELINO ALMEIDA DE ARAÚJO
RAYLAN CORDEIRO DE SOUZA
VAGNER SANCHES VASCONCELOS
VICTOR DE SOUSA ROCHA

O projeto em questão tem como objetivo desenvolver um sistema que permita o auxílio ao diagnóstico de patologias da corneá por meio de imagens de microscopia especular. Para isto, foi utilizado o conjunto de dados (*data set*): data_especular_crop.



Em todos os casos, o treinamento ocorreu utilizando o Google Colab com GPUs, as métricas de avaliação foram *precision (*precisão*)* e *recall (*sensibilidade*) e as imagens foram n*ormalização em 256x256.

Foram implementadas 4 estratégias, sendo elas:

1. Sem nenhum pré-processamento (sem estratégia);
2. Os dados das classes confluente e esparsa foram unidos, gerando a classe confluente_esparsa (1ª estratégia);
3. Limitando os dados da classe integra em 574 imagens (2ª estratégia);
4. Limitando as imagens da classe integra, da mesma forma que na 2ª estratégia, e ainda aplicando filtro bilateral em todas as imagens (3ª estratégia) .

Segue abaixo um quadro resumo dos resultados das métricas calculadas com os dados de testes. Nele se observa que a 1ª estratégia para a rede VGG19 foi a de melhor resultado.

Quadro 1: Quadro resumo das métricas com os dados de teste

		CNN	VGG19
Sem Estratégia	Precision	0.7327	0.8864
	Recall	0.7227	0.8864
1ª Estratégia	Precision	0.8119	0.8904
	Recall	0.8082	0.8904
2ª Estratégia	Precision	0.7410	0.8429
	Recall	0.7357	0.8429
3ª Estratégia	Precision	0.7279	0.8143
	Recall	0.7071	0.8143

1) SEM ESTRATÉGIA

Neste primeiro modelo não utilizaremos nenhuma estratégia de pré-processamento.

1.1) ANÁLISE DOS DADOS:

(a) Quantidade total de Imagens: 2181
(b) Classes/rótulos: ['rara' 'confluente' 'integra' 'esparsa']
(c) Tamanho de cada classe: [574, 118, 1368, 121]
(d) Tamanho percentual de cada classe: [26.3, 5.4, 62.7, 5.5]
(e) Grande parte das imagens possui dimensões de 245x460.
(f) Total de imagens para treinamento: 1527
(g) Total para cada classe: [402, 82, 959, 84]
(h) Dados de treinamento: (1527, 256, 256, 3)
(i) Rótulos de treinamento: (1527, 4)
(j) Total de imagens para validação: 433
(k) Total para cada classe: [114, 24, 271, 24]
(l) Dados de validação: (433, 256, 256, 3)
(m) Rótulos de validação: (433, 4)
(n) Total de imagens para testes: 220
(o) Total para cada classe: [58, 12, 137, 13]
(p) Dados de testes: (220, 256, 256, 3)
(q) Rótulos de testes: (220, 4)
(r) Código: Colab.

1.2) PARÂMETROS DE TREINAMENTO

(a) batch_size = 16
(b) epochs = 32

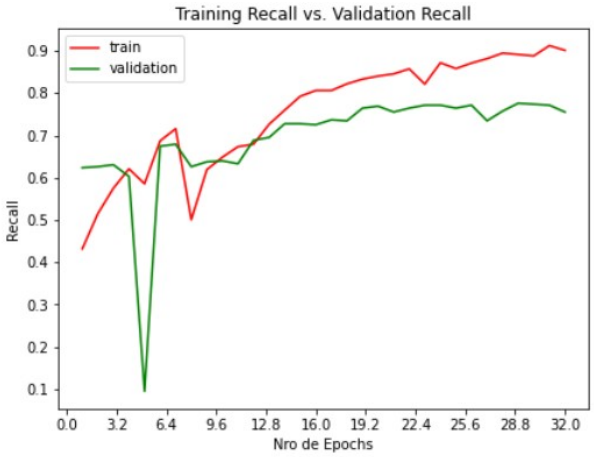
1.3) REDE: CNN

(a) Arquitetura, ver código.
(b) Duração 8.4 minutos para treinamento de 32 épocas
(c) Epoch 1/32: 28s 168ms/step – loss: 18.0346 – precision: 0.5921 – recall: 0.4316 – val_loss:

