# **Classificação de imagens na triagem de pacientes com deficiência visual**

Douglas Jonatas do Carmo Dias¹; Ana Beatriz Pereira Sette2

¹ Centro Universitário de Lins. Engenheiro de Computação. Rua: José Bonifácio n°244 – Ribeiro; 16401-115. Lins, São Paulo, Brasil.

2 Universidade Federal de Viçosa. Doutora em Economia Aplicada. Departamento de Economia Rural, Avenida Purdue, s/nº, Edifício Edson Potsch Magalhães – Campus Universitário; 36570-900. Viçosa, Minas Gerais, Brasil.

# **Classificação de imagens na triagem de pacientes com deficiência visual**

# **Resumo**

A automação de processos cada vez mais crescente obriga a existência de aplicações deste tipo na área de visão computacional. Visão computacional é uma ferramenta que pode ser útil em todas as áreas em que estiver. O objetivo deste trabalho foi a aplicação de redes neurais convolucionais para a identificação de existência ou não de quatro classes de objetos em imagens. Para isso, utilizou-se técnicas como “fine-tuning” e “data augmentation”. Este trabalho deu grande importância à explicação de conceitos de visão computacional visando a facilitação da replicação dos procedimentos aqui aplicados. As principais redes neurais convolucionais utilizadas neste trabalho foram: VGG16, Xception e ResNet101. Como principais resultados, foram obtidos as matrizes de confusão destas redes neurais convolucionais e algumas métricas oriundas destas matrizes. Por fim, comparou-se os resultados obtidos em cada umas das redes e observou-se um melhor desempenho da Xception diante das outras redes neurais convolucionais. Esta rede apresentou métricas de acurácia, precisão, sensibilidade e “f1-score” superiores a 98%. As diferenças de resultados entre as redes foram analisadas e foram sugeridas novas linhas de ação para o melhoramento dos resultados destas redes.

**Palavras-chave:** redes neurais convolucionais; objetos; imagens; visão computacional.

# **Introdução**

Devido ao crescimento das divícias sociodemográficas e da perspectiva de vida mais longa, vários países ao redor do mundo começaram a notar sua população atingindo a idade adulta, aumentando assim a idade média da população e como consequência trazendo mudanças na carga de doenças para doenças não transmissíveis e deficiências. A maioria das principais causas de deficiência visual é a catarata (R. Bourne et al., 2021).

Carata é a causa mais prevalente de cegueira reversível no mundo, sendo responsável por aproximadamente 50% dos 50 milhões de casos. Por definição, se refere a opacidade do cristalino, podendo ser adquirida, o que inclui o processo de envelhecimento. As fontes causais da catarata não foram plenamente elucidadas, porém existem fatores de riscos que devem ser evitados, pois a catarata é uma questão de saúde pública intimamente relacionada com o aumento da expectativa de vida no Brasil e no mundo. (V. O. Domingues et al., 2016).

Outra doença que contribui para a cegueira no mundo é o glaucoma, destacando-se como a principal causa de cegueira e deficiência visual irreversível no Brasil e no mundo. Estudos projetam que globalmente, em 2020, 76 milhões de indivíduos sejam portadores de glaucoma e que este número deve atingir a marca de 95,4 milhões em 2030. Dentre os muitos tipos de glaucoma, os mais frequentes são os glaucomas primários de ângulo aberto (GPAA) e de ângulo fechado (GPAF). A distribuição destes tipos de glaucoma sofre influência da área geográfica e da etnia da população. No Brasil, assim como no mundo ocidental, o mais comum é sem dúvida o GPAA, responsável por aproximadamente por 80% dos casos. Tanto a prevalência quanto a incidência do glaucoma aumentam com a idade, sofrendo grande influência da raça do indivíduo. Aos 40 anos, surgem anualmente aproximadamente 1,6 novos casos de glaucoma para cada 100.000 habitantes e aos 80 anos, 94,3/100.000 habitantes. A prevalência para indivíduos brancos e negros, respectivamente, na faixa etária de 73 a 74 anos é de 3,4% e 5,7%. Estas taxas aumentam para 9,4% e 23,2% para estes mesmos grupos se considerarmos a faixa etária de 75 anos ou mais.

