UFMA - CEET
ALUNO: FILIPE CORREIA BELFORT

Este código foi desenvolvido no ambiente do Google Colab

BIBLIOTECAS UTILIZADAS:

Pandas (pd): Usada para manipulação e análise de dados, especialmente para trabalhar com estruturas de dados como DataFrames.

Glob: Usada para encontrar arquivos que correspondam a um determinado padrão de nome de arquivo.

Numpy (np): Oferece suporte para arrays e operações matemáticas de alto desempenho.

CV2 (OpenCV): Biblioteca utilizada para processamento de imagens, incluindo carregamento, manipulação e exibição de imagens.

Sklearn (sklearn):

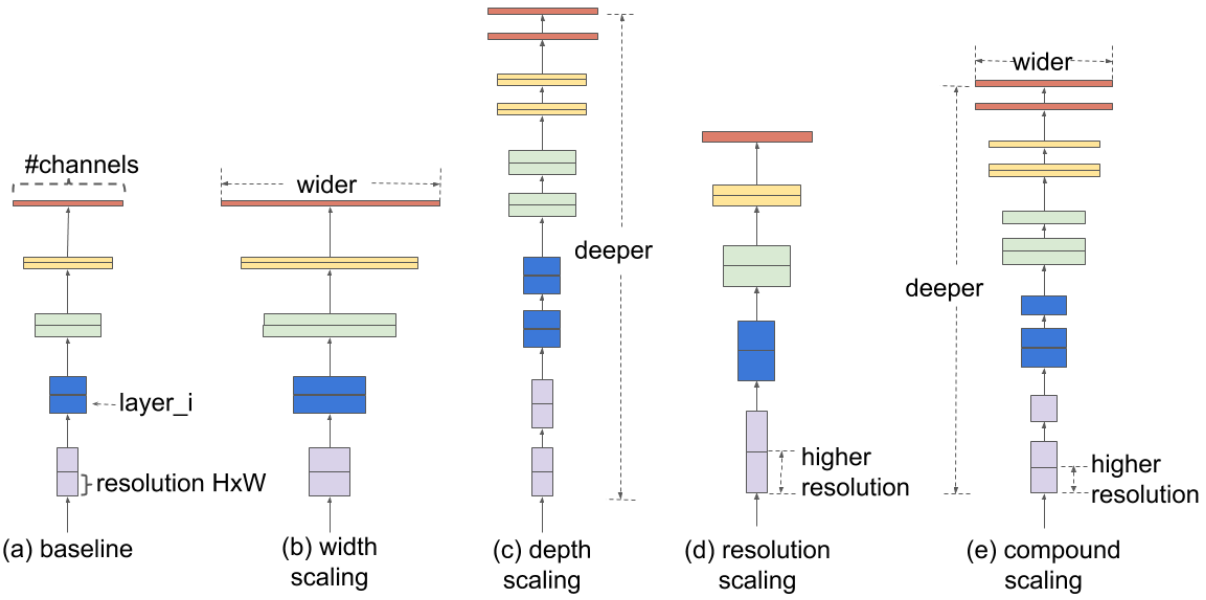
train_test_split: Para dividir conjuntos de dados em conjuntos de treino e teste.
RandomForestClassifier: Implementação de um classificador de floresta aleatória.
classification_report, confusion_matrix, accuracy_score: Métricas e ferramentas para avaliação de modelos de classificação.

TensorFlow (tf): Uma plataforma de código aberto para machine learning e deep learning.

Keras:

