LAMIA
Machine Learning for Industry

Detector de doenças oculares

Lucas Scheffer Hundsdorfer

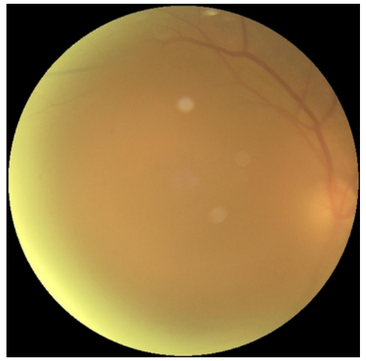
Objetivos do meu projeto final

- **Conseguir identificar as doenças oculares através de uma rede neural.**
- **Aplicar o fine tuning em um modelo pré treinado.**
- **Treinar um modelo do 0**
- **Analisar as métricas e comparar os modelos.**

Base de dados

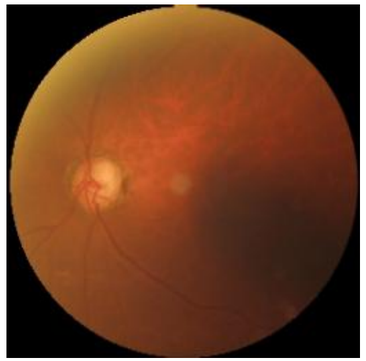
- Catarata - 1038 exemplares.
- Retinopatia diabética - 1098 exemplares.
- Glaucoma - 1007 exemplares.
- Normal - 1074 exemplares.

Catarata



Retinopatia diabética

Glaucoma



Normal

Link da base de dados

Modelos a serem utilizados

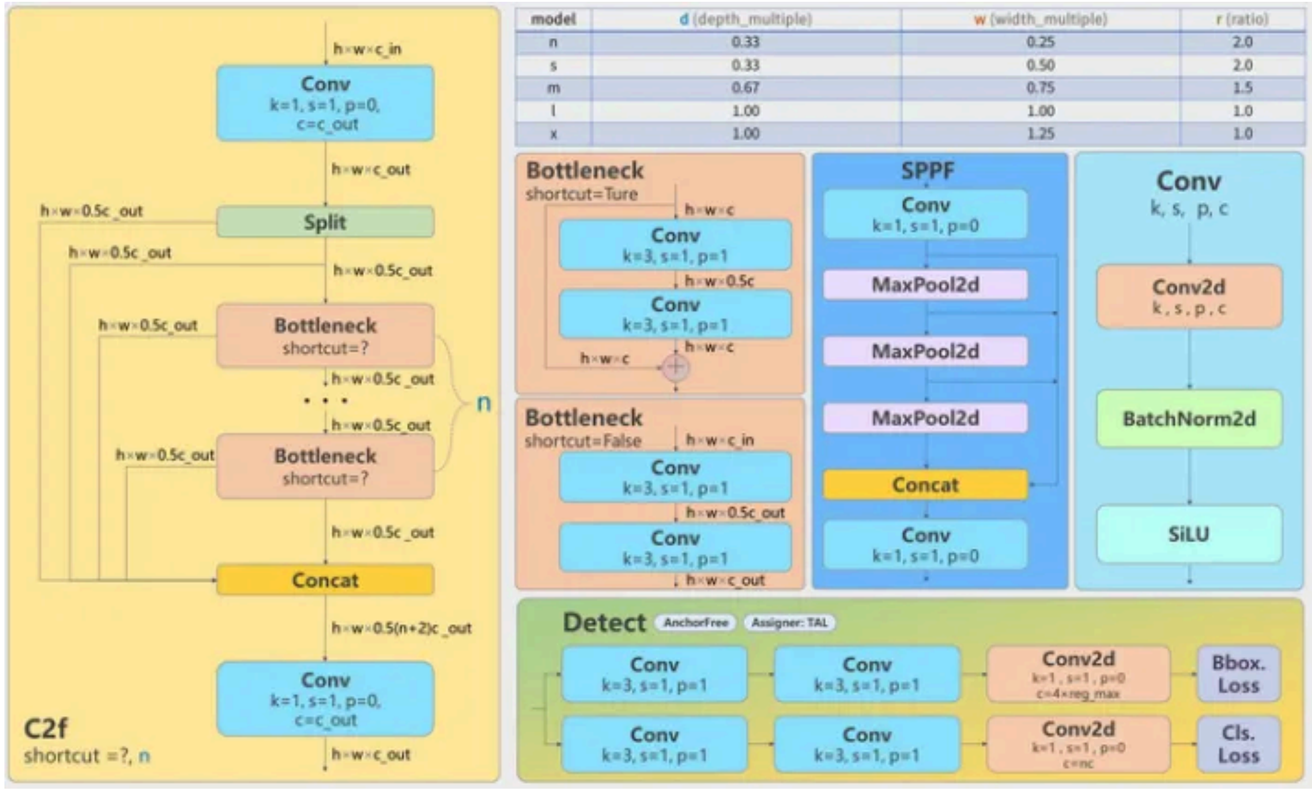
- **YoloV8:**
  - **Um modelo concreto da Ultralytics.**
  - **Alta velocidade**
  - **Adaptabilidade.**
  - **Fácil implementação**
- **Modelo Próprio:**
  - **Arquitetura desenvolvida do zero**
  - **Menor dependência de frameworks externos.**
  - **Controle total do processo**
  - **Flexibilidade estrutural**