Sendo o glaucoma uma doença de origem genética, a sua prevenção primária (evitar o seu aparecimento) ainda é impraticável e o único modo de se evitar a progressão para a cegueira é através da prevenção secundária (diagnóstico precoce e tratamento eficaz), prevenção terciária (limitar as sequelas da doença e realizar a reabilitação) e a prevenção quaternária (evitar intervenções diagnósticas e/ou terapêuticas inapropriadas, iatrogênicas ou eticamente questionáveis) (R.A.P. Guedes.,2021).

A triagem para os pacientes tanto com possibilidade de adquirirem catarata ou glaucoma é de extrema valia pois utilizando a classificação de imagem irá fortalecer a prevenção secundário, ou seja, possibilitando um diagnóstico precoce e como consequência possibilitando um tratamento eficaz.

# **Material e Métodos**

Os materiais utilizados para o desenvolvimento do trabalho foram:

* + Google Colaboratory;
  + Python;
  + Dataset;
  + Matplotlib;
  + Pyplot;
  + TensorFlow;
  + TensorFlow Hub;
  + Keras;
  + efficientnetv2-b2-21k;
  + Google Driver;

O **Google Colaboratory** mais conhecido como Google Colab ou Colab é um serviço de nuvem gratuito hospedado pela próprio Google, é uma ferramenta que permite você criar código fonte e texto com imagens e tudo isso sem a necessidade de fazer download de software, além de te acesso a GPUs sem custo financeiro e com possibilidade de compartilhamento de uma forma fácil.

**Python** é uma linguagem de programação de alto nível e de uso geral. Sua filosofia de design enfatiza a legibilidade do código com o uso de recuo significativo**.** Python é uma das linguagens mais usadas quando se trata de data science, principalmente pela sua linguagem flexível e código aberto, o que facilita seu uso para computações mais quantitativas, potencializado pelas suas enormes bibliotecas que permite uma melhor análise de dados.

Os **Datasets** ou conjunto de dados, são o principal insumo dos processos de análise de dados, para esse trabalho foi utilizado um conjunto de imagem já rotuladas retiradas do Kaggle, na seção Coleta e tratamento de dados será explicado sobre o mesmo.

**Matplotlib** é uma biblioteca da linguagem de programação Python, utilizada para visualização de dados e plotagem gráfica. Ela é utilizada pela extensão de matemática numérica do Python, a NumPy, e pela biblioteca SciPy.

**Pyplot** é um módulo Matplotlib que fornece uma interface semelhante ao MATLAB. Matplotlib é projetado para ser tão usável quanto MATLAB, com a capacidade de usar Python e a vantagem de ser gratuito e de código aberto.

**TensorFlow**, a estrutura de aprendizado de máquina muito popular para o desenvolvimento de ML, é uma biblioteca de software de código aberto gratuita e para aprendizado de máquina e inteligência artificial.

O **TensorFlow Hub** é um repositório de modelos de machine learning treinados prontos para ajustes finais e implantação em qualquer lugar. Possibiliatando assim a reutilização de modelos já treinados.

o projeto foi desenvolvido em python utilizando as bibliotecas: matplotlib.pyplot, numpy, tensorflow, tensorflow\_hub e keras. Preprocessing. Image.

O modelo utilizado para fazer a classificação das imagens foi o efficientnetv2-b2-21k, é um modelo de classificação encontrado no TensorFlow Hub, tem vários modelos de classificação e para o desenvolvimento do projeto esse é o que se adequou melhor.

---------------------------------------------------------------------------------------------------------------------------

## Coleta e tratamento de dados

Os dados coletados para esse trabalho foram retirados do Kaggle. As imagens já estavam com os rótulos e as mesmas foram colocas no google driver, sendo possível fazer o carregamento para a área de trabalho do Google Colab.