0.9517 – val_precision: 0.6353 – val_recall: 0.6236

(d) Epoch 32/32: 14s 148ms/step – loss: 0.2757 – precision: 0.9083 – recall: 0.9011 – val_loss: 1.0645 – val_precision: 0.7640 – val_recall: 0.7552

(e) Gráficos da precisão e sensibilidade (recall) durante o treinamento.



(f) Avaliação do modelo com o conjunto de testes.
Comando: model_cnn.evaluate(test_data, test_labels, verbose=2)
Saída: 7/7 – 1s – loss: 1.1713 – precision: 0.7327 – recall: 0.7227 – 571ms/epoch – 82ms/step

1.4) REDE: VGG19

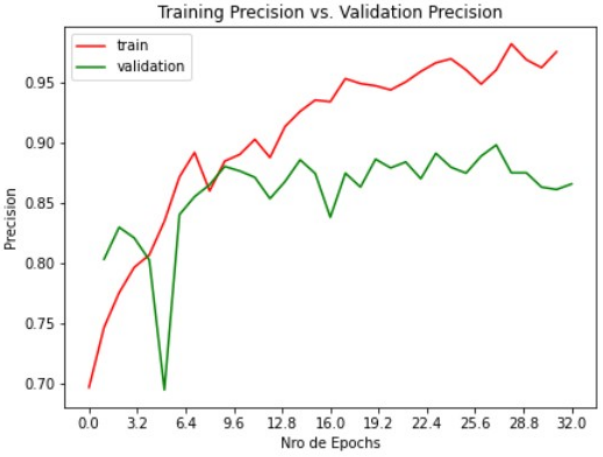
(a) Arquitetura: ver código.

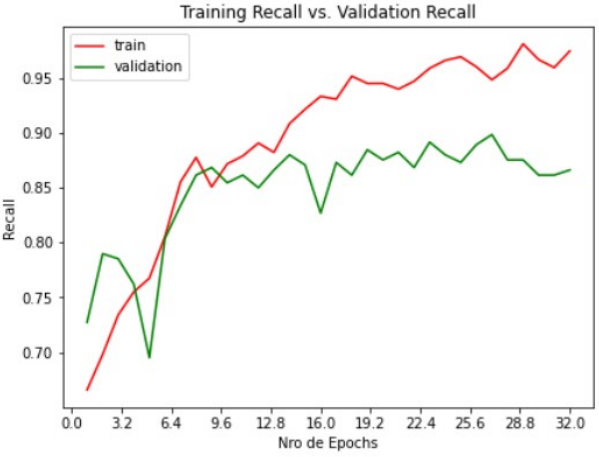
(b) Duração 12.9 minutos para treinamento de 32 épocas

(c) Epoch 1/32: 36s 297ms/step – loss: 2.0332 – precision: 0.6971 – recall: 0.6658 – val_loss: 0.5922 – val_precision: 0.8036 – val_recall: 0.7275

(d) Epoch 32/32: 24s 248ms/step – loss: 0.0648 – precision: 0.9757 – recall: 0.9745 – val_loss: 0.7241 – val_precision: 0.8661 – val_recall: 0.8661

(e) Gráficos da precisão e sensibilidade (*recall*) durante o treinamento.





(f) Avaliação do modelo com o conjunto de testes.
Comando: model_vgg19.evaluate(test_data, test_labels, verbose=2)
Saída: 7/7 – 2s – loss: 1.0322 – precision: 0.8864 – recall: 0.8864 – 2s/epoch – 346ms/step

2) PRIMEIRA ESTRATÉGIA.