Arquitetura Yolo v8

LAMIA
Machine Learning for Industry

Estrutura principal:

- Backbone
- Neck
- Head



Arquitetura CNN própria

LAMIA
Machine Learning for Industry

Bloco convolucional:

- Camada convolucional
- Batch normalization
- Camada convolucional
- Batch normalization
- Pooling 2D
- Dropout

Bloco denso:

- Global Pooling
- Camada densa
- Batch Normalization
- Dropout
- Camada densa
- Batch Normalization
- Dropout
- Camada densa

```
model = models.Sequential([
        layers.Input(shape=(224, 224, 3)),
        layers.Rescaling(1./255),

        layers.Conv2D(64, (3, 3), padding='same', activation='relu', input_shape=(224,224,3)),
        layers.BatchNormalization(),
        layers.Conv2D(64, (3, 3), padding='same', activation='relu'),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),
        layers.SpatialDropout2D(0.15),

        layers.Conv2D(128, (3, 3), padding='same', activation='relu'),
        layers.BatchNormalization(),
        layers.Conv2D(128, (3, 3), padding='same', activation='relu'),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),
        layers.SpatialDropout2D(0.20),

        layers.Conv2D(256, (3, 3), padding='same', activation='relu'),
        layers.BatchNormalization(),
        layers.Conv2D(256, (3, 3), padding='same', activation='relu'),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),
        layers.SpatialDropout2D(0.25),

        layers.Conv2D(256, (3, 3), padding='same', activation='relu'),
        layers.BatchNormalization(),
        layers.Conv2D(256, (3, 3), padding='same', activation='relu'),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),
        layers.SpatialDropout2D(0.30),


        layers.GlobalAveragePooling2D(),
        layers.Dense(512, activation='relu'),
        layers.BatchNormalization(),
        layers.Dropout(0.4),
        layers.Dense(128, activation='relu'),
        layers.BatchNormalization(),
        layers.Dropout(0.3),
        layers.Dense(4, activation='softmax')
```

Métodos utilizados:

LAMIA
Machine Learning for Industry

YOLO

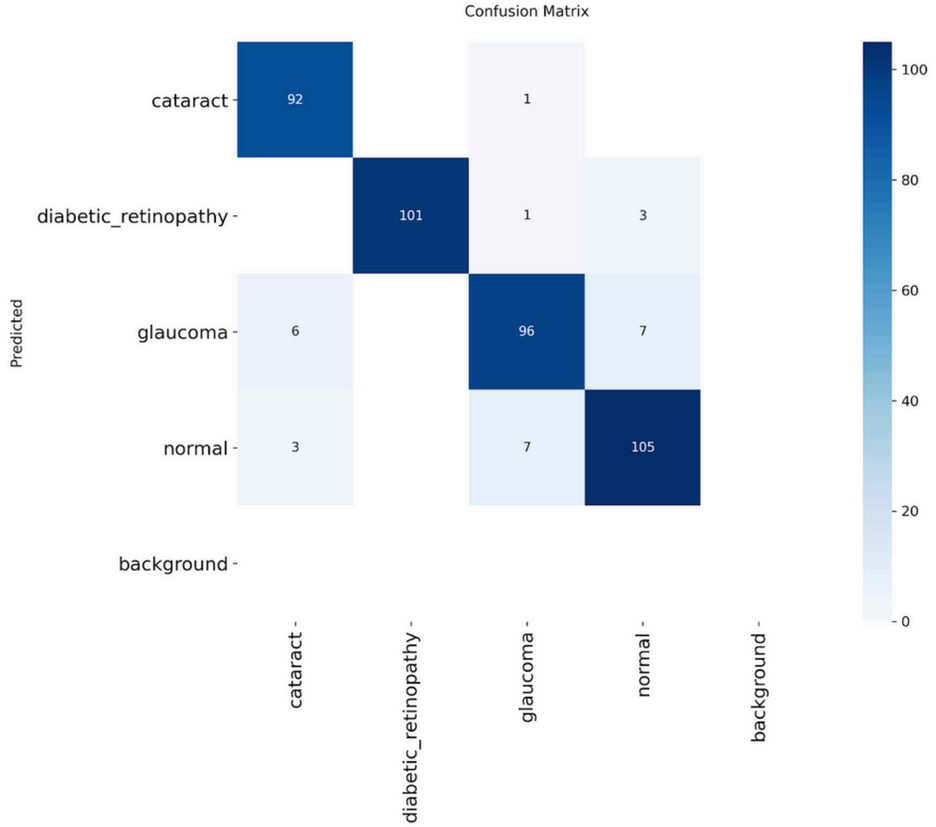
- Normalização entre 0 e1
- 224x224 pixels
- Data augmentation leve
- Batch Size de 16
- 100 épocas
- Automatic mixed precision
- AdamW

Modelo personalizado

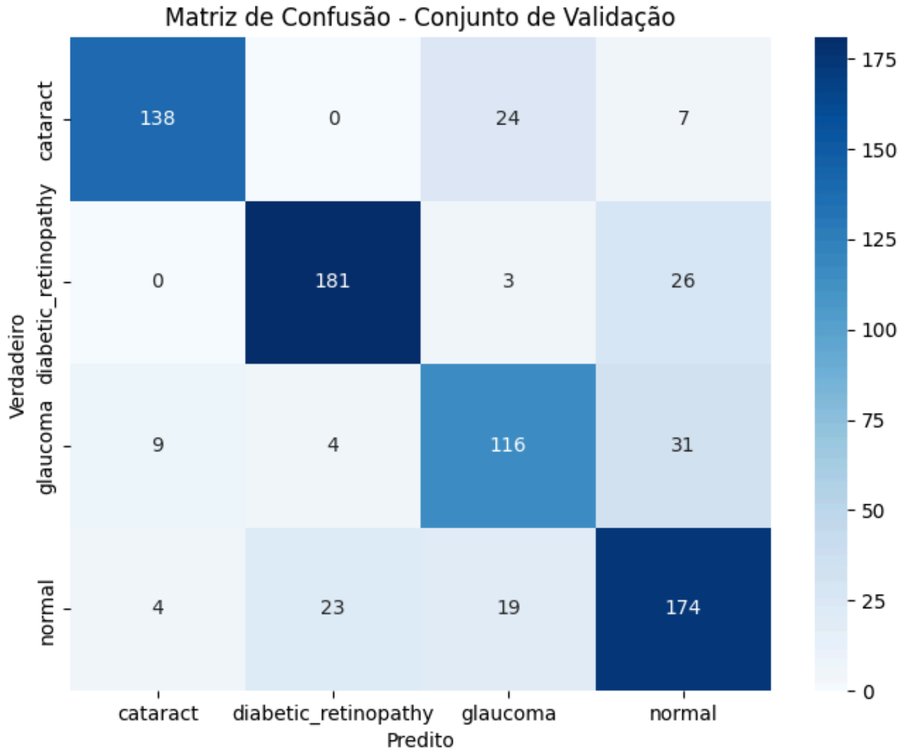
- Normalização entre 0 e 1
- 224x224 pixels
- Data augmentation leve
- Batch size de 32
- 80 épocas
- Early Stopping
- AdamW

Resultados obtidos:

YOLO



Modelo personalizado



Conclusões:

LAMIA
Machine Learning for Industry

Quando é bom usar um modelo pronto:

- Base de dados menor
- Projeto com prazo curto
- Alta precisão e estabilidade
- Aplicações em produção
- Ajustes mínimos

Quando é bom treinar um modelo do zero:

- Dados muito específicos
- Quiser maior transparência
- Quiser maior controle
- Testes e estudos de arquiteturas
- Projeto com longo prazo

Obrigado pela sua atenção!