Tratando-se de qualquer modelo de machine learning ou deep learning uma das partes mais importantes no treinamento de qualquer modelo é a parte do pré-processamento dos dados de entrada. Sendo alguns pontos importantes para o mesmo:

* Escolha de um conjunto de imagens já com os rótulos e configuração do tamanho;

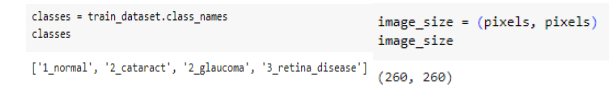


Figura 1: Rótulo das imagens e configuração dos pixels

Fonte: Dados originais da pesquisa

Após a seleção do conjunto de imagens já com os rótulos é necessário definir o tamanho das imagens de acordo com o modelo utilizado, nesse caso as imagens terão um tamanho de 260x260 pixels

* Divisão das imagens para o treino;

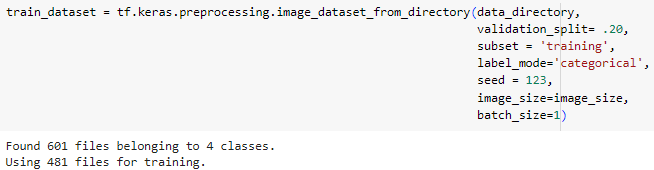


Figura 2: Treino do dataset

Fonte: Imagem proveniente da pesquisa

Para fazer o treinamento foi utilizado 80% das imagens (481) e ficando 20% (120) para o teste(validação) para isso foi utilizado o método split, a etiqueta desse modelo será por categoria, ou seja: [‘1\_normal’, ‘2\_cataract’, ‘2\_glaucoma’, ‘3\_retina\_disease’], toda vez que a função for chamada será composta pelas mesmas imagens, o tamanho das imagens serão 260x260 e o número de amostras processadas antes do modelo ser atualizado será 1.

* Pré-processamento e “criando” das imagens.

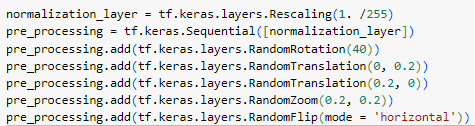


Figura 3: Adicionando e “criando” as imagens

Fonte: Autor

Foi feito a transformação de dados (com normalization) que alinha os valores dos dados a uma escola comum, utilizado o rescaling para reduzindo a escala (pixels) das imagens de 0-255 para 0-1. Como será passado uma sequencial de camadas para a rede neural será utilizado o Sequential. Também serão adicionadas algumas imagens no pré- processamento, pois pode ser que as imagens separadas para o treinamento não sejam o suficiente, logo será necessário “criar” imagens para o mesmo, sendo adicionadas algumas imagens aleatórias com algumas modificações como: girando 40°, deslocando 20% das imagens no horizontal e vertical, aumentando o zoom em 20% e com o zoom reduzido em 20% e girando as imagens horizontalmente.

* Divisão das imagens para o teste.

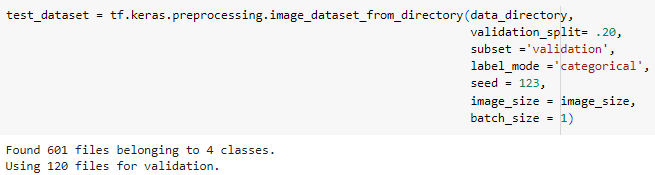


Figura 4: Imagens para o teste

Fonte: Criado pelo Autor

O preprocessamento adotado neste trabalho englobou o embaralhamento das imagens antes da divisão em conjunto de treino, validação e teste, o redimensionamento da imagem para o formato requerido pelas redes neurais convolucionais e a transformação da imagem em um objeto do tipo “numpy.array”, que é uma espécie de representação de tensores em “python”.

Para o embaralhamento das imagens utilizou-se a função “train\_test\_split” do pacote “Scikit-Learn”. Essa função é bastante empregada para dividir um conjunto de dados em dois subconjuntos distintos: um conjunto de treinamento e um conjunto de teste. Essa divisão é essencial para avaliar o desempenho do modelo de forma realista e evitar problemas de “overfitting”. Os resultados foram considerados satisfatórios, sendo todos eles bem altos. Ressalta-se que as menores métricas obtidas pela VGG16 foram nas classes avião e drone. Este resultado abaixo da média nestas duas classes faz sentido, pois ambas podem possuir algumas semelhanças (planos de fundo parecidos, por exemplo) que geram certa confusão na CNN. Ressalta-se que embora estes resultados tenham sido abaixo da média das outras classes, ainda assim eles estão bem altos de forma absoluta.