Identificamos um grande desbalanceamento nas classes, e com o entendendo melhor o problema de negócio, tanto as imagens classificadas como confluente, bem como esparsa são problemas na córnea, desta forma decidimos agrupá-las em uma única classe batizada como confluente_esparsa, indo assim ao encontro de balancear o conjunto de dados.

Desta forma nosso classificador passou de 4 para 3 classes.

2.1) ANÁLISE DOS DADOS

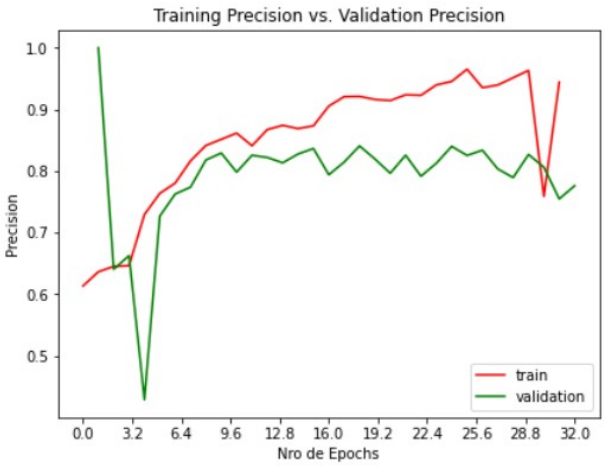
(a) Classes/rótulos: ['confluente_esparsa' 'rara' 'integra']
(b) Quantidade total de Imagens: 2181
(c) Contudo agora as classes ficaram assim:
(d) Tamanho de cada classe: [239, 574, 1368]
(e) Sendo os percentuais: Tamanho percentual de cada classe: [10.96, 26.32, 62.72]
(f) As imagens continuaram padronizadas em 255x255
(g) Total de imagens para treinamento: 1528
(h) Total para cada classe: [167, 402, 959]
(i) Dados de treinamento: (1528, 256, 256, 3)
(j) Rótulos de treinamento: (1528, 3)
(k) Total de imagens para validação: 433
(l) Dados de validação: (433, 256, 256, 3)
(m) Rótulos de validação: (433, 3)
(n) Total de imagens para testes: 219
(o) Total para cada classe: [48, 114, 271]
(p) Total para cada classe: [24, 58, 137]
(q) Dados de testes: (219, 256, 256, 3)
(r) Rótulos de testes: (219, 3)
(s) Código: Colab

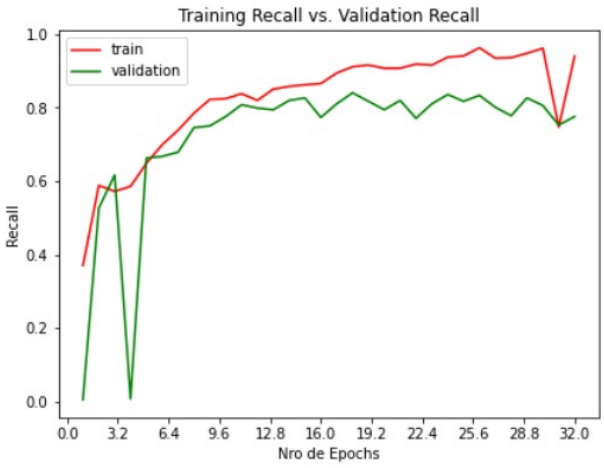
2.2) PARÂMETROS DE TREINAMENTO:

(a) batch_size = 16
(b) epochs = 32

2.3) REDE CNN

(a) Duração 8.4 minutos para treinamento de 32 épocas
(b) Epoch 1/32: 28s 173ms/step – loss: 14.1373 – precision: 0.6132 – recall: 0.3704 – val_loss: 0.9537 – val_precision: 1.0000 – val_recall: 0.0046
(c) Epoch 32/32: 14s 151ms/step – loss: 0.1821 – precision: 0.9442 – recall: 0.9404 – val_loss: 1.0404 – val_precision: 0.7760 – val_recall: 0.7760
(d) Gráficos da precisão e sensibilidade (recall) durante o treinamento.