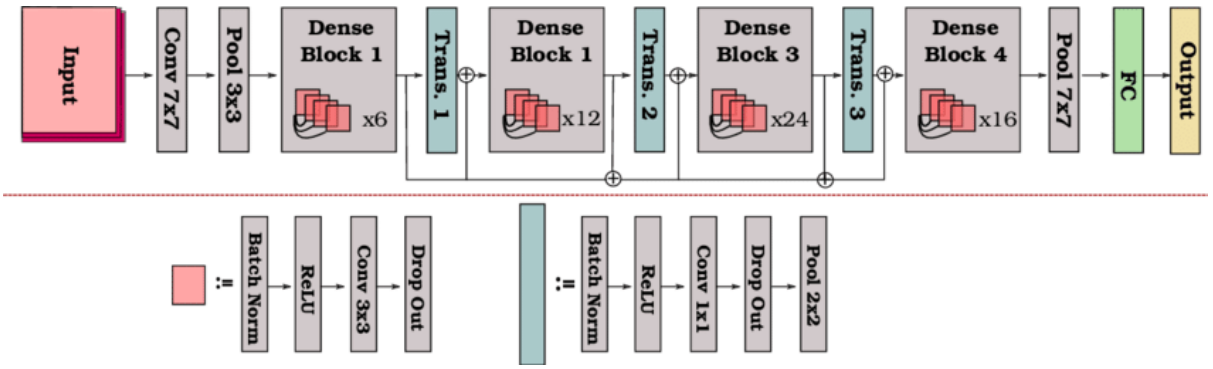
Keras: layers, models, load_model, optimizers, ImageDataGenerator, ModelCheckpoint: Diferentes funcionalidades e camadas fornecidas pela biblioteca Keras para construção e treinamento de modelos de deep learning.

Efficient Net: Uma familia de arquiteturas de rede neural disponíveis na Keras para transfer learning e classificação de imagens.



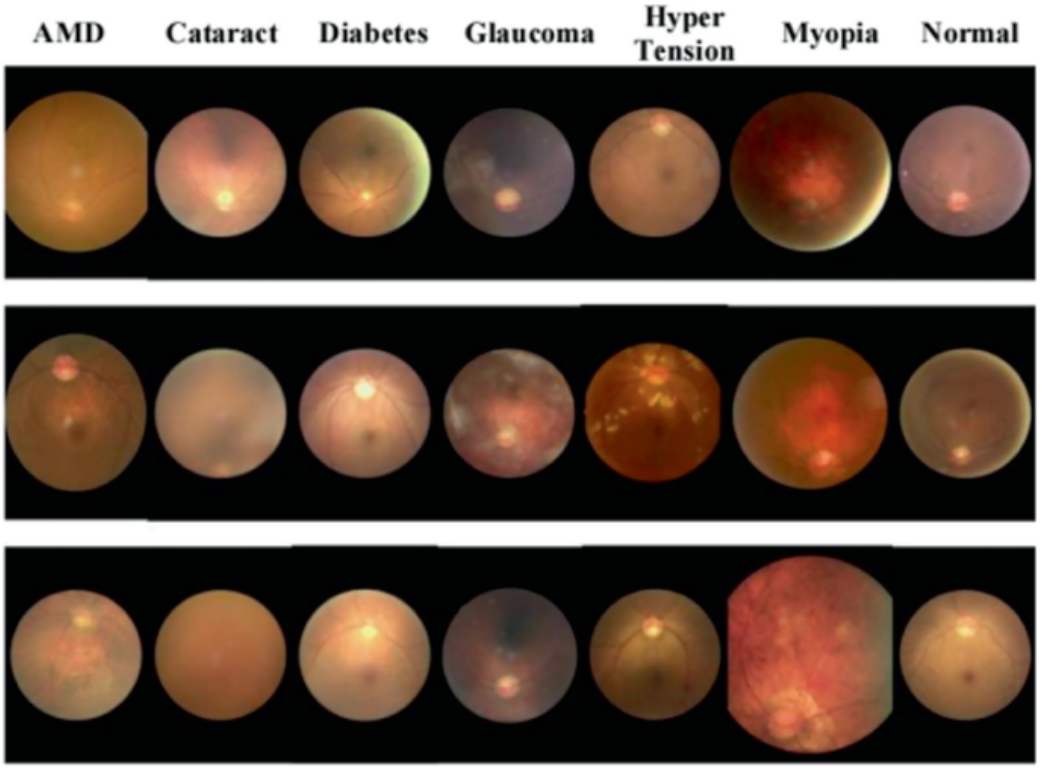
UFMA - CEET
ALUNO: FILIPE CORREIA BELFORT

DenseNet: A Dense Convolutional Network, é uma arquitetura de rede neural profunda conhecida por sua estrutura densamente conectada.