As Figuras 15 e 16 apresentam o gráfico de perdas e de acurácias durante o treinamento (dados de treinamento e validação foram testados nesta fase). As perdas representam a diferença entre os valores preditos e os valores reais durante a fase de treinamento.

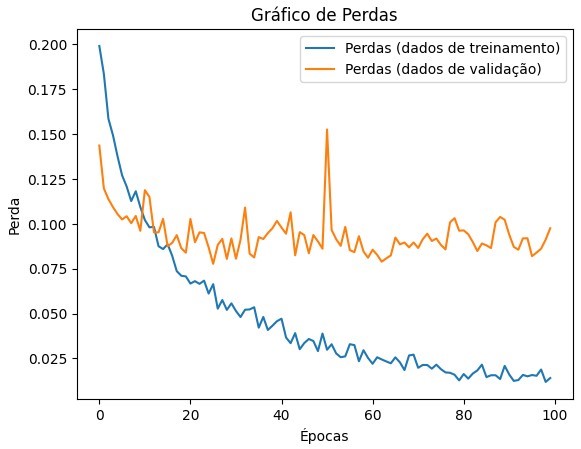


Figura 15: Gráfico de perdas da VGG16

Fonte: Resultados originais da pesquisa

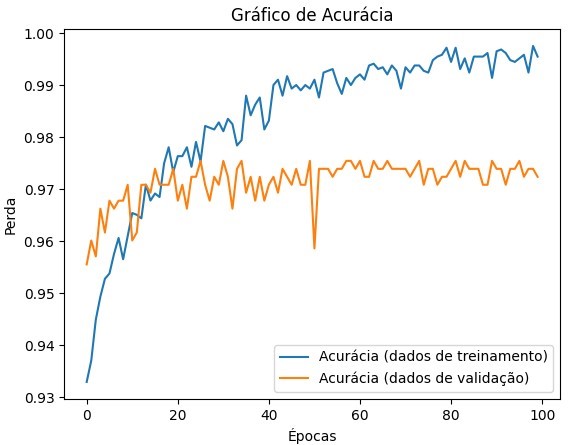


Figura 16: Gráfico de acurácias da VGG16

Fonte: Resultados originais da pesquisa

Nas figuras apresentadas acima, percebe-se o fenômeno de “overfitting”, pois a partir aproximadamente da época 20, embora as perdas e acurácias nos dados de treinamento apresentem resultados melhores, nos dados de validação estas métricas se mantém praticamente constantes.

## Xception

Abaixo apresenta-se a Tabela 4. Nesta tabela pode-se ver as matrizes de confusão de cada uma das classes de imagens. Abaixo de cada matriz de confusão há as métricas daquela classe.

Tabela 4: Matriz de confusão e métricas da Xception

|  |  |  |  |
| --- | --- | --- | --- |
| Aviões | | Carros | |
| **(TP)** 519 | **(FP)** 3 | **(TP)** 490 | **(FP)** 1 |
| **(FN)** 2 | **(TN)** 128 | **(FN)** 0 | **(TN)** 161 |
| Acurácia | 99% | Acurácia | 99% |
| Precisão | 98% | Precisão | 99% |
| Sensibilidade | 98% | Sensibilidade | 100% |
| “f1-score” | 98% | “f1-score” | 100% |
| Drones | | Navios | |
| **(TP)** 445 | **(FP)** 2 | **(TP)** 495 | **(FP)** 1 |
| **(FN)** 5 | **(TN)** 200 | **(FN)** 0 | **(TN)** 156 |
| Acurácia | 99% | Acurácia | 99% |
| Precisão | 99% | Precisão | 99% |
| Sensibilidade | 100% | Sensibilidade | 100% |
| “f1-score” | 100% | “f1-score” | 100% |

Fonte: Resultados originais da pesquisa

As Figuras 17 e 18 apresentam os gráficos de perda e acurácia durante o treinamento da Xception para os dados de treinamento e validação.