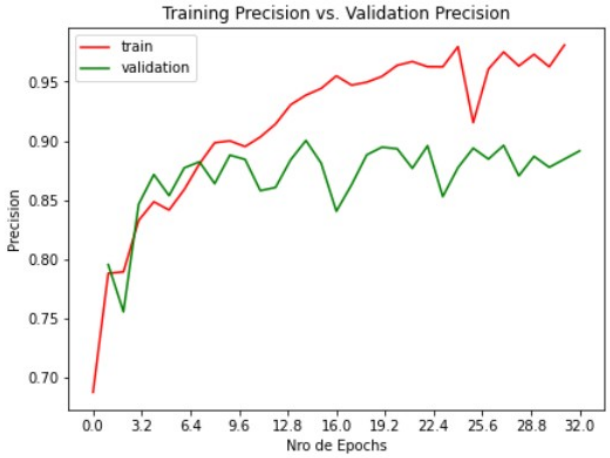


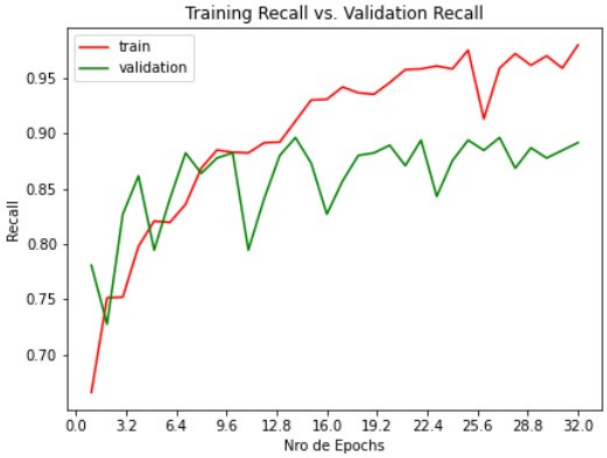


(e) Avaliação do modelo com o conjunto de testes.
Comando: model_cnn.evaluate(test_data, test_labels, verbose=2)
Saída: 7/7 – 1s – loss: 0.9151 – precision: 0.8119 – recall: 0.8082 – 603ms/epoch – 86ms/step

2.4) REDE: VGG19

(a) Duração 14.4 minutos para treinamento de 32 épocas
(b) Epoch 1/32: 40s 333ms/step – loss: 1.9935 – precision: 0.6874 – recall: 0.6660 – val_loss: 0.6832 – val_precision: 0.7953 – val_recall: 0.7806
(c) Epoch 32/32: 25s 265ms/step – loss: 0.0640 – precision: 0.9810 – recall: 0.9797 – val_loss: 0.7843 – val_precision: 0.8915 – val_recall: 0.8915
(d) Gráficos da precisão e sensibilidade (recall) durante o treinamento.





(e) Avaliação do modelo com o conjunto de testes.
Comando: model_vgg19.evaluate(test_data, test_labels, verbose=2)
Saída: 7/7 – 3s – loss: 0.6087 – precision: 0.8904 – recall: 0.8904 – 3s/epoch – 379ms/step

3) SEGUNDA ESTRATÉGIA

Buscando um melhor balançamento das classes, utilizamos somente 574 imagens (aleatórias) da classe integra.

3.1 – ANÁLISE DOS DADOS:

(a) Quantidade total de Imagens: 1386
(b) Classes/rótulos: ['integra' 'rara' 'confluente_esparsa']
(c) Tamanho de cada classe: [574, 573, 239]