DATASET: ODIR 5k]

Classes utilizadas: Normal, Glaucoma e Retinopatia diabetica




```
Train: 1474
test: 507
val: 126
```

UFMA - CEET
ALUNO: FILIPE CORREIA BELFORT

PRÉ PROCESSAMENTOS:

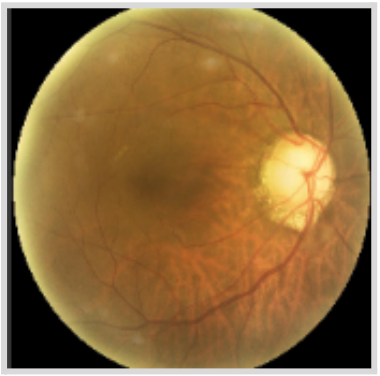
Redimensionamento para 224 x 224

Realiza um cropped na imagem:

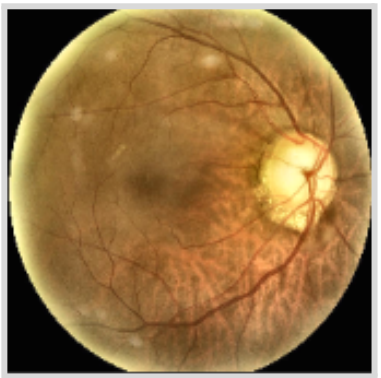
```python
def remove_border(image):

    # Converter para escala de cinza
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Aplicar threshold para binarizar a imagem
    _, thresh = cv2.threshold(gray, 1, 255, cv2.THRESH_BINARY)

    # Encontrar contornos na imagem binarizada
    contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    # Encontrar o maior contorno (o objeto principal)
    main_contour = max(contours, key=cv2.contourArea)

    # Obter as dimensões do retângulo que envolve o contorno principal
    x, y, w, h = cv2.boundingRect(main_contour)

    # Recortar a região de interesse (ROI) da imagem original
    cropped_image = image[y:y+h, x:x+w]

    return cropped_image
```

Aplicação do algoritmo CLAHE

ANTES



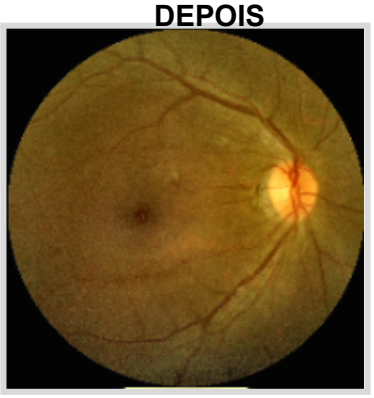
DEPOIS



UFMA - CEET
ALUNO: FILIPE CORREIA BELFORT

ANTES

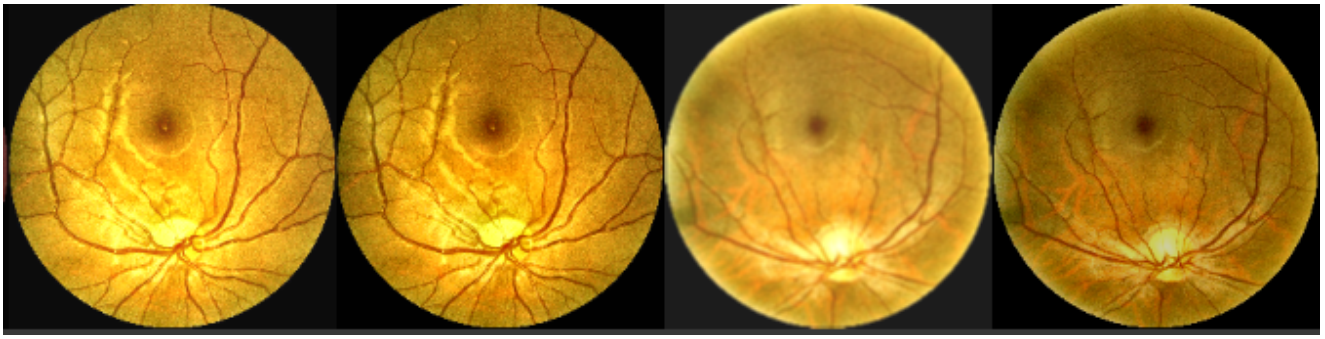
DEPOIS



DATA AUGMENTATION:

```
def data_augmentation(image):

    augmentation = A.Compose([
        A.RandomBrightnessContrast(),
        A.RandomGamma(),
        A.Blur(blur_limit=1)
    ])

    augmented_image = augmentation(image=image)['image']
    guarda.append(augmented_image)

    return augmented_image
```