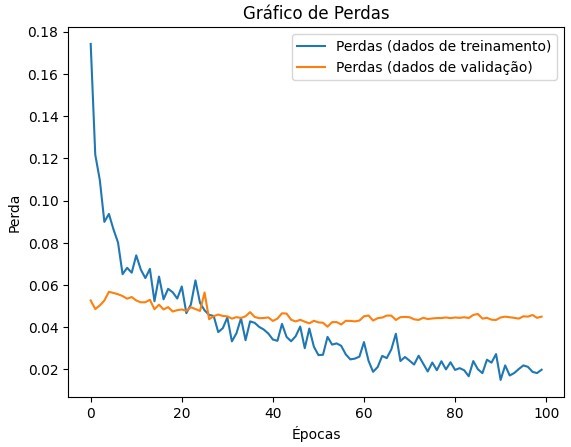


Figura 17: Gráfico de perdas da Xception

Fonte: Resultados originais da pesquisa

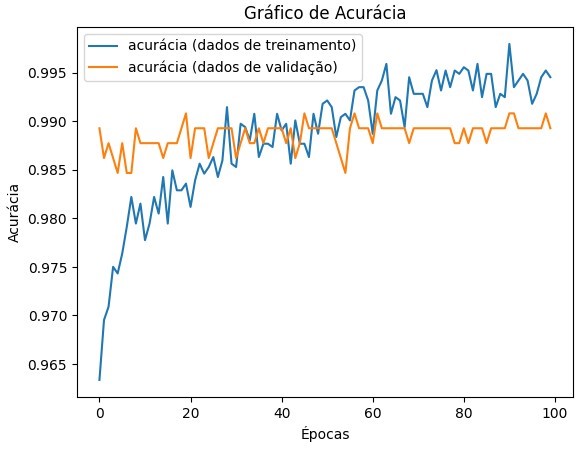


Figura 18: Gráfico de acurácias da Xception

Fonte: Resultados originais da pesquisa

Analisando os gráficos acima, percebe-se que a partir aproximadamente da época 30 do treinamento tanto as acurácias quanto as perdas nos dados de validação se mantém constante, enquanto que nos dados de treinamento elas apresentam melhora. Isso é um sinal de “overfitting”. A Xcepion está capturando algumas “features” bem específicas dos dados de treinamento e não está generalizando.

## ResNet

Abaixo apresenta-se a Tabela 5 com a matriz de confusão da ResNet101.

Tabela 5: Matriz de confusão e métricas da ResNet101

|  |  |  |  |
| --- | --- | --- | --- |
| Aviões | | Carros | |
| **(TP)** 453 | **(FP)** 47 | **(TP)** 474 | **(FP)** 38 |
| **(FN)** 66 | **(TN)** 86 | **(FN)** 16 | **(TN)** 124 |
| Acurácia | 83% | Acurácia | 92% |
| Precisão | 65% | Precisão | 77% |
| Sensibilidade | 57% | Sensibilidade | 89% |
| “f1-score” | 60% | “f1-score” | 82% |
| Drones | | Navios | |
| **(TP)** 390 | **(FP)** 60 | **(TP)** 483 | **(FP)** 11 |
| **(FN)** 15 | **(TN)** 187 | **(FN)** 59 | **(TN)** 99 |
| Acurácia | 88% | Acurácia | 89% |
| Precisão | 76% | Precisão | 90% |
| Sensibilidade | 93% | Sensibilidade | 63% |
| “f1-score” | 83% | “f1-score” | 74% |

Fonte: Resultados originais da pesquisa

As Figuras 19 e 20 apresentam os gráficos de perda e acurácia.