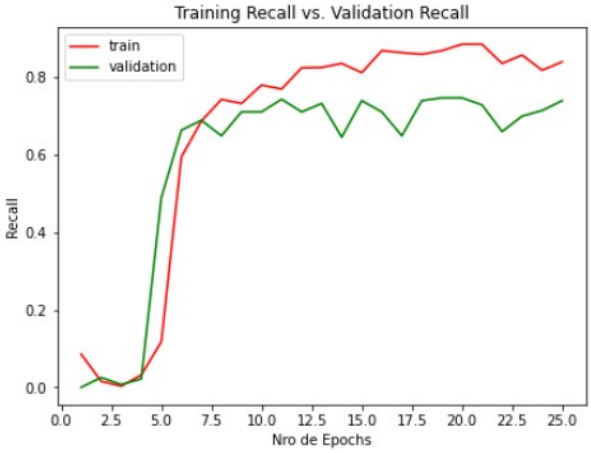
(d) Tamanho percentual de cada classe: [41.4, 41.3, 17.2]
(e) Total de imagens para treinamento: 970
(f) Total para cada classe: [402, 401, 167]
(g) Dados de treinamento: (970, 256, 256, 3)
(h) Rótulos de treinamento: (970, 3)
(i) Total de imagens para validação: 276
(j) Total para cada classe: [114. 114, 48]
(k) Dados de validação: (276, 256, 256, 3)
(l) Rótulos de validação: (276, 3)
(m) Total de imagens para testes: 140
(n) Total para cada classe: [58, 58, 24]
(o) Dados de testes: (140, 256, 256, 3)
(p) Rótulos de testes: (140, 3)
(q) Código: Colab

3.2) PARÂMETROS DE TREINAMENTO

(a) batch_size = 16
(b) epochs = 25

3.3) REDE: CNN

(a) Arquitetura, ver código.
(b) Duração 4.0 minutos para treinamento de 25 épocas
(c) Epoch 1/25: 23s 186ms/step – loss: 46.0286 – precision: 0.3773 – recall: 0.0856 – val_loss: 1.0616 – val_precision: 0.0000e+00 – val_recall: 0.0000e+00
(d) Epoch 25/25: 9s 148ms/step – loss: 0.4370 – precision: 0.8435 – recall: 0.8392 – val_loss: 0.9856 – val_precision: 0.7445 – val_recall: 0.7391
(e) Gráficos da precisão e sensibilidade (recall) durante o treinamento.



(f) Avaliação do modelo com o conjunto de testes.
Comando: model_cnn.evaluate(test_data, test_labels, verbose=2)
Saída: 5/5 – 1s – loss: 0.9815 – precision: 0.7410 – recall: 0.7357 – 506ms/epoch – 101ms/step

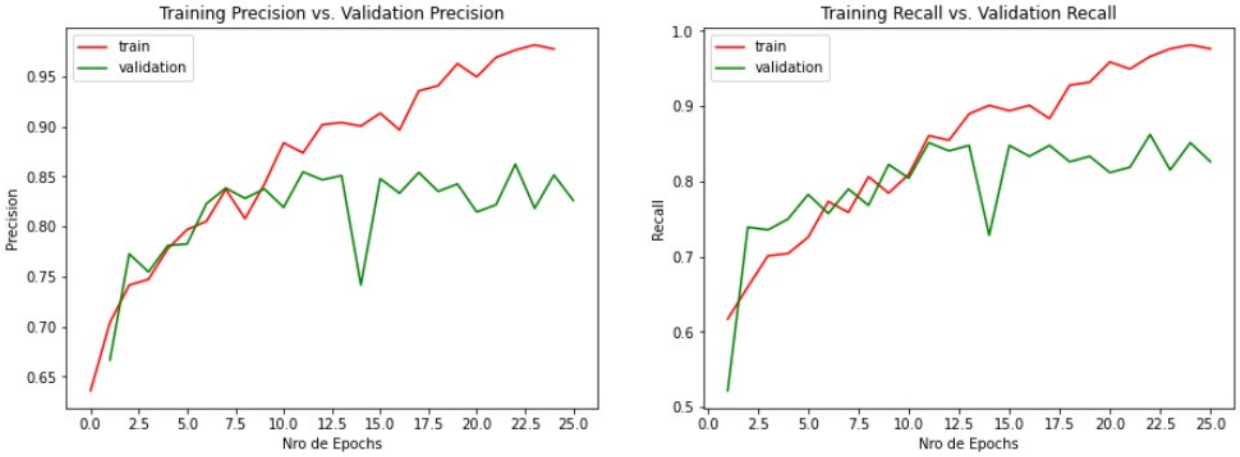
3.4) REDE: VGG19

(a) Duração 443.91 7.4 minutos para treinamento de 25 épocas
(b) Epoch 1/25: 31s 378ms/step – loss: 2.5848 – precision: 0.6361 – recall: 0.6172 –