Exemplo de imagens resultantes do data augmentation:



UFMA - CEET
ALUNO: FILIPE CORREIA BELFORT

Construção da rede:

```python
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential, load_model, Model
from keras.layers import Activation, Dropout, Flatten, Dense, GlobalAveragePooling2D

def buildModel(size, classes):
    baseModel = keras.applications.EfficientNetB2(include_top=False, weights="imagenet")
    baseModel.trainable = False

    for layer in baseModel.layers:
        if isinstance(layer, keras.layers.BatchNormalization):
            layer.trainable = True
        else:
            layer.trainable = False

    preprocess = keras.applications.efficientnet.preprocess_input  # Corrigindo o acesso ao preprocessamento

    inputs = keras.Input(shape=(size, size, 3))
    x = preprocess(inputs)  # Pré-processamento aplicado às entradas
    x = baseModel(x)
    x = GlobalAveragePooling2D()(x)
    x = Dense(128, activation='relu')(x)
    x = Dropout(0.2)(x)
    outputs = Dense(classes, activation='softmax')(x)

    modelFinal = Model(inputs=inputs, outputs=outputs)
    return modelFinal, baseModel

#congela ou descongela camadas
def freeze(model_final, until):
  for layer in model_final.layers[:until]:
    layer.trainable = False
  for layer in model_final.layers[until:]:
    layer.trainable = True

model, baseModel = buildModel(224, 3)
model.summary()
```

Adição do Class weights:

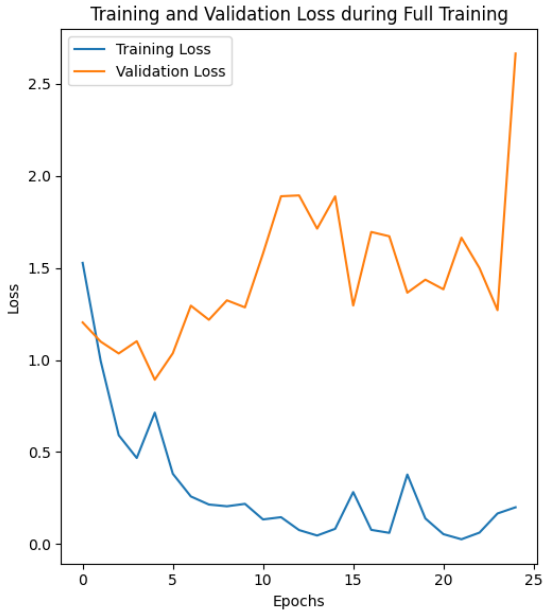
```python
class_weights = {
    0: 1.0,  # Peso para a classe 0
    1: 2.0,  # Peso para a classe 1
    2: 10.0   # Peso para a classe 2
}


H_warmup = model.fit(
    newX,
    newY,
    batch_size=batch_size,
    epochs=5,  # Apenas algumas épocas para warm-up
    validation_data=(x_val, y_val),
    class_weight=class_weights,
)
```

UFMA - CEET
ALUNO: FILIPE CORREIA BELFORT

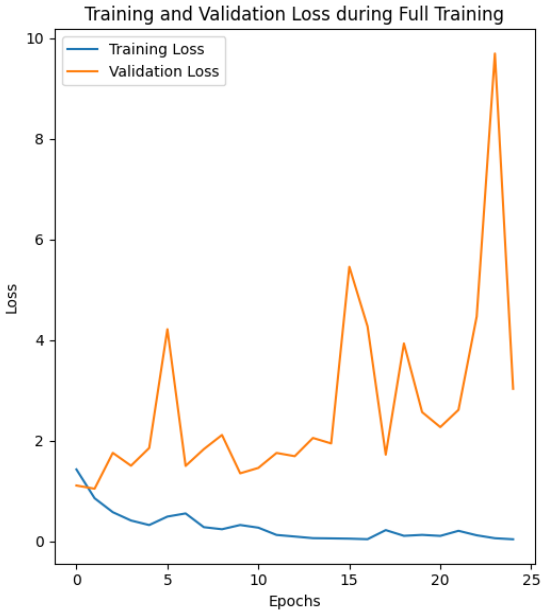
RESULTADOS:

RESULTADOS			
REDE	PRECISÃO	RECALL	F1-SCORE
EfficientNet B1 (Normal)	0.73	0.58	0.65
EfficientNet B1 (Retinopatia)	0.42	0.54	0.47
EfficientNet B1 (Glaucoma)	0.31	0.65	0.42
EfficientNet B2 (Normal)	0.78	0.51	0.62
EfficientNet B2 (Retinopatia)	0.37	0.67	0.48
EfficientNet B2 (Glaucoma)	0.31	0.52	0.39
EfficientNetV2 B1 (Normal)	0.74	0.68	0.71
EfficientNetV2 B1 (Retinopatia)	0.38	0.51	0.44
EfficientNetV2 B1 (Glaucoma)	0.55	0.21	0.31
EfficientNetV2 B2 (Normal)	0.78	0.40	0.53
EfficientNetV2 B2 (Retinopatia)	0.34	0.78	0.48
EfficientNetV2 B2 (Glaucoma)	0.38	0.39	0.37
DenseNet 121 (Normal)	0.75	0.44	0.55
DenseNet 121 (Retinopatia)	0.38	0.75	0.51
DenseNet 121 (Glaucoma)	0.61	0.47	0.53

EfficientNetB1:

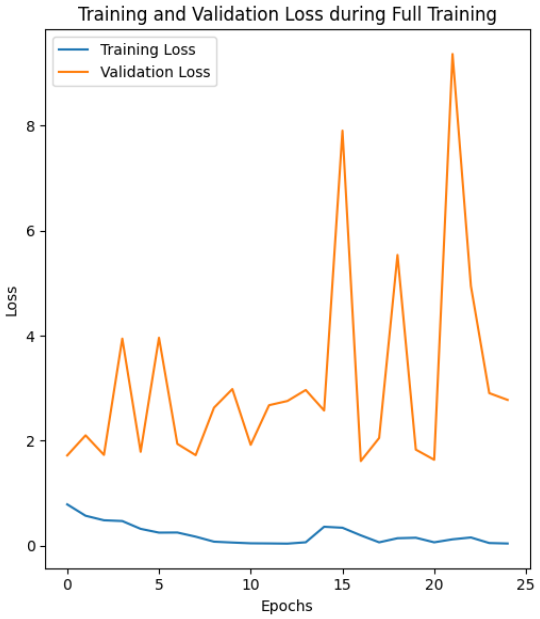


EfficientNetB2:

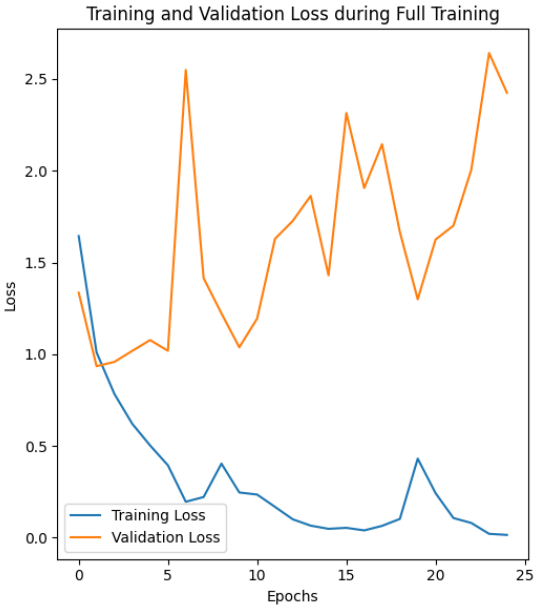


UFMA - CEET
ALUNO: FILIPE CORREIA BELFORT

EfficientNetV2 B1:



EfficientNetV2 B2:



Conclusão: Após a resolução de alguns problemas anteriores na construção da rede, resolver os problemas de distorção das imagens da base, realizando aplicação de cropped nas imagens, aplicação de outros tipos de data augmentation como, adição aleatória de gamma, blur e brilho contraste, mais a adição de pesos para cada classe através do class weights, apesar de muitos resultados ainda estarem abaixo de 50%, existem alguns resultados mas estáveis como a DenseNet 121 que obteve F1-score respectiovo de 55% pra classe normal, 51% pra retinopatia e 53% pra glaucoma, e por vez, a rede não zerou a métrica de nenhuma classe.