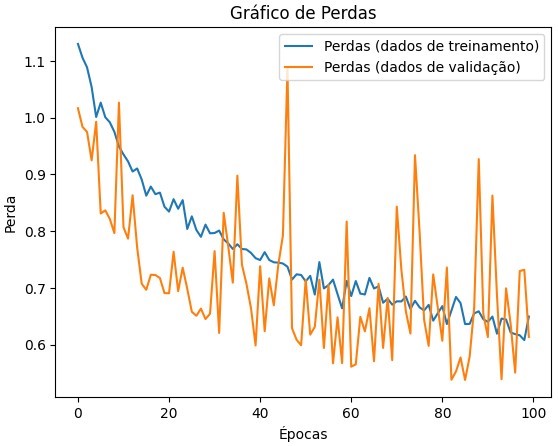


Figura 19: Gráfico de perdas da ResNet101

Fonte: Resultados originais da pesquisa

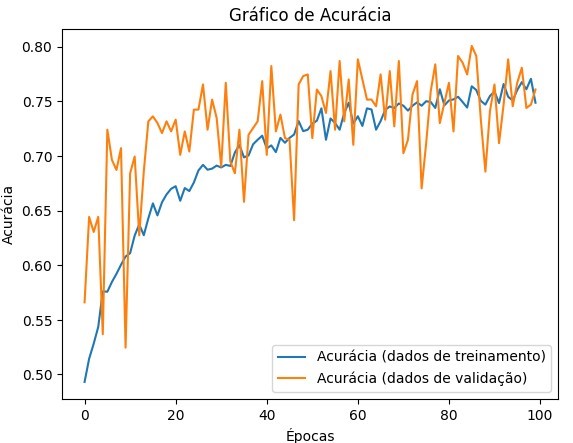


Figura 20: Gráfico de acurácia da ResNet101

Fonte: Resultados originais da pesquisa

Analisando os gráficos acima e os resultados obtidos pelas métricas oriundas da matriz de confusão, supõe-se que a principal causa para estes resultados seja a quantidade de épocas. Os resultados não apresentaram convergência significativa, devendo-se, portanto, aumentar o número de épocas.

## Considerações Finais

Das três redes neurais convolucionais implementadas neste trabalho, a VGG16 e Xception apresentaram bons resultados nos dados de validação, enquanto que a ResNet101 não apresentou um resultado satisfatório.

A VGG16 e a Xception apresentaram o fenômeno de “overfitting”. A aparição deste fenômeno indica que houve um gasto computacional desnecessário, pois a partir de determinada época estas CNNs aprendiam padrões bem específicos nos dados de treinamento, mas não conseguiam generalizar este aprendizado para dados de teste.

Duas soluções poderiam ser utilizadas para este caso. A primeira, levando em conta os elevados níveis de acurácia obtidos pela VGG16 e Xception, seria simplemente reduzir o número de épocas de treinamento da rede para evitar um gasto computacional desnecessario (sem a realização de alterações na arquitetura da rede). A segunda solução seria a alteração de alguns parâmetros da rede tais como:

* Taxa de aprendizado (“learning rate”);
* Mudança do otimizador utilizado tais como: Adam, RMSprop, Adagrap ou Adadelta;
* Mudança nas últimas camadas alteradas no processo de “fine-tuning” (incluindo ou retirando camadas); e
* Alteração da quantidade de camadas pré-treinadas congeladas na seguda fase do treinamento.

Quanto a ResNet101, quando se analisam os gráficos de perdas e de acurácias, observa-se uma grande variação nos dados de validação quando comparados com os dados de teste. Uma possível solução para este caso, como já dito, seria o aumento do número de épocas. Um ponto importante a se analisar, porém, é a viabilidade da utilização da ResNet101 para essa aplicação já que tanto a VGG16 quando a Xception apresentaram bons resultados e um aumento do número de épocas significaria um aumento do custo computacional.

## Agradecimento

Agradeço a Deus pela força concedida para a realização deste trabalho. Que as habilidades aprendidas aqui sejam úteis para Seu Reino. Agradeço também à Fernanda, minha esposa, pelo apoio incondicional a todo instante.

## Referências

Ricardo Augusto Paletta Guedes. 2021.Glaucoma, saúde coletiva e impacto social. Revista Brasileira de Oftalmologia.

Disponível em: <https://www.scielo.br/j/rbof/a/kHLnFkWBc6jDWz3sQbvyhtR/>

R. Bourne et al.2020. Causes of blindness and vision impairment in 2020 and trends over 30 years, and prevalence of avoidable blindness in relation to VISION 2020: the Right to Sight: an analysis for the Global Burden of Disease Study.The Lancet Global Health.Volume 9.

Disponível em: <https://www.thelancet.com/JOURNALS/LANGLO/ARTICLE/PIIS2214-109X(20)30489-7/FULLTEXT>

V. O. Domingues et al.2016. Catarata senil: uma revisão de literatura. Revista de Medicina e Saúde de Brasília. Volume 5.