val_loss: 0.8642 – val_precision: 0.6667 – val_recall: 0.5217

(c) Epoch 25/25: 16s 257ms/step – loss: 0.0856 – precision: 0.9773 – recall: 0.9763 – val_loss: 1.3034 – val_precision: 0.8261 – val_recall: 0.8261

(d) Gráficos da precisão e sensibilidade (recall) durante o treinamento.



(g) Avaliação do modelo com o conjunto de testes

Comando: model_vgg19.evaluate(test_data, test_labels, verbose=2)

Saída: 5/5 – 2s – loss: 1.3359 – precision: 0.8429 – recall: 0.8429 – 2s/epoch – 342ms/step

4) TERCEIRA ESTRATÉGIA

Aplicado filtro bilateral em todas as imagens.

4.1) ANÁLISE DOS DADOS:

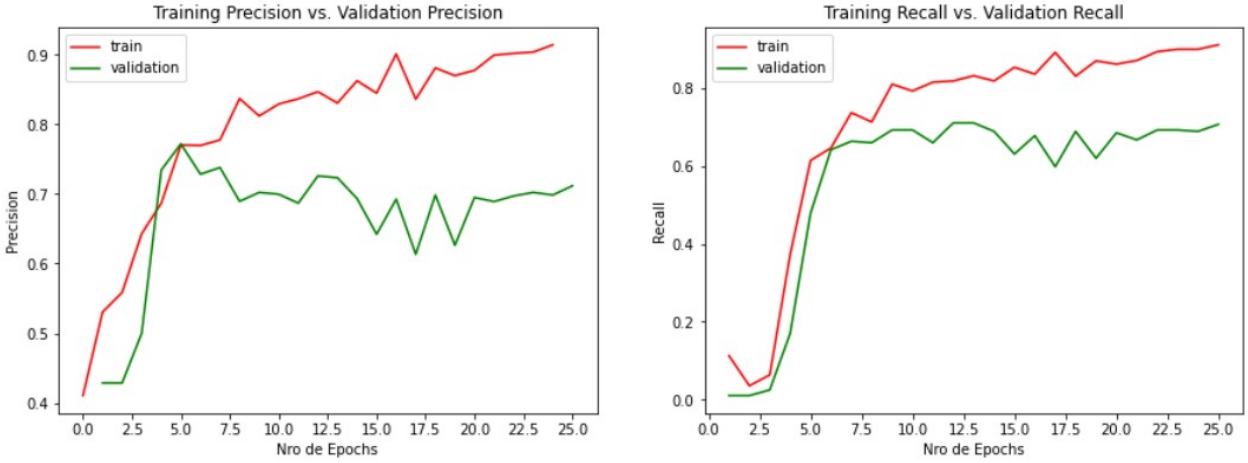
(a) Quantidade total de Imagens: 1387
(b) Classes/rótulos: ['integra' 'rara' 'confluente_esparsa']
(c) Tamanho de cada classe: [574, 574, 239]
(d) Tamanho percentual de cada classe: [41.4, 41.4, 17.2]
(e) Total para cada classe: [402, 402, 167]
(f) Total de imagens para treinamento: 971
(g) Total para cada classe: [114, 114, 48]
(h) Dados de treinamento: (971, 256, 256, 3)
(i) Rótulos de treinamento: (971, 3)
(j) Total de imagens para validação: 276
(k) Total para cada classe: [58, 58, 24]
(l) Dados de validação: (276, 256, 256, 3)
(m) Rótulos de validação: (276, 3)
(n) Total de imagens para testes: 140
(o) Dados de testes: (140, 256, 256, 3)
(p) Rótulos de testes: (140, 3)
(q) Código: Colab