Disponível em: <https://portalrevistas.ucb.br/index.php/rmsbr/article/view/6756>

-----------------------------------------------------------------------------------------------------------------

Evanson, NIck. 2020. Explainer: What Are Tensor Cores. TechSpot, Estados Unidos da América. Disponível em: https://www.techspot.com/article/2049-what-are-tensor-cores/. Acesso em: 18, abril 2023.

Géron, Aurelien. 2021. Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow. 2ed. O’Reilly, Sebastopol, CA, USA.

Joshi, Prateek. 2015. Opencv with python by examples. 1ed. Packt Publishing. Birmingham, Reino Unido.

Khan, Rizwan. 2019. Ilustração da Arquitetura da VGG16. Disponível em: https://www.researchgate.net/figure/An-illustration-of-VGG16-architecture38\_fig12\_329413866. Acesso em: 18, abril 2023.

Lamb, Thomas. 2003. Ship Design and Construction. 1ed. Andrew MacBride. Estados Unidos da América

Milano, D. Honorato, L.B. 2010. Visão Computacional. Universidade Estadual de Campinas, Faculdade de Tecnologia. Campinas, São Paulo, Brasil. Disponível em: https://d1wqtxts1xzle7.cloudfront.net/35825905/2010\_IA\_FT\_UNICAMP\_visaoComputaciona l-libre.pdf?1417700841=&response-content-disposition=inline%3B+filename

%3DVISAO\_COMPUTACIONAL\_Palavras\_Chaves.pdf&Expires=1692832812&Signature=M 22S1zhfgmgGxVmecB~1XP1ERaYij3HpYTGSKhYcbz4SAPz7dm4nBu~ObF4dujMq2X1woS d~ofRi~asohiPM927JOTfCsj~qqiyzaki5~gTgrT8lrdgKTDmciKv2kl4ZOMfBDhCmor9jOoaMS7 LHlHSIJYTRaaqnVCIYdiFQ74ZchXLPyPG79i1FIUZANRQS459oi1leavDWIkAeHFeS98padK0YMYbjKEvUjDcDE

QVRWNmnk6a2dCEG7ZxW0XRLTKxdsxv1fd6MP445yIop0RtsTAmzAzbvK3T2C8n2gE2WJ G6N5cesOY-yIiyjVqCoZmSN9x0ysH5NPM0Qjj3Bg\_\_&Key-PairId=APKAJLOHF5GGSLRBV4ZA. Acesso em 23, agosto 2023.

Ozel, Mehdi. 2019. Drone Dataset (UAV). Disponível em: https://www.kaggle.com/datasets/dasmehdixtr/drone-dataset-uav. Acesso em: 18, abril 2023.

Rampurawala, Abduttayyeb. 2021. multiclass-image-dataset-airplane-car-ship. Disponível em: https://www.kaggle.com/datasets/abtabm/multiclassimagedatasetairplanecar. Acesso em: 18, abril 2023.

Redmon, Joseph; Divvala, Santosh; Girshick, Ross; Farhadi, Ali. 2015. You Only Look Once: Unified, Real-Time Object Detection. Estados Unidos da América. Disponível em https://arxiv.org/pdf/1506.02640.pdf. Acesso em: 18, abril 2023.

Rosebrock, Adrian. 2021. Convolutional Neural Networks (CNNs) and Layer Types. PyImageSearch, Estados Unidos da América. Disponível em https://pyimagesearch.com/2021/05/14/convolutional-neural-networks-cnns-and-layer-types/. Acesso em: 18, abril 2023.

Rosebrock, Adrian. 2017. Deep Learning for Computer Vision with Python. 1ed. PyImageSearch. Estados Unidos da América.

Raj, Bharath. 2018. Data Augmentation: How to Use Deep Learning When You Have Limited Data. KDNuggets, Estados Unidos da América. Disponível em https://www.kdnuggets.com/2018/05/data-augmentation-deep-learning-limited-data.html. Acesso em: 18, abril 2023.

## Apêndice

Os códigos implementados nas três CNNs utilizadas foram os mesmos, com apenas algumas pequenas especificidades cada um. Desta forma, apenas um será apresentado para eventual análise: <https://github.com/GMSantos4/CV/blob/main/main_vgg16.ipynb>.