4.2) PARÂMETROS DE TREINAMENTO

(a) batch_size = 16
(b) epochs = 25

4.3) REDE CNN

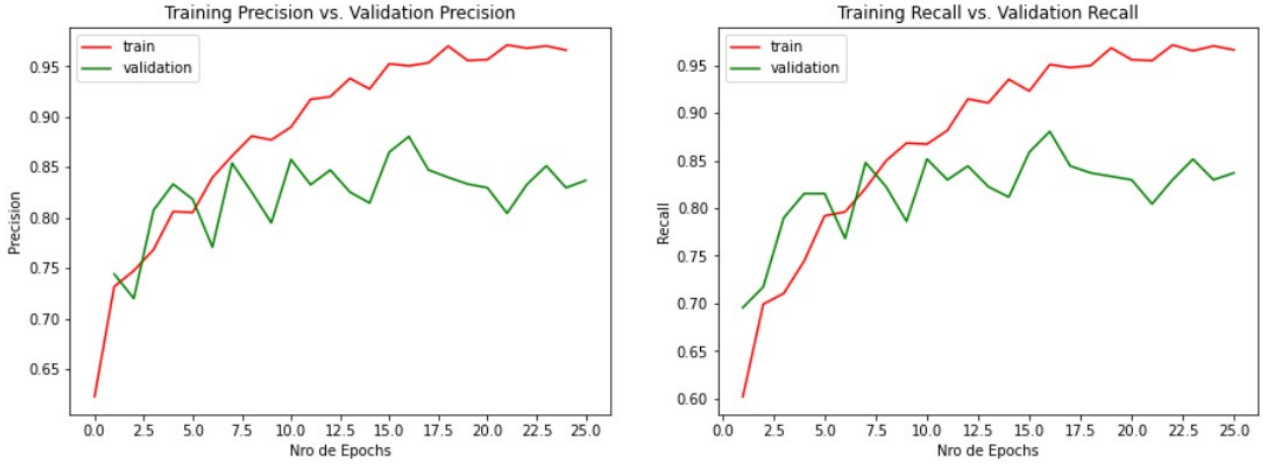
(a) Duração 263.46 segundos (4.4 minutos) para treinamento de 25 épocas
(b) Epoch 1/25 - 22s 184ms/step – loss: 59.0195 – precision: 0.4104 – recall: 0.1133 – val_loss: 1.0665 – val_precision: 0.4286 – val_recall: 0.0109
(c) Epoch 25/25 – 9s 148ms/step – loss: 0.2111 – precision: 0.9142 – recall: 0.9104 – val_loss: 1.6075 – val_precision: 0.7117 – val_recall: 0.7065
(d) Gráficos da precisão e sensibilidade (recall) durante o treinamento.



(e) Avaliação do modelo com o conjunto de testes
Comando: model_cnn.evaluate(test_data, test_labels, verbose=2)
Saída: 5/5 – 3s – loss: 1.3735 – precision: 0.7279 – recall: 0.7071 – 3s/epoch – 527ms/step

4.4) REDE: VGG19

(a) Duração 443.82 seconds (7.4 minutos) para treinamento de 25 épocas
(b) Epoch 1/25 – 30s 373ms/step – loss: 2.4197 – precision: 0.6227 – recall: 0.6022 – val_loss: 0.6105 – val_precision: 0.7442 – val_recall: 0.6957
(c) Epoch 25/25 – 15s 253ms/step – loss: 0.1559 – precision: 0.9660 – recall: 0.9660 – val_loss: 1.4630 – val_precision: 0.8370 – val_recall: 0.8370
(d) Gráficos da precisão e sensibilidade (recall) durante o treinamento.



(e) Avaliação do modelo com o conjunto de testes
Comando: model_vgg19.evaluate(test_data, test_labels, verbose=2)
Saída: 5/5 – 19s – loss: 4.5351 – precision: 0.8143 – recall: 0.8143 – 19s/epoch – 4s/step